Introduction to Statistics for Researchers, Second Semester

Derek Sonderegger

April 30, 2014

These notes were originally written for an introductory statistics year long course sequence for grad students in biological sciences.

The problem with most introductory statistics courses is that they don't prepare the student for the use of advanced statistics. Rote hand calculation is easy to test, easy to grade, and easy for students to learn to do, but is useless for actually understanding how to apply statistics. Since students pursuing a Ph.D. will likely be using statistics for the rest of their professional careers, I feel that this sort of course should attempt to steer away from a "cookbook" undergraduate pedagogy, and give the student enough theoretical background to continue their statistical studies at a high level while staying away from the painful mathematical details that statisticians must work through.

Statistical software has progressed by leaps and bounds over the last decades. Scientists need access to reliable software that is flexible enough to handle new problems, with minimal headaches. R has become a widely used and robust open source platform for statistical computing and most new methodologies will appear in R before being incorporated into commercial software. The only downside is that there is a substantial learning curve to learning a scripting language, particularly for students without any programming background. I attempt to introduce the software with as little pain as possible, but some frustration is inevitable.

Since the mathematical and statistical background of typical students varies widely, the course seems to have a split-personality disorder. We wish to talk about using calculus to maximize the likelihood function, but also must spend time defining how to calculate the a mean. I attempt to address both audiences, but recognize that it is not ideal. The perfect solution for students with infinite cash reserves is to buy an undergrad statistics textbook along with a more detailed technical introduction, and finally a good R reference manual and read from different books as their comfort level with the material dictates. However, this is unrealistic and students won't know when to skip from one book to another. This set of notes is an attempt to bridge that gap.

This set of notes was substantially influenced by the *Linear Models with R* by Julian Faraway, however, there are substantial differences in content as well.

I hope you'll find these notes useful and would appreciate knowing who is using them outside of my university. In particular, I will be happy to learn about people using these notes to teach from. I am also grateful for useful feed back and error reports sent via email.

Derek Sonderegger, Ph.D.
Department of Mathematics and Statistics
Northern Arizona University
derek.sonderegger@nau.edu

Contents

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.

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.

1.2.3.1 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.

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

1.2.3.2 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 $z_{ij} = \boldsymbol{x}_{i\cdot}\boldsymbol{w}_{\cdot j}$ 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 $\boldsymbol{W}_{4\times2}$. Since the *inner* dimensions match for this multiplication, then $\boldsymbol{Z}_{3\times2} = \boldsymbol{X}_{3\times4}\boldsymbol{W}_{4\times2}$ is defined where

$$\begin{aligned} z_{11} &= \boldsymbol{x}_{1\cdot}\boldsymbol{w}_{\cdot1} \\ &= (1 \cdot 13) + (2 \cdot 15) + (3 \cdot 17) + (4 \cdot 19) = 170 \end{aligned}$$

and similarly

$$\begin{aligned} z_{21} &= \boldsymbol{x}_{2\cdot}\boldsymbol{w}_{\cdot1} \\ &= (5 \cdot 13) + (6 \cdot 15) + (7 \cdot 17) + (8 \cdot 19) = 426 \end{aligned}$$

so that

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

For another example, we note that

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

Notice that this definition of multiplication means that the order matters. Above, we calculated $\boldsymbol{X}_{3\times4}\boldsymbol{W}_{4\times2}$ but we cannot reverse the order because the inner dimensions do not match up.

1.2.3.3 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.

1.2.4 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.

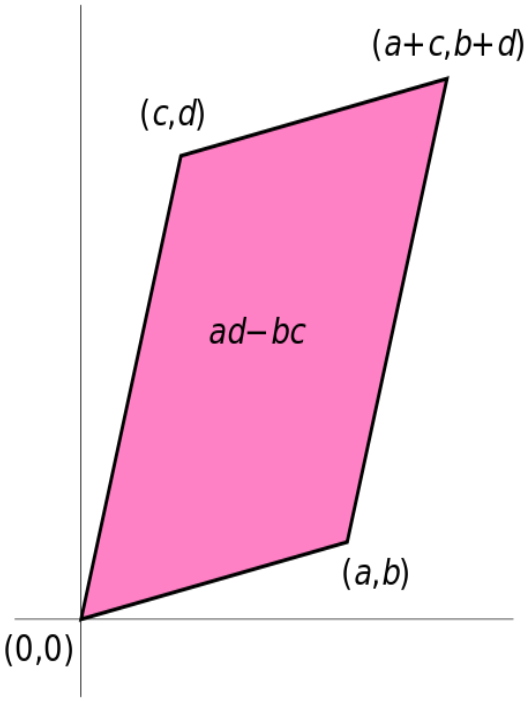
For a 2×2 matrix

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

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.



1.2.5 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

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

and thus

$$\begin{aligned} \boldsymbol{W}\boldsymbol{W}^{-1} &= \begin{bmatrix} 1 & 2 \\ 5 & 3 \end{bmatrix}\begin{bmatrix} -\frac{3}{7} & \frac{2}{7} \\ \frac{5}{7} & -\frac{1}{7} \end{bmatrix} \\ &= \begin{bmatrix} -\frac{3}{7} + \frac{10}{7} & \frac{2}{7} - \frac{2}{7} \\ -\frac{15}{7} + \frac{15}{7} & \frac{10}{7} - \frac{3}{7} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boldsymbol{I}_2 \end{aligned}$$

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.

Chapter 2

Estimation of linear model parameters

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:

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

Notice in the first example of multiple regression, 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 p^{th} 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 doesn't 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 Matrix Formulation

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

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

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 parameters we are interested in estimating.

2.1.1 Estimation of $\boldsymbol{\beta}$

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 σ^2

Recall our model is

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

where $\epsilon_i \overset{iid}{\sim} N\left(0, \sigma^2\right)$. 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 "y-hat" and has many interesting theoretical properties.[1]

We can now estimate the error terms via

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

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

$$\begin{aligned}\hat{\sigma}^2 &= \frac{1}{n-2}\sum_{i=1}^{n}\hat{\epsilon}_i^2 \\ &= \frac{1}{n-2}\,\hat{\boldsymbol{\epsilon}}^T\hat{\boldsymbol{\epsilon}}\end{aligned}$$

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

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

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

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

and any constants are pulled through the expectation

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

[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 $\boldsymbol{H}$.

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

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

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

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

2.1.4 Variance of $\hat{\boldsymbol{\beta}}$

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

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

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

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

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

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

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

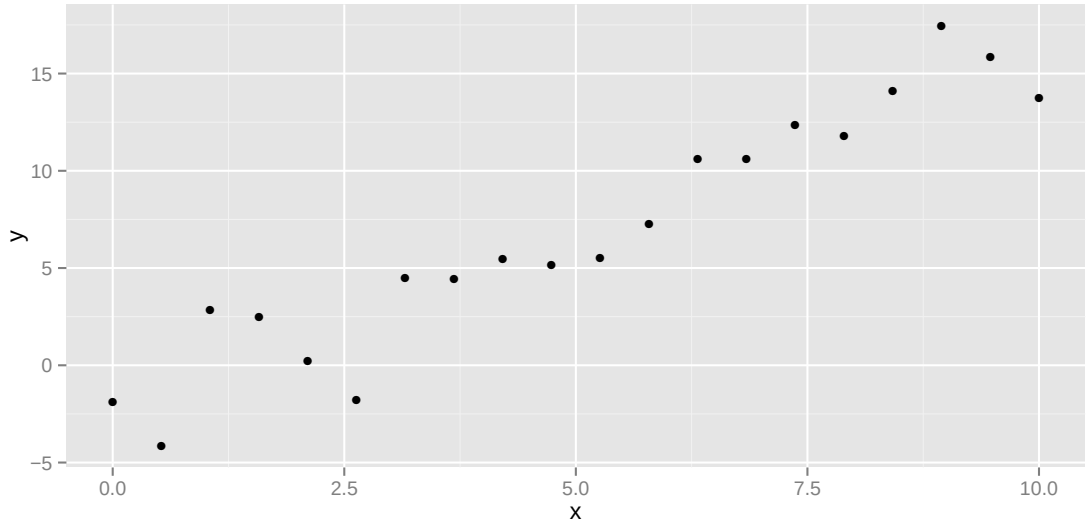
2.1.6 Summary of pertinent results

1. $\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$ is the unbiased maximum-likelihood estimator of $\boldsymbol{\beta}$.
2. The Central Limit Theorem applies to each element of $\boldsymbol{\beta}$. That is, as $n \to \infty$, the distribution of $\hat{\beta}_j \to N\left(\beta_j, \left[\sigma^2\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\right]_{jj}\right)$.
3. The error terms can be calculated via
$$\begin{aligned}\hat{\boldsymbol{y}} &= \boldsymbol{X}\hat{\boldsymbol{\beta}}\\ \hat{\boldsymbol{\epsilon}} &= \boldsymbol{y} - \hat{\boldsymbol{y}}\end{aligned}$$
4. The estimate of σ^2 is
$$\hat{\sigma}^2 = \frac{1}{n-p}\,\hat{\boldsymbol{\epsilon}}^T\hat{\boldsymbol{\epsilon}}$$
5. 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:

```
library(ggplot2)
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]
  -2.750
x  1.915
```

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

$$\begin{aligned} \hat{\boldsymbol{y}} &= \boldsymbol{X}\hat{\boldsymbol{\beta}} \\ \hat{\boldsymbol{\epsilon}} &= \boldsymbol{y} - \hat{\boldsymbol{y}} \end{aligned}$$

```
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(  diag(XtXinv) * sigma2.hat )
```

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

```
beta.hat

    [,1]
  -2.750
x  1.915

sigma.hat

      [,1]
[1,] 1.964

std.errs

[1] 0.8464 0.1447

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


Call:
lm(formula = y ~ x)

Residuals:
   Min     1Q Median     3Q    Max
 -4.05  -1.08   0.19   1.04   3.58

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -2.750      0.846   -3.25   0.0045 **
x              1.915      0.145   13.24    1e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.96 on 18 degrees of freedom
Multiple R-squared:  0.907,Adjusted R-squared:  0.902
F-statistic:  175 on 1 and 18 DF,  p-value: 1.02e-10
```

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 -4.528 -0.9718
x          1.611  2.2194
```

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

```
confint(model)

             2.5 %  97.5 %
(Intercept) -4.528 -0.9718
x            1.611  2.2194
```

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

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

Chapter 3

Inference

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[1]

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.

3.1.2 Example[2]

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:

[1] Section 3.1 in Faraway
[2] Section 3.2 in Faraway

Species	Number of species of tortoise found on the island
Endemics	Number of endemic species
Elevation	Highest point of the island (m)
Area	Island area (km^2)
Nearest	Distance to the nearest island (km)
Scruz	Distance to the Santa Cruz Islands (km)
Adjacent	Area of the closest neighboring island (km^2)

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

```
library(faraway)
data(gala)
gala[1:6,]  # print out the first 6 observations of the data frame
```

```
             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)

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

```

Call:
lm(formula = Species ~ Area + Elevation + Nearest + Scruz + Adjacent,
    data = gala)

Residuals:
    Min      1Q  Median      3Q     Max
-111.68  -34.90   -7.86   33.46  182.58

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.06822   19.15420    0.37   0.7154
Area        -0.02394    0.02242   -1.07   0.2963
Elevation    0.31946    0.05366    5.95  3.8e-06 ***
Nearest      0.00914    1.05414    0.01   0.9932
Scruz       -0.24052    0.21540   -1.12   0.2752
Adjacent    -0.07480    0.01770   -4.23   0.0003 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 61 on 24 degrees of freedom
Multiple R-squared:  0.766,Adjusted R-squared:  0.717
F-statistic: 15.7 on 5 and 24 DF,  p-value: 6.84e-07
```

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] 15.7
```

and should be compared against the F-distribution with 5 and 24 degrees of freedom. Since *large* differences 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] 6.839e-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
Difference	$p-1$	RSS_d	$MSE_d = RSS_d/df_d$	MSE_d/MSE_c
Complex	$n-p$	RSS_c	$MSE_c = RSS_c/df_c$	
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 15.7 6.8e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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] 1.14

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

[1] 0.2963

sqrt(F.stat)

[1] 1.068
```

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)  7.068221   19.15420  0.369017 7.154e-01
Area        -0.023938    0.02242 -1.067611 2.963e-01
Elevation    0.319465    0.05366  5.953188 3.823e-06
Nearest      0.009144    1.05414  0.008674 9.932e-01
Scruz       -0.240524    0.21540 -1.116628 2.752e-01
Adjacent    -0.074805    0.01770 -4.226217 2.971e-04

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

[1] -1.068

2 * pt(t, 24)

[1] 0.2963
```

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      4238 1.14    0.3
```

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 0.97   0.43
```

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 β Confidence Intervals[3]

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,]  6.095e-02 -8.164e-05  9.312e-06
[2,] -8.164e-05  2.724e-07 -7.126e-08
[3,]  9.312e-06 -7.126e-08  6.478e-08

diag(XtXinv)

[1] 6.095e-02 2.724e-07 6.478e-08
```

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

[3] Section 3.4 in Faraway

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

          [,1]
[1,]  1.43287
[2,]  0.27657
[3,] -0.06889
```

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] 60.86
```

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

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

[1] 15.02469  0.03176  0.01549
```

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
-103.4  -34.3  -11.4   22.6  203.7

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.4329    15.0247    0.10  0.92473
Elevation     0.2766     0.0318    8.71  2.5e-09 ***
Adjacent     -0.0689     0.0155   -4.45  0.00013 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.9 on 27 degrees of freedom
Multiple R-squared:  0.738,Adjusted R-squared:  0.718
F-statistic: 37.9 on 2 and 27 DF,  p-value: 1.43e-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,] -29.3952 32.2610
[2,]   0.2114  0.3417
[3,]  -0.1007 -0.0371
```

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) -29.3952 32.2610
Elevation     0.2114  0.3417
Adjacent     -0.1007 -0.0371
```

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 $\boldsymbol{X}$. Since 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 200 km^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,] 98.28
```

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] -28.7 225.3
```

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]  75.21 121.35
```

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 98.28 -28.7 225.3

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

    fit   lwr   upr
1 98.28 75.21 121.4
```

3.4 Interpretation[4]

The standard interpretation of the slope parameter is that β_j is the amount of increase in y if the jth covariate were to increase by 1, 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:

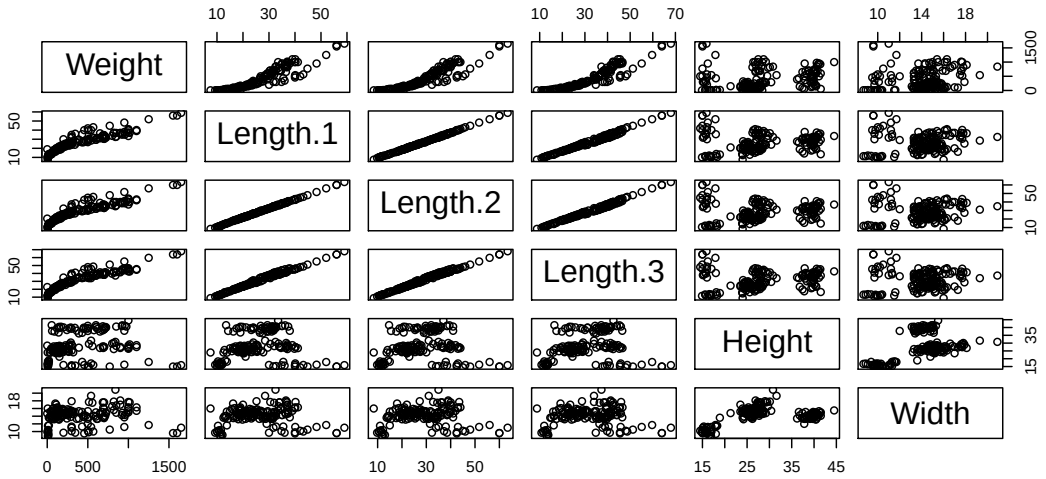
Weight	Weight of the fish (in grams)
Length 1	Length from the nose to the beginning of the tail (in cm)
Length 2	Length from the nose to the notch of the tail (in cm)
Length 3	Length from the nose to the end of the tail (in cm)
Height	Maximal height as % of Length3
Width	Maximal width as % of Length3
Sex	1 = male 0 = female
Species	Species of Perch (1-7)

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

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

```
#setwd('~/Dropbox/NAU/Teaching/STA 571/Notes/c1.3_Inference')
```

```
fish <- read.table('~/Dropbox/NAU/Teaching/STA 571/Notes/c1.3_Inference/Fish.csv',
                   header=TRUE, skip=111, sep=',')
pairs(fish[,c('Weight','Length.1','Length.2','Length.3','Height','Width')])
```



We might then naively create a linear model that has all of the length measurements.

[4]Sections 3.7 - 3.9 in Faraway

```
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
-302.2  -79.7  -39.9   92.6  344.8

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -724.54      77.13   -9.39   <2e-16 ***
Length.1       32.39      45.13    0.72    0.474
Length.2       -9.18      48.37   -0.19    0.850
Length.3        8.75      16.28    0.54    0.592
Height          4.95       2.77    1.79    0.076 .
Width           8.64       6.97    1.24    0.217
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 133 on 152 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.867,Adjusted R-squared:  0.863
F-statistic:  199 on 5 and 152 DF,  p-value: <2e-16
```

This is crazy. There is a negative relationship between `Length.2` and `Weight`. That doesn't 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
-306.1  -75.1  -36.4   89.5  338.0

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -701.075     71.044   -9.87  < 2e-16 ***
Length.2      30.436      0.984   30.93  < 2e-16 ***
Height         5.514      1.431    3.85  0.00017 ***
Width          5.651      5.202    1.09  0.27897
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 132 on 154 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.867,Adjusted R-squared:  0.864
F-statistic:  334 on 3 and 154 DF,  p-value: <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 variables 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.

Chapter 4

One-Way Analysis of Variance

Given a categorical covariate (which I will call a *factor*) with I levels, we are interested in fitting the model

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

where $\epsilon_{ij} \overset{iid}{\sim} N\left(0, \sigma^2\right)$, μ is the overall mean, and α_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 α_i and the predicted value $\mu + \alpha_i$ would not change. There are two easy restrictions we could make to make the model identifiable:

1. Set $\mu = 0$. In this case, α_i represents the expected value of an observation in group level i. We call this the "*cell means*" representation.
2. Set $\alpha_1 = 0$. Then μ represents the expected value of treatment 1, and the α_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.

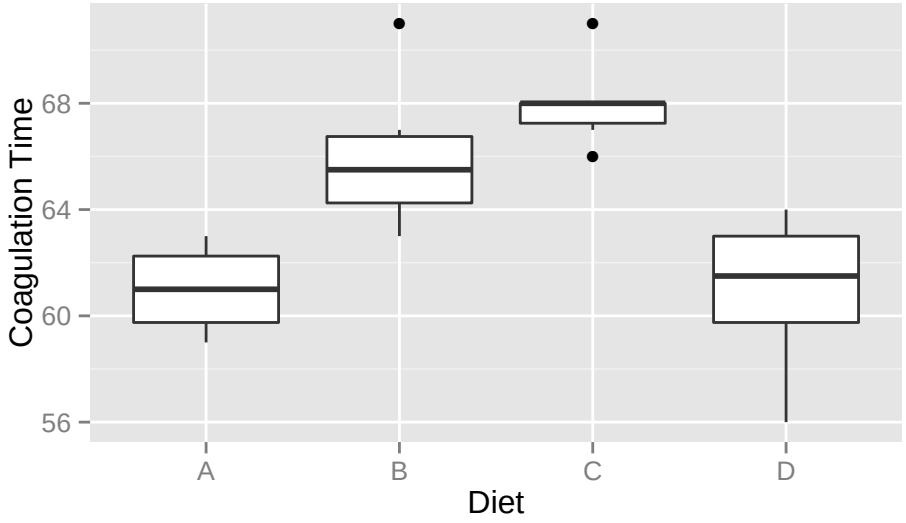
We will be interested in testing the null and alternative hypotheses

$$\begin{aligned} H_0: \ y_{ij} &= \mu + \epsilon_{ij} \\ H_a: \ y_{ij} &= \mu + \alpha_i + \epsilon_{ij} \end{aligned}$$

4.1 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.

```
library(ggplot2)
library(faraway)
data(coagulation)
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) 6.10e+01   1.18e+00   51.55  < 2e-16 ***
dietB       5.00e+00   1.53e+00    3.27  0.00380 **
dietC       7.00e+00   1.53e+00    4.58  0.00018 ***
dietD       2.99e-15   1.45e+00    0.00  1.00000
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.37 on 20 degrees of freedom
Multiple R-squared:  0.671,Adjusted R-squared:  0.621
F-statistic: 13.6 on 3 and 20 DF,  p-value: 4.66e-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.

