Statistical Methods II

Derek L. Sonderegger

2018-12-06

Contents

Preface

These notes are intended to be used in the second semester of a two-semester sequence of Statistical Methodology. We assume that students have seen t-tests, Simple Regression, and ANOVA. The second semester emphasizes the uniform matrix notation $y = X\beta + \epsilon$ and the interpretation of the coefficients. We cover model diagnostics, transformation, model selection, interactions of continuous and categorical predictors as well as introduce random effects in the experimental design context. Finally we introduce logistic regression.

Chapter 1

Matrix Theory

Almost all of the calculations done in classical statistics require formulas with large number of subscripts and many different sums. In this chapter we will develop the mathematical machinery to write these formulas in a simple compact formula using *matrices*.

1.1 Types of Matrices

We will first introduce the idea behind a matrix and give several special types of matrices that we will encounter.

1.1.1 Scalars

To begin, we first define a *scalar*. A scalar is just a single number, either real or complex. The key is that a scalar is just a single number. For example, 6 is a scalar, as is -3. By convention, variable names for scalars will be lower case and not in bold typeface.

Examples could be $a = 5$, $b = \sqrt{3}$, or $\sigma = 2$.

1.1.2 Vectors

A vector is collection of scalars, arranged as a row or column. Our convention will be that a vector will be a lower cased letter but written in a bold type. In other branches of mathematics is common to put a bar over the variable name to denote that it is a vector, but in statistics, we have already used a bar to denote a mean.

Examples of column vectors could be

$$\boldsymbol{a} = \begin{bmatrix} 2 \\ -3 \\ 4 \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} 2 \\ 8 \\ 3 \\ 4 \\ 1 \end{bmatrix}$$

and examples of row vectors are

$$\boldsymbol{c} = \begin{bmatrix} 8 & 10 & 43 & -22 \end{bmatrix}$$

$$\boldsymbol{d} = \begin{bmatrix} -1 & 5 & 2 \end{bmatrix}$$

To denote a specific entry in the vector, we will use a subscript. For example, the second element of $\boldsymbol{d}$ is $d_2 = 5$. Notice, that we do not bold this symbol because the second element of the vector is the scalar value 5.

1.1.3 Matrix

Just as a vector is a collection of scalars, a matrix can be viewed as a collection of vectors (all of the same length). We will denote matrices with bold capitalized letters. In general, I try to use letters at the end of the alphabet for matrices. Likewise, I try to use symmetric letters to denote symmetric matrices.

For example, the following is a matrix with two rows and three columns

$$\boldsymbol{W} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

and there is no requirement that the number of rows be equal, less than, or greater than the number of columns. In denoting the size of the matrix, we first refer to the number of rows and then the number of columns. Thus $\boldsymbol{W}$ is a 2×3 matrix and it sometimes is helpful to remind ourselves of this by writing $\boldsymbol{W}_{2\times 3}$.

To pick out a particular element of a matrix, I will again use a subscripting notation, always with the row number first and then column. Notice the notational shift to lowercase, non-bold font.

$$w_{1,2} = 2 \qquad \text{and} \qquad w_{2,3} = 6$$

There are times I will wish to refer to a particular row or column of a matrix and we will use the following notation

$$\boldsymbol{w}_{1,\cdot} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

is the first row of the matrix $\boldsymbol{W}$. The second column of matrix $\boldsymbol{W}$ is

$$\boldsymbol{w}_{\cdot,2} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

1.1.4 Square Matrices

A square matrix is a matrix with the same number of rows as columns. The following are square

$$\boldsymbol{Z} = \begin{bmatrix} 3 & 6 \\ 8 & 10 \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{bmatrix}$$

1.1.5 Symmetric Matrices

In statistics we are often interested in square matrices where the i, j element is the same as the j, i element. For example, $x_{1,2} = x_{2,1}$ in the above matrix $\boldsymbol{X}$.

Consider a matrix $\boldsymbol{D}$ that contains the distance from four towns to each of the other four towns. Let $d_{i,j}$ be the distance from town i to town j. It only makes sense that the distance doesn't matter which direction you are traveling, and we should therefore require that $d_{i,j} = d_{j,i}$.

In this example, it is the values $d_{i,i}$ represent the distance from a town to itself, which should be zero. It turns out that we are often interested in the terms $d_{i,i}$ and I will refer to those terms as the *main diagonal* of matrix $\boldsymbol{D}$.

Symmetric matrices play a large role in statistics because matrices that represent the covariances between random variables must be symmetric because $Cov(Y,Z) = Cov(Z,Y)$.

1.1.6 Diagonal Matrices

A square matrix that has zero entries in every location except the main diagonal is called a diagonal matrix. Here are two examples:

$$\boldsymbol{Q} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 6 \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

Sometimes to make matrix more clear, I will replace the 0 with a dot to emphasize the non-zero components.

$$\boldsymbol{R} = \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & 2 & \cdot & \cdot \\ \cdot & \cdot & 2 & \cdot \\ \cdot & \cdot & \cdot & 3 \end{bmatrix}$$

1.1.7 Identity Matrices

A diagonal matrix with main diagonal values exactly 1 is called the identity matrix. The 3×3 identity matrix is denoted I_3.

$$\boldsymbol{I}_3 = \begin{bmatrix} 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \\ \cdot & \cdot & 1 \end{bmatrix}$$

1.2 Operations on Matrices

1.2.1 Transpose

The simplest operation on a square matrix matrix is called *transpose*. It is defined as $\boldsymbol{M} = \boldsymbol{W}^T$ if and only if $m_{i,j} = w_{j,i}$.

$$\boldsymbol{Z} = \begin{bmatrix} 1 & 6 \\ 8 & 3 \end{bmatrix} \qquad \boldsymbol{Z}^T = \begin{bmatrix} 1 & 8 \\ 6 & 3 \end{bmatrix}$$

$$\boldsymbol{M} = \begin{bmatrix} 3 & 1 & 2 \\ 9 & 4 & 5 \\ 8 & 7 & 6 \end{bmatrix} \qquad \boldsymbol{M}^T = \begin{bmatrix} 3 & 9 & 8 \\ 1 & 4 & 7 \\ 2 & 5 & 6 \end{bmatrix}$$

We can think of this as swapping all elements about the main diagonal. Alternatively we could think about the transpose as making the first row become the first column, the second row become the second column, etc. In this fashion we could define the transpose of a non-square matrix.

$$\boldsymbol{W} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\boldsymbol{W}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

1.2.2 Addition and Subtraction

Addition and subtraction are performed *element-wise*. This means that two matrices or vectors can only be added or subtracted if their dimensions match.

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \\ 10 \\ 12 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 8 \\ 2 & 4 \\ 11 & 15 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & -6 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ -1 & 0 \\ 6 & 21 \end{bmatrix}$$

1.2.3 Multiplication

Multiplication is the operation that is vastly different for matrices and vectors than it is for scalars. There is a great deal of mathematical theory that suggests a useful way to define multiplication. What is presented below is referred to as the *dot-product* of vectors in calculus, and is referred to as the standard *inner-product* in linear algebra.

1.2.4 Vector Multiplication

We first define multiplication for a row and column vector. For this multiplication to be defined, both vectors must be the same length. The product is the sum of the element-wise multiplications.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} = (1 \cdot 5) + (2 \cdot 6) + (3 \cdot 7) + (4 \cdot 8) = 5 + 12 + 21 + 32 = 70$$

1.2.5 Matrix Multiplication

Matrix multiplication is just a sequence of vector multiplications. If $\boldsymbol{X}$ is a $m \times n$ matrix and $\boldsymbol{W}$ is $n \times p$ matrix then $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}$ is a $m \times p$ matrix where $z_{i,j} = \boldsymbol{x}_{i,\cdot}\boldsymbol{w}_{\cdot,j}$ where $\boldsymbol{x}_{i,\cdot}$ is the ith column of $\boldsymbol{X}$ and $\boldsymbol{w}_{\cdot,j}$ is the jth column of $\boldsymbol{W}$. For example, let

$$\boldsymbol{X} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} \qquad \boldsymbol{W} = \begin{bmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \\ 19 & 20 \end{bmatrix}$$

so $\boldsymbol{X}$ is 3×4 (which we remind ourselves by adding a 3×4 subscript to $\boldsymbol{X}$ as $\boldsymbol{X}_{3\times4}$) and $\boldsymbol{W}$ is $\boldsymbol{W}_{4\times2}$. Because the *inner* dimensions match for this multiplication, then $\boldsymbol{Z}_{3\times2} = \boldsymbol{X}_{3\times4}\boldsymbol{W}_{4\times2}$ is defined where

$$\begin{aligned} z_{1,1} &= \boldsymbol{x}_{1,\cdot}\boldsymbol{w}_{\cdot,1} \\ &= (1 \cdot 13) + (2 \cdot 15) + (3 \cdot 17) + (4 \cdot 19) = 170 \end{aligned}$$

and similarly

$$\begin{aligned} z_{2,1} &= \boldsymbol{x}_{2,\cdot}\boldsymbol{w}_{\cdot,1} \\ &= (5 \cdot 13) + (6 \cdot 15) + (7 \cdot 17) + (8 \cdot 19) \\ &= 426 \end{aligned}$$

so that

$$\boldsymbol{Z} = \begin{bmatrix} 170 & 180 \\ 426 & 452 \\ 682 & 724 \end{bmatrix}$$

For another example, we note that

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 2 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1+4+3 & 2+4+6 \\ 2+6+4 & 4+6+8 \end{bmatrix} = \begin{bmatrix} 8 & 12 \\ 12 & 18 \end{bmatrix}$$

Notice that this definition of multiplication means that the order matters. Above, we calculated $\boldsymbol{X}_{3\times 4}\boldsymbol{W}_{4\times 2}$ but we cannot reverse the order because the inner dimensions do not match up.

1.2.6 Scalar times a Matrix

Strictly speaking, we are not allowed to multiply a matrix by a scalar because the dimensions do not match. However, it is often notationally convenient. So we define $a\boldsymbol{X}$ to be the *element-wise* multiplication of each element of $\boldsymbol{X}$ by the scalar a. Because this is just a notational convenience, the mathematical theory about inner-products does not apply to this operation.

$$5 \begin{bmatrix} 4 & 5 \\ 7 & 6 \\ 9 & 10 \end{bmatrix} = \begin{bmatrix} 20 & 25 \\ 35 & 30 \\ 45 & 50 \end{bmatrix}$$

Because of this definition, it is clear that $a\boldsymbol{X} = \boldsymbol{X}a$ and the order does not matter. Thus when mixing scalar multiplication with matrices, it is acceptable to reorder scalars, but not matrices.

1.2.7 Determinant

The determinant is defined only for square matrices and can be thought of as the matrix equivalent of the absolute value or magnitude (i.e. $|-6| = 6$). The determinant gives a measure of the multi-dimensional size of a matrix (say the matrix $\boldsymbol{A}$) and as such is denoted $\det(\boldsymbol{A})$ or $|\boldsymbol{A}|$. Generally this is a very tedious thing to calculate by hand and for completeness sake, we will give a definition and small examples.

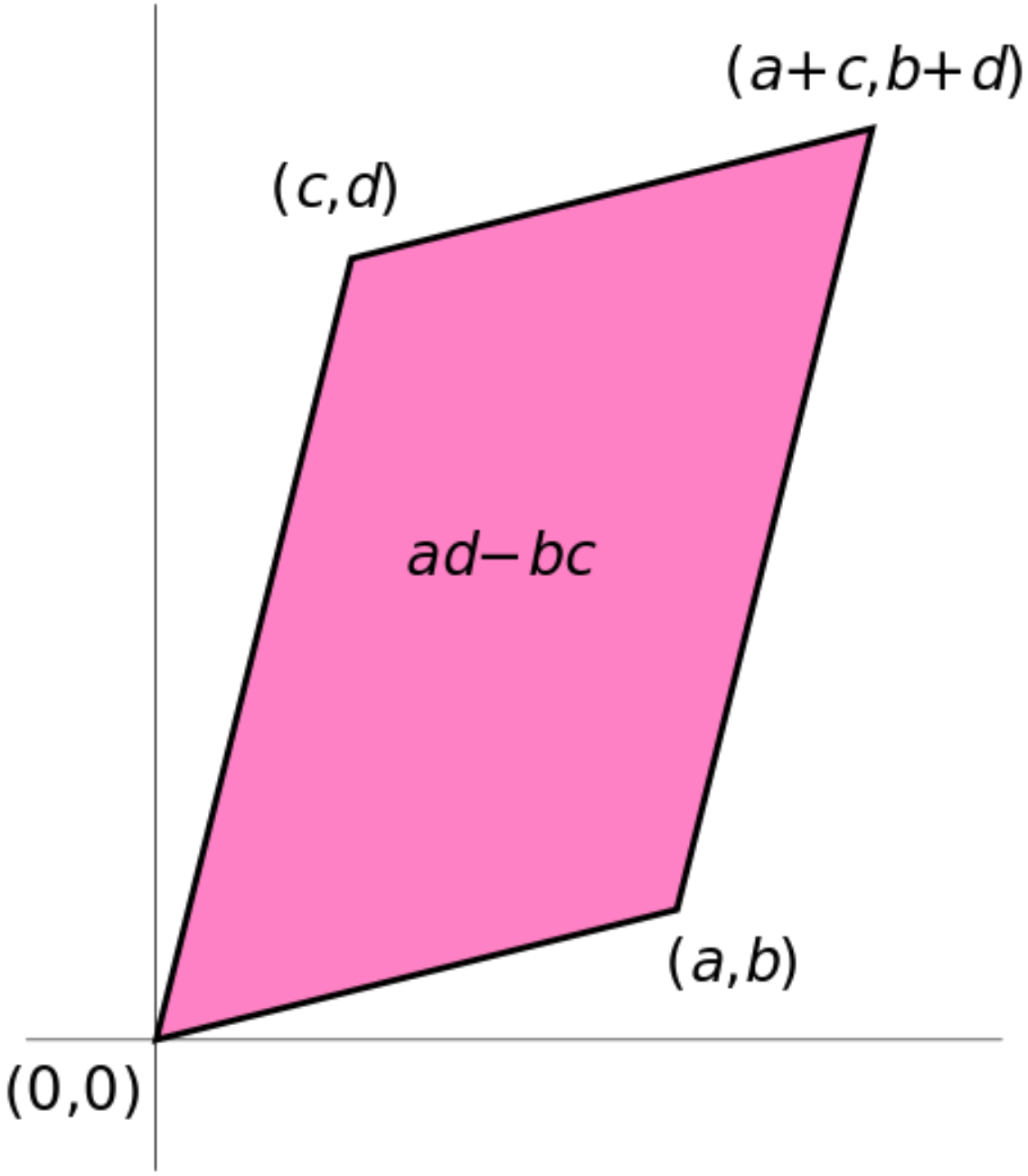
For a 2×2 matrix

$$\begin{vmatrix} a & c \\ b & d \end{vmatrix} = ad - cb$$

So a simple example of a determinant is

$$\begin{vmatrix} 5 & 2 \\ 3 & 10 \end{vmatrix} = 50 - 6 = 44$$

The determinant can be thought of as the area of the parallelogram created by the row or column vectors of the matrix.



1.2.8 Inverse

In regular algebra, we are often interested in solving equations such as

$$5x = 15$$

for x. To do so, we multiply each side of the equation by the inverse of 5, which is $1/5$.

$$\begin{aligned} 5x &= 15 \\ \frac{1}{5} \cdot 5 \cdot x &= \frac{1}{5} \cdot 15 \\ 1 \cdot x &= 3 \\ x &= 3 \end{aligned}$$

For scalars, we know that the inverse of scalar a is the value that when multiplied by a is 1. That is we see to find a^{-1} such that $aa^{-1} = 1$.

In the matrix case, I am interested in finding $\boldsymbol{A}^{-1}$ such that $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$ and $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$. For both of these multiplications to be defined, $\boldsymbol{A}$ must be a square matrix and so the inverse is only defined for square matrices.

For a 2×2 matrix

$$\boldsymbol{W} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

the inverse is given by:

$$\boldsymbol{W}^{-1} = \frac{1}{\det \boldsymbol{W}} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

For example, suppose

$$\boldsymbol{W} = \begin{bmatrix} 1 & 2 \\ 5 & 3 \end{bmatrix}$$

then $\det W = 3 - 10 = -7$ and

$$\boldsymbol{W}^{-1} = \frac{1}{-7} \begin{bmatrix} 3 & -2 \\ -5 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{3}{7} & \frac{2}{7} \\ \frac{5}{7} & -\frac{1}{7} \end{bmatrix}$$

and thus

$$\boldsymbol{W}\boldsymbol{W}^{-1} = \begin{bmatrix} 1 & 2 \\ 5 & 3 \end{bmatrix} \begin{bmatrix} -\frac{3}{7} & \frac{2}{7} \\ \frac{5}{7} & -\frac{1}{7} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{3}{7} + \frac{10}{7} & \frac{2}{7} - \frac{2}{7} \\ -\frac{15}{7} + \frac{15}{7} & \frac{10}{7} - \frac{3}{7} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boldsymbol{I}_2$$

Not every square matrix has an inverse. If the determinant of the matrix (which we think of as some measure of the magnitude or *size* of the matrix) is zero, then the formula would require us to divide by zero. Just as we cannot find the inverse of zero (i.e. solve $0x = 1$ for x), a matrix with zero determinate is said to have no inverse.

1.3 Exercises

Consider the following matrices:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 6 & 4 & 3 \\ 8 & 7 & 6 \end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad \mathbf{d} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \qquad \mathbf{E} = \begin{bmatrix} 1 & 2 \\ 2 & 6 \end{bmatrix}$$

1. Find $\mathbf{Bc}$
2. Find $\mathbf{AB}^T$
3. Find $\mathbf{c}^T\mathbf{d}$
4. Find $\mathbf{cd}^T$
5. Confirm that
$$\mathbf{E}^{-1} = \begin{bmatrix} 3 & -1 \\ -1 & 1/2 \end{bmatrix}$$
is the inverse of $\mathbf{E}$ by calculating $\mathbf{E}\mathbf{E}^{-1} = \mathbf{I}$.

Chapter 2

Parameter Estimation

```
library(tidyverse)  # dplyr, tidyr, ggplot2
```

We have previously looked at ANOVA and regression models and, in many ways, they felt very similar. In this chapter we will introduce the theory that allows us to understand both models as a particular flavor of a larger class of models known as *linear models*.

First we clarify what a linear model is. A linear model is a model where the data (which we will denote using roman letters as $\boldsymbol{x}$ and $\boldsymbol{y}$) and parameters of interest (which we denote using Greek letters such as $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) interact only via addition and multiplication. The following are linear models:

Model	Formula
ANOVA	$y_{ij} = \mu + \tau_i + \epsilon_{ij}$
Simple Regression	$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$
Quadratic Term	$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$
General Regression	$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p} + \epsilon_i$

Notice in the Quadratic model, the square is not a parameter and we can consider x_i^2 as just another column of data. This leads to the second example of multiple regression where we just add more slopes for other covariates where the pth covariate is denoted $\boldsymbol{x}_{\cdot,p}$ and might be some transformation (such as x^2 or $\log x$) of another column of data. The critical point is that the transformation to the data $\boldsymbol{x}$ does not depend on a parameter. Thus the following is *not* a linear model

$$y_i = \beta_0 + \beta_1 x_i^{\alpha} + \epsilon_i$$

2.1 Simple Regression

We would like to represent all linear models in a similar compact matrix representation. This will allow us to make the transition between simple and multiple regression (and ANCOVA) painlessly.

To begin, we think about how to write the simple regression model using matrices and vectors that correspond the the data and the parameters. Notice we have

$$
\begin{aligned}
y_1 &= \beta_0 + \beta_1 x_1 + \epsilon_1 \\
y_2 &= \beta_0 + \beta_1 x_2 + \epsilon_2 \\
y_3 &= \beta_0 + \beta_1 x_3 + \epsilon_3 \\
&\vdots \\
y_{n-1} &= \beta_0 + \beta_1 x_{n-1} + \epsilon_{n-1} \\
y_n &= \beta_0 + \beta_1 x_n + \epsilon_n
\end{aligned}
$$

where, as usual, $\epsilon_i \overset{iid}{\sim} N\left(0, \sigma^2\right)$. These equations can be written using matrices as RR

$$
\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}}_{\boldsymbol{y}} = \underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_{n-1} \\ 1 & x_n \end{bmatrix}}_{\boldsymbol{X}} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}}_{\boldsymbol{\beta}} + \underbrace{\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_{n-1} \\ \epsilon_n \end{bmatrix}}_{\boldsymbol{\epsilon}}
$$

and we compactly write the model as

$$
\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}
$$

where $\boldsymbol{X}$ is referred to as the *design matrix* and $\boldsymbol{\beta}$ is the vector of *location parameters* we are interested in estimating.

2.1.1 Estimation of Location Paramters

Our next goal is to find the best estimate of $\boldsymbol{\beta}$ given the data. To justify the formula, consider the case where there is no error terms (i.e. $\epsilon_i = 0$ for all i). Thus we have

$$
\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}
$$

and our goal is to solve for $\boldsymbol{\beta}$. To do this, we must use a matrix inverse, but since inverses only exist for square matrices, we pre-multiple by $\boldsymbol{X}^T$ (notice that $\boldsymbol{X}^T\boldsymbol{X}$ is a symmetric 2×2 matrix).

$$
\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}
$$

and then pre-multiply by $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}$.

$$
\begin{aligned}
\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} &= \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\
\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} &= \boldsymbol{\beta}
\end{aligned}
$$

This exercise suggests that $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$ is a good place to start when looking for the maximum-likelihood estimator for $\boldsymbol{\beta}$. It turns out that this quantity is in fact the maximum-likelihood estimator (and equivalently minimizes the sum-of-squared error). Therefore we will use it as our estimate of $\boldsymbol{\beta}$.

$$
\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}
$$

2.1.2 Estimation of Variance Parameter

Recall our model is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

where $\epsilon_i \overset{iid}{\sim} N\left(0, \sigma^2\right)$.

Using our estimates $\hat{\boldsymbol{\beta}}$ we can obtain predicted values for the regression line at any x-value. In particular we can find the predicted value for each x_i value in our dataset.

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

Using matrix notation, I would write $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$.

As usual we will find estimates of the noise terms (which we will call residuals or errors) via

$$\begin{aligned}\hat{\epsilon}_i &= y_i - \hat{y}_i \\ &= y_i - \left(\hat{\beta}_0 + \hat{\beta}_1 x_i\right)\end{aligned}$$

Writing $\hat{\boldsymbol{y}}$ in matrix terms we have

$$\begin{aligned}\hat{\boldsymbol{y}} &= \boldsymbol{X}\hat{\boldsymbol{\beta}} \\ &= \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \\ &= \boldsymbol{H}\boldsymbol{y}\end{aligned}$$

where $\boldsymbol{H} = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T$ is often called the *hat-matrix* because it takes y to $\hat{y}$ and has many interesting theoretical properties.[1]

We can now estimate the error terms via

$$\begin{aligned}\hat{\boldsymbol{\epsilon}} &= \boldsymbol{y} - \hat{\boldsymbol{y}} \\ &= \boldsymbol{y} - \boldsymbol{H}\boldsymbol{y} \\ &= \left(\boldsymbol{I}_n - \boldsymbol{H}\right)\boldsymbol{y}\end{aligned}$$

As usual we estimate σ^2 using the mean-squared error

$$\hat{\sigma}^2 = \frac{1}{n-2}\sum_{i=1}^{n}\hat{\epsilon}_i^2$$

$$= \frac{1}{n-2}\,\hat{\boldsymbol{\epsilon}}^T\hat{\boldsymbol{\epsilon}}$$

In the general linear model case where $\boldsymbol{\beta}$ has p elements (and thus we have $n-p$ degrees of freedom), the formula is

$$\hat{\sigma}^2 = \frac{1}{n-p}\,\hat{\boldsymbol{\epsilon}}^T\hat{\boldsymbol{\epsilon}}$$

[1] Mathematically, $\boldsymbol{H}$ is the projection matrix that takes a vector in n-dimensional space and projects it onto a p-dimension subspace spanned by the vectors in $\boldsymbol{X}$. Projection matrices have many useful properties and much of the theory of linear models utilizes $\boldsymbol{H}$.

2.1.3 Expectation and variance of a random vector

Just as we needed to derive the expected value and variance of $\bar{x}$ in the previous semester, we must now do the same for $\hat{\boldsymbol{\beta}}$. But to do this, we need some properties of expectations and variances.

In the following, let $\boldsymbol{A}_{n\times p}$ and $\boldsymbol{b}_{n\times 1}$ be constants and $\boldsymbol{\epsilon}_{n\times 1}$ be a random vector.

Expectations are very similar to the scalar case where

$$E\left[\boldsymbol{\epsilon}\right] = \begin{bmatrix} E\left[\epsilon_1\right] \\ E\left[\epsilon_2\right] \\ \vdots \\ E\left[\epsilon_n\right] \end{bmatrix}$$

and any constants are pulled through the expectation

$$E\left[\boldsymbol{A}^T\boldsymbol{\epsilon} + \boldsymbol{b}\right] = \boldsymbol{A}^T E\left[\boldsymbol{\epsilon}\right] + \boldsymbol{b}$$

Variances are a little different. The variance of the vector $\boldsymbol{\epsilon}$ is

$$Var\left(\boldsymbol{\epsilon}\right) = \begin{bmatrix} Var\left(\epsilon_1\right) & Cov\left(\epsilon_1,\epsilon_2\right) & \dots & Cov\left(\epsilon_1,\epsilon_n\right) \\ Cov\left(\epsilon_2,\epsilon_1\right) & Var\left(\epsilon_2\right) & \dots & Cov\left(\epsilon_2,\epsilon_n\right) \\ \vdots & \vdots & \ddots & \vdots \\ Cov\left(\epsilon_n,\epsilon_1\right) & Cov\left(\epsilon_n,\epsilon_2\right) & \dots & Var\left(\epsilon_1\right) \end{bmatrix}$$

and additive constants are ignored, but multiplicative constants are pulled out as follows:

$$Var\left(\boldsymbol{A}^T\boldsymbol{\epsilon} + \boldsymbol{b}\right) = Var\left(\boldsymbol{A}^T\boldsymbol{\epsilon}\right) = \boldsymbol{A}^T Var\left(\boldsymbol{\epsilon}\right)\boldsymbol{A}$$

2.1.4 Variance of Location Parameters

We next derive the sampling variance of our estimator $\hat{\boldsymbol{\beta}}$ by first noting that $\boldsymbol{X}$ and $\boldsymbol{\beta}$ are constants and therefore

$$\begin{aligned} Var\left(\boldsymbol{y}\right) &= Var\left(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}\right) \\ &= Var\left(\boldsymbol{\epsilon}\right) \\ &= \sigma^2\boldsymbol{I}_n \end{aligned}$$

because the error terms are independent and therefore $Cov\left(\epsilon_i,\epsilon_j\right) = 0$ when $i \neq j$ and $Var\left(\epsilon_i\right) = \sigma^2$. Recalling that constants come out of the variance operator as the constant *squared*.

$$\begin{aligned} Var\left(\hat{\boldsymbol{\beta}}\right) &= Var\left(\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}\right) \\ &= \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T Var\left(\boldsymbol{y}\right)\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \\ &= \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\sigma^2\boldsymbol{I}_n\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \\ &= \sigma^2\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \\ &= \sigma^2\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \end{aligned}$$

Using this, the standard error (i.e. the estimated standard deviation) of $\hat{\beta}_j$ (for any j in $1, \ldots, p$) is

$$StdErr\left(\hat{\beta}_j\right) = \sqrt{\hat{\sigma}^2 \left[\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\right]_{jj}}$$

2.1.5 Confidence intervals and hypothesis tests

We can now state the general method of creating confidence intervals and perform hypothesis tests for any element of $\boldsymbol{\beta}$.

The confidence interval formula is (as usual)

$$\hat{\beta}_j \pm t_{n-p}^{1-\alpha/2}\, StdErr\left(\hat{\beta}_j\right)$$

and a test statistic for testing $H_0 : \beta_j = 0$ versus $H_a : \beta_j \neq 0$ is

$$t_{n-p} = \frac{\hat{\beta}_j - 0}{StdErr\left(\hat{\beta}_j\right)}$$

2.1.6 Summary of pertinent results

The statistic $\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$ is the unbiased maximum-likelihood estimator of $\boldsymbol{\beta}$.

The Central Limit Theorem applies to each element of $\boldsymbol{\beta}$. That is, as $n \to \infty$, the distribution of $\hat{\beta}_j \to N\left(\beta_j, \left[\sigma^2\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\right]_{jj}\right)$.

The error terms can be calculated via

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$$
$$\hat{\boldsymbol{\epsilon}} = \boldsymbol{y} - \hat{\boldsymbol{y}}$$

The estimate of σ^2 is

$$\hat{\sigma}^2 = \frac{1}{n-p}\,\hat{\boldsymbol{\epsilon}}^T\hat{\boldsymbol{\epsilon}}$$

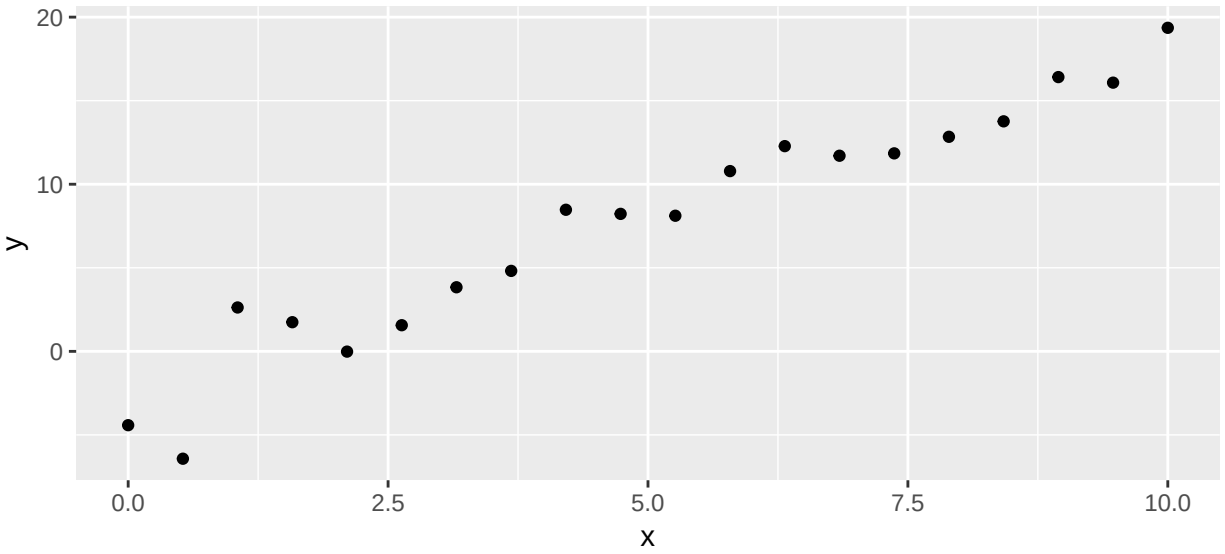
The standard error (i.e. the estimated standard deviation) of $\hat{\beta}_j$ (for any j in $1, \ldots, p$) is

$$StdErr\left(\hat{\beta}_j\right) = \sqrt{\hat{\sigma}^2 \left[\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\right]_{jj}}$$

2.1.7 An example in R

Here we will work an example in R and see the calculations. Consider the following data:

```
n <- 20
x <- seq(0,10, length=n)
y <- -3 + 2*x + rnorm(n, sd=2)
my.data <- data.frame(x=x, y=y)
ggplot(my.data) + geom_point(aes(x=x,y=y))
```



First we must create the design matrix $\boldsymbol{X}$. Recall

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_{n-1} \\ 1 & x_n \end{bmatrix}$$

and can be created in R via the following:

```
X <- cbind( rep(1,n), x)
```

Given $\boldsymbol{X}$ and $\boldsymbol{y}$ we can calculate

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

in R using the following code:

```
XtXinv <- solve( t(X) %*% X )
beta.hat <- XtXinv %*% t(X) %*% y
beta.hat
```

```
##        [,1]
##   -3.252611
## x  2.186690
```

Our next step is to calculate the predicted values $\hat{\boldsymbol{y}}$ and the residuals $\hat{\boldsymbol{\epsilon}}$

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$$
$$\hat{\boldsymbol{\epsilon}} = \boldsymbol{y} - \hat{\boldsymbol{y}}$$

```
y.hat <- X %*% beta.hat
residuals <- y - y.hat
```

Now that we have the residuals, we can calculate $\hat{\sigma}^2$ and the standard errors of $\hat{\beta}_j$

$$\hat{\sigma}^2 = \frac{1}{n-p}\hat{\boldsymbol{\epsilon}}^T\hat{\boldsymbol{\epsilon}}$$

$$StdErr\left(\hat{\beta}_j\right) = \sqrt{\hat{\sigma}^2 \left[\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\right]_{jj}}$$

```
sigma2.hat <- ( t(residuals) %*% residuals) / (n-2)
sigma.hat <- sqrt( sigma2.hat )
std.errs <- sqrt( sigma2.hat * diag(XtXinv) )
```

```
## Warning in sigma2.hat * diag(XtXinv): Recycling array of length 1 in array-vector arithmetic is deprecated
##   Use c() or as.vector() instead.
```

We now print out the important values and compare them to the summary output given by the `lm()` function in R.

```
beta.hat
```

```
##        [,1]
##   -3.252611
## x  2.186690
```

```
sigma.hat
```

```
##          [,1]
## [1,] 1.784656
```

```
std.errs
```

```
## [1] 0.7690898 0.1314914
```

```
model <- lm(y~x)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3226 -1.1683  0.0038  1.1865  3.5764
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.2526     0.7691  -4.229 0.000504 ***
## x             2.1867     0.1315  16.630 2.27e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.785 on 18 degrees of freedom
## Multiple R-squared:  0.9389, Adjusted R-squared:  0.9355
## F-statistic: 276.6 on 1 and 18 DF,  p-value: 2.267e-12
```

We calculate 95% confidence intervals via:

```
lwr <- beta.hat - qt(.975, n-2) * std.errs
upr <- beta.hat + qt(.975, n-2) * std.errs
CI <- cbind(lwr,upr)
colnames(CI) <- c('lower','upper')
rownames(CI) <- c('Intercept', 'x')
CI
```

```
##               lower     upper
## Intercept -4.868409 -1.636813
## x          1.910437  2.462943
```

These intervals are the same as what we get when we use the `confint()` function.

```
confint(model)
```

```
##                 2.5 %    97.5 %
## (Intercept) -4.868409 -1.636813
## x            1.910437  2.462943
```

2.2 ANOVA model

The anova model is also a linear model and all we must do is create a appropriate design matrix. Given the design matrix $\boldsymbol{X}$, all the calculations are identical as in the simple regression case.

2.2.1 Cell means representation

Recall the cell means representation is

$$y_{i,j} = \mu_i + \epsilon_{i,j}$$

where $y_{i,j}$ is the jth observation within the ith group. To clearly show the creation of the $\boldsymbol{X}$ matrix, let the number of groups be $p = 3$ and the number of observations per group be $n_i = 4$. We now expand the formula to show all the data.

$$\begin{aligned}
y_{1,1} &= \mu_1 + \epsilon_{1,1} \\
y_{1,2} &= \mu_1 + \epsilon_{1,2} \\
y_{1,3} &= \mu_1 + \epsilon_{1,3} \\
y_{1,4} &= \mu_1 + \epsilon_{1,4} \\
y_{2,1} &= \mu_2 + \epsilon_{2,1} \\
y_{2,2} &= \mu_2 + \epsilon_{2,2} \\
y_{2,3} &= \mu_2 + \epsilon_{2,3} \\
y_{2,4} &= \mu_2 + \epsilon_{2,4} \\
y_{3,1} &= \mu_3 + \epsilon_{3,1} \\
y_{3,2} &= \mu_3 + \epsilon_{3,2} \\
y_{3,3} &= \mu_3 + \epsilon_{3,3} \\
y_{3,4} &= \mu_3 + \epsilon_{3,4}
\end{aligned}$$

In an effort to write the model as $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ we will write the above as

$$
\begin{aligned}
y_{1,1} &= 1\mu_1 + 0\mu_2 + 0\mu_3 + \epsilon_{1,1} \\
y_{1,2} &= 1\mu_1 + 0\mu_2 + 0\mu_3 + \epsilon_{1,2} \\
y_{1,3} &= 1\mu_1 + 0\mu_2 + 0\mu_3 + \epsilon_{1,3} \\
y_{1,4} &= 1\mu_1 + 0\mu_2 + 0\mu_3 + \epsilon_{1,4} \\
y_{2,1} &= 0\mu + 1\mu_2 + 0\mu_3 + \epsilon_{2,1} \\
y_{2,2} &= 0\mu + 1\mu_2 + 0\mu_3 + \epsilon_{2,2} \\
y_{2,3} &= 0\mu + 1\mu_2 + 0\mu_3 + \epsilon_{2,3} \\
y_{2,4} &= 0\mu + 1\mu_2 + 0\mu_3 + \epsilon_{2,4} \\
y_{3,1} &= 0\mu + 0\mu_2 + 1\mu_3 + \epsilon_{3,1} \\
y_{3,2} &= 0\mu + 0\mu_2 + 1\mu_3 + \epsilon_{3,2} \\
y_{3,3} &= 0\mu + 0\mu_2 + 1\mu_3 + \epsilon_{3,3} \\
y_{3,4} &= 0\mu + 0\mu_2 + 1\mu_3 + \epsilon_{3,4}
\end{aligned}
$$

and we will finally be able to write the matrix version

$$
\underbrace{\begin{bmatrix} y_{1,1} \\ y_{1,2} \\ y_{1,3} \\ y_{1,4} \\ y_{2,1} \\ y_{2,2} \\ y_{2,3} \\ y_{2,4} \\ y_{3,1} \\ y_{3,2} \\ y_{3,3} \\ y_{3,4} \end{bmatrix}}_{\boldsymbol{y}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}}_{\boldsymbol{\beta}} + \underbrace{\begin{bmatrix} \epsilon_{1,1} \\ \epsilon_{1,2} \\ \epsilon_{1,3} \\ \epsilon_{1,4} \\ \epsilon_{2,1} \\ \epsilon_{2,2} \\ \epsilon_{2,3} \\ \epsilon_{2,4} \\ \epsilon_{3,1} \\ \epsilon_{3,2} \\ \epsilon_{3,3} \\ \epsilon_{3,4} \end{bmatrix}}_{\boldsymbol{\epsilon}}
$$

Notice that each column of the $\boldsymbol{X}$ matrix is acting as an indicator if the observation is an element of the appropriate group. As such, these are often called *indicator variables*. Another term for these, which I find less helpful, is *dummy variables*.

2.2.2 Offset from reference group

In this model representation of ANOVA, we have an overall mean and then offsets from the control group (which will be group one). The model is thus

$$y_{i,j} = \mu + \tau_i + \epsilon_{i,j}$$

where $\tau_1 = 0$. We can write this in matrix form as

$$\underbrace{\begin{bmatrix} y_{1,1} \\ y_{1,2} \\ y_{1,3} \\ y_{1,4} \\ y_{2,1} \\ y_{2,2} \\ y_{2,3} \\ y_{2,4} \\ y_{3,1} \\ y_{3,2} \\ y_{3,3} \\ y_{3,4} \end{bmatrix}}_{\boldsymbol{y}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} \mu \\ \tau_2 \\ \tau_3 \end{bmatrix}}_{\boldsymbol{\beta}} + \underbrace{\begin{bmatrix} \epsilon_{1,1} \\ \epsilon_{1,2} \\ \epsilon_{1,3} \\ \epsilon_{1,4} \\ \epsilon_{2,1} \\ \epsilon_{2,2} \\ \epsilon_{2,3} \\ \epsilon_{2,4} \\ \epsilon_{3,1} \\ \epsilon_{3,2} \\ \epsilon_{3,3} \\ \epsilon_{3,4} \end{bmatrix}}_{\boldsymbol{\epsilon}}$$

2.3 Exercises

1. We will do a simple ANOVA analysis on example 8.2 from Ott & Longnecker using the matrix representation of the model. A clinical psychologist wished to compare three methods for reducing hostility levels in university students, and used a certain test (HLT) to measure the degree of hostility. A high score on the test indicated great hostility. The psychologist used 24 students who obtained high and nearly equal scores in the experiment. eight were selected at random from among the 24 problem cases and were treated with method 1. Seven of the remaining 16 students were selected at random and treated with method 2. The remaining nine students were treated with method 3. All treatments were continued for a one-semester period. Each student was given the HLT test at the end of the semester, with the results show in the following table. (This analysis was done in section 8.3 of my STA 570 notes)

Method	Values
1	96, 79, 91, 85, 83, 91, 82, 87
2	77, 76, 74, 73, 78, 71, 80
3	66, 73, 69, 66, 77, 73, 71, 70, 74

We will be using the cell means model of ANOVA

$$y_{ij} = \beta_i + \epsilon_{ij}$$

where β_i is the mean of group i and $\epsilon_{ij} \overset{iid}{\sim} N\left(0, \sigma^2\right)$.

a. Create one vector of all 24 hostility test scores `y`. (Use the `c()` function.)

b. Create a design matrix `X` with dummy variables for columns that code for what group an observation belongs to. Notice that `X` will be a 24 rows by 3 column matrix. *Hint: An R function that might be handy is `cbind(a,b)` which will bind two vectors or matrices together along the columns. (There is also a corresponding `rbind()` function that binds vectors/matrices along rows.) Also you'll like to have the repeat command `rep()`.*

c) Find $\hat{\boldsymbol{\beta}}$ using the matrix formula given in class. *Hint: The R function `t(A)` computes the matrix transpose $\mathbf{A}^T$, `solve(A)` computes $\mathbf{A}^{-1}$, and the operator `%*%` does matrix multiplication (used as `A %*% B`).*

d) Examine the matrix $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$. What do you notice about it? In particular, think about the

result when you right multiply by $\mathbf{y}$. How does this matrix calculate the appropriate group means and using the appropriate group sizes n_i?

2. We will calculate the y-intercept and slope estimates in a simple linear model using matrix notation. We will use a data set that gives the diameter at breast height (DBH) versus tree height for a randomly selected set of trees. In addition, for each tree, a ground measurement of crown closure (CC) was taken. Larger values of crown closure indicate more shading and is often associated with taller tree morphology (possibly). We will be interested in creating a regression model that predicts height based on DBH and CC. In the interest of reduced copying, we will only use 10 observations. *(Note: I made this data up and the DBH values might be unrealistic. Don't make fun of me.)*

DBH	30.5	31.5	31.7	32.3	33.3	35	35.4	35.6	36.3	37.8
CC	0.74	0.69	0.65	0.72	0.58	0.5	0.6	0.7	0.52	0.6
Height	58	64	65	70	68	63	78	80	74	76

We are interested in fitting the regression model

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \epsilon_i$$

where β_0 is the y-intercept and β_1 is the slope parameter associated with DBH and β_2 is the slope parameter associated with Crown Closure.

a) Create a vector of all 10 heights $\mathbf{y}$.
b) Create the design matrix $\mathbf{X}$.
c) Find $\hat{\boldsymbol{\beta}}$ using the matrix formula given in class.
d) Compare your results to the estimated coefficients you get using the `lm()` function. To add the second predictor to the model, your call to `lm()` should look something like `lm(Height ~ DBH + CrownClosure)`.

Chapter 3

Inference

```
library(tidyverse)  # ggplot2, dplyr, tidyr
```

3.1 F-tests

We wish to develop a rigorous way to compare nested models and decide if a complicated model explains enough more variability than a simple model to justify the additional intellectual effort of thinking about the data in the complicated fashion.

It is important to specify that we are developing a way of testing nested models. By nested, we mean that the simple model can be created from the full model just by setting one or more model parameters to zero.

3.1.1 Theory

Recall that in the simple regression and ANOVA cases we were interested in comparing a simple model versus a more complex model. For each model we computed the residual sum of squares (RSS) and said that if the complicated model performed much better than the simple then $RSS_{simple} \gg RSS_{complex}$. To do this we needed to standardize by the number of parameters added to the model and the degrees of freedom remaining in the full model. We first defined $RSS_{diff} = RSS_{simple} - RSS_{complex}$ and let df_{diff} be the number of parameters difference between the simple and complex models. Then we had

$$F = \frac{RSS_{difference}/df_{diff}}{RSS_{complex}/df_{complex}}$$

and we claimed that if the null hypothesis was true (i.e. the complex model is an unnecessary obfuscation of the simple), then this ratio follows an F -distribution with degrees of freedom df_{diff} and $df_{complex}$.

The critical assumption for the F-test to be appropriate is that the error terms are independent and normally distributed with constant variance.

We will consider a data set from Johnson and Raven (1973) which also appears in Weisberg (1985). This data set is concerned with the number of tortoise species on $n = 30$ different islands in the Galapagos. The variables of interest in the data set are:

Variable	Description
`Species`	Number of tortoise species found on the island
`Endimics`	Number of tortoise species endemic to the island

Variable	Description
`Elevation`	Elevation of the highest point on the island
`Area`	Area of the island (km^2)
`Nearest`	Distance to the nearest neighboring island (km)
`Scruz`	Distance to the Santa Cruz islands (km)
`Adjacent`	Area of the nearest adjacent island (km^2)

We will first read in the data set from the package `faraway`.

```
data('gala', package='faraway')     # import the data set
head(gala)                          # show the first couple of rows
```

```
##              Species Endemics  Area Elevation Nearest Scruz Adjacent
## Baltra            58       23 25.09       346     0.6   0.6     1.84
## Bartolome         31       21  1.24       109     0.6  26.3   572.33
## Caldwell           3        3  0.21       114     2.8  58.7     0.78
## Champion          25        9  0.10        46     1.9  47.4     0.18
## Coamano            2        1  0.05        77     1.9   1.9   903.82
## Daphne.Major      18       11  0.34       119     8.0   8.0     1.84
```

First we will create the full model that predicts the number of species as a function of elevation, area, nearest, scruz and adjacent. Notice that this model has $p = 6$ β_i values (one for each coefficient plus the intercept).

$$y_i = \beta_0 + \beta_1 Area_i + \beta_2 Elevation_i + \beta_3 Nearest_i + \beta_4 Scruz_i + \beta_5 Adjacent_i + \epsilon_i$$

We can happily fit this model just by adding terms on the left hand side of the model formula. Notice that R creates the design matrix X for us.

```
M.c <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent, data=gala)
model.matrix(M.c)  # this is the design matrix X.
```

```
##              (Intercept)    Area Elevation Nearest Scruz Adjacent
## Baltra                 1   25.09       346     0.6   0.6     1.84
## Bartolome              1    1.24       109     0.6  26.3   572.33
## Caldwell               1    0.21       114     2.8  58.7     0.78
## Champion               1    0.10        46     1.9  47.4     0.18
## Coamano                1    0.05        77     1.9   1.9   903.82
## Daphne.Major           1    0.34       119     8.0   8.0     1.84
## Daphne.Minor           1    0.08        93     6.0  12.0     0.34
## Darwin                 1    2.33       168    34.1 290.2     2.85
## Eden                   1    0.03        71     0.4   0.4    17.95
## Enderby                1    0.18       112     2.6  50.2     0.10
## Espanola               1   58.27       198     1.1  88.3     0.57
## Fernandina             1  634.49      1494     4.3  95.3  4669.32
## Gardner1               1    0.57        49     1.1  93.1    58.27
## Gardner2               1    0.78       227     4.6  62.2     0.21
## Genovesa               1   17.35        76    47.4  92.2   129.49
## Isabela                1 4669.32      1707     0.7  28.1   634.49
## Marchena               1  129.49       343    29.1  85.9    59.56
## Onslow                 1    0.01        25     3.3  45.9     0.10
## Pinta                  1   59.56       777    29.1 119.6   129.49
## Pinzon                 1   17.95       458    10.7  10.7     0.03
## Las.Plazas             1    0.23        94     0.5   0.6    25.09
## Rabida                 1    4.89       367     4.4  24.4   572.33
```

```
## SanCristobal           1  551.62       716    45.2  66.6     0.57
## SanSalvador            1  572.33       906     0.2  19.8     4.89
## SantaCruz              1  903.82       864     0.6   0.0     0.52
## SantaFe                1   24.08       259    16.5  16.5     0.52
## SantaMaria             1  170.92       640     2.6  49.2     0.10
## Seymour                1    1.84       147     0.6   9.6    25.09
## Tortuga                1    1.24       186     6.8  50.9    17.95
## Wolf                   1    2.85       253    34.1 254.7     2.33
## attr(,"assign")
## [1] 0 1 2 3 4 5
```

All the usual calculations from chapter two can be calculated and we can see the summary table for this regression as follows:

```
summary(M.c)
```

```
##
## Call:
## lm(formula = Species ~ Area + Elevation + Nearest + Scruz + Adjacent,
##     data = gala)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -111.679  -34.898   -7.862   33.460  182.584
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.068221  19.154198   0.369 0.715351
## Area        -0.023938   0.022422  -1.068 0.296318
## Elevation    0.319465   0.053663   5.953 3.82e-06 ***
## Nearest      0.009144   1.054136   0.009 0.993151
## Scruz       -0.240524   0.215402  -1.117 0.275208
## Adjacent    -0.074805   0.017700  -4.226 0.000297 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60.98 on 24 degrees of freedom
## Multiple R-squared:  0.7658, Adjusted R-squared:  0.7171
## F-statistic:  15.7 on 5 and 24 DF,  p-value: 6.838e-07
```

3.1.2 Testing All Covariates

The first test we might want to do is to test if any of the covariates are significant. That is to say that we want to test the full model versus the simple null hypothesis model

$$y_i = \beta_0 + \epsilon_i$$

that has no covariates and only a y-intercept. So we will create a simple model

```
M.s <- lm(Species ~ 1, data=gala)
```

and calculate the appropriate Residual Sums of Squares (RSS) for each model, along with the difference in degrees of freedom between the two models.

```
RSS.c <- sum(resid(M.c)^2)
RSS.s <- sum(resid(M.s)^2)
```

```
df.diff <- 5                # complex model has 5 additional parameters
df.c <- 30 - 6              # complex model has 24 degrees of freedom left
```

The F-statistic for this test is therefore

```
F.stat <-  ( (RSS.s - RSS.c) / df.diff ) / ( RSS.c / df.c )
F.stat
```

```
## [1] 15.69941
```

and should be compared against the F-distribution with 5 and 24 degrees of freedom. Because a large difference between RSS.s and RSS.c would be evidence for the alternative, larger model, the p-value for this test is

$$p - value = P\left(F_{5,24} \geq \texttt{F.stat}\right)$$

```
p.value <-  1 - pf(15.699, 5, 24)
p.value
```

```
## [1] 6.839486e-07
```

Both the F.stat and its p-value are given at the bottom of the summary table. However, I might be interested in creating an ANOVA table for this situation.

Source	df	Sum Sq	Mean Sq	F	p-value
Difference	$p-1$	RSS_d	$MSE_d = RSS_d/(p-1)$	MSE_d/MSE_c	$P(F > F_{p-1,n-p})$
Complex	$n-p$	RSS_c	$MSE_c = RSS_c/(n-p)$		
Simple	$n-1$	RSS_s			

This table can be obtained from R by using the `anova()` function on the two models of interest. As usual with R, it does not show the simple row, but rather concentrates on the difference row.

```
anova(M.s, M.c)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ 1
## Model 2: Species ~ Area + Elevation + Nearest + Scruz + Adjacent
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     29 381081
## 2     24  89231  5    291850 15.699 6.838e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.1.3 Testing a Single Covariate

For a particular covariate, β_j, we might wish to perform a test to see if it can be removed from the model. It can be shown that the F-statistic can be re-written as

$$
\begin{aligned}
F &= \frac{\left[RSS_s - RSS_c\right]/1}{RSS_c/\left(n-p\right)} \\
&= \vdots \\
&= \left[\frac{\hat{\beta}_j}{SE\left(\hat{\beta}_j\right)}\right]^2 \\
&= t^2
\end{aligned}
$$

where t has a t-distribution with $n-p$ degrees of freedom under the null hypothesis that the simple model is sufficient.

We consider the case of removing the covariate `Area` from the model and will calculate our test statistic using both methods.

```
M.c <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent, data=gala)
M.s <- lm(Species ~        Elevation + Nearest + Scruz + Adjacent, data=gala)
RSS.c <- sum( resid(M.c)^2 )
RSS.s <- sum( resid(M.s)^2 )
df.d <- 1
df.c <- 30-6
F.stat <- ((RSS.s - RSS.c)/1) / (RSS.c / df.c)
F.stat
```

```
## [1] 1.139792
```

```
1 - pf(F.stat, 1, 24)
```

```
## [1] 0.296318
```

```
sqrt(F.stat)
```

```
## [1] 1.067611
```

To calculate it using the estimated coefficient and its standard error, we must grab those values from the summary table

```
temp <- summary(M.c)
temp$coefficients
```

```
##                 Estimate  Std. Error      t value     Pr(>|t|)
## (Intercept)  7.068220709 19.15419782  0.369016796 7.153508e-01
## Area        -0.023938338  0.02242235 -1.067610554 2.963180e-01
## Elevation    0.319464761  0.05366280  5.953187968 3.823409e-06
## Nearest      0.009143961  1.05413595  0.008674366 9.931506e-01
## Scruz       -0.240524230  0.21540225 -1.116628222 2.752082e-01
## Adjacent    -0.074804832  0.01770019 -4.226216850 2.970655e-04
```

```
beta.area <- temp$coefficients[2,1]
SE.beta.area <- temp$coefficients[2,2]
t <- beta.area / SE.beta.area
t
```

```
## [1] -1.067611
```

```
2 * pt(t, 24)
```

```
## [1] 0.296318
```

All that hand calculation is tedious, so we can again use the `anova()()` command to compare the two models.

```
anova(M.s, M.c)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ Elevation + Nearest + Scruz + Adjacent
## Model 2: Species ~ Area + Elevation + Nearest + Scruz + Adjacent
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1     25 93469
## 2     24 89231  1    4237.7 1.1398 0.2963
```

3.1.4 Testing a Subset of Covariates

Often a researcher will want to remove a subset of covariates from the model. In the Galapagos example, Area, Nearest, and Scruz all have non-significant p-values and would be removed when comparing the full model to the model without that one covariate. While each of them might be non-significant, is the sum of all three significant?

Because the individual $\hat{\beta}_j$ values are not independent, then we cannot claim that the subset is not statistically significant just because each variable in turn was insignificant. Instead we again create simple and complex models in the same fashion as we have previously done.

```
M.c <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent, data=gala)
M.s <- lm(Species ~        Elevation +                   Adjacent, data=gala)
anova(M.s, M.c)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ Elevation + Adjacent
## Model 2: Species ~ Area + Elevation + Nearest + Scruz + Adjacent
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     27 100003
## 2     24  89231  3     10772 0.9657  0.425
```

We find a large p-value associated with this test and can safely stay with the null hypothesis, that the simple model is sufficient to explain the observed variability in the number of species of tortoise.

3.2 Confidence Intervals for location parameters

Recall that

$$\hat{\boldsymbol{\beta}} \sim N\left(\boldsymbol{\beta}, \sigma^2 \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right)$$

and it is easy to calculate the estimate of σ^2. This estimate will be the "average" squared residual

$$\hat{\sigma}^2 = \frac{RSS}{df}$$

where RSS is the residual sum of squares and df is the degrees of freedom $n - p$ where p is the number of β_j parameters. Therefore the standard error of the $\hat{\beta}_j$ values is

$$SE\left(\hat{\beta}_j\right) = \sqrt{\hat{\sigma}^2 \left(\mathbf{X}^T\mathbf{X}\right)^{-1}_{jj}}$$

We can see this calculation in the summary regression table. We again consider the Galapagos Island data set. First we must create the design matrix

```
y <- gala$Species
X <- cbind( rep(1,30), gala$Elevation, gala$Adjacent )
```

And then create $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$

```
XtXinv <- solve(  t(X) %*% X )
XtXinv
```

```
##               [,1]          [,2]          [,3]
## [1,]  6.094829e-02 -8.164025e-05  9.312123e-06
## [2,] -8.164025e-05  2.723835e-07 -7.126027e-08
## [3,]  9.312123e-06 -7.126027e-08  6.478031e-08
```

```
diag(XtXinv)
```

```
## [1] 6.094829e-02 2.723835e-07 6.478031e-08
```

Eventually we will need $\hat{\beta}$

```
beta.hat <- XtXinv %*% t(X) %*% y
beta.hat
```

```
##             [,1]
## [1,]  1.4328722
## [2,]  0.2765683
## [3,] -0.0688855
```

And now find the estimate $\hat{\sigma}$

```
H <- X %*% XtXinv %*% t(X)
y.hat <- H %*% y
RSS <- sum( (y-y.hat)^2 )
sigma.hat <- sqrt(  RSS/(30-3) )
sigma.hat
```

```
## [1] 60.85898
```

The standard errors of $\hat{\beta}$ is thus

```
sqrt( sigma.hat^2 * diag(XtXinv) )
```

```
## [1] 15.02468680  0.03176253  0.01548981
```

We can double check that this is what R calculates in the summary table

```
model <- lm(Species ~ Elevation + Adjacent, data=gala)
summary(model)
```

```
##
## Call:
## lm(formula = Species ~ Elevation + Adjacent, data = gala)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -103.41  -34.33  -11.43   22.57  203.65
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.43287   15.02469   0.095 0.924727
## Elevation    0.27657    0.03176   8.707 2.53e-09 ***
```

```
## Adjacent    -0.06889    0.01549  -4.447 0.000134 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60.86 on 27 degrees of freedom
## Multiple R-squared:  0.7376, Adjusted R-squared:  0.7181
## F-statistic: 37.94 on 2 and 27 DF,  p-value: 1.434e-08
```

It is highly desirable to calculate confidence intervals for the regression parameters. Recall that the general form of a confidence interval is

$$Estimate \pm Critical\,Value \cdot StandardError\,(Estimate)$$

For any specific β_j we will have

$$\hat{\beta}_j \pm t_{n-p}^{1-\alpha/2}\,\hat{\sigma}\sqrt{(\mathbf{X}^T\mathbf{X})_{jj}^{-1}}$$

where $\hat{\sigma}^2\left(\mathbf{X}^T\mathbf{X}\right)_{jj}^{-1}$ is the $[j,j]$ element of the variance/covariance of $\hat{\boldsymbol{\beta}}$.

To demonstrate this, we return to the Galapagos Island data set.

Finally we can calculate confidence intervals for our three β_j values

```
lower <- beta.hat - qt(.975, 27) * sigma.hat * sqrt( diag(XtXinv) )
upper <- beta.hat + qt(.975, 27) * sigma.hat * sqrt( diag(XtXinv) )
cbind(lower, upper)
```

```
##             [,1]        [,2]
## [1,] -29.395239 32.26098305
## [2,]   0.211397  0.34173962
## [3,]  -0.100668 -0.03710303
```

That is certainly a lot of work to do by hand (even with R doing all the matrix multiplication) but we can get these from R by using the `confint()`() command.

```
confint(model)
```

```
##                   2.5 %      97.5 %
## (Intercept) -29.395239 32.26098305
## Elevation     0.211397  0.34173962
## Adjacent     -0.100668 -0.03710303
```

3.3 Prediction and Confidence Intervals for a response

Given a vector of predictor covariates $\boldsymbol{x}_0$ (think of $\boldsymbol{x}_0^T$ as potentially one row in $\boldsymbol{X}$. Because we might want to predict some other values than what we observe, we do not restrict ourselves to *only* rows in $\boldsymbol{X}$), we want to make inference on the expected value $\hat{y}_0$. We can calculate the value by

$$\hat{y}_0 = \boldsymbol{x}_0^T\hat{\boldsymbol{\beta}}$$

and we are interested in two different types of predictions.

1. We might be interested in the uncertainty of a new data point. This uncertainty has two components: the uncertainty of the regression model and uncertainty of a new data point from its expected value.
2. Second, we might be interested in only the uncertainty about the regression model.

We note that because $\boldsymbol{x}_0^T$ is just a constant, we can calculate the variance of this value as

$$\begin{aligned} Var\left(\boldsymbol{x}_0^T\hat{\boldsymbol{\beta}}\right) &= \boldsymbol{x}_0^T\, Var\left(\hat{\boldsymbol{\beta}}\right)\, \boldsymbol{x}_0 \\ &= \boldsymbol{x}_0^T\, \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\sigma^2\, \boldsymbol{x}_0 \\ &= \boldsymbol{x}_0^T\, \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\, \boldsymbol{x}_0\, \sigma^2 \end{aligned}$$

and use this to calculate two types of intervals. First, a prediction interval for a new observation is

$$\hat{y}_0 \pm t_{n-p}^{1-\alpha/2}\, \hat{\sigma}\sqrt{1 + \boldsymbol{x}_0^T\, \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\, \boldsymbol{x}_0}$$

and a confidence interval for the mean response for the given $\boldsymbol{x}_0$ is

$$\hat{y}_0 \pm t_{n-p}^{1-\alpha/2}\, \hat{\sigma}\sqrt{\boldsymbol{x}_0^T\, \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\, \boldsymbol{x}_0}$$

Again using the Galapagos Island data set as an example, we might be interested in predicting the number of tortoise species of an island with highest point 400 meters and nearest adjacent island with area $200km^2$. We then have

$$\boldsymbol{x}_0^T = \begin{bmatrix} 1 & 400 & 200 \end{bmatrix}$$

and we can calculate

```
x0 <- c(1, 400, 200)
y0 <- t(x0) %*% beta.hat
y0
```

```
##           [,1]
## [1,] 98.28309
```

and then calculate $\boldsymbol{x}_0^T\, \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\, \boldsymbol{x}_0$

```
xt.XtXinv.x <- t(x0) %*% solve( t(X) %*% X ) %*% x0
```

Thus the prediction interval will be

```
c(y0 - qt(.975, 27) * sigma.hat * sqrt(1 + xt.XtXinv.x),
  y0 + qt(.975, 27) * sigma.hat * sqrt(1 + xt.XtXinv.x))
```

```
## [1] -28.70241 225.26858
```

while a confidence interval for the expectation is

```
c(y0 - qt(.975, 27) * sigma.hat * sqrt(xt.XtXinv.x),
  y0 + qt(.975, 27) * sigma.hat * sqrt(xt.XtXinv.x))
```

```
## [1]  75.21317 121.35301
```

These prediction and confidence intervals can be calculated in R using the predict() function

```
x0 <- data.frame(Elevation=400, Adjacent=200)
predict(model, newdata=x0, interval='prediction')
```

```
##        fit       lwr      upr
## 1 98.28309 -28.70241 225.2686
```

```
predict(model, newdata=x0, interval='confidence')
```

```
##        fit      lwr     upr
## 1 98.28309 75.21317 121.353
```

3.4 Interpretation with Correlated Covariates

The standard interpretation of the slope parameter is that β_j is the amount of increase in y for a one unit increase in the jth covariate, provided that all other covariates stayed the same.

The difficulty with this interpretation is that covariates are often related, and the phrase "all other covariates stayed the same" is often not reasonable. For example, if we have a dataset that models the mean annual temperature of a location as a function of latitude, longitude, and elevation, then it is not physically possible to hold latitude, and longitude constant while changing elevation.

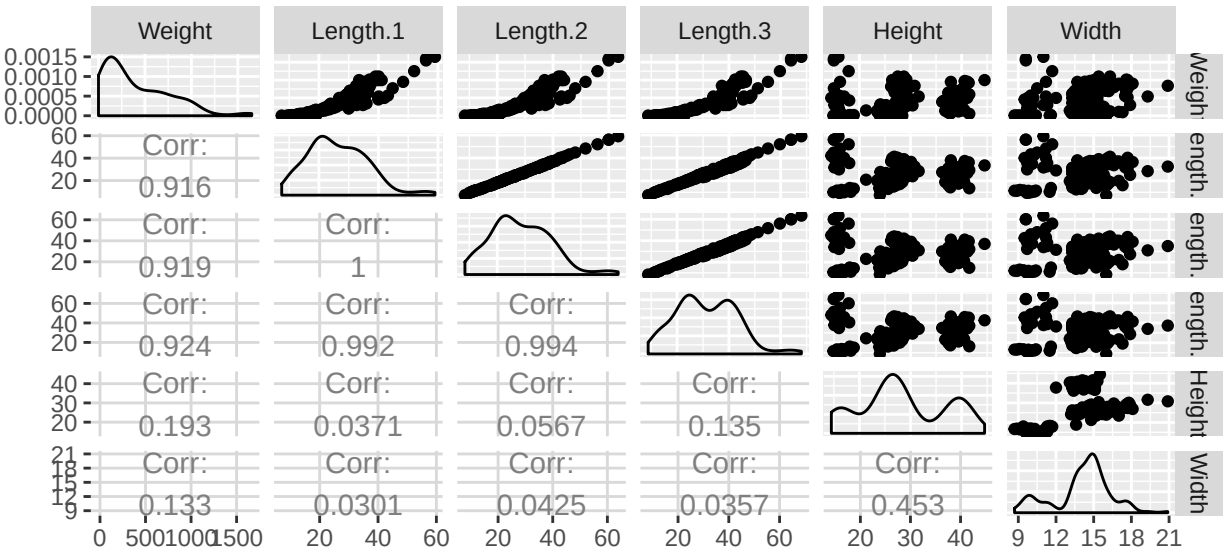
One common issue that make interpretation difficult is that covariates can be highly correlated.

Perch Example: We might be interested in estimating the weight of a fish based off of its length and width. The dataset we will consider is from fishes are caught from the same lake (Laengelmavesi) near Tampere in Finland. The following variables were observed:

Variable	Interpretation
`Weight`	Weight (g)
`Length.1`	Length from nose to beginning of Tail (cm)
`Length.2`	Length from nose to notch of Tail (cm)
`Length.3`	Length from nose to tip of tail (cm)
`Height`	Maximal height as a percentage of `Length.3`
`Width`	Maximal width as a percentage of `Length.3`
`Sex`	0=Female, 1=Male
`Species`	Which species of perch (1-7)

We first look at the data and observe the expected relationship between length and weight.

```
file <- 'https://raw.githubusercontent.com/dereksonderegger/571/master/data-raw/Fish.csv'  # online
file <- '~/github/571/data-raw/Fish.csv'                                                   # on my comp
fish <- read.table(file, header=TRUE, skip=111, sep=',')

### generate a pairs plot in a couple of different ways...
# pairs(fish)
# pairs( Weight ~ Length.1 + Length.2 + Length.3 + Height + Width, data=fish )
# pairs( Weight ~ ., data=fish )

fish %>%
  dplyr::select(Weight, Length.1, Length.2, Length.3, Height, Width) %>%
  GGally::ggpairs(upper=list(continuous='points'),
                  lower=list(continuous='cor'))
```



Naively, we might consider the linear model with all the length effects present.