4.2 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

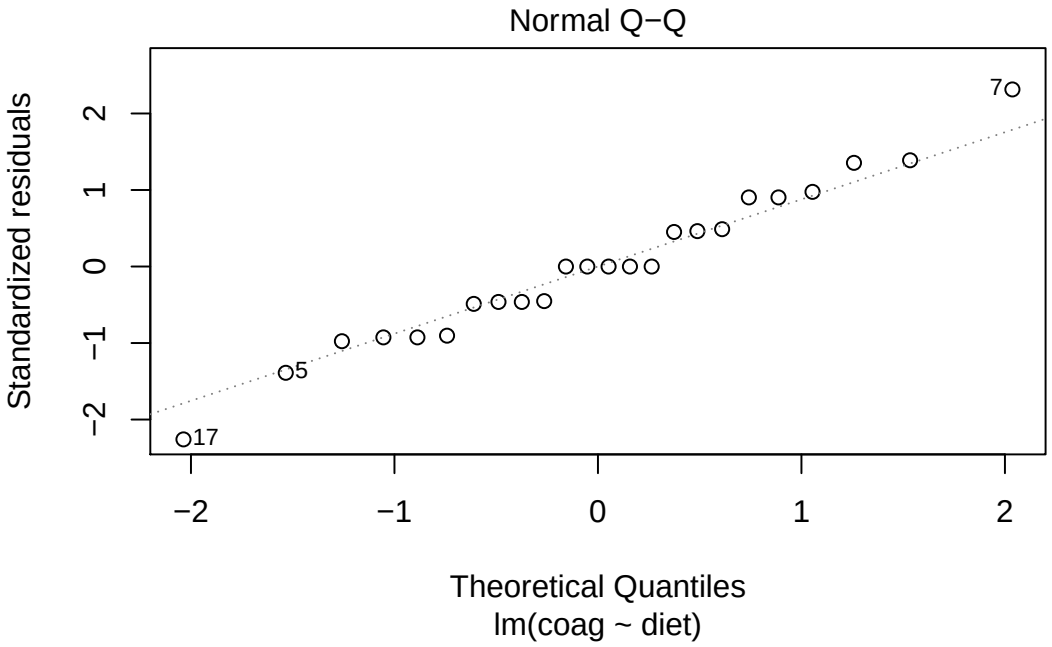
$$y_{ij} \sim \mu + \alpha_i + \epsilon_{ij}$$

where there really are only 4 parameters that define the expected value because $\alpha_1 = 0$. 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.

4.3 Diagnostics

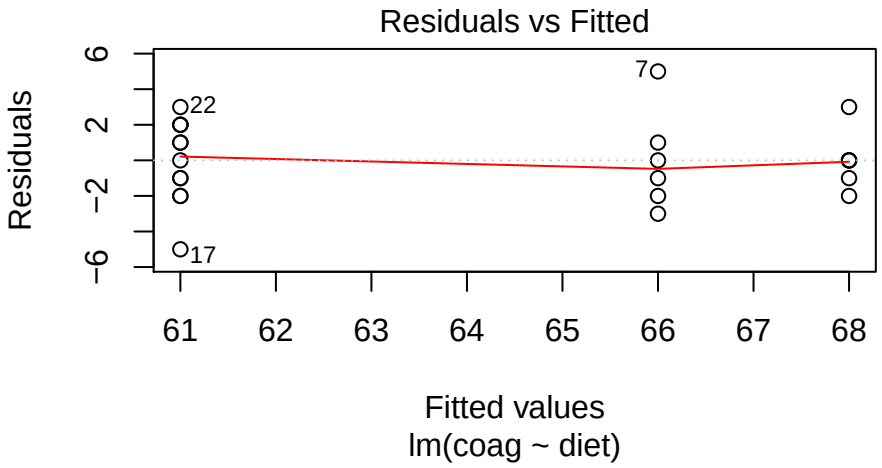
It is still important to check the diagnostics plots, but certain diagnostic plots will be useless. In particular, we need to be concerned about constant variance among the groups and normality of the residuals.

```
m <- lm(coag ~ diet, data=coagulation)
plot(m, which=2)  #  QQ plot
```



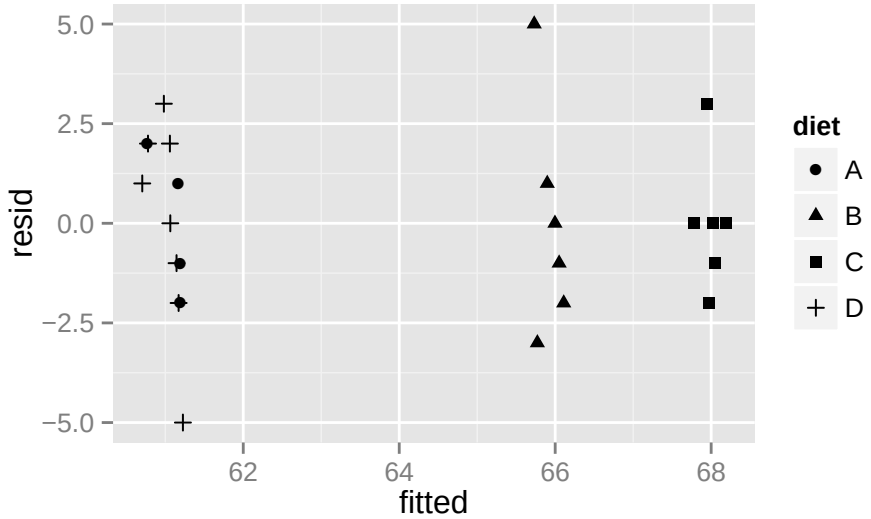
The residual plots however, might need a little bit of extra work, because there are only four possible predicted values (actually 3 because group A and D have the same predicted values). Note that we actually have $n = 24$ observations, but I can only see 16 of them.

```
plot(m, which=1)  #  Residals vs fitted
```



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.

```
data <- coagulation
data$fitted <- predict(m)
data$resid <- resid(m)
p <- ggplot(data, aes(x=fitted, y=resid, shape=diet)) +
   geom_point(position=position_jitter(w=0.3, h=0))
print(p)
```



4.4 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, but the easiest is to use `TukeyHSD()`. This function will calculate all of these pairwise intervals using the above formula, which is commonly known as Tukey's Honestly Significant Differences. This function expects to receive output from the `aov()` function. The `aov()` is similar to `lm()` but does not accept continuous covariates in the model.

```
m <- aov(coag ~ diet, data=coagulation)
TukeyHSD(m, level=.90)

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

Fit: aov(formula = coag ~ diet, data = coagulation)

$diet
    diff      lwr    upr  p adj
B-A    5   0.7246  9.275 0.0183
C-A    7   2.7246 11.275 0.0010
D-A    0  -4.0560  4.056 1.0000
C-B    2  -1.8241  5.824 0.4766
D-B   -5  -8.5771 -1.423 0.0044
D-C   -7 -10.5771 -3.423 0.0001
```

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.

Chapter 5

Two-Way Analysis of Variance

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):

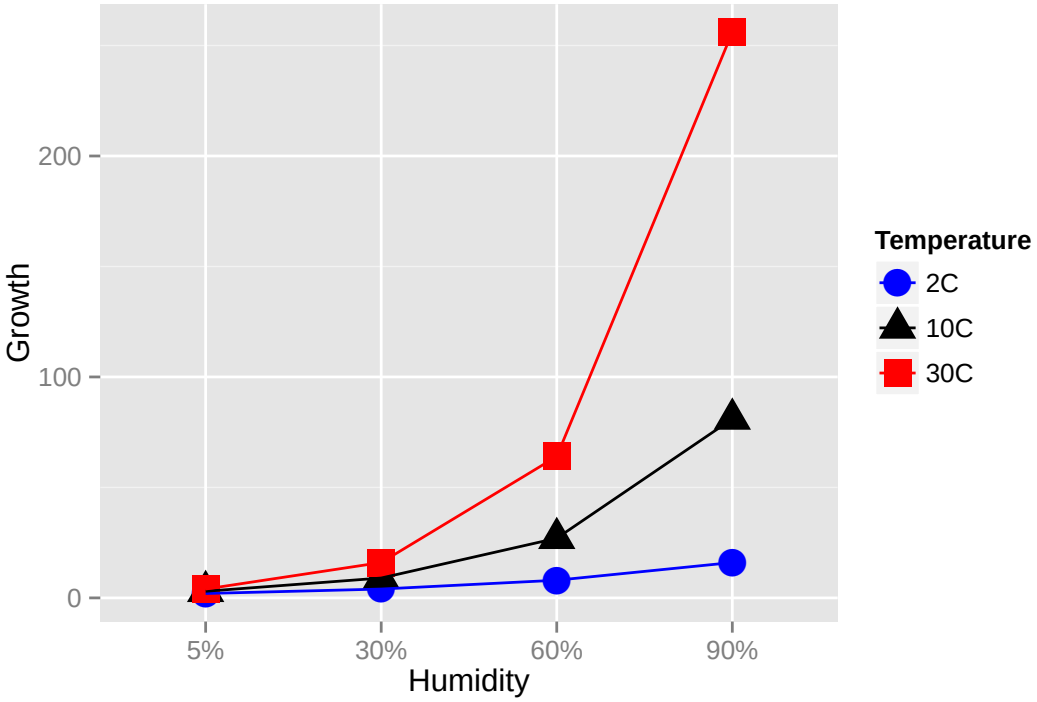
		Humidity			
		5%	30%	60%	90%
Temperature	2 C	2	4	8	16
	10 C	3	9	27	81
	30 C	4	16	64	256

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.[1]

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.

[1] 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.

A more scientific example can be found in metallurgy. Just heating and cooling iron does nothing to increase the strength of the material. Adding carbon to the iron increases the strength of it, but adding the carbon *and* using a tempering process results in a material with impressive strength.



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*.

5.1 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
Drug 1	0	10
Drug 2	6	0

Design 2	Males	Females
Drug 1	1	9
Drug 2	5	1

Design 3	Males	Females
Drug 1	3	5
Drug 2	3	5

Design 4	Males	Females
Drug 1	4	4
Drug 2	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.[2]

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).

5.2 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_i 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

Expected Values		Humidity			
		5%	30%	60%	90%
Temperature	2 C	$\mu+0+0$	$\mu+\alpha_2+0$	$\mu+\alpha_3+0$	$\mu+\alpha_4+0$
	10 C	$\mu+0+\beta_2$	$\mu+\alpha_2+\beta_2$	$\mu+\alpha_3+\beta_2$	$\mu+\alpha_4+\beta_2$
	30 C	$\mu+0+\beta_3$	$\mu+\alpha_2+\beta_3$	$\mu+\alpha_3+\beta_3$	$\mu+\alpha_4+\beta_3$

[2]We could show this by calculating the design matrix for each design and calculating the correlation coefficients between each of pairs of columns.

5.2.1 Example - Fruit Trees

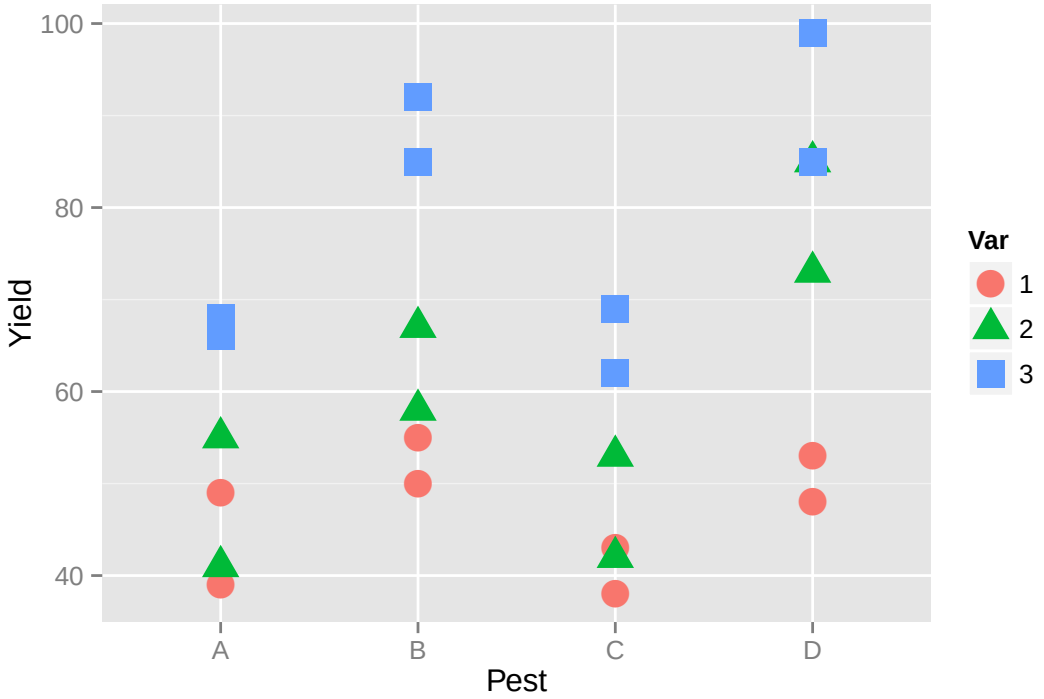
An experiment was conducted to determine the effects of four different pesticides on the yield of fruit from three different varieties of a citrus tree. Eight trees of each variety were randomly selected from an orchard. The four pesticides were randomly assigned to two trees of each variety and applications 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.

```
# type in the data by hand
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)
```

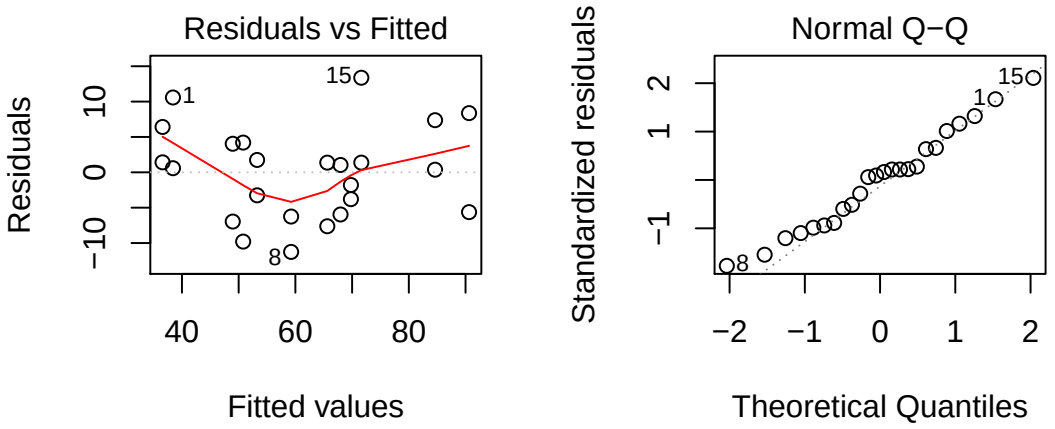
The first thing to do is to look at our data

```
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)
par(mfrow=c(1,2))  # two plots side-by-side
plot(m3, which=1)
plot(m3, which=2)
```



There might be a little curvature in the fitted vs residuals, but since 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
```

```
            Estimate Std. Error t value  Pr(>|t|)
(Intercept)    46.88      4.359  10.754 5.345e-10
Var2           12.37      6.164   2.008 5.773e-02
Var3           31.37      6.164   5.090 4.850e-05
```

```
summary(m2)$coef
```

```
            Estimate Std. Error t value  Pr(>|t|)
(Intercept)   53.000      6.429  8.2434 7.314e-08
PestB         14.833      9.093  1.6314 1.185e-01
PestC         -1.833      9.093 -0.2016 8.422e-01
PestD         20.833      9.093  2.2912 3.294e-02
```

```
summary(m3)$coef
```

```
            Estimate Std. Error t value  Pr(>|t|)
(Intercept)   38.417      3.660 10.4967 4.214e-09
Var2          12.375      3.660  3.3813 3.327e-03
Var3          31.375      3.660  8.5727 9.030e-08
PestB         14.833      4.226  3.5100 2.501e-03
PestC         -1.833      4.226 -0.4338 6.696e-01
PestD         20.833      4.226  4.9297 1.081e-04
```

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.

5.2.2 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:

	df	Sum Sq (SS)	Mean Sq (MS)	F	P-Value
A	$df_A = I - 1$	$SS_A = RSS_1 - RSS_A$	$MS_A = SS_A/df_A$	MS_A/MSE	$P\left(F_{df_A,df_\epsilon} > F_A\right.$
B	$df_B = J - 1$	$SS_B = RSS_A - RSS_{A+B}$	$MS_B = SS_B/df_B$	MS_B/MSE	$P\left(F_{df_B,df_\epsilon} > F_B\right.$
Error	$df_\epsilon = n - I - J + 1$	RSS_{A+B}	$MSE = RSS_{A+B}/df_\epsilon$		

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    1998    37.3 4.0e-07 ***
Pest       3   2227     742    13.9 6.3e-05 ***
Residuals 18    964      54
---
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
2     21 3192  2      3996 13.2  2e-04 ***
---
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 3192
2     18  964  3      2227 13.9 6.3e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5.2.3 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 `TukeyHSD`, which requires us to use the `aov()` function instead of `lm()`.

```
TukeyHSD(aov(Yield ~ Var + Pest, data=fruit))

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

Fit: aov(formula = Yield ~ Var + Pest, data = fruit)

$Var
     diff    lwr   upr  p adj
2-1 12.37  3.034 21.72 0.0089
3-1 31.37 22.034 40.72 0.0000
3-2 19.00  9.659 28.34 0.0002

$Pest
       diff     lwr    upr  p adj
B-A  14.833   2.889 26.777 0.0122
C-A  -1.833 -13.777 10.111 0.9719
D-A  20.833   8.889 32.777 0.0006
C-B -16.667 -28.611 -4.723 0.0048
D-B   6.000  -5.944 17.944 0.5038
D-C  22.667  10.723 34.611 0.0002
```

5.3 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

$E(\cdot)$		Humidity			
		5%	30%	60%	90%
Temp	2 C	$\mu+0+0+0$	$\mu+\alpha_2+0+0$	$\mu+\alpha_3+0+0$	$\mu+\alpha_4+0+0$
	10 C	$\mu+0+\beta_2+0$	$\mu+\alpha_2+\beta_2+(\alpha\beta)_{22}$	$\mu+\alpha_3+\beta_2+(\alpha\beta)_{32}$	$\mu+\alpha_4+\beta_2+(\alpha\beta)_{42}$
	30 C	$\mu+0+\beta_3+0$	$\mu+\alpha_2+\beta_3+(\alpha\beta)_{23}$	$\mu+\alpha_3+\beta_3+(\alpha\beta)_{33}$	$\mu+\alpha_4+\beta_3+(\alpha\beta)_{43}$

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 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.

5.3.1 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 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. 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)	Mean Sq (MS)	F	P-Value
A	$df_A = I-1$	$SS_A = RSS_1 - RSS_A$	$MS_A = SS_A/df_A$	MS_A/MSE	$P(F_{df_A,df_\epsilon} > F_A)$
B	$df_B = J-1$	$SS_B = RSS_A - RSS_{A+B}$	$MS_B = SS_B/df_B$	MS_B/MSE	$P(F_{df_B,df_\epsilon} > F_B)$
AB	$(I-1)(J-1)$	$RSS_{A+B} - RSS_{A*B}$	$MS_{AB} = SS_{AB}/df_{AB}$	MS_{AB}/MSE	$P(F_{df_{AB},df_\epsilon} > F_{AB})$
Error	$df_\epsilon = n-IJ$	RSS_{A*B}	$MSE = RSS_{A*B}/df_\epsilon$		

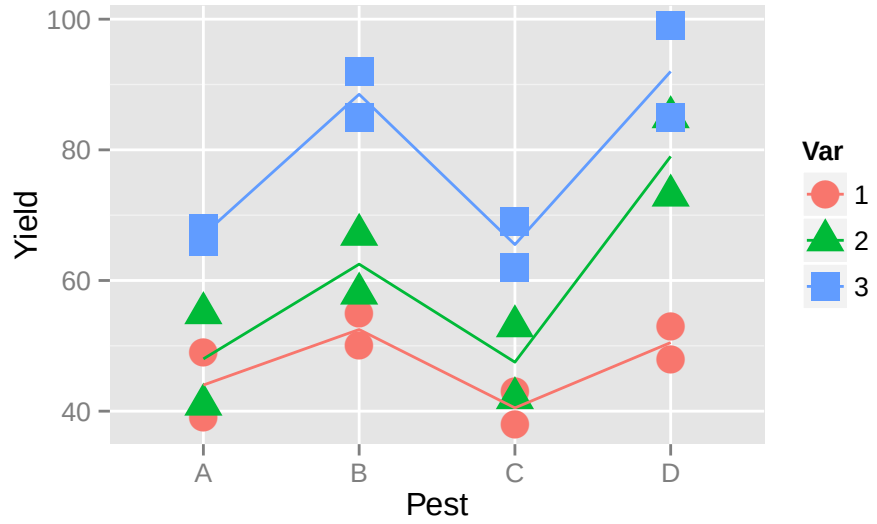
This arrangement of the ANOVA table is referred to as "Type I" sum of squares.[3]

5.3.1.1 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.

[3]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`.

```
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    1998    47.2   2e-06 ***
Pest       3   2227     742    17.6 0.00011 ***
Var:Pest   6    457      76     1.8 0.18168
Residuals 12    508      42
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

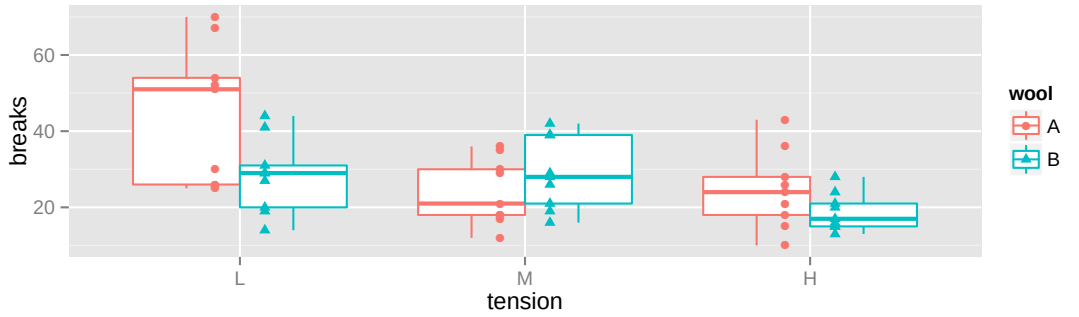
Examining the ANOVA table, we see that the interaction effect is not significant and we will stay with simpler model `Yield~Var+Pest`.

5.3.1.2 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.

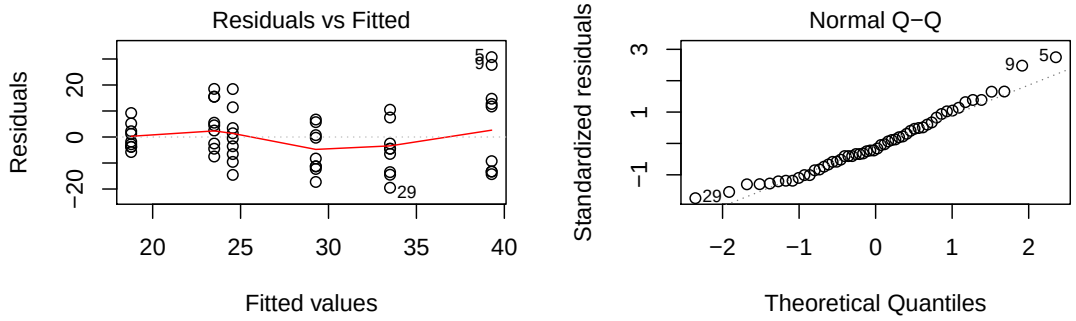
As always, the first thing we do is look at the data. In this case, it looks like the number of breaks decreases with increasing tension and perhaps wool B has fewer breaks than wool A.

```
library(ggplot2)
library(faraway)
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
```



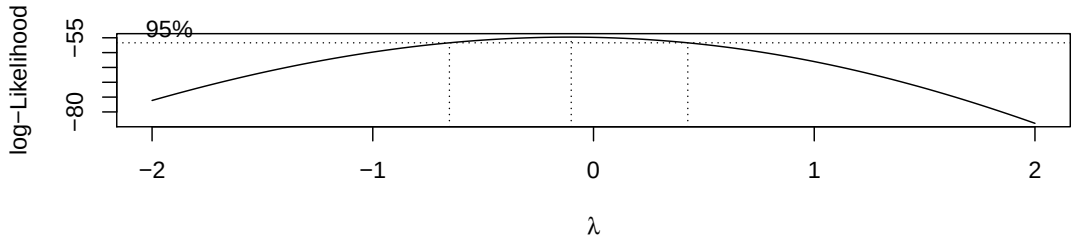
We next fit our linear model and examine the diagnostic plots.

```
model <- lm(breaks ~ tension + wool, data=warpbreaks)
par(mfrow=c(1,2)) # side-by-side plots
plot(model, which=c(1,2))
```



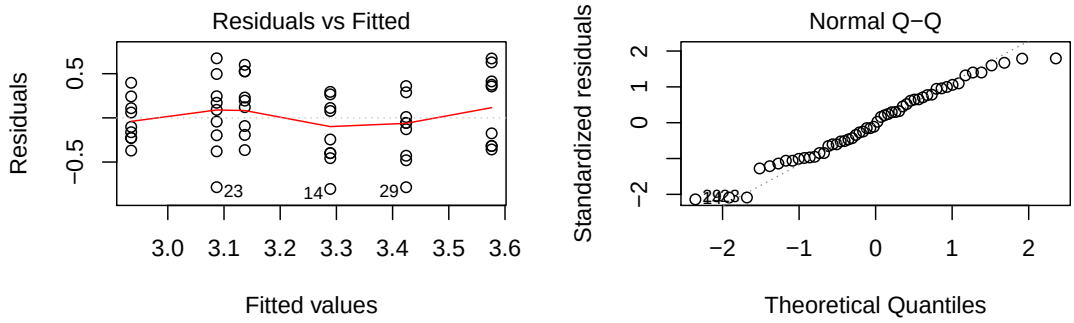
The residuals vs fitted values plot is a little worrisome and appears to be an issue with non-constant variance, but the normality assumption looks good. We'll check for a Box-Cox transformation next.

```
library(MASS)
boxcox(model)
```



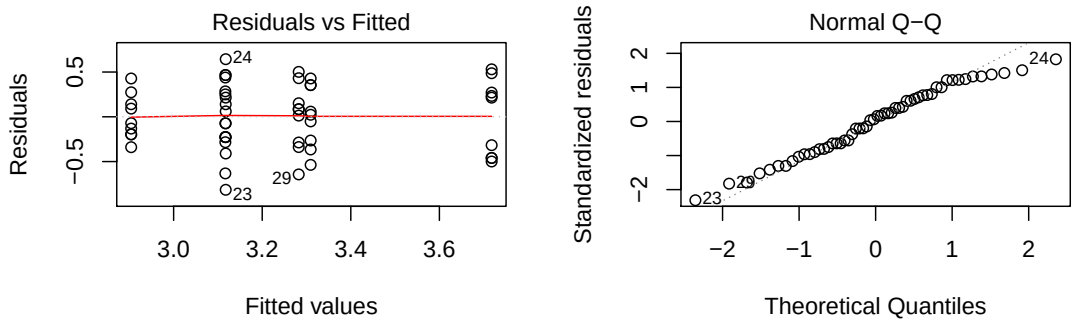
This suggests we should make a log transformation, though because the confidence interval is quite wide we might consider if the increased difficulty in interpretation makes sufficient progress towards making the data meet the model assumptions.. The diagnostic plots of the resulting model look better 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)
par(mfrow=c(1,2)) # make side-by-side plots
plot(model.1, which=c(1,2))
```



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)
par(mfrow=c(1,2)) # side-by-side plots
plot(model.2, which=c(1,2))
```

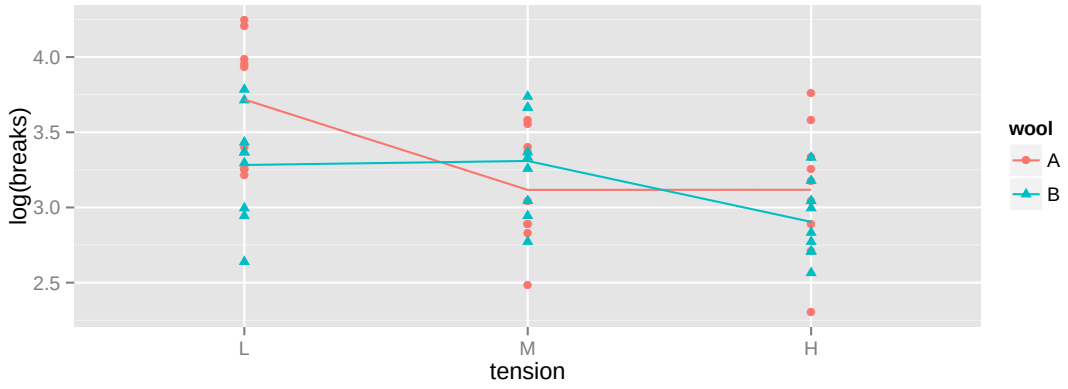


Then we'll do an F-test to see if it is a better model than the main effects model. The p-value is marginally significant, so we'll keep the interaction in the model, but recognize that it is a weak interaction.

```
anova(model.1, model.2)


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 7.63
2     48 6.71  2     0.913 3.26  0.047 *
---
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$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)))
```



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
-0.8150 -0.2789  0.0404  0.2732  0.6436

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     3.718      0.125   29.82   <2e-16 ***
tensionM       -0.601      0.176   -3.41   0.0013 **
tensionH       -0.600      0.176   -3.41   0.0013 **
woolB          -0.436      0.176   -2.47   0.0171 *
tensionM:woolB  0.628      0.249    2.52   0.0151 *
tensionH:woolB  0.222      0.249    0.89   0.3775
---
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:  0.336,Adjusted R-squared:  0.267
F-statistic: 4.86 on 5 and 48 DF,  p-value: 0.00112
```

$E(\cdot)$		Tension		
		Low	Medium	High
Wool Type	A	$\mu+0+0+0$	$\mu+\alpha_2+0+0$	$\mu+\alpha_3+0+0$
	B	$\mu+0+\beta_2+0$	$\mu+\alpha_2+\beta_2+(\alpha\beta)_{22}$	$\mu+\alpha_3+\beta_2+(\alpha\beta)_{32}$

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

$$\begin{aligned} H_0: \ (\mu+\alpha_2+\beta_2+(\alpha\beta)_{22})-(\mu+\alpha_3+\beta_2+(\alpha\beta)_{32}) &= 0 \\ H_a: \ (\mu+\alpha_2+\beta_2+(\alpha\beta)_{22})-(\mu+\alpha_3+\beta_2+(\alpha\beta)_{32}) &\neq 0 \end{aligned}$$

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 $-.6012+.6003+.6281-.2221=0.4051$. But 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 Tukey's HSD. Fortunately, we can do the same thing here. In this case, we'll look at the interactions piece of the TukeyHSD command. In this case, we find the test H:B - M:B in the last row of the interactions.

```
model.2 <- aov(log(breaks) ~ tension * wool, data=warpbreaks)
TukeyHSD(model.2)


  Tukey multiple comparisons of means
```

```
    95% family-wise confidence level

Fit: aov(formula = log(breaks) ~ tension * wool, data = warpbreaks)

$tension
       diff     lwr      upr  p adj
M-L -0.2871 -0.5886  0.01438 0.0649
H-L -0.4893 -0.7908 -0.18777 0.0008
H-M -0.2022 -0.5037  0.09935 0.2465

$wool
       diff     lwr     upr  p adj
B-A -0.1522 -0.3568 0.05251 0.1415

$`tension:wool`
             diff     lwr      upr  p adj
M:A-L:A -0.601196 -1.1244 -0.07795 0.0158
H:A-L:A -0.600323 -1.1236 -0.07708 0.0160
L:B-L:A -0.435567 -0.9588  0.08768 0.1535
M:B-L:A -0.408619 -0.9319  0.11463 0.2071
H:B-L:A -0.813794 -1.3370 -0.29055 0.0004
H:A-M:A  0.000873 -0.5224  0.52412 1.0000
L:B-M:A  0.165629 -0.3576  0.68888 0.9341
M:B-M:A  0.192577 -0.3307  0.71582 0.8821
H:B-M:A -0.212598 -0.7358  0.31065 0.8319
L:B-H:A  0.164756 -0.3585  0.68800 0.9355
M:B-H:A  0.191704 -0.3315  0.71495 0.8840
H:B-H:A -0.213471 -0.7367  0.30978 0.8295
M:B-L:B  0.026948 -0.4963  0.55020 1.0000
H:B-L:B -0.378227 -0.9015  0.14502 0.2823
H:B-M:B -0.405175 -0.9284  0.11807 0.2149
```

So `TukeyHSD` gives us all the tests comparing the cell means, but what are those main effects testing? 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.

Using `TukeyHSD`, 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:

```
library(dplyr)
foo <- warpbreaks %.% group_by(wool) %.% summarise( mean(log(breaks)) )
foo

Source: local data frame [2 x 2]

  wool mean(log(breaks))
1    A             3.317
2    B             3.165
```

```
foo[2,2] - foo[1,2]


[1] -0.1522
```

In the unbalanced case taking the average of the cell means produces a different answer than taking the average of the data. It isn't clear which is preferred and `TukeyHSD` produces a result that is between those two methods.

Chapter 6

Analysis of Covariance (ANCOVA)

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.*

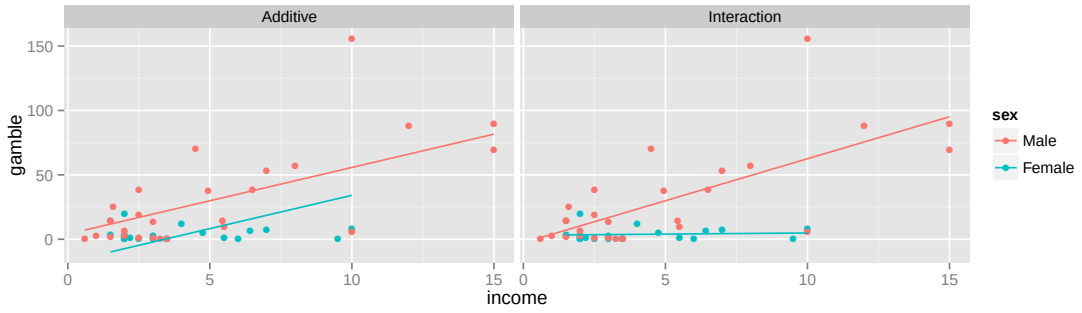
The dataset `teengamb` in the package `faraway` has data regarding the rates of gambling among teenagers in Britain and their gender and socioeconomic status. One question we might be interested in is how gender and income relate to how much a person gambles. But what should be the effect of gender be?

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.

```
library(ggplot2)
library(faraway)
teengamb$sex <- factor(teengamb$sex, labels=c('Male','Female'))
m1 <- lm( gamble ~ sex + income, data=teengamb )
m2 <- lm( gamble ~ sex * income, data=teengamb )
yhat.1 <- predict(m1)
yhat.2 <- predict(m2)
temp <- rbind( cbind(teengamb,yhat=yhat.1,method='Additive'),
               cbind(teengamb,yhat=yhat.2,method='Interaction') )
ggplot(temp, aes(x=income, col=sex)) +
      geom_point(aes(y=gamble)) +
      geom_line(aes(y=yhat)) +
      facet_wrap(~ method)
```



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 $\boldsymbol{X}$.

6.1 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. Since 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 6 observations with three observations per group where $y_{i,j}$ is the jth replication of the ith group.

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

6.2 Lines with different slopes (aka an interaction between between discrete and continuous covariates)

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.

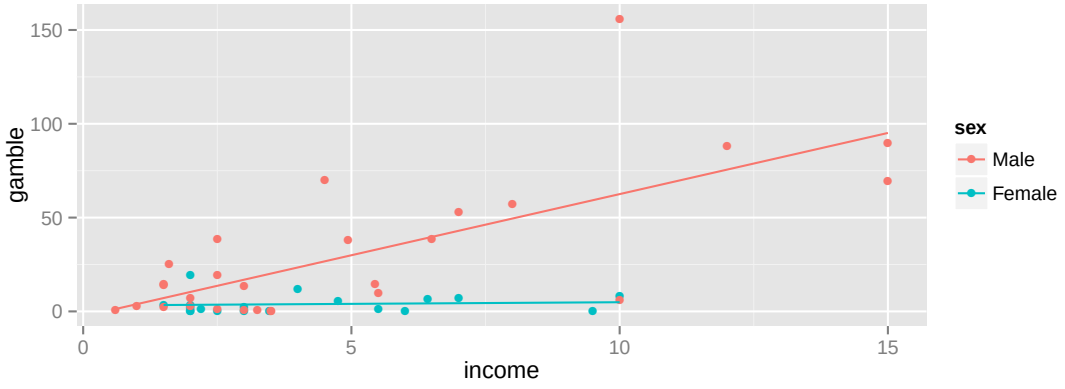
$$y_i = \begin{cases} (\beta_0 + \beta_1) + (\beta_2 + \beta_3)\, x_i + \epsilon_i & \text{if female} \\ \beta_0 + \beta_2 x_i + \epsilon_i & \text{if male} \end{cases}$$

and we have β_1 is the offset in y-intercept of the female group from the male group, while 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

```
library(ggplot2)
library(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 )

# Make a nice plot of the model
teengamb$gamble.hat <- predict(m1)
ggplot(teengamb, aes(x=income, col=sex)) +
      geom_point(aes(y=gamble)) +
      geom_line(aes(y=gamble.hat))
```




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


Call:
lm(formula = gamble ~ sex * income, data = teengamb)

Residuals:
   Min     1Q Median     3Q    Max
-56.52  -4.86  -1.79   6.27  93.48

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       -2.660      6.316   -0.42    0.676
sexFemale          5.800     11.200    0.52    0.607
income             6.518      0.988    6.60    5e-08 ***
sexFemale:income  -6.343      2.145   -2.96    0.005 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21 on 43 degrees of freedom
Multiple R-squared:  0.586,Adjusted R-squared:  0.557
F-statistic: 20.3 on 3 and 43 DF,  p-value: 2.45e-08
```

6.3 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`.

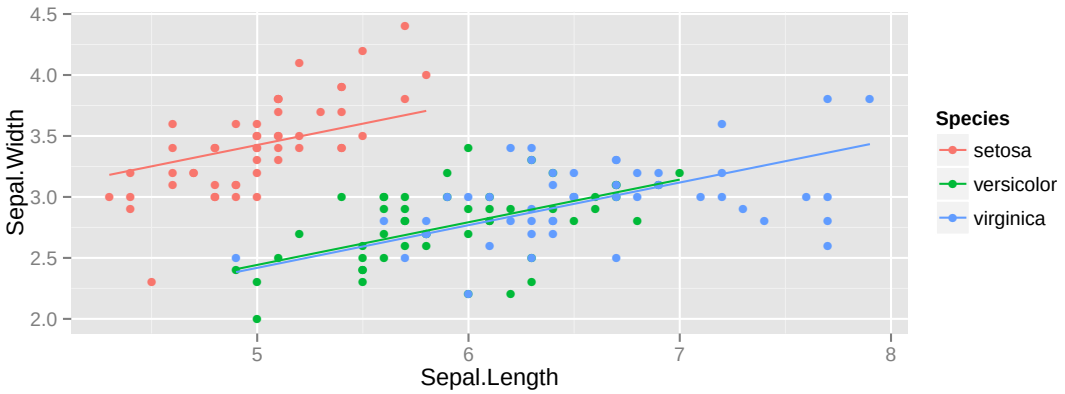
```
data(iris)            # read in the iris dataset
levels(iris$Species)  # notice the order of levels of Species
```

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

```
# Now fit the linear model with offset lines
m <- lm( Sepal.Width ~ Sepal.Length + Species, data=iris)
iris$Sepal.Width.hat <- predict(m)

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

```

Call:
lm(formula = Sepal.Width ~ Sepal.Length + Species, data = iris)

Residuals:
    Min      1Q  Median      3Q     Max
-0.9510 -0.1652  0.0017  0.1842  0.7292

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         1.6765     0.2354    7.12  4.5e-11 ***
Sepal.Length        0.3499     0.0463    7.56  4.2e-12 ***
Speciesversicolor  -0.9834     0.0721  -13.64  < 2e-16 ***
Speciesvirginica   -1.0075     0.0933  -10.80  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.289 on 146 degrees of freedom
Multiple R-squared:  0.569,Adjusted R-squared:  0.56
F-statistic: 64.3 on 3 and 146 DF,  p-value: <2e-16
```

```
library(ggplot2)      # a plotting package
# Now make a plot of the data and regression lines
ggplot(iris, aes(x=Sepal.Length, col=Species)) +
       geom_point(aes(y=Sepal.Width)) +
       geom_line(aes(y=Sepal.Width.hat))
```



The significant p-values for `Speciesversicolor` and for `Speciesvirginica` tell us that those two y-intercepts are significantly different from that of the reference group `setosa`, but looking at the graph, we are interested in if the y-intercept for `versicolor` is different than that of `virginica`. By far the easiest way to do this in R is to change which level is the reference group.

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

# Rerun the linear model
m <- lm( Sepal.Width ~ Sepal.Length + Species, data=iris)
summary(m)


Call:
lm(formula = Sepal.Width ~ Sepal.Length + Species, data = iris)

Residuals:
    Min      1Q  Median      3Q     Max
-0.9510 -0.1652  0.0017  0.1842  0.7292

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.6690     0.3078    2.17    0.031 *
Sepal.Length        0.3499     0.0463    7.56  4.2e-12 ***
Speciessetosa       1.0075     0.0933   10.80  < 2e-16 ***
Speciesversicolor   0.0241     0.0652    0.37    0.712
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.289 on 146 degrees of freedom
Multiple R-squared:  0.569,Adjusted R-squared:  0.56
F-statistic: 64.3 on 3 and 146 DF,  p-value: <2e-16
```

To conclude the `iris` example, we can look at the following models

```
library(ggplot2)
data(iris)

# make virginica the reference group
iris$Species <- relevel(iris$Species, ref='virginica')
m1 <- lm( Sepal.Width ~ Sepal.Length, data=iris )
m2 <- lm( Sepal.Width ~ Sepal.Length + Species, data=iris )
m3 <- lm( Sepal.Width ~ Sepal.Length * Species, data=iris )
summary(m1)$coefficients  # just get the table of coefficients
```

```
             Estimate Std. Error t value  Pr(>|t|)
(Intercept)   3.41895    0.25356   13.48 1.552e-27
Sepal.Length -0.06188    0.04297   -1.44 1.519e-01
```

```
summary(m2)$coefficients  # just get the table of coefficients
```

```
                  Estimate Std. Error t value  Pr(>|t|)
(Intercept)        0.66899    0.30776  2.1737 3.134e-02
Sepal.Length       0.34988    0.04630  7.5566 4.187e-12
Speciessetosa      1.00751    0.09331 10.7980 2.407e-20
Speciesversicolor  0.02412    0.06521  0.3699 7.120e-01
```