```
model <- lm(Weight ~ Length.1 + Length.2 + Length.3 + Height + Width, data=fish)
summary(model)
```

```
##
## Call:
## lm(formula = Weight ~ Length.1 + Length.2 + Length.3 + Height +
##     Width, data = fish)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -302.22  -79.72  -39.88   92.63  344.85
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -724.539     77.133  -9.393   <2e-16 ***
## Length.1      32.389     45.134   0.718   0.4741
## Length.2      -9.184     48.367  -0.190   0.8497
## Length.3       8.747     16.283   0.537   0.5919
## Height         4.947      2.768   1.787   0.0759 .
## Width          8.636      6.972   1.239   0.2174
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 132.9 on 152 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.8675, Adjusted R-squared:  0.8631
## F-statistic:   199 on 5 and 152 DF,  p-value: < 2.2e-16
```

This is crazy. There is a negative relationship between `Length.2` and `Weight`. That does not make any sense unless you realize that this is the effect of `Length.2` assuming the other covariates are in the model and can be held constant while changing the value of `Length.2`, which is obviously ridiculous.

If we remove the highly correlated covariates then we see a much better behaved model

```
model <- lm(Weight ~ Length.2 + Height + Width, data=fish)
summary(model)
```

```
##
## Call:
## lm(formula = Weight ~ Length.2 + Height + Width, data = fish)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -306.14  -75.11  -36.45   89.54  337.95
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -701.0750    71.0438  -9.868  < 2e-16 ***
## Length.2      30.4360     0.9841  30.926  < 2e-16 ***
## Height         5.5141     1.4311   3.853 0.000171 ***
## Width          5.6513     5.2016   1.086 0.278974
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 132.3 on 154 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.8669, Adjusted R-squared:  0.8643
## F-statistic: 334.2 on 3 and 154 DF,  p-value: < 2.2e-16
```

When you have two variables in a model that are highly positively correlated, you often find that one will have a positive coefficient and the other will be negative. Likewise, if two variables are highly negatively correlated, the two regression coefficients will often be the same sign.

In this case the sum of the three length covariate estimates was approximately 31 in both cases, but with three length variables, the second could be negative the third be positive with approximately the same magnitude and we get approximately the same model as with both the second and third length variables missing from the model.

In general, you should be very careful with the interpretation of the regression coefficients when the covariates are highly correlated. We will talk about how to recognize these situations and what to do about them later in the course.

3.5 Exercises

1. The dataset prostate in package `faraway` has information about a study of 97 men with prostate cancer. We import the data and examine the first four observations using the following commands.

   ```
   data(prostate, package='faraway')
   head(prostate)
   ```

   It is possible to get information about the data set using the command `help(prostate)`. Fit a model with `lpsa` as the response and all the other variables as predictors.

   a) Compute 90% and 95% confidence intervals for the parameter associated with `age`. Using just these intervals, what could we deduced about the p-value for age in the regression summary. *Hint: look at the help for the function `confint()`. You'll find the `level` option to be helpful.*

   b) Remove all the predictors that are not significant at the 5% level. Test this model against the original model. Which is preferred?

2. Thirty samples of cheddar cheese were analyzed for their content of acetic acid, hydrogen sulfide and lactic acid. Each sample was tasted and scored by a panel of judges and the average taste score

produces. Used the `cheddar` dataset from the `faraway` package (import it the same way you did in problem one, but now use `cheddar`) to answer the following:

a) Fit a regression model with taste as the response and the three chemical contents as predictors. Identify the predictors that are statistically significant at the 5% level.

b) `Acetic` and `H2S` are measured on a $\log_{10}$ scale. Create two new columns in the `cheddar` data frame that contain the values on their original scale. Fit a linear model that uses the three covariates on their non-log scale. Identify the predictors that are statistically significant at the 5% level for this model.

c) Can we use an F-test to compare these two models? Explain why or why not. Which model provides a better fit to the data? Explain your reasoning.

d) For the model in part (a), if a sample of cheese were to have `H2S` increased by 2 (where H2S is on the log scale and we increase this value by 2 using some method), what change in taste would be expected? What caveates must be made in this interpretation? *Hint: I don't want to get into interpreting parameters on the log scale just yet. So just interpret this as adding 2 to the covariate value and predicting the change in taste.*

3. The `sat` data set in the `faraway` package gives data collected to study the relationship between expenditures on public education and test results.

a) Fit a model that with `total` SAT score as the response and only the intercept as a covariate.

b) Fit a model with `total` SAT score as the response and `expend`, `ratio`, and `salary` as predictors (along with the intercept).

c) Compare the models in parts (a) and (b) using an F-test. Is the larger model superior?

d) Examine the summary table of the larger model? Does this contradict your results in part (c)? What might be causing this issue? Create a graph or summary diagnostics to support your guess.

e) Fit the model with `salary` and `ratio` (along with the intercept) as predictor variables and examine the summary table. Which covariates are significant?

f) Now add `takers` to the model (so the model now includes three predictor variables along with the intercept). Test the hypothesis that $\beta_{takers} = 0$ using the summary table.

g) Discuss why `ratio` was not significant in the model in part (e) but was significant in part (f). *Hint: Look at the Residual Standard Error $\hat{\sigma}$ in each model and argue that each t-statistic is some variant of a "signal-to-noise" ratio and that the "noise" part is reduced in the second model.*

Chapter 4

Analysis of Covariance (ANCOVA)

```
library(tidyverse)  # ggplot2, tidyr, dplyr
```

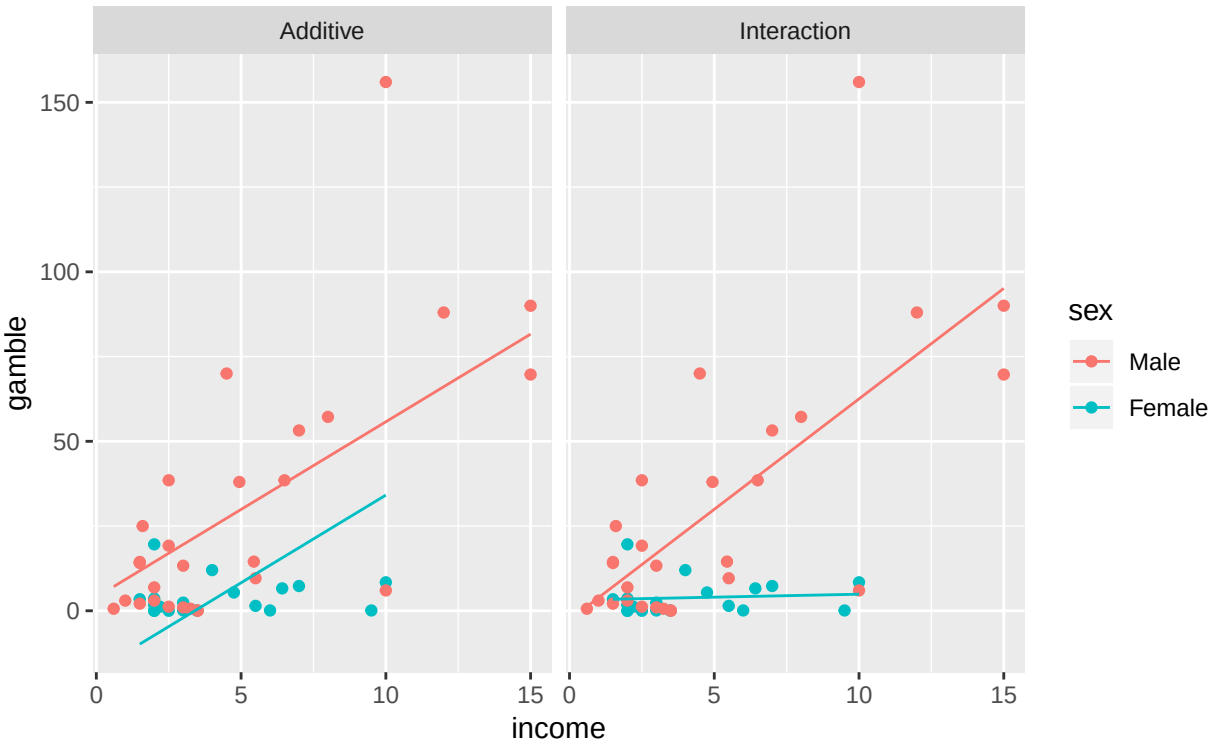
One way that we could extend the ANOVA and regression models is to have both categorical and continuous predictor variables. This model is commonly called ANCOVA which stands for *Analysis of Covariance*.

The dataset `teengamb` in the package `faraway` has data regarding the rates of gambling among teenagers in Britain and their gender and socioeconomic status. One question we might be interested in is how gender and income relate to how much a person gambles. But what should be the effect of gender be?

There are two possible ways that gender could enter the model. Either:

1. We could fit two lines to the data one for males and one for females but require that the lines be parallel (i.e. having the same slopes for income). This is accomplished by having a separate y-intercept for each gender. In effect, the line for the females would be offset by a constant amount from the male line.

2. We could fit two lines but but allow the slopes to differ as well as the y-intercept. This is referred to as an "interaction" between income and gender.



We will now see how to go about fitting these two models. As might be imagined, these can be fit in the same fashion we have been solving the linear models, but require a little finesse in defining the appropriate design matrix $\boldsymbol{X}$.

4.1 Offset parallel Lines (aka additive models)

In order to get offset parallel lines, we want to write a model

$$y_i = \begin{cases} \beta_0 + \beta_1 + \beta_2 x_i + \epsilon_i & \text{if female} \\ \beta_0 + \beta_2 x_i + \epsilon_i & \text{if male} \end{cases}$$

where β_1 is the vertical offset of the female group regression line to the reference group, which is the males regression line. Because the first 19 observations are female, we can this in in matrix form as

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{19} \\ y_{20} \\ \vdots \\ y_{47} \end{bmatrix} = \begin{bmatrix} 1 & 1 & x_1 \\ \vdots & \vdots & \vdots \\ 1 & 1 & x_{19} \\ 1 & 0 & x_{20} \\ \vdots & \vdots & \vdots \\ 1 & 0 & x_{47} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_{19} \\ \epsilon_{20} \\ \vdots \\ \epsilon_{47} \end{bmatrix}$$

I like this representation where β_1 is the offset from the male regression line because it makes it very convenient to test if the offset is equal to zero. The second column of the design matrix referred to as a "dummy variable" or "indicator variable" that codes for the female gender. Notice that even though I have two genders, I only had to add one additional variable to my model because we already had a y-intercept β_0 and we only added one indicator variable for females.

What if we had a third group? Then we would fit another column of indicator variable for the third group. The new beta coefficient in the model would be the offset of the new group to the reference group. For

example we consider $n = 9$ observations with $n_i = 3$ observations per group where $y_{i,j}$is the j th replication of the ith group.

$$\begin{bmatrix} y_{1,1} \\ y_{1,2} \\ y_{1,3} \\ y_{2,1} \\ y_{2,2} \\ y_{2,3} \\ y_{3,1} \\ y_{3,2} \\ y_{3,3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_{1,1} \\ 1 & 0 & 0 & x_{1,2} \\ 1 & 0 & 0 & x_{1,3} \\ 1 & 1 & 0 & x_{2,1} \\ 1 & 1 & 0 & x_{2,2} \\ 1 & 1 & 0 & x_{2,3} \\ 1 & 0 & 1 & x_{3,1} \\ 1 & 0 & 1 & x_{3,2} \\ 1 & 0 & 1 & x_{3,3} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \epsilon_{1,1} \\ \epsilon_{1,2} \\ \epsilon_{1,3} \\ \epsilon_{2,1} \\ \epsilon_{2,2} \\ \epsilon_{2,3} \\ \epsilon_{3,3} \\ \epsilon_{3,2} \\ \epsilon_{3,3} \end{bmatrix}$$

In this model, β_0 is the y-intercept for group 1. The parameter β_1 is the vertical offset from the reference group (group 1) for the second group. Similarly β_2 is the vertical offset for group 3. All groups will share the same slope, β_4.

4.2 Lines with different slopes (aka Interaction model)

We can now include a discrete random variable and create regression lines that are parallel, but often that is inappropriate, such as in the teenage gambling dataset. We want to be able to fit a model that has different slopes.

$$y_i = \begin{cases} (\beta_0 + \beta_1) + (\beta_2 + \beta_3)\, x_i + \epsilon_i & \text{if female} \\ \beta_0 + \beta_2 x_i + \epsilon_i & \text{if male} \end{cases}$$

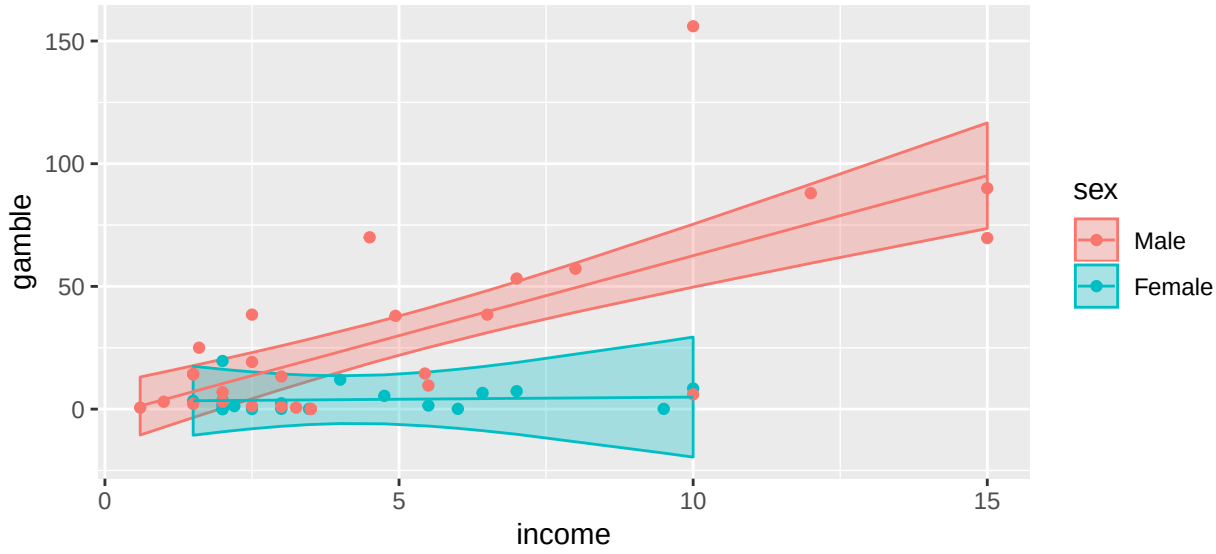
Where β_1 is the offset in y-intercept of the female group from the male group, and β_3 is the offset in slope. Now our matrix formula looks like

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{19} \\ y_{20} \\ \vdots \\ y_{47} \end{bmatrix} = \begin{bmatrix} 1 & 1 & x_1 & x_1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & x_{19} & x_{19} \\ 1 & 0 & x_{20} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_{47} & 0 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_{19} \\ \epsilon_{20} \\ \vdots \\ \epsilon_{47} \end{bmatrix}$$

where the new fourth column is the what I would get if I multiplied the $\boldsymbol{x}$ column element-wise with the dummy-variable column. To fit this model in R we have

```
data('teengamb', package='faraway')

# Forces R to recognize that 0, 1 are categorical, also
# relabels the levels to something I understand.
teengamb$sex <- factor(teengamb$sex, labels=c('Male','Female'))

# Fit a linear model with the interaction of sex and income
# Interactions can be specified useing a colon :
m1 <- lm( gamble ~ sex + income + sex:income, data=teengamb )

# R allows a shortcut for the prior definition
m1 <- lm( gamble ~ sex * income, data=teengamb )

# save the fit, lwr, upr values for each observation
# these are the yhat and CI
```

```
# If columns for fit, upr, lwr are already present, remove them
teengamb <- teengamb %>%
  dplyr::do(if(!is.null(.$fit)){dplyr::select(-fit,-upr,-lwr )}else{.}) %>%
  cbind( predict(m1, interval='conf') )

# Make a nice plot that includes the regreesion line.
ggplot(teengamb, aes(x=income, col=sex, fill=sex)) +
      geom_ribbon(aes(ymin=lwr, ymax=upr),
                  alpha=.3) +   # how solid the layer is
      geom_point(aes(y=gamble)) +
      geom_line(aes(y=fit))
```




```
# print the model summary
summary(m1)
```

```
##
## Call:
## lm(formula = gamble ~ sex * income, data = teengamb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -56.522  -4.860  -1.790   6.273  93.478
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -2.6596     6.3164  -0.421  0.67580
## sexFemale         5.7996    11.2003   0.518  0.60724
## income            6.5181     0.9881   6.597 4.95e-08 ***
## sexFemale:income -6.3432     2.1446  -2.958  0.00502 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.98 on 43 degrees of freedom
## Multiple R-squared:  0.5857, Adjusted R-squared:  0.5568
## F-statistic: 20.26 on 3 and 43 DF,  p-value: 2.451e-08
```

4.3 Iris Example

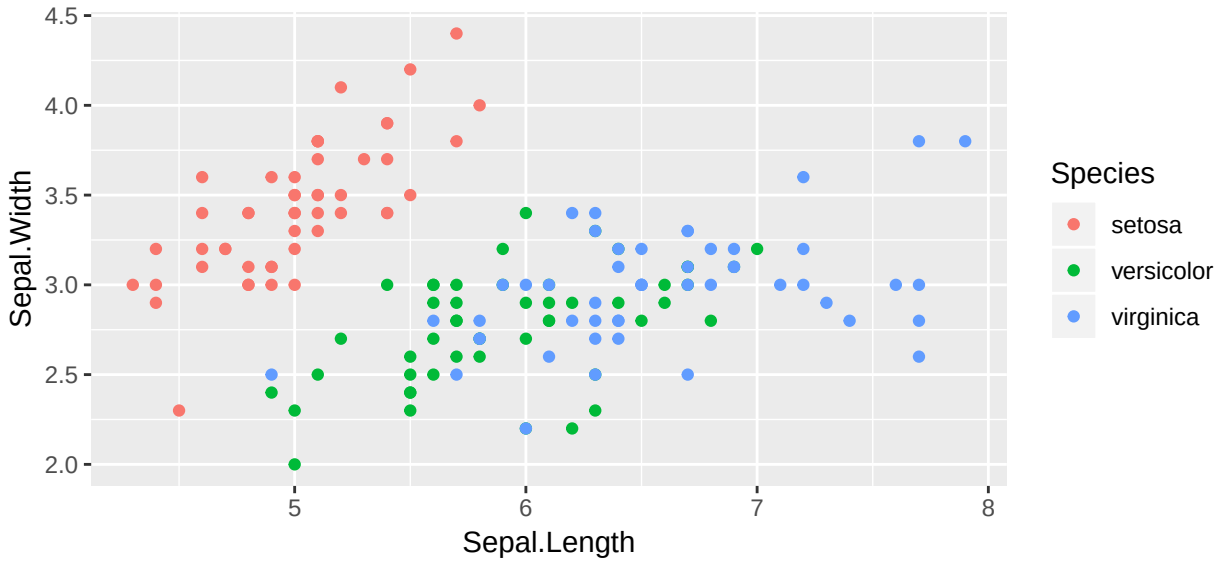
For a second example, we will explore the relationship between sepal length and sepal width for three species of irises. This data set is available in R as `iris`.

```
data(iris)             # read in the iris dataset
levels(iris$Species)   # notice the order of levels of Species
```

```
## [1] "setosa"     "versicolor" "virginica"
```

The very first thing we should do when encountering a dataset is to do some sort of graphical summary to get an idea of what model seems appropriate.

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point()
```

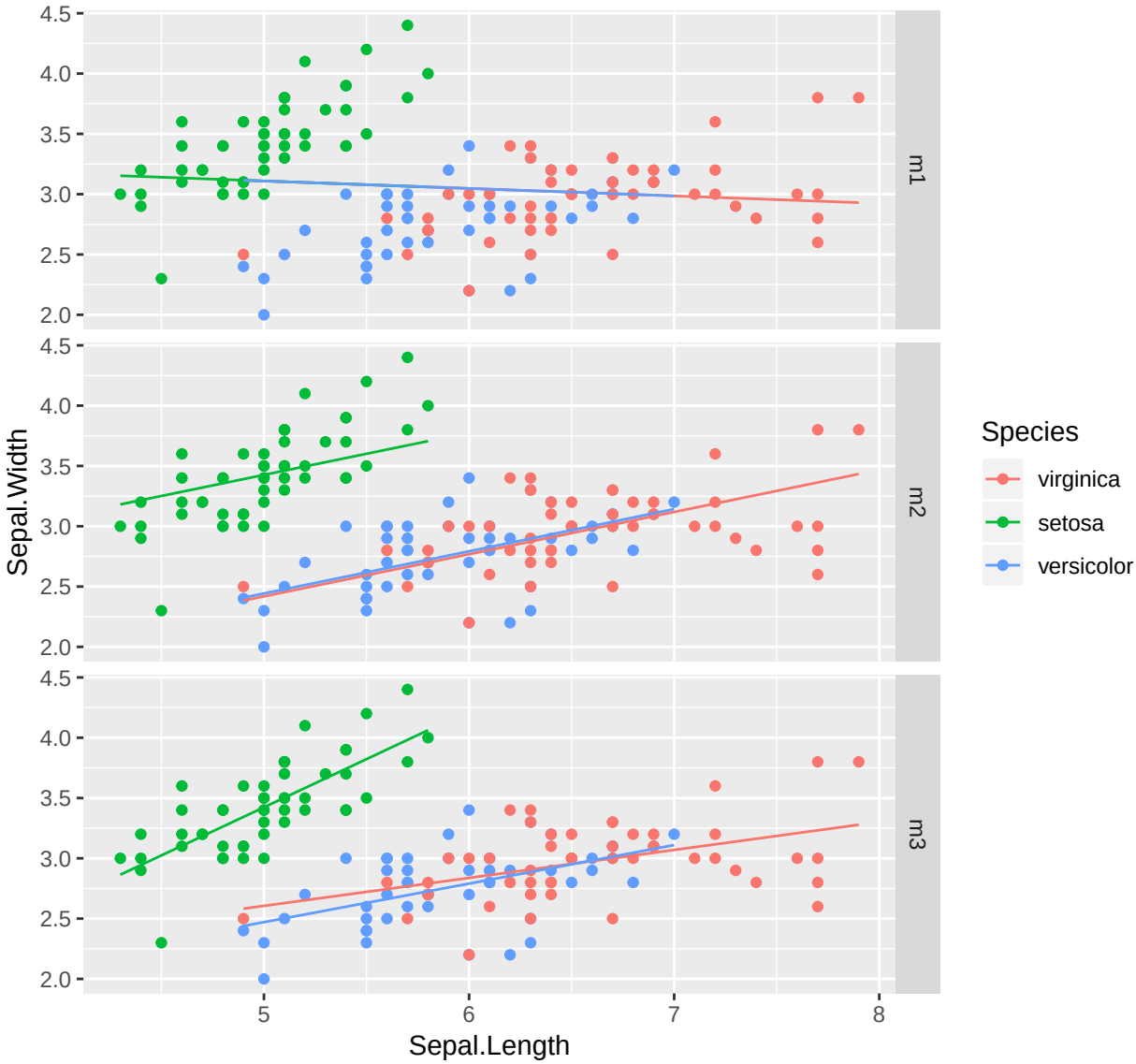


Looking at this graph, it seems that I will likely have a model with different y-intercepts for each species, but it isn't clear to me if we need different slopes.

We consider the sequence of building successively more complex models:

```
# make virginica the reference group
iris$Species <- relevel(iris$Species, ref='virginica')

m1 <- lm( Sepal.Width ~ Sepal.Length, data=iris )            # One line
m2 <- lm( Sepal.Width ~ Sepal.Length + Species, data=iris )  # Parallel Lines
m3 <- lm( Sepal.Width ~ Sepal.Length * Species, data=iris )  # Non-parallel Lines
```

The three models we consider are the following:



Looking at these, it seems obvious that the simplest model where we ignore Species is horrible. The other two models seem decent, and I am not sure about the parallel lines model vs the differing slopes model.

```
summary(m1)$coefficients %>% round(digits=3)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.419      0.254  13.484    0.000
## Sepal.Length   -0.062      0.043  -1.440    0.152
```

For the simplest model, there is so much unexplained noise that the slope variable isn't significant.

Moving onto the next most complicated model, where each species has their own y-intercept, but they share a slope, we have

```
summary(m2)$coefficients %>% round(digits=3)
```

```
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.669      0.308   2.174    0.031
## Sepal.Length         0.350      0.046   7.557    0.000
## Speciessetosa        1.008      0.093  10.798    0.000
```

```
## Speciesversicolor    0.024      0.065   0.370    0.712
```

The first two lines are the y-intercept and slope associated with the reference group and the last two lines are the y-intercept offsets from the reference group to *Setosa* and *Versicolor*, respectively. We have that the slope associated with increasing Sepal Length is significant and that *Setosa* has a statistically different y-intercept than the reference group *Virginica* and that *Versicolor* does not have a statistically different y-intercept than the reference group.

Finally we consider the most complicated model that includes two more slope parameters

```
summary(m3)$coefficients %>% round(digits=3)
```

```
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       1.446      0.405   3.572    0.000
## Sepal.Length                      0.232      0.061   3.790    0.000
## Speciessetosa                    -2.016      0.686  -2.938    0.004
## Speciesversicolor                -0.574      0.605  -0.950    0.344
## Sepal.Length:Speciessetosa        0.567      0.126   4.490    0.000
## Sepal.Length:Speciesversicolor    0.088      0.097   0.905    0.367
```

These parameters are:

Meaning	R-label
Reference group y-intercept	(Intercept)
Reference group slope	Sepal.Length
offset to y-intercept for Setosa	Speciessetosa
offset to y-intercept for Versicolor	Speciesversicolor
offset to slope for Setosa	Sepal.Length:Speciessetosa
offset to slope for Versicolor	Sepal.Length:Speciesversicolor

It appears that slope for *Setosa* is different from the reference group *Virginica*. However because we've added 2 parameters to the model, testing Model2 vs Model3 is not equivalent to just looking at the p-value for that one slope. Instead we need to look at the F-test comparing the two models which will evaluate if the decrease in SSE is sufficient to justify the addition of two parameters.

```
anova(m2, m3)
```

```
## Analysis of Variance Table
##
## Model 1: Sepal.Width ~ Sepal.Length + Species
## Model 2: Sepal.Width ~ Sepal.Length * Species
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    146 12.193
## 2    144 10.680  2    1.5132 10.201 7.19e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-test concludes that there is sufficient decrease in the SSE to justify adding two additional parameters to the model.

4.4 Exercises

1. The in the `faraway` package, there is a dataset named `phbirths` that gives babies birth weights along with their gestational time in utero along with the mother's smoking status.

a. Load and inspect the dataset using

```
data('phbirths', package='faraway')   # load the data within the package
?faraway::phbirths                    # Look at the help file
```

b. Create a plot of the birth weight vs the gestational age. Color code the points based on the mother's smoking status. Does it appear that smoking matters?

c. Fit the simple model (one regression line) along with both the main effects (parallel lines) and interaction (non-parallel lines) ANCOVA model to these data. Which model is preferred?

d. Using whichever model you selected in the previous section, create a graph of the data along with the confidence region for the regression line(s).

e. Now consider only the "full term babies" which are babies with gestational age at birth $\geq$ 36 weeks. With this reduced dataset, repeat parts c,d.

f. Interpret the relationship between gestational length and mother's smoking status on birth weight.

2. The in the `faraway` package, there is a dataset named `clot` that gives information about the time for blood to clot verses the blood dilution concentration when the blood was diluted with prothrombin-free plasma. Unfortunately the researchers had to order the plasma in two different lots (could think of this as two different sources) and need to ascertain if the lot number makes any difference in clotting time.

   a. Log transform the `time` and `conc` variable and plot the log-transformed data with color of the data point indicating the lot number.

   b. Ignoring the slight remaining curvature in the data, perform the appropriate analysis using transformed variables. Does `lot` matter?

3. In the `faraway` package, there is a data set `ToothGrowth` which is data from an experiment giving Vitamin C to guinea pigs. Guinea pigs were give vitamin C doses either via orange juice or ascorbic acid and the response of interest was a measure of tooth growth.

   a. Log transform the `dose` and use that throughout this problem. Use e as the base, which R does by default when you use the `log()` function.

   b. Graph the data, fit appropriate ANCOVA models, and describe the relationship between the delivery method, log(dose) level, and tooth growth. Produce a graph with the data and the regression line(s) along with the confidence region for the line(s).

   c. Just using your graphs and visual inspection, at low dose levels, say $\log(dose) = -0.7$, is there a difference in delivery method? What about at high dose levels, say $\log(dose) = 0.7$? At this point we don't know how to answer this question using appropriate statistical inference, but we will address this in the chapter on contrasts.

Chapter 5

Contrasts

```
library(tidyverse)  # ggplot2, dplyr, tidyr
library(emmeans)    # for emmeans()
library(multcomp)   # for glht()
```

We often are interested in estimating a function of the parameters $\boldsymbol{\beta}$. For example in the offset representation of the ANOVA model with 3 groups we have

$$y_{ij} = \mu + \tau_i + \epsilon_{ij}$$

where

$$\boldsymbol{\beta} = [\mu \ \tau_2 \ \tau_3]^T$$

and μ is the mean of the control group, group one is the control group and thus $\tau_1 = 0$, and τ_2 and τ_3 are the offsets of group two and three from the control group. In this representation, the mean of group two is $\mu + \tau_2$ and is estimated with $\hat{\mu} + \hat{\tau}_2$.

5.1 Estimate and variance

A contrast is a linear combinations of elements of $\hat{\boldsymbol{\beta}}$, which is a fancy way of saying that it is a function of the elements of $\hat{\boldsymbol{\beta}}$ where the elements can be added, subtracted, or multiplied by constants. In particular, the contrast can be represented by the vector $\boldsymbol{c}$ such that the function we are interested in is $\boldsymbol{c}^T\hat{\boldsymbol{\beta}}$.

In the ANOVA case with $k = 3$ where we have the offset representation, I might be interested in the mean of group 2, which could be written as

$$\mu + \tau_2 = \underbrace{\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}}_{\boldsymbol{c}^T} \cdot \underbrace{\begin{bmatrix} \hat{\mu} \\ \hat{\tau}_2 \\ \hat{\tau}_3 \end{bmatrix}}_{\hat{\boldsymbol{\beta}}}$$

Similarly in the simple regression case, I will be interested in the height of the regression line at x_0. This height can be written as

$$\hat{\beta}_0 + \hat{\beta}_1 x_0 = \underbrace{\begin{bmatrix} 1 & x_0 \end{bmatrix}}_{\boldsymbol{c}^T} \cdot \underbrace{\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix}}_{\hat{\boldsymbol{\beta}}}$$

In this manner, we could think of the predicted values $\hat{y}_i$ as just the result of the contrasts $\boldsymbol{X}\hat{\boldsymbol{\beta}}$ where our design matrix takes the role of the contrasts.

One of the properties of maximum likelihood estimator (MLEs), is that they are invariant under transformations. Meaning that since $\hat{\boldsymbol{\beta}}$ is the MLE of $\boldsymbol{\beta}$, then $\boldsymbol{c}^T\hat{\boldsymbol{\beta}}$ is the MLE of $\boldsymbol{c}^T\boldsymbol{\beta}$. The only thing we need to perform hypotheses tests and create confidence intervals is an estimate of the variance of $\boldsymbol{c}^T\hat{\boldsymbol{\beta}}$.

Because we know the variance of $\hat{\boldsymbol{\beta}}$ is

$$Var\left(\hat{\boldsymbol{\beta}}\right) = \sigma^2\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}$$

and because $\boldsymbol{c}$ is a constant, then

$$Var\left(\boldsymbol{c}^T\hat{\boldsymbol{\beta}}\right) = \sigma^2\boldsymbol{c}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{c}$$

and the standard error is found by plugging in our estimate of σ^2 and taking the square root.

$$StdErr\left(\boldsymbol{c}^T\hat{\boldsymbol{\beta}}\right) = \sqrt{\hat{\sigma}^2\boldsymbol{c}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{c}} = \hat{\sigma}\sqrt{\boldsymbol{c}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{c}}$$

As usual, we can now calculate confidence intervals for $\boldsymbol{c}^T\hat{\boldsymbol{\beta}}$ using the usual formula

$$Est \pm t_{n-p}^{1-\alpha/2}\ StdErr\left(\ Est\ \right)$$

$$\boldsymbol{c}^T\hat{\boldsymbol{\beta}} \pm t_{n-p}^{1-\alpha/2}\ \hat{\sigma}\sqrt{\boldsymbol{c}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{c}}$$

Recall the hostility example which was an ANOVA with three groups with the data

Method	Test Scores
1	96 79 91 85 83 91 82 87
2	77 76 74 73 78 71 80
3	66 73 69 66 77 73 71 70 74

We have analyzed this data using both the cell means model and the offset and we will demonstrate how to calculate the group means from the offset representation. Thus we are interested in estimating $\mu + \tau_2$ and $\mu + \tau_3$. I am also interested in estimating the difference between treatment 2 and 3 and will therefore be interested in estimating $\tau_2 - \tau_3$.

```
y <- c(96,79,91,85,83,91,82,87,
       77,76,74,73,78,71,80,
       66,73,69,66,77,73,71,70,74)
groups <- factor(c( rep('Group1',8), rep('Group2',7),rep('Group3',9) ))
```

We can fit the offset model and obtain the design matrix and estimate of $\hat{\sigma}$ via the following code.

```
m <- lm(y ~ groups)             # Fit the ANOVA model (offset representation)
coef(m)                         # Show me beta.hat
```

```
##  (Intercept) groupsGroup2 groupsGroup3
##     86.75000    -11.17857    -15.75000
```

```
X <- model.matrix(m)            # obtains the design matrix
sigma.hat <- summary(m)$sigma   # create the summary table and grab sigma.hat
```

```
beta.hat <- coef(m)
XtX.inv <- solve( t(X) %*% X )
```

Now we calculate

```
contr <- c(1,1,0)  # define my contrast
ctb <- t(contr) %*% beta.hat
std.err <- sigma.hat * sqrt( t(contr) %*% XtX.inv %*% contr )
ctb
```

```
##          [,1]
## [1,] 75.57143
```

```
std.err
```

```
##          [,1]
## [1,] 1.622994
```

and notice this is the exact same estimate and standard error we got for group two when we fit the cell means model.

```
CellMeansModel <- lm(y ~ groups - 1)
summary(CellMeansModel)$coefficients
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## groupsGroup1 86.75000   1.518172 57.14108 1.567052e-24
## groupsGroup2 75.57143   1.622994 46.56296 1.117034e-22
## groupsGroup3 71.00000   1.431347 49.60364 2.993023e-23
```

5.2 Estimating contrasts using glht()

Instead of us doing all the matrix calculations ourselves, all we really need is to is specify the row vector $\boldsymbol{c}^T$. The function that will do the rest of the calculations is the generalized linear hypothesis test function `glht()` that can be found in the multiple comparisons package `multcomp`. The p-values will be adjusted to correct for testing multiple hypothesis, so there may be slight differences compared to the p-value seen in just the regular summary table.

5.2.1 1-way ANOVA

We will again use the Hostility data set and demonstrate how to calculate the point estimates, standard errors and confidence intervals for the group means given a model fit using the offset representation.

```
y <- c(96,79,91,85,83,91,82,87,
       77,76,74,73,78,71,80,
       66,73,69,66,77,73,71,70,74)
groups <- factor(c( rep('Group1',8), rep('Group2',7),rep('Group3',9) ))
m <- lm(y ~ groups)
summary(m)$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)   86.75000   1.518172 57.141079 1.567052e-24
## groupsGroup2 -11.17857   2.222377 -5.030008 5.583942e-05
## groupsGroup3 -15.75000   2.086528 -7.548424 2.063600e-07
```

We will now define a row vector (and it needs to be a matrix or else `glht()` will throw an error. First we note that the simple contrast $\boldsymbol{c}^T = [1\ 0\ 0]$ just grabs the first coefficient and gives us the same estimate and standard error as the summary did.

```
library(multcomp)
contr <- rbind("Intercept"=c(1,0,0)) # 1x3 matrix with row named "Intercept"
test <- glht(m, linfct=contr)        # the linear function to be tested is contr
summary(test)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = y ~ groups)
##
## Linear Hypotheses:
##                Estimate Std. Error t value Pr(>|t|)
## Intercept == 0   86.750      1.518   57.14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Next we calculate the estimate of all the group means μ, $\mu + \tau_2$ and $\mu + \tau_3$ and the difference between group 2 and 3. Notice I can specify more than one contrast at a time.

```
contr <- rbind("Mean of Group 1"=c(1,0,0),
               "Mean of Group 2"=c(1,1,0),
               "Mean of Group 3"=c(1,0,1),
               "Diff G2-G3"     =c(0,1,-1))
test <- glht(m, linfct=contr)
summary(test)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = y ~ groups)
##
## Linear Hypotheses:
##                       Estimate Std. Error t value Pr(>|t|)
## Mean of Group 1 == 0   86.750      1.518  57.141   <0.001 ***
## Mean of Group 2 == 0   75.571      1.623  46.563   <0.001 ***
## Mean of Group 3 == 0   71.000      1.431  49.604   <0.001 ***
## Diff G2-G3 == 0         4.571      2.164   2.112    0.144
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Finally we calculate confidence intervals in the usual manner using the `confint()` function.

```
confint(test, level=0.95)
```

```
##
##   Simultaneous Confidence Intervals
##
## Fit: lm(formula = y ~ groups)
##
## Quantile = 2.6453
## 95% family-wise confidence level
```

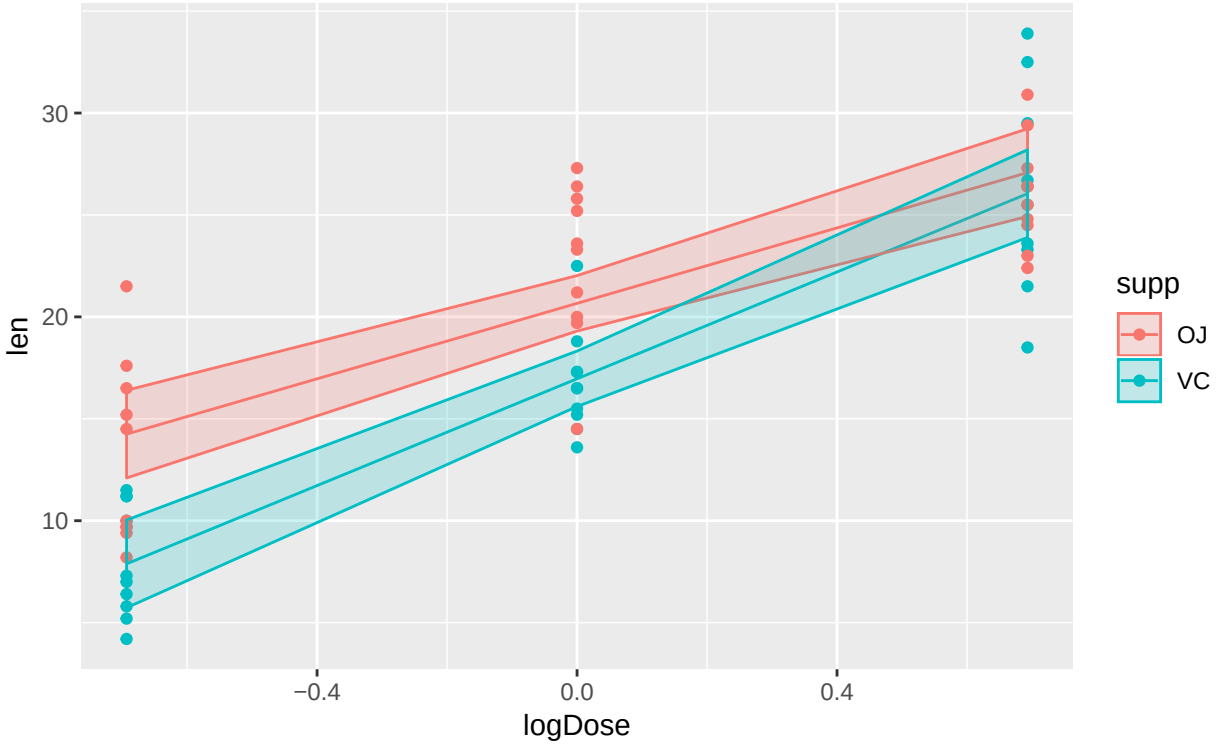
```
##
##
## Linear Hypotheses:
##                       Estimate lwr     upr
## Mean of Group 1 == 0 86.7500  82.7340 90.7660
## Mean of Group 2 == 0 75.5714  71.2782 79.8647
## Mean of Group 3 == 0 71.0000  67.2137 74.7863
## Diff G2-G3 == 0       4.5714  -1.1529 10.2958
```

5.2.2 ANCOVA example

In the ANCOVA case, there are more interesting contrasts to be made. For this example we will use the `ToothGrowth` dataset. This data measuring the effects of different dose levels of vitamin C on tooth growth of guinea pigs. The two different delivery methods are encoded by the variable supp which has levels of orange juice (OJ) and ascorbic acid (VC).

We first fit a ANCOVA model with an interaction between log(dose) level and delivery method and graph the result.

```
data('ToothGrowth')
ToothGrowth$logDose <- log( ToothGrowth$dose )
m <- lm(len ~ logDose * supp, data=ToothGrowth)

# predict() gives me the yhat values and optional CI
# these are just the "contrasts" defined by X matrix!
ToothGrowth <- ToothGrowth %>%
  dplyr::do(if(!is.null(.$fit)){dplyr::select(-fit,-upr,-lwr )}else{.}) %>%
  cbind( predict(m, interval='confidence'))

# Plot the results using ggplot2
ggplot( ToothGrowth, aes(x=logDose, col=supp, fill=supp)) +
    geom_point(aes(y=len)) +
    geom_line(aes(y=fit)) +
    geom_ribbon(aes(ymin=lwr, ymax=upr), alpha=.2)
```



R has fit this model using the offset representation. First we present the summary coefficients so we know which parameters are which.

```
summary(m)$coefficient
```

```
##                 Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)    20.663333  0.6791481 30.425371 1.629404e-36
## logDose         9.254889  1.2000095  7.712346 2.302639e-10
## suppVC         -3.700000  0.9604605 -3.852319 3.033467e-04
## logDose:suppVC  3.844782  1.6970696  2.265542 2.736578e-02
```

For a warm-up, we will calculate the y-intercepts of both groups and the slopes of both. For the OJ group, this is just the 1st and 2nd coefficients, while for the VC group it is $\beta_0 + \beta_2$ and $\beta_1 + \beta_3$.

```
# Add a heading so that I know which parameter is which!
#                               Int  logDose  suppVC   logDose:suppVC
contr <- rbind("Intercept OJ" = c(1,     0,     0,          0          ),
               "Slope     OJ" = c(0,     1,     0,          0          ),
               "Intercept VC" = c(1,     0,     1,          0          ),
               "Slope     VC" = c(0,     1,     0,          1          ) )
test <- glht(m, linfct=contr)
summary(test)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = len ~ logDose * supp, data = ToothGrowth)
##
## Linear Hypotheses:
##                     Estimate Std. Error t value Pr(>|t|)
## Intercept OJ == 0  20.6633     0.6791  30.425   <1e-09 ***
## Slope     OJ == 0   9.2549     1.2000   7.712   <1e-09 ***
```

```
## Intercept VC == 0  16.9633     0.6791  24.977   <1e-09 ***
## Slope     VC == 0  13.0997     1.2000  10.916   <1e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

In our original table of summary coefficients, the (intercept) term corresponds to the tooth growth of a guinea pig when the logDose level is 0 (and therefore dose=1). If I wanted to estimate the tooth growth of a guinea pig fed OJ but with only 1/2 a dose (and therefore $logDose = \log(1/2) = -0.69$), then we want

$$\begin{aligned}\hat{y}_{oj,dose=0.5} &= \hat{\beta}_0 + \hat{\beta}_1 \log(0.5) \\ &= 20.6 + 9.25 \log(0.5) \\ &= 20.6 + 9.25 (-0.69) \\ &= 14.21\end{aligned}$$

which I can write as

$$y_{oj,dose=0.5} = \left[\begin{array}{cccc} 1 & -0.69 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{array}\right] = \boldsymbol{c}_1^T \hat{\boldsymbol{\beta}}$$

To calculate the same value for the VC group, we need the following contrast:

$$\begin{aligned}\hat{y}_{vc,dose=0.5} &= 16.9633 + 13.0997 \log(0.5) \\ &= (20.66 - 3.7) + (9.25 + 3.8) \log(0.5) \\ &= \left(\hat{\beta}_0 + \hat{\beta}_2\right) + \left(\hat{\beta}_1 + \hat{\beta}_3\right) (-0.69) \\ &= \left[\begin{array}{cccc} 1 & -0.69 & 1 & -0.69 \end{array}\right] \left[\begin{array}{c} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{array}\right] \\ &= \boldsymbol{c}_2^T \hat{\boldsymbol{\beta}}\end{aligned}$$

```
#                                 Int  logDose  suppVC   logDose:suppVC
contr <- rbind("OJ; 1/2 dose" = c(1,   -0.69,    0,          0         ),
               "VC; 1/2 dose" = c(1,   -0.69,    1,        -0.69       ))
test <- glht(m, linfct=contr)
summary(test)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = len ~ logDose * supp, data = ToothGrowth)
##
## Linear Hypotheses:
##                     Estimate Std. Error t value Pr(>|t|)
## OJ; 1/2 dose == 0   14.277      1.071   13.33  < 1e-10 ***
## VC; 1/2 dose == 0    7.925      1.071    7.40 1.51e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Finally we might be interested in testing if there is a treatment difference between OJ and VC at a 1/2 dose level. So to do this, we want to calculate the difference between these two previous contrasts, i.e.

$$\begin{aligned} \boldsymbol{c}_1^T\hat{\boldsymbol{\beta}} - \boldsymbol{c}_2^T\hat{\boldsymbol{\beta}} &= \left(\boldsymbol{c}_1^T - \boldsymbol{c}_2^T\right)\hat{\boldsymbol{\beta}} \\ &= \left(\begin{bmatrix} 1 & -0.69 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & -0.69 & 1 & -0.69 \end{bmatrix}\right)\hat{\boldsymbol{\beta}} \\ &= \begin{bmatrix} 0 & 0 & -1 & 0.69 \end{bmatrix}\hat{\boldsymbol{\beta}} \end{aligned}$$

and we can calculate

```
contr <- rbind("OJ - VC; 1/2 dose" = c(0, 0, -1, 0.69))
test <- glht(m, linfct=contr)
summary(test)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = len ~ logDose * supp, data = ToothGrowth)
##
## Linear Hypotheses:
##                         Estimate Std. Error t value Pr(>|t|)
## OJ - VC; 1/2 dose == 0    6.353      1.514   4.195 9.83e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

When we do the same test, but at dose level 2 (logDose $=\log 2 = 0.69$) we see that the difference is not statistically significant.

```
contr <- rbind("OJ - VC; dose 2" = c(0, 0, -1, -0.69))
test <- glht(m, linfct=contr)
summary(test)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = len ~ logDose * supp, data = ToothGrowth)
##
## Linear Hypotheses:
##                       Estimate Std. Error t value Pr(>|t|)
## OJ - VC; dose 2 == 0    1.047      1.514   0.691    0.492
## (Adjusted p values reported -- single-step method)
```

5.3 Using `emmeans` Package

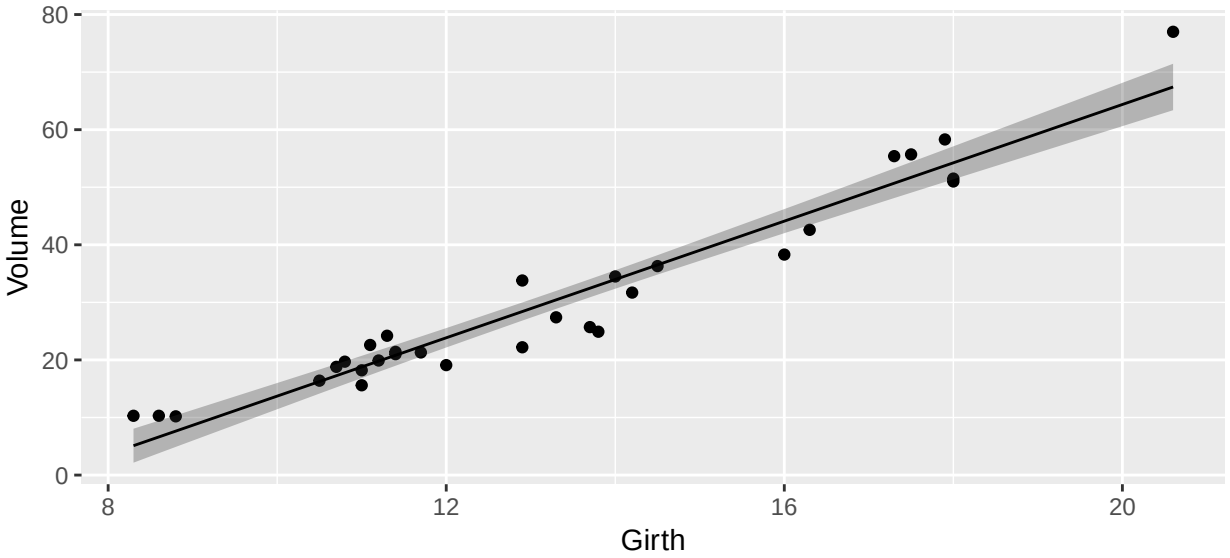
Specifying the contrasts by hand is extremely difficult to do correctly and instead we would prefer to specify the contrasts using language like "create all possible pairwise contrasts" where each pair is just a subtraction. The R-package `emmeans` tries to simply the creation of common contrasts.

To show how to use the `emmeans` package, we'll consider a bunch of common models and show how to address common statistical questions for each.

5.3.1 Simple Regression

There is a dataset built into R named `trees` which describes a set of $n = 31$ cherry trees and the goal is to predict the volume of timber produced by each tree just using the tree girth a 4.5 feet above the ground.

```
data(trees)
model <- lm( Volume ~ Girth, data=trees )
trees <- trees %>%
  dplyr::do(if(!is.null(.$fit)){dplyr::select(-fit,-upr,-lwr )}else{.}) %>%
  cbind( predict(model, interval='conf'))
ggplot(trees, aes(x=Girth, y=Volume)) +
  geom_point() +
  geom_ribbon( aes(ymin=lwr, ymax=upr), alpha=.3 ) +
  geom_line( aes(y=fit) )
```



Using the `summary()` function, we can test hypotheses about if the y-intercept or slope could be equal to zero, but we might be interested in confidence intervals for the regression line at girth values of 10 and 12.

```
# We could find the regression line heights and CI using
# either predict() or emmeans()
predict(model, newdata=data.frame(Girth=c(10,12)), interval='conf' )
```

```
##        fit      lwr      upr
## 1 13.71511 11.44781 15.98240
## 2 23.84682 22.16204 25.53159
```

```
emmeans(model, specs = ~Girth, at=list(Girth=c(10,12)) )
```

```
##  Girth   emmean        SE df lower.CL upper.CL
##     10 13.71511 1.1085761 29 11.44781 15.98240
##     12 23.84682 0.8237582 29 22.16204 25.53159
##
## Confidence level used: 0.95
```

The `emmeans()` function requires us to specify the grid of reference points we are interested as well as which variable or variables we wish to separate out. In the simple regression case, the `specs` argument is just the single covariate.

We might next ask if the difference in volume between a tree with 10 inch girth is statistically different than

a tree with 12 inch girth? In other words, we want to test

$$H_0 : (\beta_0 + \beta_1 \cdot 10) - (\beta_0 + \beta_1 \cdot 12) = 0$$

$$H_a : (\beta_0 + \beta_1 \cdot 10) - (\beta_0 + \beta_1 \cdot 12) \neq 0$$

In this case, we want to look at all possible pairwise differences between the predicted values at 10 and 12.

```
emmeans(model, specs = pairwise~Girth,
        at=list(Girth=c(10,12)) )
```

```
## $emmeans
##  Girth   emmean        SE df lower.CL upper.CL
##     10 13.71511 1.1085761 29 11.44781 15.98240
##     12 23.84682 0.8237582 29 22.16204 25.53159
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast  estimate        SE df t.ratio p.value
##  10 - 12  -10.13171 0.4947539 29 -20.478  <.0001
```

Notice that if I was interested in 3 points, we would get all of the differences.

```
emmeans(model, specs = pairwise~Girth,
        at=list(Girth=c(10,11,12)) )
```

```
## $emmeans
##  Girth   emmean        SE df lower.CL upper.CL
##     10 13.71511 1.1085761 29 11.44781 15.98240
##     11 18.78096 0.9447560 29 16.84872 20.71320
##     12 23.84682 0.8237582 29 22.16204 25.53159
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast   estimate        SE df t.ratio p.value
##  10 - 11   -5.065856 0.2473770 29 -20.478  <.0001
##  10 - 12  -10.131713 0.4947539 29 -20.478  <.0001
##  11 - 12   -5.065856 0.2473770 29 -20.478  <.0001
##
## P value adjustment: tukey method for comparing a family of 3 estimates
```

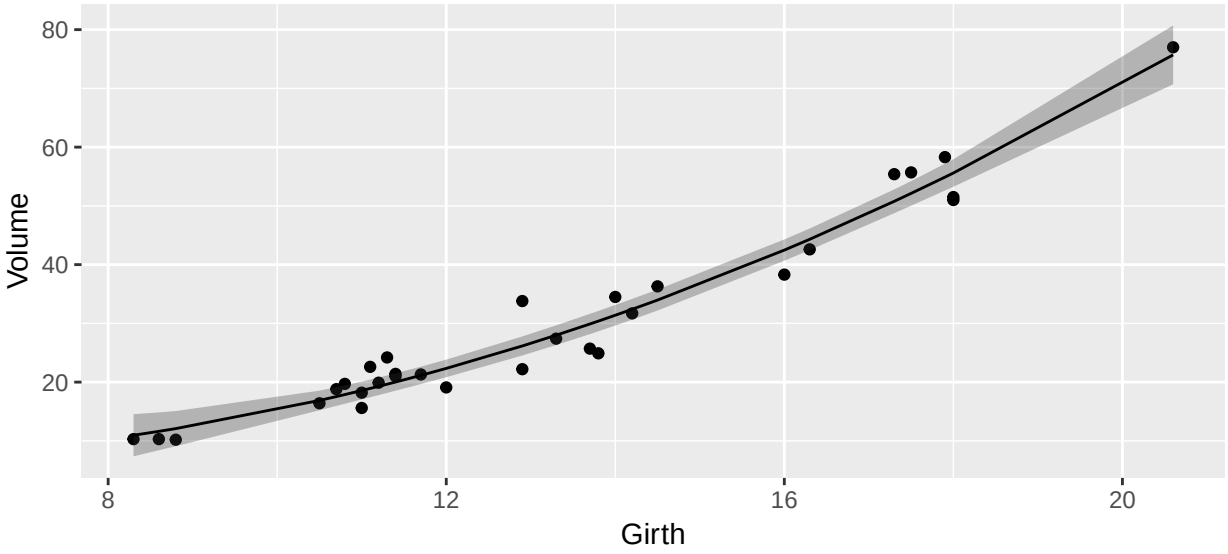
In this very simple case, the slope parameter is easily available as a parameter value, but we could use the `emtrends()` function to obtain the slope.

```
emtrends( model, ~Girth, 'Girth' )
```

```
##     Girth Girth.trend       SE df lower.CL upper.CL
##  13.24839    5.065856 0.247377 29 4.559914 5.571799
##
## Confidence level used: 0.95
```

This output is a bit mysterious because because of the `13.248` component. What has happened is that `emtrends` is telling us the slope of the line at a particular point on the x-axis (the mean of all the girth values). While this doesn't matter in this example, because the slope is the same for all values of girth, if we had fit a quadratic model, it would not.

```
model <- lm( Volume ~ poly(Girth, 2), data=trees )      # Girth + Girth^2
trees <- trees %>%
  dplyr::select(Volume, Girth) %>%
  cbind(predict(model, interval='conf'))

ggplot(trees, aes(x=Girth, y=Volume)) +
  geom_point() +
  geom_ribbon( aes(ymin=lwr, ymax=upr), alpha=.3 ) +
  geom_line( aes(y=fit) )
```




```
emtrends( model, ~ poly(Girth,2), 'Girth',
          at=list(Girth=c(10,11,12)) )
```

```
##  Girth Girth.trend        SE df lower.CL upper.CL
##     10    3.029920 0.5041192 28 1.997278 4.062561
##     11    3.538995 0.3992560 28 2.721156 4.356834
##     12    4.048070 0.3028961 28 3.427616 4.668525
##
## Confidence level used: 0.95
```

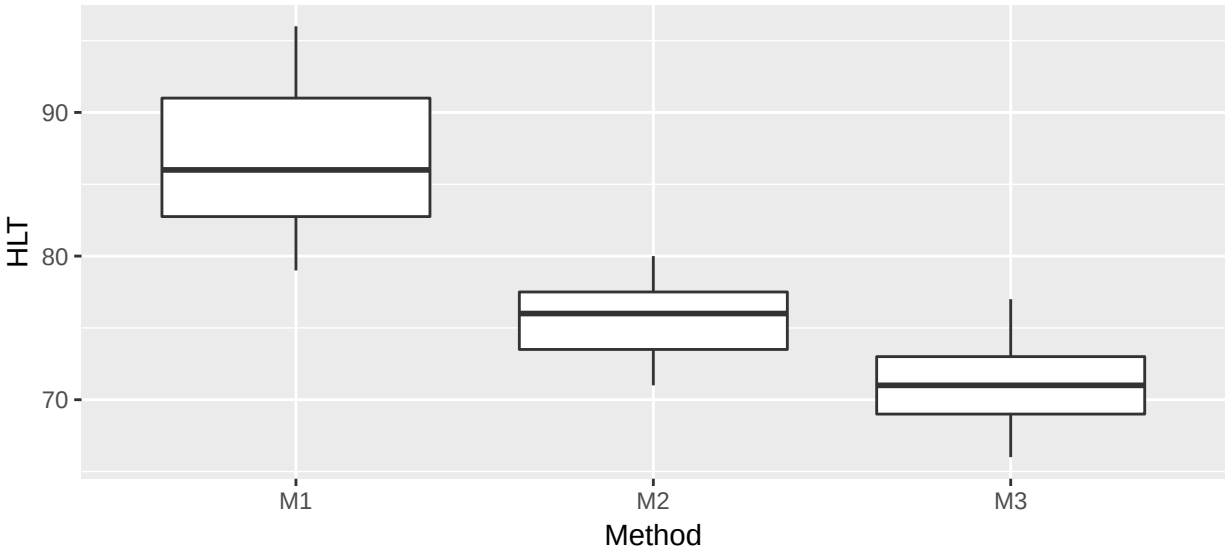
5.3.2 1-way ANOVA

To consider the pairwise contrasts between different levels we will consider the college student hostility data again. A clinical psychologist wished to compare three methods for reducing hostility levels in university students, and used a certain test (HLT) to measure the degree of hostility. A high score on the test indicated great hostility. The psychologist used 24 students who obtained high and nearly equal scores in the experiment. Eight subjects were selected at random from among the 24 problem cases and were treated with method 1, seven of the remaining 16 students were selected at random and treated with method 2 while the remaining nine students were treated with method 3. All treatments were continued for a one-semester period. Each student was given the HLT test at the end of the semester, with the results show in the following table.

```
Hostility <- data.frame(
  HLT = c(96,79,91,85,83,91,82,87,
          77,76,74,73,78,71,80,
          66,73,69,66,77,73,71,70,74),
```

```
  Method = c( rep('M1',8), rep('M2',7), rep('M3',9) ) )
```

```
ggplot(Hostility, aes(x=Method, y=HLT)) +
  geom_boxplot()
```



To use the `emmeans()`, we again will use the pairwise command where we specify that we want all the pairwise contrasts between Method levels.

```
model <- lm( HLT ~ Method, data=Hostility )
emmeans(model, pairwise~Method)
```

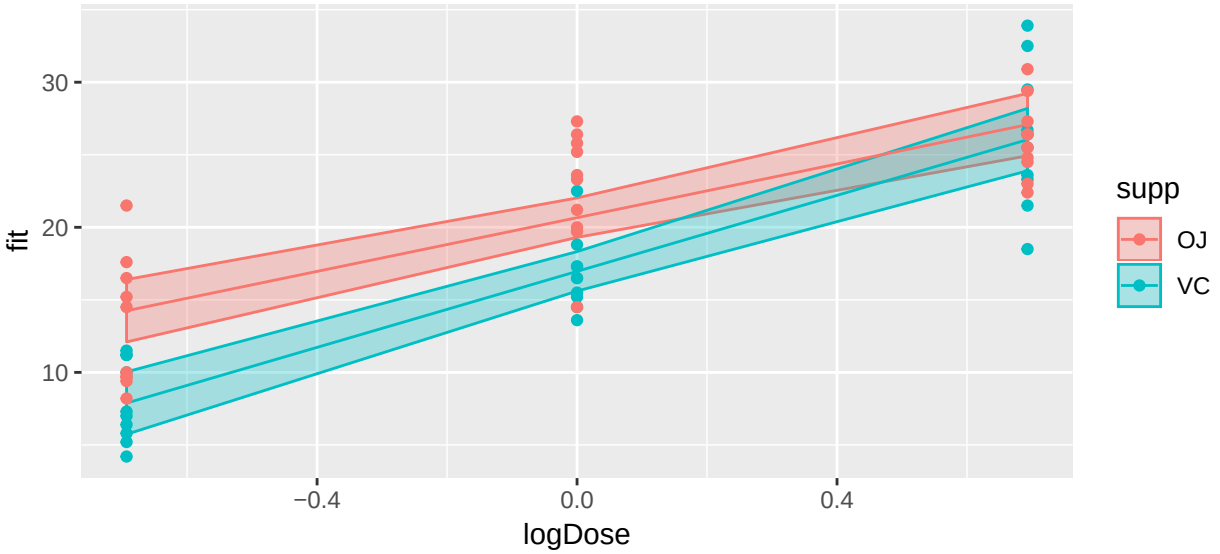
```
## $emmeans
##  Method   emmean       SE df lower.CL upper.CL
##  M1     86.75000 1.518172 21 83.59279 89.90721
##  M2     75.57143 1.622994 21 72.19623 78.94663
##  M3     71.00000 1.431347 21 68.02335 73.97665
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast  estimate       SE df t.ratio p.value
##  M1 - M2  11.178571 2.222377 21   5.030  0.0002
##  M1 - M3  15.750000 2.086528 21   7.548  <.0001
##  M2 - M3   4.571429 2.163993 21   2.112  0.1114
##
## P value adjustment: tukey method for comparing a family of 3 estimates
```

5.3.3 ANCOVA

We will return to the `ToothGrowth` dataset and calculate all the interesting contrasts we previously built by hand.

```
data('ToothGrowth')
ToothGrowth$logDose <- log( ToothGrowth$dose )
m <- lm(len ~ logDose * supp, data=ToothGrowth)
ToothGrowth <- ToothGrowth %>%
  dplyr::do(if(!is.null(.$fit)){dplyr::select(-fit,-upr,-lwr )}else{.}) %>%
```

```
  cbind( predict(m, interval='conf') )
ggplot(ToothGrowth, aes(x=logDose, color=supp, fill=supp)) +
  geom_ribbon( aes(ymin=lwr, ymax=upr), alpha=.3 ) +
  geom_line( aes(y=fit) ) +
  geom_point( aes(y=len) )
```



First, lets get the y-intercept for each group. In the following code, the formula with the `| supp` part says that we want to do all the calculations for each level of the supplement types. The call to `emmeans()` calculates the response at logDose level 0. By passing the result of the `emmeans()` function call to the `summary()` function, we can tell the summary function to print out both the confidence intervals and t-tests.

```
emmeans(m, ~logDose | supp, at=list(logDose=0)) %>%
  summary(infer=c(TRUE,TRUE))
```

```
## supp = OJ:
##  logDose   emmean        SE df lower.CL upper.CL t.ratio p.value
##        0 20.66333 0.6791481 56 19.30284 22.02383  30.425  <.0001
##
## supp = VC:
##  logDose   emmean        SE df lower.CL upper.CL t.ratio p.value
##        0 16.96333 0.6791481 56 15.60284 18.32383  24.977  <.0001
##
## Confidence level used: 0.95
```

Next we estimated the tooth growth for a guinea pig with a logdose = -.69 (i.e. dose=1/2) for both the OJ and VC supplements. We also wanted to calculate the contrast between the two supplement predictions.

```
emmeans(m, pairwise~logDose*supp, at=list(logDose=-.69)) %>%
  summary(infer=c(TRUE,TRUE))
```

```
## $emmeans
##  logDose supp   emmean       SE df lower.CL upper.CL t.ratio p.value
##    -0.69 OJ   14.27746 1.070905 56 12.13218 16.42274  13.332  <.0001
##    -0.69 VC    7.92456 1.070905 56  5.77928 10.06984   7.400  <.0001
##
## Confidence level used: 0.95
##
## $contrasts
```

```
##  contrast            estimate       SE df lower.CL upper.CL t.ratio
##  -0.69,OJ - -0.69,VC   6.3529 1.514488 56 3.319016 9.386784   4.195
##  p.value
##   0.0001
##
## Confidence level used: 0.95
```

We next make the same comparison for logdose = 0.69 (i.e. dose=2).

```
emmeans(m, pairwise~logDose*supp, at=list(logDose=.69)) %>%
  summary(infer=c(TRUE,TRUE))
```

```
## $emmeans
##  logDose supp   emmean       SE df lower.CL upper.CL t.ratio p.value
##     0.69 OJ   27.04921 1.070905 56 24.90393 29.19449  25.258  <.0001
##     0.69 VC   26.00211 1.070905 56 23.85683 28.14739  24.281  <.0001
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast          estimate       SE df  lower.CL upper.CL t.ratio p.value
##  0.69,OJ - 0.69,VC   1.0471 1.514488 56 -1.986784 4.080984   0.691  0.4922
##
## Confidence level used: 0.95
```

While `emmeans()` makes it very easy to calculate the $\hat{y}$ value for some covariate combination or some difference of $\hat{y}$ values, it doesn't handle slope parameters particularly well. The companion function `emtrends()` is intended to make similar calculations among slopes easy to create. We will now assess the difference in slopes between the two groups.

```
emtrends(m, ~logDose*supp, var = 'logDose')
```

```
##  logDose supp logDose.trend       SE df  lower.CL upper.CL
##        0 OJ        9.254889 1.200009 56  6.850981 11.65880
##        0 VC       13.099671 1.200009 56 10.695763 15.50358
##
## Confidence level used: 0.95
```

```
emtrends(m, pairwise~logDose*supp, var = 'logDose')
```

```
## $emtrends
##  logDose supp logDose.trend       SE df  lower.CL upper.CL
##        0 OJ        9.254889 1.200009 56  6.850981 11.65880
##        0 VC       13.099671 1.200009 56 10.695763 15.50358
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast     estimate      SE df t.ratio p.value
##  0,OJ - 0,VC -3.844782 1.69707 56  -2.266  0.0274
```

5.3.3.1 Common `emmeans()` mistake

It is very common to make a mistake using `emmeans()` or `emtrends()` if you don't understand what the specification formula means. In particular what happens if you were to use `~logDose` instead of the correct `~logDose*supp`.

Consider the case where we are interested in making predictions for each supplement type at 1/2 dose (logDose=-0.69).

```
emmeans(m, ~logDose*supp, at=list(logDose=-.69))
```

```
##  logDose supp   emmean       SE df lower.CL upper.CL
##    -0.69 OJ   14.27746 1.070905 56 12.13218 16.42274
##    -0.69 VC    7.92456 1.070905 56  5.77928 10.06984
##
## Confidence level used: 0.95
```

If I were to accidentally forget the `supp` part of this, then `emmeans()` would figure that I wanted to *average* over the two levels and give me the average of 14.28 and 7.92, which is 11.1. With the interaction term in the model, it is unlikely that I would want to do this, but `emmeans()` complains a little but lets me do it.

```
emmeans(m, ~logDose, at=list(logDose=-.69))
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
##  logDose   emmean       SE df lower.CL upper.CL
##    -0.69 11.10101 0.757244 56 9.584068 12.61795
##
## Results are averaged over the levels of: supp
## Confidence level used: 0.95
```

What would happen if we only use `supp` in the formula specification? In this case we are lucky because the `logDose=-0.69` specified what dose level to look at. If we had forgotten to specify this, then we would have done this calculation at the mean `logDose` value, which happens to be zero.

```
emmeans(m, ~supp*logDose, at=list(logDose=0))
```

```
##  supp logDose   emmean        SE df lower.CL upper.CL
##  OJ         0 20.66333 0.6791481 56 19.30284 22.02383
##  VC         0 16.96333 0.6791481 56 15.60284 18.32383
##
## Confidence level used: 0.95
```

```
emmeans(m, ~supp)
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
##  supp   emmean        SE df lower.CL upper.CL
##  OJ   20.66333 0.6791481 56 19.30284 22.02383
##  VC   16.96333 0.6791481 56 15.60284 18.32383
##
## Confidence level used: 0.95
```

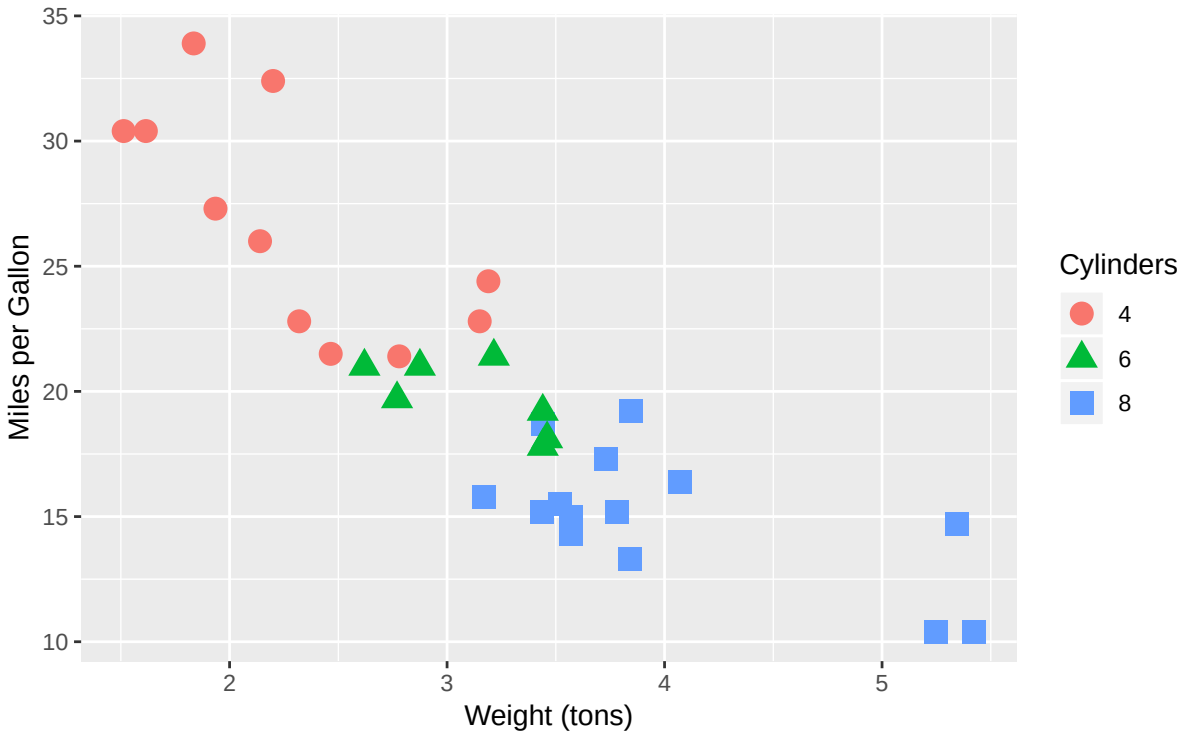
Because the averaging over factor levels isn't what I typically want to do, I start the analysis making sure the model formula I used in the `lm()` command is the same as the one I pass into `emmeans()`.