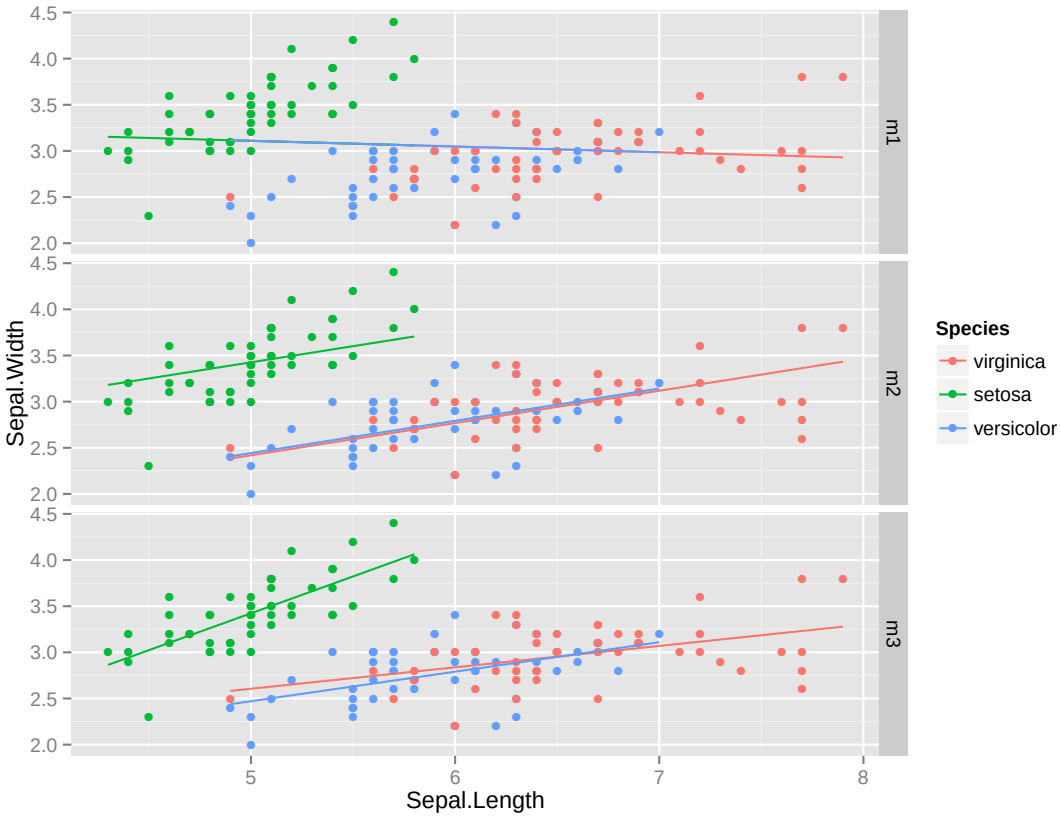
```
summary(m3)$coefficients  # just get the table of coefficients
```

```
                              Estimate Std. Error t value  Pr(>|t|)
(Intercept)                    1.44631    0.40491  3.5719 4.824e-04
Sepal.Length                   0.23189    0.06118  3.7901 2.207e-04
Speciessetosa                 -2.01574    0.68609 -2.9380 3.848e-03
Speciesversicolor             -0.57416    0.60466 -0.9496 3.439e-01
Sepal.Length:Speciessetosa     0.56664    0.12620  4.4902 1.448e-05
Sepal.Length:Speciesversicolor 0.08783    0.09708  0.9047 3.671e-01
```

```
# It looks like m3 is a better model, but check using an F-test
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 12.2
2    144 10.7  2      1.51 10.2 7.2e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
library(ggplot2)
library(faraway)
yhat.1 <- predict(m1)
yhat.2 <- predict(m2)
yhat.3 <- predict(m3)
temp <- rbind(
cbind(iris, yhat=yhat.1, method='m1'),
cbind(iris, yhat=yhat.2, method='m2'),
cbind(iris, yhat=yhat.3, method='m3'))
p <- ggplot(temp, aes(x=Sepal.Length, col=Species)) +
       geom_point(aes(y=Sepal.Width)) +
       geom_line(aes(y=yhat)) +
       facet_grid(method ~ .)
print(p)
```



Chapter 7

Contrasts

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 $\mu + \tau_2$.

7.1 Estimate and variance of a contrast

A *contrast* is a linear combinations of elements of $\boldsymbol{\beta}$, which is a fancy way of saying that it is a function of the elements of $\boldsymbol{\beta}$ 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 $\boldsymbol{c}^T\boldsymbol{\beta}$.

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 $\boldsymbol{c}^T\hat{\boldsymbol{\beta}}$.

Since 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.

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

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

$$\begin{aligned} Est &\pm t_{n-p}^{1-\alpha/2}\, StdErr\,(\,Est\,) \\ \boldsymbol{c}^T\hat{\boldsymbol{\beta}} &\pm t_{n-p}^{1-\alpha/2}\,\hat{\sigma}\sqrt{\boldsymbol{c}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{c}} \end{aligned}$$

Example: Hostility scores

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 $\mu + \tau_3$.

```
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)
coef(m)

 (Intercept) groupsGroup2 groupsGroup3
       86.75       -11.18       -15.75

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,] 75.57

std.err

      [,1]
[1,] 1.623
```

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    86.75      1.518   57.14 1.567e-24
groupsGroup2    75.57      1.623   46.56 1.117e-22
groupsGroup3    71.00      1.431   49.60 2.993e-23
```

7.2 Estimating contrasts in R

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 **g**eneralized **l**inear **h**ypothesis **t**est 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.

Example: 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)     86.75      1.518  57.141 1.567e-24
groupsGroup2   -11.18      2.222  -5.030 5.584e-05
groupsGroup3   -15.75      2.087  -7.548 2.064e-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    86.75       1.52    57.1   <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 $\mu + \tau_3$. 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) )
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    86.75       1.52    57.1   <2e-16 ***
Mean of Group 2 == 0    75.57       1.62    46.6   <2e-16 ***
Mean of Group 3 == 0    71.00       1.43    49.6   <2e-16 ***
---
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.

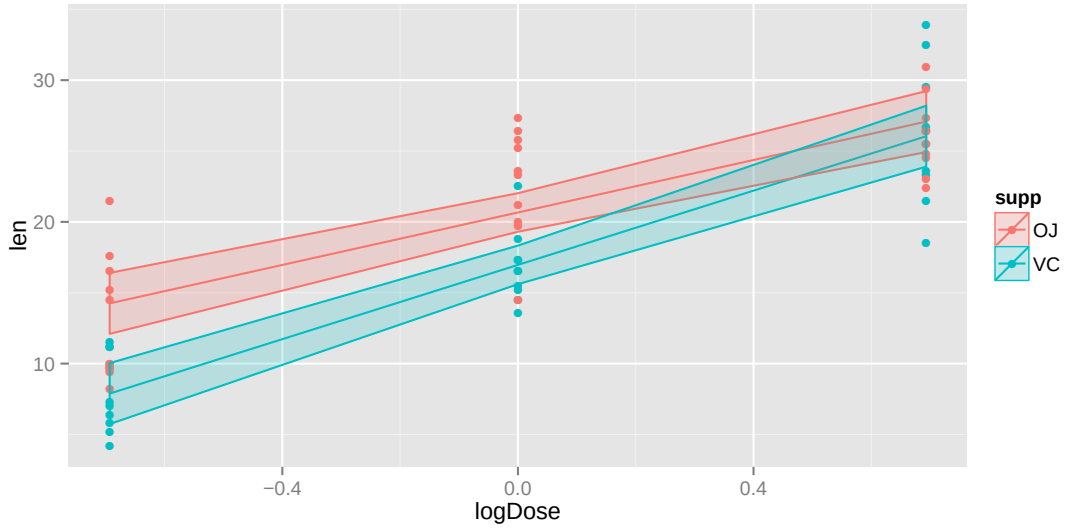
```
confint(test, level=0.95)


 Simultaneous Confidence Intervals

Fit: lm(formula = y ~ groups)

Quantile = 2.583
95% family-wise confidence level


Linear Hypotheses:
                     Estimate lwr    upr
Mean of Group 1 == 0 86.750   82.829 90.671
Mean of Group 2 == 0 75.571   71.379 79.764
Mean of Group 3 == 0 71.000   67.303 74.697
```

Example: Tooth growth

In the ANCOVA case, there are more interesting contrasts to be made. For this example we will use the `ToothGrowth` dataset from the `faraway` package. 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.

```
library(ggplot2)
library(faraway)
data(ToothGrowth)
ToothGrowth$logDose <- log( ToothGrowth$dose )
m <- lm(len ~ logDose * supp, data=ToothGrowth)
ToothGrowth$len.hat <- predict(m, interval='confidence')[,1]
ToothGrowth$len.lwr <- predict(m, interval='confidence')[,2]
ToothGrowth$len.upr <- predict(m, interval='confidence')[,3]
ggplot( ToothGrowth, aes(x=logDose, col=supp, fill=supp)) +
    geom_point(aes(y=len)) +
    geom_line(aes(y=len.hat)) +
    geom_ribbon(aes(ymin=len.lwr, ymax=len.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)      20.663     0.6791  30.425 1.629e-36
logDose           9.255     1.2000   7.712 2.303e-10
suppVC           -3.700     0.9605  -3.852 3.033e-04
logDose:suppVC    3.845     1.6971   2.266 2.737e-02
```

First 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 $\beta_1 + \beta_3$.

```
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   20.663      0.679   30.43   <1e-09 ***
Slope     OJ == 0    9.255      1.200    7.71   <1e-09 ***
Intercept VC == 0   16.963      0.679   24.98   <1e-09 ***
Slope     VC == 0   13.100      1.200   10.92   <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

$$\begin{aligned} y_{oj,dose=0.5} &= \begin{bmatrix} 1 & -0.69 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{bmatrix} \\ &= \boldsymbol{c}_1^T \hat{\boldsymbol{\beta}} \end{aligned}$$

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

$$\begin{aligned} \hat{y}_{vc,dose=0.5} &= 16.9633 + 13.0997 \log(0.5) \\ &= (20.66 - 3.7) + (9.25 + 3.8) \log(0.5) \\ &= \left(\hat{\beta}_0 + \hat{\beta}_2\right) + \left(\hat{\beta}_1 + \hat{\beta}_4\right) (-0.69) \\ &= \begin{bmatrix} 1 & -0.69 & 1 & -0.69 \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{bmatrix} \\ &= \boldsymbol{c}_2^T \hat{\boldsymbol{\beta}} \end{aligned}$$

```
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    14.28       1.07    13.3  < 1e-10 ***
VC; 1/2 dose == 0     7.92       1.07     7.4  1.5e-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     6.35       1.51    4.19  9.8e-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

```
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     1.05       1.51    0.69     0.49
(Adjusted p values reported -- single-step method)
```

Chapter 8

Diagnostics

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 $\boldsymbol{\epsilon} \sim MVN\left(\boldsymbol{0}, \sigma^2\boldsymbol{I}\right)$ but autocorrelation, heteroscedasticity, and non-normality might be present.

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

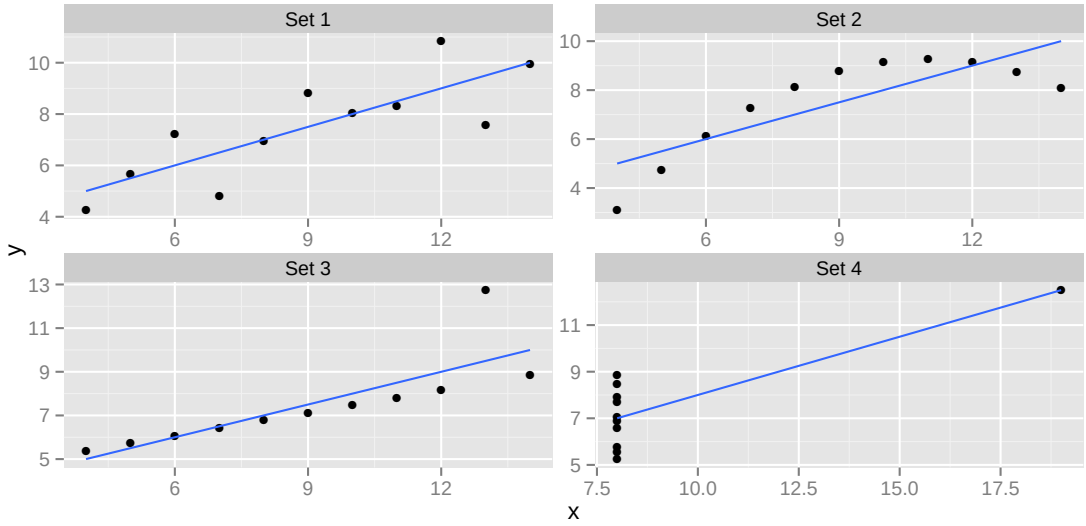
$$\hat{y} = 3 + 0.5\,x$$

but the data are drastically different.

```
library(ggplot2)
str(anscombe)
```

```
'data.frame': 11 obs. of  8 variables:
 $ x1: num  10 8 13 9 11 14 6 4 12 7 ...
 $ x2: num  10 8 13 9 11 14 6 4 12 7 ...
 $ x3: num  10 8 13 9 11 14 6 4 12 7 ...
 $ x4: num  8 8 8 8 8 8 8 19 8 8 ...
 $ y1: num  8.04 6.95 7.58 8.81 8.33 ...
 $ y2: num  9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...
 $ y3: num  7.46 6.77 12.74 7.11 7.81 ...
 $ y4: num  6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 ...
```

```
attach(anscombe)  # So I can just refer to x1, x2, instead of anscombe$x1
data <- data.frame( x=c(x1,x2,x3,x4), y=c(y1,y2,y3,y4),
                    set=paste('Set', rep(1:4, each=11)) )
data$set <- factor(data$set)
str(data)

'data.frame': 44 obs. of  3 variables:
 $ x  : num  10 8 13 9 11 14 6 4 12 7 ...
 $ y  : num  8.04 6.95 7.58 8.81 8.33 ...
 $ set: Factor w/ 4 levels "Set 1","Set 2",..: 1 1 1 1 1 1 1 1 1 1 ...

p <- ggplot(data, aes(x=x, y=y)) + geom_point() +
facet_wrap(~set, scales='free') +
stat_smooth(method="lm", formula=y~x, se=FALSE)
print(p)
```



8.1 Measures of Influence

8.1.1 Studentized Residuals

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 $\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$ but the elements of $\boldsymbol{H}$ can be quite useful in diagnostics. It can be shown that the variance of the ith 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 $N(0,1)$ sample. These rescaled residuals are called "studentized residuals". Since we have a good intuition about the scale of a standard normal distribution, the scale of studentized residuals will give a good indicator if normality is violated.

Many of the built in diagnostic plots in R will make use of the studentized residuals[1], but R labels them as "standardized".

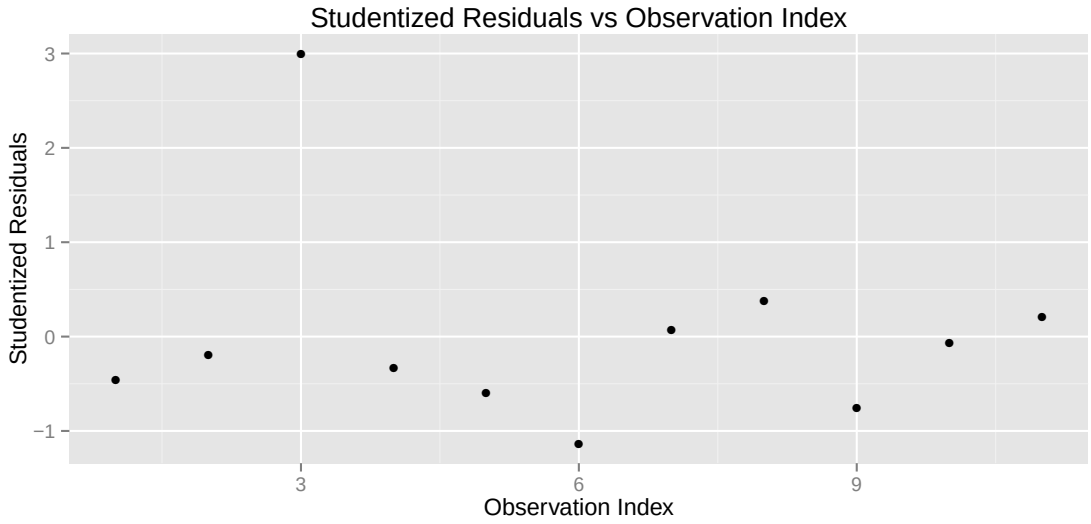
Example - Anscombe's set 3

```
# define a function to create studentized residuals
s.resid <- function(m){
X <- model.matrix(m)
H <- X %*% solve( t(X) %*% X ) %*% t(X)
sigma.hat <- summary(m)$sigma
return( resid(m) / (sigma.hat*sqrt(1-diag(H))) )
}
```

For the third dataset, the outlier is the third observation with $x_3 = 13$ and $y_3 = 12.74$. We calculate and plot the studentized residuals

```
model <- lm(y3 ~ x3, data=anscombe)
s.resids <- s.resid(model)
s.resids

      1       2       3       4       5       6       7       8
-0.46018 -0.19633  2.99999 -0.33085 -0.59695 -1.13497  0.07042  0.38070
      9      10      11
-0.75518 -0.06974  0.21188

rstandard(model)

      1       2       3       4       5       6       7       8
-0.46018 -0.19633  2.99999 -0.33085 -0.59695 -1.13497  0.07042  0.38070
      9      10      11
-0.75518 -0.06974  0.21188

rstudent(model)

        1         2         3         4         5         6
 -0.43906  -0.18550 1203.53946  -0.31384  -0.57429  -1.15598
        7         8         9        10        11
  0.06641   0.36185  -0.73568  -0.06577   0.20026

qplot(x=1:11, y=s.resids,
      xlab='Observation Index',
      ylab='Studentized Residuals',
      main='Studentized Residuals vs Observation Index')
```

[1]There are actually two types of studentized residuals, typically called 'internal' and 'external'. 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.



and we notice that the outlier residual is *really* big. If the model assumptions were true, then the studentized 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.

8.1.2 Leverage

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

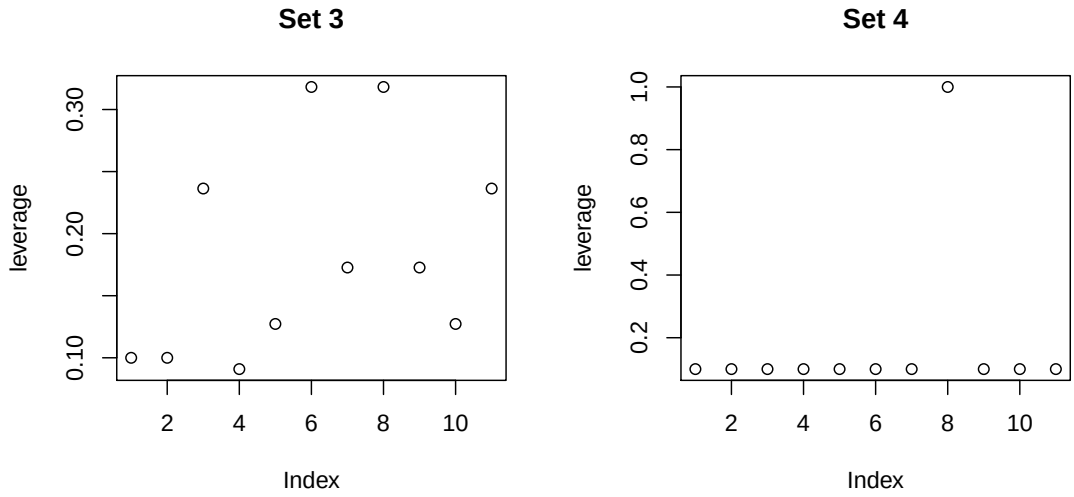
One way to quantify this is to look at the elements of $\boldsymbol{H}$. Since

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

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.

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.

```
model3 <- lm(y3 ~ x3, data=anscombe)
model4 <- lm(y4 ~ x4, data=anscombe)
par(mfrow=c(1,2))
plot(hatvalues(model3), ylab='leverage', main='Set 3')
plot(hatvalues(model4), ylab='leverage', main='Set 4')
```



So 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.

8.1.3 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

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

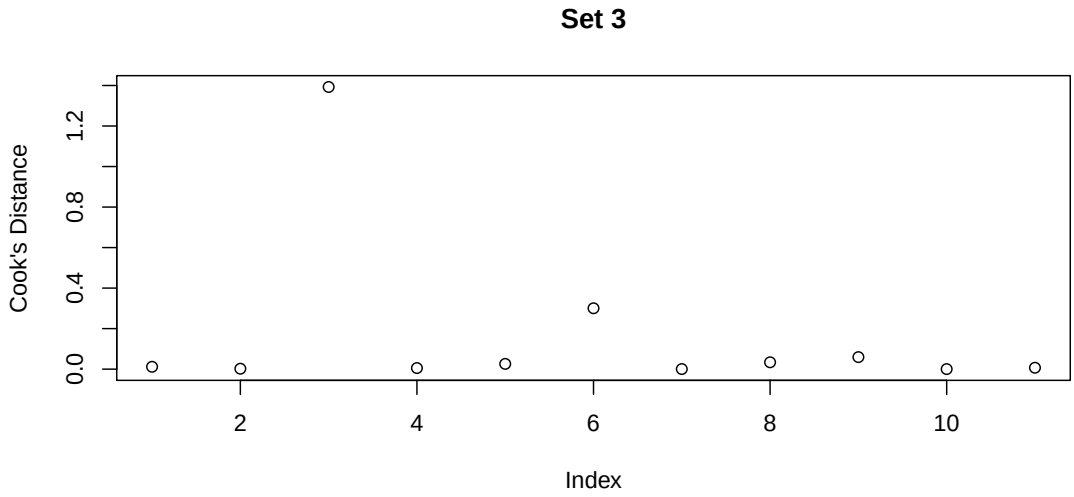
which boils down to saying if the predicted values have large changes when the ith element is removed, then the distance is big. It can be shown that this formula can be simplified to

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

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.

```
model3 <- lm(y3 ~ x3, data=anscombe)
plot(cooks.distance(model3), ylab="Cook's Distance", main='Set 3')
```



8.2 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)`.

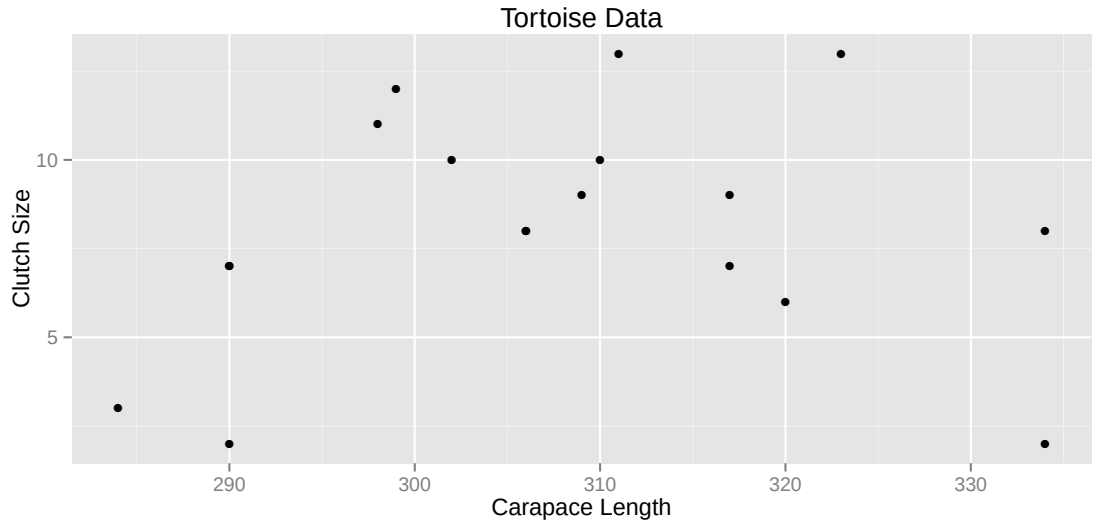
8.2.1 Residuals vs Fitted

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.

8.2.1.1 Polynomial relationships

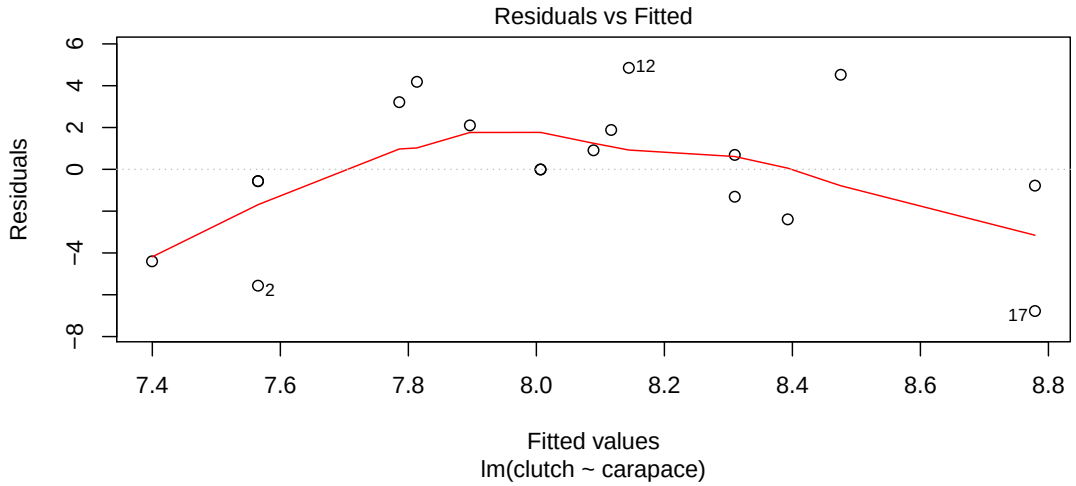
To explore how this plot can detect non-linear relationships between y and x, we will examine a data set from Ashton et al. (2007) that relates the length of a tortoise's carapace to the number of eggs laid in a clutch. The data are

```
carapace <-
c(284,290,290,290,298,299,302,306,306,309,310,311,317,317,320,323,334,334)
clutch   <- c(3,2,7,7,11,12,10,8,8,9,10,13,7,9,6,13,2,8)
qplot(x=carapace, y=clutch,
      main="Tortoise Data", xlab='Carapace Length', ylab='Clutch Size')
```



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 ~ carapace )
plot(model, which=1) # which=1 tells R to only make the first plot
```



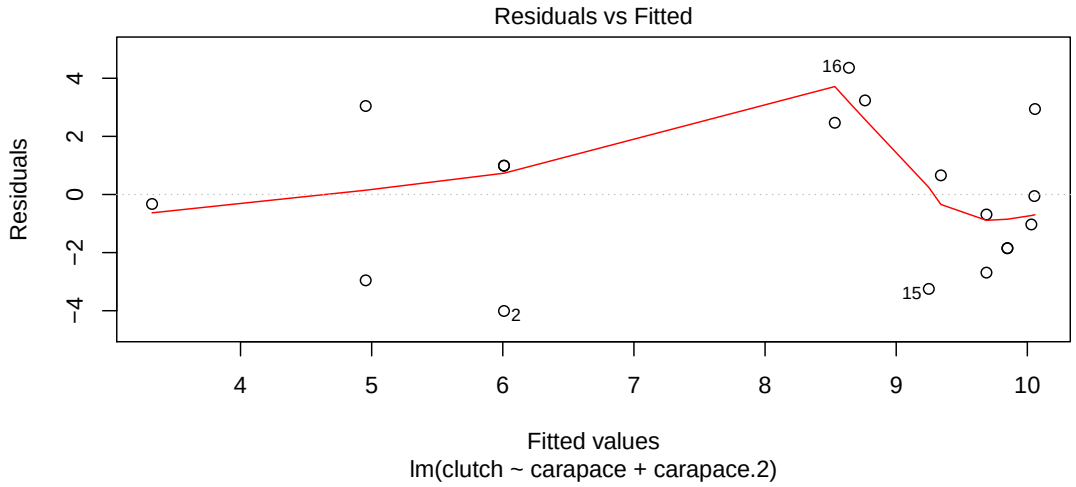
The red line going through the plot is a smoother of the data. Ideally this should be a flat line and I should see no trend in this plot. Clearly there is a quadratic trend as larger tortoises have larger clutch sizes until some point where the extremely large tortoises start laying fewer (perhaps the 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

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

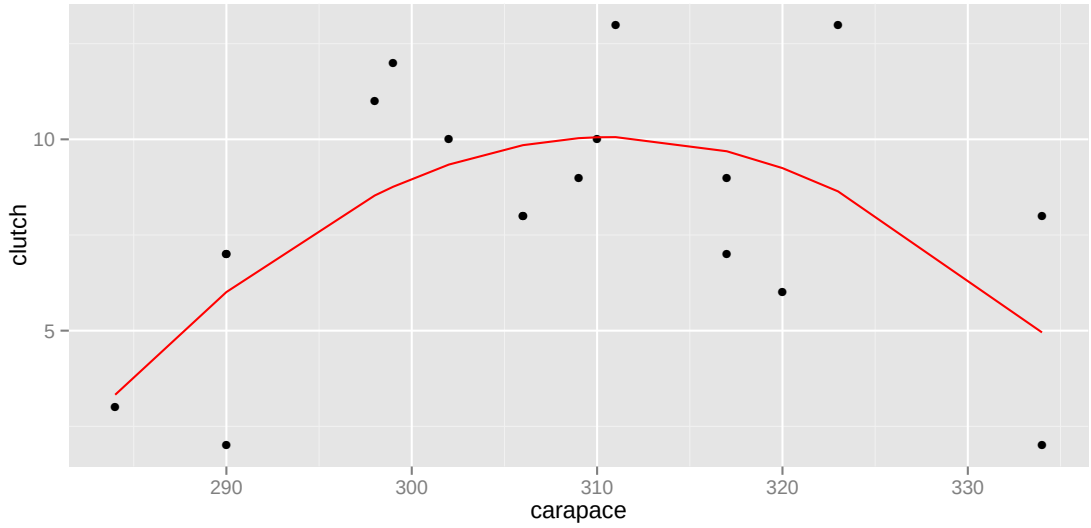
and note that my model is linear model with respect to covariates $\boldsymbol{x}$ and $\boldsymbol{x}^2$.

```
carapace.2 <- carapace^2
model <- lm( clutch ~ carapace + carapace.2 )
plot(model, which=1) # which=1 tells R to only make the first plot
```



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.

```
data <- data.frame(clutch=clutch, carapace=carapace, yhat=fitted(model))
ggplot(data, aes(x=carapace)) +
  geom_point(aes(y=clutch)) +
  geom_line(aes(y=yhat), color='red')
```



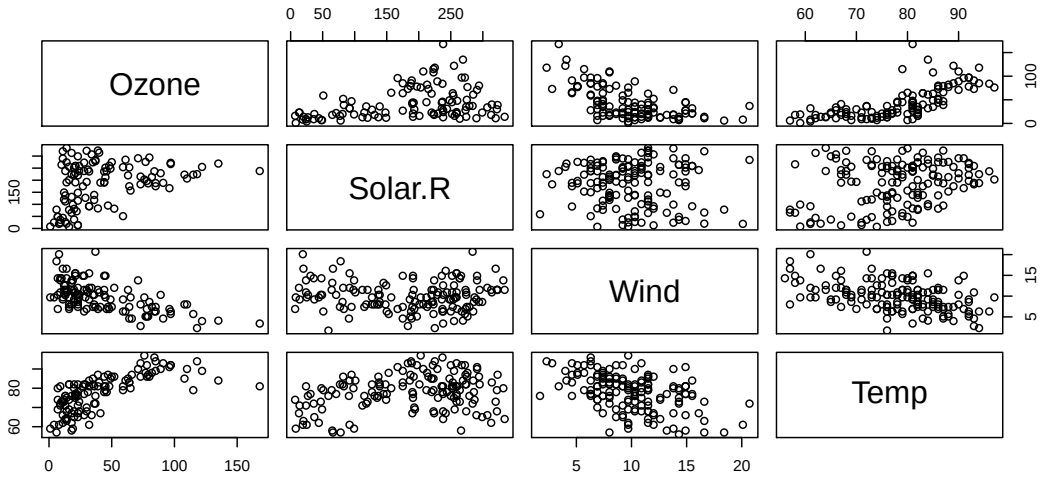
8.2.1.2 Heteroskedasticity

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 windspeed for 153 consecutive days in New York. The goal is to predict the level of ozone using the weather variables.

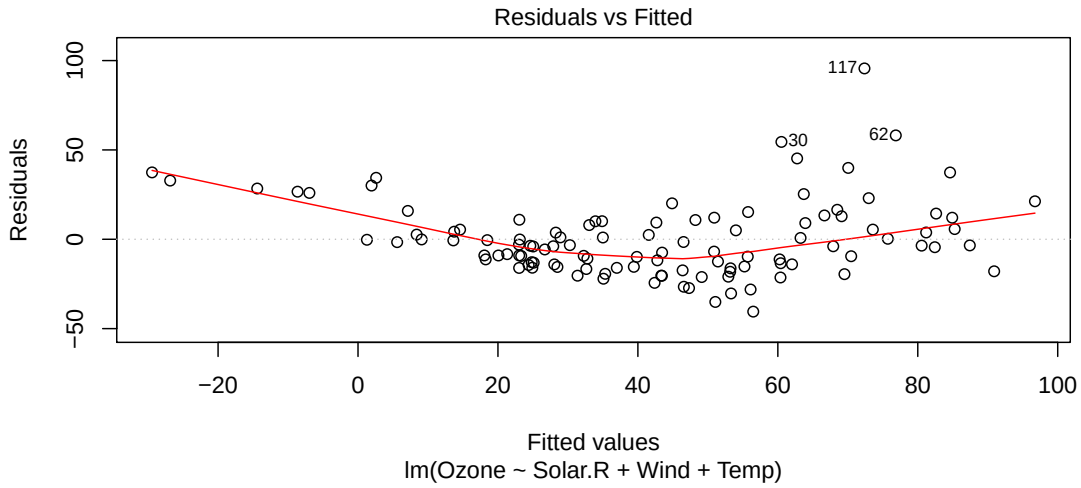
We first graph all pairs of variables in the dataset.

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



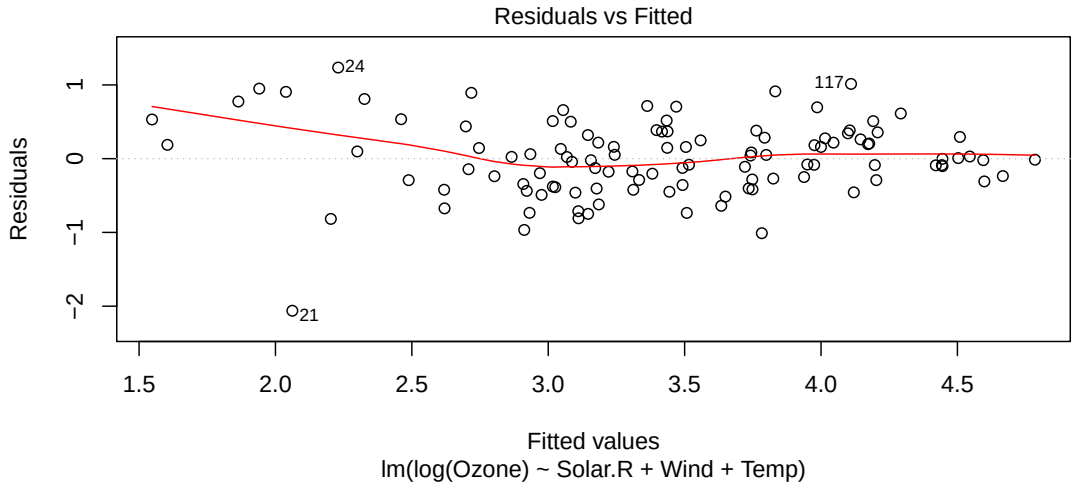
and notice that ozone levels are positively correlated with solar radiation and temperature, and negatively correlated with wind speed. A linear relationship with wind might be suspect as is the increasing variability in the response to high temperature. However, we don't know if those trends will remain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

```
model <- lm(Ozone ~ Solar.R + Wind + Temp, data=airquality)
plot(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. Looking at a log transformation of the y variable, we see a distinct improvement.

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



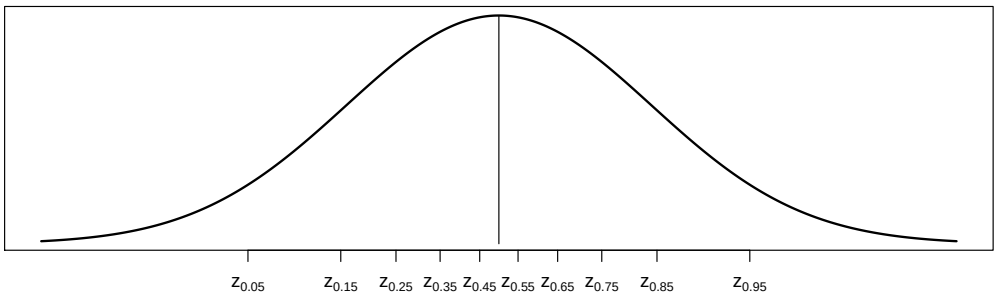
8.2.2 QQplots

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

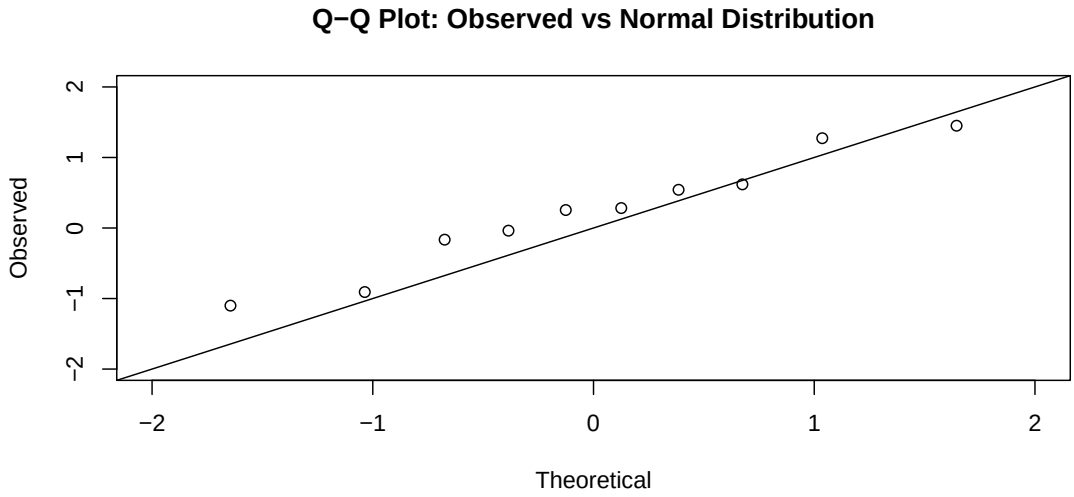
increments, but starting at the $0.5/n$ quantile instead of the $1/n$ quantile. So the smallest observation should be near the 0.05 quantile, the second smallest should be near the 0.15 quantile, and the largest observation should be near the 0.95 quantile. I will refer to these as the *theoretical quantiles*.

```
x <- seq(-3,3,length=1000)
labels <- NULL
labels[1] <- expression(paste(z[.05]))
labels[2] <- expression(paste(z[.15]))
labels[3] <- expression(paste(z[.25]))
labels[4] <- expression(paste(z[.35]))
labels[5] <- expression(paste(z[.45]))
labels[6] <- expression(paste(z[.55]))
labels[7] <- expression(paste(z[.65]))
labels[8] <- expression(paste(z[.75]))
labels[9] <- expression(paste(z[.85]))
labels[10]<- expression(paste(z[.95]))
plot(x, dnorm(x), axes=FALSE, type='l', lwd=2, ylab='', xlab='');
box(); axis(1, at=qnorm((1:10-.5)/10), lab=labels);
lines(c(0,0), c(0,dnorm(0)))
```



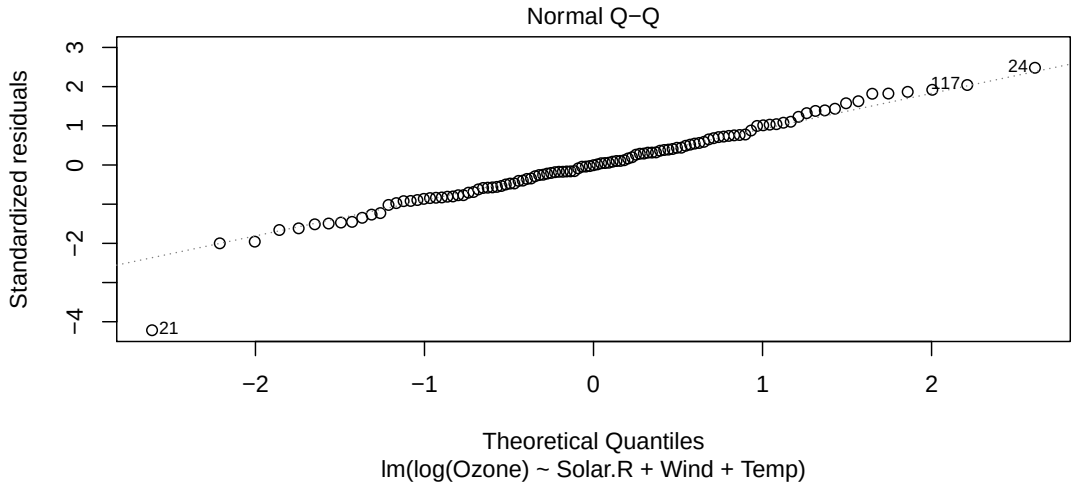
I can then graph the theoretical quantiles vs my observed values and if they lie on the 1-to-1 line, then my data comes from a standard normal distribution.