5.4 Exercises

1. We will examine a dataset that summarizes cars featured in the magazine Motor Trend during the year 1974. In particular we will examine the relationship between the vehicles gas mileage (`mpg`) versus the weight (`wt`) and number of cylinders (`cyl`) of the vehicle.

   ```
   library(ggplot2)
   library(dplyr)
   ```

```
data(mtcars)
mtcars$cyl <- factor(mtcars$cyl)      # treat cylinders as categorical
ggplot(mtcars, aes(x=wt, col=cyl)) +
  geom_point(aes(y=mpg, shape=cyl), size=4) +
  xlab('Weight (tons)') +
  ylab('Miles per Gallon') +
  labs(color='Cylinders', shape='Cylinders')
```



a. Create a smaller set of data with just 9 observations (3 from each cylinder group) and create the design matrix $\boldsymbol{X}$ for fitting the the model for estimating mpg using both weight and number of cylinders and their interaction using the following code:

```
mtcars %>%
  group_by(cyl) %>%              # For each cylinder group
  arrange(cyl, wt) %>%           # Reorder my dataset
  slice(1:3)  %>%                # grab first three rows of each group
  lm( mpg ~ wt*cyl, data=. ) %>% # Linear model
  model.matrix()
```

```
##   (Intercept)    wt cyl6 cyl8 wt:cyl6 wt:cyl8
## 1           1 1.513    0    0   0.000   0.000
## 2           1 1.615    0    0   0.000   0.000
## 3           1 1.835    0    0   0.000   0.000
## 4           1 2.620    1    0   2.620   0.000
## 5           1 2.770    1    0   2.770   0.000
## 6           1 2.875    1    0   2.875   0.000
## 7           1 3.170    0    1   0.000   3.170
## 8           1 3.435    0    1   0.000   3.435
## 9           1 3.440    0    1   0.000   3.440
## attr(,"assign")
## [1] 0 1 2 2 3 3
```

```
## attr(,"contrasts")
## attr(,"contrasts")$cyl
## [1] "contr.treatment"
```

Hint: the purpose of this step is to make sure everyone knows the column order of the design matrix so that you interprete the β terms in the correct order.

b. Denote the coefficients obtained from the summary of your `lm()` call as $\hat{\beta}_0$ to $\hat{\beta}_5$ (in the order given by R). Write your interaction model out using subscript notation and should be in the form

$$y_i = \begin{cases} ??????? & \text{if 4 cylinder} \\ ??????? & \text{if 6 cylinder} \\ ??????? & \text{if 8 cylinder} \end{cases}$$

and give an interpretation for each β_j value.

i. $\hat{\beta}_0 = ???$
ii. $\hat{\beta}_1 = ???$
iii. $\hat{\beta}_2 = ???$
iv. $\hat{\beta}_3 = ???$
v. $\hat{\beta}_4 = ???$
vi. $\hat{\beta}_5 = ???$

c. Using the full dataset, fit the model that predicts mpg using both weight and number of cylinders and their interaction using the command

```
model <- lm(mpg ~ wt*cyl, data=mtcars)
```

Create a vector of fitted values, add it to the data frame `mtcars` and then create a plot that includes the regression lines.

d. What is the estimated mpg of a 6-cylinder vehicle weighing 3.5 tons? What is the estimated mpg of a 8-cylinder vehicle weight 3.5 tons? Give the associated 95% confidence intervals. Calculate these using the `predict()` function. *Warning: When making the new data frame, make sure that R interprets cyl as a factor by either coercing it to a factor after creation or inputing the cylinder as a string (i.e. as "6" instead of 6).*

e. Recalculate your answer to part (d) using the `glht()` function. Also give the estimate and confidence interval for the difference in mpg betwen the 8 and 6 cylinder vehicles that weight 3.5 tones? Is there a statistically significant difference in mpg at 3.5 tons?

f. Recalculate your answer to part (e) using the `emmeans()` function.

g. Are the slopes of the 8-cylinder and 6-cylinder vehicles statistically different? Use `glht()` to produce this result.

h. Recalculate your answer to part (g) using the `emtrends()` function.

Chapter 6

Diagnostics and Transformations

```
library(tidyverse)    # for dplyr, tidyr, ggplot2
library(ggfortify)    # for autoplot for lm objects
library(emmeans)      # emmeans for pairwise constrasts.
```

We will be interested in analyzing whether or not our linear model is a good model and whether or not the data violate any of the assumptions that are required. In general we will be interested in three classes of assumption violations and our diagnostic measures might be able detect one or more of the following issues:

1. Unusual observations that contribute too much influence to the analysis. These few observations might drastically change the outcome of the model.
2. Model misspecification. Our assumption that $E[\boldsymbol{y}] = \boldsymbol{X}\boldsymbol{\beta}$ might be wrong and we might need to include different covariates in the model to get a satisfactory result.
3. Error distribution. We have assumed that $\epsilon_i \overset{iid}{\sim} N\left(0, \sigma^2\right)$ but autocorrelation, heteroscedasticity, and non-normality might be present.

Often problems with one of these can be corrected by transforming either the explanatory or response variables.

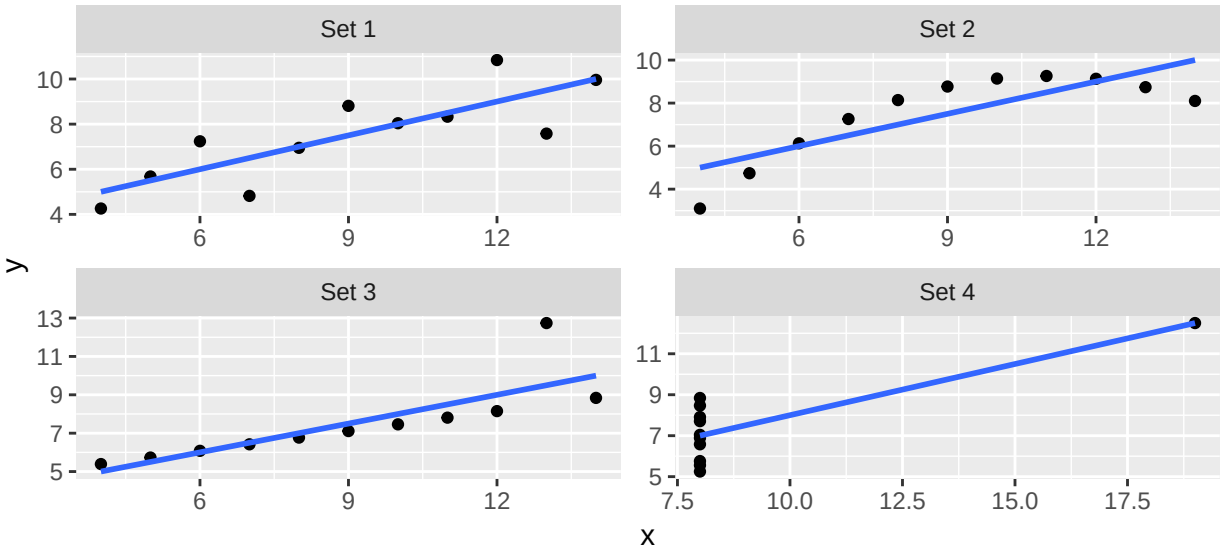
6.1 Detecting Assumption Violations

Throughout this chapter I will use data created by Francis Anscombe that show how simple linear regression can be misused. In particular, these data sets will show how our diagnostic measures will detect various departures from the model assumptions.

The data are available in R as a data frame `anscombe` and is loaded by default. The data consists of four datasets, each having the same linear regression $\hat{y} = 3 + 0.5\,x$ but the data are drastically different.

```
# The anscombe dataset has 8 columns - x1,x2,x3,x4,y1,y2,y3,y4
# and I want it to have 3 columns - Set, X, Y
Anscombe <- rbind(
  data.frame(x=anscombe$x1, y=anscombe$y1, set='Set 1'),
  data.frame(x=anscombe$x2, y=anscombe$y2, set='Set 2'),
  data.frame(x=anscombe$x3, y=anscombe$y3, set='Set 3'),
  data.frame(x=anscombe$x4, y=anscombe$y4, set='Set 4'))

# order them by their x values, and add an index column
Anscombe <- Anscombe %>%
```

```
  group_by(set) %>%       # Every subsequent action happens by dataset
  arrange(x,y) %>%        # sort them on the x-values and if tied, by y-value
  mutate( index = 1:n() ) # give each observation within a set, an ID number

# Make a nice graph
ggplot(Anscombe, aes(x=x, y=y)) +
 geom_point() +
 facet_wrap(~set, scales='free') +
 stat_smooth(method="lm", formula=y~x, se=FALSE)
```



6.1.1 Measures of Influence

6.1.1.1 Standardized Residuals (aka Studentized)

Recall that we have

$$\begin{aligned}\hat{\boldsymbol{y}} &= \boldsymbol{X}\hat{\boldsymbol{\beta}} \\ &= \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \\ &= \boldsymbol{H}\boldsymbol{y}\end{aligned}$$

where the "Hat Matrix" is $\boldsymbol{H} = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T$ because we have $\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$. The elements of $\boldsymbol{H}$ can be quite useful in diagnostics. It can be shown that the variance of the $i$the residual is

$$Var\left(\hat{\epsilon}_i\right) = \sigma^2\left(1 - \boldsymbol{H}_{ii}\right)$$

where $\boldsymbol{H}_{ii}$ is the ith element of the main diagonal of $\boldsymbol{H}$. This suggests that I could rescale my residuals to

$$\hat{\epsilon}_i^* = \frac{\hat{\epsilon}_i}{\hat{\sigma}\sqrt{1 - \boldsymbol{H}_{ii}}}$$

which, if the normality and homoscedasticity assumptions hold, should behave as a $N\left(0, 1\right)$ sample.

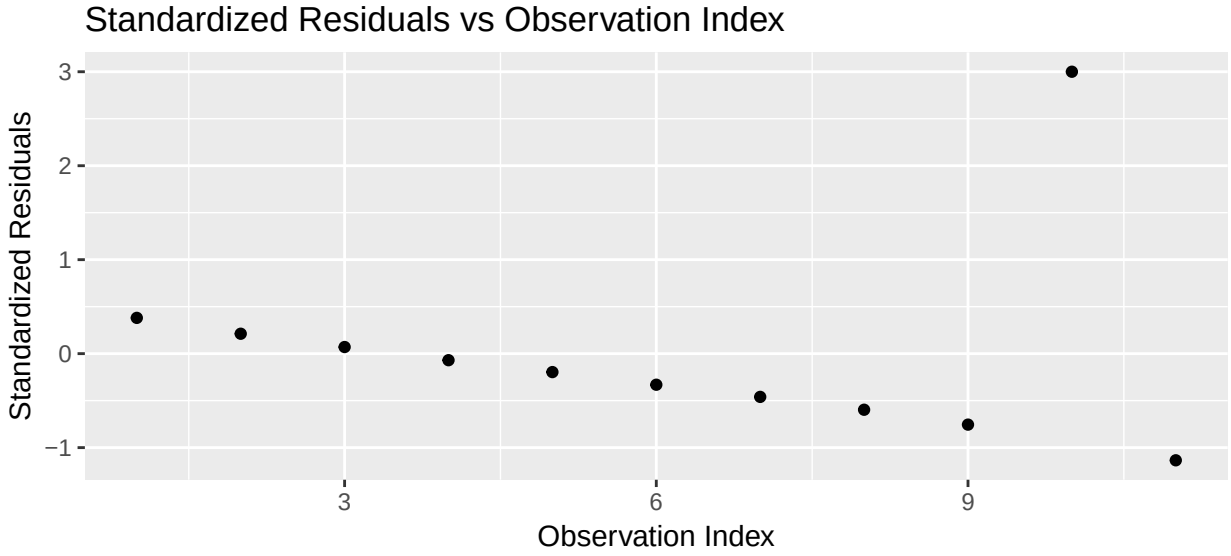
These rescaled residuals are called "studentized residuals", though R typically refers to them as "standardized". Since we have a good intuition about the scale of a standard normal distribution, the scale of standardized residuals will give a good indicator if normality is violated.

There are actually two types of studentized residuals, typically called *internal* and *external* among statisticians. The version presented above is the *internal* version which can be obtained using the R function `rstandard()` while the *external* version is available using `rstudent()`. Whenever you see R present standardized residuals, they are talking about internally studentized residuals. For sake of clarity, I will use the term *standardized* as well.

6.1.1.1.1 Example - Anscombe's set 3

For the third dataset, the outlier is the ninth observation with $x_9 = 13$ and $y_9 = 12.74$. We calculate the standardized residuals using the function `rstandard()` and plot them

```
Set3 <- Anscombe %>% filter(set == 'Set 3') # Just set 3
model <- lm(y ~ x, data=Set3)               # Fit the regression line
Set3$stdresid <- rstandard(model)           # rstandard() returns the standardized residuals

ggplot(Set3, aes(x=index, y=stdresid)) +    # make a plot
  geom_point() +
  labs(x='Observation Index',
       y='Standardized Residuals',
       title='Standardized Residuals vs Observation Index')
```



and we notice that the outlier residual is really big. If the model assumptions were true, then the standardized residuals should follow a standard normal distribution, and I would need to have hundreds of observations before I wouldn't be surprised to see a residual more than 3 standard deviations from 0.

6.1.1.2 Leverage

The extremely large standardized residual suggests that this data point is important, but we would like to quantify how important this observation actually is.

One way to quantify this is to look at the elements of $\boldsymbol{H}$. Because

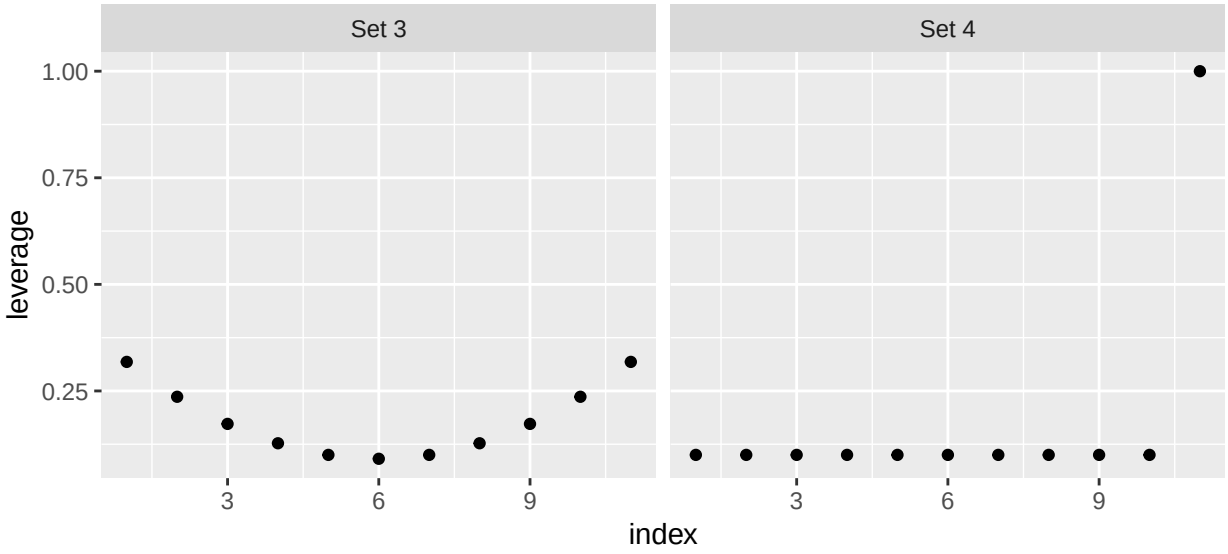
$$\hat{y}_i = \sum_{j=1}^{n} \boldsymbol{H}_{ij} y_j$$

then the ith row of $\boldsymbol{H}$ is a vector of weights that tell us how influential a point y_j is for calculating the predicted value $\hat{y}_i$. If I look at just the main diagonal of $\boldsymbol{H}$, these are how much weight a point has on its

predicted value. As such, I can think of the $\boldsymbol{H}_{ii}$ as the amount of leverage a particular data point has on the regression line. It can be shown that the leverages must be $0 \leq \boldsymbol{H}_{ii} \leq 1$ and that $\sum \boldsymbol{H}_{ii} = p$.

Fortunately there is already a function `hatvalues()` to compute these $\boldsymbol{H}_{ii}$ values for me. We will compare the leverages from Anscombe's set 3 versus set 4.

```
Set3 <- Anscombe %>% filter( set == 'Set 3')
Set4 <- Anscombe %>% filter( set == 'Set 4')

model3 <- lm(y ~ x, data = Set3 )
model4 <- lm(y ~ x, data = Set4 )

Set3 <- Set3 %>% mutate(leverage = hatvalues(model3))  # add leverage columns
Set4 <- Set4 %>% mutate(leverage = hatvalues(model4))

ggplot( rbind(Set3,Set4), aes(x=index, y=leverage) ) +
  geom_point() +
  facet_grid( . ~ set )
```



This leverage idea only picks out the *potential* for a specific value of x to be influential, but does not actually measure influence. It has picked out the issue with the fourth data set, but does not adequately address the outlier in set 3.

6.1.1.3 Cook's Distance

To attempt to measure the actual influence of an observation $\left\{y_i, \boldsymbol{x}_i^T\right\}$ on the linear model, we consider the effect on the regression if we removed the observation and fit the same model. Let

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$$

be the vector of predicted values, where $\hat{\boldsymbol{\beta}}$ is created using all of the data, and $\hat{\boldsymbol{y}}_{(i)} = \boldsymbol{X}\hat{\boldsymbol{\beta}}_{(i)}$ be the vector of predicted values where $\hat{\boldsymbol{\beta}}_{(i)}$ was estimated using all of the data except the ith observation. Letting p be the number of β_j parameters as usual we define Cook's distance of the ith observation as

$$D_i = \frac{\left(\hat{\boldsymbol{y}} - \hat{\boldsymbol{y}}_{(i)}\right)^T \left(\hat{\boldsymbol{y}} - \hat{\boldsymbol{y}}_{(i)}\right)}{p\hat{\sigma}^2}$$

which boils down to saying if the predicted values have large changes when the ith element is removed, then the distance is big. It can be shown that this formula can be simplified to

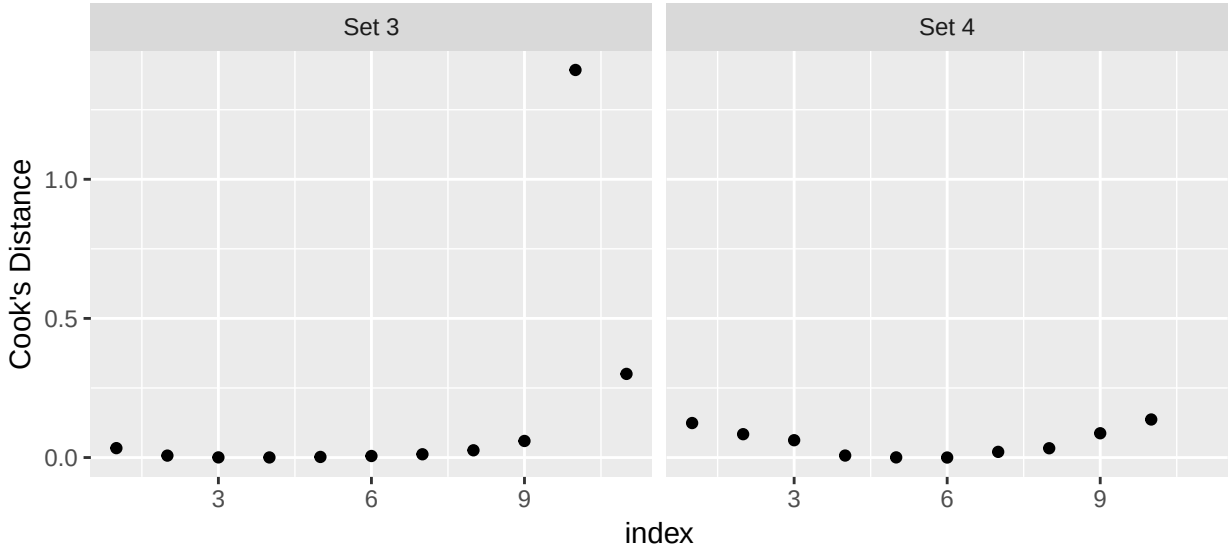
$$D_i = \frac{\hat{\epsilon}_i^* \boldsymbol{H}_{ii}}{p\,(1 - H_{ii})}$$

which expresses Cook's distance in terms of the ith studentized residual and the ith leverage.

Nicely, the R function `cooks.distance()` will calculate Cook's distance.

```
Set3 <- Set3 %>% mutate(cooksd = cooks.distance(model3))
Set4 <- Set4 %>% mutate(cooksd = cooks.distance(model4))

# Note: The high leverage point in set 4 has a Cook's distance of Infinity.
ggplot(rbind(Set3,Set4), aes(x=index, y=cooksd)) +
  geom_point() +
  facet_grid(. ~ set) +
  labs(y="Cook's Distance")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Some texts will give a rule of thumb that points with Cook's distances greater than 1 should be considered influential, while other books claim a reasonable rule of thumb is $4/(n-p-1)$ where n is the sample size, and p is the number of parameters in $\boldsymbol{\beta}$. My take on this, is that you should look for values that are highly different from the rest of your data.

6.1.2 Diagnostic Plots

After fitting a linear model in R, you have the option of looking at diagnostic plots that help to decide if any assumptions are being violated. We will step through each of the plots that are generated by the function `plot(model)` or using `ggplot2` using the package `ggfortify`.

In the package `ggfortify` there is a function that will calculate the diagnostics measures and add them to your dataset. This will simplify our graphing process.

```
Set1 <- Anscombe %>% filter(set == 'Set 1')
model <- lm( y ~ x, data=Set1)
Set1 <- fortify(model)  # add dignostic measures to the dataset
Set1 %>% round(digits=3) # show the dataset nicely
```

```
##        y  x  .hat .sigma .cooksd .fitted .resid .stdresid
## 1   4.26  4 0.318  1.273   0.123   5.000 -0.740    -0.725
## 2   5.68  5 0.236  1.310   0.004   5.501  0.179     0.166
## 3   7.24  6 0.173  1.220   0.127   6.001  1.239     1.102
## 4   4.82  7 0.127  1.147   0.154   6.501 -1.681    -1.455
## 5   6.95  8 0.100  1.311   0.000   7.001 -0.051    -0.043
## 6   8.81  9 0.091  1.218   0.062   7.501  1.309     1.110
## 7   8.04 10 0.100  1.312   0.000   8.001  0.039     0.033
## 8   8.33 11 0.127  1.310   0.002   8.501 -0.171    -0.148
## 9  10.84 12 0.173  1.100   0.279   9.001  1.839     1.635
## 10  7.58 13 0.236  1.056   0.489   9.501 -1.921    -1.778
## 11  9.96 14 0.318  1.311   0.000  10.001 -0.041    -0.041
```

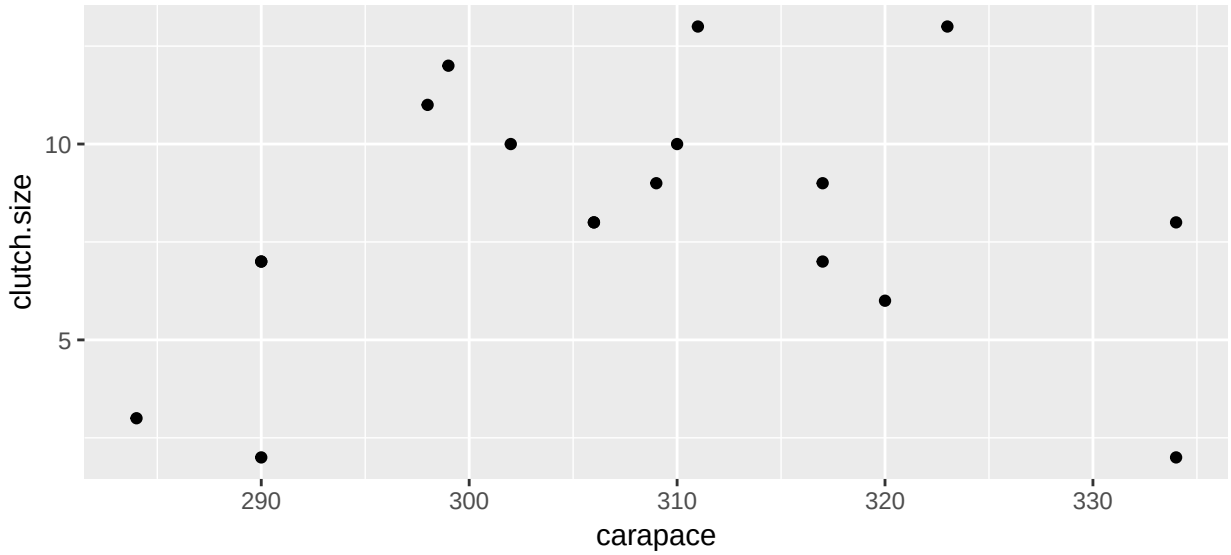
6.1.2.1 Residuals vs Fitted

In the simple linear regression the most useful plot to look at was the residuals versus the x-covariate, but we also saw that this was similar to looking at the residuals versus the fitted values. In the general linear model, we will look at the residuals versus the fitted values or possibly the studentized residuals versus the fitted values.

6.1.2.1.1 Polynomial relationships

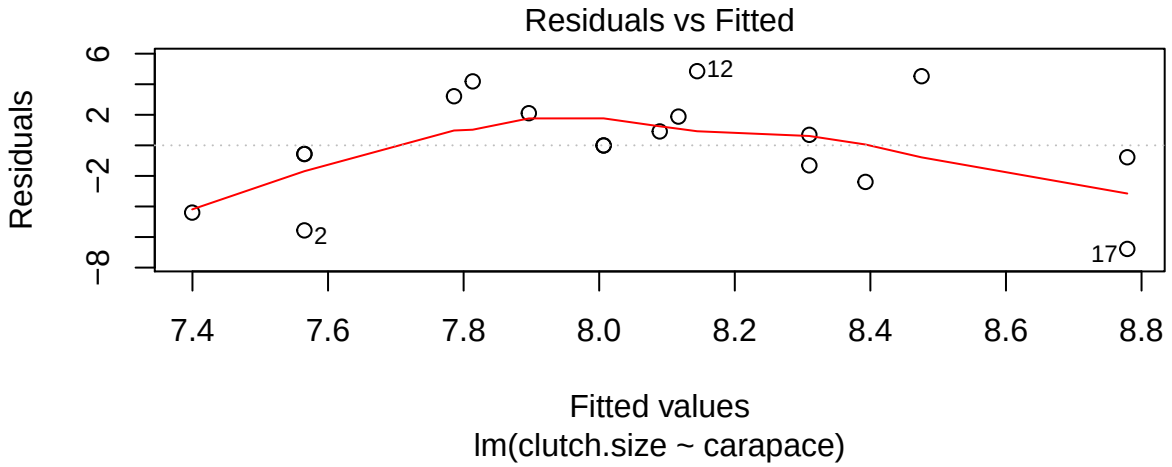
To explore how this plot can detect non-linear relationships between y and x, we will examine a data set from Ashton et al. (2007) that relates the length of a tortoise's carapace to the number of eggs laid in a clutch. The data are

```
Eggs <- data.frame(
  carapace    = c(284,290,290,290,298,299,302,306,306,
                  309,310,311,317,317,320,323,334,334),
  clutch.size = c(3,2,7,7,11,12,10,8,8,
                  9,10,13,7,9,6,13,2,8))
ggplot(Eggs, aes(x=carapace, y=clutch.size)) +
  geom_point()
```

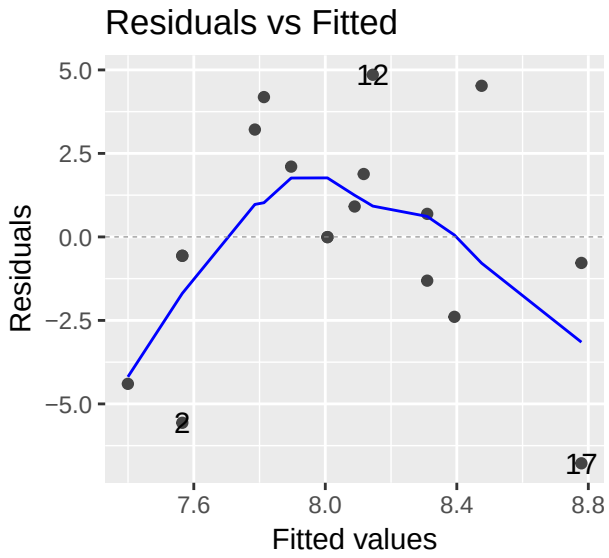


Looking at the data, it seems that we are violating the assumption that a linear model is appropriate, but we will fit the model anyway and look at the residual graph.

```
model <- lm( clutch.size ~ carapace, data=Eggs )
plot(model, which=1)       # which=1 tells R to only make the first plot
```




```
library(ggfortify)         # need the ggfortify library for autoplot.lm() to work
autoplot(model, which=1)   # same plot using ggplot2
```



The red/blue lines going through the plot is a smoother of the residuals. Ideally this should be a flat line and I should see no trend in this plot. Clearly there is a quadratic trend as larger tortoises have larger clutch sizes until some point where the extremely large tortoises start laying fewer (perhaps the extremely large tortoises are extremely old as well). To correct for this, we should fit a model that is quadratic in `carapace length`. We will create a new covariate, `carapace.2`, which is the square of the carapace length and add it to the model.

In general I could write the quadratic model as

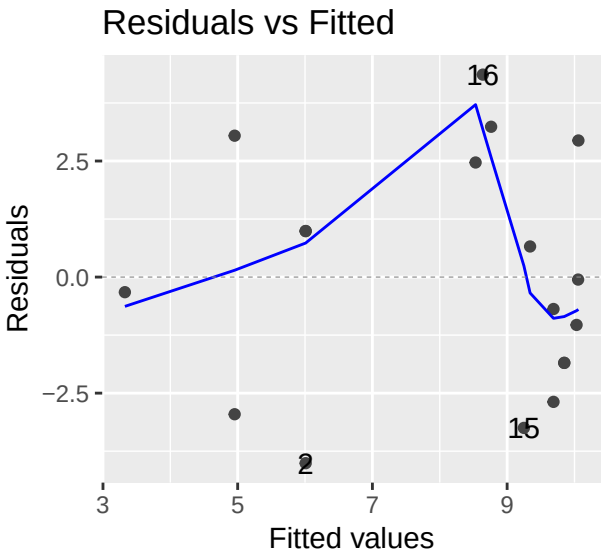
$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$$

and note that my model is still a linear model with respect to covariates $\boldsymbol{x}$ and $\boldsymbol{x}^2$ because I can still write

the model as

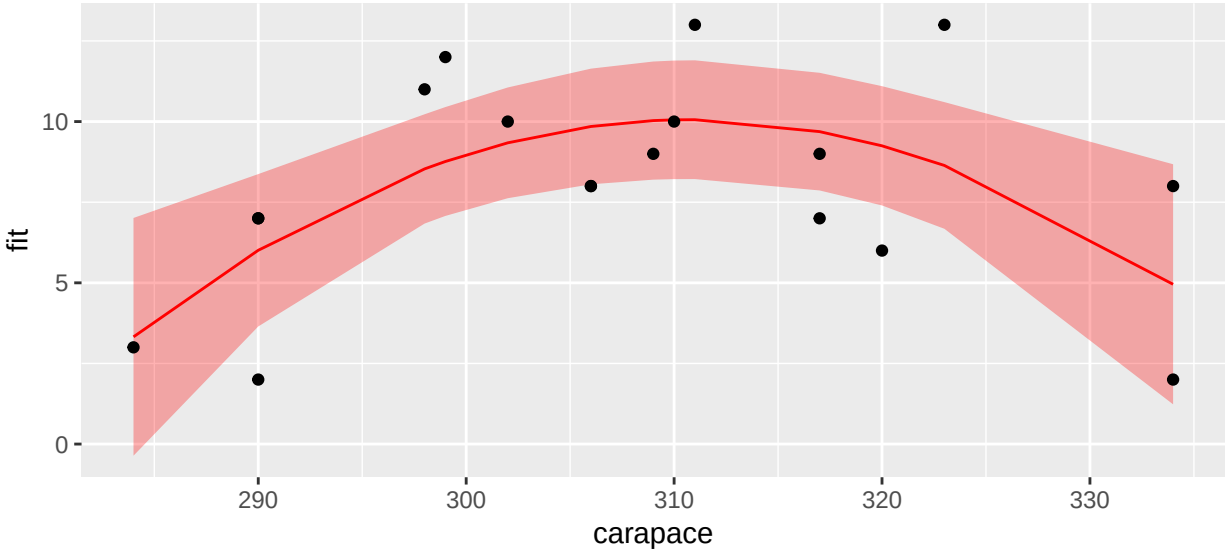
$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \\ &= \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{bmatrix} \end{aligned}$$

```
# add a new column that is carapace^2
Eggs2 <- Eggs %>% mutate( carapace.2 = carapace^2 )
model <- lm( clutch.size ~ carapace + carapace.2,    data=Eggs2 )

# make R do it inside the formula... convenient
model <- lm( clutch.size ~ carapace + I(carapace^2), data=Eggs )

# Fit an arbitrary degree polynomial
model <- lm( clutch.size ~ poly(carapace, 2),        data=Eggs )

# If you use poly() in the formula, you must use 'data=' here,
# otherwise you can skip it and R will do the right thing.
autoplot(model, which=1, data=Eggs)
```



Now our residual plot versus fitted values does not show any trend, suggesting that the quadratic model is fitting the data well. Graphing the original data along with the predicted values confirms this.

```
# add the fitted and CI lwr/upr columns to my dataset
Eggs <- Eggs %>%
  dplyr::do(if(!is.null(.$fit)){dplyr::select(-fit,-upr,-lwr )}else{.}) %>%
  cbind( predict(model, interval='confidence') )

ggplot(Eggs, aes(x=carapace)) +
  geom_ribbon( aes(ymin=lwr, ymax=upr), fill='red', alpha=.3) +
  geom_line(aes(y=fit), color='red') +
  geom_point(aes(y=clutch.size))
```



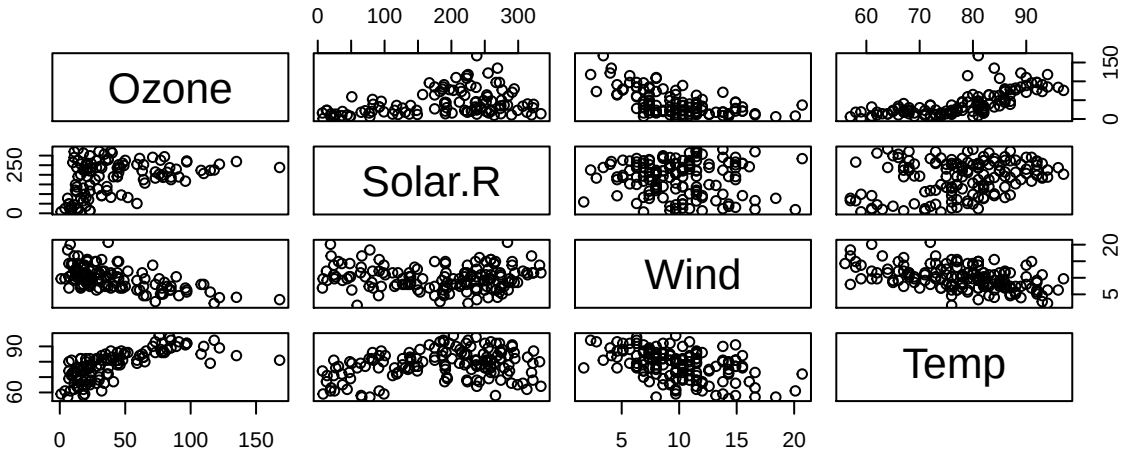
6.1.2.1.2 Heteroskedasticity

The plot of residuals versus fitted values can detect heteroskedasticity (non-constant variance) in the error terms.

To illustrate this, we turn to another dataset in the Faraway book. The dataset `airquality` uses data taken from an environmental study that measured four variables, ozone, solar radiation, temperature and wind speed for 153 consecutive days in New York. The goal is to predict the level of ozone using the weather variables.

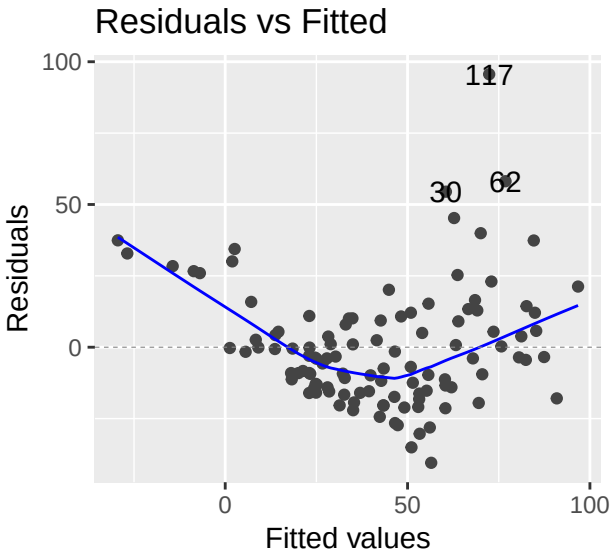
We first graph all pairs of variables in the dataset.

```
data(airquality)
pairs(~ Ozone + Solar.R + Wind + Temp, data=airquality)
```



and notice that ozone levels are positively correlated with solar radiation and temperature, and negatively correlated with wind speed. A linear relationship with wind might be suspect as is the increasing variability in the response to high temperature. However, we don't know if those trends will remain after fitting the model, because there is some covariance among the predictors.

```
model <- lm(Ozone ~ Solar.R + Wind + Temp, data=airquality)
autoplot(model, which=1)
```

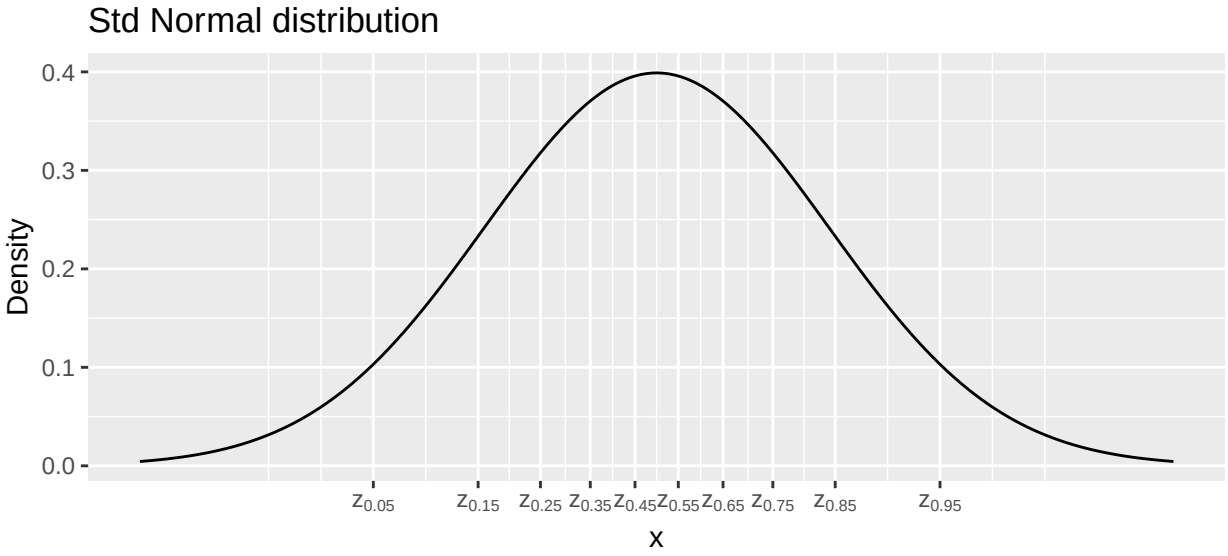


As we feared, we have both a non-constant variance and a non-linear relationship. A transformation of the y variable might be able to fix our problem.

6.1.2.2 QQplots

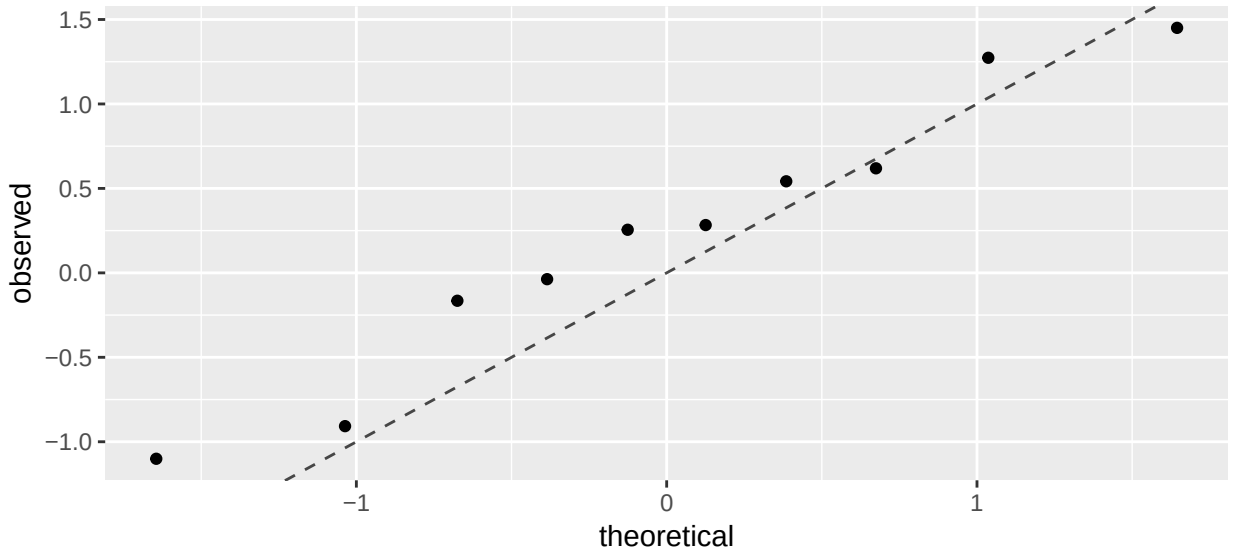
If we are taking a sample of size $n = 10$ from a standard normal distribution, then I should expect that the smallest observation will be negative. Intuitively, you would expect the smallest observation to be near the 10th percentile of the standard normal, and likewise the second smallest should be near the 20th percentile.

This idea needs a little modification because the largest observation cannot be near the 100th percentile (because that is ∞). So we'll adjust the estimates to still be spaced at $(1/n)$ quantile increments, but starting at the $0.5/n$ quantile instead of the $1/n$ quantile. So the smallest observation should be near the 0.05 quantile, the second smallest should be near the 0.15 quantile, and the largest observation should be near the 0.95 quantile. I will refer to these as the *theoretical quantiles*.



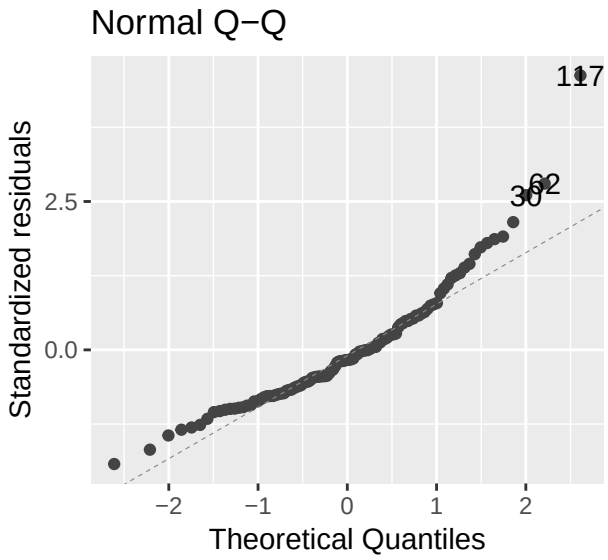
I can then graph the theoretical quantiles vs my observed values and if they lie on the 1-to-1 line, then my data comes from a standard normal distribution.

```r
set.seed(93516)  # make random sample in the next code chunk consistant run-to-run

n <- 10
data <- data.frame( observed = rnorm(n, mean=0, sd=1) ) %>%
  arrange(observed) %>%
  mutate( theoretical = qnorm( (1:n -.5)/n ) )

ggplot(data, aes(x=theoretical, y=observed) ) +
  geom_point() +
  geom_abline( intercept=0, slope=1,  linetype=2, alpha=.7) +
  labs(main='Q-Q Plot: Observed vs Normal Distribution')
```



In the context of a regression model, we wish to look at the residuals and see if there are obvious departures from normality. Returning to the air quality example, R will calculate the qqplot for us.

```r
model <- lm(Ozone ~ Solar.R + Wind + Temp, data=airquality)
autoplot(model, which=2)
```



In this case, we have a large number of residuals that are bigger than I would expect them to be based on

them being from a normal distribution. We could further test this using the Shapiro-Wilks test and compare the standardized residuals against a $N(0,1)$ distribution.

```
shapiro.test( rstandard(model) )
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rstandard(model)
## W = 0.9151, p-value = 2.819e-06
```

The tail of the distribution of observed residuals is *far* from what we expect to see.

6.1.2.3 Scale-Location Plot

This plot is a variation on the fitted vs residuals plot, but the y-axis uses the square root of the absolute value of the standardized residuals. Supposedly this makes detecting increasing variance easier to detect, but I'm not convinced.

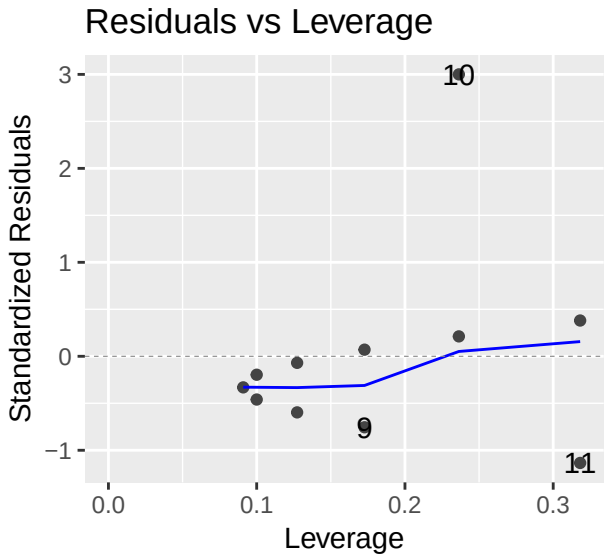
6.1.2.4 Residuals vs Leverage (plus Cook's Distance)

This plot lets the user examine the which observations have a high potential for being influential (i.e. high leverage) versus how large the residual is. Because Cook's distance is a function of those two traits, we can also divide the graph up into regions by the value of Cook's Distance.

Returning to Anscombe's third set of data, we see

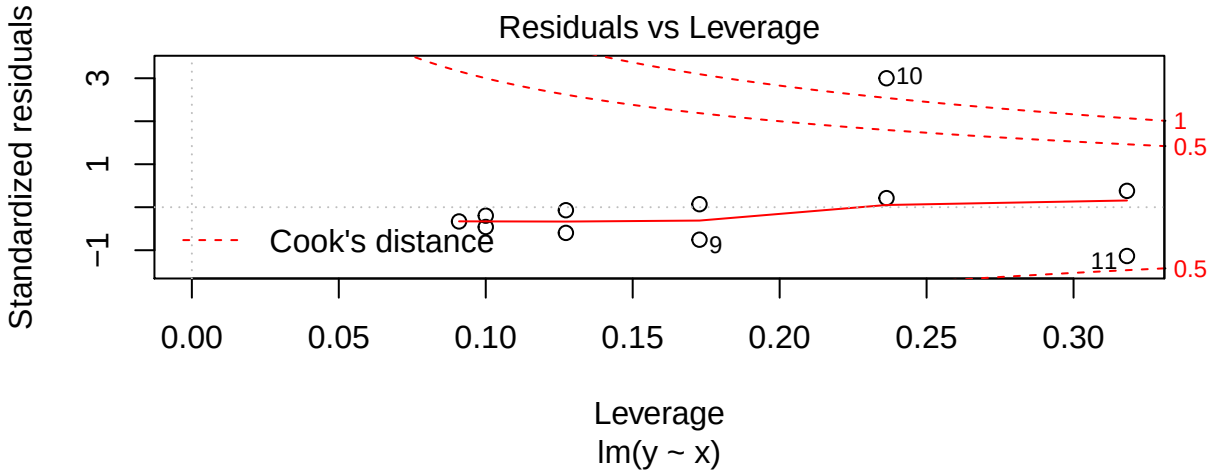
<<>>=

```
model3 <- lm(y ~ x, data=Set3)
autoplot(model3, which=5)
```



that one data point (observation 10) has an extremely large standardized residual. This is one plot where I prefer what the base graphics in R does compared to the `ggfortify` version. The base version of R adds some contour lines that mark where the contours of where Cook's distance is 1/2 and 1.

```
plot(model3, which=5)
```



6.2 Transformations

Transformations of the response variable and/or the predictor variables can drastically improve the model fit and can correct violations of the model assumptions. We might also create new predictor variables that are functions of existing variables. These include quadratic and higher order polynomial terms and interaction terms.

Often we are presented with data and we would like to fit a linear model to the data. Unfortunately the data might not satisfy all of the assumptions of a linear model. For the simple linear model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

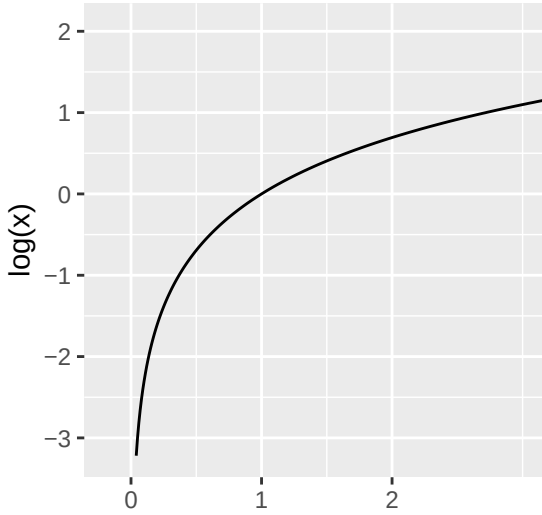
where $\epsilon_i \overset{iid}{\sim} N\left(0, \sigma^2\right)$, the necessary assumptions are:

1. Independent errors
2. Errors have constant variance, no matter what the x-value (or equivalently the fitted value)
3. Errors are normally distributed
4. The model contains all the appropriate covariates and no more.

In general, a transformation of the response variable can be used to address the 2nd and 3rd assumptions, and adding new covariates to the model will be how to address deficiencies of assumption 4.

6.2.1 A review of $\log(x)$ and e^x

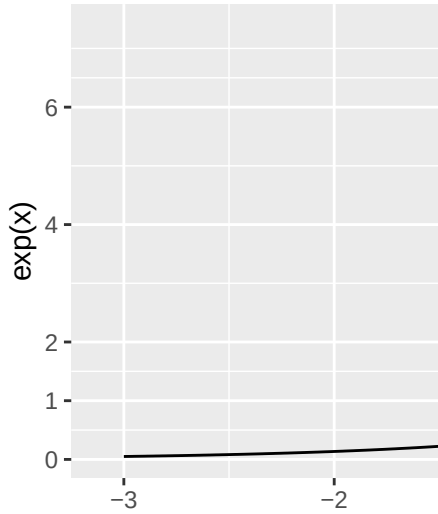
One of the most common transformations that is used on either the response y or the covariates x is the log() function. In this next section we will consider log() with base e. However, if you prefer $\log_2()$ or $\log_{10}$ you may substitute e with 2 or 10 everywhere.



In primary school you might have learned that the log() function looks like this:

Critical aspects to notice about $\log(x)$:

1. As $x \to 0$, $\log(x) \to -\infty$.
2. At $x = 1$ we have $log(x = 1) = 0$.
3. As $x \to \infty$, $\log(x) \to \infty$ as well, but at a *much* slower rate.
4. Even though $log(x)$ is only defined for $x > 0$, the result can take on any real value, positive or negative.



The inverse function of $\log(x)$ is $e^x = \exp(x)$, where $e = 2.71828\ldots$ which looks like this:

Critical aspects to notice about e^x:

1. as $x \to -\infty$, $e^x \to 0$.
2. At $x = 0$ we have $e^0 = 1$.
3. as $x \to \infty$, $e^x \to \infty$ as well, but at a *much* faster rate.
4. The function e^x can be evaluated for any real number, but the result is always > 0.

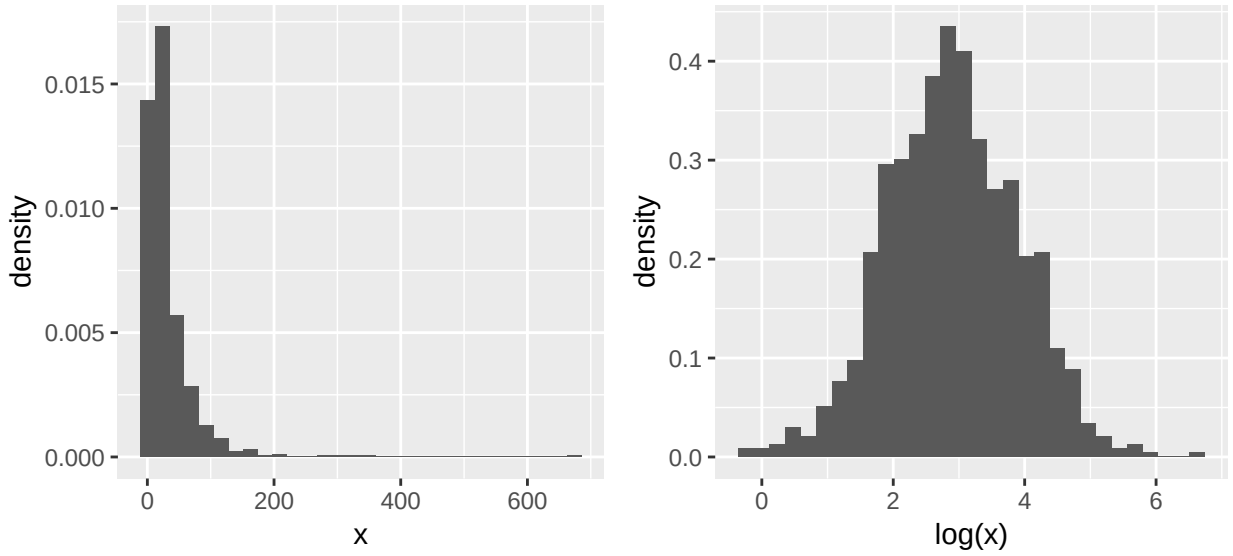
Finally we have that e^x and $log(x)$ are inverse functions of each other by the following identity:

$$x = \log\left(e^x\right)$$

and

$$x = e^{\log(x)} \quad \text{if } x > 0$$

The reason we like using a log() transformation is that it acts differentially on large values than small. In particular for $x > 1$ we have that $\log(x)$ makes all of the smaller, but the transformation on big values of x is more extreme. Consider the following, where most of the x-values are small, but we have a few that are quite large. Those large values will have extremely high leverage and we'd like to reduce that.



6.2.2 Transforming the Response

When the normality or constant variance assumption is violated, sometimes it is possible to transform the response to satisfy the assumption. Often times count data is analyzed as `log(count)` and weights are analyzed after taking a square root or cube root transform. Statistics involving income or other monetary values are usually analyzed on the log scale so as to reduce the leverage of high income observations.

While we may want to transform the response in order to satisfy the statistical assumptions, it is often necessary to back-transform to the original scale. For example if we fit a linear model for income (y) based on the amount of schooling the individual has received (x)

$$\log y = \beta_0 + \beta_1 x + \epsilon$$

then we might want to give a prediction interval for an x_0 value. The predicted $log(income)$ value is

$$\log\left(\hat{y}_0\right) = \hat{\beta}_0 + \hat{\beta}_x x_0$$

and we could calculate the appropriate predicted income as $\hat{y}_0 = e^{log(\hat{y}_0)}$. Likewise if we had a confidence interval or prediction interval for $\log\left(\hat{y}_0\right)$ of the form (l, u) then the appropriate interval for $\hat{y}_0$ is $\left(e^l, e^u\right)$. Notice that while (l, u) might be symmetric about $\log\left(\hat{y}_0\right)$, the back-transformed interval is not symmetric about $\hat{y}_0$.

Unfortunately the interpretation of the regression coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ on the untransformed scale becomes more complicated. This is a very serious difficulty and might sway a researcher from transforming their data.

6.2.2.1 Box-Cox Family of Transformations

The Box-Cox method is a popular way of determining what transformation to make. It is intended for responses that are strictly positive (because $\log 0 = -\infty$ and the square root of a number gives complex numbers, which we don't know how to address in regression). The transformation is defined as

$$g\left(y\right) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \log y & \lambda = 0 \end{cases}$$

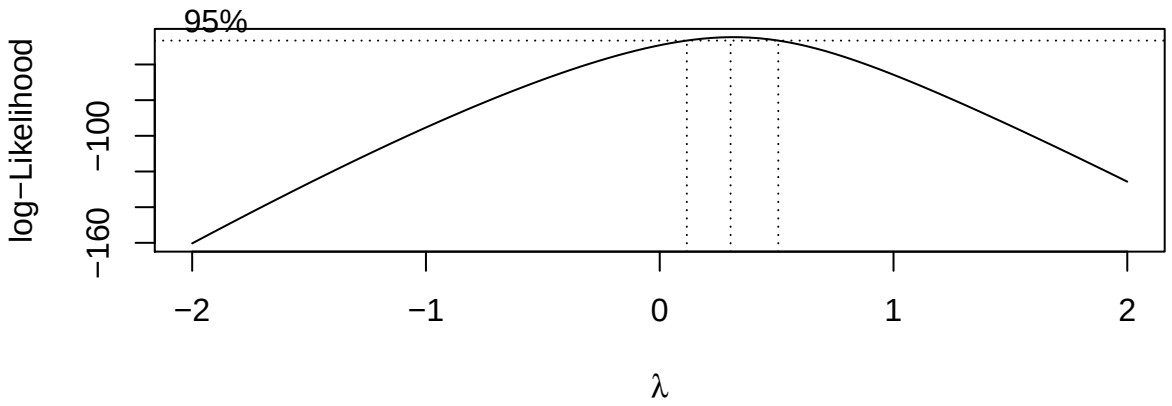
This transformation is a smooth family of transformations because

$$\lim_{\lambda \to 0} \frac{y^\lambda - 1}{\lambda} = \log y$$

In the case that $\lambda \neq 0$, then a researcher will usually use the simpler transformation y^λ because the subtraction and division does not change anything in a non-linear fashion. Thus for purposes of addressing the assumption violations, all we care about is the y^λ and prefer the simpler (i.e. more interpretable) transformation.

Finding the best transformation can be done by adding the λ parameter to the model and finding the value that maximizes the log-likelihood function. Fortunately, we don't have to do this by hand, as the function `boxcox()` in the `MASS` library will do all the heavy calculation for us.

```
data(gala, package='faraway')
g <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent, data=gala)

# I don't like loading the MASS package because it includes a select() function
# that fights with dplyr::select(), so whenever I use a function in the MASS
# package, I just call it using the package::function() naming.
MASS::boxcox(g)
```



The optimal transformation for these data would be $y^{1/4} = \sqrt[4]{y}$ but that is an extremely uncommon transformation. Instead we should pick the nearest "standard" transformation which would suggest that we should use either the $\log y$ or $\sqrt{y}$ transformation.

Thoughts on the Box-Cox transformation:

1. In general, I prefer to using a larger-than-optimal model when picking a transformation and then go about the model building process. After a suitable model has been chosen, I'll double check the my transformation was appropriate given the model that I ended up with.
2. Outliers can have a profound effect on this method. If the "optimal" transformation is extreme ($\lambda = 5$ or something silly) then you might have to remove the outliers and refit the transformation.
3. If the range of the response y is small, then the method is not as sensitive.
4. These are not the only possible transformations. For example, for binary data, the `logit` and `probit` transformations are common. In classical non-parametric statistics, we take a rank transformation to the y-values.

6.2.3 Transforming the predictors

6.2.3.1 Polynomials of a predictor

Perhaps the most common transformation to make is to make a quadratic function in x. Often the relationship between x and y follows a curve and we want to fit a quadratic model

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2$$

and we should note that this is still a linear model because $\hat{y}$ is a linear function of x and x^2. As we have already seen, it is easy to fit the model. Adding the column of x^2 values to the design matrix does the trick.

The difficult part comes in the interpretation of the parameter values. No longer is $\hat{\beta}_1$ the increase in y for every one unit increase in x. Instead the three parameters in my model interact in a complicated fashion. For example, the peak of the parabola is at $-\hat{\beta}_1/2\hat{\beta}_2$ and whether the parabola is cup shaped vs dome shaped and its steepness is controlled by $\hat{\beta}_2$. Because my geometric understanding of degree q polynomials relies on have all factors of degree q or lower, whenever I include a covariate raised to a power, I should include all the lower powers as well.

6.2.3.2 Log and Square Root of a predictor

Often the effect of a covariate is not linearly related to response, but rather some function of the covariate. For example the area of a circle is not linearly related to its radius, but it is linearly related to the radius squared.

$$Area = \pi r^2$$

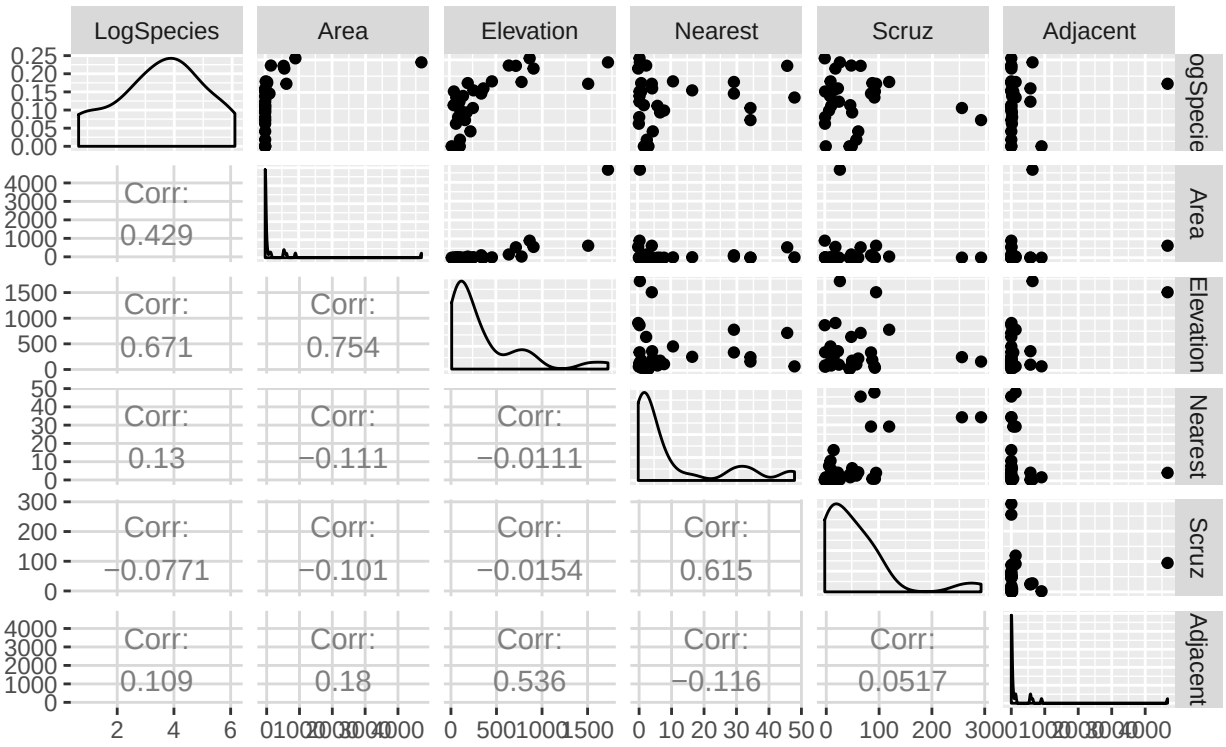
Similar situations might arise in biological settings, such as the volume of conducting tissue being related to the square of the diameter. Or perhaps an animals metabolic requirements are related to some power of body length. In sociology, it is often seen that the utility of, say, $1000 drops off in a logarithmic fashion according to the person's income. To a graduate student, $1K is a big deal, but to a corporate CEO, $1K is just another weekend at the track. Making a log transformation on any monetary covariate, might account for the non-linear nature of "utility".

Picking a good transformation for a covariate is quite difficult, but most fields of study have spent plenty of time thinking about these issues. When in doubt, look at scatter plots of the covariate vs the response and ask what transformation would make the data fall onto a line?

6.2.3.3 Examples of transformation of predictors

To illustrate how to add a transformation of a predictor to a linear model in R, we will consider the Galapagos data in `faraway`.