```
n <- 10
x <- rnorm(n, mean=0, sd=1)
theoretical <- qnorm( (1:n - .5)/(n), mean=0, sd=1)
plot(theoretical, sort(x), xlab='Theoretical', ylab='Observed',
xlim=c(-2,2), ylim=c(-2,2),
main="Q-Q Plot: Observed vs Normal Distribution")
abline(0, 1)
```



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 `airquality` example, R will calculate the qqplot for us.

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



In this case, we have one observation that is worrisome, and we might wish to investigate the circumstances surrounding that point more closely.

8.2.3 Scale-Location Plot

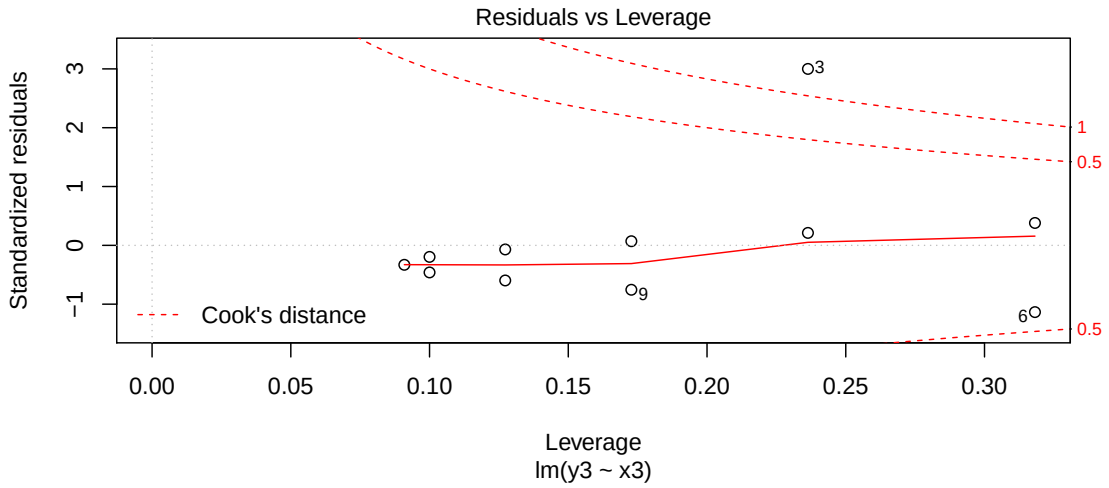
This plot is a variation on the fitted vs residuals plot, but the y-axis uses the square root of the absolute value of the standardized residuals. Supposedly this makes detecting increasing variance easier to detect, but I'm not convinced.

8.2.4 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. Since 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(y3 ~ x3, data=anscombe)
plot(model3, which=5)
```



that one data point (observation 3) has an extremely large standardized residual and large value of Cook's Distance.

Chapter 9

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

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

where $\epsilon \sim N\left(0, \sigma^2\right)$, 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.

9.1 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.

9.1.1 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 (since $\log 0 = -\infty$ 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

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

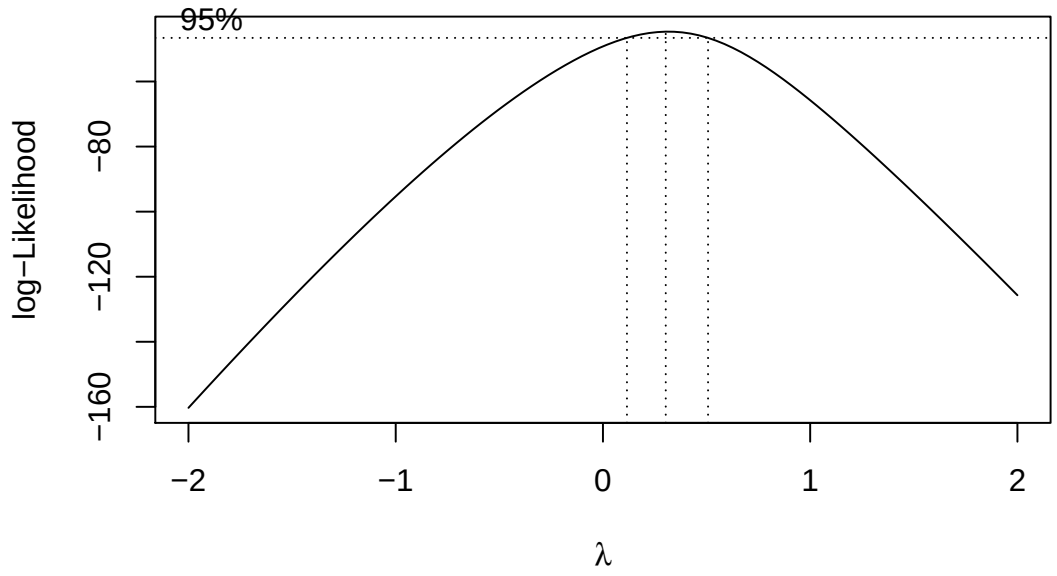
This transformation is a smooth family of transformations since

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

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.

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

9.2 Transforming the predictors

9.2.1 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.

9.2.2 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.

$$Area = \pi r^2$$

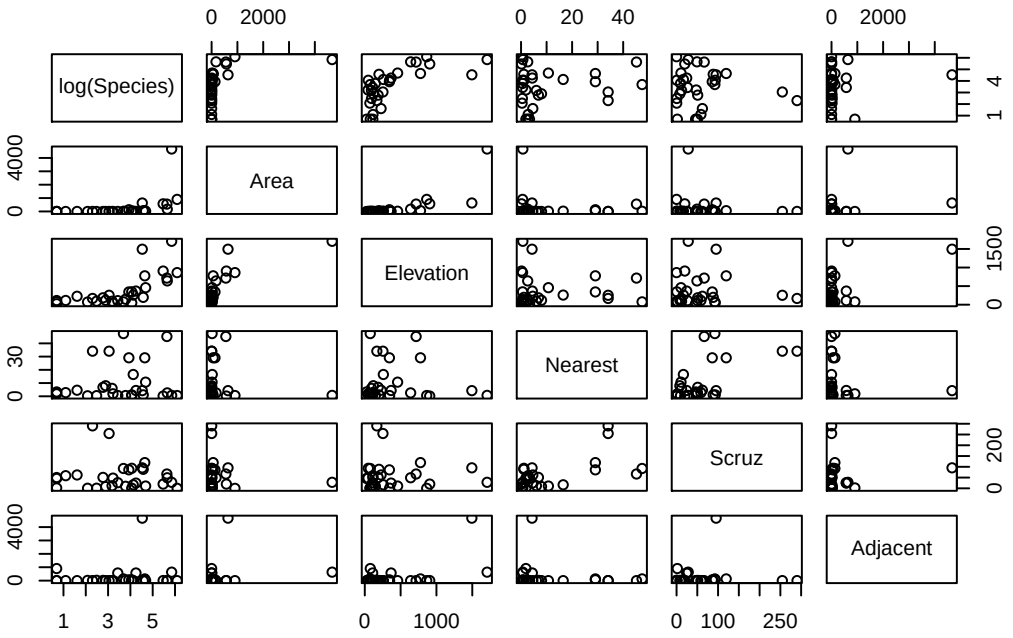
Similar situations might arise in biological settings, such as the volume of conducting tissue being related to the square of the diameter. Or perhaps an animals metabolic requirements are related to some power of body length. In sociology, it is often seen that the utility of, say, \$1000 drops off 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?

9.2.3 Examples of transformation of predictors

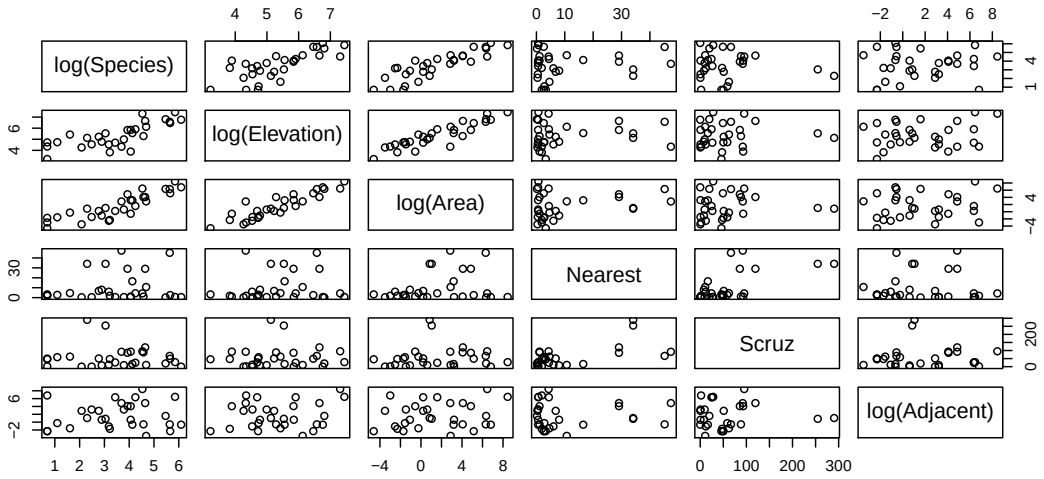
To illustrate how to add a transformation of a predictor to a linear model in R, we will consider the Galapagos data in `faraway`.

```
library(faraway)
data(gala)
# look at all the scatterplots
pairs(log(Species) ~ Area + Elevation + Nearest + Scruz + Adjacent, data=gala)
```



Looking at these graphs, I think I should transform `elevation` , `Adjacent`, and `Area` and perhaps a log transformation is a good idea.

```
pairs( log(Species) ~ log(Elevation) + log(Area) +
                      Nearest + 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(Elevation) + log(Area) +
                       Nearest + Scruz + log(Adjacent), data=gala)
round(summary(m.c)$coefficients, digits=3) # more readable...

               Estimate Std. Error t value Pr(>|t|)
(Intercept)       5.026      1.641   3.063    0.005
log(Elevation)   -0.383      0.328  -1.170    0.254
log(Area)         0.508      0.104   4.897    0.000
Nearest          -0.004      0.014  -0.324    0.749
Scruz            -0.003      0.003  -1.121    0.273
log(Adjacent)    -0.021      0.045  -0.455    0.653
```

We will remove all the parameters that appear to be superfluous, and perorm 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(Elevation) + log(Area) + Nearest + Scruz +
    log(Adjacent)
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1     28 17.2
2     24 14.7  4      2.55 1.04    0.4
```

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   18.48  < 2e-16 ***
log(Area)     0.3886     0.0416    9.34  4.2e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.784 on 28 degrees of freedom
Multiple R-squared:  0.757,Adjusted R-squared:  0.748
F-statistic: 87.3 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 section 3 of this chapter.

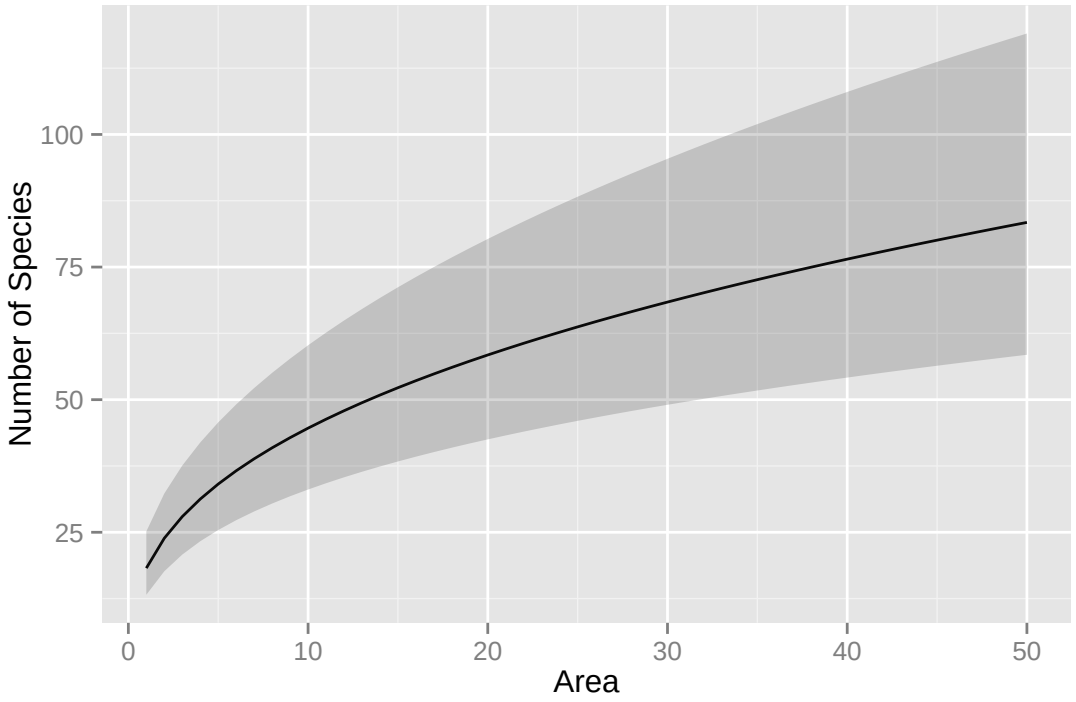
Finally, we might be interested in creating a confidence interval for the expected number of tortoise species for an island with `Area=50`.

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

    fit   lwr   upr
1 4.424 4.068 4.779

exp(log.Species.CI)  # Species scale

    fit   lwr upr
1 83.42 58.46 119
```

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)
log.Species.CI <- predict(m.s, pred.data, interval='confidence')
pred.data$log.Species.hat <- log.Species.CI[,1]
pred.data$log.Species.lwr <- log.Species.CI[,2]
pred.data$log.Species.upr <- log.Species.CI[,3]
ggplot(pred.data, aes(x=Area)) +
  geom_line(aes(y=exp(log.Species.hat))) +
  geom_ribbon(aes(ymin=exp(log.Species.lwr), ymax=exp(log.Species.upr)), alpha=.2) +
  ylab('Number of Species')
```

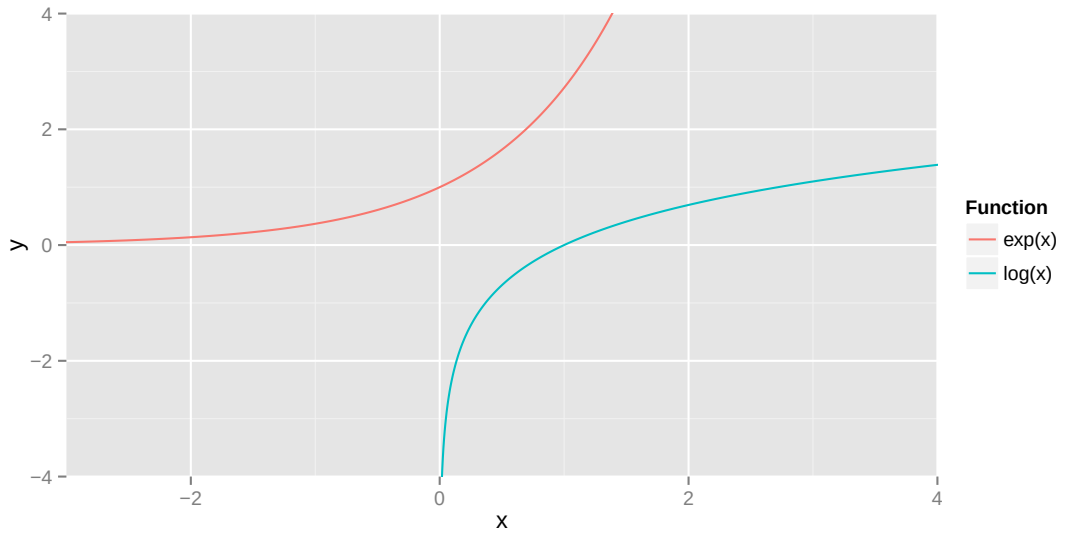


9.3 Interpretation of log transformed variables

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.

```
library(ggplot2)
x <- seq(-4,5, length=1000)
data <- data.frame(x=x, y=exp(x), funct='exp(x)')
x <- seq(0,5, length=1000)[-1]
data <- rbind(data, data.frame(x=x, y=log(x), funct='log(x)'))
ggplot(data, aes(x=x, y=y, color=funct)) +
    geom_line() + coord_cartesian(ylim = c(-4, 4), xlim=c(-3,4)) +
labs(color="Function")
```



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.

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

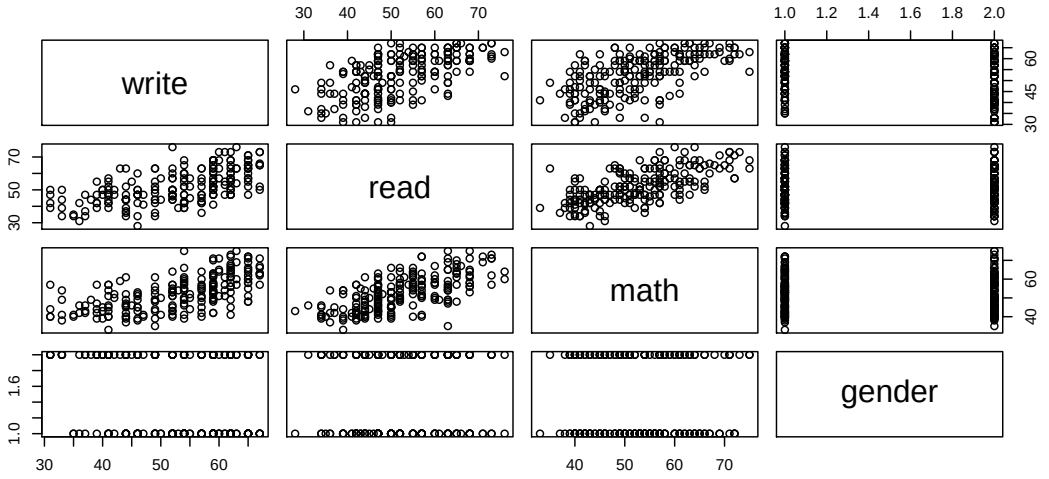
Also it is important to note that the log function has some interesting properties in that it makes operations "1-operation easier".

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

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[1].

[1]This example was taken from the UCLA Statistical Consulting Group

```
scores <- read.table(
  file='http://www.ats.ucla.edu/stat/mult_pkg/faq/general/lgtrans.csv',
  header=TRUE, sep=',')
scores$gender <- scores$female
pairs(write~read+math+gender, data=scores)
```



9.3.1 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)
round(summary(m)$coefficients, digits=3)

            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.948      0.014   288.4        0
```

We interpret the intercept as the mean of the log-transformed response values. We could back transform this to the original scale $e^{\hat{\beta}_0} = e^{3.948} = 51.83$ 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)
round(summary(m)$coefficients, 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 the offset for males is the change in `log(write)` from the female group. Notice that for the males, we have

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

and therefore we see that males tend to have writing scores $e^{-0.103} = 0.90 = 90\%$ of 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)
round(summary(m)$coefficients, 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.

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

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 justification:

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

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 5-unit increase in x, then that would result in an increase of

$$\frac{e^{\hat{\beta}_0 + \hat{\beta}_2 (x+5)}}{e^{\hat{\beta}_0 + \hat{\beta}_2 x}} = \frac{e^{\hat{\beta}_0} e^{\hat{\beta}_2 x} e^{5\hat{\beta}_2}}{e^{\hat{\beta}_0} e^{\hat{\beta}_2 x}} = e^{5\hat{\beta}_2}$$

and for the writing scores we have $e^{5\hat{\beta}_2} = \left(e^{\hat{\beta}_2}\right)^5 = 1.01^5$.

In short, we can interpret $e^{\hat{\beta}_i}$ as the percent increase/decrease in the non-transformed response variable.[2]

9.3.2 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 $\hat{y}$.

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

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 changing x from 1 to 2 changes $\hat{y}$ the same amount as changing x from 50 to 100.

9.3.3 Log-transformed response, log-transformed covariate

This combines the interpretation of the in the previous two section. 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.

[2] Some students get confused by what is meant by a % increase or decrease in x.

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

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

```
round(summary(m.s)$coefficients, 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 $\texttt{Area}_2/\texttt{Area}_1 = 2$) 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 30%.

Chapter 10

Variable Selection

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 collinear.

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.

10.1 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.

10.2 Testing-Based Model Selection

10.2.1 Backward Elimination

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.

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.

10.2.2 Forward Selection

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.

10.2.3 Stepwise Selection

This is a hybrid between forward selection and backward elimination. At every stage, a term is either added or removed.

10.2.4 Thoughts on testing based methods

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.

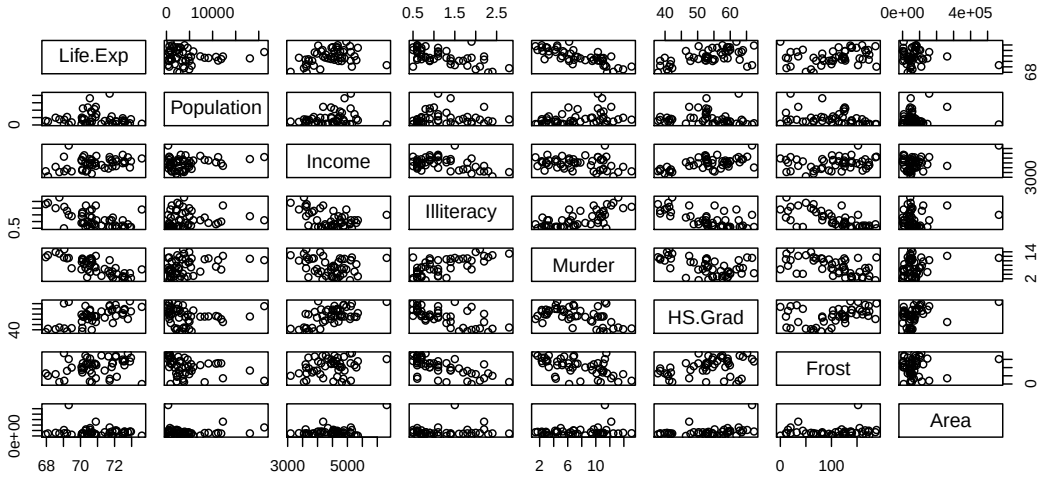
10.2.5 Example - U.S. Life Expectancy

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.