```
data('gala', package='faraway')
# look at all the scatterplots
gala %>%
  mutate(LogSpecies = log(Species)) %>%
  dplyr::select(LogSpecies, Area, Elevation, Nearest, Scruz, Adjacent) %>%
  GGally::ggpairs(upper=list(continuous='points'), lower=list(continuous='cor'))
```



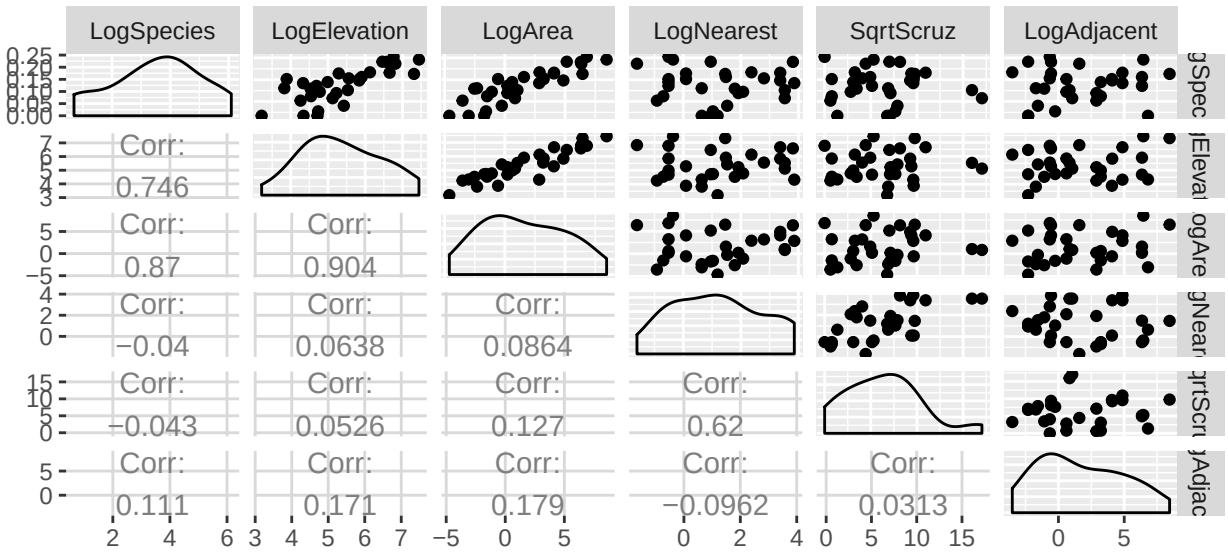
Looking at these graphs, I think I should definately transform `Area` and `Adjacent`, and I wouldn't object to doing the same to `Elevation`, `Nearest` and `Scruz`. Given the high leverages, a log transformation should be a good idea. One problem is that $\log(0) = -\infty$. A quick look at the data set summary:

```
gala %>%
  dplyr::select(Species, Area, Elevation, Nearest,Scruz, Adjacent) %>%
  summary()
```

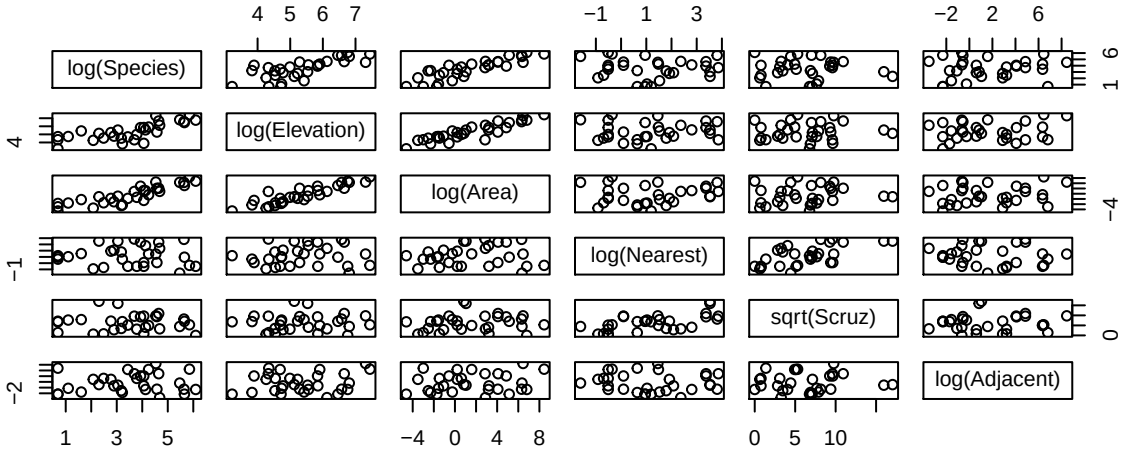
```
##     Species            Area           Elevation         Nearest
##  Min.   :  2.00   Min.   :   0.010   Min.   :  25.00   Min.   : 0.20
##  1st Qu.: 13.00   1st Qu.:   0.258   1st Qu.:  97.75   1st Qu.: 0.80
##  Median : 42.00   Median :   2.590   Median : 192.00   Median : 3.05
##  Mean   : 85.23   Mean   : 261.709   Mean   : 368.03   Mean   :10.06
##  3rd Qu.: 96.00   3rd Qu.:  59.237   3rd Qu.: 435.25   3rd Qu.:10.03
##  Max.   :444.00   Max.   :4669.320   Max.   :1707.00   Max.   :47.40
##      Scruz           Adjacent
##  Min.   :  0.00   Min.   :   0.03
##  1st Qu.: 11.03   1st Qu.:   0.52
##  Median : 46.65   Median :   2.59
##  Mean   : 56.98   Mean   : 261.10
##  3rd Qu.: 81.08   3rd Qu.:  59.24
##  Max.   :290.20   Max.   :4669.32
```

reveals that `Scruz` has a zero value, and so a log transformation will result in a $-\infty$. So, lets take the square root of `Scruz`

```
gala %>%
  mutate(LogSpecies = log(Species), LogElevation=log(Elevation), LogArea=log(Area), LogNearest=log(Near
         SqrtScruz=sqrt(Scruz), LogAdjacent=log(Adjacent)) %>%
  dplyr::select(LogSpecies, LogElevation, LogArea, LogNearest, SqrtScruz, LogAdjacent) %>%
  GGally::ggpairs(upper=list(continuous='points'), lower=list(continuous='cor'))
```




```
pairs( log(Species) ~ log(Elevation) + log(Area) +
                      log(Nearest) + sqrt(Scruz) + log(Adjacent), data=gala)
```



Looking at these graphs, it is clear that `log(Elevation)` and `log(Area)` are highly correlated and we should probably have one or the other, but not both in the model.

```
m.c <- lm(log(Species) ~  log(Area) + log(Nearest) + sqrt(Scruz) + log(Adjacent), data=gala)
summary(m.c)$coefficients %>% round(digits=3) # more readable...
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.285      0.275  11.960    0.000
## log(Area)         0.402      0.043   9.443    0.000
## log(Nearest)     -0.041      0.118  -0.351    0.728
## sqrt(Scruz)      -0.049      0.045  -1.085    0.288
## log(Adjacent)    -0.024      0.046  -0.529    0.602
```

We will remove all the parameters that appear to be superfluous, and perform an F-test to confirm that the simple model is sufficient.

```
m.s <- lm(log(Species) ~ log(Area), data=gala)
anova(m.s, m.c)
```

```
## Analysis of Variance Table
##
```

```
## Model 1: log(Species) ~ log(Area)
## Model 2: log(Species) ~ log(Area) + log(Nearest) + sqrt(Scruz) + log(Adjacent)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     28 17.218
## 2     25 15.299  3    1.9196 1.0456 0.3897
```

Next we will look at the coefficients.

```
summary(m.s)
```

```
##
## Call:
## lm(formula = log(Species) ~ log(Area), data = gala)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5442 -0.4001  0.0941  0.5449  1.3752
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.9037     0.1571  18.484  < 2e-16 ***
## log(Area)     0.3886     0.0416   9.342 4.23e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7842 on 28 degrees of freedom
## Multiple R-squared:  0.7571, Adjusted R-squared:  0.7484
## F-statistic: 87.27 on 1 and 28 DF,  p-value: 4.23e-10
```

The slope coefficient (0.3886) is the increase in log(Species) for every 1 unit increase in log(Area). Unfortunately that is not particularly convenient to interpretation and we will address this in the next section of this chapter.

Finally, we might be interested in creating a confidence interval for the expected number of tortoise species for an island with `Area=50`.

```
x0 <- data.frame(Area=50)
log.Species.CI <- predict(m.s, newdata=x0, interval='confidence')
log.Species.CI       # Log(Species) scale
```

```
##        fit      lwr      upr
## 1 4.423903 4.068412 4.779394
```

```
exp(log.Species.CI)  # Species scale
```

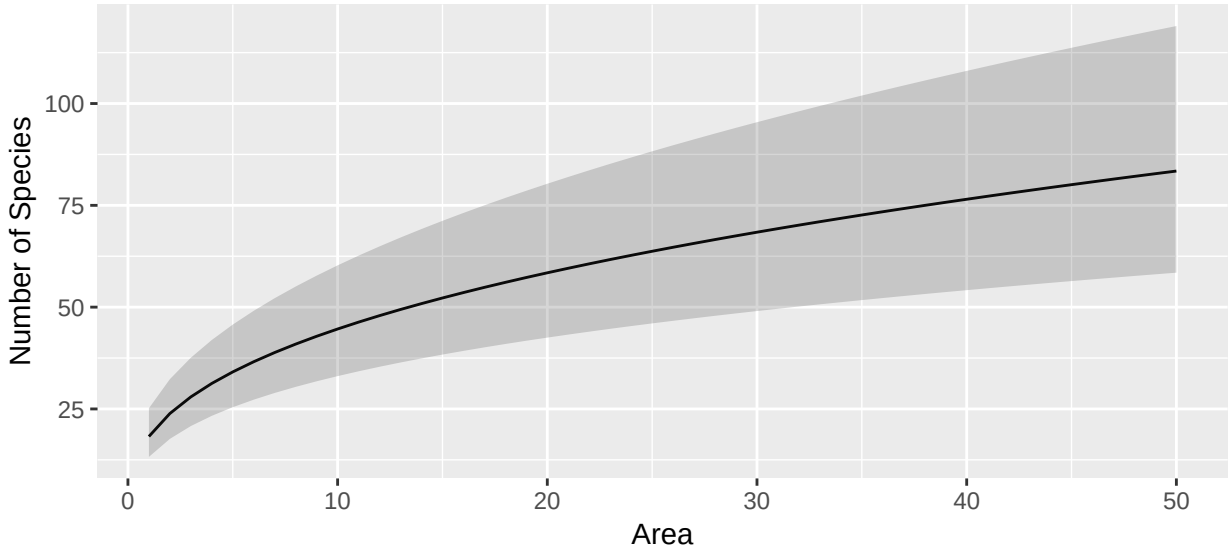
```
##        fit      lwr      upr
## 1 83.42122 58.46403 119.0322
```

Notice that on the species-scale, we see that the fitted value is not in the center of the confidence interval.

To help us understand what the log transformations are doing, we can produce a plot with the island Area on the x-axis and the expected number of Species on the y-axis and hopefully that will help us understand the relationship between the two.

```
library(ggplot2)
pred.data <- data.frame(Area=1:50)
pred.data <- pred.data %>%
  cbind( predict(m.s, newdata=pred.data, interval='conf'))
ggplot(pred.data, aes(x=Area)) +
```

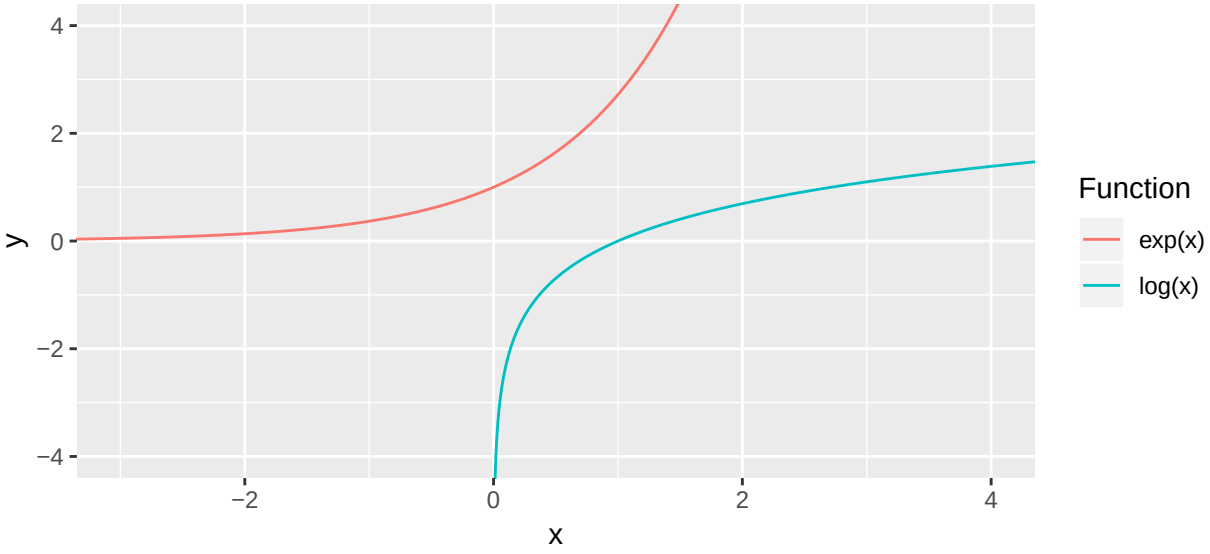
```
  geom_line(aes(y=exp(fit))) +
  geom_ribbon(aes(ymin=exp(lwr), ymax=exp(upr)), alpha=.2) +
  ylab('Number of Species')
```



6.2.4 Interpretation of log transformed variables

One of the most difficult issues surrounding transformed variables is that the interpretation is difficult. Here we look at the interpretation of log transformed variables.

To begin with, we need to remind ourselves of what the functions $\log x$ and e^x look like.



In particular we notice that

$$e^0 = 1$$

and

$$\log(1) = 0$$

and the functions e^x and $\log x$ are inverse functions of each other.

$$e^{\log x} = \log\left(e^x\right) = x$$

Also it is important to note that the log function has some interesting properties in that it makes operations "1-operation easier".

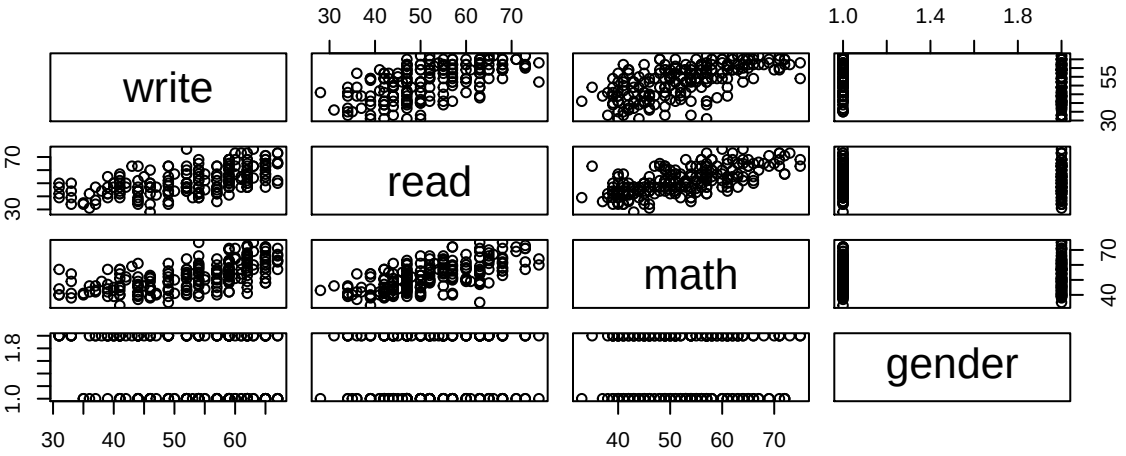
$$\log\left(a^b\right) = b\log a$$
$$\log\left(\frac{a}{b}\right) = \log a - \log b$$
$$\log\left(ab\right) = \log a + \log b$$

One final aspect of exponents that we will utilize is that

$$e^{a+b} = e^a e^b$$

To investigate the effects of a log transformation, we'll examine a dataset that predicts the writing scores of $n = 200$ students using the gender, reading and math scores. This example was taken from the UCLA Statistical Consulting Group.

```
file <- 'https://stats.idre.ucla.edu/wp-content/uploads/2016/02/lgtrans.csv'  # on the web
file <- 'data-raw/lgtrans.csv'                                                # on my laptop
scores <- read.csv(file=file)
scores <- scores %>% mutate(gender = female)
pairs(write~read+math+gender, data=scores)
```



These data look pretty dencent, and I'm not certain that I would do *any* transformation, but for the sake of having a concrete example that has both continuous and categorical covariates, we will interpret effects on a student's writing score.

6.2.4.1 Log-transformed response, untransformed covariates

We consider the model where we have transformed the response variable and just an intercept term.

$$\log y = \beta_0 + \epsilon$$

```
m <- lm(log(write) ~ 1, data=scores)
summary(m)$coef %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.948      0.014 288.402        0
```

We interpret the intercept as the mean of the log-transformed response values. We could back transform this to the original scale $\hat{y} = e^{\hat{\beta}_0} = e^{3.948} = 51.83$ as a typical value of write. To distinguish this from the usually defined mean of the write values, we will call this as the *geometric mean*.

Next we examine how to interpret the model when a categorical variable is added to the model.

$$\log y = \begin{cases} \beta_0 + \epsilon & \text{if female} \\ \beta_0 + \beta_1 + \epsilon & \text{if male} \end{cases}$$

```
m <- lm(log(write) ~ gender, data=scores)
summary(m)$coef %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.995      0.018 222.949        0
## gendermale    -0.103      0.027  -3.887        0
```

The intercept is now the mean of the log-transformed `write` responses for the females and thus $e^{\hat{\beta}_0} = \hat{y}_f$ and the offset for males is the change in `log(write)` from the female group. Notice that for the males, we have

$$\begin{aligned} \log \hat{y}_m &= \hat{\beta}_0 + \hat{\beta}_1 \\ \hat{y}_m &= e^{\hat{\beta}_0 + \hat{\beta}_1} \\ &= \underbrace{e^{\hat{\beta}_0}}_{\hat{y}_f} \quad * \quad \underbrace{e^{\hat{\beta}_1}}_{\text{percent change for males}} \end{aligned}$$

and therefore we see that males tend to have writing scores $e^{-0.103} = 0.90 = 90\%$ of the females. Typically this sort of result would be reported as the males have a 10% lower writing score than the females.

The model with a continuous covariate has a similar interpretation.

$$\log y = \begin{cases} \beta_0 + \beta_2 x + \epsilon & \text{if female} \\ \beta_0 + \beta_1 + \beta_2 x + \epsilon & \text{if male} \end{cases}$$

We will use the reading score read to predict the writing score. Then $\hat{\beta}_2$ is the predicted increase in `log(write)` for every 1-unit increase in read score. The interpretation of $\hat{\beta}_0$ is now $\log \hat{y}$ when $x = 0$ and therefore $\hat{y} = e^{\hat{\beta}_0}$ when $x = 0$.

```
m <- lm(log(write) ~ gender + read, data=scores)  # main effects model
summary(m)$coefficients %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.412      0.055  62.452        0
## gendermale    -0.116      0.021  -5.516        0
## read           0.011      0.001  11.057        0
```

For females, we consider the difference in $\log \hat{y}$ for a 1-unit increase in x and will interpret this on the original write scale.

$$\begin{aligned} \log \hat{y}_f &= \hat{\beta}_0 + \hat{\beta}_2 x \\ \hat{y}_f &= e^{\hat{\beta}_0 + \hat{\beta}_2 x} \end{aligned}$$

therefore we consider $e^{\hat{\beta}_2}$ as the *percent* increase in write score for a 1-unit increase in x because of the following. Consider x_1 and $x_2 = x_1 + 1$. Then we consider the ratio of predicted values:

$$\frac{\hat{y}_2}{\hat{y}_1} = \frac{e^{\hat{\beta}_0 + \hat{\beta}_2 (x+1)}}{e^{\hat{\beta}_0 + \hat{\beta}_2 x}} = \frac{e^{\hat{\beta}_0} e^{\hat{\beta}_2 x} e^{\hat{\beta}_2}}{e^{\hat{\beta}_0} e^{\hat{\beta}_2 x}} = e^{\hat{\beta}_2}$$

For our writing scores example we have that $e^{\hat{\beta}_2} = e^{0.011} = 1.01$ meaning there is an estimated 1% increase in `write` score for every 1-point increase in `read` score.

If we are interested in, say, a 20-unit increase in x, then that would result in an increase of

$$\frac{e^{\hat{\beta}_0+\hat{\beta}_2\,(x+20)}}{e^{\hat{\beta}_0+\hat{\beta}_2\,x}} = \frac{e^{\hat{\beta}_0}e^{\hat{\beta}_2\,x}e^{20\hat{\beta}_2}}{e^{\hat{\beta}_0}e^{\hat{\beta}_2\,x}} = e^{20\hat{\beta}_2} = \left(e^{\hat{\beta}_2}\right)^{20}$$

and for the writing scores we have

$$e^{20\hat{\beta}_2} = \left(e^{\hat{\beta}_2}\right)^{20} = 1.01^{20} = 1.22$$

or a 22% increase in writing score for a 20-point increase in reading score.

In short, we can interpret $e^{\hat{\beta}_i}$ as the percent increase/decrease in the non-transformed response variable. Some students get confused by what is meant by a % increase or decrease in y.

- A 75% decrease in y has a resulting value of $(1 - 0.75)\,y = (0.25)\,y$
- A 75% increase in y has a resulting value of $(1 + 0.75)\,y = (1.75)\,y$
- A 100% increase in y has a resulting value of $(1 + 1.00)\,y = 2x$ and is a doubling of y.
- A 50% decrease in y has a resulting value of $(1 - 0.5)\,y = (0.5)\,x$ and is a halving of y.

6.2.4.2 Untransformed response, log-transformed covariate

We consider the model

$$y = \beta_0 + \beta_2 \log x + \epsilon$$

and consider two different values of x (which we'll call x_1 and x_2 and we are considering the effect of moving from x_1 to x_2) and look at the differences between the predicted values $\hat{y}_2 - \hat{y}_1$.

$$\begin{aligned}\hat{y}_2 - \hat{y}_1 &= \left[\hat{\beta}_0 + \hat{\beta}_2 \log x_2\right] - \left[\hat{\beta}_0 + \hat{\beta}_2 \log x_1\right] \\ &= \hat{\beta}_2\left[\log x_2 - \log x_1\right] \\ &= \hat{\beta}_2 \log\left[\frac{x_2}{x_1}\right]\end{aligned}$$

This means that so long as the ratio between the two x-values is constant, then the change in $\hat{y}$ is the same. So doubling the value of x from 1 to 2 has the same effect on $\hat{y}$ as changing x from 50 to 100.

```
m <- lm( write ~ gender + log(read), data=scores)
summary(m)$coefficients %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -59.076      9.948  -5.938        0
## gendermale    -5.431      1.013  -5.362        0
## log(read)     29.045      2.527  11.493        0
```

```
# predict writing scores for three females,
# each with a reading score 50% larger than the other previous
predict(m, newdata=data.frame(gender=rep('female',3),
                              read=c(40, 60, 90)))
```

```
##        1        2        3
## 48.06622 59.84279 71.61936
```

We should see a

$$29.045\ \log(1.5) = 11.78$$

difference in $\hat{y}$ values for the first and second students and the second and third.

6.2.4.3 Log-transformed response, log-transformed covariate

This combines the interpretations in the previous two sections. We consider

$$\log y = \beta_0 + \beta_2 \log x + \epsilon$$

and we again consider two x values (again x_1 and x_2). We then examine the difference in the $\log \hat{y}$ values as

$$\begin{aligned}
\log \hat{y}_2 - \log \hat{y}_1 &= \left[\hat{\beta}_0 + \hat{\beta}_2 \log x_2\right] - \left[\hat{\beta}_0 + \hat{\beta}_2 \log x_1\right] \\
\log\left[\frac{\hat{y}_2}{\hat{y}_1}\right] &= \hat{\beta}_2 \log\left[\frac{x_2}{x_1}\right] \\
\log\left[\frac{\hat{y}_2}{\hat{y}_1}\right] &= \log\left[\left(\frac{x_2}{x_1}\right)^{\hat{\beta}_2}\right] \\
\frac{\hat{y}_2}{\hat{y}_1} &= \left(\frac{x_2}{x_1}\right)^{\hat{\beta}_2}
\end{aligned}$$

This allows us to examine the effect of some arbitrary percentage increase in x value as a percentage increase in y value.

```
m <- lm(log(write) ~ gender + log(read), data=scores)
summary(m)$coefficients %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.714      0.205   8.358        0
## gendermale    -0.114      0.021  -5.483        0
## log(read)      0.581      0.052  11.148        0
```

which implies for a 10% increase in `read` score, we should see a $1.10^{0.581} = 1.05$ multiplier in `write` score. That is to say, a 10% increase in reading score results in a 5% increase in writing score.

For the Galapagos islands, we had

```
m.s <- lm(log(Species) ~ log(Area), data=gala)
summary(m.s)$coefficients %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.904      0.157  18.484        0
## log(Area)      0.389      0.042   9.342        0
```

and therefore doubling of Area (i.e. the ratio of the $Area_2/Area_1 = 2$) results in a $2^{0.389} = 1.31$ multiplier of the `Species` value. That is to say doubling the island area increases the number of species by 31%.

6.3 Exercises

1. The dataset `infmort` in the `faraway` package has information about infant mortality from countries around the world. Be aware that this is a old data set and does not necessarily reflect current conditions. More information about the dataset can be found using `?faraway::infmort`. We will be interested

in understanding how infant mortality is predicted by per capita income, world region, and oil export status.

a. Plot the relationship between income and mortality. This can be done using the command

```
data('infmort', package='faraway')
pairs(mortality ~., data=infmort)
```

What do you notice about the relationship between mortality and income?

b. Fit a linear model without any interaction terms with all three covariates as predictors of infant mortality. Examine the diagnostic plots. What stands out?

c. Use the `boxcox()` function in the library MASS to determine a what a good transformation to the mortality response variable.

d. Make a log transformation to the mortality variable and refit the model without interactions. Use the log transformed mortality for all further questions.

e. Examine the pairs plot with log(mortality), income, and log(income). Which should be used in our model, `income` or `log(income)`?

f. Examine models that have all three main effects and either the `region:log(income)` or `oil:log(income)` interaction. Are either interaction significant vs the model with just the three main effects? What about a model that contains both interactions (and the three main effects)?

g. Interpret the effects of income, world region, and oil exports on log infant mortality based on these data. *Hint: graph the data and the predicted values.*

2. Using the `pressure` data in the `datasets` package, fit a model with pressure as the response and temperature as the predictor using transformations to obtain a good fit. Feel free to experiment with what might be considered a ridiculously complicated model with a high degree polynomial.

a. Document your process of building your final model. Do not show graphs or computer output that is not relevant to your decision or that you do not wish to comment on.

b. Comment on the interpretability of your (possibly ridiculously complicated) model.

3. Use transformations to find a good model for `volume` in terms of `girth` and `height` using the `trees` dataset in the `faraway` package.

a. Document your process of building your final model. Again, only include output or graphs that are relevant to your decisions and include discussion about anything you include.

b. Create a prediction interval for the volume of a tree with `girth=16` and `height=70`. Notice that if you have transformed your response variable in you model, you'll have to back-transform to the original y-scale.

4. For this problem, we will look at a manufacturing problem. We will investigate the relationship predicting the `Time` taken polishing a newly manufactured dish versus the dish `Diameter` (in inches), `Type`, and `Price`.

a. The data live in a package I have on GitHub. The following code will download the data package:

```
library(devtools)    # You might need to load this package the usual way...
install_github('dereksonderegger/dsData') # load an R package that lives on GitHub.
```

b. Load the data and examine it using the commands

```
library(dsData)
data('Dishes')
str(Dishes)
pairs(Time ~ ., data=Dishes)
```

Comment on the relationships and possible transformations to be made.

c. Fit a linear model predicting `Time` as a function the main of `Diameter`, `Price`, and `Type`, but with no interaction.

d. Examine the diagnostic plots. What stands out to you?

e. While most of the diagnostics look fine, there is weak evidence that there might be some heteroskedasticity. To address this (and provide an interesting model to interpret), refit your linear model but with a log-transformed Time and Price variables.

f. Using your model from part (e), what is your interpretation of the parameter associated with the Diameter variable on the original scale of the Time variable? Does this make sense to you

considering the `pairs()` plot?

g. How does a 20% increase in Price affect the polishing Time?

Chapter 7

Variable Selection

```
library(tidyverse)    # dplyr, tidyr, ggplot2, etc
```

Given a set of data, we are interested in selecting the best subset of predictors for the following reasons:

1. Occam's Razor tells us that from a list of plausible model or explanations, the simplest is usually the best. In the statistics sense, I want the smallest model that adequately explains the observed data patterns.

2. Unnecessary predictors add noise to the estimates of other quantities and will waste degrees of freedom, possibly increasing the estimate of $\hat{\sigma}^2$.

3. We might have variables that are co-linear.

The problems that arise in the diagnostics of a model will often lead a researcher to consider other models, for example to include a quadratic term to account for curvature. The model building process is often an iterative procedure where we build a model, examine the diagnostic plots and consider what could be added or modified to correct issues observed.

7.1 Nested Models

Often one model is just a simplification of another and can be obtained by setting some subset of β_i values equal to zero. Those models can be adequately compared by the F-test, which we have already made great use of.

We should be careful to note that we typically do not want to remove the main covariate from the model if the model uses the covariate in a more complicated fashion. For example, if my model is

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$$

where $\epsilon \sim N\left(0, \sigma^2\right)$, then considering the simplification $\beta_1 = 0$ and removing the effect of x is not desirable because that forces the parabola to be symmetric about $x = 0$. Similarly, if the model contains an interaction effect, then the removal of the main effect drastically alters the interpretation of the interaction coefficients and should be avoided. Often times removing a lower complexity term while keeping a higher complexity term results in unintended consequences and is typically not recommended.

7.2 Testing-Based Model Selection

Starting with a model that is likely too complex, consider a list of possible terms to remove and remove each in turn while evaluating the resulting model to the starting model using an F-test. Whichever term has the highest p-value is removed and the process is repeated until no more terms have non-significant p-values. This is often referred to as *backward selection*.

It should be noted that the cutoff value for significance here does not have to be $\alpha = 0.05$. If prediction performance is the primary goal, then a more liberal α level is appropriate.

Starting with a model that is likely too small, consider adding terms until there are no more terms that when added to the model are significant. This is called *forward selection*.

This is a hybrid between forward selection and backward elimination. At every stage, a term is either added or removed. This is referred to as *stepwise selection*.

Stepwise, forward, and backward selection are commonly used but there are some issues.

1. Because of the "one-at-a-time" nature of the addition/deletion, the most optimal model might not be found.
2. p-values should not be treated literally. Because the multiple comparisons issue is completely ignored, the p-values are lower than they should be if multiple comparisons were accounted for. As such, it is possible to sort through a huge number of potential covariates and find one with a low p-value simply by random chance. This is "data dredging" and is a serious issue.
3. As a non-thinking algorithm, these methods ignore the science behind that data and might include two variables that are highly collinear or might ignore variables that are scientifically interesting.

7.2.1 Example - U.S. Life Expectancy

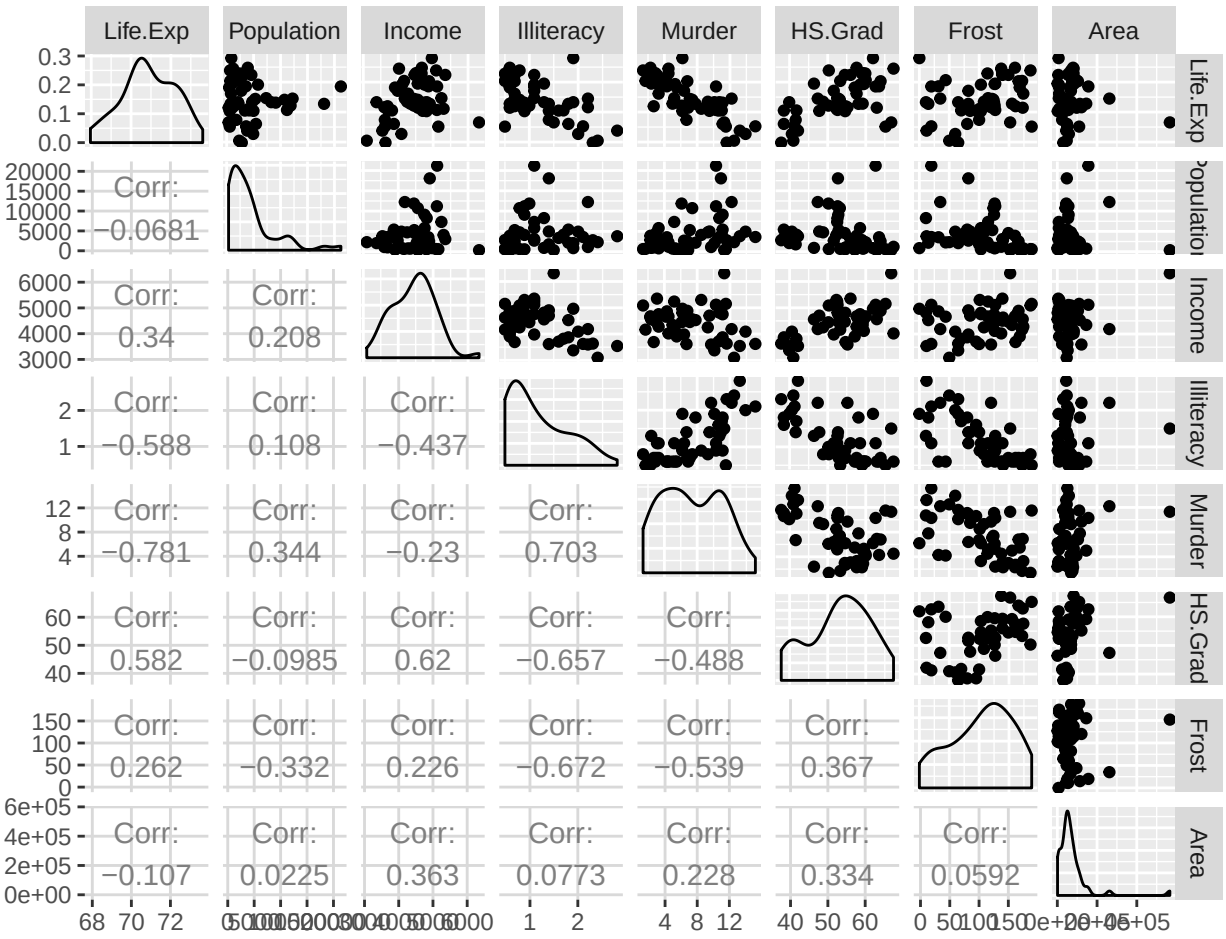
Using data from the Census Bureau we can look at the life expectancy as a response to a number of predictors. One R function that is often convenient to use is the update() function that takes a `lm()` object and adds or removes things from the formula. The notation `. ~ .` means to leave the response and all the predictors alone, while `. ~ . + vnew` will add the main effect of `vnew` to the model.

```
data('state')  # loads a matrix state.x77 and a vector of stat abbreviations

# Convert from a matrix to a data frame with state abbreviations
state.data <- data.frame(state.x77, row.names=state.abb)
str(state.data)
```

```
## 'data.frame':    50 obs. of  8 variables:
##  $ Population: num  3615 365 2212 2110 21198 ...
##  $ Income    : num  3624 6315 4530 3378 5114 ...
##  $ Illiteracy: num  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2 ...
##  $ Life.Exp  : num  69 69.3 70.5 70.7 71.7 ...
##  $ Murder    : num  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10.7 13.9 ...
##  $ HS.Grad   : num  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 52.6 40.6 ...
##  $ Frost     : num  20 152 15 65 20 166 139 103 11 60 ...
##  $ Area      : num  50708 566432 113417 51945 156361 ...
```

We should first look at the

```
state.data %>%
  dplyr::select( Life.Exp, Population:Area ) %>%
  GGally::ggpairs( upper=list(continuous='points'), lower=list(continuous='cor') )
```



I want to add a quadratic effect for `HS.Grad` rate and for `Income`. Also, we see that `Population` and `Area` seem to have some high skew to their distributions, so a log transformation might help. We'll modify the data and then perform the backward elimination method starting with the model with all predictors as main effects.

```
state.data <- state.data %>%
  mutate( HS.Grad.2 = HS.Grad ^ 2,
          Income.2  = Income  ^ 2,
          Log.Population = log(Population),
          Log.Area = log(Area)) %>%
  dplyr::select( -Population, -Area )   # remove the original Population and Area covariates

# explicitly define my starting model
m1 <- lm(Life.Exp ~ Log.Population + Income + Illiteracy +
                    Murder + HS.Grad + Frost + HS.Grad.2 + Income.2 +  Log.Area, data=state.data)
#
# Define the same model, but using shorthand
# The '.' means everything else in the data frame
m1 <- lm( Life.Exp ~ ., data=state.data)
summary(m1)$coefficients %>% round(digits=3)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      61.174      6.984   8.759    0.000
## Income            0.004      0.003   1.517    0.137
## Illiteracy        0.367      0.388   0.946    0.350
```

```
## Murder          -0.304      0.049  -6.214    0.000
## HS.Grad         -0.031      0.208  -0.151    0.881
## Frost           -0.004      0.003  -1.068    0.292
## HS.Grad.2        0.001      0.002   0.394    0.696
## Income.2         0.000      0.000  -1.518    0.137
## Log.Population   0.191      0.150   1.273    0.211
## Log.Area         0.100      0.113   0.885    0.382
```

The signs make reasonable sense (higher murder rates decrease life expectancy) but covariates like `Income` are not significant, which is surprising. The largest p-value is `HS.Grad`. However, I don't want to remove the lower-order graduation term and keep the squared-term. So instead I will remove both of them since they are the highest p-values. Notice that `HS.Grad` is correlated with `Income` and `Illiteracy`.

```
# Remove Graduation Rate from the model from the model
m1 <- update(m1, .~. - HS.Grad - HS.Grad.2)
summary(m1)$coefficients %>% round(digits=3)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      60.894      6.100   9.983    0.000
## Income            0.004      0.002   1.711    0.094
## Illiteracy        0.087      0.373   0.233    0.817
## Murder           -0.318      0.048  -6.686    0.000
## Frost            -0.006      0.003  -1.807    0.078
## Income.2          0.000      0.000  -1.581    0.121
## Log.Population    0.041      0.132   0.309    0.759
## Log.Area          0.206      0.103   1.995    0.053
```

```
# Next remove Illiteracy
m1 <- update(m1, .~. - Illiteracy)
summary(m1)$coefficients %>% round(digits=3)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      61.779      4.717  13.098    0.000
## Income            0.004      0.002   1.872    0.068
## Murder           -0.314      0.042  -7.423    0.000
## Frost            -0.006      0.003  -2.345    0.024
## Income.2          0.000      0.000  -1.699    0.097
## Log.Population    0.041      0.130   0.314    0.755
## Log.Area          0.198      0.097   2.048    0.047
```

```
# And Log.Population...
m1 <- update(m1, .~. - Log.Population)
summary(m1)$coefficients %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   61.413      4.523  13.577    0.000
## Income         0.004      0.002   2.257    0.029
## Murder        -0.309      0.039  -7.828    0.000
## Frost         -0.006      0.002  -2.612    0.012
## Income.2       0.000      0.000  -2.044    0.047
## Log.Area       0.200      0.096   2.091    0.042
```

The removal of `Income.2` is a tough decision because the p-value is very close to $\alpha = 0.05$ and might be left in if it makes model interpretation easier or if the researcher feels a quadratic effect in income is appropriate (perhaps rich people are too stressed?).

```
summary(m1)
```

```
##
## Call:
## lm(formula = Life.Exp ~ Income + Murder + Frost + Income.2 +
##     Log.Area, data = state.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.28858 -0.50631 -0.07242  0.49738  1.75839
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.141e+01  4.523e+00  13.577  < 2e-16 ***
## Income       4.212e-03  1.867e-03   2.257   0.0290 *
## Murder      -3.092e-01  3.950e-02  -7.828 7.14e-10 ***
## Frost       -6.487e-03  2.483e-03  -2.612   0.0123 *
## Income.2    -4.188e-07  2.049e-07  -2.044   0.0470 *
## Log.Area     2.002e-01  9.576e-02   2.091   0.0424 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7349 on 44 degrees of freedom
## Multiple R-squared:  0.7309, Adjusted R-squared:  0.7003
## F-statistic:  23.9 on 5 and 44 DF,  p-value: 1.549e-11
```

We are left with a model that adequately explains `Life.Exp` but we should be careful to note that just because a covariate was removed from the model does not imply that it isn't related to the response. For example, being a high school graduate is highly correlated with not being illiterate as is `Income` and thus replacing `Illiteracy` shows that illiteracy is associated with lower life expectancy, but it is not as predictive as `Income`.

```
m2 <- lm(Life.Exp ~ Illiteracy+Murder+Frost, data=state.data)
summary(m2)
```

```
##
## Call:
## lm(formula = Life.Exp ~ Illiteracy + Murder + Frost, data = state.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.59010 -0.46961  0.00394  0.57060  1.92292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 74.556717   0.584251 127.611  < 2e-16 ***
## Illiteracy  -0.601761   0.298927  -2.013  0.04998 *
## Murder      -0.280047   0.043394  -6.454 6.03e-08 ***
## Frost       -0.008691   0.002959  -2.937  0.00517 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7911 on 46 degrees of freedom
## Multiple R-squared:  0.6739, Adjusted R-squared:  0.6527
## F-statistic: 31.69 on 3 and 46 DF,  p-value: 2.915e-11
```

Notice that the R^2 values for both models are quite similar 0.7309 vs 0.6739 but the first model with the higher R^2 has one more predictor variable? Which model should I prefer? I can't do an F-test because these

models are not nested.

7.3 Criterion Based Procedures

7.3.1 Information Criterions

It is often necessary to compare models that are not nested. For example, I might want to compare

$$y = \beta_0 + \beta_1 x + \epsilon$$

vs

$$y = \beta_0 + \beta_2 w + \epsilon$$

This comparison comes about naturally when doing forward model selection and we are looking for the "best" covariate to add to the model first.

Akaike introduced his criterion (which he called "An Information Criterion") as

$$AIC = \underbrace{-2 \log L\left(\hat{\boldsymbol{\beta}}, \hat{\sigma} | \text{ data}\right)}_{\text{decreases if RSS decreases}} + \underbrace{2p}_{\text{increases as p increases}}$$

where $L\left(\hat{\boldsymbol{\beta}} | \text{ data}\right)$ is the likelihood function and p is the number of elements in the $\hat{\boldsymbol{\beta}}$ vector and we regard a lower AIC value as better. Notice the $2p$ term is essentially a penalty on adding addition covariates so to lower the AIC value, a new predictor must lower the negative log likelihood more than it increases the penalty.

To convince ourselves that the first summand decreases with decreasing RSS in the standard linear model, we examine the likelihood function

$$\begin{aligned} f\left(\boldsymbol{y} \,|\, \boldsymbol{\beta}, \sigma, \boldsymbol{X}\right) &= \frac{1}{\left(2\pi\sigma^2\right)^{n/2}} \exp\left[-\frac{1}{2\sigma^2}\left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\right)^T \left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\right)\right] \\ &= L\left(\boldsymbol{\beta}, \sigma \,|\, \boldsymbol{y}, \boldsymbol{X}\right) \end{aligned}$$

and we could re-write this as

$$\begin{aligned} \log L\left(\hat{\boldsymbol{\beta}}, \hat{\sigma} \,|\, \text{data}\right) &= -\log\left(\left(2\pi\hat{\sigma}^2\right)^{n/2}\right) - \frac{1}{2\hat{\sigma}^2}\left(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\right)^T \left(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\right) \\ &= -\frac{n}{2}\log\left(2\pi\hat{\sigma}^2\right) - \frac{1}{2\hat{\sigma}^2}\left(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\right)^T \left(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\right) \\ &= -\frac{1}{2}\left[n\log\left(2\pi\hat{\sigma}^2\right) + \frac{1}{\hat{\sigma}^2}\left(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\right)^T \left(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\right)\right] \\ &= -\frac{1}{2}\left[+n\log\left(2\pi\right) + n\log\hat{\sigma}^2 + \frac{1}{\hat{\sigma}^2}RSS\right] \end{aligned}$$

It isn't clear what we should do with the $n\log\left(2\pi\right)$ term in the $\log L()$ function. There are some compelling reasons to ignore it and just use the second, and there are reasons to use both terms. Unfortunately, statisticians have not settled on one convention or the other and different software packages might therefore report different values for AIC.

As a general rule of thumb, if the difference in AIC values is less than two then the models are not significantly different, differences between 2 and 4 AIC units are marginally significant and any difference greater than 4 AIC units is highly significant.

Notice that while this allows us to compare models that are not nested, it does require that the same data are used to fit both models. Because I could start out with my data frame including both x and x^2, (or more generally x and $f(x)$ for some function $f()$) you can regard a transformation of a covariate as "the same data". However, a transformation of a y-variable is not and therefore we cannot use AIC to compare a models `log(y) ~ x` versus the model `y ~ x`.

Another criterion that might be used is *Bayes Information Criterion* (BIC) which is

$$BIC = -2 \log L\left(\hat{\boldsymbol{\beta}}, \hat{\sigma} | \text{ data}\right) + p \log n$$

and this criterion punishes large models more than AIC does (because $\log n > 2$ for $n \geq 8$)

The AIC value of a linear model can be found using the AIC() on a `lm()()` object.

```
AIC(m1)
```

```
## [1] 118.6942
```

```
AIC(m2)
```

```
## [1] 124.2947
```

Because the AIC value for the first model is lower, we would prefer the first model that includes both `Income` and `Income.2` compared to model 2, which was `Life.Exp ~ Illiteracy+Murder+Frost`.

7.3.2 Adjusted `R-sq`

One of the problems with R^2 is that it makes no adjustment for how many parameters in the model. Recall that R^2 was defined as

$$R^2 = \frac{RSS_S - RSS_C}{RSS_S} = 1 - \frac{RSS_C}{RSS_S}$$

where the simple model was the intercept only model. We can create an R^2_{adj} statistic that attempts to add a penalty for having too many parameters by defining

$$R^2_{adj} = 1 - \frac{RSS_C / (n-p)}{RSS_S / (n-1)}$$

With this adjusted definition, adding a variable to the model that has no predictive power will decrease R^2_{adj}.

7.3.3 Example

Returning to the life expectancy data, we could start with a simple model add covariates to the model that have the lowest AIC values. R makes this easy with the function `add1()` which will take a linear model (which includes the data frame that originally defined it) and will sequentially add all of the possible terms that are not currently in the model and report the AIC values for each model.

```
# Define the biggest model I wish to consider
biggest <- Life.Exp ~ Log.Population + Income + Illiteracy + Murder +
                      HS.Grad + Frost + Log.Area + HS.Grad.2 + Income.2

# Define the model I wish to start with
m <- lm(Life.Exp ~ 1, data=state.data)

add1(m, scope=biggest)  # what is the best addition to make?
```

```
## Single term additions
##
## Model:
## Life.Exp ~ 1
##                Df Sum of Sq    RSS     AIC
## <none>                      88.299  30.435
## Log.Population  1     1.054 87.245  31.835
## Income          1    10.223 78.076  26.283
## Illiteracy      1    30.578 57.721  11.179
## Murder          1    53.838 34.461 -14.609
## HS.Grad         1    29.931 58.368  11.737
## Frost           1     6.064 82.235  28.878
## Log.Area        1     1.042 87.257  31.842
## HS.Grad.2       1    27.414 60.885  13.848
## Income.2        1     7.464 80.835  28.020
```

Clearly the addition of `Murder` to the model results in the lowest AIC value, so we will add `Murder` to the model. Notice the `<none>` row corresponds to the model m which we started with and it has a `RSS=88.299`. For each model considered, R will calculate the `RSS_{C}` for the new model and will calculate the difference between the starting model and the more complicated model and display this in the Sum of Squares column.

```
m <- update(m, . ~ . + Murder)  # add murder to the model
add1(m, scope=biggest)          # what should I add next?
```

```
## Single term additions
##
## Model:
## Life.Exp ~ Murder
##                Df Sum of Sq    RSS     AIC
## <none>                      34.461 -14.609
## Log.Population  1    2.9854 31.476 -17.140
## Income          1    2.4047 32.057 -16.226
## Illiteracy      1    0.2732 34.188 -13.007
## HS.Grad         1    4.6910 29.770 -19.925
## Frost           1    3.1346 31.327 -17.378
## Log.Area        1    1.4583 33.003 -14.771
## HS.Grad.2       1    4.4396 30.022 -19.505
## Income.2        1    1.8972 32.564 -15.441
```

There is a companion function to `add1()` that finds the best term to drop. It is conveniently named `drop1()` but here the `scope` parameter defines the smallest model to be considered.

It would be nice if all of this work was automated. Again, R makes our life easy and the function `step()` does exactly this. The set of models searched is determined by the scope argument which can be a *list* of two formulas with components upper and lower or it can be a single formula, or it can be blank. The right-hand-side of its lower component defines the smallest model to be considered and the right-hand-side of the upper component defines the largest model to be considered. If `scope` is a single formula, it specifies the upper component, and the lower model taken to be the intercept-only model. If scope is missing, the initial model is used as the upper model.

```
smallest <- Life.Exp ~ 1
biggest <- Life.Exp ~ Log.Population + Income + Illiteracy +
                      Murder + HS.Grad + Frost + Log.Area + HS.Grad.2 + Income.2
m <- lm(Life.Exp ~ Income, data=state.data)
stats::step(m, scope=list(lower=smallest, upper=biggest))
```

```
## Start:  AIC=26.28
```

```
## Life.Exp ~ Income
##
##                   Df Sum of Sq    RSS     AIC
## + Murder           1    46.020 32.057 -16.226
## + Illiteracy       1    21.109 56.968  12.523
## + HS.Grad          1    19.770 58.306  13.684
## + Income.2         1    19.062 59.015  14.288
## + HS.Grad.2        1    17.193 60.884  15.847
## + Frost            1     3.188 74.889  26.199
## <none>                         78.076  26.283
## + Log.Population   1     1.298 76.779  27.445
## + Log.Area         1     0.994 77.082  27.642
## - Income           1    10.223 88.299  30.435
##
## Step:  AIC=-16.23
## Life.Exp ~ Income + Murder
##
##                   Df Sum of Sq    RSS     AIC
## + Frost            1     3.918 28.138 -20.745
## + Income.2         1     3.036 29.021 -19.200
## + HS.Grad          1     2.388 29.668 -18.097
## + Log.Population   1     2.371 29.686 -18.068
## + HS.Grad.2        1     2.199 29.857 -17.780
## <none>                         32.057 -16.226
## + Log.Area         1     1.229 30.827 -16.181
## - Income           1     2.405 34.461 -14.609
## + Illiteracy       1     0.011 32.046 -14.242
## - Murder           1    46.020 78.076  26.283
##
## Step:  AIC=-20.74
## Life.Exp ~ Income + Murder + Frost
##
##                   Df Sum of Sq    RSS     AIC
## + HS.Grad          1     2.949 25.189 -24.280
## + HS.Grad.2        1     2.764 25.375 -23.914
## + Log.Area         1     2.122 26.017 -22.664
## + Income.2         1     2.017 26.121 -22.465
## <none>                         28.138 -20.745
## + Illiteracy       1     0.950 27.189 -20.461
## + Log.Population   1     0.792 27.347 -20.172
## - Income           1     3.188 31.327 -17.378
## - Frost            1     3.918 32.057 -16.226
## - Murder           1    46.750 74.889  26.199
##
## Step:  AIC=-24.28
## Life.Exp ~ Income + Murder + Frost + HS.Grad
##
##                   Df Sum of Sq    RSS     AIC
## + Log.Population   1     2.279 22.911 -27.021
## + Income.2         1     1.864 23.326 -26.124
## - Income           1     0.182 25.372 -25.920
## <none>                         25.189 -24.280
## + Log.Area         1     0.570 24.619 -23.425
## + HS.Grad.2        1     0.218 24.972 -22.714
```

```
## + Illiteracy      1     0.131 25.058 -22.541
## - HS.Grad         1     2.949 28.138 -20.745
## - Frost           1     4.479 29.668 -18.097
## - Murder          1    32.877 58.067  15.478
##
## Step:  AIC=-27.02
## Life.Exp ~ Income + Murder + Frost + HS.Grad + Log.Population
##
##                  Df Sum of Sq    RSS     AIC
## - Income          1     0.011 22.921 -28.998
## <none>                        22.911 -27.021
## + Income.2        1     0.579 22.331 -26.302
## + Log.Area        1     0.207 22.704 -25.475
## + Illiteracy      1     0.052 22.859 -25.134
## + HS.Grad.2       1     0.009 22.901 -25.042
## - Frost           1     2.107 25.017 -24.623
## - Log.Population  1     2.279 25.189 -24.280
## - HS.Grad         1     4.436 27.347 -20.172
## - Murder          1    33.706 56.616  16.214
##
## Step:  AIC=-29
## Life.Exp ~ Murder + Frost + HS.Grad + Log.Population
##
##                  Df Sum of Sq    RSS     AIC
## <none>                        22.921 -28.998
## + Log.Area        1     0.216 22.705 -27.471
## + Illiteracy      1     0.052 22.870 -27.111
## + Income.2        1     0.034 22.887 -27.073
## + HS.Grad.2       1     0.012 22.909 -27.024
## + Income          1     0.011 22.911 -27.021
## - Frost           1     2.214 25.135 -26.387
## - Log.Population  1     2.450 25.372 -25.920
## - HS.Grad         1     6.959 29.881 -17.741
## - Murder          1    34.109 57.031  14.578

##
## Call:
## lm(formula = Life.Exp ~ Murder + Frost + HS.Grad + Log.Population,
##     data = state.data)
##
## Coefficients:
##    (Intercept)          Murder           Frost         HS.Grad
##      68.720810       -0.290016       -0.005174        0.054550
## Log.Population
##       0.246836
```

Notice that our model selected by `step()` is not the same model we obtained when we started with the biggest model and removed things based on p-values.

The log-likelihood is only defined up to an additive constant, and there are different conventional constants used. This is more annoying than anything because all we care about for model selection is the difference between AIC values of two models and the additive constant cancels. The only time it matters is when you have two different ways of extracting the AIC values. Recall the model we fit using the top-down approach was

```
# m1 was
m1 <- lm(Life.Exp ~ Income + Murder + Frost + Income.2, data = state.data)
AIC(m1)
```

```
## [1] 121.4293
```

and the model selected by the stepwise algorithm was

```
m3 <- lm(Life.Exp ~ Murder + Frost + HS.Grad + Log.Population, data = state.data)
AIC(m3)
```

```
## [1] 114.8959
```

Because `step()` and `AIC()` are following different conventions the absolute value of the AICs are different, but the difference between the two is constant no matter which function we use.

First we calculate the difference using the AIC() function:

```
AIC(m1) - AIC(m3)
```

```
## [1] 6.533434
```

and next we use `add1()` on both models to see what the AIC values for each.

```
add1(m1, scope=biggest)
```

```
## Single term additions
##
## Model:
## Life.Exp ~ Income + Murder + Frost + Income.2
##                Df Sum of Sq    RSS     AIC
## <none>                      26.121 -22.465
## Log.Population  1   0.10296 26.018 -20.662
## Illiteracy      1   0.10097 26.020 -20.658
## HS.Grad         1   2.79527 23.326 -26.124
## Log.Area        1   2.36019 23.761 -25.200
## HS.Grad.2       1   2.79698 23.324 -26.127
```

```
add1(m3, scope=biggest)
```

```
## Single term additions
##
## Model:
## Life.Exp ~ Murder + Frost + HS.Grad + Log.Population
##             Df Sum of Sq    RSS     AIC
## <none>                   22.921 -28.998
## Income       1  0.010673 22.911 -27.021
## Illiteracy   1  0.051595 22.870 -27.111
## Log.Area     1  0.215741 22.706 -27.471
## HS.Grad.2    1  0.011894 22.909 -27.024
## Income.2     1  0.034356 22.887 -27.073
```

Using these results, we can calculate the difference in AIC values to be the same as we calculated before

$$\begin{aligned} -22.465 - -28.998 &= -22.465 + 28.998 \\ &= 6.533 \end{aligned}$$

7.4 Exercises

1. Consider the `prostate` data from the `faraway` package. The variable `lpsa` is a measurement of a prostate specific antigen which higher levels are indicative of prostate cancer. Use `lpsa` as the response and all the other variables as predictors (no interactions). Determine the "best" model using:
   a. Backward elimination using the analysis of variance F-statistic as the criteria.
   b. Forward selection using AIC as the criteria.
2. Again from the `faraway` package, use the `divusa` which has divorce rates for each year from 1920-1996 along with other population information for each year. Use `divorce` as the response variable and all other variables as the predictors.
   a. Determine the best model using stepwise selection starting from the intercept only model and the most complex model being all main effects (no interactions). Use the F-statistic to determine significance. Note: add1(), drop1(), and step() allow an option of test='F' to use an F-test instead of AIC.
   b. Following the stepwise selection, comment on the relationship between p-values used and the AIC difference observed. Do the AIC rules of thumb match the p-value interpretation?

Chapter 8

One way ANOVA

```
# Load the libraries I'll use
library(tidyverse)   # dplyr, tidyr, ggplot2
library(emmeans)     # all the pairwise contrasts stuff
library(ggfortify)   # for the autoplot function on lm() objects
```

Given a categorical covariate (which I will call a factor) with I levels, we are interested in fitting the model

$$y_{ij} = \mu + \tau_i + \epsilon_{ij}$$

where $\epsilon_{ij} \overset{iid}{\sim} N\left(0, \sigma^2\right)$, μ is the overall mean, and τ_i are the offset of factor level i from μ. Unfortunately this model is not identifiable because I could add a constant (say 5) to μ and subtract that same constant from each of the τ_i values and the group mean $\mu + \tau_i$ would not change. There are two easy restrictions we could make to make the model identifiable:

1. Set $\mu = 0$. In this case, τ_i represents the expected value of an observation in group level i. We call this the "cell means" representation.
2. Set $\tau_1 = 0$. Then μ represents the expected value of treatment 1, and the τ_i values will represent the offsets from group 1. The group or level that we set to be zero is then referred to as the reference group. We can call this the "offset from reference" model.

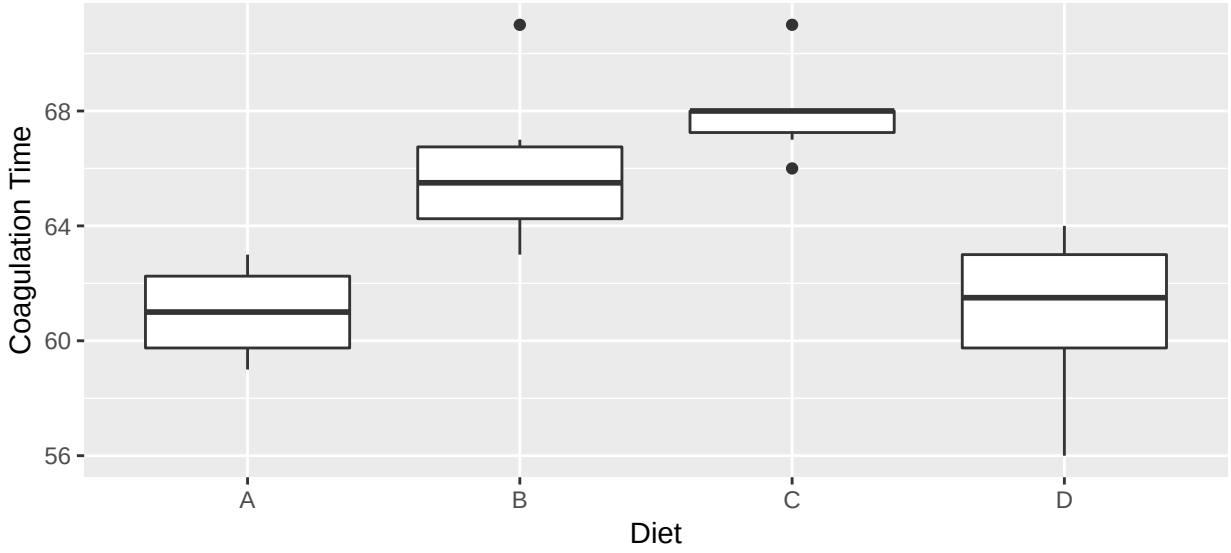
We will be interested in testing the null and alternative hypotheses

$$\begin{aligned} H_0: &\quad y_{ij} = \mu + \epsilon_{ij} \\ H_a: &\quad y_{ij} = \mu + \alpha_i + \epsilon_{ij} \end{aligned}$$

8.1 An Example

We look at a dataset that comes from the study of blood coagulation times: 24 animals were randomly assigned to four different diets and the samples were taken in a random order. The diets are denoted as A,B,C,and D and the response of interest is the amount of time it takes for the blood to coagulate.

```
data('coagulation', package='faraway')
ggplot(coagulation, aes(x=diet, y=coag)) +
    geom_boxplot() +
    labs( x='Diet', y='Coagulation Time' )
```



Just by looking at the graph, we expect to see that diets A and D are similar while B and C are different from A and D and possibly from each other, too. We first fit the offset model.

```
m <- lm(coag ~ diet, data=coagulation)
summary(m)
```

```
##
## Call:
## lm(formula = coag ~ diet, data = coagulation)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -5.00  -1.25   0.00   1.25   5.00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.100e+01  1.183e+00  51.554  < 2e-16 ***
## dietB       5.000e+00  1.528e+00   3.273 0.003803 **
## dietC       7.000e+00  1.528e+00   4.583 0.000181 ***
## dietD       2.991e-15  1.449e+00   0.000 1.000000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.366 on 20 degrees of freedom
## Multiple R-squared:  0.6706, Adjusted R-squared:  0.6212
## F-statistic: 13.57 on 3 and 20 DF,  p-value: 4.658e-05
```

Notice that diet A is the reference level and it has a mean of 61. Diet B has an offset from A of 5, etc. From the very small F-statistic, we conclude that simple model

$$y_{ij} = \mu + \epsilon_{ij}$$

is not sufficient to describe the data.

8.2 Degrees of Freedom

Throughout the previous example, the degrees of freedom that are reported keeps changed depending on what models we are comparing. The simple model we are considering is

$$y_{ij} \sim \mu + \epsilon_{ij}$$

which has 1 parameter that defines the expected value versus

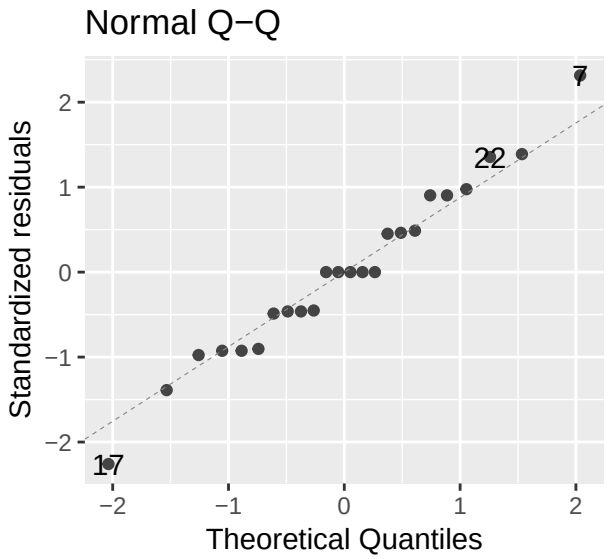
$$y_{ij} \sim \mu + \tau_i + \epsilon_{ij}$$

where there really are only 4 parameters that define the expected value because $\tau_1 = 0$. In general, the larger model is only adding $I - 1$ terms to the model where I is the number of levels of the factor of interest.

8.3 Diagnostics

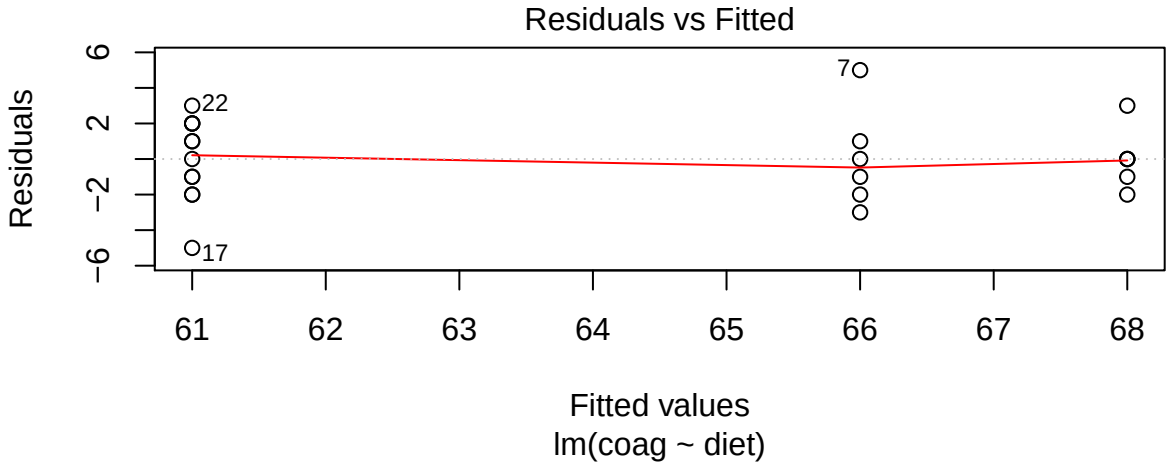
It is still important to check the diagnostics plots, but certain diagnostic plots will be useless. In particular, we need to be concerned about constant variance among the groups and normality of the residuals.

```
m <- lm(coag ~ diet, data=coagulation)
autoplot(m, which=2)  #  QQ plot
```

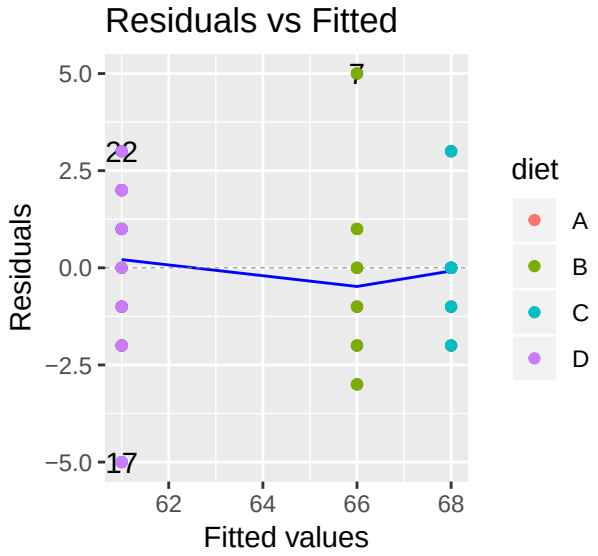


The residual plots however, might need a little bit of extra work, because there are only four possible predicted values (actually 3 because group A and D have the same predicted values). Note that we actually have $n = 24$ observations, but I can only see 16 of them.

```
plot(m, which=1)  #  Residals vs fitted
```

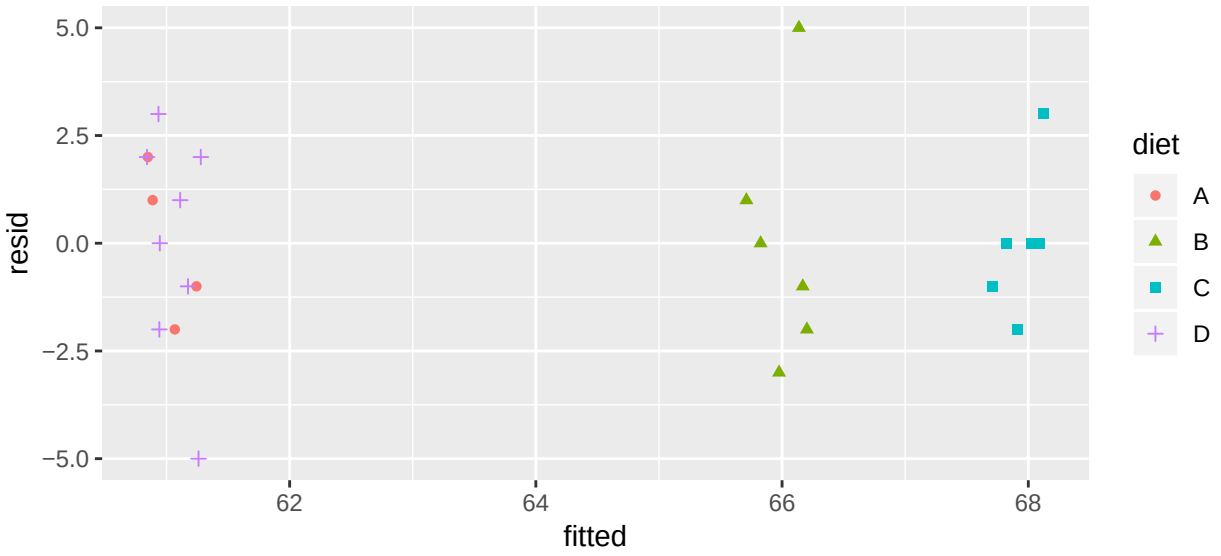



```
autoplot(m, which=1) + geom_point(aes(color=diet))
```



To remedy this, we will plot the residuals vs fitted by hand, and add a little bit of random noise to the fitted value, just so that we don't have points stack up on top of each other. Lets also add a different shape for each diet.