```
library(faraway)

# 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 data.

```
pairs(Life.Exp ~ ., data=state.data)
```



I want to add a quadratic effect for `HS.Grad` rate and for `Income`. We'll add it to the data frame and then perform the backward elimination method starting with the model with all predictors as main effects.

```
state.data$HS.Grad.2 <- state.data$HS.Grad ^ 2
state.data$Income.2  <- state.data$Income  ^ 2
# explicitly define my starting model
m1 <- lm(Life.Exp ~ Population + Income + Illiteracy +
                Murder + HS.Grad + Frost + HS.Grad.2 + Income.2, 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)
round(summary(m1)$coefficients, digits=3)

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   63.413      7.449   8.513    0.000
Population     0.000      0.000   1.310    0.198
Income         0.004      0.003   1.605    0.116
Illiteracy     0.255      0.388   0.656    0.515
Murder        -0.300      0.049  -6.088    0.000
HS.Grad       -0.053      0.214  -0.248    0.806
Frost         -0.004      0.003  -1.374    0.177
Area           0.000      0.000   0.656    0.516
HS.Grad.2      0.001      0.002   0.460    0.648
Income.2       0.000      0.000  -1.618    0.114
```

The signs make reasonable sense (higher murder rates decrease life expectancy) but covariates like `income` are not significant, which is surprising. We will first remove the term with the highest p-value, which is the state's `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)
round( summary(m1)$coefficients, digits=3)

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   62.497      6.395   9.772    0.000
Population     0.000      0.000   0.838    0.407
Income         0.005      0.003   1.804    0.078
Illiteracy    -0.076      0.358  -0.212    0.833
Murder        -0.308      0.047  -6.573    0.000
Frost         -0.006      0.003  -1.948    0.058
Area           0.000      0.000   1.688    0.099
Income.2       0.000      0.000  -1.750    0.087
```

```
# Next remove Illiteracy
m1 <- update(m1, .~. - Illiteracy)
round( summary(m1)$coefficients, digits=3)

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   61.779      5.367  11.510    0.000
Population     0.000      0.000   0.860    0.394
Income         0.005      0.002   2.165    0.036
Murder        -0.312      0.043  -7.304    0.000
Frost         -0.006      0.003  -2.261    0.029
Area           0.000      0.000   1.736    0.090
Income.2       0.000      0.000  -2.069    0.045
```

```
# And Population...
m1 <- update(m1, .~. - Population)
round( summary(m1)$coefficients, digits=3)

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   59.985      4.931  12.166    0.000
Income         0.006      0.002   2.702    0.010
Murder        -0.297      0.039  -7.633    0.000
Frost         -0.006      0.003  -2.471    0.017
Area           0.000      0.000   1.746    0.088
Income.2       0.000      0.000  -2.533    0.015
```

```
# Remove Area from the model
m1 <- update(m1, .~. - Area)
round( summary(m1)$coefficients, digits=3)

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   63.685      4.552  13.990    0.000
Income         0.004      0.002   2.078    0.043
Murder        -0.285      0.039  -7.278    0.000
Frost         -0.006      0.003  -2.235    0.030
Income.2       0.000      0.000  -1.864    0.069
```

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, data = state.data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4440 -0.5323 -0.0499  0.5016  1.5175

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.37e+01   4.55e+00   13.99  < 2e-16 ***
Income       4.02e-03   1.93e-03    2.08    0.043 *
Murder      -2.85e-01   3.92e-02   -7.28  3.9e-09 ***
Frost       -5.69e-03   2.54e-03   -2.24    0.030 *
Income.2    -3.95e-07   2.12e-07   -1.86    0.069 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.762 on 45 degrees of freedom
Multiple R-squared:  0.704,Adjusted R-squared:  0.678
F-statistic: 26.8 on 4 and 45 DF,  p-value: 2.11e-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
-1.5901 -0.4696  0.0039  0.5706  1.9229

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 74.55672    0.58425  127.61   <2e-16 ***
Illiteracy  -0.60176    0.29893   -2.01   0.0500 *
Murder      -0.28005    0.04339   -6.45    6e-08 ***
Frost       -0.00869    0.00296   -2.94   0.0052 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.791 on 46 degrees of freedom
Multiple R-squared:  0.674,Adjusted R-squared:  0.653
F-statistic: 31.7 on 3 and 46 DF,  p-value: 2.91e-11
```

Notice that the R^2 values for both models are quite similar 0.7042 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.

10.3 Criterion Based Procedures

10.3.1 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 thus

$$\begin{aligned} \log L\left(\hat{\boldsymbol{\beta}}, \hat{\sigma} \,|\, \text{data}\right) &= -\log\left(\left(2\pi\hat{\sigma}^2\right)^{n/2}\right) - \frac{1}{2\hat{\sigma}^2}\left(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{\beta}}\right)^T\left(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{\beta}}\right) \\ &= -\frac{n}{2}\log\left(2\pi\hat{\sigma}^2\right) - \frac{1}{2\hat{\sigma}^2}\left(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{\beta}}\right)^T\left(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{\beta}}\right) \\ &= -\frac{1}{2}\left[n\log\left(2\pi\hat{\sigma}^2\right) + \frac{1}{\hat{\sigma}^2}\left(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{\beta}}\right)^T\left(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{\beta}}\right)\right] \\ &= -\frac{1}{2}\left[+n\log\left(2\pi\right) + n\log\hat{\sigma}^2 + \frac{1}{\hat{\sigma}^2}RSS\right] \end{aligned}$$

It isn't clear what we should do with the $n\log(2\pi)$ 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. Since 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` vs `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] 121.4
```

```
AIC(m2)
```

```
[1] 124.3
```

Therefore since the AIC value for the first model is *lower*, we would prefer the first model that includes both `Income` and `Income.2`.

10.3.2 Adjusted R^2

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}.

10.3.3 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.

```
m <- lm(Life.Exp ~ 1, data=state.data)
#
# Define the biggest model I wish to consider
biggest <- Life.Exp ~ Population + Income + Illiteracy + Murder +
                      HS.Grad + Frost + Area + HS.Grad.2 + Income.2
add1(m, scope=biggest)

Single term additions

Model:
Life.Exp ~ 1
           Df Sum of Sq  RSS   AIC
<none>                  88.3  30.4
Population  1       0.4 87.9  32.2
Income      1      10.2 78.1  26.3
Illiteracy  1      30.6 57.7  11.2
Murder      1      53.8 34.5 -14.6
HS.Grad     1      29.9 58.4  11.7
Frost       1       6.1 82.2  28.9
Area        1       1.0 87.3  31.9
HS.Grad.2   1      27.4 60.9  13.8
Income.2    1       7.5 80.8  28.0
```

Clearly `Murder` has 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)
add1(m, scope=biggest)

Single term additions

Model:
Life.Exp ~ Murder
           Df Sum of Sq  RSS   AIC
<none>                   34.5 -14.6
Population  1      4.02 30.4 -18.8
Income      1      2.40 32.1 -16.2
Illiteracy  1      0.27 34.2 -13.0
HS.Grad     1      4.69 29.8 -19.9
Frost       1      3.13 31.3 -17.4
Area        1      0.47 34.0 -13.3
HS.Grad.2   1      4.44 30.0 -19.5
Income.2    1      1.90 32.6 -15.4
```

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 ~ Population + Income + Illiteracy +
                      Murder + HS.Grad + Frost + Area + HS.Grad.2 + Income.2
m <- lm(Life.Exp ~ Income, data=state.data)
step(m, scope=list(lower=smallest, upper=biggest))

Start:  AIC=26.28
Life.Exp ~ Income

             Df Sum of Sq  RSS   AIC
+ Murder      1      46.0 32.1 -16.2
+ Illiteracy  1      21.1 57.0  12.5
+ HS.Grad     1      19.8 58.3  13.7
+ Income.2    1      19.1 59.0  14.3
+ HS.Grad.2   1      17.2 60.9  15.8
+ Area        1       5.4 72.7  24.7
+ Frost       1       3.2 74.9  26.2
<none>                    78.1  26.3
+ Population  1       1.8 76.3  27.1
- Income      1      10.2 88.3  30.4

Step:  AIC=-16.23
Life.Exp ~ Income + Murder
```

```
             Df Sum of Sq  RSS   AIC
+ Frost       1       3.9 28.1 -20.7
+ Income.2    1       3.0 29.0 -19.2
+ Population  1       2.6 29.5 -18.4
+ HS.Grad     1       2.4 29.7 -18.1
+ HS.Grad.2   1       2.2 29.9 -17.8
<none>                    32.1 -16.2
- Income      1       2.4 34.5 -14.6
+ Illiteracy  1       0.0 32.0 -14.2
+ Area        1       0.0 32.1 -14.2
- Murder      1      46.0 78.1  26.3

Step:  AIC=-20.74
Life.Exp ~ Income + Murder + Frost

             Df Sum of Sq  RSS   AIC
+ HS.Grad     1       2.9 25.2 -24.3
+ HS.Grad.2   1       2.8 25.4 -23.9
+ Income.2    1       2.0 26.1 -22.5
+ Population  1       1.3 26.8 -21.2
<none>                    28.1 -20.7
+ Illiteracy  1       0.9 27.2 -20.5
+ Area        1       0.1 28.0 -19.0
- Income      1       3.2 31.3 -17.4
- Frost       1       3.9 32.1 -16.2
- Murder      1      46.8 74.9  26.2

Step:  AIC=-24.28
Life.Exp ~ Income + Murder + Frost + HS.Grad

             Df Sum of Sq  RSS   AIC
+ Population  1       1.9 23.3 -26.2
+ Income.2    1       1.9 23.3 -26.1
- Income      1       0.2 25.4 -25.9
<none>                    25.2 -24.3
+ HS.Grad.2   1       0.2 25.0 -22.7
+ Illiteracy  1       0.1 25.1 -22.5
+ Area        1       0.1 25.1 -22.4
- HS.Grad     1       2.9 28.1 -20.7
- Frost       1       4.5 29.7 -18.1
- Murder      1      32.9 58.1  15.5

Step:  AIC=-26.17
Life.Exp ~ Income + Murder + Frost + HS.Grad + Population

             Df Sum of Sq  RSS   AIC
- Income      1       0.0 23.3 -28.2
<none>                    23.3 -26.2
+ Income.2    1       0.8 22.5 -25.9
- Population  1       1.9 25.2 -24.3
+ HS.Grad.2   1       0.0 23.3 -24.2
+ Illiteracy  1       0.0 23.3 -24.2
+ Area        1       0.0 23.3 -24.2
```

```
- Frost       1       3.0 26.3 -22.1
- HS.Grad     1       3.5 26.8 -21.2
- Murder      1      34.7 58.0  17.5

Step:  AIC=-28.16
Life.Exp ~ Murder + Frost + HS.Grad + Population

             Df Sum of Sq  RSS   AIC
<none>                    23.3 -28.2
+ Income.2    1       0.0 23.3 -26.2
+ HS.Grad.2   1       0.0 23.3 -26.2
+ Income      1       0.0 23.3 -26.2
+ Illiteracy  1       0.0 23.3 -26.2
+ Area        1       0.0 23.3 -26.2
- Population  1       2.1 25.4 -25.9
- Frost       1       3.1 26.4 -23.9
- HS.Grad     1       5.1 28.4 -20.2
- Murder      1      34.8 58.1  15.5

Call:
lm(formula = Life.Exp ~ Murder + Frost + HS.Grad + Population,
    data = state.data)

Coefficients:
(Intercept)       Murder        Frost      HS.Grad   Population
   7.10e+01    -3.00e-01    -5.94e-03     4.66e-02     5.01e-05
```

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] 121.4
```

and the model selected by the stepwise algorithm was

```
m3 <- lm(Life.Exp ~ Murder + Frost + HS.Grad + Population, data = state.data)
AIC(m3)

[1] 115.7
```

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] 5.697
```

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>                  26.1 -22.5
Population  1     0.424 25.7 -21.3
Illiteracy  1     0.101 26.0 -20.7
HS.Grad     1     2.795 23.3 -26.1
Area        1     1.693 24.4 -23.8
HS.Grad.2   1     2.797 23.3 -26.1
```

```
add1(m3, scope=biggest)

Single term additions

Model:
Life.Exp ~ Murder + Frost + HS.Grad + Population
           Df Sum of Sq  RSS   AIC
<none>                  23.3 -28.2
Income      1   0.00606 23.3 -26.2
Illiteracy  1   0.00392 23.3 -26.2
Area        1   0.00079 23.3 -26.2
HS.Grad.2   1   0.00734 23.3 -26.2
Income.2    1   0.03142 23.3 -26.2
```

Using these results, we can calculate the difference in AIC values to be the same as we calculated before

$$\begin{aligned} -22.465 - -28.161 &= -22.465 + 28.161 \\ &= 5.696 \end{aligned}$$

Chapter 11

Block Designs

Block designs have a similar feel as fractional factorial designs and are commonly found in biological and ecological studies.

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.*

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.

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.[1] 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.

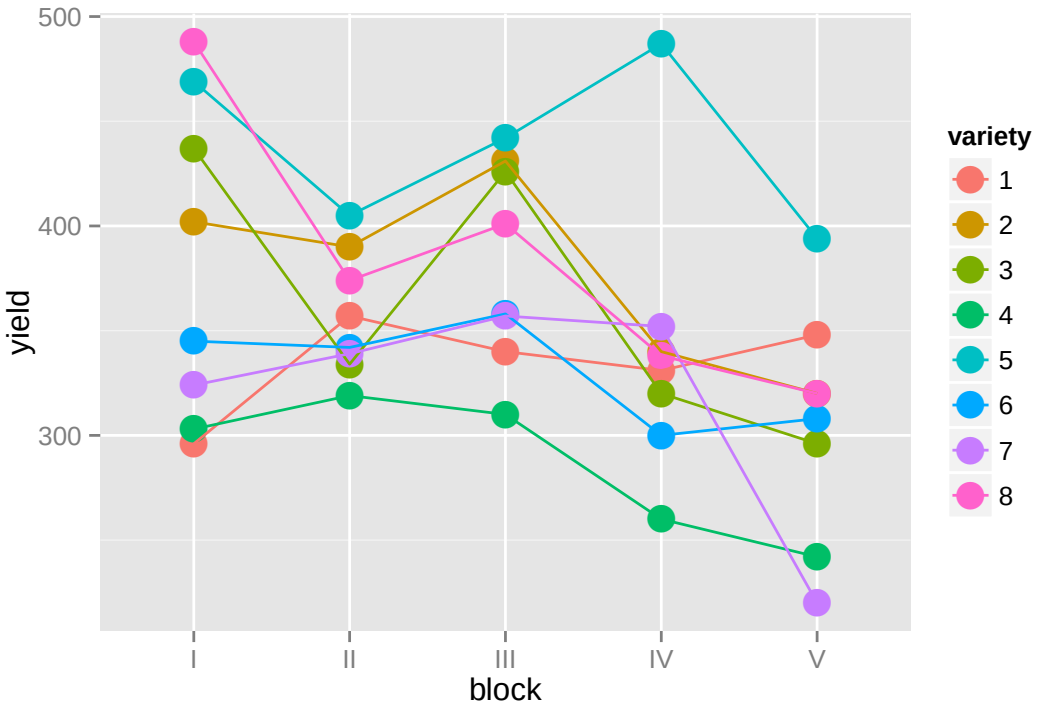
11.1 One blocking variable

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

[1] See *Pinky and the Brain* episodes from the show *The Animaniacs* for animated "proof".

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.

```
library(faraway)
library(ggplot2)
data(oatvar)
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 degrees of freedom because

$$\begin{aligned} df &= n - p \\ &= n - (1 + [(I-1) + (J-1)]) \\ &= n - (1 + [(5-1) + (8-1)]) \\ &= n - 12 \\ &= 28 \end{aligned}$$

```
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    8349    6.24   0.001 **
variety    7  77524   11075    8.28 1.8e-05 ***
Residuals 28  37433    1337
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# plot(m1)      # check diagnostic plots - they are fine...
# summary(m1)   # get coefficients, R-sq, etc... next, pairwise contrasts...
# TukeyHSD( aov(yield ~ block + variety, data=oatvar), which='variety') )
```

Unsurprisingly both the blocking and variety factors are significant. Since this is an orthogonal design, it doesn't matter which order we add the factors, but if we remove one or two observations, it would.

```
# remove two observations so that the design is not balanced
oatvar2 <- oatvar[c(-2,-9),]
m1 <- lm( yield ~ block + variety, data=oatvar2)
m2 <- lm( yield ~ variety + block, data=oatvar2)
anova(m1)

Analysis of Variance Table

Response: yield
          Df Sum Sq Mean Sq F value Pr(>F)
block      4  33186    8296    5.98 0.0015 **
variety    7  78885   11269    8.12  3e-05 ***
Residuals 26  36070    1387
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(m2)

Analysis of Variance Table

Response: yield
          Df Sum Sq Mean Sq F value  Pr(>F)
variety    7  79625   11375    8.20 2.8e-05 ***
block      4  32446    8111    5.85  0.0017 **
Residuals 26  36070    1387
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Because we are primarily interested in the effect of variety, we prefer to look at the effect of variety *after* the effect of the blocking variable(s) have been accounted for and we should use `m1`. The reason is that we decided *a priori* that the blocking variables would matter and the smallest model we should reasonably consider is the one that includes the blocking variable. So our test of if variety matters should be to compare the simple model with just the blocking variable to

the complex model with both the blocking variable and the variety main effects. Since this is the comparison we wish to make, ordering the model as `yield ~ block + variety` allows us to make that comparison directly from the ANOVA table.

We might consider an interaction model, in which the effect of variety depends on which block the observation is in. Because we only have one observation per block/variety combination, we cannot fit the interaction model and estimate a variance parameter so we cannot do an F-test to compare the interaction model with the main effect model. To be explicit, the degrees of freedom in the interaction model is 0 because

$$\begin{aligned} df &= n - p \\ &= n - (1 + [(I-1) + (J-1)] + (I-1)(J-1)) \\ &= n - (1 + [(5-1) + (8-1)] + (5-1)(8-1)) \\ &= n - (1 + 11 + 28) \\ &= 40 - 40 \end{aligned}$$

This means that we have the same number of parameters as we have data points and we can perfectly fit the observed data and all residuals will be zero. R will fit this model and give coefficient estimates, but because all residuals are zero, σ cannot be estimated and all standard errors, p-values, and confidence intervals cannot be calculated and will be reported as `NA` (not available).

```
m3 <- lm( yield ~ block * variety, data=oatvar)
summary(m3)
```

```

Call:
lm(formula = yield ~ block * variety, data = oatvar)

Residuals:
ALL 40 residuals are 0: no residual degrees of freedom!

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)            296         NA      NA       NA
blockII                 61         NA      NA       NA
blockIII                44         NA      NA       NA
blockIV                 35         NA      NA       NA
blockV                  52         NA      NA       NA
variety2               106         NA      NA       NA
variety3               141         NA      NA       NA
variety4                 7         NA      NA       NA
variety5               173         NA      NA       NA
variety6                49         NA      NA       NA
variety7                28         NA      NA       NA
variety8               192         NA      NA       NA
blockII:variety2       -73         NA      NA       NA
blockIII:variety2      -15         NA      NA       NA
blockIV:variety2       -97         NA      NA       NA
blockV:variety2       -134         NA      NA       NA
blockII:variety3      -164         NA      NA       NA
blockIII:variety3      -55         NA      NA       NA
blockIV:variety3      -152         NA      NA       NA
```

```
blockV:variety3       -193         NA      NA       NA
blockII:variety4       -45         NA      NA       NA
blockIII:variety4      -37         NA      NA       NA
blockIV:variety4       -78         NA      NA       NA
blockV:variety4       -113         NA      NA       NA
blockII:variety5      -125         NA      NA       NA
blockIII:variety5      -71         NA      NA       NA
blockIV:variety5       -17         NA      NA       NA
blockV:variety5       -127         NA      NA       NA
blockII:variety6       -64         NA      NA       NA
blockIII:variety6      -31         NA      NA       NA
blockIV:variety6       -80         NA      NA       NA
blockV:variety6        -89         NA      NA       NA
blockII:variety7       -46         NA      NA       NA
blockIII:variety7      -11         NA      NA       NA
blockIV:variety7        -7         NA      NA       NA
blockV:variety7       -156         NA      NA       NA
blockII:variety8      -175         NA      NA       NA
blockIII:variety8     -131         NA      NA       NA
blockIV:variety8      -185         NA      NA       NA
blockV:variety8       -220         NA      NA       NA

Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:     1,Adjusted R-squared:   NaN
F-statistic:  NaN on 39 and 0 DF,  p-value: NA
```

11.2 Latin Squares

It is often the case that we might have more than one blocking variable. We consider a manufacturing process in which we are testing the abrasion resistance of four different materials. We have a machine that will take four material samples and provide mechanical abrasion. The amount of wear is recorded and the lower the number, the better. However experience with the machine suggests that the four positions that are simultaneously abraded are not abraded equally. Furthermore the abrasion from run to run might not be completely constant. In these uncertain economic times, we are not allowed to buy new equipment and must make due with an intelligent design. We will make four runs of the abrasion machine, leading to 16 observations. Denote the four materials as A, B, C, and D, and the positions as P1, P2, P3, and P4.