```
coagulation$fitted <- predict(m)
coagulation$resid <- resid(m)
ggplot(coagulation, aes(x=fitted, y=resid, shape=diet, color=diet)) +
  geom_point(position=position_jitter(w=0.3, h=0))
```



8.4 Pairwise Comparisons

After detecting differences in the factor levels, we are often interested in which factor levels are different from which. Often we are interested in comparing the mean of level i with the mean of level j. As usual we let the vector of parameter estimates be $\hat{\boldsymbol{\beta}}$ then the contrast of interested can be written as

$$\boldsymbol{c}^T\hat{\boldsymbol{\beta}} \pm t_{n-p}^{1-\alpha/2}\,\hat{\sigma}\sqrt{\boldsymbol{c}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{c}}$$

for some vector $\boldsymbol{c}$.

Unfortunately this interval does not take into account the multiple comparisons issue (i.e. we are making $I(I-1)/2$ contrasts if our factor has I levels). To account for this, we will not use a quantile from a t-distribution, but from Tukey's studentized range distribution $q_{n,n-I}$ divided by $\sqrt{2}$. The intervals we will use are:

$$\boldsymbol{c}^T\hat{\boldsymbol{\beta}} \pm \frac{q_{n,n-I}^{1-\alpha/2}}{\sqrt{2}}\,\hat{\sigma}\sqrt{\boldsymbol{c}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{c}}$$

There are several ways to make R calculate this interval (See the contrasts chapter for a more general treatment of this.), but the easiest is to use the `emmeans` package. This package computes the above intervals which are commonly known as Tukey's Honestly Significant Differences.

```
m <- lm(coag ~ diet, data=coagulation)   # use the lm() function as usual
emmeans(m, specs= pairwise~diet) %>%
  summary(level=0.90)
```

```
## $emmeans
##  diet emmean        SE df lower.CL upper.CL
##  A        61 1.1832160 20 58.95929 63.04071
##  B        66 0.9660918 20 64.33376 67.66624
##  C        68 0.9660918 20 66.33376 69.66624
##  D        61 0.8366600 20 59.55700 62.44300
##
## Confidence level used: 0.9
##
## $contrasts
```

```
##  contrast      estimate       SE df t.ratio p.value
##  A - B    -5.000000e+00 1.527525 20  -3.273  0.0183
##  A - C    -7.000000e+00 1.527525 20  -4.583  0.0010
##  A - D    -2.991428e-15 1.449138 20   0.000  1.0000
##  B - C    -2.000000e+00 1.366260 20  -1.464  0.4766
##  B - D     5.000000e+00 1.278019 20   3.912  0.0044
##  C - D     7.000000e+00 1.278019 20   5.477  0.0001
##
## P value adjustment: tukey method for comparing a family of 4 estimates
```

Here we see that diets A and D are similar to each other, but different than B and C and that B and C are not statistically different from each other at the 0.10 level.

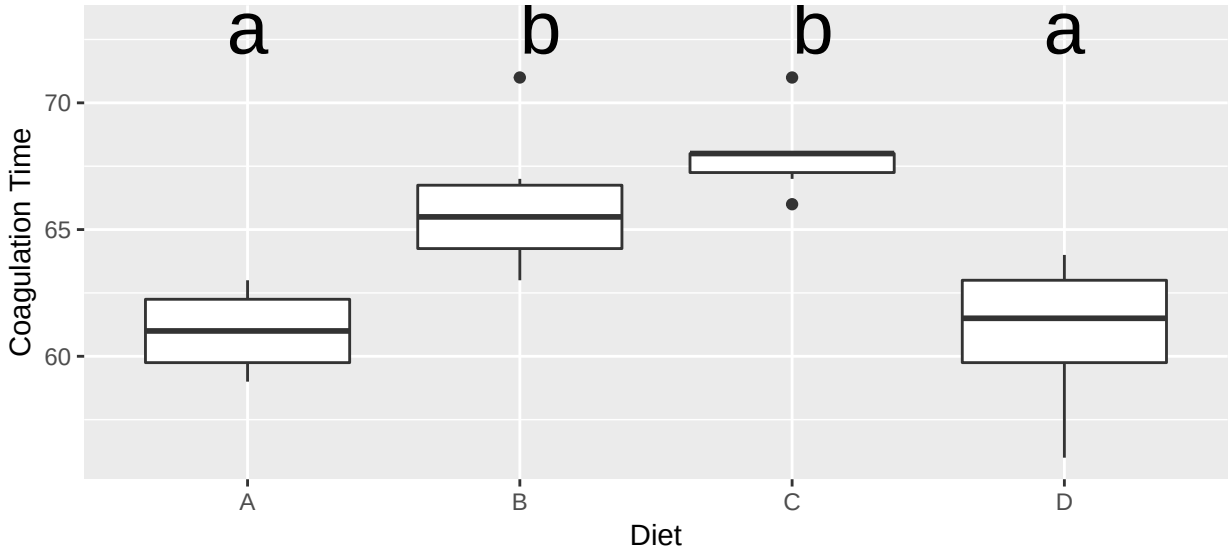
Often I want to produce the "Compact Letter Display" which identifies which groups are significantly different.

```
LetterData <- emmeans(m, specs= ~ diet) %>%  # cld() will freak out if you have pairwise here...
  cld(Letters=letters, level=0.95) %>%
  mutate( y = 73 )   # height to place the letters at.
LetterData
```

```
##   diet emmean        SE df lower.CL upper.CL .group  y
## 1    A     61 1.1832160 20 58.53185 63.46815     a  73
## 2    D     61 0.8366600 20 59.25476 62.74524     a  73
## 3    B     66 0.9660918 20 63.98477 68.01523      b 73
## 4    C     68 0.9660918 20 65.98477 70.01523      b 73
```

I can easily add these to my boxplot with the following:

```
ggplot(coagulation, aes(x=diet, y=coag)) +
    geom_boxplot() +
    labs( x='Diet', y='Coagulation Time' ) +
  geom_text( data=LetterData, aes(x=diet, y=y, label=.group), size=10 )
```



8.5 Exercises

1. Use the dataset `chickwts` in the `datasets` package. This was an experiment to determine which feed types result in the largest chickens. A set of 71 chicks were all randomly assigned one of six feed types and their weight in grams after six weeks was recorded. Determine whether there are differences in the weights of chickens according to their feed. Perform all necessary model diagnostics and examine the contrasts between each pair of feed levels. Summarize these results.

Chapter 9

Two-way ANOVA

```
# Load my usual packages
library(tidyverse)    # ggplot2, dplyr, tidyr
library(ggfortify)    # autoplot() for lm objects
library(emmeans)      # pairwise contrasts stuff
```

Given a response that is predicted by two different categorical variables. Suppose we denote the levels of the first factor as α_i and has I levels. The second factor has levels β_j and has J levels. As usual we let $\epsilon_{ijk} \overset{iid}{\sim} N\left(0, \sigma^2\right)$, and we wish to fit the model

$$y_{ijk} = \mu + \alpha_i + \beta_j + \epsilon_{ijk}$$

which has the main effects of each covariate or possibly the model with the interaction

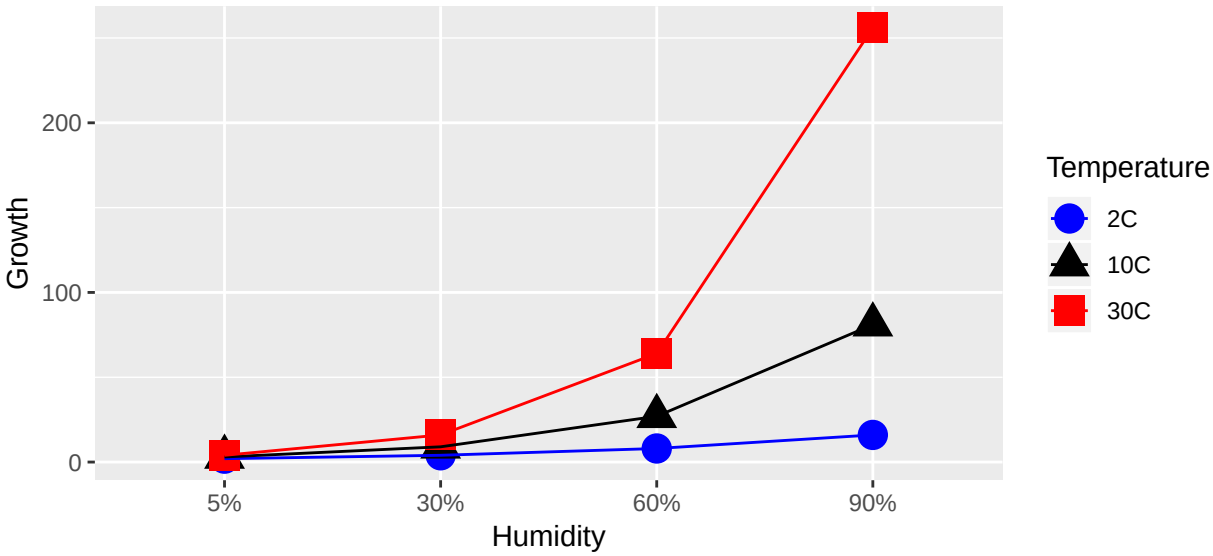
$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}$$

To consider what an interaction term might mean consider the role of temperature and humidity on the amount of fungal growth. You might expect to see data similar to this (where the numbers represent some sort of measure of fungal growth):

		5%	30%	60%	90%
Temperature	**2C**	2	4	8	16
	10C	3	9	27	81
	30C	4	16	64	256

In this case we see that increased humidity increases the amount of fungal growth, but the amount of increase depends on the temperature. At 2 C, the increase is humidity increases are significant, but at 10 C the increases are larger, and at 30 C the increases are larger yet. The effect of changing from one humidity level to the next *depends on which temperature level we are at.* This change in effect of humidity is an interaction effect. A memorable example is that chocolate by itself is good. Strawberries by themselves are also good. But the combination of chocolate and strawberries is a delight greater than the sum of the individual treats.

We can look at a graph of the Humidity and Temperature vs the Response and see the effect of increasing humidity changes based on the temperature level. Just as in the ANCOVA model, the interaction manifested itself in non-parallel slopes, the interaction manifests itself in non-parallel slopes when I connect the dots across the factor levels.



Unfortunately the presence of a significant interaction term in the model makes interpretation difficult, but examining the interaction plots can be quite helpful in understanding the effects. Notice in this example, we 3 levels of temperature and 4 levels of humidity for a total of 12 different possible treatment combinations. In general I will refer to these combinations as cells.

9.1 Orthogonality

When designing an experiment, I want to make sure than none of my covariates are confounded with each other and I'd also like for them to not be correlated. Consider the following three experimental designs, where the number in each bin is the number of subjects of that type. I am interested in testing 2 different drugs and studying its effect on heart disease within the gender groups.

Design 1	Males	Females	Design 2	Males	Females
Treatment A	0	10	Treatment A	1	9
Treatment B	6	0	Treatment B	5	1

Design 3	Males	Females	Design 4	Males	Females
Treatment A	3	5	Treatment A	4	4
Treatment B	3	5	Treatment B	4	4

1. This design is very bad. Because we have no males taking drug 1, and no females taking drug 2, we can't say if any observed differences are due to the effect of drug 1 versus 2, or gender. When this situation happens, we say that the gender effect is confounded with the drug effect.
2. This design is not much better. Because we only have one observation in the Male-Drug 1 group, any inference we make about the effect of drug 1 on males is based on one observation. In general that is a bad idea.
3. Design 3 is better than the previous 2 because it evenly distributes the males and females among the two drug categories. However, it seems wasteful to have more females than males because estimating average of the male groups, I only have 6 observations while I have 10 females.
4. This is the ideal design, with equal numbers of observations in each gender-drug group.

Designs 3 and 4 are good because the correlation among my predictors is 0. In design 1, the drug covariate

is perfectly correlated to the gender covariate. The correlation is less in design 2, but is zero in designs 3 and 4.We could show this by calculating the design matrix for each design and calculating the correlation coefficients between each of pairs of columns.

Having an orthogonal design with equal numbers of observations in each group has many nice ramifications. Most importantly, with an orthogonal design, the interpretation of parameter is not dependent on what other factors are in the model. Balanced designs are also usually optimal in the sense that the variances of $\hat{\boldsymbol{\beta}}$ are as small as possible given the number of observations we have (barring any other *a priori* information).

9.2 Main Effects Model

In the one factor ANOVA case, the additional degrees of freedom used by adding a factor with I levels was $I-1$. In the case that we consider two factors with the first factor having I levels and the second factor having J levels, then model

$$y_{ijk} = \mu + \alpha_i + \beta_j + \epsilon_{ijk}$$

adds $(I-1)+(J-1)$ parameters to the model because both $\alpha_1 = \beta_1 = 0$.

- The intercept term, μ is the reference point for all the other parameters. This is the expected value for an observation in the first level of factor 1 and the first level of factor two.
- α_i is the amount you expect the response to increase when changing from factor 1 level 1, to factor 1 level i (while the second factor is held constant).
- β_j is the amount you expect the response to increase when changing from factor 2 level 1 to factor 2 level j (while the first factor is held constant).

Referring back to the fungus example, let the α_i values be associated with changes in humidity and β_j values be associated with changes in temperature levels. Then the expected value of each treatment combination is

	5%	**30%**	**60%**	**90%**
2C	$\mu+0+0$	$\mu+\alpha_2+0$	$\mu+\alpha_3+0$	$\mu+\alpha_4+0$
10C	$\mu+0+\beta_2$	$\mu+\alpha_2+\beta_2$	$\mu+\alpha_3+\beta_2$	$\mu+\alpha_4+\beta_2$
30C	$\mu+0+\beta_3$	$\mu+\alpha_2+\beta_3$	$\mu+\alpha_3+\beta_3$	$\mu+\alpha_4+\beta_3$

9.2.1 Example - Fruit Trees

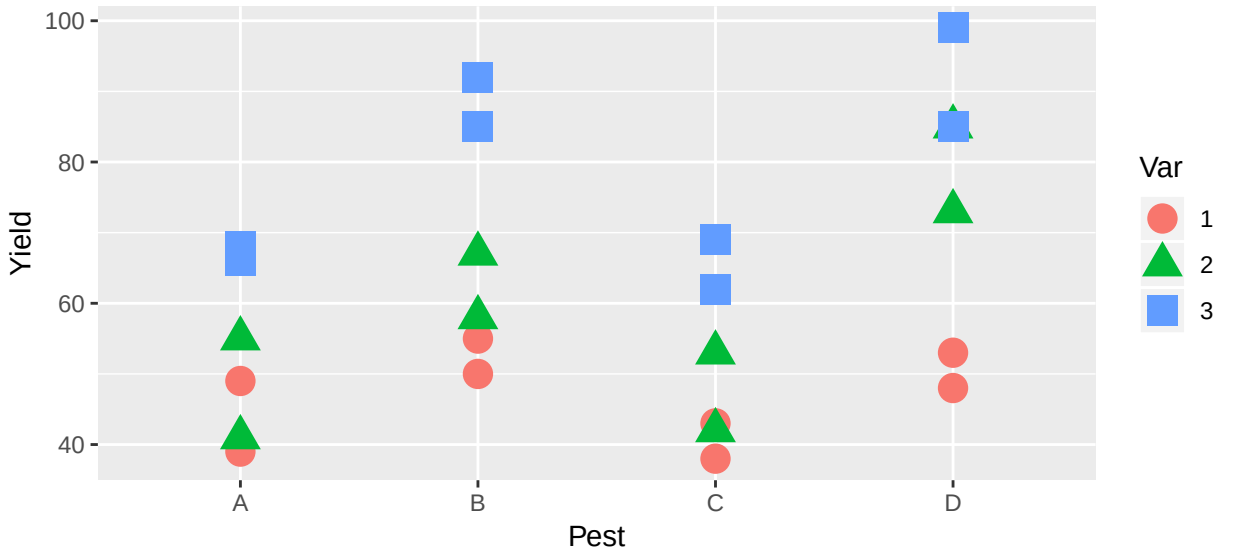
An experiment was conducted to determine the effects of four different pesticides on the yield of fruit from three different varieties of a citrus tree. Eight trees of each variety were randomly selected from an orchard. The four pesticides were randomly assigned to two trees of each variety and applications were made according to recommended levels. Yields of fruit (in bushels) were obtained after the test period.

Critically notice that we have equal number of observations for each treatment combination.

```
# Typing the data in by hand because I got this example from a really old text book...
Pesticide <- factor(c('A','B','C','D'))
Variety <- factor(c('1','2','3'))
fruit <- data.frame( expand.grid(rep=1:2, Pest=Pesticide, Var=Variety) )
fruit$Yield <- c(49,39,50,55,43,38,53,48,55,41,67,58,53,42,85,73,66,68,85,92,69,62,85,99)
```

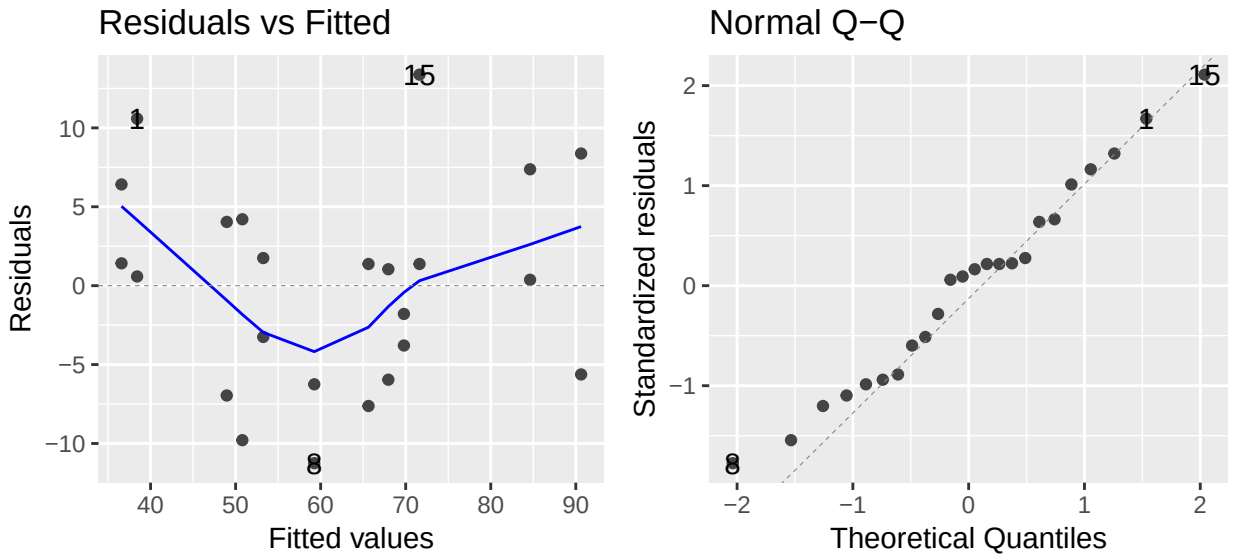
The first thing to do (as always) is to look at our data

```
ggplot(fruit, aes(x=Pest, color=Var, y=Yield, shape=Var)) +
    geom_point(size=5)
```



The first thing we notice is that pesticides B and D seem to be better than the others and that variety 3 seems to be the best producer. The effect of pesticide treatment seems consistent between varieties, so we don't expect that the interaction effect will be significant. We next fit a linear model and look at the diagnostic plots.

```
m3 <- lm(Yield ~ Var + Pest, data=fruit)
autoplot(m3, which=1:2)
```



There might be a little curvature in the fitted vs residuals, but because we can't fit a polynomial to a categorical variable, and the QQ-plot looks good, we'll ignore it for now and eventually consider an interaction term. Just for fun, we can examine the smaller models with just Variety or Pesticide.

```
m1 <- lm(Yield ~ Var, data=fruit)
m2 <- lm(Yield ~ Pest, data=fruit)
m3 <- lm(Yield ~ Var + Pest, data=fruit)
summary(m1)$coef  %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   46.875      4.359  10.754    0.000
## Var2          12.375      6.164   2.008    0.058
```

```
## Var3          31.375      6.164   5.090    0.000
```

```
summary(m2)$coef  %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   53.000      6.429   8.243    0.000
## PestB         14.833      9.093   1.631    0.118
## PestC         -1.833      9.093  -0.202    0.842
## PestD         20.833      9.093   2.291    0.033
```

```
summary(m3)$coef  %>% round(digits=3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   38.417      3.660  10.497    0.000
## Var2          12.375      3.660   3.381    0.003
## Var3          31.375      3.660   8.573    0.000
## PestB         14.833      4.226   3.510    0.003
## PestC         -1.833      4.226  -0.434    0.670
## PestD         20.833      4.226   4.930    0.000
```

Notice that the affects for Variety and Pesticide are the same *whether or not the other is in the model*. This is due to the orthogonal design of the experiment and makes it much easier to interpret the main effects of Variety and Pesticide.

9.2.2 ANOVA Table

Most statistical software will produce an analysis of variance table when fitting a two-way ANOVA. This table is very similar to the analysis of variance table we have seen in the one-way ANOVA, but has several rows which correspond to the additional factors added to the model.

Consider the two-way ANOVA with factors A and B which have levels I and J discrete levels respectively. For convenience let RSS_1 is the residual sum of squares of the intercept-only model, and RSS_A be the residual sum of squares for the model with just the main effect of factor A, and RSS_{A+B} be the residual sum of squares of the model with both main effects. Finally assume that we have a total of n observations. The ANOVA table for this model is as follows:

Source	df	Sum of Sq (SS)	Mean Sq	F p-value
A	$df_A = I-1$	$SS_A = RSS_1 - RSS_A$	$MS_A = SS_A/df_A$	$MS_A/MSEP\,(F_{df_A,df_e} > F_A)$
B	$df_B = J-1$	$SS_B = RSS_A - RSS_{A+B}$	$MS_B = SS_B/df_B$	$MS_B/MSEP\,(F_{df_B,df_e} > F_B)$
Error	$df_e = n-I-J+1$	RSS_{A+B}	$MSE = RSS_{A+B}/df_e$	

Note, if the table is cut off, you can change decrease your font size and have it all show up…

This arrangement of the ANOVA table is referred to as "Type I" sum of squares.

We can examine this table in the fruit trees example using the `anova()`() command but just passing a single model.

```
m4 <- lm(Yield ~ Var + Pest, data=fruit)
anova( m4 )
```

```
## Analysis of Variance Table
##
```

```
## Response: Yield
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Var        2 3996.1 1998.04  37.292 3.969e-07 ***
## Pest       3 2227.5  742.49  13.858 6.310e-05 ***
## Residuals 18  964.4   53.58
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We might think that this is the same as fitting three nested models and running an F-test on each successive pairs of models, but it isn't. While both will give the same Sums of Squares, the F statistics are different because the MSE of the complex model is different. In particular, the F-statistics are larger and thus the p-values are smaller for detecting significant effects.

```
m1 <- lm(Yield ~ 1, data=fruit)
m2 <- lm(Yield ~ Var, data=fruit)
m3 <- lm(Yield ~ Var + Pest, data=fruit)
anova( m1, m2 )
```

```
## Analysis of Variance Table
##
## Model 1: Yield ~ 1
## Model 2: Yield ~ Var
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     23 7188.0
## 2     21 3191.9  2    3996.1 13.146 0.0001987 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova( m2, m3 )
```

```
## Analysis of Variance Table
##
## Model 1: Yield ~ Var
## Model 2: Yield ~ Var + Pest
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1     21 3191.9
## 2     18  964.4  3    2227.5 13.858 6.31e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

9.2.3 Estimating Contrasts

As in the one-way ANOVA, we are interested in which factor levels differ. For example, we might suspect that it makes sense to group pesticides B and D together and claim that they are better than the group of A and C.

Just as we did in the one-way ANOVA model, this is such a common thing to do that there is an easy way to do this, using `emmeans`.

```
m3 <- lm(Yield ~ Var + Pest, data=fruit)
emmeans(m3, spec=pairwise~Var)
```

```
## $emmeans
##  Var emmean       SE df lower.CL upper.CL
##  1   46.875 2.587922 18 41.43798 52.31202
##  2   59.250 2.587922 18 53.81298 64.68702
```

```
##  3   78.250 2.587922 18 72.81298 83.68702
##
## Results are averaged over the levels of: Pest
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate       SE df t.ratio p.value
##  1 - 2     -12.375 3.659874 18  -3.381  0.0089
##  1 - 3     -31.375 3.659874 18  -8.573  <.0001
##  2 - 3     -19.000 3.659874 18  -5.191  0.0002
##
## Results are averaged over the levels of: Pest
## P value adjustment: tukey method for comparing a family of 3 estimates
```

```
emmeans(m3, spec=pairwise~Pest)
```

```
## $emmeans
##  Pest   emmean       SE df lower.CL upper.CL
##  A    53.00000 2.988274 18 46.72187 59.27813
##  B    67.83333 2.988274 18 61.55520 74.11146
##  C    51.16667 2.988274 18 44.88854 57.44480
##  D    73.83333 2.988274 18 67.55520 80.11146
##
## Results are averaged over the levels of: Var
## Confidence level used: 0.95
##
## $contrasts
##  contrast   estimate       SE df t.ratio p.value
##  A - B    -14.833333 4.226058 18  -3.510  0.0122
##  A - C      1.833333 4.226058 18   0.434  0.9719
##  A - D    -20.833333 4.226058 18  -4.930  0.0006
##  B - C     16.666667 4.226058 18   3.944  0.0048
##  B - D     -6.000000 4.226058 18  -1.420  0.5038
##  C - D    -22.666667 4.226058 18  -5.364  0.0002
##
## Results are averaged over the levels of: Var
## P value adjustment: tukey method for comparing a family of 4 estimates
```

These outputs are nice and they show the main effects of variety and pesticide. Similar to the 1-way ANOVA, we also want to be able to calculate the compact letter display.

```
m3 <- lm(Yield ~ Var + Pest, data=fruit)
emmeans(m3, spec= ~Var) %>% cld(Letters=letters)
```

```
##  Var emmean       SE df lower.CL upper.CL .group
##  1   46.875 2.587922 18 41.43798 52.31202  a
##  2   59.250 2.587922 18 53.81298 64.68702   b
##  3   78.250 2.587922 18 72.81298 83.68702    c
##
## Results are averaged over the levels of: Pest
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 3 estimates
## significance level used: alpha = 0.05
```

```
emmeans(m3, spec= ~Pest) %>% cld(Letters=letters)
```

```
##  Pest   emmean       SE df lower.CL upper.CL .group
```

```
##  C    51.16667 2.988274 18 44.88854 57.44480  a
##  A    53.00000 2.988274 18 46.72187 59.27813  a
##  B    67.83333 2.988274 18 61.55520 74.11146   b
##  D    73.83333 2.988274 18 67.55520 80.11146   b
##
## Results are averaged over the levels of: Var
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 4 estimates
## significance level used: alpha = 0.05
```

So we see that each variety is significantly different from all the others and among the pesticides, A and C are indistinguishable as are B and D, but there is a difference between the A, C and B, D groups.

9.3 Interaction Model

When the model contains the interaction of the two factors, our model is written as

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}$$

Interpreting effects effects can be very tricky. Under the interaction, the effect of changing from factor 1 level 1 to factor 1 level i depends on what level of factor 2 is. In essence, we are fitting a model that allows each of the $I \times J$ cells in my model to vary independently. As such, the model has a total of $I \times J$ parameters but because the model without interactions had $1 + (I-1) + (J-1)$ terms in it, the interaction is adding df_{AB} parameters. We can solve for this via:

$$\begin{aligned}
I \times J &= 1 + (I-1) + (J-1) + df_{AB} \\
I \times J &= I + J - 1 + df_{AB} \\
IJ - I - J &= -1 + df_{AB} \\
I(J-1) - J &= -1 + df_{AB} \\
I(J-1) - J + 1 &= df_{AB} \\
I(J-1) - (J-1) &= df_{AB} \\
(I-1)(J-1) &= df_{AB}
\end{aligned}$$

This makes sense because the first factor added $(I-1)$ columns to the design matrix and an interaction with a continuous covariate just multiplied the columns of the factor by the single column of the continuous covariate. Creating an interaction of two factors multiplies each column of the first factor by all the columns defined by the second factor.

The expected value of the ij combination is $\mu + \alpha_i + \beta_j + (\alpha\beta)_{ij}$. Returning to our fungus example, the expected means for each treatment under the model with main effects and the interaction is

	5%	**30%**	**60%**	**90%**
2C	$\mu + 0 + 0 + 0$	$\mu + \alpha_2 + 0 + 0$	$\mu + \alpha_3 + 0 + 0$	$\mu + \alpha_4 + 0 + 0$
10C	$\mu + 0 + \beta_2 + 0$	$\mu + \alpha_2 + \beta_2 + (\alpha\beta)_{22}$	$\mu + \alpha_3 + \beta_2 + (\alpha\beta)_{32}$	$\mu + \alpha_4 + \beta_2 + (\alpha\beta)_{42}$
30C	$\mu + 0 + \beta_3 + 0$	$\mu + \alpha_2 + \beta_3 + (\alpha\beta)_{23}$	$\mu + \alpha_3 + \beta_2 + (\alpha\beta)_{33}$	$\mu + \alpha_4 + \beta_2 + (\alpha\beta)_{43}$

Notice that we have added $6 = 3 \cdot 2 = (4-1)(3-1) = (I-1)(J-1)$ interaction parameters $(\alpha\beta)_{ij}$ to the

main effects only model. The interaction model has $p = 12$ parameters, one for each cell in my treatment array.

In general it is hard to interpret the meaning of α_i, β_j, and $(\alpha\beta)_{ij}$ and the best way to make sense of them is to look at the interaction plots.

9.3.1 ANOVA Table

Most statistical software will produce an analysis of variance table when fitting a two-way ANOVA. This table is very similar to the analysis of variance table we have seen in the one-way ANOVA, but has several rows which correspond to the additional factors added to the model.

Consider the two-way ANOVA with factors A and B which have levels I and J discrete levels respectively. For convenience let RSS_1 be the residual sum of squares of the intercept-only model, and RSS_A be the residual sum of squares for the model with just the main effect of factor A. Likewise RSS_{A+B} and RSS_{A*B} shall be the residual sum of squares of the model with just the main effects and the model with main effects and the interaction. Finally assume that we have a total of n observations. The ANOVA table for this model is as follows:

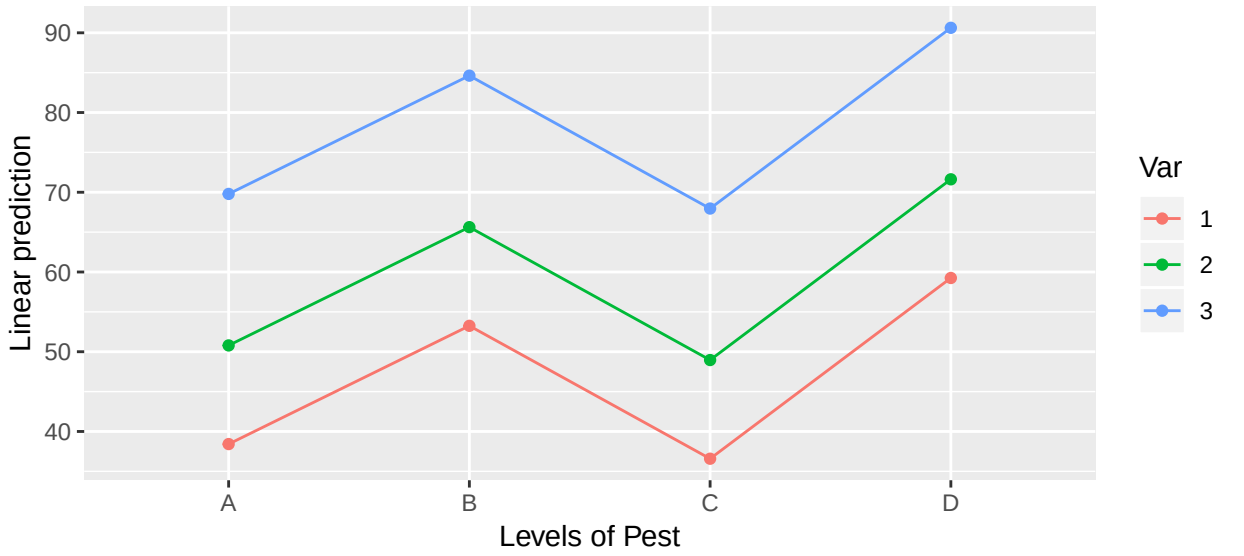
	df	Sum Sq (SS)	MS	F	$Pr(\geq F)$
A	$df_A = I - 1$	$SS_A = RSS_1 - RSS_A$	$MS_A = SS_A/df_A$	MS_A/MSE	$Pr(F_{df_A,df_\epsilon} \geq F_A$
B	$df_B = J - 1$	$SS_B = RSS_A - RSS_{A+B}$	$MS_B = SS_B/df_B$	MS_B/MSE	$Pr(F_{df_B,df_\epsilon} \geq F_B$
AB	$df_{AB} = (I-1)(J-1)$	$SS_{A*B} = RSS_{A+B} - RSS_{A*B}$	$MS_{AB} = SS_{AB}/df_{AB}$	MS_{AB}/MSE	$Pr(F_{df_{AB},df_\epsilon} \geq F_{AB}$
Error	$df_\epsilon = n-IJ$	RSS_{A*B}	$MSE = RSS_{A*B}/df_\epsilon$		

This arrangement of the ANOVA table is referred to as "Type I" sum of squares. Type III sums of squares are the difference between the full interaction model and the model removing each parameter group, even when it doesn't make sense. For example in the Type III table, $SS_A = RSS_{B+A:B} - RSS_{A*B}$. There is an intermediate form of the sums of squares called Type II, that when removing a main effect also removes the higher order interaction. In the case of balanced (orthogonal) designs, there is no difference between the different types, but for non-balanced designs, the numbers will change. To access these other types of sums of squares, use the `Anova()` function in the package `car`.

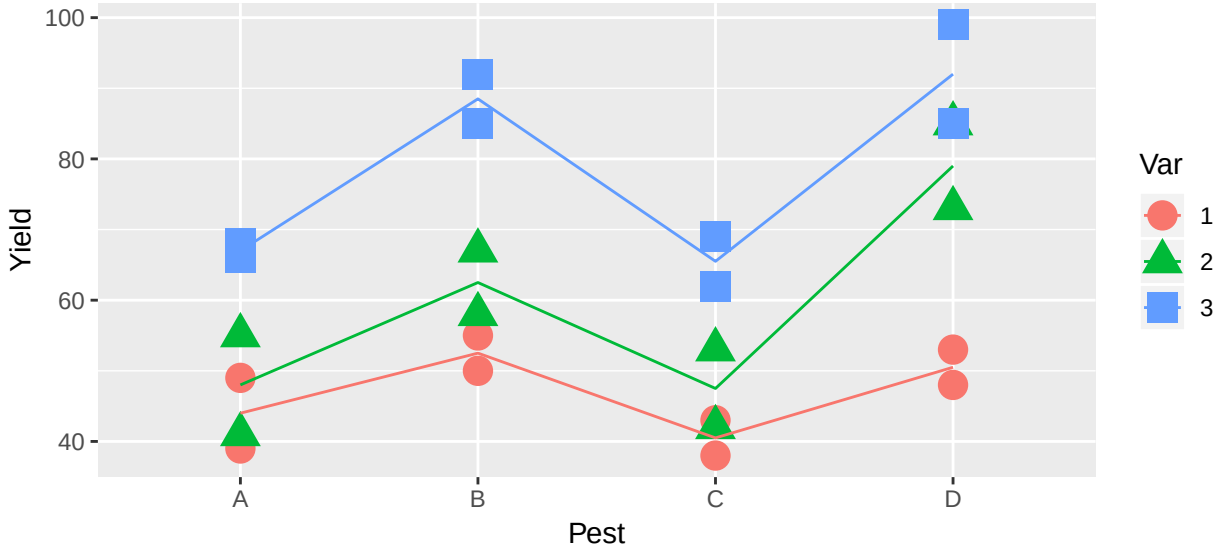
9.3.2 Example - Fruit Trees (continued)

We next consider whether or not to include the interaction term to the fruit tree model. We fit the model with the interaction and then graph the results.

```
# Create the Interaction Plot using emmeans
emmip( m4, Var ~ Pest )  # color is LHS of the formula
```




```
# Create the intercation plot by hand
m4 <- lm(Yield ~ Var * Pest, data=fruit)
fruit$y.hat <- predict(m4)
ggplot(fruit, aes(x=Pest, color=Var, shape=Var, y=Yield)) +
    geom_point(size=5) +
    geom_line(aes(y=y.hat, x=as.integer(Pest)))
```



All of the line segments are close to parallel so, we don't expect the interaction to be significant.

```
anova( m4 )
```

```
## Analysis of Variance Table
##
## Response: Yield
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Var        2 3996.1 1998.04 47.2443 2.048e-06 ***
## Pest       3 2227.5  742.49 17.5563 0.0001098 ***
## Var:Pest   6  456.9   76.15  1.8007 0.1816844
## Residuals 12  507.5   42.29
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

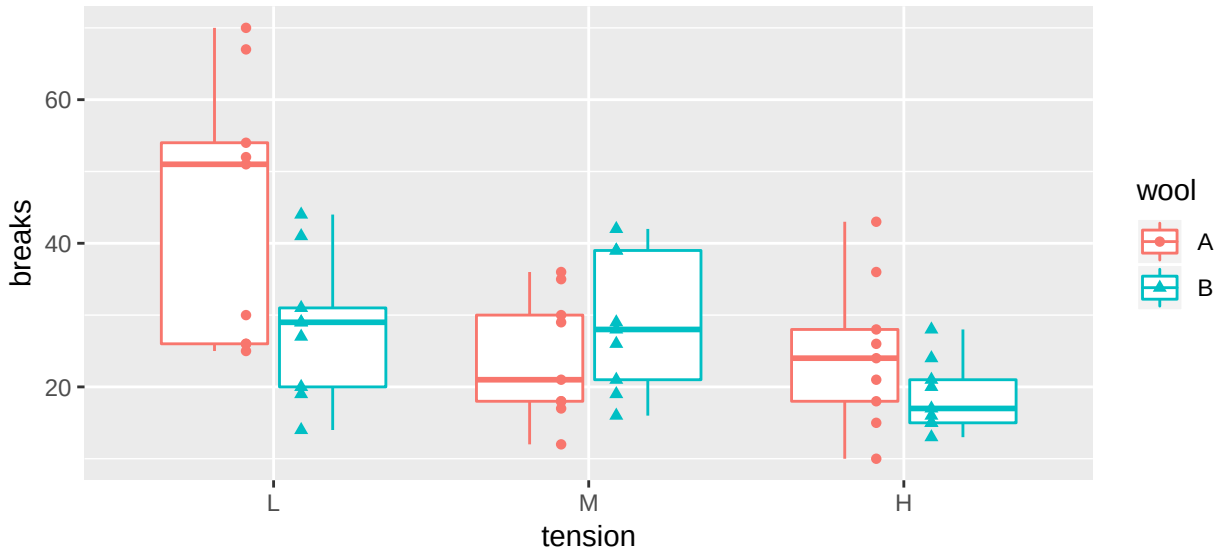
Examining the ANOVA table, we see that the interaction effect is not significant and we will stay with simpler model `Yield~Var+Pest`.

9.3.3 Example - Warpbreaks

This data set looks at the number of breaks that occur in two different types of wool under three different levels of tension (low, medium, and high). The fewer number of breaks, the better.

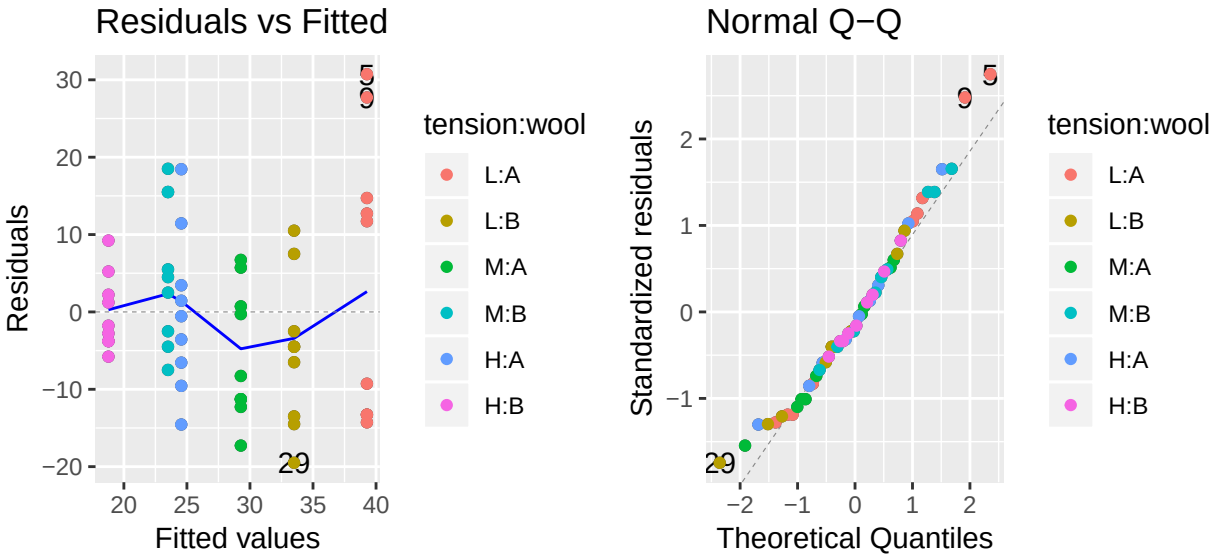
As always, the first thing we do is look at the data. In this case, it looks like the number of breaks decreases with increasing tension and perhaps wool B has fewer breaks than wool A.

```
data(warpbreaks)
ggplot(warpbreaks, aes(x=tension, y=breaks, color=wool, shape=wool), size=2) +
  geom_boxplot() +
  geom_point(position=position_dodge(width=.35)) # offset the wool groups
```



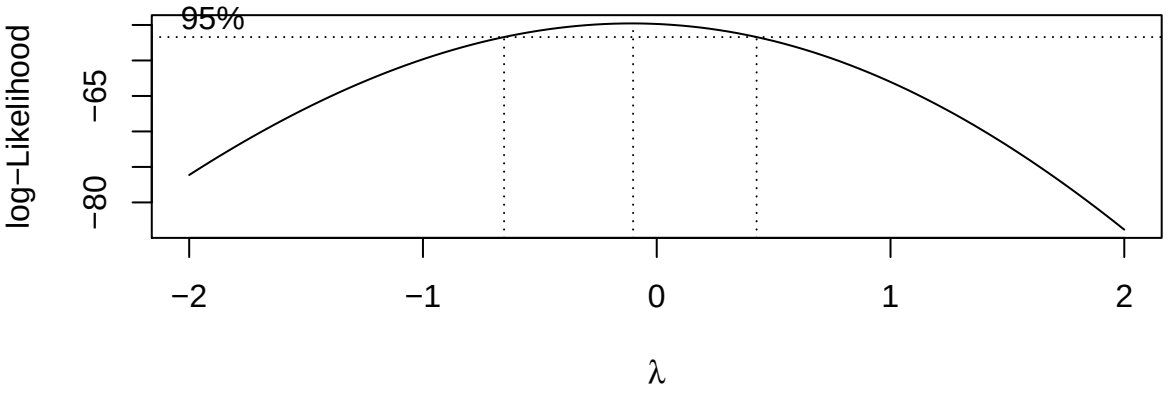
We next fit our linear model and examine the diagnostic plots.

```
model <- lm(breaks ~ tension + wool, data=warpbreaks)
autoplot(model, which=c(1,2)) + geom_point( aes(color=tension:wool))
```



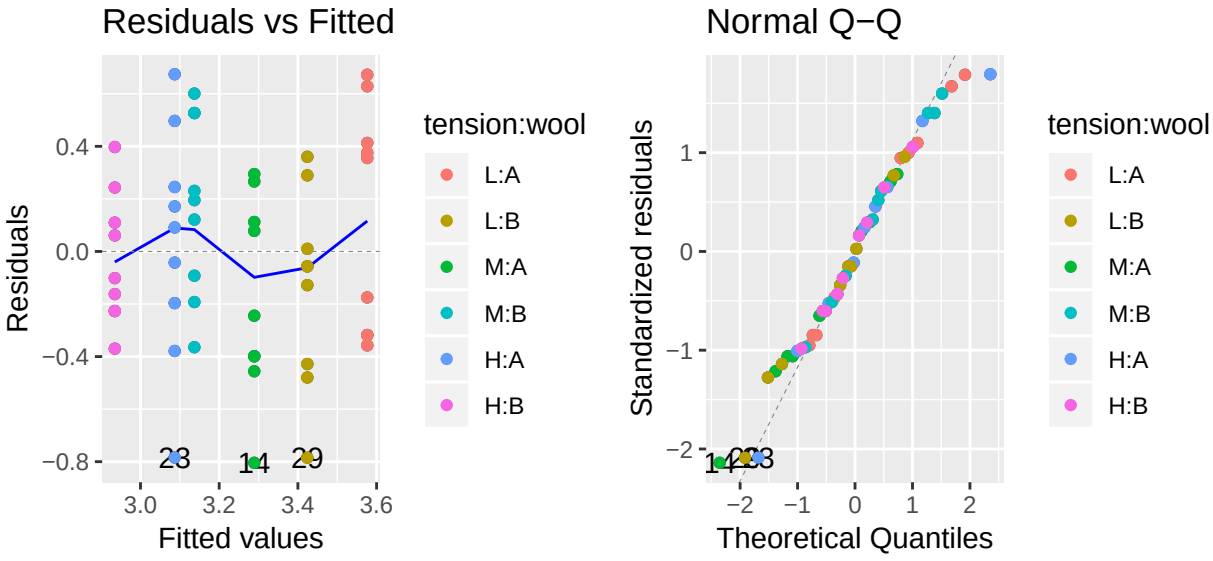
The residuals vs fitted values plot is a little worrisome and appears to be an issue with non-constant variance, but the normality assumption looks good. We'll check for a Box-Cox transformation next.

```
MASS::boxcox(model)
```



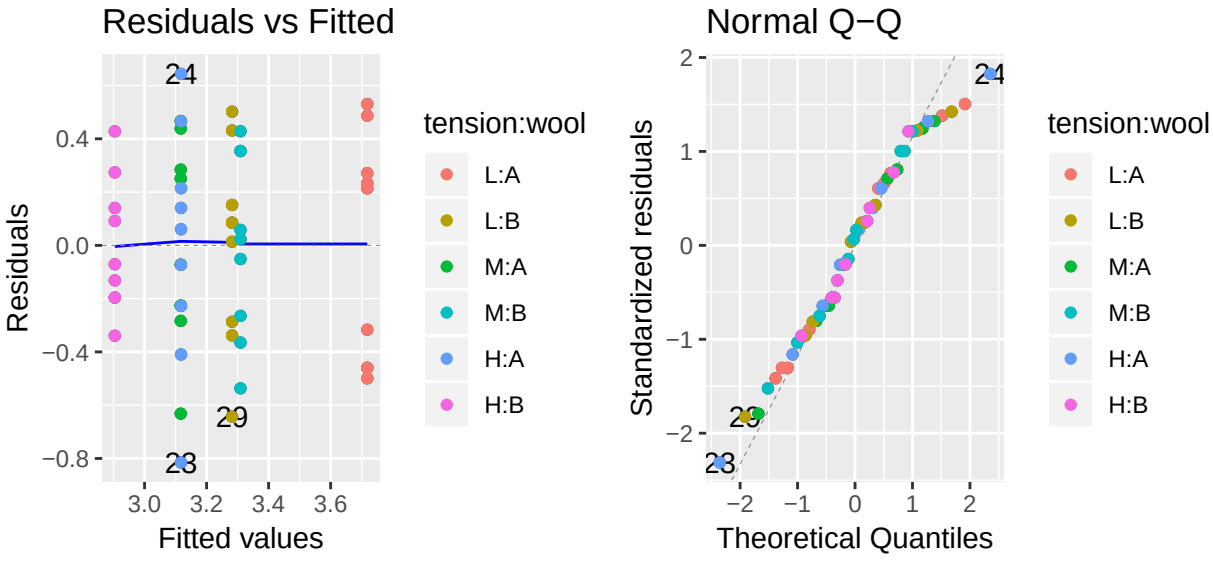
This suggests we should make a log transformation, though because the confidence interval is quite wide we might consider if the increased difficulty in interpretation makes sufficient progress towards making the data meet the model assumptions. The diagnostic plots of the resulting model look better for the constant variance assumption, but the normality is now a worse off. Because the Central Limit Theorem helps deal with the normality question, I'd rather stabilize the variance at the cost of the normality.

```
model.1 <- lm(log(breaks) ~ tension + wool, data=warpbreaks)
autoplot(model.1, which=c(1,2)) + geom_point( aes(color=tension:wool))
```



Next we'll fit the interaction model and check the diagnostic plots. The diagnostic plots look good and this appears to be a legitimate model.

```
model.2 <- lm(log(breaks) ~ tension * wool, data=warpbreaks)
autoplot(model.2, which=c(1,2)) + geom_point( aes(color=tension:wool))
```



Then we'll do an F-test to see if it is a better model than the main effects model. The p-value is marginally significant, so we'll keep the interaction in the model, but recognize that it is a weak interaction.

```
anova(model.1, model.2)  # explicitly look model1 vs model2
```

```
## Analysis of Variance Table
##
## Model 1: log(breaks) ~ tension + wool
## Model 2: log(breaks) ~ tension * wool
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     50 7.6270
## 2     48 6.7138  2   0.91315 3.2642 0.04686 *
## ---
```

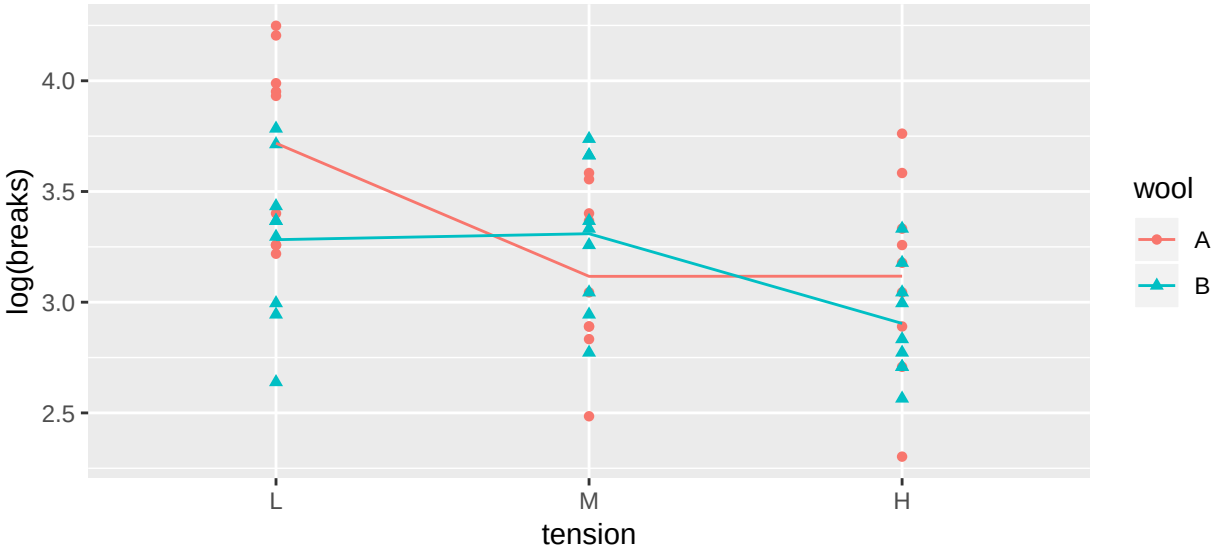
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(model.2)           # table of sequentially added terms in model 2
```

```
## Analysis of Variance Table
##
## Response: log(breaks)
##              Df Sum Sq Mean Sq F value   Pr(>F)
## tension       2 2.1762 1.08808  7.7792 0.001185 **
## wool          1 0.3125 0.31253  2.2344 0.141511
## tension:wool  2 0.9131 0.45657  3.2642 0.046863 *
## Residuals    48 6.7138 0.13987
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Next we look at the effect of the interaction and the easiest way to do this is to look at the interaction plot. This plot shows the raw data and connects lines to the cell mean of each factor combination.

```
warpbreaks <- warpbreaks %>%
  mutate( logy.hat = predict(model.2) )
ggplot(warpbreaks, aes(x=tension, y=log(breaks), color=wool, shape=wool)) +
  geom_point() +
  geom_line(aes(y=logy.hat, x=as.integer(tension)))  # make tension continuous to draw the lines
```



We can see that it appears that wool A has a decrease in breaks between low and medium tension, while wool B has a decrease in breaks between medium and high. It is actually quite difficult to see this interaction when we examine the model coefficients.

```
summary(model.2)
```

```
##
## Call:
## lm(formula = log(breaks) ~ tension * wool, data = warpbreaks)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81504 -0.27885  0.04042  0.27319  0.64358
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.7179     0.1247  29.824  < 2e-16 ***
## tensionM        -0.6012     0.1763  -3.410  0.00133 **
## tensionH        -0.6003     0.1763  -3.405  0.00134 **
## woolB           -0.4356     0.1763  -2.471  0.01709 *
## tensionM:woolB   0.6281     0.2493   2.519  0.01514 *
## tensionH:woolB   0.2221     0.2493   0.891  0.37749
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.374 on 48 degrees of freedom
## Multiple R-squared:  0.3363, Adjusted R-squared:  0.2672
## F-statistic: 4.864 on 5 and 48 DF,  p-value: 0.001116
```

To test if there is a statistically significant difference between medium and high tensions for wool type B, we really need to test the following hypothesis:

$$H_0: \ (\mu + \alpha_2 + \beta_2 + (\alpha\beta)_{22}) - (\mu + \alpha_3 + \beta_2 + (\alpha\beta)_{32}) = 0$$
$$H_a: \ (\mu + \alpha_2 + \beta_2 + (\alpha\beta)_{22}) - (\mu + \alpha_3 + \beta_2 + (\alpha\beta)_{32}) \neq 0$$

This test reduces to testing if $\alpha_2 - \alpha_3 + (\alpha\beta)_{22} - (\alpha\beta)_{23} = 0$. Calculating this difference from the estimated values of the summery table we have $-.6012 + .6003 + .6281 - .2221 = 0.4051$, we don't know if that is significantly different than zero.

In the main effects model, we were able to read off the necessary test using `emmeans`. Fortunately, we can do the same thing here. In this case, we'll look at the interactions piece of the `emmeans` command. In this case, we find the test H:B - M:B in the last row of the interactions.

```
emmeans(model.2, specs= pairwise~tension*wool)
```

```
## $emmeans
##  tension wool   emmean        SE df lower.CL upper.CL
##  L       A    3.717945 0.1246647 48 3.467290 3.968601
##  M       A    3.116750 0.1246647 48 2.866094 3.367405
##  H       A    3.117623 0.1246647 48 2.866967 3.368278
##  L       B    3.282378 0.1246647 48 3.031723 3.533034
##  M       B    3.309327 0.1246647 48 3.058671 3.559982
##  H       B    2.904152 0.1246647 48 2.653496 3.154807
##
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
##
## $contrasts
##  contrast       estimate        SE df t.ratio p.value
##  L,A - M,A  0.6011957092 0.1763026 48   3.410  0.0158
##  L,A - H,A  0.6003226799 0.1763026 48   3.405  0.0160
##  L,A - L,B  0.4355668365 0.1763026 48   2.471  0.1535
##  L,A - M,B  0.4086186238 0.1763026 48   2.318  0.2071
##  L,A - H,B  0.8137936293 0.1763026 48   4.616  0.0004
##  M,A - H,A -0.0008730293 0.1763026 48  -0.005  1.0000
##  M,A - L,B -0.1656288727 0.1763026 48  -0.939  0.9341
##  M,A - M,B -0.1925770854 0.1763026 48  -1.092  0.8821
##  M,A - H,B  0.2125979201 0.1763026 48   1.206  0.8319
##  H,A - L,B -0.1647558434 0.1763026 48  -0.935  0.9355
##  H,A - M,B -0.1917040562 0.1763026 48  -1.087  0.8840
##  H,A - H,B  0.2134709494 0.1763026 48   1.211  0.8295
```

```
##  L,B - M,B -0.0269482127 0.1763026 48  -0.153  1.0000
##  L,B - H,B  0.3782267929 0.1763026 48   2.145  0.2823
##  M,B - H,B  0.4051750056 0.1763026 48   2.298  0.2149
##
## Results are given on the log (not the response) scale.
## P value adjustment: tukey method for comparing a family of 6 estimates
```

The last call to `emmeans` gives us all the pairwise tests comparing the cell means. If we don't want to wade through all the other pairwise contrasts we could do the following:

```
# If I want to not wade through all those contrasts and just grab the
# contrasts for wool type 'B' and tensions 'M' and 'H'
emmeans(model.2, pairwise~tension*wool, at=list(wool='B', tension=c('M','H')))
```

```
## $emmeans
##  tension wool   emmean        SE df lower.CL upper.CL
##  M       B    3.309327 0.1246647 48 3.058671 3.559982
##  H       B    2.904152 0.1246647 48 2.653496 3.154807
##
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
##
## $contrasts
##  contrast  estimate        SE df t.ratio p.value
##  M,B - H,B 0.405175 0.1763026 48   2.298  0.0260
##
## Results are given on the log (not the response) scale.
```

What would happen if we just looked at the main effects? In the case where our experiment is balanced with equal numbers of observations in each treatment cell, we can interpret these differences as follows. Knowing that each cell in our table has a different estimated mean, we could consider the average of all the type A cells as the typical wool A. Likewise we could average all the cell means for the wool B cells. Then we could look at the difference between those two averages. In the balanced design, this is equivalent to removing the tension term from the model and just looking at the difference between the average log number of breaks.

```
emmeans(model.2, specs= pairwise~tension )
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
## $emmeans
##  tension   emmean         SE df lower.CL upper.CL
##  L       3.500162 0.08815128 48 3.322922 3.677402
##  M       3.213038 0.08815128 48 3.035798 3.390278
##  H       3.010887 0.08815128 48 2.833647 3.188127
##
## Results are averaged over the levels of: wool
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
##
## $contrasts
##  contrast  estimate        SE df t.ratio p.value
##  L - M    0.2871237 0.1246647 48   2.303  0.0649
##  L - H    0.4892747 0.1246647 48   3.925  0.0008
##  M - H    0.2021510 0.1246647 48   1.622  0.2465
##
## Results are averaged over the levels of: wool
## Results are given on the log (not the response) scale.
```

```
## P value adjustment: tukey method for comparing a family of 3 estimates
```

```
emmeans(model.2, specs= pairwise~wool )
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
## $emmeans
##  wool   emmean         SE df lower.CL upper.CL
##  A    3.317439 0.07197522 48 3.172723 3.462155
##  B    3.165286 0.07197522 48 3.020570 3.310001
##
## Results are averaged over the levels of: tension
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
##
## $contrasts
##  contrast  estimate        SE df t.ratio p.value
##  A - B    0.1521536 0.1017883 48   1.495  0.1415
##
## Results are averaged over the levels of: tension
## Results are given on the log (not the response) scale.
```

Using `emmeans`, we can see the wool effect difference between types B and A is -0.1522. We can calculate the mean number of log breaks for each wool type and take the difference by the following:

```
warpbreaks %>%
  group_by(wool) %>%
  summarise( wool.means = mean(log(breaks)) ) %>%
  summarise( diff(wool.means) )
```

```
## # A tibble: 1 x 1
##   `diff(wool.means)`
##                <dbl>
## 1             -0.152
```

In the unbalanced case taking the average of the cell means produces a different answer than taking the average of the data. The `emmeans` package chooses to take the average of the cell means.

9.4 Exercises

1. In the `faraway` package, the data set `rats` has data on a gruesome experiment that examined the time till death of 48 rats when they were subjected to three different types of poison administered in four different manners (which they called treatments). We are interested in assessing which poison works the fastest as well as which administration method is most effective.
   a. Consider the interaction model which allows for a different effect of treatment for each poison type. Fit this model and examine the diagnostic plots. What stands out?
   b. Perform a Box-Cox analysis and perform the recommended transformation (with the realm of "common" transformations). Call the transformed variable `speed`.
   c. Fit the interaction model using the transformed response. Create a graph of data and the predicted values. Visually assess if you think the interaction is significant.
   d. Perform an appropriate statistical test to see if the interaction is statistically significant.
   e. What do you conclude about the poisons and treatment (application) types?
2. In the `faraway` package, the dataset `butterfat` has information about the the percent of the milk was butterfat (more is better) taken from $n = 100$ cows. There are 5 different breeds of cows and 2 different ages. We are interested in assessing if `Age` and `Breed` affect the butterfat content

a. Graph the data. Do you think an interaction model is justified?
b. Perform an appropriate set of tests to select a model for predicting `Butterfat`.
c. Discuss your findings.

3. In the `faraway` package, the dataset `alfalfa` has information from a study that examined the effect of seed innoculum, irrigation, and shade on alfalfa yield. This data has $n = 24$ observations.
   a. Graph the data.
   b. Consider the main effects model with all three predictor variables. Examine the diagnostic plots and comment. Consider a Box-Cox transformation to your response (you might need to use `lambda = seq(-2,6, by=.01)` to see the whole curve). Make any transformation you feel is justified.
   c. Consider the model with `shade` and `inoculum` and the interaction between the two. Examine the anova table. Why does R complain that the fit is perfect? *Hint: Think about the degrees of freedom of the model compared to the sample size.*
   d. Discuss your findings and the limitations of your investigation based on data.

Chapter 10

Block Designs

```
# packages for this chapter
library(tidyverse)    # ggplot2, dplyr, etc...
library(emmeans)     # TukeyLetters stuff
```

Often there are covariates in the experimental units that are known to affect the response variable and must be taken into account. Ideally an experimenter can group the experimental units into blocks where the within block variance is small, but the block to block variability is large. For example, in testing a drug to prevent heart disease, we know that gender, age, and exercise levels play a large role. We should partition our study participants into gender, age, and exercise groups and then randomly assign the treatment (placebo vs drug) within the group. This will ensure that we do not have a gender, age, and exercise group that has all placebo observations.

Often blocking variables are not the variables that we are primarily interested in, but must nevertheless be considered. We call these nuisance variables. We already know how to deal with these variables by adding them to the model, but there are experimental designs where we must be careful because the experimental treatments are *nested*.

Example 1. An agricultural field study has three fields in which the researchers will evaluate the quality of three different varieties of barley. Due to how they harvest the barley, we can only create a maximum of three plots in each field. In this example we will block on field since there might be differences in soil type, drainage, etc from field to field. In each field, we will plant all three varieties so that we can tell the difference between varieties without the block effect of field confounding our inference. In this example, the varieties are nested within the fields.

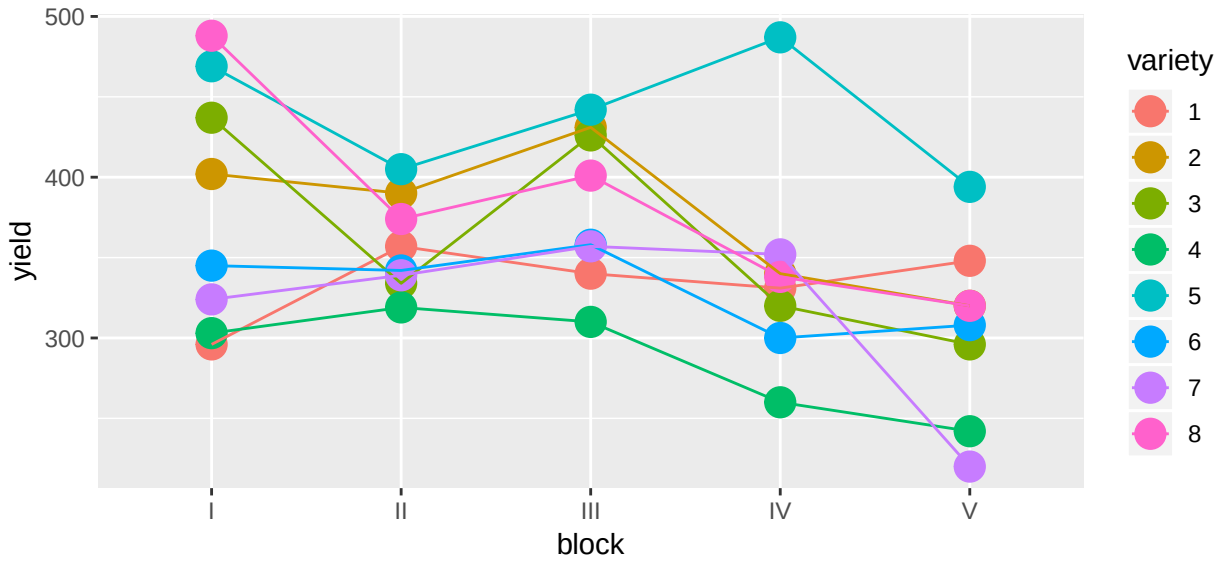
	Field 1	Field 2	Field 3
Plot 1	Variety A	Variety C	Variety B
Plot 2	Variety B	Variety A	Variety C
Plot 3	Variety C	Variety B	Variety A

Example 2. We are interested in how a mouse responds to five different materials inserted into subcutaneous tissue to evaluate the materials' use in medicine. Each mouse can have a maximum of 3 insertions. Here we will block on the individual mice because even lab mice have individual variation. We actually are not interested in estimating the effect of the mice because they aren't really of interest, but the mouse block effect should be accounted for before we make any inferences about the materials. Notice that if we only have one insertion per mouse, then the mouse effect will be confounded with materials.

10.1 Randomized Complete Block Design (RCBD)

The dataset `oatvar` in the faraway library contains information about an experiment on eight different varieties of oats. The area in which the experiment was done had some systematic variability and the researchers divided the area up into five different blocks in which they felt the area inside a block was uniform while acknowledging that some blocks are likely superior to others for growing crops. Within each block, the researchers created eight plots and randomly assigned a variety to a plot. This type of design is called a Randomized Complete Block Design (RCBD) because each block contains all possible levels of the factor of primary interest.

```
data('oatvar', package='faraway')
ggplot(oatvar, aes(y=yield, x=block, color=variety)) +
    geom_point(size=5) +
    geom_line(aes(x=as.integer(block))) # connect the dots
```



While there is one unusual observation in block IV, there doesn't appear to be a blatant interaction. We will consider the interaction shortly. For the main effects model of yield ~ block + variety we have $p = 12$ parameters and 28 residual degrees of freedom because

$$\begin{aligned} df_\epsilon &= n - p \\ &= n - (1 + [(I-1) + (J-1)]) \\ &= 40 - (1 + [(5-1) + (8-1)]) \\ &= 40 - 12 \\ &= 28 \end{aligned}$$

```
m1 <- lm( yield ~ block + variety, data=oatvar)
anova(m1)
```

```
## Analysis of Variance Table
##
## Response: yield
##           Df Sum Sq Mean Sq F value    Pr(>F)
## block      4  33396  8348.9  6.2449  0.001008 **
## variety    7  77524 11074.8  8.2839 1.804e-05 ***
## Residuals 28  37433  1336.9
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# plot(m1)      # check diagnostic plots - they are fine...
```

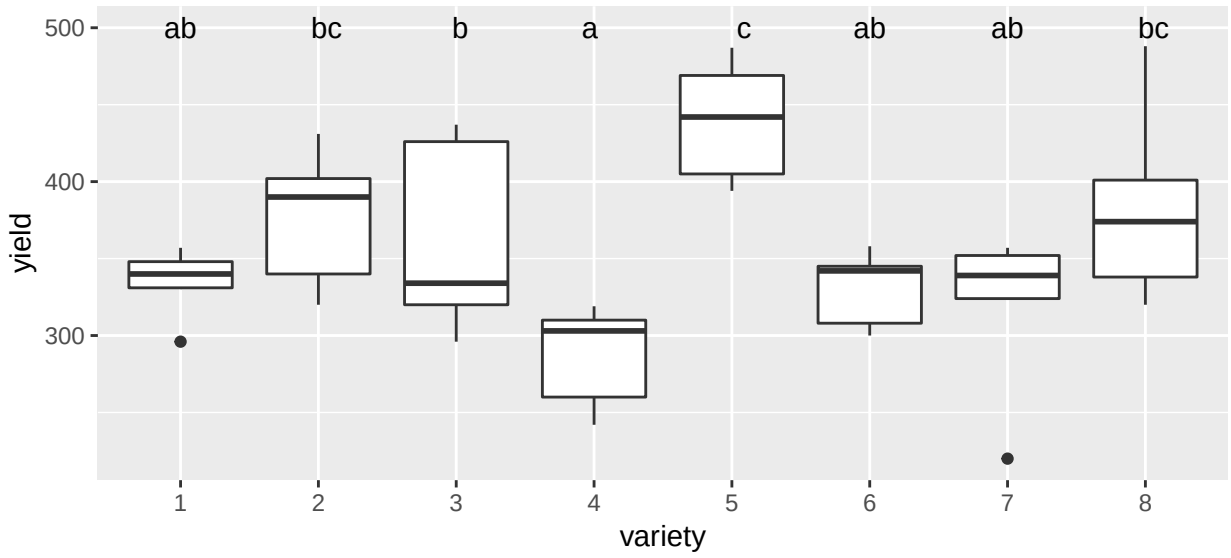
Because this is an orthogonal design, the sums of squares doesn't change regardless of which order we add the factors, but if we remove one or two observations, they would.

In determining the significance of `variety` the above F-value and p-value is correct. We have 40 observations (5 per variety), and after accounting for the model structure (including the extraneous blocking variable), we have 28 residual degrees of freedom.

But the F-value and p-value for testing if `block` is significant is nonsense! Imagine that variety didn't matter we just have 8 replicate samples per block, but these aren't true replicates, they are what is called *pseudoreplicates*. Imagine taking a sample of $n = 3$ people and observing their height at 1000 different points in time during the day. You don't have 3000 data points for estimating the mean height in the population, you have 3. Unless we account for the this, the inference for the block variable is wrong. In this case, we only have one observation for each block, so we can't do any statistical inference at the block scale!

Fortunately in this case, we don't care about the blocking variable and including it in the model was simply guarding us in case there was a difference, but I wasn't interested in estimating it. If the only covariate we care about is the most deeply nested effect, then we can do the usual analysis and recognize the p-value for the blocking variable is nonsense, and we don't care about it.

```
# Ignore any p-values regarding block, but I'm happy with the analysis for variety
letter_df <- emmeans(m1, ~variety) %>%
  cld(Letters=letters) %>%
  dplyr::select(variety, .group) %>%
  mutate(yield = 500)

ggplot(oatvar, aes(x=variety, y=yield)) +
  geom_boxplot() +
  geom_text( data=letter_df, aes(label=.group) )
```



However it would be pretty sloppy to not do the analysis correctly because our blocking variable might be something we care about. To make R do the correct analysis, we have to denote the nesting. In this case we have block-to-block errors, and then variability within blocks. To denote the nesting we use the `Error()` function within our formula. By default, `Error()` just creates independent error terms, but when we add a covariate, it adds the appropriate nesting.

```
m3 <- aov( yield ~ variety + Error(block), data=oatvar)
summary(m3)
```

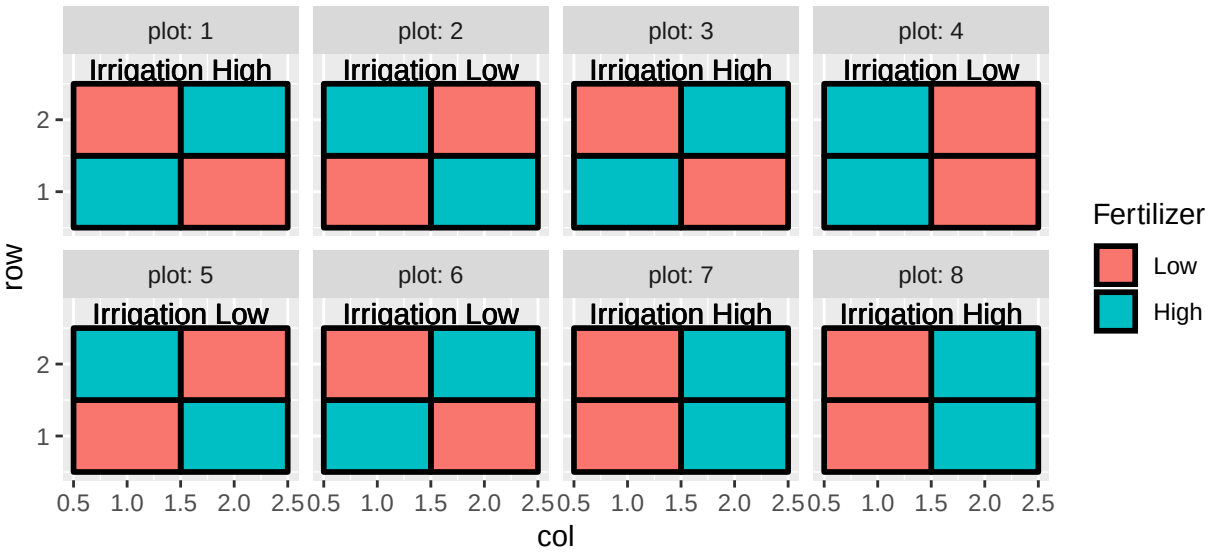
```
##
## Error: block
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  4  33396    8349
##
## Error: Within
##           Df Sum Sq Mean Sq F value  Pr(>F)
## variety    7  77524   11075   8.284 1.8e-05 ***
## Residuals 28  37433    1337
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that in our block level, there is no p-value to assess if the blocks are different. This is because we don't have any replication of the blocks. So our analysis respects that blocks are present, but does not attempt any statistical analyses on them.

10.2 Split-plot designs

There are plenty of experimental designs where we have levels of treatments nested within each other for practical reasons. The literature often gives the example of an agriculture experiment where we investigate the effect of irrigation and fertilizer on the yield of a crop. However because our irrigation system can't be fine-tuned, we have plots with different irrigation levels and within each plot we have perhaps four subplots that have the fertilizer treatment. To summarize, Irrigation treatments were randomly assigned to plots, and fertilizer treatments were randomly assigned to sub-plots.

```
## # A tibble: 6 x 5
## # Groups:   plot, subplot, Fertilizer [6]
##   plot  subplot Fertilizer Irrigation yield
##   <fct> <fct>   <fct>      <fct>      <dbl>
## 1 1     1       Low        Low         20.2
## 2 1     2       High       Low         24.4
## 3 1     3       Low        Low         18.0
## 4 1     4       High       Low         21.0
## 5 2     1       Low        Low         23.2
## 6 2     2       High       Low         26.5
```



So all together we have 8 plots, and 32 subplots. When I analyze the fertilizer, I have 32 experimental units (the thing I have applied my treatment to), but when analyzing the effect of irrigation, I only have 8 experimental units.

I like to think of this set up as having some lurking variables that act at the plot level (changes in aspect, maybe something related to what was planted prior) and some lurking variables that act on a local subplot scale (maybe variation in clay/silt/sand ratios). So even after I account for Irrigation and Fertilizer treatments, observations within a plot will be more similar to each other than observations in two different plots.

We can think about doing two separate analyses, one for the effect of irrigation, and another for the effect of the fertilizer.

```
# AgData came from my data package, dsData, (however I did some summarization
# first.)

# To analyze Irrigation, average over the subplots first...
Irrigation.data <-   AgData %>%
  group_by(plot, Irrigation) %>%
  summarise( yield = mean(yield)) %>%
  as.data.frame()  # the aov command doesn't like tibbles.

# Now do a standard analysis. I use the aov() command instead of lm()
# because we will shortly do something very tricky that can only be
# done with aov(). For the most part, everything is
# identical from what you are used to.
m <- aov( yield ~ Irrigation, data=Irrigation.data )
anova(m)
```

```
## Analysis of Variance Table
##
## Response: yield
##            Df Sum Sq Mean Sq F value Pr(>F)
## Irrigation  1 26.064 26.0645  3.4281 0.1136
## Residuals   6 45.619  7.6032
```

In this case we see that we have insufficient evidence to conclude that the observed difference between the Irrigation levels could not be due to random chance.

Next we can do the appropriate analysis for the fertilizer, recognizing that all the p-values for the plot effects are nonsense and should be ignored.

```
m <- aov( yield ~ plot + Fertilizer, data=AgData )
summary(m)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## plot         7 286.73   40.96   3.220 0.0157 *
## Fertilizer   1   6.67    6.67   0.524 0.4762
## Residuals   23 292.59   12.72
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ideally I wouldn't have to do the averaging over the nested observations and we would like to not have the misleading p-values for the plots. To do this, we only have to specify the nesting of the error terms and R will figure out the appropriate degrees of freedom for the covariates.

```
# To do this right, we have to abandon the general lm() command and use the more
# specialized aov() command.  The Error() part of the formula allows me to nest
# the error terms and allow us to do the correct analysis. The order of these is
# to start with the largest/highest level and then work down the nesting.
m2 <- aov( yield ~ Irrigation + Fertilizer + Error(plot/subplot), data=AgData )
summary(m2)
```

```
##
## Error: plot
##            Df Sum Sq Mean Sq F value Pr(>F)
## Irrigation  1  104.3  104.26   3.428  0.114
## Residuals   6  182.5   30.41
##
## Error: plot:subplot
##            Df Sum Sq Mean Sq F value Pr(>F)
## Fertilizer  1   6.67   6.672   0.524  0.476
## Residuals  23 292.59  12.721
```

In the output, we see that the ANOVA table row for the Fertilizer is the same for both analyses, but the sums-of-squares for Irrigation are different between the two analyses (because of the averaging) while the F and p values are the same between the two analyses.

What would have happened if we had performed the analysis incorrectly and had too many degrees of freedom for the Irrigation test?

```
bad.model <- aov( yield ~ Irrigation + Fertilizer, data=AgData)
anova(bad.model)
```

```
## Analysis of Variance Table
##
## Response: yield
##            Df Sum Sq Mean Sq F value  Pr(>F)
## Irrigation  1 104.26 104.258  6.3643 0.01738 *
## Fertilizer  1   6.67   6.672  0.4073 0.52835
## Residuals  29 475.07  16.382
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case we would have concluded that we had statistically significant evidence to conclude the Irrigation levels are different. Notice that the sums-of-squares in this **wrong** analysis match up with the sums-of-squares in the correct design and the only difference is that when we figure out the sum-of-squares for the

residuals we split that into different pools.

$$RSS_{total} = RSS_{Fertilizer} + RSS_{Irrigation}$$
$$456.12 = 273.64 + 182.5$$

When we want to infer if the amount of noise explained by adding Irrigation or Fertilizer is sufficiently large to justify their inclusion into the model, we compare the sum-of-squares value to the RSS but now we have to use the appropriate pool.

A second example of a slightly more complex split plot is given in the package `MASS` under the dataset `oats`. From the help file the data describes the following experiment:

> The yield of oats from a split-plot field trial using three varieties and four levels of manurial treatment. The experiment was laid out in 6 blocks of 3 main plots, each split into 4 sub-plots. The varieties were applied to the main plots and the manurial treatments to the sub-plots.

This is a lot to digest so lets unpack it. First we have 6 blocks and we'll replicate the exact same experiment in each block. Within a block, we'll split it into three sections, which we'll call plots (within the block). Finally within each plot, we'll have 4 subplots.

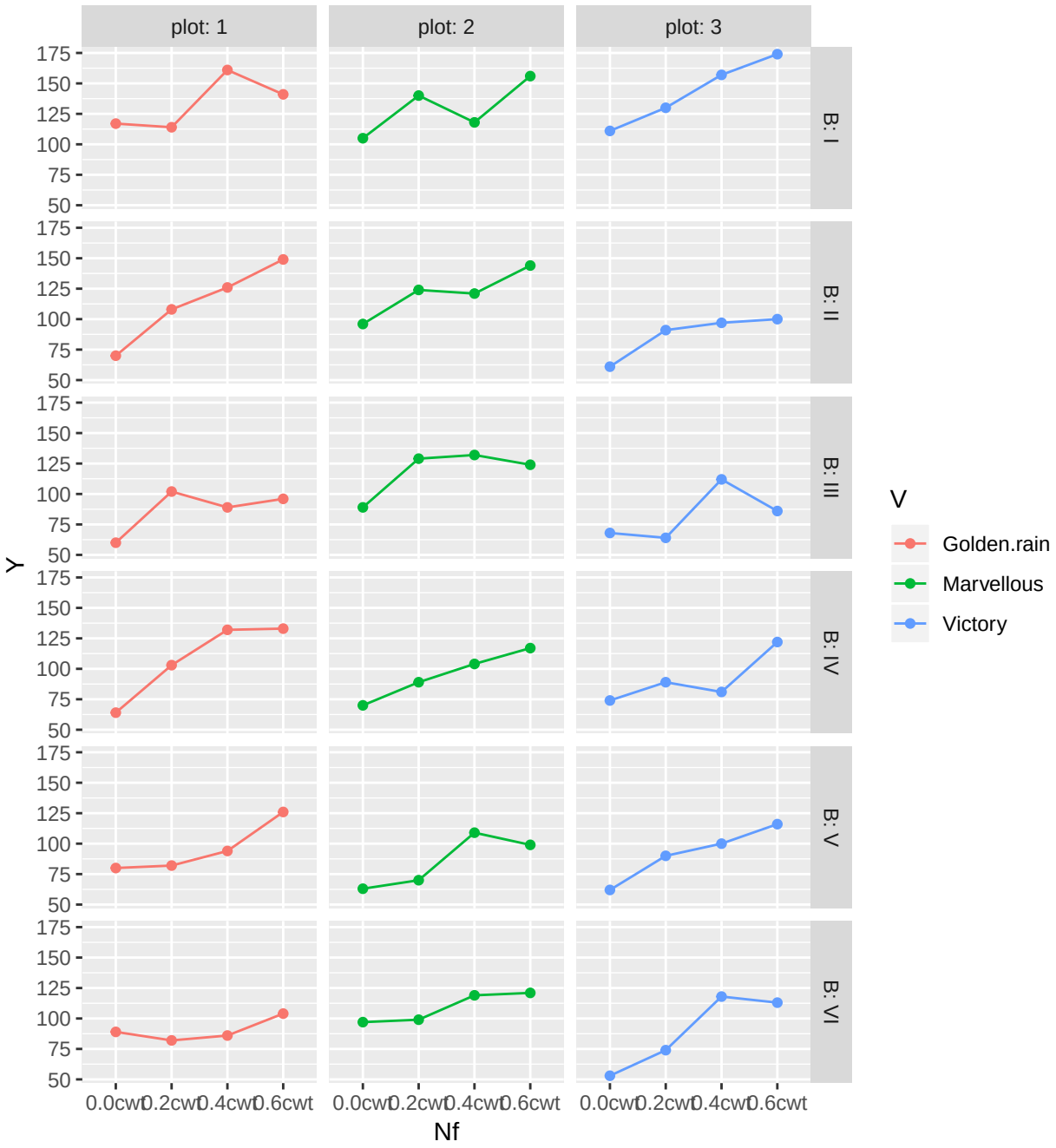
We have 3 varieties of oats, and 4 levels of fertilizer (manure). To each set of 3 plots, we'll randomly assign the 3 varieties, and to each set of subplots, we'll assign the fertilizers.

One issue that makes this issue confusing for students is that most texts get lazy and don't define the blocks, plots, and sub-plots when there are no replicates in a particular level. I prefer to be clear about defining those so.

```
data('oats', package='MASS')
oats <- oats %>% mutate(
  Nf = ordered(N, levels = sort(levels(N))),  # make manure an ordered factor
  plot = as.integer(V),                       # plot
  subplot = as.integer(Nf))                   # sub-plot
```

As always we first create a graph to examine the data

```
oats <- oats %>% mutate(B_Plot = interaction(B, plot))
ggplot(oats, aes(x=Nf, y=Y, color=V)) +
  facet_grid( B ~ plot, labeller=label_both) +
  geom_point() +
  geom_line(aes(x=as.integer(Nf)))
```



This graph also makes me think that variety doesn't matter and it is unlikely that there an interaction between oat variety and fertilizer level, but we should check.

```
#  What makes sense to me
# m.c <- aov( Y ~ V * Nf + Error(B/plot/subplot), data=oats)
```

Unfortunately the above model isn't correct because R isn't smart enough to understand that the levels of plot and subplot are exact matches to the Variety and Fertilizer levels. As a result if I defined the model above, the degrees of freedom will be all wrong because there is too much nesting. So we have to be smart enough to recognize that plot and subplot are actually Variety and Fertilizer.

```
m.c <- aov( Y ~ V * Nf + Error(B/V/Nf), data=oats)
summary(m.c)
```

```
##
## Error: B
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  5  15875    3175
##
## Error: B:V
##           Df Sum Sq Mean Sq F value Pr(>F)
## V          2   1786   893.2   1.485  0.272
## Residuals 10   6013   601.3
##
## Error: B:V:Nf
##           Df Sum Sq Mean Sq F value   Pr(>F)
## Nf         3  20020    6673  37.686 2.46e-12 ***
## V:Nf       6    322      54   0.303    0.932
## Residuals 45   7969     177
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Sure enough the interaction term is not significant. We next consider the Variety term.

```
m.s <- aov( Y ~ V + Nf + Error(B/V/Nf), data=oats)
summary(m.s)
```

```
##
## Error: B
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  5  15875    3175
##
## Error: B:V
##           Df Sum Sq Mean Sq F value Pr(>F)
## V          2   1786   893.2   1.485  0.272
## Residuals 10   6013   601.3
##
## Error: B:V:Nf
##           Df Sum Sq Mean Sq F value   Pr(>F)
## Nf         3  20020    6673   41.05 1.23e-13 ***
## Residuals 51   8291     163
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We conclude by noticing that the Variety does not matter, but that the fertilizer level is quite significant.

There are many other types of designs out there. For example you might have 5 levels of a factor, but when you split your block into plots, you can only create 3 plots. So not every block will have every level of the factor. This is called *Randomized Incomplete Block Designs* (RIBD).

You might have a design where you apply even more levels of nesting. Suppose you have a green house study where you have rooms where you can apply a temperature treatment, within the room you have four tables and can apply a light treatment to each table. Finally within each table you can have four trays where can apply a soil treatment to each tray. This is a continuation of the split-plot design and by extending the nesting we can develop *split-split-plot* and *split-split-split-plot* designs.

You might have 7 covariates each with two levels (High, Low) and you want to investigate how these influence your response but also allow for second and third order interactions. If you looked at every treatment combination you'd have $2^7 = 128$ different treatment combinations and perhaps you only have the budget

for a sample of $n = 32$. How should you design your experiment? This question is addressed by *fractional factorial* designs.

If your research interests involve designing experiments such as these, you should consider taking an Experimental design course.

10.3 Exercises

1. ???
2. ???
3. ???

Chapter 11

Mixed Effects Models

```
library(tidyverse)  # dplyr, tidyr, ggplot
library(stringr)    # string manipulation stuff

library(lme4)        # Our primary analysis routine
library(lmerTest)    # A user friendly interface to lme4 that produces p-values
library(emmeans)     # For all of my pairwise contrasts
library(car)         # For bootstrap Confidence/Prediction Intervals

# library(devtools)
# install_github('dereksonderegger/dsData')  # datasets I've made; only install once...
library(dsData)
```

The assumption of independent observations is often not supported and dependent data arises in a wide variety of situations. The dependency structure could be very simple such as rabbits within a litter being correlated and the litters being independent. More complex hierarchies of correlation are possible. For example we might expect voters in a particular part of town (called a precinct) to vote similarly, and particular districts in a state tend to vote similarly as well, which might result in a precinct / district / state hierarchy of correlation.

Many of the designs mentioned in the Block Designs section could be similarly modeled using Mixed Effects Models. In many respects, the random effects structure provides a more flexible framework to consider many of the traditional experimental designs as well as many non-traditional designs with the benefit of more easily assessing variability at each hierarchical level.

Mixed effects models combine what we call "fixed" and "random" effects.

Fixed effects	Unknown constants that we wish to estimate from the model and could be similarly estimated in subsequent experimentation. The research is interested in these particular levels.
Random effects	Random variables sampled from a population which cannot be observed in subsequent experimentation. The research is not interested in these particular levels, but rather how the levels vary from sample to sample.

For example, in a rabbit study that examined the effect of diet on the growth of domestic rabbits and we had 10 litters of rabbits and used the 3 most similar from each litter to test 6 different diets. Here, the 6 different diets are fixed effects because they are not randomly selected from a population, these exact same diets can be further studied, and these are the diets we are interested it. The litters of rabbits and the individual rabbits are randomly selected from populations, cannot be exactly replicated in future studies, and we are

not interested in the individual litters but rather what the variability is between individuals and between litters.

Often random effects are not of primary interest to the researcher, but must be considered. Often blocking variables are random effects because the arise from a random sample of possible blocks that are potentially available to the researcher.

Mixed effects models are models that have both fixed and random effects. We will first concentrate on understanding how to address a model with two sources error and then complicate the matter with fixed effects.

11.1 Review of Maximum Likelihood Methods

Recall that the likelihood function is the function links the model parameters to the data and is found by taking the probability density function and interpreting it as a function of the parameters instead of the a function of the data. Loosely, the probability function tells us what outcomes are most probable, with the height of the function telling us which values (or regions of values) are most probable given a set of parameter values. The higher the probability function, the higher the probability of seeing that value (or data in that region). The likelihood function turns that relationship around and tells us what parameter values are most likely to have generated the data we have, again with the parameter values with a higher likelihood value being more "likely".

The likelihood function for a sample $y_i \overset{iid}{\sim} N(\mu, \sigma)$ can be written as a function of our parameters μ and σ^2 then we have defined our likelihood function

$$L\left(\mu, \sigma^2 | y_1, \ldots, y_n\right) = \frac{1}{(2\pi)^{n/2} \left[\det\left(\boldsymbol{\Omega}\right)\right]^{1/2}} \exp\left[-\frac{1}{2}\left(\boldsymbol{y}-\boldsymbol{\mu}\right)^T \boldsymbol{\Omega}^{-1}\left(\boldsymbol{y}-\boldsymbol{\mu}\right)\right]$$

where the variance/covariance matrix is $\boldsymbol{\Omega} = \sigma I_n$.

We can use to this equation to find the maximum likelihood estimators by either taking the derivatives and setting them equal to zero and solving for the parameters or by using numerical methods. In the normal case, we can find the maximum likelihood estimators (MLEs) using the derivative trick and we find that

$$\hat{\mu}_{MLE} = \hat{y} = \bar{y}$$

and

$$\hat{\sigma}^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}\right)^2$$

and we notice that this is not our usual estimator $\hat{\sigma}^2 = s^2$ where s^2 is the sample variance. It turns out that the MLE estimate of σ^2 is biased (the correction is to divide by $n-1$ instead of n). This is normally not an issue if our sample size is large, but with a small sample, the bias is not insignificant.

Notice if we happened to know that $\mu = 0$, then we could use

$$\hat{\sigma}^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n} y_i^2$$

and this would be unbiased for σ^2.

In general (a not just in the normal case above) the *Likelihood Ratio Test* (LRT) provides a way for us to compare two nested models. Given m_0 which is a simplification of m_1 then we could calculate the likelihoods functions of the two models $L\left(\boldsymbol{\theta}_0\right)$ and $L\left(\boldsymbol{\theta}_1\right)$ where $\boldsymbol{\theta}_0$ is a vector of parameters for the null model and $\boldsymbol{\theta}_1$

is a vector of parameter for the alternative. Let $\hat{\boldsymbol{\theta}}_0$ be the maximum likelihood estimators for the null model and $\hat{\boldsymbol{\theta}}_1$ be the maximum likelihood estimators for the alternative. Finally we consider the value of

$$\begin{aligned} D &= -2 * \log\left[\frac{L\left(\hat{\boldsymbol{\theta}}_0\right)}{L\left(\hat{\boldsymbol{\theta}}_1\right)}\right] \\ &= -2\left[\log L\left(\hat{\boldsymbol{\theta}}_0\right) - \log L\left(\hat{\boldsymbol{\theta}}_1\right)\right] \end{aligned}$$

Under the null hypothesis that m_0 is the true model, the $D \dot{\sim} \chi^2_{p_1-p_0}$ where $p_1 - p_0$ is the difference in number of parameters in the null and alternative models. That is to say that asymptotically D has a Chi-squared distribution with degrees of freedom equal to the difference in degrees of freedom of the two models.

We could think of $L\left(\hat{\boldsymbol{\theta}}_0\right)$ as the maximization of the likelihood when some parameters are held constant (at zero) and all the other parameters are vary. But we are not required to hold it constant at zero. We could chose any value of interest and perform a LRT.

Because we often regard a confidence interval as the set of values that would not be rejected by a hypothesis test, we could consider a sequence of possible values for a parameter and figure out which would not be rejected by the LRT. In this fashion we can construct confidence intervals for parameter values.

Unfortunately all of this hinges on the asymptotic distribution of D and often this turns out to be a poor approximation. In simple cases more exact tests can be derived (for example the F-tests we have used prior) but sometimes nothing better is currently known. Another alternative is to use resampling methods for the creation of confidence intervals or p-values.

11.2 1-way ANOVA with a random effect

We first consider the simplest model with two sources of variability, a 1-way ANOVA with a random factor covariate

$$y_{ij} = \mu + \gamma_i + \epsilon_{ij}$$

where $\gamma_i \overset{iid}{\sim} N\left(0, \sigma^2_\gamma\right)$ and $\epsilon_{ij} \overset{iid}{\sim} N\left(0, \sigma^2_\epsilon\right)$. This model could occur, for example, when looking at the adult weight of domestic rabbits where the random effect is the effect of litter and we are interested in understanding how much variability there is between litters $\left(\sigma^2_\gamma\right)$ and how much variability there is within a litter $\left(\sigma^2_\epsilon\right)$. Another example is the the creation of computer chips. Here a single wafer of silicon is used to create several chips and we might have wafer-to-wafer variability and then within a wafer, you have chip-to-chip variability.

First we should think about what the variances and covariances are for any two observations.

$$\begin{aligned} Var\left(y_{ij}\right) &= Var\left(\mu + \gamma_i + \epsilon_{ij}\right) \\ &= Var\left(\mu\right) + Var\left(\gamma_i\right) + Var\left(\epsilon_{ij}\right) \\ &= 0 + \sigma^2_\gamma + \sigma^2_\epsilon \end{aligned}$$

and $Cov\left(y_{ij}, y_{ik}\right) = \sigma^2_\gamma$ because the two observations share the same litter γ_i. For two observations in different litters, the covariance is 0. These relationships induce a correlation on observations within the same litter of

$$\rho = \frac{\sigma^2_\gamma}{\sigma^2_\gamma + \sigma^2_\epsilon}$$

For example, suppose that we have $I = 3$ litters and in each litter we have $J = 3$ rabbits per litter. Then the variance-covariance matrix looks like

$$\boldsymbol{\Omega} = \begin{bmatrix} \sigma_\gamma^2 + \sigma_\epsilon^2 & \sigma_\gamma^2 & \sigma_\gamma^2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_\gamma^2 & \sigma_\gamma^2 + \sigma_\epsilon^2 & \sigma_\gamma^2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_\gamma^2 & \sigma_\gamma^2 & \sigma_\gamma^2 + \sigma_\epsilon^2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \sigma_\gamma^2 + \sigma_\epsilon^2 & \sigma_\gamma^2 & \sigma_\gamma^2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \sigma_\gamma^2 & \sigma_\gamma^2 + \sigma_\epsilon^2 & \sigma_\gamma^2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \sigma_\gamma^2 & \sigma_\gamma^2 & \sigma_\gamma^2 + \sigma_\epsilon^2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \sigma_\gamma^2 + \sigma_\epsilon^2 & \sigma_\gamma^2 & \sigma_\gamma^2 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \sigma_\gamma^2 & \sigma_\gamma^2 + \sigma_\epsilon^2 & \sigma_\gamma^2 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \sigma_\gamma^2 & \sigma_\gamma^2 & \sigma_\gamma^2 + \sigma_\epsilon^2 \end{bmatrix}$$

Substituting this new variance-covariance matrix into our likelihood function, we now have a likelihood function which we can perform our usual MLE tricks with.

In the more complicated situation where we have a full mixed effects model, we could write

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

where $\boldsymbol{X}$ is the design matrix for the fixed effects, $\boldsymbol{\beta}$ is the vector of fixed effect coefficients, $\boldsymbol{Z}$ is the design matrix for random effects, $\boldsymbol{\gamma}$ is the vector of random effects such that $\gamma_i \overset{iid}{\sim} N\left(0, \sigma_\gamma^2\right)$ and finally $\boldsymbol{\epsilon}$ is the vector of error terms such that $\epsilon_{ij} \overset{iid}{\sim} N\left(0, \sigma_\epsilon^2\right)$. Notice in our rabbit case

$$\boldsymbol{Z} = \begin{bmatrix} 1 & \cdot & \cdot \\ 1 & \cdot & \cdot \\ 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \\ \cdot & 1 & \cdot \\ \cdot & 1 & \cdot \\ \cdot & \cdot & 1 \\ \cdot & \cdot & 1 \\ \cdot & \cdot & 1 \end{bmatrix} \quad ZZ^T = \begin{bmatrix} 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 \end{bmatrix}$$

which makes it easy to notice

$$\boldsymbol{\Omega} = \sigma_\gamma^2 \boldsymbol{Z}\boldsymbol{Z}^T + \sigma_\epsilon^2 \boldsymbol{I}$$

In practice we tend to have relatively small numbers of block parameters and thus have a small number of observations in which to estimate σ_γ^2 which means that the biased nature of MLE estimates will be sub-optimal. If we knew that $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{0}$ we could use that fact and have an unbiased estimate of our variance parameters. Because $\boldsymbol{X}$ is known, we can find linear functions $\boldsymbol{k}$ such that $\boldsymbol{k}^T\boldsymbol{X} = 0$. We can form a matrix $\boldsymbol{K}$ that represents all of these possible transformations and we notice that

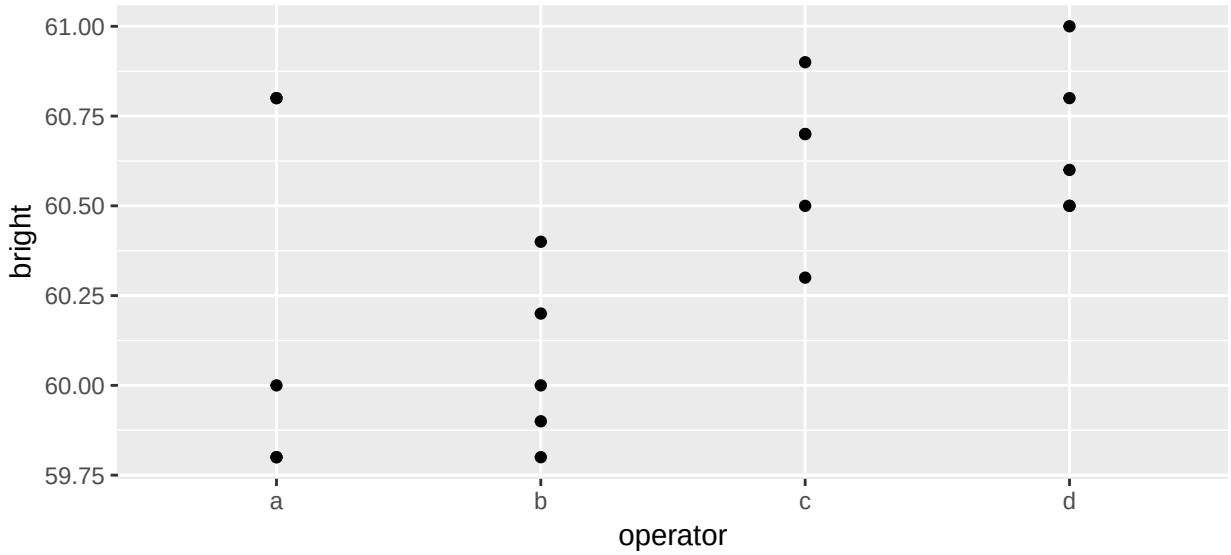
$$\boldsymbol{K}^T\boldsymbol{y} \sim N\left(\boldsymbol{K}^T\boldsymbol{X}\beta,\, \boldsymbol{K}^T\boldsymbol{\Omega}\boldsymbol{K}\right) = N\left(\boldsymbol{0}, \boldsymbol{K}^T\boldsymbol{\Omega}\boldsymbol{K}\right)$$

and perform our maximization on this transformed set of data. Once we have our unbiased estimates of σ_γ^2 and σ_ϵ^2, we can substitute these back into the untransformed likelihood function and find the MLEs for $\boldsymbol{\beta}$. This process is called Restricted Maximum Likelihood (REML) and is generally preferred over the variance component estimates found simply maximizing the regular likelihood function. As usual, if our experiment is balanced these complications aren't necessary as the REML estimates of $\boldsymbol{\beta}$ are usually the same as the ML estimates.

Our first example comes from an experiment to test the paper brightness as affected by the shift operator. The data has 20 observations with 4 different operators. Each operator had 5 different observations made. The data set is `pulp` in the package `faraway`. We will first analyze this using a fixed-effects one-way ANOVA,

but we will use a different model representation. Instead of using the first operator as the reference level, we will use the sum-to-zero constraint (to make it easier to compare with the output of the random effects model).

```
data('pulp', package='faraway')
ggplot(pulp, aes(x=operator, y=bright)) + geom_point()
```




```
# set the contrasts to sum-to-zero constraint
op <- options(contrasts=c('contr.sum', 'contr.poly'))
m <- aov(bright ~ operator, data=pulp)
summary(m)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## operator     3   1.34  0.4467   4.204 0.0226 *
## Residuals   16   1.70  0.1062
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coef(m)
```

```
## (Intercept)   operator1   operator2   operator3
##       60.40       -0.16       -0.34        0.22
```

The sum-to-zero constraint forces the operator parameters to sum to zero so we can find the value of the fourth operator as operator4 = -(-0.16-0.34+0.22) = 0.28

To fit the random effects model we will use the package `lmerTest` which is a nicer user interface to the package `lme4`. The reason we won't use `lme4` directly is that the authors of `lme4` refuse to calculate p-values. The reason for this is that in mixed models it is not always clear what the appropriate degrees of freedom are for the residuals, and therefore we don't know what the appropriate t-distribution is to compare the t-values to. In simple balanced designs the degrees of freedom can be calculated, but in complicated unbalanced designs the appropriate degrees of freedom is not known and all proposed heuristic methods (including what is calculated by SAS) can fail spectacularly in certain cases. The authors of `lme4` are adamant that until robust methods are developed, they prefer to not calculate any p-values. There are other packages out there that recognize that we need approximate p-values and the package `lmerTest` provides reasonable answers that match was SAS calculates.

```
m2 <- lmer( bright ~ 1 + (1|operator), data=pulp )
summary(m2)  # because there are no fixed effects, lmerTest bailed out to lme4.
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: bright ~ 1 + (1 | operator)
##    Data: pulp
##
## REML criterion at convergence: 18.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.4666 -0.7595 -0.1244  0.6281  1.6012
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  operator (Intercept) 0.06808  0.2609
##  Residual             0.10625  0.3260
## Number of obs: 20, groups:  operator, 4
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)  60.4000     0.1494  3.0000   404.2 3.34e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that the estimate of the fixed effect (the overall mean) is the same in the fixed-effects ANOVA and in the mixed model. However the fixed effects ANOVA estimates the effect of each operator while the mixed model is interested in estimating the variance between operators. In the model statement the (1|operator) denotes the random effect and this notation tells us to fit a model with a random intercept term for each operator. Here the variance associated with the operators is $\sigma^2_\gamma = 0.068$ while the "pure error" is $\sigma^2_\epsilon = 0.106$. The column for standard deviation is not the variability associated with our estimate, but is simply the square-root of the variance terms σ_γ and σ_ϵ. This was fit using the REML method.

We might be interested in the estimated effect of each operator

```
ranef(m2)
```

```
## $operator
##   (Intercept)
## a  -0.1219403
## b  -0.2591231
## c   0.1676679
## d   0.2133955
```

These effects are smaller than the values we estimated in the fixed effects model due to distributional assumption that penalizes large deviations from the mean. In general, the estimated random effects are of smaller magnitude than the effect size estimated using a fixed effect model.

```
# reset the contrasts to the default
options(contrasts=c("contr.treatment", "contr.poly" ))
```

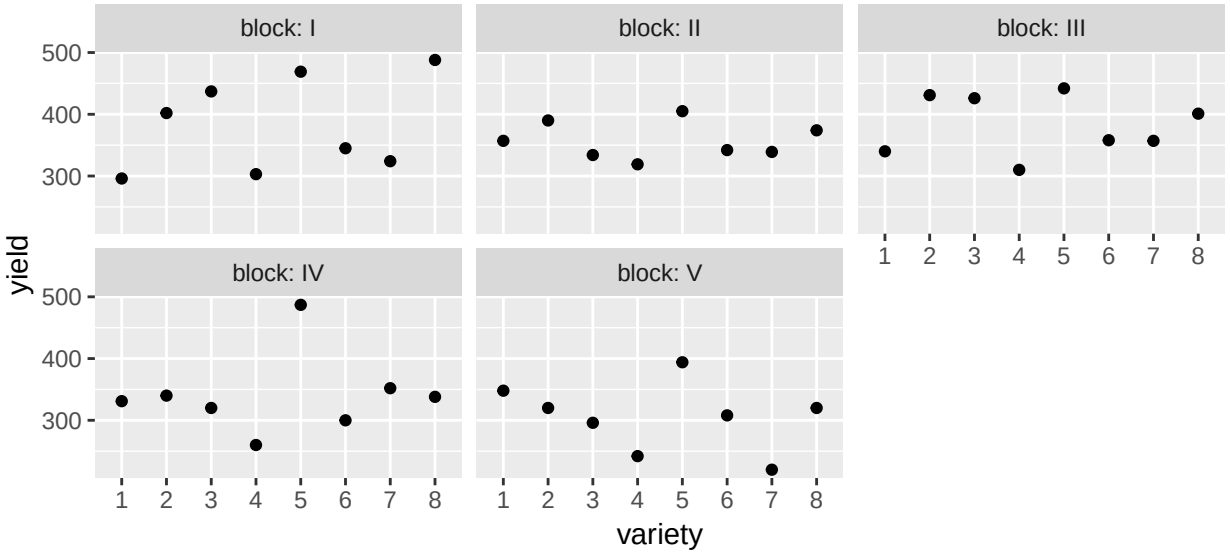
11.3 Blocks as Random Variables

Blocks are properties of experimental designs and usually we are not interested in the block levels *per se* but need to account for the variability introduced by them.

Recall the agriculture experiment in the dataset `oatvar` from the `faraway` package. We had 8 different

varieties of oats and we had 5 different fields (which we called blocks). Because of limitations on how we plant, we could only divide the blocks into 8 plots and in each plot we planted one of the varieties.

```
data('oatvar', package='faraway')
ggplot(oatvar, aes(y=yield, x= variety)) +
  geom_point() +
  facet_wrap(~block, labeller=label_both)
```



In this case, we don't really care about these particular fields (blocks) and would prefer to think about these as a random sample of fields that we might have used in our experiment.

```
model.0 <- lmer( yield ~ (1|block), data=oatvar)
model.1 <- lmer( yield ~ variety + (1|block), data=oatvar)
anova(model.0, model.1)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: oatvar
## Models:
## model.0: yield ~ (1 | block)
## model.1: yield ~ variety + (1 | block)
##         Df    AIC    BIC  logLik deviance Chisq Chi Df Pr(>Chisq)
## model.0  3 446.94 452.01 -220.47   440.94
## model.1 10 421.67 438.56 -200.84   401.67 39.27      7  1.736e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that this is doing a Likelihood Ratio Test

$$-2 * \log\left(\frac{L_0}{L_a}\right) = -2\left(\log(L_0) - \log(L_a)\right) = -2 * (-220.47 - -200.56) = 39.24$$

This shows that the variety matters, though this is pretty annoying. We'd prefer to use the `anova` command with just model and see the p-values for each covariate.

```
anova(model.1)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##         Sum Sq Mean Sq NumDF DenDF F value    Pr(>F)
## variety  77524   11075     7    28  8.2839 1.804e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is quite a bit of debate among statisticians about which test should be recommended in different scenarios using random effects and this is an active area of research. In this case, instead of performing a LRT, the `lmerTest` package opted to use a Satterthwaite approximation.

Now that we have chosen our model, we can examine is model.

```
summary(model.1)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: yield ~ variety + (1 | block)
##    Data: oatvar
##
## REML criterion at convergence: 341.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.7135 -0.5503 -0.1280  0.4862  2.1756
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  block    (Intercept)  876.5   29.61
##  Residual             1336.9   36.56
## Number of obs: 40, groups:  block, 5
##
## Fixed effects:
##             Estimate Std. Error     df t value Pr(>|t|)
## (Intercept)   334.40      21.04  15.26  15.894 6.64e-11 ***
## variety2       42.20      23.12  28.00   1.825   0.0787 .
## variety3       28.20      23.12  28.00   1.219   0.2328
## variety4      -47.60      23.12  28.00  -2.058   0.0490 *
## variety5      105.00      23.12  28.00   4.541 9.73e-05 ***
## variety6       -3.80      23.12  28.00  -0.164   0.8707
## variety7      -16.00      23.12  28.00  -0.692   0.4947
## variety8       49.80      23.12  28.00   2.154   0.0400 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) varty2 varty3 varty4 varty5 varty6 varty7
## variety2 -0.550
## variety3 -0.550  0.500
## variety4 -0.550  0.500  0.500
## variety5 -0.550  0.500  0.500  0.500
## variety6 -0.550  0.500  0.500  0.500  0.500
## variety7 -0.550  0.500  0.500  0.500  0.500  0.500
## variety8 -0.550  0.500  0.500  0.500  0.500  0.500  0.500
```

We start with the Random effects. This section shows us the *block-to-block* variability (and the square root of that, the Standard Deviation) as well as the "pure-error", labeled residuals, which is an estimate of the variability associated with two different observations (after the difference in variety is accounted for) planted *within* the same block. For this we see that block-to-block variability is only slightly smaller than the within block variability.

Why do we care about this? This actually tells us quite a lot about the spatial variability. Because yield is affected by soil nutrients, micro-climate, soil water availability, etc, I expect that two identical seedlings planted in slightly different conditions will have slightly different yields. By examining how the yield changes over small distances (the residual within block variability) vs how it changes over long distances (block to block variability) we can get a sense as to the scale at which these background lurking processes operate.

Next we turn to the fixed effects. These will be the offsets from the reference group, as we've typically worked with. Here we see that varieties 2,5, and 8 are the best performers (relative to variety 1),

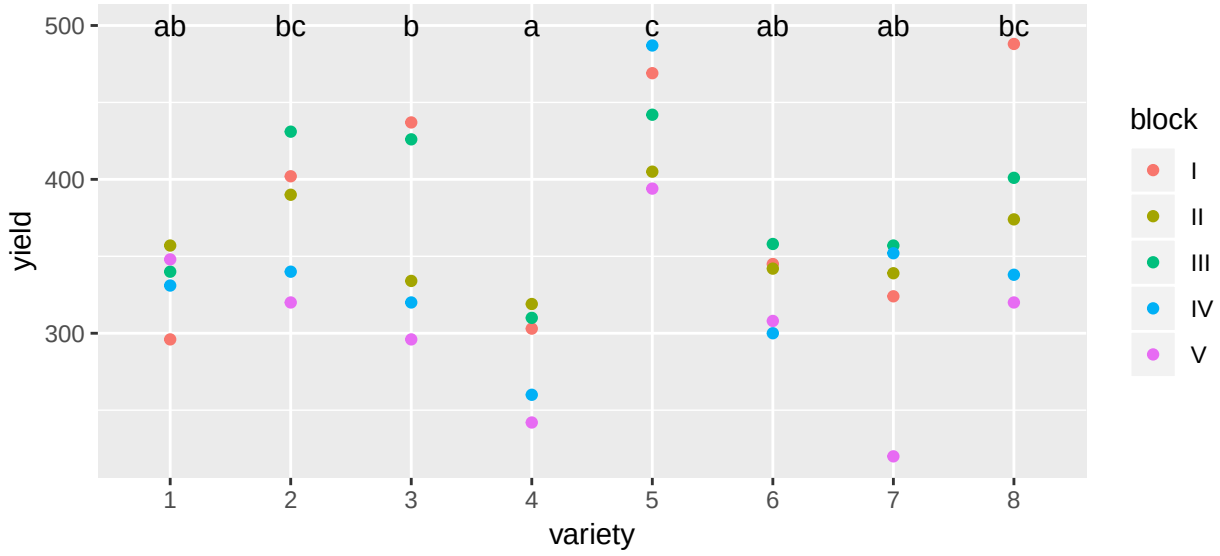
We are certain that there are differences among the varieties, and we should look at all of the pairwise contrasts among the variety levels. As usual we could use the package `emmeans`, which automates much of this (and uses `lmerTest` produced p-values for the tests).

```
LetterResults <- emmeans( model.1, ~ variety) %>% cld(Letters=letters)
LetterResults
```

```
##  variety emmean       SE    df lower.CL upper.CL .group
##  4        286.8 21.03996 15.25 242.0196 331.5804  a
##  7        318.4 21.03996 15.25 273.6196 363.1804  ab
##  6        330.6 21.03996 15.25 285.8196 375.3804  ab
##  1        334.4 21.03996 15.25 289.6196 379.1804  ab
##  3        362.6 21.03996 15.25 317.8196 407.3804   b
##  2        376.6 21.03996 15.25 331.8196 421.3804   bc
##  8        384.2 21.03996 15.25 339.4196 428.9804   bc
##  5        439.4 21.03996 15.25 394.6196 484.1804    c
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 8 estimates
## significance level used: alpha = 0.05
```

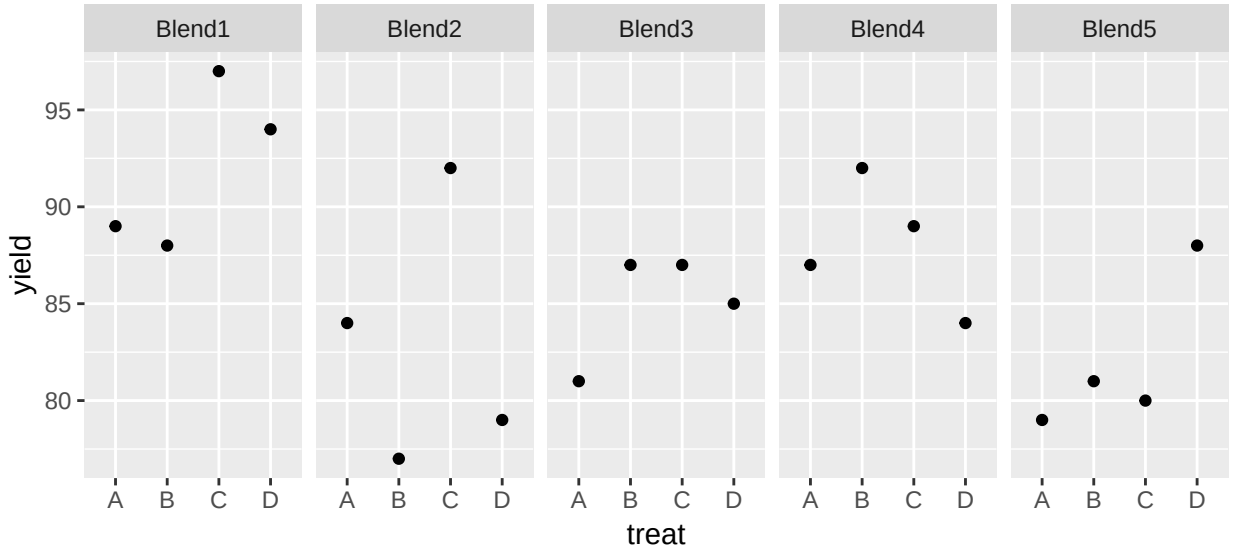
As usual we'll join this information into the original data table and then make a nice summary graph.

```
LetterResults <- LetterResults %>%
  mutate(LetterHeight=500,  .group = str_trim(.group))

ggplot(oatvar, aes(x=variety, y=yield)) +
  geom_point(aes(color=block)) +
  geom_text(data=LetterResults, aes(label=.group, y=LetterHeight))
```



We'll consider a second example using data from the pharmaceutical industry. We are interested in 4 different processes (our treatment variable) used in the biosynthesis and purification of the drug penicillin. The biosynthesis requires a nutrient source (corn steep liquor) as a nutrient source for the fungus and the nutrient source is quite variable. Each batch of the nutrient is is referred to as a 'blend' and each blend is sufficient to create 4 runs of penicillin. We avoid confounding our biosynthesis methods with the blend by using a Randomized Complete Block Design and observing the yield of penicillin from each of the four methods (A,B,D, and D) in each blend.

```
data(penicillin, package='faraway')
ggplot(penicillin, aes(y=yield, x=treat)) +
  geom_point() +
  facet_wrap( ~ blend, ncol=5)
```



It looks like there is definitely a `Blend` effect (e.g. Blend1 is much better than Blend5) but it isn't clear that there is a treatment effect.

```
model.0 <- lmer(yield ~  1    + (1 | blend), data=penicillin)
model.1 <- lmer(yield ~ treat + (1 | blend), data=penicillin)
anova(model.0, model.1)  # Analysis using a LRT
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: penicillin
## Models:
## model.0: yield ~ 1 + (1 | blend)
## model.1: yield ~ treat + (1 | blend)
##         Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## model.0  3 127.33 130.31 -60.662   121.33
## model.1  6 129.28 135.25 -58.639   117.28 4.0474      3     0.2564
```

```
anova(model.1)  # using whatever lmerTest thinks is appropriate
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##       Sum Sq Mean Sq NumDF DenDF F value Pr(>F)
## treat     70  23.333     3    12  1.2389 0.3387
```

It looks like we don't have a significant effect of the treatments. Next we'll examine the simple model to understand the variability.

```
summary(model.0)   # the lack of fixed effects caused lmerTest to revert to lme4
```

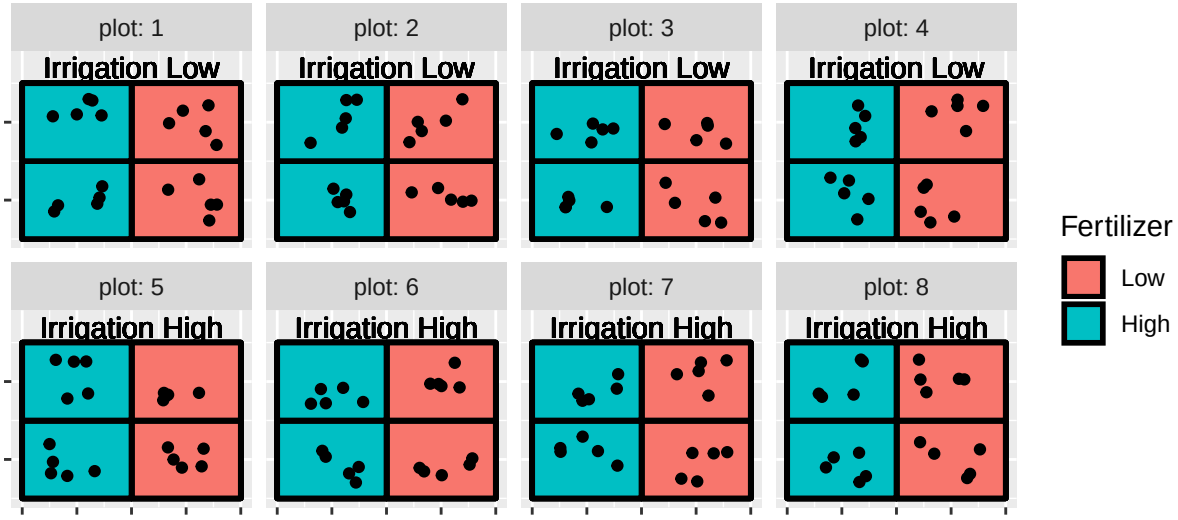
```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: yield ~ 1 + (1 | blend)
##    Data: penicillin
##
## REML criterion at convergence: 118.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.5526 -0.7310 -0.0789  0.5007  1.8241
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  blend    (Intercept) 11.57    3.401
##  Residual             19.73    4.442
## Number of obs: 20, groups:  blend, 5
##
## Fixed effects:
##             Estimate Std. Error     df t value Pr(>|t|)
## (Intercept)   86.000      1.817  4.000   47.34 1.19e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that the noise is more in the within blend rather than the between blends. If my job were to understand the variability and figure out how to improve production, this suggests that understanding the both how variability is introduced at the blend level *and* at the run level. The run level has slightly more variability, so I might start there.

11.4 Nested Effects

When the levels of one factor vary only within the levels of another factor, that factor is said to be nested. For example, when measuring the performance of workers at several job locations, if the workers only work at one site, then the workers are nested within site. If the workers work at more than one location, we would say that workers are *crossed* with site.

We've already seen a number of nested designs when we looked at split plot designs. Recall the `AgData` set that I made up that simulated an agricultural experiment with 8 plots and 4 subplots per plot. We applied an irrigation treatment at the plot level and a fertilizer treatment at the subplot level. I actually have 5 replicate observations per subplot.



So all together we have 8 plots, 32 subplots, and 5 replicates per subplot. When I analyze the fertilizer, I have 32 experimental units (the thing I have applied my treatment to), but when analyzing the effect of irrigation, I only have 8 experimental units. In other words, I should have 8 random effects for plot, and 32 random effects for subplot.

```
# The following model definitions are equivalent
model <- lmer(yield ~ Irrigation + Fertilizer + (1|plot) + (1|plot:subplot), data=AgData )
model <- lmer(yield ~ Irrigation + Fertilizer + (1|plot/subplot), data=AgData)
anova(model)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##             Sum Sq Mean Sq NumDF DenDF F value    Pr(>F)
## Irrigation  3.4769  3.4769     1     6  3.4281    0.1136
## Fertilizer 31.3825 31.3825     1    23 30.9421 1.168e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we saw before, the effect of irrigation is not significant and the fertilizer effect is highly significant. We'll remove the irrigation covariate and refit the model.

```
model <- lmer(yield ~ Fertilizer + (1|plot/subplot), data=AgData)
summary(model)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: yield ~ Fertilizer + (1 | plot/subplot)
##    Data: AgData
##
## REML criterion at convergence: 572.6
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.78714 -0.62878 -0.08602  0.64093  2.36353
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  subplot:plot (Intercept) 5.345    2.312
##  plot         (Intercept) 8.854    2.975
##  Residual                 1.014    1.007
```

```
## Number of obs: 160, groups:  subplot:plot, 32; plot, 8
##
## Fixed effects:
##                Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)     21.0211     1.2056  8.9745  17.436 3.14e-08 ***
## FertilizerHigh   4.6323     0.8328 23.0000   5.563 1.17e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## FertilzrHgh -0.345
```

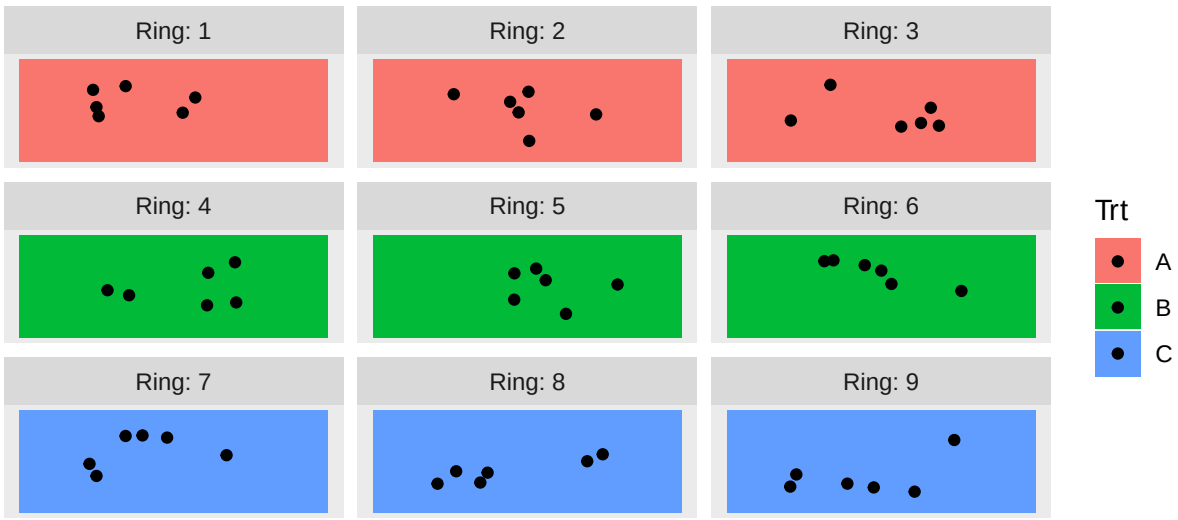
Notice the plant-to-plant noise is about 1/3 of the noise associated with subplot-to-subplot or even plot-to-plot.

A number of *in-situ* experiments looking at the addition CO_2 and warming on landscapes have been done (typically called Free Air CO_2 Experiments (FACE)) and these are interesting from an experimental design perspective because we have limited number of replicates because the cost of exposing plants to different CO_2 levels outside a greenhouse is extraordinarily expensive. In the `dsData` package, there is a dataset that is inspired by one of those studies.

The experimental units for the CO_2 treatment will be called a ring, and we have nine rings. We have three treatments (A,B,C) which correspond to an elevated CO_2 treatment, an ambient CO_2 treatment with all the fans, and a pure control. For each ring we'll have some measure of productivity but we have six replicate observations per ring.

```
data("HierarchicalData", package = 'dsData')
head(HierarchicalData)
```

```
##   Trt Ring Rep        y
## 1   A    1   1 363.9684
## 2   A    1   2 312.0613
## 3   A    1   3 332.9916
## 4   A    1   4 320.0109
## 5   A    1   5 292.2656
## 6   A    1   6 315.8136
```



We can easily fit this model using random effects for each ring.

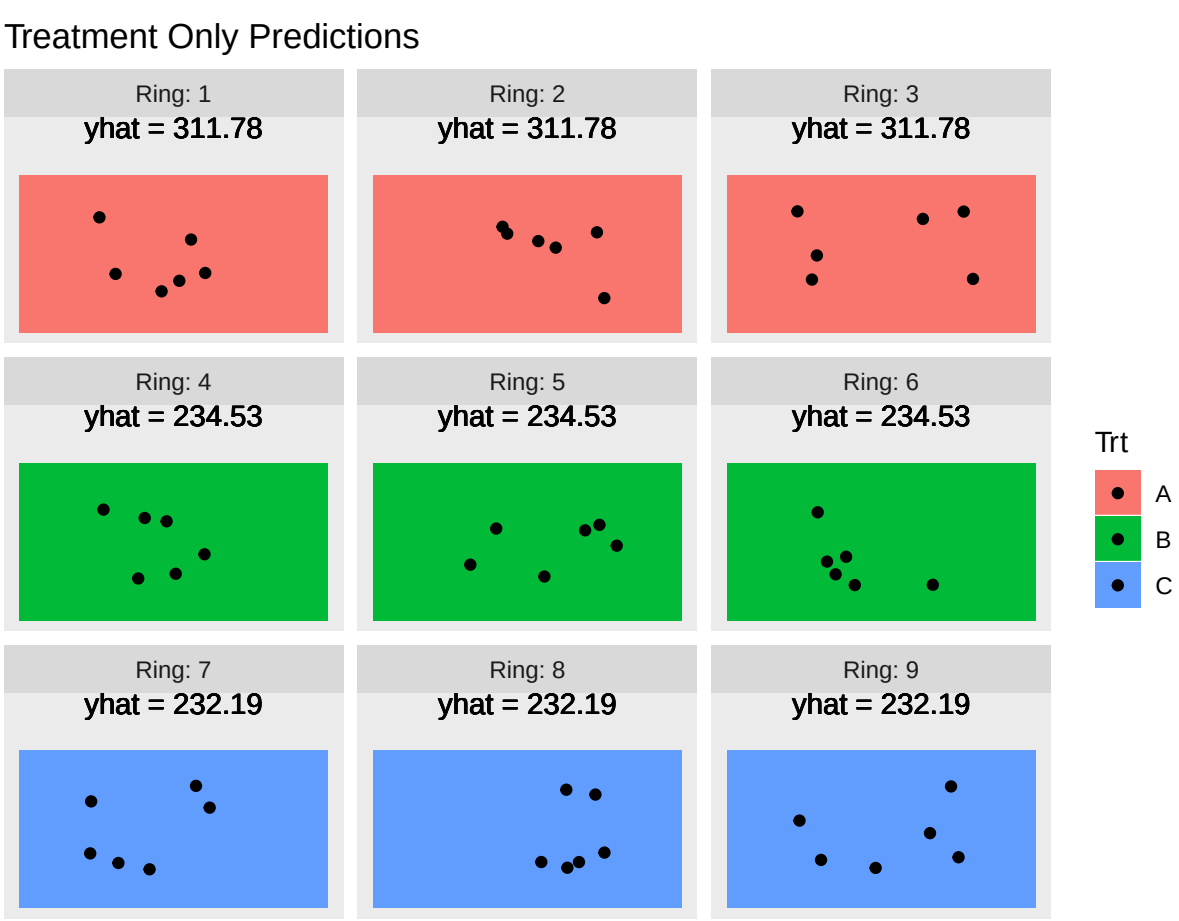
```
model <- lmer( y ~ Trt + (1|Ring), data=HierarchicalData )
anova(model)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##     Sum Sq Mean Sq NumDF DenDF F value  Pr(>F)
## Trt  10776  5388.1     2     6  5.7176 0.04075 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To think about what is actually going on, it is helpful to consider the predicted values from this model. As usual we will use the `predict` function, but now we have the option of including the random effects or not.

First lets consider the predicted values if we completely ignore the Ring random effect while making predictions.

```
HierarchicalData <- HierarchicalData %>%
  mutate( y.hat   = predict(model, re.form= ~ 0),  # don't include any random effects
          y.hat   = round( y.hat, digits=2),
          my.text = paste('yhat =', y.hat),
          text.height = 1.8)
```



Now we consider the predicted values, but created using the Ring random effect. These random effects provide for a slight perturbation up or down depending on the quality of the Ring, but the sum of all 9 Ring effects is *required* to be 0.

```
ranef(model)
```

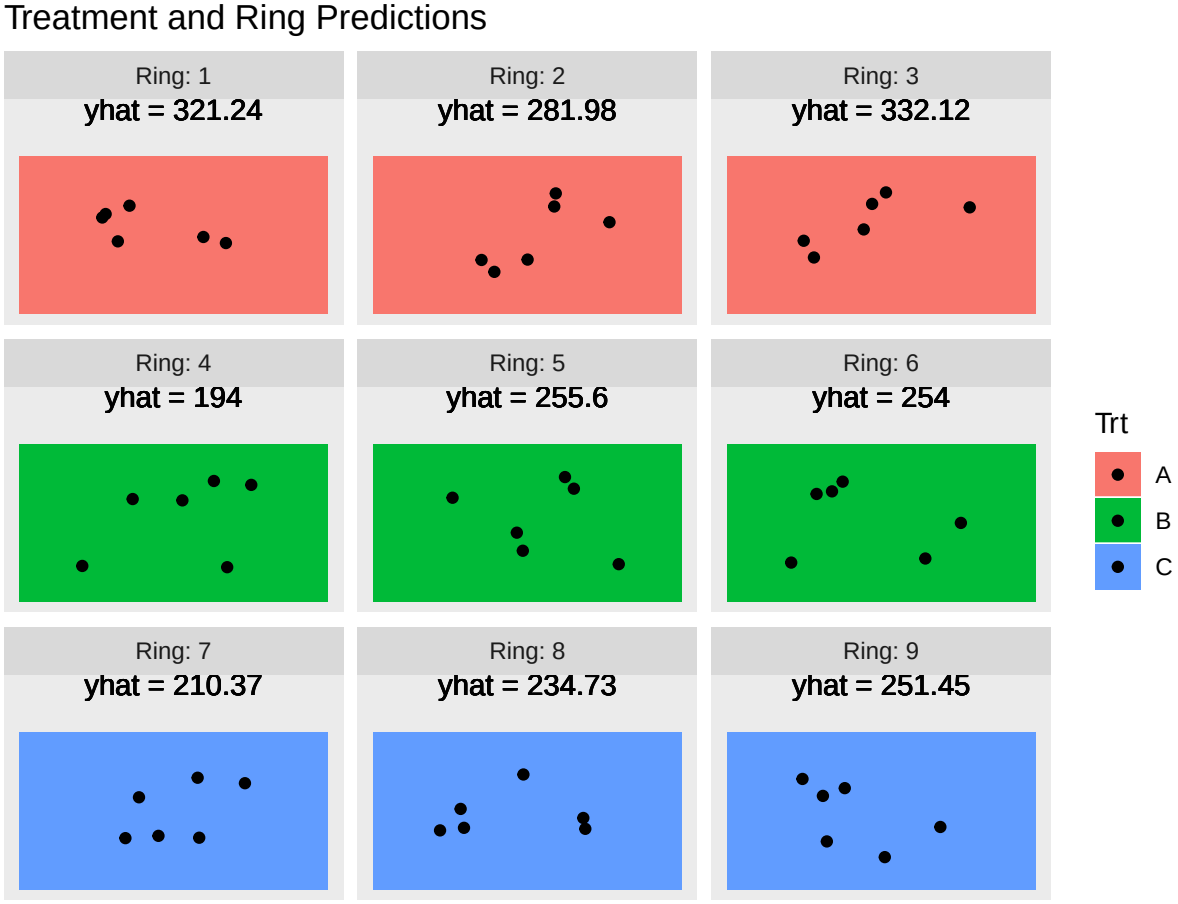
```
## $Ring
##   (Intercept)
## 1    9.458738
## 2  -29.798425
## 3   20.339687
## 4  -40.532972
## 5   21.067503
## 6   19.465469
## 7  -21.814405
## 8    2.548287
## 9   19.266118
```

```
sum(ranef(model)$Ring)
```

```
## [1] -5.317524e-12
```

Also notice that the sum of the random effects *within a treatment* is zero! (Recall Ring 1:3 was treatment A, 4:6 was treatment B, and 7:9 was treatment C).

```
HierarchicalData <- HierarchicalData %>%
  mutate( y.hat   = predict(model, re.form= ~ (1|Ring)),  # Include Ring Random effect
          y.hat   = round( y.hat, digits=2),
          my.text = paste('yhat =', y.hat),
          text.height = 1.8)
```



We interpret the random effect of Ring as a perturbation to expected value of the response that you expect just based on the treatment provided.

We'll now consider an example with a somewhat ridiculous amount of nesting. We will consider an experiment run to test the consistency between laboratories. A large jar of dried egg power was fully homogenized and divided into a number of samples and the fat content between the samples should be the same. Six laboratories were randomly selected and each lab would receive 4 samples, two labeled H and two labeled G. The labs are instructed to give two samples to two different technicians who are to divide each sample into two sub-samples and measures the fat content twice within a sub sample. So our hierarchy is that observations are nested within sub-samples which are nested within technicians which are nested in labs.

In terms of notation, we will refer to the 6 labs as L_i and the lab technicians as T_{ij} and we note that j is either 1 or 2 which doesn't uniquely identify the technician unless we include the lab subscript as well. Finally the sub-samples are nested within the technicians and we denote them as S_{ijk}. Finally our "pure" error is the two measurements from the same sub-sample. So the model we wish to fit is:

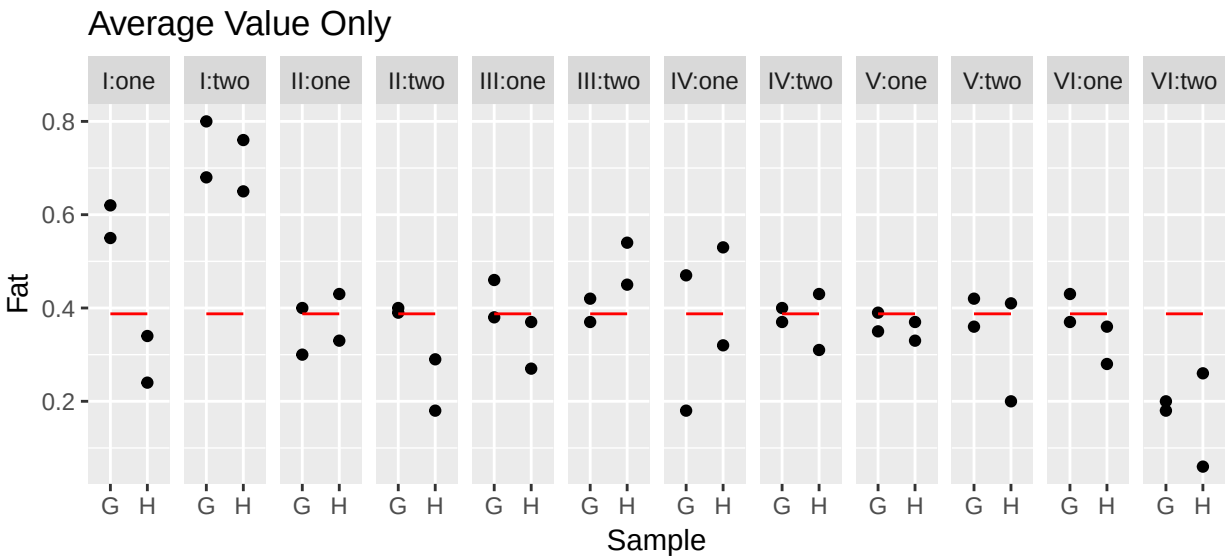
$$y_{ijkl} = \mu + L_i + T_{ij} + S_{ijk} + \epsilon_{ijkl}$$

where $L_i \overset{iid}{\sim} N\left(0, \sigma_L^2\right)$, $T_{ij} \overset{iid}{\sim} N\left(0, \sigma_T^2\right)$, $S_{ijk} \overset{iid}{\sim} N\left(0, \sigma_S^2\right)$, $\epsilon_{ijkl} \overset{iid}{\sim} N\left(0, \sigma_\epsilon^2\right)$.

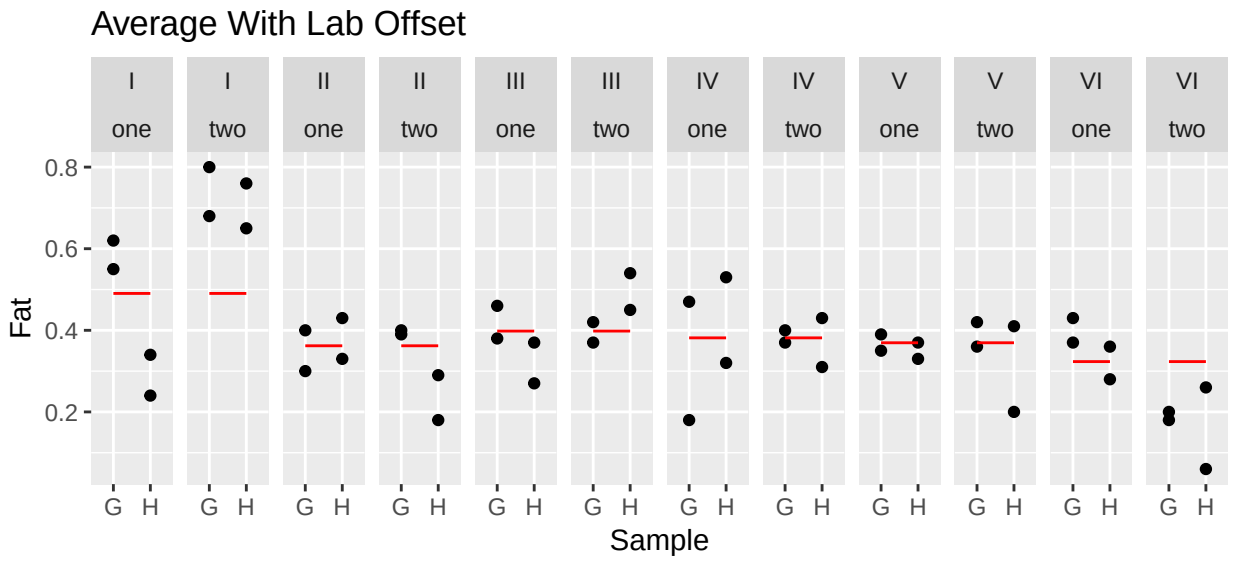
We need a convenient way to tell `lmer` which factors are nested in which. We can do this by creating data columns that make the interaction terms. For example there are 12 technicians (2 from each lab), but in our data frame we only see two levels, so to create all 12 random effects, we need to create an interaction column (or tell `lmer` to create it and use it). Likewise there are 24 sub-samples and 48 "pure" random effects.

```
data('eggs', package='faraway')
model <- lmer( Fat ~ 1 + (1|Lab) + (1|Lab:Technician) +
                     (1|Lab:Technician:Sample), data=eggs)
model <- lmer( Fat ~ 1 + (1|Lab/Technician/Sample), data=eggs)
```

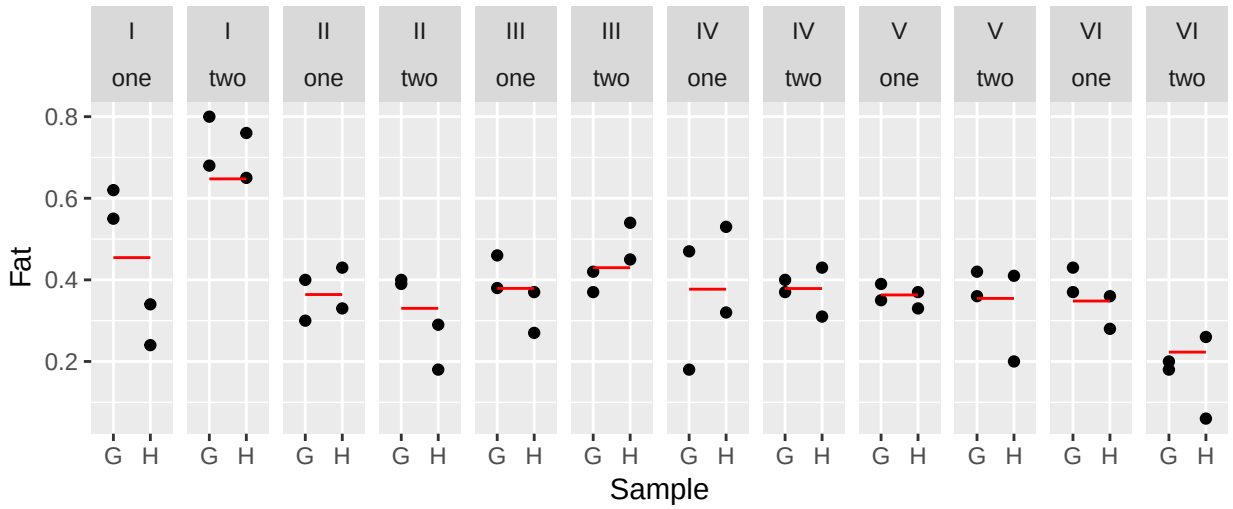
```
eggs <- eggs %>%
  mutate( yhat = predict(model, re.form=~0))
ggplot(eggs, aes(x=Sample, y=Fat)) +
  geom_point() +
  geom_line(aes(y=yhat, x=as.integer(Sample)), color='red') +
  facet_grid(. ~ Lab:Technician) +
  ggtitle('Average Value Only')
```



```
eggs <- eggs %>%
  mutate( yhat = predict(model, re.form=~(1|Lab)))
ggplot(eggs, aes(x=Sample, y=Fat)) +
  geom_point() +
  geom_line(aes(y=yhat, x=as.integer(Sample)), color='red') +
  facet_grid(. ~ Lab+Technician) +
  ggtitle('Average With Lab Offset')
```

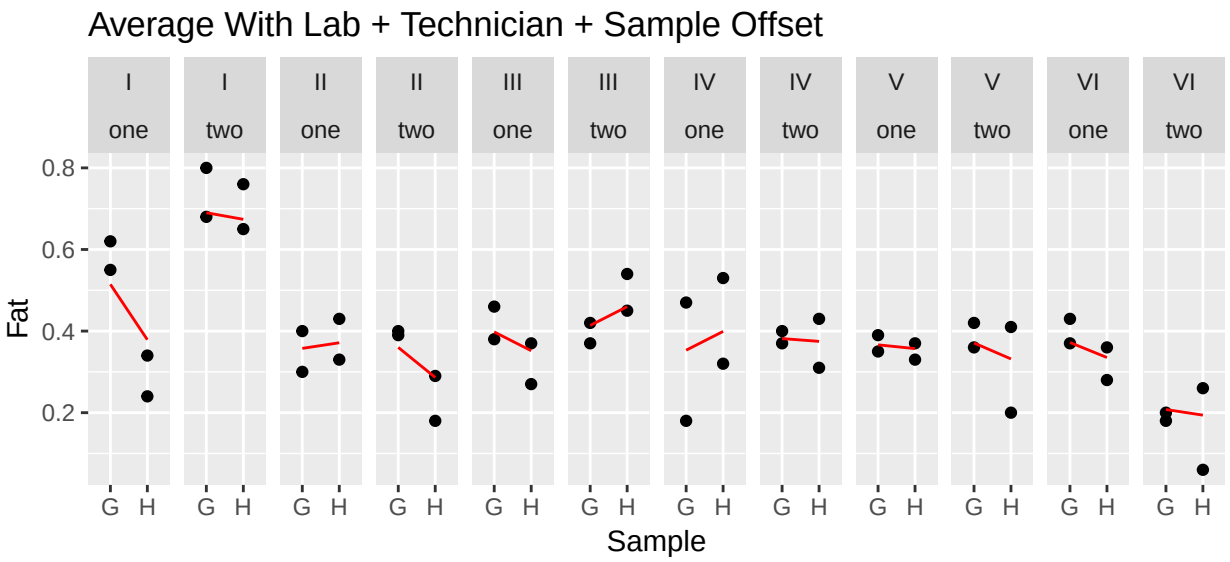



```
eggs <- eggs %>%
  mutate( yhat = predict(model, re.form=~(1|Lab/Technician)))
ggplot(eggs, aes(x=Sample, y=Fat)) +
  geom_point() +
  geom_line(aes(y=yhat, x=as.integer(Sample)), color='red') +
  facet_grid(. ~ Lab+Technician) +
  ggtitle('Average With Lab + Technician Offset')
```




```
eggs <- eggs %>%
  mutate( yhat = predict(model, re.form=~(1|Lab/Technician/Sample)))
```

```
ggplot(eggs, aes(x=Sample, y=Fat)) +
  geom_point() +
  geom_line(aes(y=yhat, x=as.integer(Sample)), color='red') +
  facet_grid(. ~ Lab+Technician) +
  ggtitle('Average With Lab + Technician + Sample Offset')
```



No that we have an idea of how things vary, we can look at the summary table.

```
summary(model)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: Fat ~ 1 + (1 | Lab/Technician/Sample)
##    Data: eggs
##
## REML criterion at convergence: -64.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.04098 -0.46576  0.00927  0.59713  1.54276
##
## Random effects:
##  Groups                  Name        Variance Std.Dev.
##  Sample:(Technician:Lab) (Intercept) 0.003065 0.05536
##  Technician:Lab          (Intercept) 0.006980 0.08355
##  Lab                     (Intercept) 0.005920 0.07694
##  Residual                            0.007196 0.08483
## Number of obs: 48, groups:
## Sample:(Technician:Lab), 24; Technician:Lab, 12; Lab, 6
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)  0.38750    0.04296 5.00000   9.019  0.00028 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

11.5 Crossed Effects

If two effects are not nested, we say they are *crossed*. In the penicillin example, the treatments and blends were not nested and are therefore crossed.

An example is a Latin square experiment to look the effects of abrasion on four different material types (A, B, C, and D). We have a machine to do the abrasion test with four positions and we did 4 different machine runs. Our data looks like the following setup:

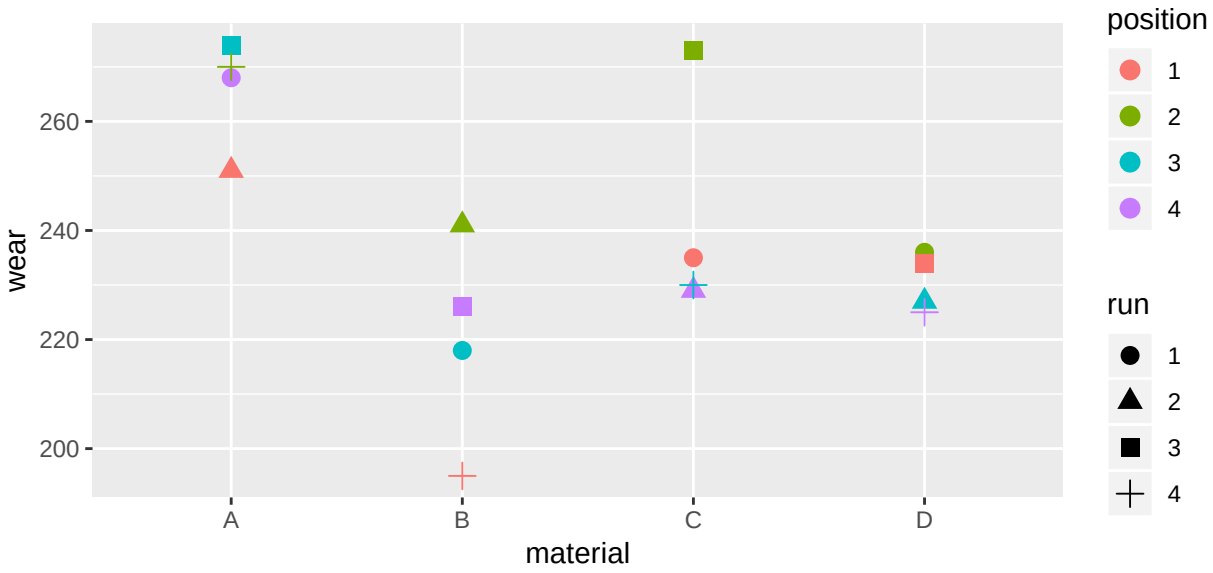
run	Position: 1	Position: 2	Position: 3	Position: 4
1	C	D	B	A
2	A	B	D	C
3	D	C	A	B
4	B	A	C	D

Our model can be written as

$$y_{ijk} = \mu + M_i + P_j + R_k + \epsilon_{ijk}$$

and we notice that the position and run effects are not nested within anything else and thus the subscript have just a single index variable. Certainly the run effect should be considered random as these four are a sample from all possible runs, but what about the position variable? Here we consider that the machine being used is a random selection from all possible abrasion machines and any position differences have likely developed over time and could be considered as a random sample of possible position effects. We'll regard both position and run as crossed random effects.

```
data('abrasion', package='faraway')
ggplot(abrasion, aes(x=material, y=wear, color=position, shape=run)) +
  geom_point(size=3)
```



It certainly looks like the materials are different. I don't think the run matters, but position 2 seems to develop excessive wear compared to the other positions.

```
m <- lmer( wear ~ material + (1|run) + (1|position), data=abrasion)
anova(m)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##          Sum Sq Mean Sq NumDF DenDF F value    Pr(>F)
```

```
## material 4621.5  1540.5     3     6  25.151 0.0008498 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The material effect is statistically significant and we can figure out the pairwise differences in the usual fashion.

```
emmeans(m, specs= pairwise~material)
```

```
## $emmeans
##  material emmean       SE   df lower.CL upper.CL
##  A        265.75 7.668252 7.48 247.8485 283.6515
##  B        220.00 7.668252 7.48 202.0985 237.9015
##  C        241.75 7.668252 7.48 223.8485 259.6515
##  D        230.50 7.668252 7.48 212.5985 248.4015
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate       SE df t.ratio p.value
##  A - B       45.75 5.533986  6   8.267  0.0007
##  A - C       24.00 5.533986  6   4.337  0.0190
##  A - D       35.25 5.533986  6   6.370  0.0029
##  B - C      -21.75 5.533986  6  -3.930  0.0295
##  B - D      -10.50 5.533986  6  -1.897  0.3206
##  C - D       11.25 5.533986  6   2.033  0.2743
##
## P value adjustment: tukey method for comparing a family of 4 estimates
```

```
emmeans(m, specs= ~material) %>% cld(Letters=letters)
```

```
##  material emmean       SE   df lower.CL upper.CL .group
##  B        220.00 7.668252 7.48 202.0985 237.9015  a
##  D        230.50 7.668252 7.48 212.5985 248.4015  ab
##  C        241.75 7.668252 7.48 223.8485 259.6515   b
##  A        265.75 7.668252 7.48 247.8485 283.6515    c
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 4 estimates
## significance level used: alpha = 0.05
```

So material D is in between materials B and C for abrasion resistance.

```
summary(m)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: wear ~ material + (1 | run) + (1 | position)
##    Data: abrasion
##
## REML criterion at convergence: 100.3
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.08973 -0.30231  0.02697  0.42254  1.21052
```

```
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  run      (Intercept)  66.90    8.179
##  position (Intercept) 107.06   10.347
##  Residual              61.25    7.826
## Number of obs: 16, groups:  run, 4; position, 4
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)  265.750      7.668   7.475  34.656 1.57e-09 ***
## materialB    -45.750      5.534   6.000  -8.267 0.000169 ***
## materialC    -24.000      5.534   6.000  -4.337 0.004892 **
## materialD    -35.250      5.534   6.000  -6.370 0.000703 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##           (Intr) matrlB matrlC
## materialB -0.361
## materialC -0.361  0.500
## materialD -0.361  0.500  0.500
```

Notice that run and the pure error have about the same magnitude, but position is more substantial. Lets see what happens if we remove the run random effect.

```
m2 <- lmer( wear ~ material + (1|position), data=abrasion)
anova(m, m2)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: abrasion
## Models:
## m2: wear ~ material + (1 | position)
## m: wear ~ material + (1 | run) + (1 | position)
##    Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## m2  6 137.74 142.38 -62.870   125.74
## m   7 134.32 139.73 -60.162   120.32 5.4164      1    0.01995 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that R is refitting the model to make an appropriate comparison. The AIC difference between the two models is about 3 units (the larger model having a lower AIC) and so we could interpret this as decent evidence for a run effect. Similarly the Likelihood Ratio Test gives a p-value of about 0.02. So while the run effect wasn't visible in our initial graph, it looks like it is a statistically significant effect.

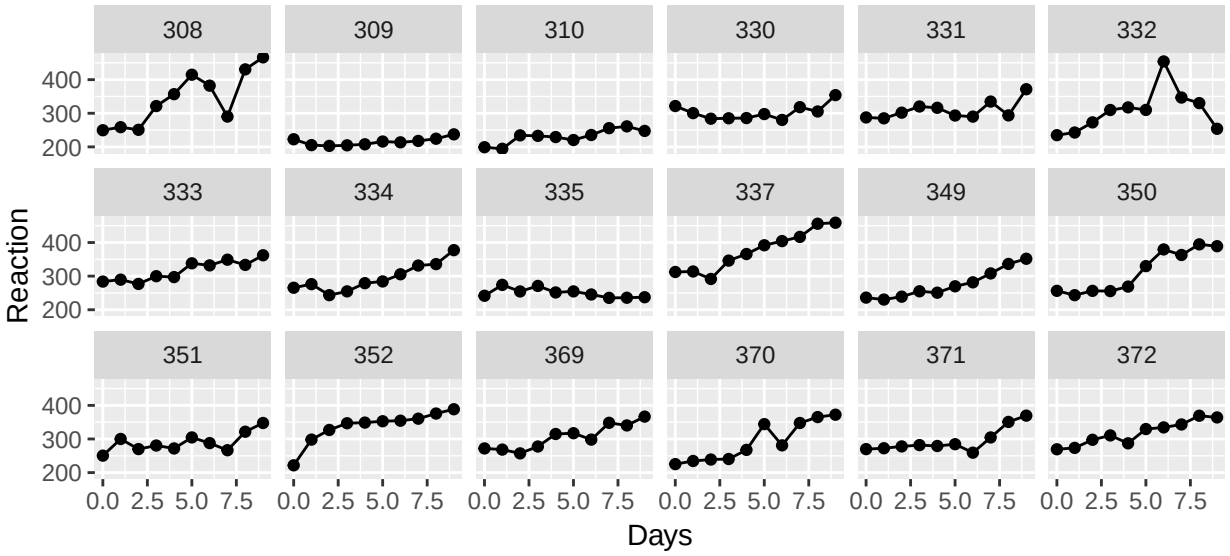
11.6 Repeated Measures / Longitudinal Studies

In repeated measurement experiments, repeated observations are taken on each subject. When those repeated measurements are taken over a sequence of time, we call it a longitudinal study. Typically covariates are also observed at the same time points and we are interested in how the response is related to the covariates.

In this case the correlation structure is that observations on the same person/object should be more similar than observations between two people/objects. As a result we need to account for repeated measures by including the person/object as a random effect.

To demonstrate a longitudinal study we turn to the data set `sleepstudy` in the `lme4` library. Eighteen patients participated in a study in which they were allowed only 3 hours of sleep per night and their reaction time in a specific test was observed. On day zero (before any sleep deprivation occurred) their reaction times were recorded and then the measurement was repeated on 9 subsequent days.

```
data('sleepstudy', package='lme4')
ggplot(sleepstudy, aes(y=Reaction, x=Days)) +
    facet_wrap(~ Subject, ncol=6) +
    geom_point() +
    geom_line()
```



We want to fit a line to these data, but how should we do this? First we notice that each subject has their own baseline for reaction time and the subsequent measurements are relative to this, so it is clear that we should fit a model with a random intercept.

```
m1 <- lmer( Reaction ~ Days + (1|Subject), data=sleepstudy)
summary(m1)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: Reaction ~ Days + (1 | Subject)
##    Data: sleepstudy
##
## REML criterion at convergence: 1786.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2257 -0.5529  0.0109  0.5188  4.2506
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Subject  (Intercept) 1378.2   37.12
##  Residual              960.5   30.99
## Number of obs: 180, groups:  Subject, 18
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept) 251.4051     9.7467  22.8102   25.79   <2e-16 ***
```

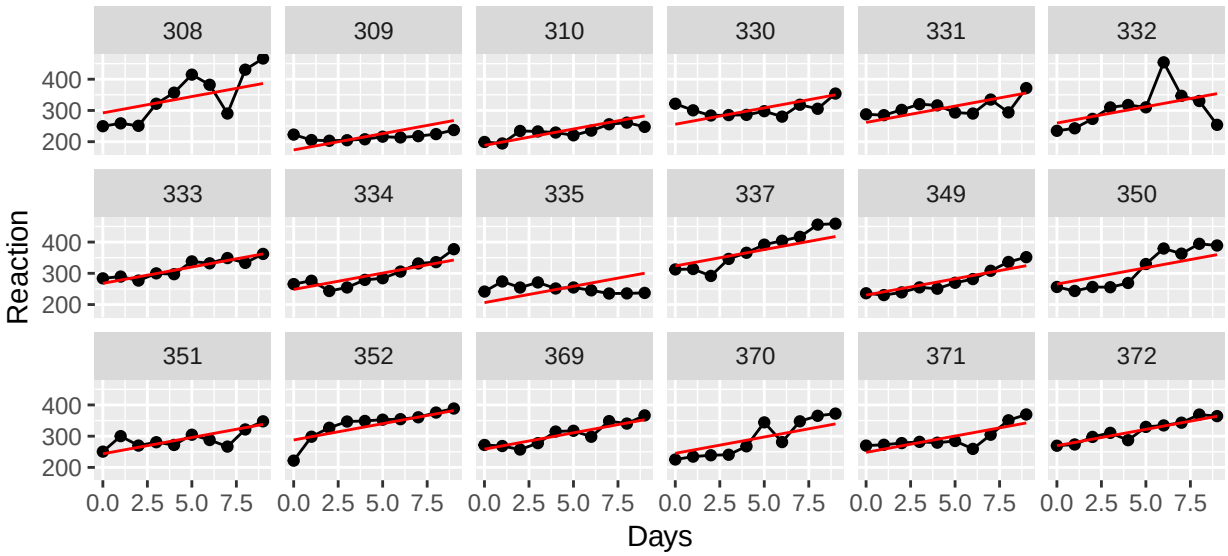
```
## Days         10.4673     0.8042 161.0000   13.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr)
## Days -0.371
```

```
ranef(m1)
```

```
## $Subject
##     (Intercept)
## 308   40.783710
## 309  -77.849554
## 310  -63.108567
## 330    4.406442
## 331   10.216189
## 332    8.221238
## 333   16.500494
## 334   -2.996981
## 335  -45.282127
## 337   72.182686
## 349  -21.196249
## 350   14.111363
## 351   -7.862221
## 352   36.378425
## 369    7.036381
## 370   -6.362703
## 371   -3.294273
## 372   18.115747
```

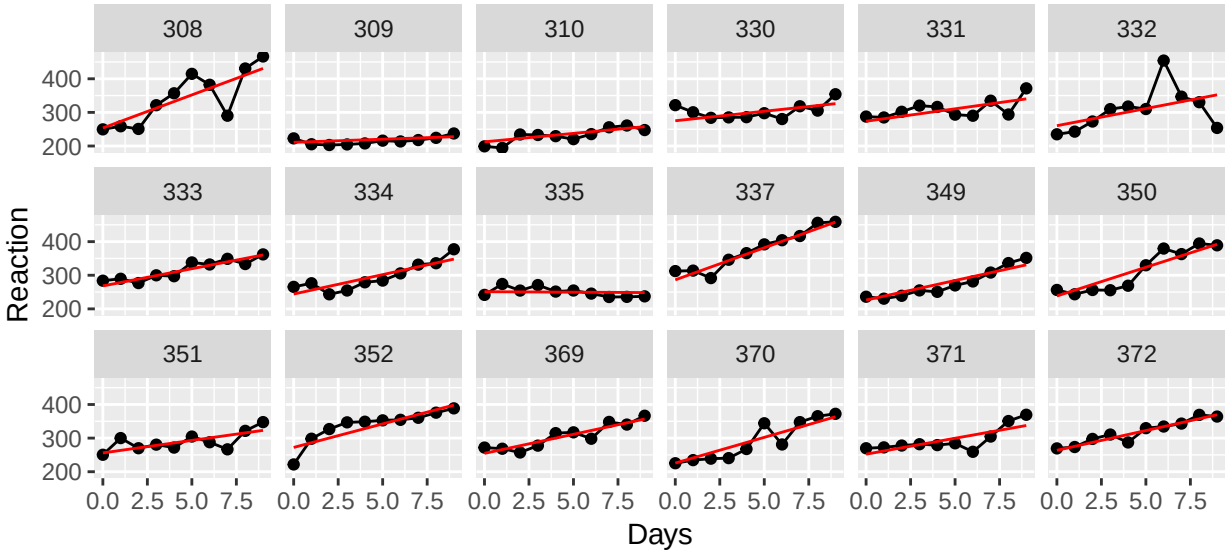
To visualize how well this model fits our data, we will plot the predicted values which are lines with y-intercepts that are equal to the sum of the fixed effect of intercept and the random intercept per subject. The slope for each patient is assumed to be the same and is approximately 10.4.

```
sleepstudy <- sleepstudy %>%
  mutate(yhat = predict(m1, re.form=~(1|Subject)))
ggplot(sleepstudy, aes(y=Reaction, x=Days)) +
    facet_wrap(~ Subject, ncol=6) +
    geom_point() +
    geom_line() +
    geom_line(aes(y=yhat), color='red')
```



This isn't too bad, but I would really like to have each patient have their own slope as well as their own y-intercept. The random slope will be calculated as a fixed effect of slope plus a random offset from that.

```
# Random effects for intercept and Slope
m2 <- lmer( Reaction ~ Days + ( 1+Days | Subject), data=sleepstudy)

sleepstudy <- sleepstudy %>%
  mutate(yhat = predict(m2, re.form=~(1+Days|Subject)))
ggplot(sleepstudy, aes(y=Reaction, x=Days)) +
    facet_wrap(~ Subject, ncol=6) +
    geom_point() +
    geom_line() +
    geom_line(aes(y=yhat), color='red')
```



This appears to fit the observed data quite a bit better, but it is useful to test this.

```
anova(m1, m2)
```

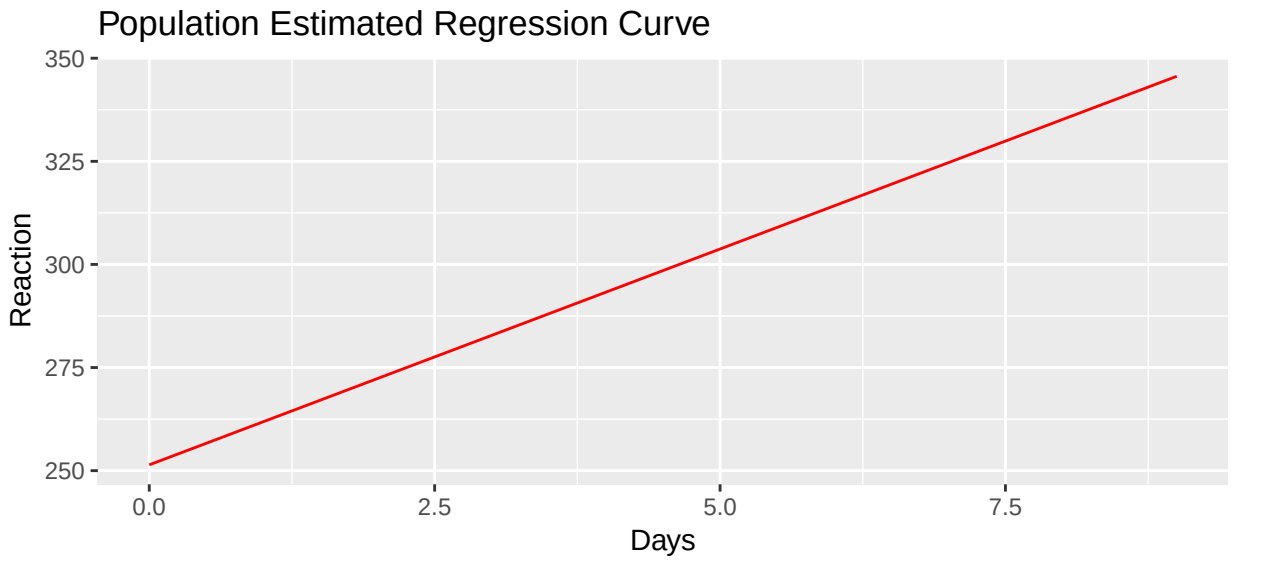
```
## refitting model(s) with ML (instead of REML)
```

```
## Data: sleepstudy
## Models:
## m1: Reaction ~ Days + (1 | Subject)
## m2: Reaction ~ Days + (1 + Days | Subject)
##    Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## m1  4 1802.1 1814.8 -897.04   1794.1
## m2  6 1763.9 1783.1 -875.97   1751.9 42.139      2  7.072e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

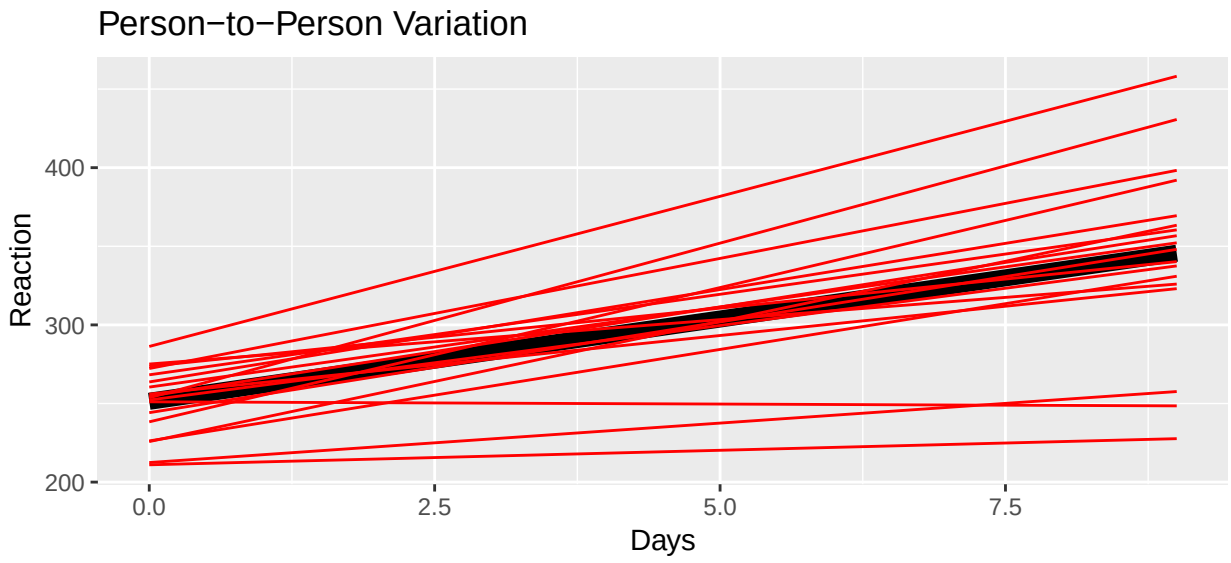
Here we see that indeed the random effect for each subject in both y-intercept and in slope is a better model that just a random offset in y-intercept.

It is instructive to look at this example from the top down. First we plot the population regression line.

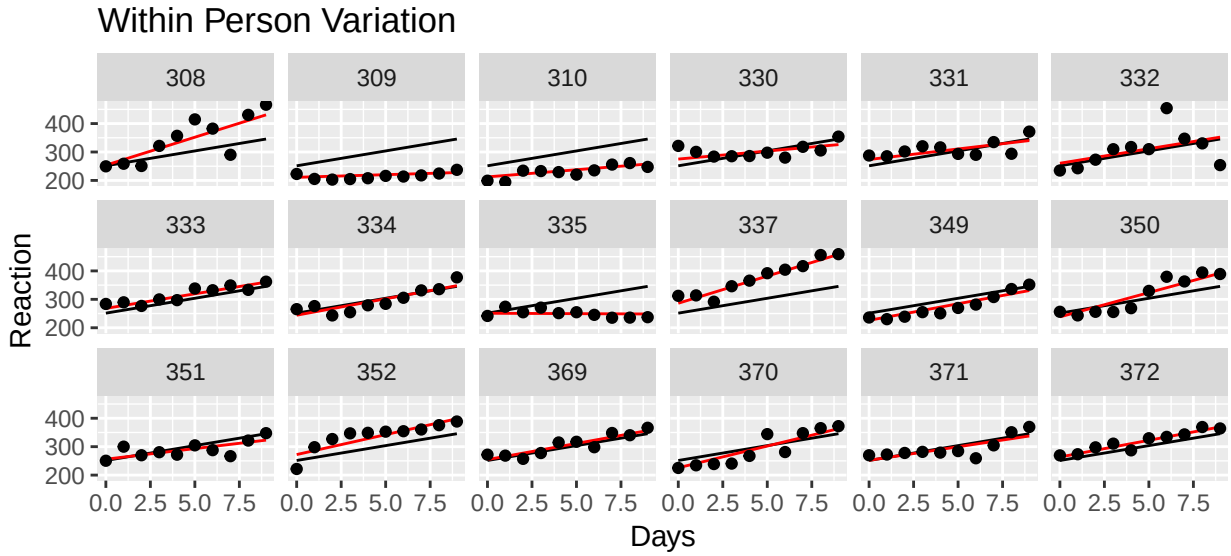
```
sleepstudy <- sleepstudy %>%
  mutate(yhat = predict(m2, re.form=~0))
ggplot(sleepstudy, aes(x=Days, y=yhat)) +
  geom_line(color='red') + ylab('Reaction') +
  ggtitle('Population Estimated Regression Curve')
```




```
sleepstudy <- sleepstudy %>%
  mutate(yhat.ind = predict(m2, re.form=~(1+Days|Subject)))
ggplot(sleepstudy, aes(x=Days)) +
  geom_line(aes(y=yhat), size=3) +
  geom_line(aes(y=yhat.ind, group=Subject), color='red') +
  ylab('Reaction') + ggtitle('Person-to-Person Variation')
```




```
ggplot(sleepstudy, aes(x=Days)) +
  geom_line(aes(y=yhat)) +
  geom_line(aes(y=yhat.ind, group=Subject), color='red') +
  ylab('Reaction') + ggtitle('Within Person Variation') +
  facet_wrap(~ Subject, ncol=6) +
  geom_point(aes(y=Reaction))
```



Finally we want to go back and look at the coefficients for the complex model.

```
summary(m2)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: Reaction ~ Days + (1 + Days | Subject)
##    Data: sleepstudy
##
## REML criterion at convergence: 1743.6
##
## Scaled residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.9536 -0.4634  0.0231  0.4634  5.1793
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  Subject  (Intercept) 612.09   24.740
##           Days         35.07    5.922   0.07
##  Residual             654.94   25.592
## Number of obs: 180, groups:  Subject, 18
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)  251.405      6.825  17.000  36.838  < 2e-16 ***
## Days          10.467      1.546  17.000   6.771 3.26e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr)
## Days -0.138
```

11.7 Confidence and Prediction Intervals

As with the standard linear model, we often want to create confidence and prediction intervals for a new observation or set of observations. Unfortunately, there isn't a nice way to easily incorporate the uncertainty of the variance components. Instead we have to rely on bootstapping techniques to produce these quantities. Fortunately the `lme4` package provides a function that will handle most of the looping required, but we have to describe to the program how to create the bootstrap samples, and given a bootstrap sample, what statistics do we want to produce intervals for.

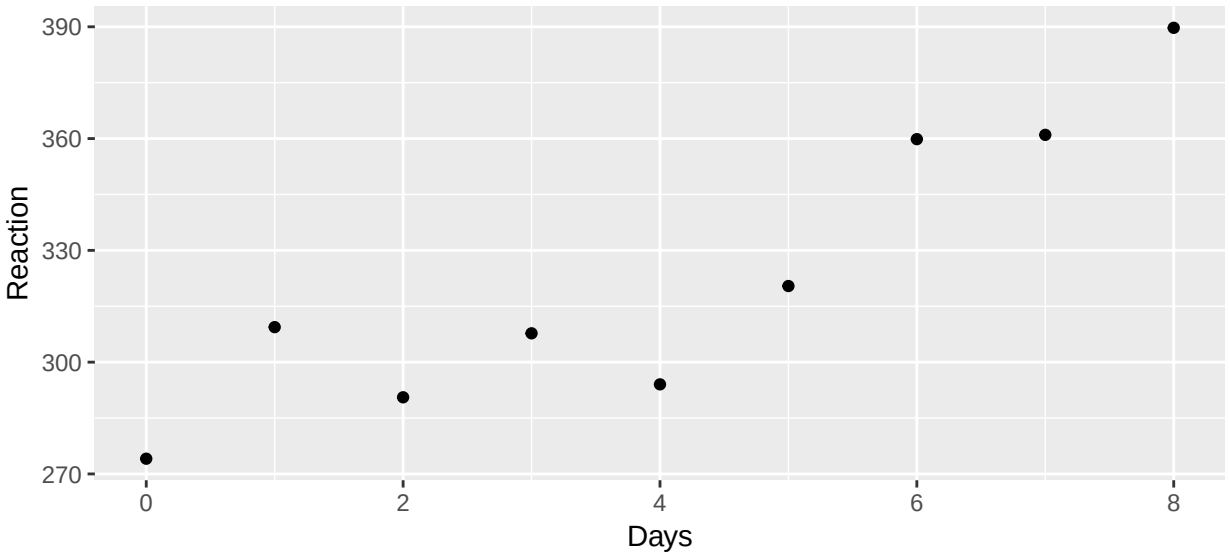
Typically the bootstrap is used when we don't want to make any distributional assumptions on the data. In that case, we sample with replacement from the observed data to create the bootstrap data. But, if we don't mind making distributional assumptions, then instead of resampling the data, we could sample from the distribution with the observed parameter. In our sleep study example, we have estimated a population intercept and slope of 251.4 and 10.5. But we also have a subject intercept and slope random effect which we assumed to be normally distributed centered at zero with and with estimated standard deviations of 24.7 and 5.9. Then given a subjects regression line, observations are just normal (mean zero, standard deviation 25.6) perterbations from the line. All of these numbers came from the `summary(m2)` output.

To create a bootstrap data simulating a new subject, we could do the following:

```
subject.intercept = 251.4 + rnorm( 1, mean = 0, sd=24.7)
subject.slope     = 10.5  + rnorm( 1, mean = 0, sd=10.5)
c(subject.intercept, subject.slope)
```

```
## [1] 258.67684  14.71306
```

```
subject.obs <- data.frame(Days = 0:8) %>%
  mutate( Reaction = subject.intercept + subject.slope*Days + rnorm(9, sd=25.6) )

ggplot(subject.obs, aes(x=Days, y=Reaction)) + geom_point()
```



This approach is commonly referred to as a "parametric" bootstrap because we are making some assumptions about the parameter distributions, whereas in a "nonparametric" bootstrap we don't make any distributional assumptions. By default, the `bootMer` function will perform a parametric bootstrap to create new bootstrap datasets and then analyze them using the same model you origally created.

11.7.1 Confidence Intervals

Now that we have a bootstrap data set, we need to take the data and then fit a model to the data and then grab the predictions from the model. At this point we are creating a confidence interval for the response line of a randomly selected person from the population. The `lme4::bootMer` function will create bootstrap data sets and then send those into the `lmer` function.

```
ConfData <- data.frame(Days=0:8)
myStats <- function(model){
  out <- predict( model, newdata=ConfData, re.form=~0 )
  return(out)
}

bootObj <- bootMer(m2, FUN=myStats, nsim = 1000)
# hist(bootObj)  # check for normality/skewness/etc

# BCa failed for me, but percentile is fine
ConfData <-  cbind( ConfData, car::Confint( bootObj,  level=0.95 ))
```

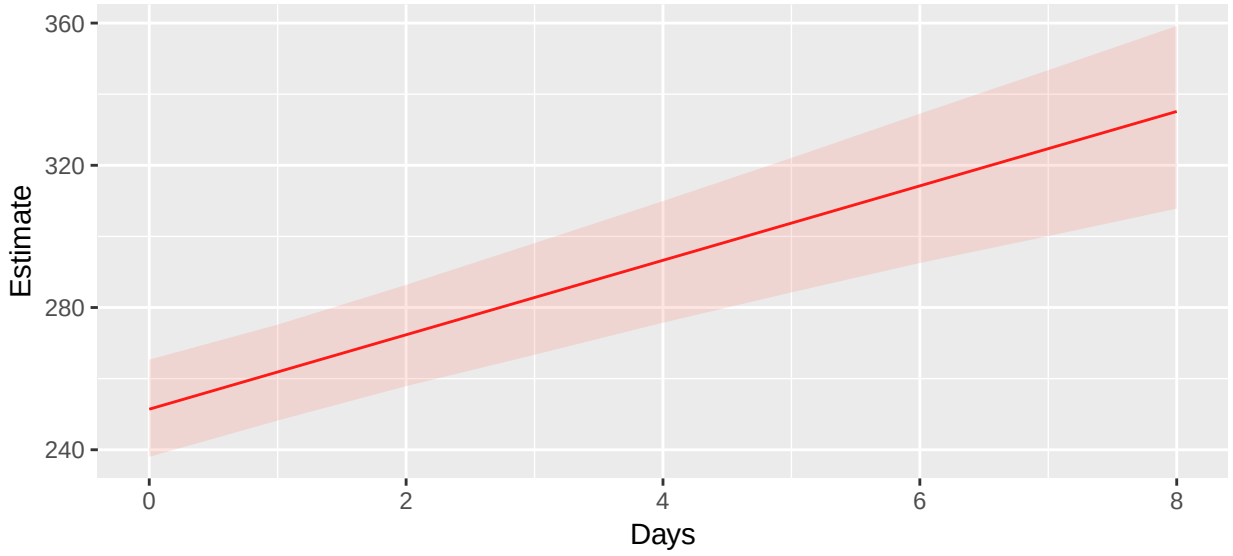
```
## Warning in confint.boot(object, parm, level, type, ...): BCa method fails
## for this problem. Using 'perc' instead
```

```
ConfData
```

```
##   Days Estimate    2.5 %   97.5 %
## 1    0 251.4051 238.0023 265.3373
## 2    1 261.8724 248.2272 275.1973
## 3    2 272.3397 257.8876 286.3786
## 4    3 282.8070 266.7016 298.1147
## 5    4 293.2742 275.7284 309.9396
## 6    5 303.7415 284.2544 322.0648
## 7    6 314.2088 292.5165 334.5147
```

```
## 8    7 324.6761 300.0826 346.7601
## 9    8 335.1434 307.8211 359.1806
```

```
# Now for a nice graph!  Unfortunately car::Confint used numbers for column names.
# So first I need to fix that.
colnames(ConfData) <- c('Days', 'Estimate', 'lwr', 'upr')
ggplot(ConfData,  aes(x=Days)) +
  geom_line(aes(y=Estimate), color='red') +
  geom_ribbon(aes(ymin=lwr, ymax= upr), fill='salmon', alpha=0.2)
```

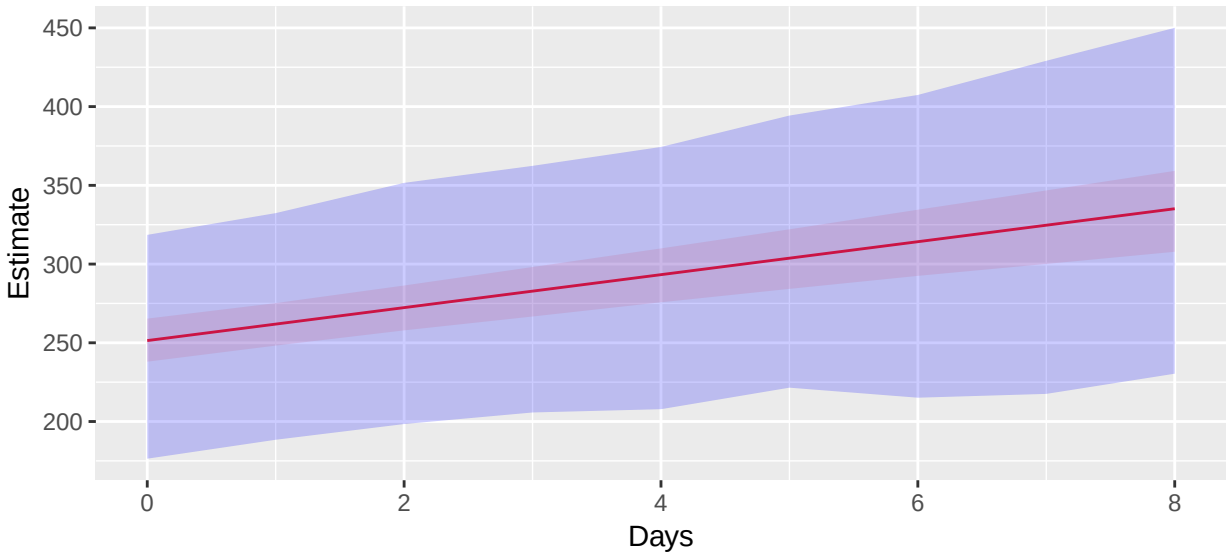


11.7.2 Prediction Intervals

For a confidence interval, we just want to find the range of observed values. In this case, we want to use the bootstrap data, but don't need to fit a model at each bootstrap step. The `lme4::simulate` function creates the bootstrap dataset and doesn't send it for more processing. It returns a vector of response values that are appropriately organized to be appended to the orginal dataset.

```
# set up the structure of new subjects
PredData <- data.frame(Subject='new', Days=0:8) # Simulate a NEW patient

# Create a n x 1000 data frame
Simulated <- simulate(m2, newdata=PredData, allow.new.levels=TRUE, nsim=1000)

# squish the Subject/Day info together with the simulated and then grab the quantiles
# for each day
PredIntervals <- cbind(PredData, Simulated) %>%
  gather('sim','Reaction', sim_1:sim_1000 ) %>%   # go from wide to long structure
  group_by(Subject, Days) %>%
  summarize(lwr = quantile(Reaction, probs = 0.025),
            upr = quantile(Reaction, probs = 0.975))
```

```
# Plot the prediction and confidence intervals
ggplot(ConfData,  aes(x=Days)) +
  geom_line(aes(y=Estimate), color='red') +
  geom_ribbon(aes(ymin=lwr, ymax= upr), fill='salmon', alpha=0.2) +
  geom_ribbon(data=PredIntervals, aes(ymin=lwr, ymax=upr), fill='blue', alpha=0.2)
```



11.8 Exercises

1. An experiment was conducted to determine the effect of recipe and baking temperature on chocolate cake quality. For each recipe, 15 batches of cake mix for were prepared (so 45 batches total). Each batch was sufficient for six cakes. Each of the six cakes was baked at a different temperature which was randomly assigned. Several measures of cake quality were recorded of which breaking angle was just one. The dataset is available in the `faraway` package as `choccake`.

    a. For the variables Temperature, Recipe, and Batch, which should be fixed and which should be random?
    b. Inspect the data. How many levels of batch are there and how will that influence your model statements in R?
    c. Build a mixed model using the main effects (no interactions).
    d. Compare your model in part (c) one models with one or both of the fixed effects removed. Which model is preferred?
    e. Compare your model in part (c) with a more complicated model that includes the interaction between temperature and recipe. Which model is preferred?
    f. Using the model you selected, discuss the impact of the different variance components.

2. An experiment was conducted to select the supplier of raw materials for production of a component. The breaking strength of the component was the objective of interest. Raw materials from four suppliers were considered. In our factory, we have four operators that can only produce one component per day. We utilized a Latin square design so that each factory operator worked with a different supplier each day. The data set is presented in the `faraway` package as `breaking`.

    a. Explain why it would be natural to treat the operators and days as random effects but the suppliers as fixed effects.
    b. Inspect the data? Does anything seem weird? It turns out that the person responsible for entering the data made an input error. Fix it making sure to preserve that each day has all 4 suppliers and 4 operators.
    c. Build a model to predict the breaking strength. Describe the variation from operator to operator and from day to day.
    d. Test the significance of the supplier effect.
    e. Is there a significant difference between the operators?

3. An experiment was performed to investigate the effect of ingestion of thyroxine or thiouracil. The

researchers took 27 newborn rats and divided them into three groups. The control group is ten rats that receive no addition to their drinking water. A second group of seven rats has thyroxine added to their drinking water and the final set ten rats have thiouracil added to their water. For each of five weeks, we take a body weight measurement to monitor the rats' growth. The data are available in the `faraway` package as `ratdrink`. *I suspect that we had 30 rats to begin with and somehow three rats in the thyroxine group had some issue unrelated to the treatment.* The following R code might be helpful for the initial visualization.

```
# we need to force ggplot to only draw lines between points for the same
# rat.  If I haven't already defined some aesthetic that is different
# for each rat, then it will connect points at the same week but for different
# rats. The solution is to add an aesthetic that does the equivalent of the
# dplyr function group_by(). In ggplot2, this aesetheic is "group".
ggplot(ratdrink, aes(y=wt, x=weeks, color=treat)) +
  geom_point(aes(shape=treat)) +
  geom_line(aes(group=subject))  # play with removing the group=subject aesthetic...
```

a. Consider the model with an interaction between Treatment and Week along with a random effect for each subject rat. Does the model with a random offset in the y-intercept perform as well as the model with random offsets in both the y-intercept and slope?
b. Next consider if you can simplify the model by removing the interaction between Treatment and Week and possibly even the Treatment main effect.

c. Comment on the effect of each treatment.

Chapter 12

Binomial Regression

```
library(tidyverse)   # dplyr, tidyr, ggplot2, etc...
library(emmeans)
library(pROC)
library(lmerTest)
```

The general linear model assumes that the observed data is distributed

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad \text{where } \epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$$

which can be re-written as

$$\boldsymbol{y} \sim N\left(\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I}\right)$$

and notably this assumes that the data are independent. This model has $E[\boldsymbol{y}] = \boldsymbol{X}\boldsymbol{\beta}$. This model is quite flexible and includes:

Model	Predictor Type	Response
Simple Linear Regression	1 Continuous	Continuous Normal Response
1-way ANOVA	1 Categorical	Continuous Normal Response
2-way ANOVA	2 Categorical	Continuous Normal Response
ANCOVA	1 Continuous, 1 Categorical	Continuous Normal Response

The general linear model expanded on the linear model and we allow the data points to be correlated

$$\boldsymbol{y} \sim N\left(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{\Omega}\right)$$

where we assume that $\boldsymbol{\Omega}$ has some known form but may include some unknown correlation parameters. This type of model includes our work with mixed models and time series data.

The study of generalized linear models removes the assumption that the error terms are normally distributed and allows the data to be distributed according to some other distribution such as Binomial, Poisson, or Exponential. These distributions are parameterized differently than the normal (instead of μ and σ, we might be interested in λ or p). However, I am still interested in how my covariates can be used to estimate my parameter of interest.

Critically, I still want to parameterize my covariates as $\boldsymbol{X}\boldsymbol{\beta}$ because we understand the how continuous and discrete covariates added and interpreted and what interactions between them mean. By keeping the $\boldsymbol{X}\boldsymbol{\beta}$ part, we continue to build on the earlier foundations.

12.1 Binomial Regression Model

To remove a layer of abstraction, we will now consider the case of binary regression. In this model, the observations (which we denote by w_i) are zeros and ones which correspond to some binary observation, perhaps presence/absence of an animal in a plot, or the success or failure of an viral infection. Recall that we could model this as $W_i \sim Bernoulli\,(p_i)$ random variable.

$$P\,(W_i = 1) = p_i$$

$$P\,(W_i = 0) = (1 - p_i)$$

which I can rewrite more formally letting w_i be the observed value as

$$P\,(W_i = w_i) = p_i^{w_i}\,(1 - p_i)^{1-w_i}$$

and the parameter that I wish to estimate and understand is the probability of a success p_i and usually I wish to know how my covariate data $\boldsymbol{X\beta}$ informs these probabilities.

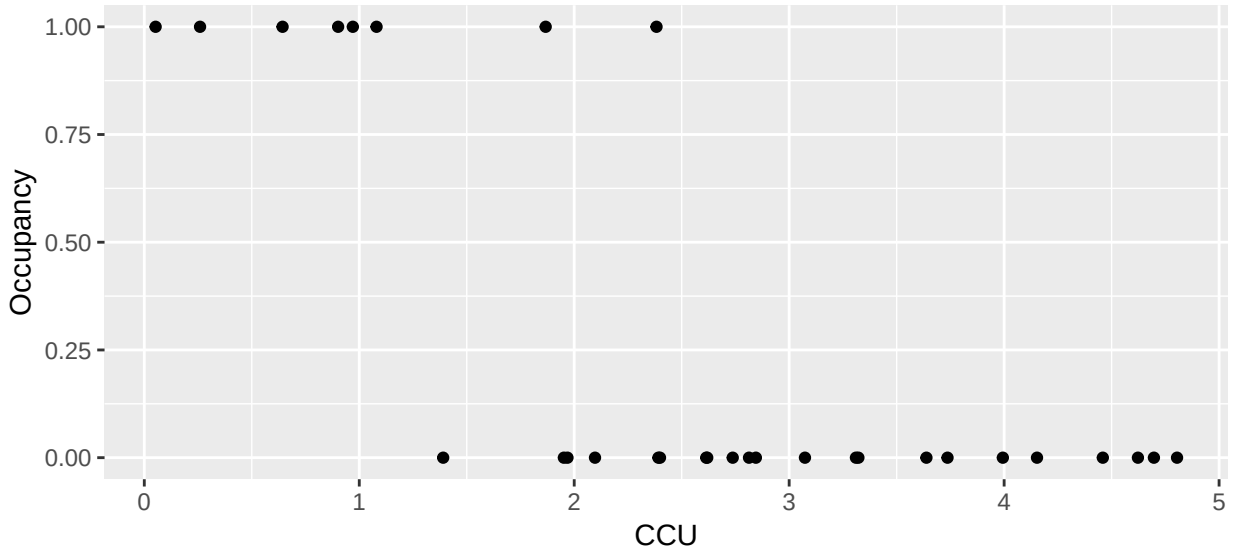
In the normal distribution case, we estimated the expected value of my response vector ($\boldsymbol{\mu}$) simply using $\hat{\boldsymbol{\mu}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$ but this will not work for an estimate of $\hat{\boldsymbol{p}}$ because there is no constraint on $\boldsymbol{X}\hat{\boldsymbol{\beta}}$, there is nothing to prevent it from being negative or greater than 1. Because we require the probability of success to be a number between 0 and 1, I have a problem.

Example: Suppose we are interested in the abundance of mayflies in a stream. Because mayflies are sensitive to metal pollution, I might be interested in looking at the presence/absence of mayflies in a stream relative to a pollution gradient. Here the pollution gradient is measured in Cumulative Criterion Units (CCU: CCU is defined as the ratio of the measured metal concentration to the hardness adjusted chronic criterion concentration, and then summed across each metal) where larger values imply more metal pollution.

```
data('Mayflies', package='dsData')
head(Mayflies)
```

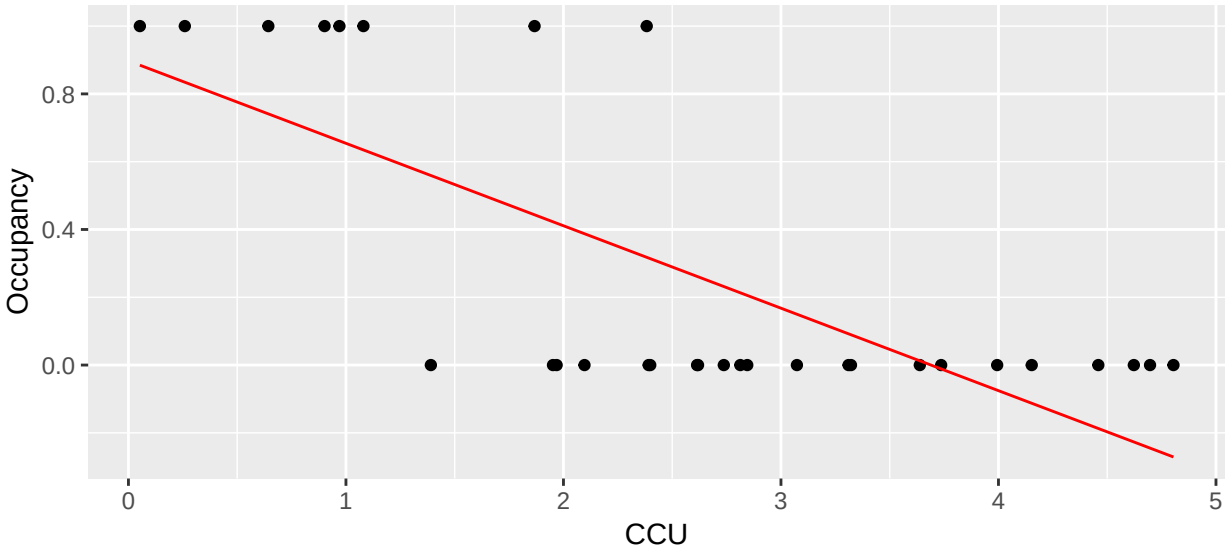
```
##          CCU Occupancy
## 1 0.05261076         1
## 2 0.25935617         1
## 3 0.64322010         1
## 4 0.90168941         1
## 5 0.97002630         1
## 6 1.08037011         1
```

```
ggplot(Mayflies, aes(x=CCU, y=Occupancy)) + geom_point()
```

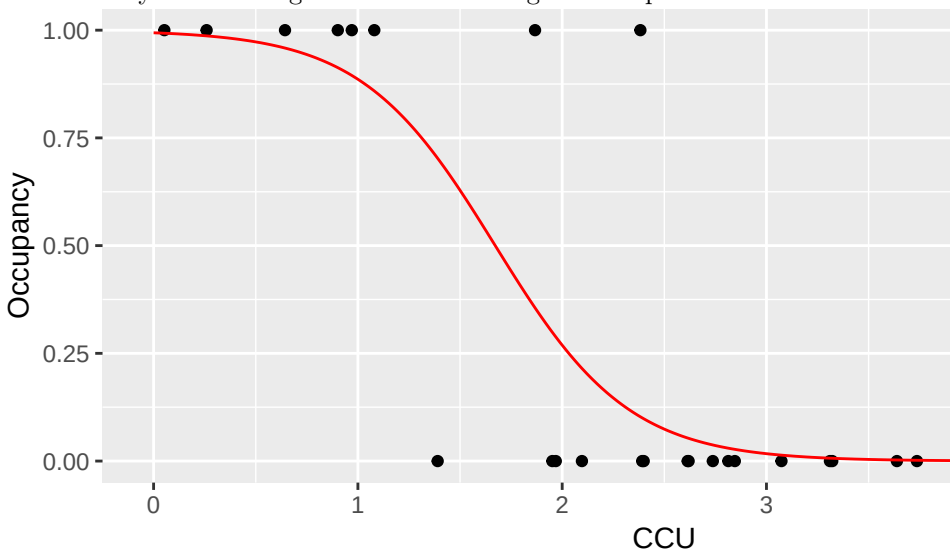


If I just fit a regular linear model to this data, we fit the following:

```
m <- lm( Occupancy ~ CCU, data=Mayflies )
Mayflies <- Mayflies %>% mutate(yhat = predict(m))
ggplot(Mayflies, aes(x=CCU, y=Occupancy)) +
  geom_point() +
  geom_line(aes(y=yhat), color='red')
```



which is horrible. First, we want the regression line to be related to the probability of occurrence and it is giving me a negative value. Instead, we want it to slowly tail off and give me more of an sigmoid-shaped curve.



Perhaps something more like the following:

We need a way to convert our covariate data $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}$ from something that can take values from $-\infty$ to $+\infty$ to something that is constrained between 0 and 1 so that we can fit the model

$$w_i \sim Bernoulli\left(\underbrace{g^{-1}\left(\underbrace{y_i}_{\text{in } [-\infty,\infty]}\right)}_{\text{in } [0,1]}\right)$$

There are several options for the link function $g^{-1}(\cdot)$ that are commonly used. We use the notation $y_i = \boldsymbol{X}_{i,\cdot}\boldsymbol{\beta}$ is unconstrained and can be in $(-\infty, +\infty)$ while $p_i = g^{-1}(y_i)$ is constrained to $[0, 1]$. When convenient, we will drop the i subscript while keeping the domain restrictions.

1. Logit (log odds) transformation. The link function is

$$g(p) = \log\left[\underbrace{\frac{p}{1-p}}_{\text{odds}}\right] = y$$

   with inverse

$$g^{-1}(y) = \frac{1}{1+e^{-y}}$$

   and we think of $g(p)$ as the log odds function.

2. Probit transformation. The link function is $g(p) = \Phi^{-1}(\boldsymbol{p})$ where Φ is the standard normal cumulative distribution function and therefore $g^{-1}(\boldsymbol{X}\boldsymbol{\beta}) = \Phi(\boldsymbol{X}\boldsymbol{\beta})$.

3. Complementary log-log transformation: $g(p) = \log[-\log(1-\boldsymbol{p})]$.

All of these functions will give a sigmoid shape with higher probability as y increases and lower probability as it decreases. The logit and probit transformations have the nice property that if $y = 0$ then $g^{-1}(0) = \frac{1}{2}$.

Usually the difference in inferences made using these different curves is relatively small and we will usually use the logit transformation because its form lends itself to a nice interpretation of my $\boldsymbol{\beta}$ values. In these cases, a slope parameter in our model will be interpreted as "the change in log odds for every one unit change in the predictor."

Because we will be using the logit transformation so often, it is useful to make the following definitions:

$$\text{logit}(p) = \log\left[\frac{p}{1-p}\right]$$

$$\text{ilogit}(y) = \frac{1}{1+e^{-y}}$$

where $p \in [0, 1]$ and $y \in (-\infty, +\infty)$.

As in the mixed model case, there are no closed form solution for $\hat{\boldsymbol{\beta}}$ and instead we must rely on numerical solutions to find the maximum likelihood estimators for $\hat{\boldsymbol{\beta}}$. To do this, we must derive the likelihood function.

$$\begin{aligned} L(\boldsymbol{\beta}|\boldsymbol{w}) &= \prod_{i=1}^{n} L(\boldsymbol{\beta}|w_i) \\ &= \prod_{i=1}^{n} p_i^{wi}(1-p_i)^{1-w_i} \\ &= \prod_{i=1}^{n} (\text{ilogit}(X_i\beta))^{w_i}(1-\text{ilogit}(X_i\beta))^{1-w_i} \end{aligned}$$

and we recognize that

$$p_i = \text{ilogit}(\boldsymbol{X}\boldsymbol{\beta}) = 1/\left(1+e^{-\boldsymbol{X}\boldsymbol{\beta}}\right)$$

and we substituted that into the equation. Often times it is more numerically stable to maximize the log-likelihood rather than the pure likelihood function because using logs helps prevent machine under-flow issues when the values of the likelihood is *really* small, but we will ignore that here and just assume that the function the performs the maximization is well designed to consider such issues.

Often we have more than one response at a particular level of $\boldsymbol{X}$. Let n_i be the number of observations observed at the particular value of $\boldsymbol{X}$, and y_i be the proportion of successes at that value of $\boldsymbol{X}$. In that

case, w_i is not a Bernoulli random variable, but rather a binomial random variable. Note that the Bernoulli distribution is the special case of the binomial distribution with $n_i = 1$.

```
# beta are the parameters
# y is the response in terms of 0,1 for this mayfly example
# X is the design matrix that we use for our covariates.
logL <- function(beta, y, X){
  out <- dbinom( y, size=1, prob= faraway::ilogit(X%*%beta), log=TRUE ) %>% sum()
  return(out)
}
NeglogL <- function(beta, y, X){ return( -logL(beta, y, X)) }
```

```
optim( par=c(0,0),      # intial bad guesses
       fn = NeglogL,    # we want to minimize the -logL function, maximize logL
       y=Mayflies$Occupancy,  # parameters to pass to the logL function
       X = model.matrix( ~ 1 + CCU, data=Mayflies ) )  # ditto
```

```
## $par
## [1]  5.100892 -3.050336
##
## $value
## [1] 6.324365
##
## $counts
## function gradient
##       71       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

In general, we don't want to have to specify the likelihood function by hand. Instead we will specify the model using the `glm` function which accepts a model formula as well as the distribution family that the data comes from.

```
head(Mayflies)
```

```
##          CCU Occupancy      yhat
## 1 0.05261076         1 0.8846278
## 2 0.25935617         1 0.8343395
## 3 0.64322010         1 0.7409692
## 4 0.90168941         1 0.6780997
## 5 0.97002630         1 0.6614776
## 6 1.08037011         1 0.6346378
```

```
# The following are equivalent and a "success" is if the site is
# occupied. If you were to switch it so a success is that the site
# is not occupied, the signs on all the beta coefficients would switch.
# m1 <- glm( (Occupancy == 1) ~ CCU, data=Mayflies, family=binomial )  # problem with emmeans!
m1 <- glm( cbind(Occupancy, 1-Occupancy) ~ CCU, data=Mayflies, family=binomial ) #ok!
m1 <- glm( Occupancy ~ CCU, data=Mayflies, family=binomial )
```

For binomial response data, we need to know the number of successes and the number of failures at each level of our covariate. In this case it is quite simple because there is only one observation at each CCU level, so the number of successes is Occupancy and the number of failures is just 1-Occupancy. For binomial data,

`glm` expect the response to be a two-column matrix where the first column is the number successes and and the second column is the number of failures. The default choice of link function for binomial data is the logit link, but the probit can be easily chosen as well using `family=binomial(link=probit)` in the call to `glm()`. If you only give a single response vector, it is assumed that the second column is to be calculated as 1-first.column.

```
summary(m1)
```

```
##
## Call:
## glm(formula = Occupancy ~ CCU, family = binomial, data = Mayflies)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.55741  -0.31594  -0.06553   0.08653   2.13362
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    5.102      2.369   2.154   0.0313 *
## CCU           -3.051      1.211  -2.520   0.0117 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 34.795  on 29  degrees of freedom
## Residual deviance: 12.649  on 28  degrees of freedom
## AIC: 16.649
##
## Number of Fisher Scoring iterations: 7
```

Notice that the summary table includes an estimate of the standard error of each $\hat{\beta}_j$ and a standardized value and z-test that are calculated in the usual manner $z_j = \frac{\hat{\beta}_j - 0}{StdErr(\hat{\beta}_j)}$ but these only approximately follow a standard normal distribution (due to the CLT results for Maximum Likelihood Estimators). We should regard the p-values given as approximate.

The sigmoid curve shown prior was the result of the logit model and we can estimate the probability of occupancy for any value of CCU. Surprisingly, R does not have a built-in function for the logit and ilogit function, but the faraway package does include them.

```
# Here are two ways to calculate the phat value for CCU = 1
new.df <- data.frame(CCU=1)
faraway::ilogit( predict(m1, newdata=new.df) )
```

```
##        1
## 0.886042
```

```
predict(m1, newdata=new.df, type='response')
```

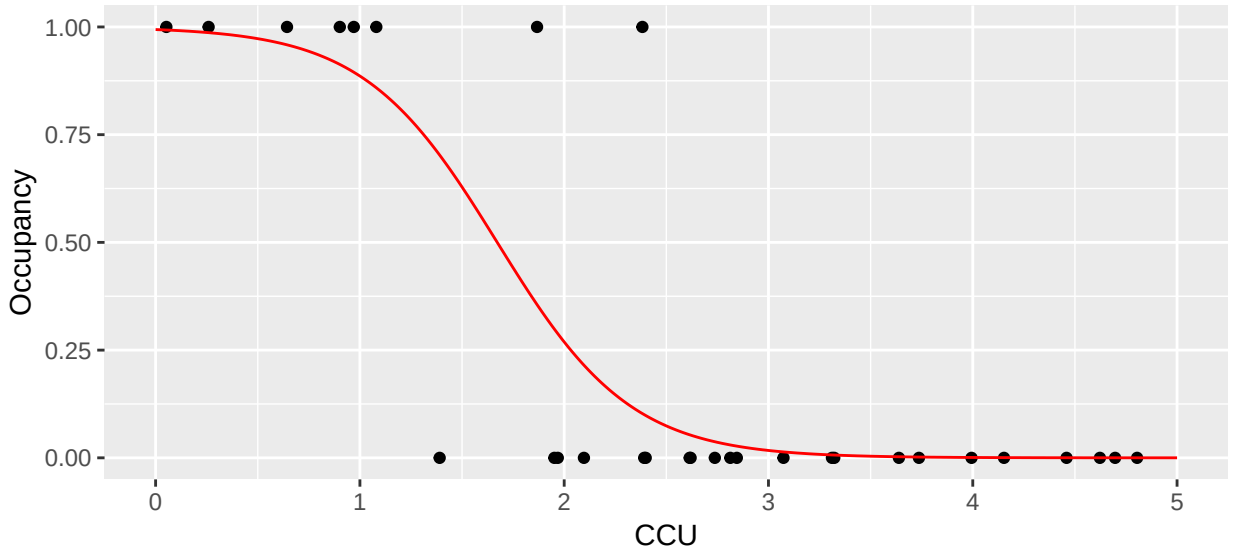
```
##        1
## 0.886042
```

```
# But the predict() function won't give you a confidence interval.
# Lets get this value through emmeans() instead.
emmeans(m1, ~CCU, type='response', at=list(CCU=1))
```

```
##  CCU     prob        SE  df asymp.LCL asymp.UCL
```

```
##    1 0.886042 0.1285373 Inf 0.3907622 0.9895016
##
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
```

```
# Finally a nice graph
new.df <- data.frame( CCU=seq(0,5, by=.01) )
yhat.df <- new.df %>% mutate(fit = predict(m1, newdata=new.df, type='response') )
ggplot(Mayflies, aes(x=CCU)) +
  geom_point(aes(y=Occupancy)) +
  geom_line( data=yhat.df, aes(y=fit), color='red')
```



Suppose that we were to give a predicted Presence/Absence class based on the $\hat{p}$ value. Lets predict presence if the probability is greater than 0.5 and absent if the the probability is less that 0.5.

```
Mayflies <- Mayflies %>%
  mutate(phat = predict(m1, type='response')) %>%
  mutate(chat = ifelse(phat >0.5, 1, 0))

# This is often called the "confusion matrix"
table(Truth = Mayflies$Occupancy, Prediction = Mayflies$chat)
```

```
##       Prediction
## Truth  0  1
##     0 21  1
##     1  2  6
```

This scheme has mis-classified 3 observations, two cases where mayflies were present but we predicted they would be absent, and one case where no mayflies were detected but we predicted we would see them.

12.2 Measures of Fit Quality

12.2.1 Deviance

In the normal linear models case, we were very interested in the Sum of Squared Error (SSE)

$$SSE = \sum_{i=1}^{n} (w_i - \hat{w}_i)^2$$

because it provided a mechanism for comparing the fit of two different models. If a model had a very small SSE, then it fit the observed data well. We used this as a basis for forming our F-test to compare nested models (some rescaling by the appropriate degrees of freedom was necessary, though).

We want an equivalent measure of goodness-of-fit for models that are non-normal, but in the normal case, I would like it to be related to my SSE statistic.

The deviance of a model with respect to some data $\boldsymbol{y}$ is defined by

$$D\left(\boldsymbol{w}, \hat{\boldsymbol{\theta}}_0\right) = -2\left[\log L\left(\hat{\boldsymbol{\theta}}_0 | \boldsymbol{w}\right) - \log L\left(\hat{\boldsymbol{\theta}}_S | \boldsymbol{w}\right)\right]$$

where $\hat{\boldsymbol{\theta}}_0$ are the fitted parameters of the model of interest, and $\hat{\boldsymbol{\theta}}_S$ are the fitted parameters under a "saturated" model that has as many parameters as it has observations and can therefore fit the data perfectly. Thus the deviance is a measure of deviation from a perfect model and is flexible enough to handle non-normal distributions appropriately.

Notice that this definition is very similar to what is calculated during the Likelihood Ratio Test. For any two models under consideration, the LRT can be formed by looking at the difference of the deviances of the two nested models

$$LRT = D\left(\boldsymbol{w}, \hat{\boldsymbol{\theta}}_{simple}\right) - D\left(\boldsymbol{w}, \hat{\boldsymbol{\theta}}_{complex}\right) \dot{\sim} \chi^2_{df_{complex} - df_{simple}}$$

```
m0 <- glm( Occupancy ~ 1, data=Mayflies, family=binomial )
anova(m0, m1)
```

```
## Analysis of Deviance Table
##
## Model 1: Occupancy ~ 1
## Model 2: Occupancy ~ CCU
##   Resid. Df Resid. Dev Df Deviance
## 1        29     34.795
## 2        28     12.649  1   22.146
```

```
1 - pchisq( 22.146, df=1 )
```

```
## [1] 2.526819e-06
```

A convenient way to get R to calculate the LRT χ^2 p-value for you is to specify the `test=LRT` inside the `anova` function.

```
anova(m1, test='LRT')
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Occupancy
##
```

```
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                    29     34.795
## CCU   1   22.146        28     12.649 2.527e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The inference of this can be confirmed by looking at the AIC values of the two models as well.

```
AIC(m0, m1)
```

```
##    df      AIC
## m0  1 36.79491
## m1  2 16.64873
```

12.2.2 Goodness of Fit

The deviance is a good way to measure if a model fits the data, but it is not the only method. Pearson's X^2 statistic is also applicable. This statistic takes the general form $X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$ where O_i is the number of observations observed in category i and E_i is the number expected in category i. In our case we need to figure out the categories we have. Since we have both the number of success and failures, we'll have two categories per observation i.

$$X^2 = \sum_{i=1}^{n} \left[\frac{(w_i - n_i\hat{p}_i)^2}{n_i\hat{p}_i} + \frac{((n_i - w_i) - n_i(1-\hat{p}_i))^2}{n_i(1-\hat{p}_i)} \right] = \sum_{i=1}^{n} \frac{(w_i - n_i\hat{p}_i)^2}{n_i\hat{p}_i(1-\hat{p}_i)}$$

and the Pearson residual can be defined as

$$r_i = \frac{w_i - n_i\hat{p}_i}{\sqrt{n_i\hat{p}_i(1-\hat{p}_i)}}$$

These can be found in R via the following commands

```
sum( residuals(m1, type='pearson')^2 )
```

```
## [1] 14.92367
```

Pearson's X^2 statistic is quite similar to the deviance statistic

```
deviance(m1)
```

```
## [1] 12.64873
```

12.3 Confidence Intervals

Confidence intervals for the regression could be constructed using normal approximations for the parameter estimates. An approximate $100(1-\alpha)\%$ confidence interval for β_i would be

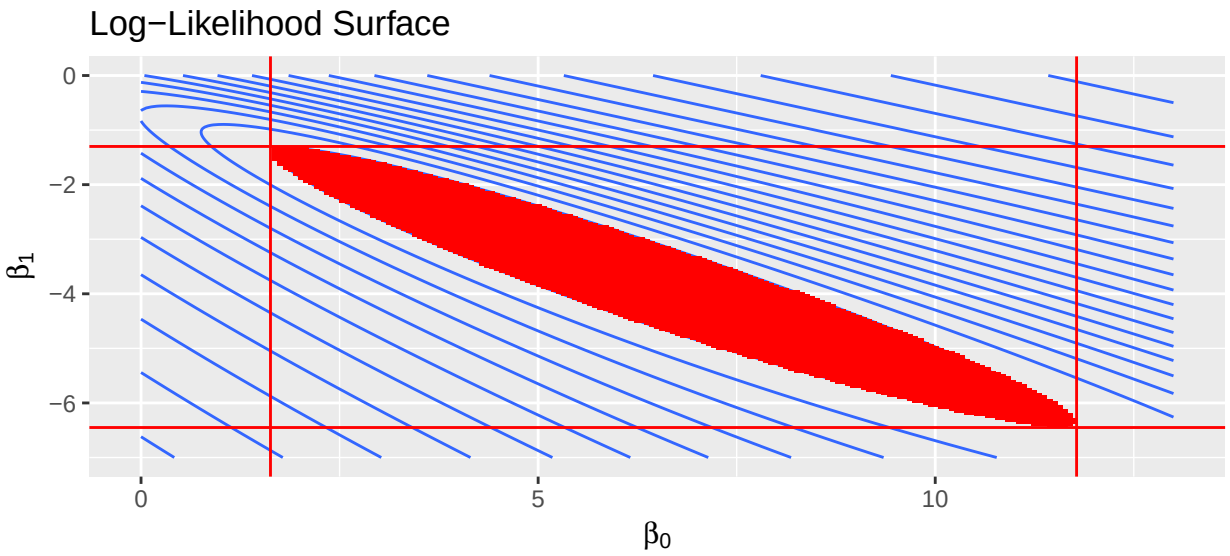
$$\hat{\beta}_i \pm z^{1-\alpha/2} \, StdErr\left(\hat{\beta}_i\right)$$

but we know that this is not a good approximation because the the normal approximation will not be good for small sample sizes and it isn't clear what is "big enough". Instead we will use an inverted LRT to develop confidence intervals for the β_i parameters.

We first consider the simplest case, where we have only an intercept and slope parameter. Below is a contour plot of the likelihood surface and the shaded region is the region of the parameter space where the parameters (β_0, β_1) would not be rejected by the LRT. This region is found by finding the maximum likelihood estimators $\hat{\beta}_0$ and $\hat{\beta}_1$, and then finding set of β_0, β_1 pairs such that

$$-2\left[\log L\left(\beta_0, \beta_1\right) - \log L\left(\hat{\beta}_0, \hat{\beta}_1\right)\right] \leq \chi^2_{df=1,0.95}$$

$$\log L\left(\beta_0, \beta_1\right) \geq \left(\frac{-1}{2}\right)\chi^2_{1,0.95} + \log L\left(\hat{\beta}_0, \hat{\beta}_1\right)$$



Looking at just the β_0 axis, this translates into a confidence interval of (1.63, 11.78). This method is commonly referred to as the "profile likelihood" interval because the interval is created by viewing the contour plot from the one axis. The physical analogy is to viewing a mountain range from afar and asking, "What parts of the mountain are higher than 8000 feet?"

This type of confidence interval is more robust than the normal approximation and should be used whenever practical. In R, the profile likelihood confidence interval for `glm` objects is available in the `MASS` library.

```
confint(m1) # using defaults
```

```
## Waiting for profiling to be done...
```

```
##                 2.5 %    97.5 %
## (Intercept)  1.629512 11.781167
## CCU         -6.446863 -1.304244
```

```
confint(m1, level=.95, parm='CCU') # Just the slope parameter
```

```
## Waiting for profiling to be done...
```

```
##     2.5 %    97.5 %
## -6.446863 -1.304244
```

12.4 Interpreting model coefficients

We first consider why we are dealing with odds $\frac{p}{1-p}$ instead of just p. They contain the same information, so the choice is somewhat arbitrary, however we've been using probabilities for so long that it feels unnatural to switch to odds. There are two good reasons for this, however.

The first is that the odds $\frac{p}{1-p}$ can take on any value from 0 to ∞ and so part of our translation of p to an unrestricted domain is already done.

The second is that it is easier to compare odds than to compare probabilities. For example, (as of this writing) I have a three month old baby who is prone to spitting up her milk.

- I think the probability that she will not spit up on me today is $p_1 = 0.10$. My wife disagrees and believes the probability is $p_2 = 0.01$. We can look at those probabilities and recognize that we differ in our assessment by a factor of 10 because $10 = p_1/p_2$. If we had assessed the chance of her spitting up using odds, I would have calculated $o_1 = 0.1/0.9 = 1/9$. My wife, on the other hand, would have calculated $o_2 = .01/.99 = 1/99$. The odds ratio of these is $[1/9]\,/\,[1/99] = 99/9 = 11$. This shows that she is much more certain that the event will not happen and the multiplying factor of the pair of odds is 11.
- But what if we were to consider the probability that my daughter will spit up? The probabilities assigned by me versus my wife are $p_1 = 0.9$ and $p_2 = 0.99$. How should I assess that our probabilities differ by a factor of 10, because $p_1/p_2 = 0.91 \neq 10$? The odds ratio remains the same calculation, however. The odds I would give are $o_1 = .9/.1 = 9$ vs my wife's odds $o_2 = .99/.01 = 99$. The odds ratio is now $9/99 = 1/11$ and gives the same information as I calculated from the where we defined a success as my daughter not spitting up.

To try to clear up the verbiage we'll consider a few different cases:

Probability	Odds	Verbiage
$p = .95$	$\frac{95}{5} = \frac{19}{1} = 19$	19 to 1 odds for
$p = .75$	$\frac{75}{25} = \frac{3}{1} = 3$	3 to 1 odds for
$p = .50$	$\frac{50}{50} = \frac{1}{1} = 1$	1 to 1 odds
$p = .25$	$\frac{25}{75} = \frac{1}{3} = 0.33$	3 to 1 odds against
$p = .05$	$\frac{95}{5} = \frac{1}{19} = 0.0526$	19 to 1 odds against

Given a logistic regression model with two continuous covariates, then using the `logit()` link function we have

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\frac{p}{1-p} = e^{\beta_0} e^{\beta_1 x_1} e^{\beta_2 x_2}$$

and we can interpret β_1 and β_2 as the increase in the log odds for every unit increase in x_1 and x_2. We could alternatively interpret β_1 and β_2 using the notion that a one unit change in x_1 as a percent change of e^{β_1} in the odds. That is to say, e^{β_1} is the odds ratio of that change.

To investigate how to interpret these effects, we will consider an example of the rates of respiratory disease of babies in the first year based on covariates of gender and feeding method (breast milk, formula from a bottle, or a combination of the two). The data percentages of babies suffering respiratory disease are

	Formula `f`	Breast Milk `b`	Breast Milk + Supplement `s`
Males `M`	$\frac{77}{485}$	$\frac{47}{494}$	$\frac{19}{147}$
Females `F`	$\frac{48}{384}$	$\frac{31}{464}$	$\frac{16}{127}$

We can fit the saturated model (6 parameters to fit 6 different probabilities) as

```
data('babyfood', package='faraway')
head(babyfood)
```

```
##   disease nondisease  sex   food
```

```
## 1      77        381  Boy Bottle
## 2      19        128  Boy  Suppl
## 3      47        447  Boy Breast
## 4      48        336 Girl Bottle
## 5      16        111 Girl  Suppl
## 6      31        433 Girl Breast
```

```
m2 <- glm( cbind(disease,nondisease) ~ sex * food, family=binomial, data=babyfood )
summary(m2)
```

```
##
## Call:
## glm(formula = cbind(disease, nondisease) ~ sex * food, family = binomial,
##     data = babyfood)
##
## Deviance Residuals:
## [1]  0  0  0  0  0  0
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.59899    0.12495 -12.797  < 2e-16 ***
## sexGirl            -0.34692    0.19855  -1.747 0.080591 .
## foodBreast         -0.65342    0.19780  -3.303 0.000955 ***
## foodSuppl          -0.30860    0.27578  -1.119 0.263145
## sexGirl:foodBreast -0.03742    0.31225  -0.120 0.904603
## sexGirl:foodSuppl   0.31757    0.41397   0.767 0.443012
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.6375e+01  on 5  degrees of freedom
## Residual deviance: 2.6401e-13  on 0  degrees of freedom
## AIC: 43.518
##
## Number of Fisher Scoring iterations: 3
```

Notice that the residual deviance is effectively zero with zero degrees of freedom indicating we just fit the saturated model.

It is nice to look at the single term deletions to see if the interaction term could be dropped from the model.

```
anova(m2, test='LRT')
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: cbind(disease, nondisease)
##
## Terms added sequentially (first to last)
##
##
##          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                         5    26.3753
## sex       1   5.4761         4    20.8992   0.01928 *
```

```
## food      2  20.1772         2     0.7219 4.155e-05 ***
## sex:food  2   0.7219         0     0.0000   0.69701
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Given this, we will look use the reduced model with out the interaction and check if we could reduce the model any more.

```
m1 <- glm( cbind(disease, nondisease) ~ sex + food, family=binomial, data=babyfood)
drop1(m1, test='Chi')  # all single term deletions, using LRT
```

```
## Single term deletions
##
## Model:
## cbind(disease, nondisease) ~ sex + food
##        Df Deviance    AIC     LRT  Pr(>Chi)
## <none>      0.7219 40.240
## sex     1   5.6990 43.217  4.9771   0.02569 *
## food    2  20.8992 56.417 20.1772 4.155e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From this we see that we cannot reduce the model any more and we will interpret the coefficients of this model.

```
coef(m1, digits=5)  # more accuracy
```

```
## (Intercept)     sexGirl  foodBreast   foodSuppl
##  -1.6127038  -0.3125528  -0.6692946  -0.1725424
```

We interpret the intercept term as the log odds that a male child fed only formula will develop a respiratory disease in their first year. With that, we could then calculate what the probability of a male formula fed baby developing respiratory disease using following

$$-1.6127 = \log\left(\frac{p_{M,f}}{1-p_{M,f}}\right) = \text{logit}\left(p_{M,f}\right)$$

thus

$$p_{M,f} = \text{ilogit}\left(-1.6127\right) = \frac{1}{1+e^{1.6127}} = 0.1662$$

We notice that the odds of respiratory disease disease is

$$\frac{p_{M,f}}{1-p_{M,f}} = \frac{0.1662}{1-0.1662} = 0.1993 = e^{-1.613}$$

For a female child bottle fed only formula, their probability of developing respiratory disease is

$$p_{F,f} = \frac{1}{1+e^{-(-1.6127-0.3126)}} = \frac{1}{1+e^{1.9253}} = 0.1273$$

and the associated odds are

$$\frac{p_{F,f}}{1-p_{F,f}} = \frac{0.1273}{1-0.1273} = 0.1458 = e^{-1.6127-0.3126}$$

so we can interpret $e^{-0.3126} = 0.7315$ as the percent change in odds from male to female infants. That is to say, it is the *odds ratio* of the female infants to the males is

$$e^{-0.3126} = \frac{\left(\frac{p_{F,f}}{1-p_{F,f}}\right)}{\left(\frac{p_{M,f}}{1-p_{M,f}}\right)} = \frac{0.1458}{0.1993} = 0.7315$$

The interpretation here is that odds of respiratory infection for females is 73.1% than that of a similarly feed male child and I might say that being female reduces the odds of respiratory illness by 27% compared to male babies. Similarly we can calculate the change in odds ratio for the feeding types:

```
exp( coef(m1) )
```

```
## (Intercept)     sexGirl  foodBreast   foodSuppl
##   0.1993479   0.7315770   0.5120696   0.8415226
```

First we notice that the intercept term can be interpreted as the odds of infection for the reference group. The each of the offset terms are the odds ratios compared to the reference group. We see that breast milk along with formula has only 84% of the odds of respiratory disease as a formula only baby, and a breast milk fed child only has 51% of the odds for respiratory disease as the formula fed baby. We can look at confidence intervals for the odds ratios by the following:

```
exp( confint(m1) )
```

```
## Waiting for profiling to be done...
```

```
##                 2.5 %    97.5 %
## (Intercept) 0.1591988 0.2474333
## sexGirl     0.5536209 0.9629225
## foodBreast  0.3781905 0.6895181
## foodSuppl   0.5555372 1.2464312
```

We should be careful in drawing conclusions here because this study was a retrospective study and the decision to breast feed a baby vs feeding with formula is inextricably tied to socio-economic status and we should investigate if the effect measured is due to feeding method or some other lurking variable tied to socio-economic status.

As usual, we don't want to calculate all these quantities by hand and would prefer if `emmeans` would do all the back-transformations for us.

```
emmeans(m1, ~ sex+food, type='response')  # probabilities for each sex/treatment
```

```
##  sex  food         prob          SE  df  asymp.LCL  asymp.UCL
##  Boy  Bottle 0.16621356 0.015578525 Inf 0.13787849 0.19902740
##  Girl Bottle 0.12727653 0.014284680 Inf 0.10180659 0.15799766
##  Boy  Breast 0.09262485 0.011112582 Inf 0.07302350 0.11682462
##  Girl Breast 0.06948992 0.009299798 Inf 0.05333055 0.09007973
##  Boy  Suppl  0.14365654 0.023380926 Inf 0.10360878 0.19580264
##  Girl Suppl  0.10931093 0.019437946 Inf 0.07662600 0.15361824
##
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
```

```
emmeans(m1, pairwise~ sex, type='response') # compare Boy / Girl
```

```
## $emmeans
##  sex        prob         SE  df asymp.LCL asymp.UCL
##  Boy  0.13086682 0.01113260 Inf 0.1105484  0.154272
##  Girl 0.09922469 0.01026642 Inf 0.0808393  0.121240
##
## Results are averaged over the levels of: food
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
##
## $contrasts
##  contrast   odds.ratio        SE  df z.ratio p.value
```

```
##  Boy / Girl    1.36691 0.1927885 Inf   2.216  0.0267
##
## Results are averaged over the levels of: food
## Tests are performed on the log odds ratio scale
```

```
emmeans(m1, revpairwise~ sex, type='response') # compare Girl / Boy
```

```
## $emmeans
##  sex        prob         SE  df asymp.LCL asymp.UCL
##  Boy  0.13086682 0.01113260 Inf 0.1105484  0.154272
##  Girl 0.09922469 0.01026642 Inf 0.0808393  0.121240
##
## Results are averaged over the levels of: food
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
##
## $contrasts
##  contrast   odds.ratio        SE  df z.ratio p.value
##  Girl / Boy   0.731577 0.1031814 Inf  -2.216  0.0267
##
## Results are averaged over the levels of: food
## Tests are performed on the log odds ratio scale
```

```
emmeans(m1, pairwise ~ food, type='response') # compare Foods
```

```
## $emmeans
##  food         prob          SE  df  asymp.LCL  asymp.UCL
##  Bottle 0.14566920 0.012202488 Inf 0.12334180 0.17124878
##  Breast 0.08030023 0.008772806 Inf 0.06470057 0.09926188
##  Suppl  0.12548068 0.019939664 Inf 0.09131680 0.17003440
##
## Results are averaged over the levels of: sex
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
##
## $contrasts
##  contrast        odds.ratio        SE  df z.ratio p.value
##  Bottle / Breast  1.9528594 0.2987986 Inf   4.374  <.0001
##  Bottle / Suppl   1.1883222 0.2443009 Inf   0.839  0.6786
##  Breast / Suppl   0.6085037 0.1316781 Inf  -2.296  0.0564
##
## Results are averaged over the levels of: sex
## P value adjustment: tukey method for comparing a family of 3 estimates
## Tests are performed on the log odds ratio scale
```

```
emmeans(m1, revpairwise ~ food, type='response') # compare Foods
```

```
## $emmeans
##  food         prob          SE  df  asymp.LCL  asymp.UCL
##  Bottle 0.14566920 0.012202488 Inf 0.12334180 0.17124878
##  Breast 0.08030023 0.008772806 Inf 0.06470057 0.09926188
##  Suppl  0.12548068 0.019939664 Inf 0.09131680 0.17003440
##
## Results are averaged over the levels of: sex
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
```

```
##
## $contrasts
##  contrast        odds.ratio         SE  df z.ratio p.value
##  Breast / Bottle  0.5120696 0.07834958 Inf  -4.374  <.0001
##  Suppl / Bottle   0.8415226 0.17300418 Inf  -0.839  0.6786
##  Suppl / Breast   1.6433753 0.35562062 Inf   2.296  0.0564
##
## Results are averaged over the levels of: sex
## P value adjustment: tukey method for comparing a family of 3 estimates
## Tests are performed on the log odds ratio scale
```

12.5 Prediction and Effective Dose Levels

To demonstrate the ideas in this section, we'll use a toxicology study that examined insect mortality as a function of increasing concentrations of an insecticide.

```
data('bliss', package='faraway')
```

We first fit the logistic regression model and plot the results

```
m1 <- glm( cbind(alive, dead) ~ conc, family=binomial, data=bliss)
```

Given this, we want to develop a confidence interval for the probabilities by first calculating using the following formula. As usual, we recall that the y values live in $(-\infty, \infty)$.

$$CI_y: \quad \hat{y} \pm z^{1-\alpha/2}\, StdErr\,(\hat{y})$$

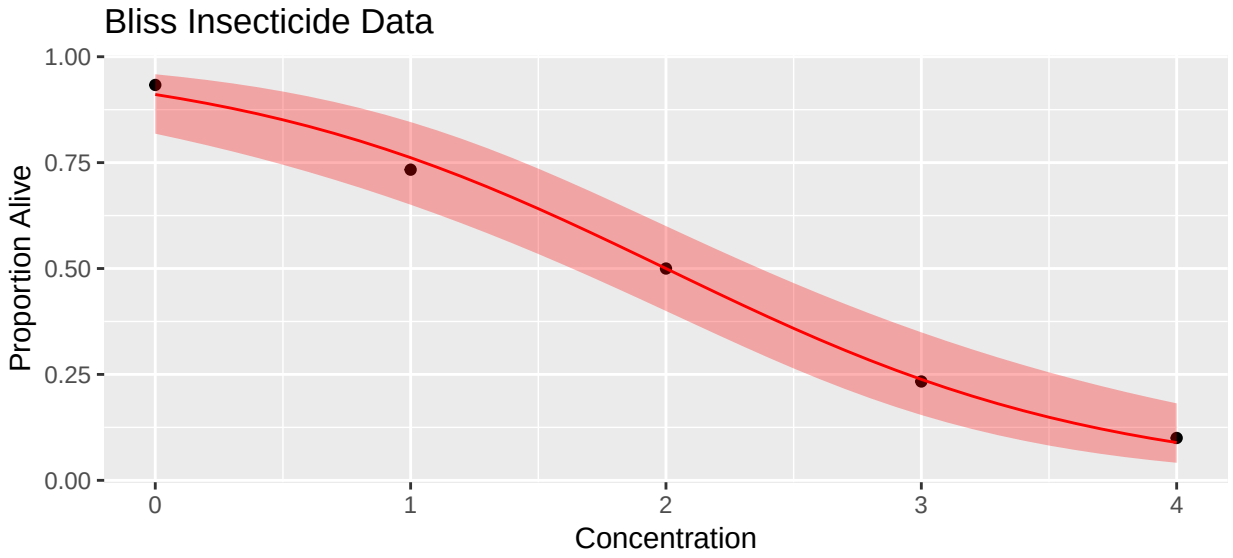
We must then convert this to the $[0, 1]$ space using the ilogit() function.

$$CI_p = \text{ilogit}\,(CI_y)$$

```
probs <- data.frame(conc=seq(0,4,by=.1))
yhat <- predict(m1, newdata=probs, se.fit=TRUE)  # list with two elements fit and se.fit
yhat <- data.frame( fit=yhat$fit, se.fit = yhat$se.fit)
probs <- cbind(probs, yhat)
head(probs)
```

```
##   conc      fit    se.fit
## 1  0.0 2.323790 0.4178878
## 2  0.1 2.207600 0.4022371
## 3  0.2 2.091411 0.3868040
## 4  0.3 1.975221 0.3716158
## 5  0.4 1.859032 0.3567036
## 6  0.5 1.742842 0.3421035
```

```
probs <- probs %>% mutate(
  phat  = faraway::ilogit(fit),
  lwr   = faraway::ilogit( fit - 1.96 * se.fit ),
  upr   = faraway::ilogit( fit + 1.96 * se.fit ))
ggplot(bliss, aes(x=conc)) +
  geom_point(aes(y=alive/(alive+dead))) +
  geom_line(data=probs, aes(y=phat), color='red') +
  geom_ribbon(data=probs, aes(ymin=lwr, ymax=upr), fill='red', alpha=.3) +
  ggtitle('Bliss Insecticide Data') +
  xlab('Concentration') + ylab('Proportion Alive')
```



Alternatively, we might want to do this calculation via `emmeans`.

```
emmeans::emmeans(m1, ~conc, at=list(conc=c(0:4)), type='response')
```

```
##  conc       prob         SE  df  asymp.LCL asymp.UCL
##     0 0.91082823 0.03394092 Inf 0.81828107 0.9586255
##     1 0.76167686 0.05000854 Inf 0.65066027 0.8457758
##     2 0.50000000 0.05183118 Inf 0.39978789 0.6002121
##     3 0.23832314 0.05000854 Inf 0.15422418 0.3493397
##     4 0.08917177 0.03394092 Inf 0.04137453 0.1817189
##
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
```

The next thing we want to do is come up with a confidence intervals for the concentration level that results in the death of $100(p)\%$ of the insects. Often we are interested in the case of $p = 0.5$. This is often called LD50, which is the lethal dose for 50% of the population. Using the link function you can set the p value and solve for the concentration value to find

$$\hat{x}_p = \frac{\text{logit}(p) - \hat{\beta}_0}{\hat{\beta}_1}$$

which gives us a point estimate of LD(p). To get a confidence interval we need to find the standard error of $\hat{x}_p$. Since this is a non-linear function of $\hat{\beta}_0$ and $\hat{\beta}_1$ which are correlated, we must be careful in the calculation. The actual calculation is done using the Delta Method Approximation:

$$Var\left(g\left(\hat{\boldsymbol{\theta}}\right)\right) = g'(\boldsymbol{\theta})^T \, Var(\boldsymbol{\theta}) \, g'(\boldsymbol{\theta})$$

Fortunately we don't have to do these calculations by hand and can use the `dose.p()` function in the `MASS` package.

```
LD <- MASS::dose.p(m1, p=c(.25, .5, .75))
LD
```

```
##                Dose        SE
## p = 0.25: 2.945535 0.2315932
## p = 0.50: 2.000000 0.1784367
## p = 0.75: 1.054465 0.2315932
```

and we can use these to create approximately confidence intervals for these $\hat{x}_p$ values via

$$\hat{x}_p \pm z^{1-\alpha/2} \, StdErr\left(\hat{x}_p\right)$$

```
# why did the MASS authors make LD a vector of the
# estimated values and have an additional attribute
# that contains the standard errors?  Whatever, lets
# turn this into a convential data.frame.
str(LD)
```

```
##  'glm.dose' Named num [1:3] 2.95 2 1.05
##  - attr(*, "names")= chr [1:3] "p = 0.25:" "p = 0.50:" "p = 0.75:"
##  - attr(*, "SE")= num [1:3, 1] 0.232 0.178 0.232
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "p = 0.25:" "p = 0.50:" "p = 0.75:"
##   .. ..$ : NULL
##  - attr(*, "p")= num [1:3] 0.25 0.5 0.75
```

```
CI <- data.frame(p = attr(LD,'p'),
                 Dose = as.vector(LD),
                 SE   = attr(LD,'SE')) %>%  # save the output table as LD
  mutate( lwr = Dose - qnorm(.975)*SE,
          upr = Dose + qnorm(.975)*SE )
CI
```

```
##      p     Dose        SE       lwr      upr
## 1 0.25 2.945535 0.2315932 2.4916207 3.399449
## 2 0.50 2.000000 0.1784367 1.6502705 2.349730
## 3 0.75 1.054465 0.2315932 0.6005508 1.508379
```

12.6 Overdispersion

In the binomial distribution, the variance is a function of the probability of success and is

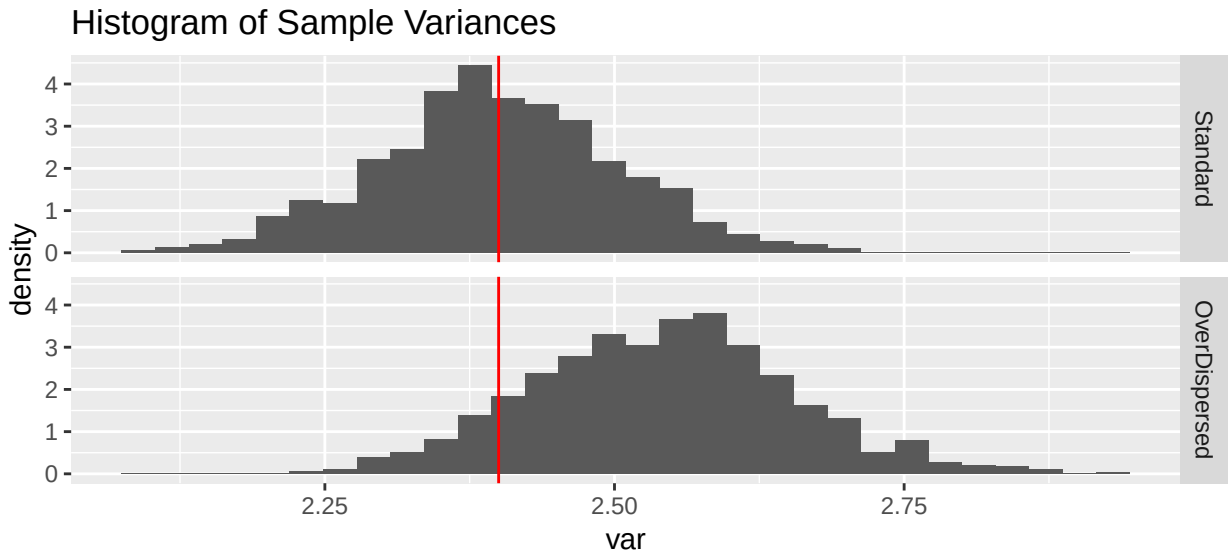
$$Var\left(W\right) = np\left(1-p\right)$$

but there are many cases where we might be interested in adding an additional variance parameter ϕ to the model. A common reason for overdispersion to appear is that we might not have captured all the covariates that influence p.

We can do a quick simulation to demonstrate that additional variability in p leads to addition variability overall.

```
N <- 1000
n <- 10
p <- .6
overdispersed_p <- p + rnorm(n, mean=0, sd=.05)
sim.data <- NULL
for( i in 1:N ){
  sim.data <- sim.data %>% rbind(data.frame(
    var = var( rbinom(N, size=n, prob=p)),
    type = 'Standard'))
  sim.data <- sim.data %>% rbind(data.frame(
    var = var( rbinom(N, size=n, prob=overdispersed_p )),
    type = 'OverDispersed'))
```

```
}
true.var <- p*(1-p)*n
ggplot(sim.data, aes(x=var, y=..density..)) +
  geom_histogram(bins=30) +
  geom_vline(xintercept = true.var, color='red') +
  facet_grid(type~.) +
  ggtitle('Histogram of Sample Variances')
```



We see that the sample variances fall neatly about the true variance of 2.4 in the case where the data is distributed with a constant value for p. However adding a small amount of random noise about the parameter p, and we'd have more variance in the samples.

The extra uncertainty of the probability of success results in extra variability in the responses.

We can recognize when overdispersion is present by examining the deviance of our model. Because the deviance is approximately distributed

$$D\left(\boldsymbol{y}, \boldsymbol{\theta}\right) \dot{\sim} \chi^2_{df}$$

where df is the residual degrees of freedom in the model. Because the χ^2_k is the sum of k independent, squared standard normal random variables, it has an expectation k and variance $2k$. For binomial data with group sizes (say larger than 5), this approximation isn't too bad and we can detect overdispersion. For binary responses, the approximation is quite poor and we cannot detect overdispersion.

The simplest approach for modeling overdispersion is to introduce an addition dispersion parameter σ^2. This dispersion parameter may be estimated using

$$\hat{\sigma}^2 = \frac{X^2}{n-p}.$$

With the addition of the overdispersion parameter to the model, the differences between a simple and complex model is no longer distributed χ^2 and we must use the following approximate F-statistic

$$F = \frac{\left(D_{simple} - D_{complex}\right) / \left(df_{small} - df_{large}\right)}{\hat{\sigma}^2}$$

Using the F-test when the the overdispersion parameter is 1 is a less powerful test than the χ^2 test, so we'll only use the F-test when the overdispersion parameter must be estimated.

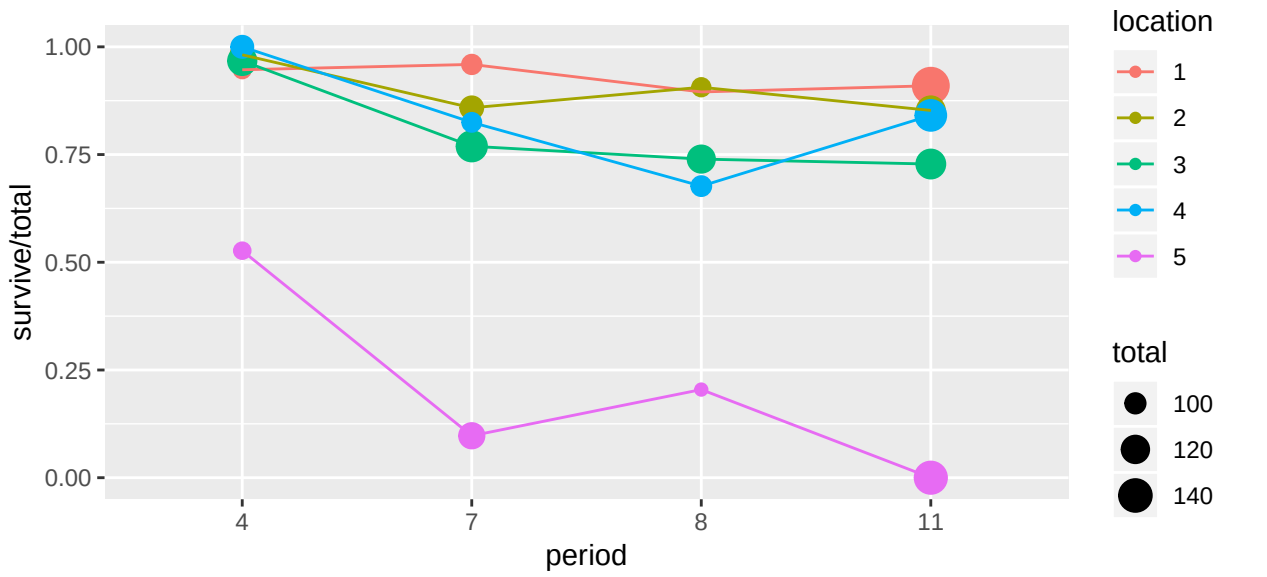
Example: We consider an experiment where at five different stream locations, four boxes of trout eggs were buried and retrieved at four different times after the original placement. The number of surviving eggs was recorded and the eggs disposed of.

```
data(troutegg, package='faraway')
troutegg <- troutegg %>%
  mutate( perish = total - survive) %>%
  dplyr::select(location, period, survive, perish, total) %>%
  arrange(location, period)

troutegg %>% arrange(location, period)
```

```
##    location period survive perish total
## 1         1      4      89      5    94
## 2         1      7      94      4    98
## 3         1      8      77      9    86
## 4         1     11     141     14   155
## 5         2      4     106      2   108
## 6         2      7      91     15   106
## 7         2      8      87      9    96
## 8         2     11     104     18   122
## 9         3      4     119      4   123
## 10        3      7     100     30   130
## 11        3      8      88     31   119
## 12        3     11      91     34   125
## 13        4      4     104      0   104
## 14        4      7      80     17    97
## 15        4      8      67     32    99
## 16        4     11     111     21   132
## 17        5      4      49     44    93
## 18        5      7      11    102   113
## 19        5      8      18     70    88
## 20        5     11       0    138   138
```

We can first visualize the data

```
ggplot(troutegg, aes(x=period, y=survive/total, color=location)) +
  geom_point(aes(size=total)) +
  geom_line(aes(x=as.integer(period)))
```



We can fit the logistic regression model (noting that the model with the interaction of location and period

would be saturated):

```
m <- glm(cbind(survive,perish) ~ location * period, family=binomial, data=troutegg)
summary(m)
```

```
##
## Call:
## glm(formula = cbind(survive, perish) ~ location * period, family = binomial,
##     data = troutegg)
##
## Deviance Residuals:
##  [1]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)             2.8792     0.4596   6.265 3.74e-10 ***
## location2               1.0911     0.8489   1.285  0.19870
## location3               0.5136     0.6853   0.749  0.45356
## location4              24.4707 51676.2179   0.000  0.99962
## location5              -2.7716     0.5044  -5.495 3.90e-08 ***
## period7                 0.2778     0.6869   0.404  0.68591
## period8                -0.7326     0.5791  -1.265  0.20582
## period11               -0.5695     0.5383  -1.058  0.29007
## location2:period7      -2.4453     1.0291  -2.376  0.01749 *
## location3:period7      -2.4667     0.8796  -2.804  0.00504 **
## location4:period7     -26.0789 51676.2179  -0.001  0.99960
## location5:period7      -2.6125     0.7847  -3.329  0.00087 ***
## location2:period8      -0.9690     0.9836  -0.985  0.32453
## location3:period8      -1.6169     0.7983  -2.025  0.04284 *
## location4:period8     -25.8783 51676.2179  -0.001  0.99960
## location5:period8      -0.7331     0.6696  -1.095  0.27354
## location2:period11     -1.6468     0.9297  -1.771  0.07651 .
## location3:period11     -1.8388     0.7672  -2.397  0.01654 *
## location4:period11    -25.1154 51676.2179   0.000  0.99961
## location5:period11    -27.1679 51597.7368  -0.001  0.99958
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.0215e+03  on 19  degrees of freedom
## Residual deviance: 5.5198e-10  on  0  degrees of freedom
## AIC: 116.53
##
## Number of Fisher Scoring iterations: 22
```

The residual deviance seems a little large. With 12 residual degrees of freedom, the deviance should be near 12. We can confirm that the deviance is quite large via:

```
1 - pchisq( 64.5, df=12 )
```

```
## [1] 3.372415e-09
```

We therefore estimate the overdispersion parameter

```
sigma2 <- sum( residuals(m, type='pearson') ^2 ) / 12
sigma2
```

```
## [1] 2.29949e-11
```

and note that this is quite a bit larger than 1, which is what it should be in the non-overdispersed setting. Using this we can now test the significance of the effects of location and period.

```
drop1(m, scale=sigma2, test='F')
```

```
## Warning in drop1.glm(m, scale = sigma2, test = "F"): F test assumes
## 'quasibinomial' family
```

```
## Warning in pf(q = q, df1 = df1, ...): NaNs produced
```

```
## Single term deletions
##
## Model:
## cbind(survive, perish) ~ location * period
##
## scale:  2.29949e-11
##
##                 Df Deviance        AIC F value Pr(>F)
## <none>                0.000 1.1700e+02
## location:period 12   64.495 2.8048e+12       0
```

and conclude that both location and period are significant predictors of trout egg survivorship.

We could have avoided having to calculate $\hat{\sigma}^2$ by hand by simply using the `quasibinomial` family instead of the binomial.

```
m2 <- glm(cbind(survive,perish) ~ location + period,
          family=quasibinomial, data=troutegg)
summary(m2)
```

```
##
## Call:
## glm(formula = cbind(survive, perish) ~ location + period, family = quasibinomial,
##     data = troutegg)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.8305  -0.3650  -0.0303   0.6191   3.2434
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.6358     0.6495   7.138 1.18e-05 ***
## location2    -0.4168     0.5682  -0.734 0.477315
## location3    -1.2421     0.5066  -2.452 0.030501 *
## location4    -0.9509     0.5281  -1.800 0.096970 .
## location5    -4.6138     0.5777  -7.987 3.82e-06 ***
## period7      -2.1702     0.5504  -3.943 0.001953 **
## period8      -2.3256     0.5609  -4.146 0.001356 **
## period11     -2.4500     0.5405  -4.533 0.000686 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 5.330358)
##
##     Null deviance: 1021.469  on 19  degrees of freedom
## Residual deviance:   64.495  on 12  degrees of freedom
```

```
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```
drop1(m2, test='F')
```

```
## Single term deletions
##
## Model:
## cbind(survive, perish) ~ location + period
##          Df Deviance F value    Pr(>F)
## <none>         64.50
## location  4   913.56  39.494 8.142e-07 ***
## period    3   228.57  10.176  0.001288 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# anova(m2, test='F')
```

While each of the time periods is different than the first, it looks like periods 7,8, and 11 aren't different from each other. As usual, we need to turn to the `emmeans` package for looking at the pairwise differences between the periods.

```
emmeans(m2, pairwise~period, type='response')
```

```
## $emmeans
##  period      prob         SE  df asymp.LCL asymp.UCL
##  4      0.9604983 0.01770177 Inf 0.9069267 0.9837862
##  7      0.7351530 0.05673286 Inf 0.6105986 0.8309006
##  8      0.7038064 0.06180293 Inf 0.5706296 0.8094684
##  11     0.6772484 0.05486759 Inf 0.5619734 0.7743668
##
## Results are averaged over the levels of: location
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
##
## $contrasts
##  contrast odds.ratio        SE  df z.ratio p.value
##  4 / 7      8.759889 4.8214944 Inf   3.943  0.0005
##  4 / 8     10.233018 5.7397360 Inf   4.146  0.0002
##  4 / 11    11.587822 6.2629001 Inf   4.533  <.0001
##  7 / 8      1.168167 0.4746254 Inf   0.383  0.9810
##  7 / 11     1.322827 0.5008150 Inf   0.739  0.8814
##  8 / 11     1.132395 0.4309174 Inf   0.327  0.9880
##
## Results are averaged over the levels of: location
## P value adjustment: tukey method for comparing a family of 4 estimates
## Tests are performed on the log odds ratio scale
```

```
emmeans(m2, ~period, type='response') %>% cld(Letters=letters)
```

```
##  period      prob         SE  df asymp.LCL asymp.UCL .group
##  11     0.6772484 0.05486759 Inf 0.5619734 0.7743668  a
##  8      0.7038064 0.06180293 Inf 0.5706296 0.8094684  a
##  7      0.7351530 0.05673286 Inf 0.6105986 0.8309006  a
##  4      0.9604983 0.01770177 Inf 0.9069267 0.9837862   b
##
```

```
## Results are averaged over the levels of: location
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
## P value adjustment: tukey method for comparing a family of 4 estimates
## Tests are performed on the log odds ratio scale
## significance level used: alpha = 0.05
```

Looking at this experiment, it looks like there is an interaction between location and period. If we fit a model with the interaction, it is the saturated model (20 covariates for 20 observations)

```
m3 <- glm( cbind(survive, perish) ~ period * location, family=binomial, data=troutegg)
summary(m3)
```

```
##
## Call:
## glm(formula = cbind(survive, perish) ~ period * location, family = binomial,
##     data = troutegg)
##
## Deviance Residuals:
##  [1]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)            2.8792     0.4596   6.265 3.74e-10 ***
## period7                0.2778     0.6869   0.404  0.68591
## period8               -0.7326     0.5791  -1.265  0.20582
## period11              -0.5695     0.5383  -1.058  0.29007
## location2              1.0911     0.8489   1.285  0.19870
## location3              0.5136     0.6853   0.749  0.45356
## location4             24.4707 51676.2184   0.000  0.99962
## location5             -2.7716     0.5044  -5.495 3.90e-08 ***
## period7:location2     -2.4453     1.0291  -2.376  0.01749 *
## period8:location2     -0.9690     0.9836  -0.985  0.32453
## period11:location2    -1.6468     0.9297  -1.771  0.07651 .
## period7:location3     -2.4667     0.8796  -2.804  0.00504 **
## period8:location3     -1.6169     0.7983  -2.025  0.04284 *
## period11:location3    -1.8388     0.7672  -2.397  0.01654 *
## period7:location4    -26.0789 51676.2184  -0.001  0.99960
## period8:location4    -25.8783 51676.2184  -0.001  0.99960
## period11:location4   -25.1154 51676.2184   0.000  0.99961
## period7:location5     -2.6125     0.7847  -3.329  0.00087 ***
## period8:location5     -0.7331     0.6696  -1.095  0.27354
## period11:location5   -27.1679 51597.7368  -0.001  0.99958
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.0215e+03  on 19  degrees of freedom
## Residual deviance: 5.5196e-10  on  0  degrees of freedom
## AIC: 116.53
##
## Number of Fisher Scoring iterations: 22
```

Notice that we've fit the saturated model and we've resolved the overdispersion problem because of that.

As usual, we can now look at the effect of `period` at each of the locations.

```
emmeans(m3, ~ period | location, type='response') %>% cld()
```

```
## location = 1:
##  period         prob           SE  df    asymp.LCL asymp.UCL .group
##  8      8.953488e-01 3.300798e-02 Inf 8.109406e-01 0.9446443  1
##  11     9.096774e-01 2.302375e-02 Inf 8.532710e-01 0.9457777  1
##  4      9.468085e-01 2.314665e-02 Inf 8.785095e-01 0.9776867  1
##  7      9.591837e-01 1.998733e-02 Inf 8.962639e-01 0.9845962  1
##
## location = 2:
##  period         prob           SE  df    asymp.LCL asymp.UCL .group
##  11     8.524590e-01 3.210799e-02 Inf 7.779341e-01 0.9050269  1
##  7      8.584906e-01 3.385381e-02 Inf 7.784456e-01 0.9128537  1
##  8      9.062500e-01 2.974911e-02 Inf 8.295443e-01 0.9504977  12
##  4      9.814815e-01 1.297276e-02 Inf 9.289964e-01 0.9953638   2
##
## location = 3:
##  period         prob           SE  df    asymp.LCL asymp.UCL .group
##  11     7.280000e-01 3.980111e-02 Inf 6.434907e-01 0.7987421  1
##  8      7.394958e-01 4.023479e-02 Inf 6.533948e-01 0.8104142  1
##  7      7.692308e-01 3.695265e-02 Inf 6.891126e-01 0.8336850  1
##  4      9.674797e-01 1.599358e-02 Inf 9.165610e-01 0.9877408   2
##
## location = 4:
##  period         prob           SE  df    asymp.LCL asymp.UCL .group
##  8      6.767677e-01 4.700674e-02 Inf 5.787856e-01 0.7613551  1
##  7      8.247423e-01 3.860215e-02 Inf 7.360186e-01 0.8881766  12
##  11     8.409091e-01 3.183558e-02 Inf 7.682755e-01 0.8939194   2
##  4      1.000000e+00 6.845126e-08 Inf 2.220446e-16 1.0000000  12
##
## location = 5:
##  period         prob           SE  df    asymp.LCL asymp.UCL .group
##  11     1.001302e-12 5.166492e-08 Inf 2.220446e-16 1.0000000  12
##  7      9.734513e-02 2.788552e-02 Inf 5.472898e-02 0.1672733  1
##  8      2.045455e-01 4.299929e-02 Inf 1.328383e-01 0.3015023  1
##  4      5.268817e-01 5.177260e-02 Inf 4.256954e-01 0.6259069   2
##
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
## P value adjustment: tukey method for comparing a family of 4 estimates
## Tests are performed on the log odds ratio scale
## significance level used: alpha = 0.05
```

Notice that there isn't a huge difference in period for locations 1-4, but in location 5 things are very different.

We might consider that location really ought to be a random effect. Fortunately `lme4` supports the family option, although it will not accept quasi families, you either fit a random effect or fit a quasibinomial. In either case, fitting the full interaction model with `period*location` doesn't work because we have a saturated model

```
# addititive model
m3 <- glmer(cbind(survive,perish) ~ period + (1|location),
          family=binomial, data=troutegg)
summary(m3)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: cbind(survive, perish) ~ period + (1 | location)
##    Data: troutegg
##
##      AIC      BIC   logLik deviance df.resid
##    180.3    185.3    -85.2    170.3       15
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.2274 -0.3592  0.0027  0.6531  3.5841
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  location (Intercept) 2.682    1.638
## Number of obs: 20, groups:  location, 5
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.1799     0.7594   4.187 2.82e-05 ***
## period7      -2.1545     0.2368  -9.097  < 2e-16 ***
## period8      -2.3085     0.2414  -9.564  < 2e-16 ***
## period11     -2.4324     0.2325 -10.460  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) perid7 perid8
## period7  -0.224
## period8  -0.225  0.732
## period11 -0.234  0.758  0.761
```

```
emmeans(m3, ~period, type='response') %>% cld(Letters=letters)
```

```
##  period      prob        SE  df asymp.LCL asymp.UCL .group
##  11     0.6786439 0.1614504 Inf 0.3310535 0.9001163  a
##  8      0.7050357 0.1546023 Inf 0.3576252 0.9112080  a
##  7      0.7360218 0.1443664 Inf 0.3939061 0.9228496  a
##  4      0.9600725 0.0291111 Inf 0.8444222 0.9906998   b
##
## Confidence level used: 0.95
## Intervals are back-transformed from the logit scale
## P value adjustment: tukey method for comparing a family of 4 estimates
## Tests are performed on the log odds ratio scale
## significance level used: alpha = 0.05
```

```
# Interaction  model
m4 <- glmer(cbind(survive,perish) ~ period + (1 + period|location),
          family=binomial, data=troutegg)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.0184817
## (tol = 0.001, component 1)
```

```
#summary(m4)
#emmeans(m4, ~period, type='response') %>% cld(Letters=letters)
```

Unfortunately, we aren't able to fit the saturated model using random effects because the numerical optimization function for finding MLE estimates failed to converge.

12.7 ROC Curves

The dataset `faraway::wbca` comes from a study of breast cancer in Wisconsin. There are 681 cases of potentially cancerous tumors of which 238 are actually malignant (ie cancerous). Determining whether a tumor is really malignant is traditionally determined by an invasive surgical procedure. The purpose of this study was to determine whether a new procedure called 'fine needle aspiration', which draws only a small sample of tissue, could be effective in determining tumor status.

```
data('wbca', package='faraway')

# clean up the data
wbca <- wbca %>%
  mutate(Class = ifelse(Class==0, 'malignant', 'benign')) %>%
  dplyr::select(Class, BNucl, UShap, USize)

# Fit the model where Malignant is considered a success
# model <- glm( I(Class=='malignant') ~ ., data=wbca, family='binomial' )  # emmeans hates this version

model <- wbca %>% mutate( Class = (Class == 'malignant') ) %>%   # Clear what is success
  glm( Class ~., data=., family='binomial' )                     # and emmeans still happy

# Get the response values
# type='response' gives phat values which live in [0,1]
# type='link' gives the Xbeta values whice live in (-infinity, infinity)
wbca <- wbca %>%
  mutate(phat = predict(model, type='response'),
         yhat = ifelse(phat > .5, 'malignant', 'benign'))

# Calculate the confusion matrix
table( Truth=wbca$Class, Predicted=wbca$yhat )
```

```
##            Predicted
## Truth       benign malignant
##   benign       432        11
##   malignant     15       223
```

As usual we can calculate the summary tables...

```
summary(model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = .)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8890  -0.1409  -0.1409   0.0287   2.2284
##
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.35433    0.54076 -11.751  < 2e-16 ***
## BNucl        0.55297    0.08041   6.877 6.13e-12 ***
## UShap        0.62583    0.17506   3.575 0.000350 ***
## USize        0.56793    0.15910   3.570 0.000358 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 881.39  on 680  degrees of freedom
## Residual deviance: 148.43  on 677  degrees of freedom
## AIC: 156.43
##
## Number of Fisher Scoring iterations: 7
```

From this table, we see that for a breast tumor, the larger values of `BNucl`, `UShap`, and `USize` imply a greater probability of it being malignant. So for a tumor with

```
##   BNucl UShap USize
## 1     2     1     2
```

We would calculate

$$\begin{aligned} X\hat{\beta} &= \hat{\beta}_0 + 2 \cdot \hat{\beta}_1 + 1 \cdot \hat{\beta}_2 + 2 \cdot \hat{\beta}_3 \\ &= -6.35 + 2 * (0.553) + 1 * (0.626) + 2 * (0.568) \\ &= -3.482 \end{aligned}$$

and therefore

$$\hat{p} = \frac{1}{1 + e^{-X\hat{\beta}}} = \frac{1}{1 + e^{3.482}} = 0.0297$$

```
newdata = data.frame( BNucl=2, UShap=1, USize=2 )
predict(model, newdata=newdata)
```

```
##         1
## -3.486719
```

```
predict(model, newdata=newdata, type='response')
```

```
##         1
## 0.0296925
```

So for a tumor with these covariates, we would classify it as most likely to be benign.

In this medical scenario where we have to decide how to classify if a tumor is malignant of benign, we shouldn't treat the misclassification errors as being the same. If we incorrectly identify a tumor as malignant when it is not, that will cause a patient to undergo a somewhat invasive surgury to remove the tumor. However if we incorrectly identify a tumor as being benign, then the cancerous tumor will likely grow and eventually kill the patient. While the first error is regretable, the second is far worse.

Given that reasoning, perhaps we shouldn't use the rule: If $\hat{p} >= 0.5$ classify as malignant. Instead perhaps we should use $\hat{p} >= 0.3$ or $\hat{p} >= 0.05$.

Whatever decision rule we make, we should consider how many of each type of error we make. Consider the following confusion matrix:

	Predict Negative	Predict Positive	Total
Negative	True Neg (TN)	False Pos (FP)	N

	Predict Negative	Predict Positive	Total
Positive	False Neg (FN)	True Pos (TP)	P
Total	N^*	P^*	

where P is the number of positive cases, N is the number of negative cases, P^* is the number of observations predicted to be positive, and N^* is the number predicted to be negative.

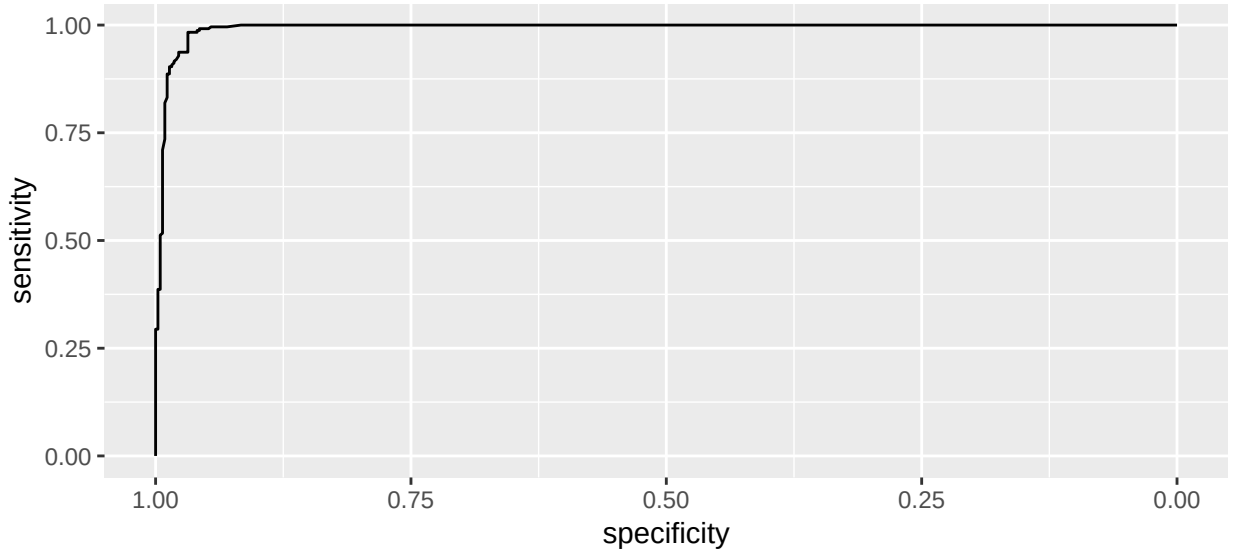
Quantity	Formula	Synonyms
False Positive Rate	FP/N	Type I Error; 1-Specificity
True Positive Rate	TP/P	Power; Sensitivity; Recall
Pos. Pred. Value	TP/P^*	Precision

We can think of the True Positive Rate as the probability that a Positive case will be correctly classified as a positive. Similarly a False Positive Rate is the probability that a Negative case will be incorrectly classified as a positive.

I wish to examine the relationship between the False Positive Rate and the True Positive Rate for any decision rule. So what we could do is select a sequence of decision rules and for each calculate the `(FPR, TPR)` pair, and then make a plot where we play connect the dots with the `(FPR, TPR)` pairs.

Of course we don't want to have to do this by hand, so we'll use the package `pROC` to do it for us.

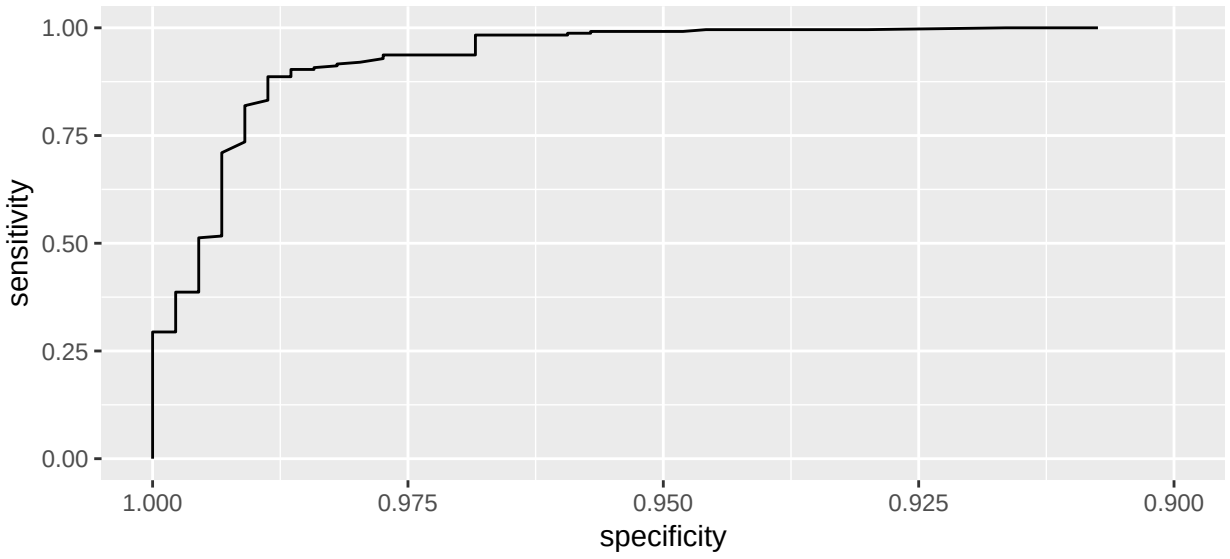
```
# Calculate the ROC information using pROC::roc()
myROC <- roc( wbca$Class,  wbca$phat )

# make a nice plot using ggplot2 and pROC::ggroc()
ggroc( myROC )
```



This looks pretty good and in an ideal classifier that makes prefect predictions, this would be a perfect right angle at the bend.

Lets zoom in a little on the high specificity values (i.e. low false positive rates)

```
ggroc( myROC ) + xlim(1, .9)
```



We see that if we want to correctly identify about 99% of maligant tumors, we will have a false positive rate of about 1-0.95 = 0.05. So about 5% of benign tumors would be incorrectly classified as malignant.

It is a little challenging to read the graph to see what the Sensitivity is for a particular value of Specificity. To do this we'll use another function because the authors would prefer you to also estimate the confidence intervals for the quantity. This is a case where bootstrap confidence intervals are quite effective.

```
ci(myROC, of='sp', sensitivities=.99)
```

```
## 95% CI (2000 stratified bootstrap replicates):
##    se sp.low sp.median sp.high
##  0.99 0.9222    0.9571  0.9774
```

```
ci(myROC, of='se', specificities=.975)
```

```
## 95% CI (2000 stratified bootstrap replicates):
##     sp se.low se.median se.high
##  0.975 0.8794    0.9412  0.9958
```

One measure of how far we are from the perfect predictor is the area under the curve. The perfect model would have an area under the curve of 1. For this model the area under the curve is:

```
auc(myROC)
```

```
## Area under the curve: 0.9929
```

```
ci(myROC, of='auc')
```

```
## 95% CI: 0.9878-0.9981 (DeLong)
```

```
ci(myROC, of='auc', method='bootstrap')
```

```
## 95% CI: 0.9876-0.9973 (2000 stratified bootstrap replicates)
```

which seems pretty good and Area Under the Curve (AUC) is often used as a way of comparing the quality of binary classifiers.

12.8 Exercises

1. The dataset `faraway::wbca` comes from a study of breast cancer in Wisconsin. There are 681 cases of potentially cancerous tumors of which 238 are actually malignant (i.e. cancerous). Determining whether a tumor is really malignant is traditionally determined by an invasive surgical procedure. The purpose of this study was to determine whether a new procedure called 'fine needle aspiration', which draws only a small sample of tissue, could be effective in determining tumor status.
   a. Fit a binomial regression with `Class` as the response variable and the other nine variables as predictors (for consistency among students, define a success as the tumor being benign and remember that `glm` wants the response to be a matrix where the first column is the number of successes). Report the residual deviance and associated degrees of freedom. Can this information be used to determine if this model fits the data?
   b. Use AIC as the criterion to determine the best subset of variables using the step function.
   c. Use the reduced model to give the estimated probability that a tumor with associated predictor variables

```
newdata <- data.frame( Adhes=1, BNucl=1, Chrom=3, Epith=2, Mitos=1,
                       NNucl=1, Thick=4, UShap=1, USize=1)
```

   is benign and give a confidence interval for your estimate.
   d. Suppose that a cancer is classified as benign if $\hat{p} > 0.5$ and malignant if $\hat{p} \leq 0.5$. Compute the number of errors of both types that will be made if this method is applied to the current data with the reduced model. *Hint: save the $\hat{p}$ as a column in the wbca data frame and use that to create a new column `Est_Class` which is the estimated class (making sure it is the same encoding scheme as Class). Then use dplyr functions to create a table of how many rows fall into each of the four Class/Est_Class combinations.*
   e. Suppose we changed the cutoff to 0.9. Compute the number of errors of each type in this case. Discuss the ethical issues in determining the cutoff.
2. Aflatoxin B1 was fed to lab animals at various doses and the number responding with liver cancer recorded and is available in the dataset `faraway::aflatoxin`.
   a. Build a model to predict the occurrence of liver cancer. Consider a square-root transformation to the dose level.
   b. Compute the ED50 level (effective dose level… same as LD50 but isn't confined to strictly lethal effects) and an approximate 95% confidence interval.
3. The dataset `faraway::pima` is data from a study of adult female Pima Indians living near Phoenix was done and resulted $n = 752$ observations after the cases of missing data (obnoxiously coded as 0) were removed. Testing positive for diabetes was the success (`test`) and the predictor variables we will use are: `pregnant`, `glucose`, and `bmi`.
   a. Remove the observations that have missing data (coded as a zero) for either `glucose` or `bmi`. *The researcher's choice of using 0 to represent missing data is a bad idea because 0 is a valid value for the number of pregnancies, so assume a zero in the `pregnant` covariate is a true value. The `dplyr` function `filter` could be used here.*
   b. Fit the logistic regression model for `test` with using the main effects of `glucose`, `bmi`, and `pregnant`.
   c. Produce a graphic that displays the relationship between the variables. *Notice I've done the part (a) for you and the assume that your model produced in part (b) is named `m`. I also split up the pregnancy and bmi values into some logical grouping for the visualization. If you've never used the `cut` function, go look it up because it is extremely handy.*

```
pima <- pima %>% filter( bmi != 0, glucose != 0)
pima <- pima %>% mutate(
  phat=ilogit(predict(m)),
  pregnant.grp = cut(pregnant, c(0,1,3,6,100), right = FALSE, labels = c('0','1:2','3:5','6+')),
  bmi.grp = cut(bmi, c(0,18,25,30,100), labels=c('Underweight','Normal','Overweight','Obese')))
ggplot(pima, aes(y=test, x=glucose)) +
```

```
  geom_point() +
  geom_line(aes(y=phat), color='red') +
  facet_grid(bmi.grp ~ pregnant.grp)
```

d. Discuss the quality of your predictions based on the graphic above and modify your model accordingly.

e. Give the probability of testing positive for diabetes for a Pima woman who had had no pregnancies, had `bmi=28` and a glucose level of `110`.
f. Give the odds that the same woman would test positive for diabetes.
g. How do her odds change to if she were to have a child? That is to say, what is the odds ratio for that change?