	P1	P2	P3	P4
Run 1	A	B	C	D
Run 2	B	C	D	A
Run 3	C	D	A	B
Run 4	D	A	B	C

There are things to like in this design. In each run I have every material type and each material type is tested in every position. This is an example of a 4×4 *latin square*. Latin squares are much like a Sudoku puzzle where the goal is to have each level of the factor of interest occur exactly once in each row and each column.

But this seems a little too organized and without some sort of randomization, I am not protected against lurking variables. There are many different 4×4 latin squares so we should randomly select one from a list of all possible 4×4 latin squares. The latin square for this experiment was:

	P1	P2	P3	P4
Run 1	C	D	B	A
Run 2	A	B	D	C
Run 3	D	C	A	B
Run 4	B	A	C	D

Now we can fit our model with the blocking variables of `position` and `run` and the factor of interest.. `material`.

```
data(abrasion)
m1 <- lm( wear ~ position + run + material, data=abrasion)
anova(m1)

Analysis of Variance Table

Response: wear
          Df Sum Sq Mean Sq F value  Pr(>F)
position   3   1468     489    7.99 0.01617 *
run        3    986     329    5.37 0.03901 *
material   3   4621    1540   25.15 0.00085 ***
Residuals  6    368      61
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

After checking the diagnostic plots (which are unremarkable) we are now interested in which material is most abrasion resistant.

```
library(MASS)
summary(m1)$coef

            Estimate Std. Error t value  Pr(>|t|)
(Intercept)   254.75      6.187 41.1738 1.373e-08
position2      26.25      5.534  4.7434 3.180e-03
position3       8.50      5.534  1.5360 1.755e-01
position4       8.25      5.534  1.4908 1.866e-01
run2           -2.25      5.534 -0.4066 6.984e-01
run3           12.50      5.534  2.2588 6.466e-02
run4           -9.25      5.534 -1.6715 1.457e-01
materialB     -45.75      5.534 -8.2671 1.695e-04
materialC     -24.00      5.534 -4.3368 4.892e-03
materialD     -35.25      5.534 -6.3697 7.032e-04
```

Material `B` seems to be the best, followed by `D`, then `C`, and then `A`, but we should use Tukey's Honestly significant differences to see if these are statistically significant differences.

```
m <- aov(wear ~ position + run + material, data=abrasion)
TukeyHSD(m, which='material')

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

Fit: aov(formula = wear ~ position + run + material, data = abrasion)

$material
      diff     lwr     upr  p adj
B-A -45.75 -64.907 -26.593 0.0007
C-A -24.00 -43.157  -4.843 0.0190
D-A -35.25 -54.407 -16.093 0.0029
C-B  21.75   2.593  40.907 0.0295
D-B  10.50  -8.657  29.657 0.3206
D-C -11.25 -30.407   7.907 0.2743
```

11.3 Balanced Incomplete Block Designs

Often it is not possible to have all possible treatment combinations within each block level. In this case some treatment contrasts will be confounded with block contrasts, but if we are clever we can design the confounded contrasts to be high order contrasts.

Suppose we have four treatments ($t = 4$) creatively denoted as A, B, C and D. Further, suppose each block can have three treatment levels ($k = 3$) and we have four blocks ($b = 4$). One possible configuration is:

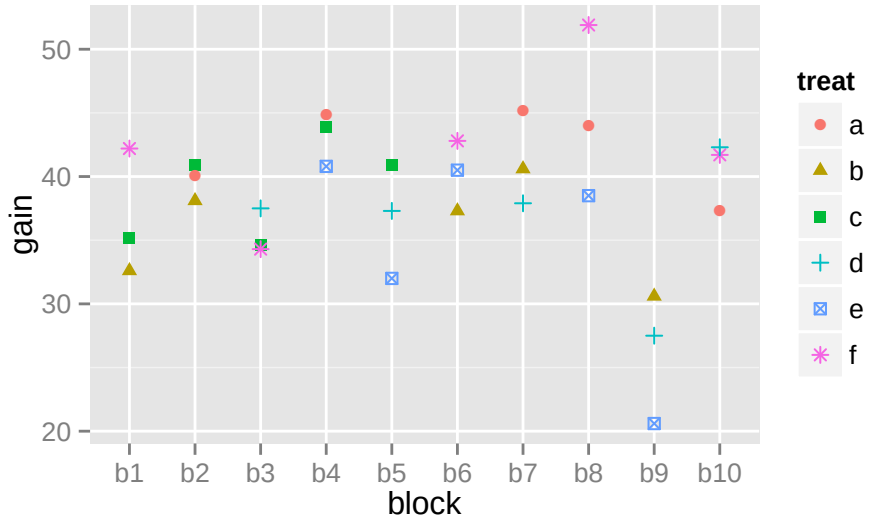
Block 1	A	B	C
Block 2	A	B	D
Block 3	A	C	D
Block 4	B	C	D

Notice that each treatment occurs 3 times, and every pair of treatments (AB, AC, AD, BC, BD, CD) occurs in two different blocks. This is the critical aspect that allows us to make simple pairwise comparisons between treatments.

Our example (`rabbit`) comes from a nutritionist examining the effects of different diets (a, b, c, d, e, f) on the growth of domestic rabbits. The nutritionist will select the three most uniform rabbits from a litter and consider them one block. We will have 10 different blocks of litters for a total of 30 rabbits.

```
data(rabbit)
levels(rabbit$block)  # ordered alphabetically

 [1] "b1"  "b10" "b2"  "b3"  "b4"  "b5"  "b6"  "b7"  "b8"  "b9"

rabbit$block <- factor(rabbit$block, levels=paste('b',1:10,sep=''))
levels(rabbit$block)  # now b1 : b10

 [1] "b1"  "b2"  "b3"  "b4"  "b5"  "b6"  "b7"  "b8"  "b9"  "b10"
```

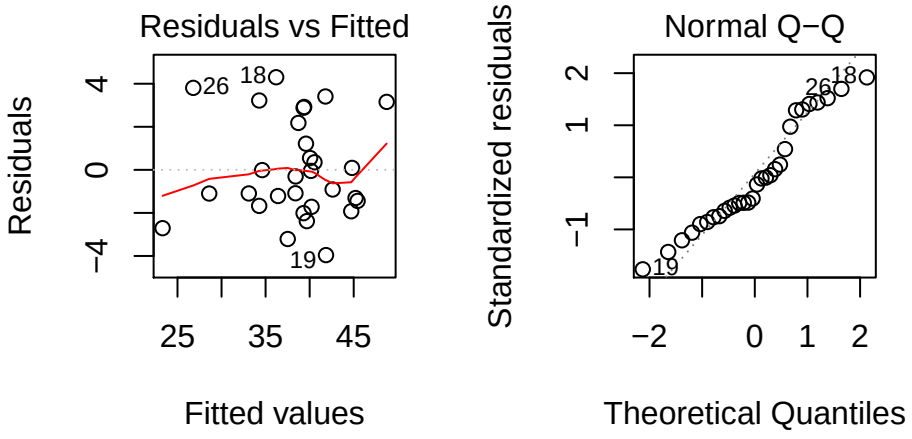
```
ggplot(rabbit, aes(y=gain, x=block, color=treat, shape=treat)) +
geom_point()
```



In this case each treatment will occur in 5 different blocks and be in the same block with the other treatment level twice. For example treatment `a` occurs in 5 blocks (`b2, b4, b7, b8, b10`) and is with treatment `b` in block `b2` and `b7`.

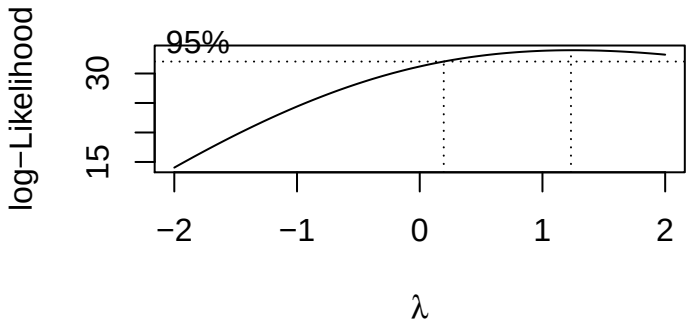
Next we fit our linear model with just the main effects, taking care to add the blocking effect before the treatment.

```
m1 <- lm( gain ~ block + treat, data=rabbit )
par(mfrow=c(1,2)) # side-by-side plots
plot(m1, which=c(1,2))
```



I'm not thrilled about the normality assumption, so we see if a Box-Cox transformation is appropriate.

```
library(MASS)
boxcox(m1)
```

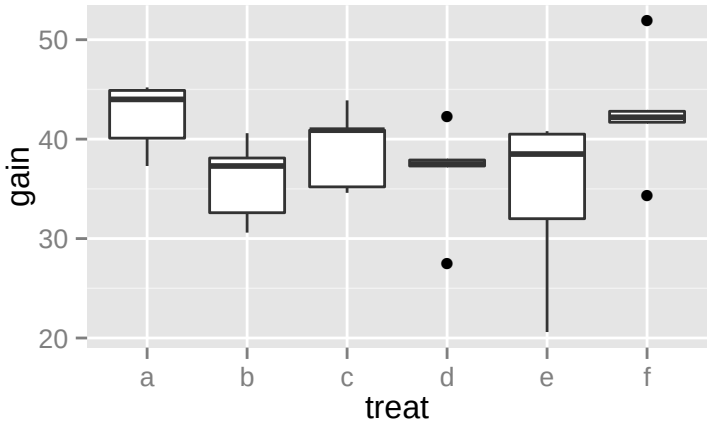


Since the confidence interval for λ contains 1, we will not make a transformation.

```
anova(m1)

Analysis of Variance Table

Response: gain
          Df Sum Sq Mean Sq F value  Pr(>F)
block      9    730    81.2    8.07 0.00025 ***
treat      5    159    31.7    3.16 0.03817 *
Residuals 15    151    10.1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that both the blocking effect (litter) and treatments are statistically significant.

```
ggplot(rabbit, aes(y=gain, x=treat)) + geom_boxplot()
```



Since the litters are a nuisance variable and I don't care about them, I am only interested in which

diets are different, and so we look at the TukeyHSD to examine which diets are different from which and we find that only diets `e` and `f` differ significantly.

```
m <- aov(gain ~ block + treat, data=rabbit)
TukeyHSD( m, which='treat' )  # only look at the treatment contrasts

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

Fit: aov(formula = gain ~ block + treat, data = rabbit)

$treat
        diff       lwr    upr  p adj
b-a -1.39333  -7.9080  5.121 0.9797
c-a  0.32000  -6.1947  6.835 1.0000
d-a  0.05333  -6.4613  6.568 1.0000
e-a -4.18000 -10.6947  2.335 0.3449
f-a  2.64000  -3.8747  9.155 0.7720
c-b  1.71333  -4.8013  8.228 0.9517
d-b  1.44667  -5.0680  7.961 0.9762
e-b -2.78667  -9.3013  3.728 0.7323
f-b  4.03333  -2.4813 10.548 0.3805
d-c -0.26667  -6.7813  6.248 1.0000
e-c -4.50000 -11.0147  2.015 0.2748
f-c  2.32000  -4.1947  8.835 0.8498
e-d -4.23333 -10.7480  2.281 0.3325
f-d  2.58667  -3.9280  9.101 0.7859
f-e  6.82000   0.3053 13.335 0.0377
```

We claimed that the design we used was good because while we couldn't look at all combinations of block/treatment combinations, the ones we did select allowed us to estimate the main effects of the blocks (because we had treatment `a` in blocks `1` and `2`, we can estimate the difference between blocks `1` and `2`). However, because we don't have all combinations, some things are not estimable. For example, because we don't have a treatment `b` in block `3`, that interaction is not estimable.

```
m <- lm(gain ~ block * treat, data=rabbit)
summary( m )  # only look at the treatment contrasts



Call:
lm(formula = gain ~ block * treat, data = rabbit)

Residuals:
ALL 30 residuals are 0: no residual degrees of freedom!

Coefficients: (30 not defined because of singularities)
               Estimate Std. Error t value Pr(>|t|)
(Intercept)         37.8         NA      NA       NA
blockb2              2.3         NA      NA       NA
blockb3             -7.9         NA      NA       NA
blockb4              7.1         NA      NA       NA
blockb5             -0.3         NA      NA       NA
blockb6              0.6         NA      NA       NA
```

```
blockb7              7.4         NA      NA       NA
blockb8              6.2         NA      NA       NA
blockb9            -11.7         NA      NA       NA
blockb10            -0.5         NA      NA       NA
treatb              -5.2         NA      NA       NA
treatc              -2.6         NA      NA       NA
treatd               5.0         NA      NA       NA
treate              -5.5         NA      NA       NA
treatf               4.4         NA      NA       NA
blockb2:treatb       3.2         NA      NA       NA
blockb3:treatb        NA         NA      NA       NA
blockb4:treatb        NA         NA      NA       NA
blockb5:treatb        NA         NA      NA       NA
blockb6:treatb       4.1         NA      NA       NA
blockb7:treatb       0.6         NA      NA       NA
blockb8:treatb        NA         NA      NA       NA
blockb9:treatb       9.7         NA      NA       NA
blockb10:treatb       NA         NA      NA       NA
blockb2:treatc       3.4         NA      NA       NA
blockb3:treatc       7.3         NA      NA       NA
blockb4:treatc       1.6         NA      NA       NA
blockb5:treatc       6.0         NA      NA       NA
blockb6:treatc        NA         NA      NA       NA
blockb7:treatc        NA         NA      NA       NA
blockb8:treatc        NA         NA      NA       NA
blockb9:treatc        NA         NA      NA       NA
blockb10:treatc       NA         NA      NA       NA
blockb2:treatd        NA         NA      NA       NA
blockb3:treatd       2.6         NA      NA       NA
blockb4:treatd        NA         NA      NA       NA
blockb5:treatd      -5.2         NA      NA       NA
blockb6:treatd        NA         NA      NA       NA
blockb7:treatd     -12.3         NA      NA       NA
blockb8:treatd        NA         NA      NA       NA
blockb9:treatd      -3.6         NA      NA       NA
blockb10:treatd       NA         NA      NA       NA
blockb2:treate        NA         NA      NA       NA
blockb3:treate        NA         NA      NA       NA
blockb4:treate       1.4         NA      NA       NA
blockb5:treate        NA         NA      NA       NA
blockb6:treate       7.6         NA      NA       NA
blockb7:treate        NA         NA      NA       NA
blockb8:treate        NA         NA      NA       NA
blockb9:treate        NA         NA      NA       NA
blockb10:treate       NA         NA      NA       NA
blockb2:treatf        NA         NA      NA       NA
blockb3:treatf        NA         NA      NA       NA
blockb4:treatf        NA         NA      NA       NA
blockb5:treatf        NA         NA      NA       NA
blockb6:treatf        NA         NA      NA       NA
blockb7:treatf        NA         NA      NA       NA
blockb8:treatf       3.5         NA      NA       NA
blockb9:treatf        NA         NA      NA       NA
```

```
blockb10:treatf       NA         NA      NA       NA

Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:     1,Adjusted R-squared:   NaN
F-statistic:  NaN on 29 and 0 DF,  p-value: NA
```

11.4 Generating Designs in R

Fundamentally we are searching for an experimental design that is optimal in some sense. Recall that the standard errors of our estimates $\hat{\boldsymbol{\beta}}$ are

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

for $j \in \{1, \ldots, p\}$ and therefore good design should result in a small values along the main diagonal of the matrix $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}$. Thus most of the optimality criteria rely on (in some fashion) the *information matrix* $\boldsymbol{X}^T\boldsymbol{X}$. This matrix is called the information matrix because it somehow measures the amount of information in our experimental design. As our sample size gets large, $\det\left(\boldsymbol{X}^T\boldsymbol{X}\right) \to \infty$. Some possible optimality criteria are:

A-optimality: which minimizes the sum of the main diagonal elements of the matrix $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}$, which is the same as minimizing the average variance of the $\hat{\boldsymbol{\beta}}$ coefficients.

D-optimality: which maximizes the determinant of the information matrix $\boldsymbol{X}^T\boldsymbol{X}$.

G-optimality: which minimizes the maximum entry in the diagonal of the hat matrix $H = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T$. This is effectively minimizing the maximum variance of a predicted value.

Unfortunately there usually is not a closed-form solution for producing the design matrix $\boldsymbol{X}$, but Federov (1972) produced a general algorithm to finding an optimal experimental design, starting from an experiment that contains balanced data that can estimate all possible contrasts. His algorithm will then select a subset of those treatment combinations to be tested that optimizes whatever criteria is chosen.

Classic balanced designs can be generated using this method, but good designs for experiments in which there is no fully balanced solution can also be found.

For example, we first generate a design for a Randomized Complete Block Experiment. In this case, we are interested in generating a Latin Square design for the abrasion example. In all of the following examples, we will use the D-optimality criteria.

```
library(AlgDesign)
## Generate all possible combinations of factors
dat <- gen.factorial(
  varNames=c('Material','Position','Run'),  # factor names
  levels=c(4,4,4), # how many levels does each factor have
  factors='all',   # all of these are categorical, (but continuous is possible!)
  center=FALSE     # Name the levels 1,2,3,etc instead of -1, 0, 1
)

## generate a design
optFederov(~., dat, nTrials=16, nRepeats=100)$design
```

```
   Material Position Run
4         4        1   1
5         1        2   1
10        2        3   1
15        3        4   1
19        3        1   2
22        2        2   2
25        1        3   2
32        4        4   2
33        1        1   3
40        4        2   3
43        3        3   3
46        2        4   3
50        2        1   4
55        3        2   4
60        4        3   4
61        1        4   4
```

If `nRepeats` is set to a small number, then we might not find the optimal solution and we may be asked to perform an experiment that is not possible. In general to generate a "classic" design, you should allow the algorithm to search for a reasonable amount of time. We next generate a design for a Randomized Incomplete Block Experiment. In this case, we'll generate a possible design for the `rabbit` example.

```
## Generate all possible combinations of factors
dat <- gen.factorial( levels=c(6, 10),
                      varNames=c('Treat','Litter'),
                      factors='all',
                      center=FALSE)
# what design could we use
optFederov(~., dat, nTrials=30, nRepeats=100)$design

   Treat Litter
2      2      1
5      5      1
6      6      1
8      2      2
9      3      2
12     6      2
15     3      3
16     4      3
17     5      3
19     1      4
23     5      4
24     6      4
27     3      5
28     4      5
30     6      5
31     1      6
34     4      6
36     6      6
37     1      7
38     2      7
39     3      7
43     1      8
45     3      8
47     5      8
49     1      9
50     2      9
52     4      9
56     2     10
58     4     10
59     5     10
```

Needless to say, this is just scratching the surface of experimental designs and substantially more complicated situations often arise. For further information see a design of experiments book take a course in it.

Chapter 12

Binomial Regression

The *linear model* assumes that the observed data is distributed

$$\boldsymbol{y} \sim N\left(\boldsymbol{\mu} = \boldsymbol{X\beta}, \sigma^2 \boldsymbol{I}\right)$$

and notably this assumes that the data are independent. This also has $E[\boldsymbol{y}] = \boldsymbol{X\beta}$. This model is quite flexible and includes:

simple linear regression	1 continuous predictor variable
1-way ANOVA	1 categorical predictor variable
ANCOVA	both continuous and categorical predictors
linear regression	multiple continuous predictors

The *general linear model* expanded on the linear model and we allow the data points to be correlated

$$\boldsymbol{y} \sim N\left(\boldsymbol{X\beta}, \sigma^2 \boldsymbol{\Omega}\right)$$

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 $\boldsymbol{X\beta}$ because we understand the how continuous and discrete covariates added and interpreted and what interactions between them mean. By keeping the $\boldsymbol{X\beta}$ 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.

$$\begin{aligned} P(W_i = 1) &= p_i \\ P(W_i = 0) &= (1 - p_i) \end{aligned}$$

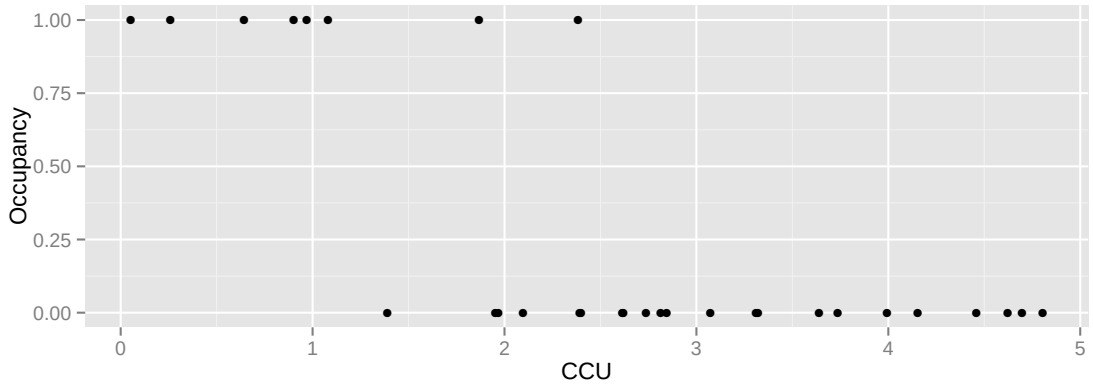
which I can rewrite more formally letting w_i be the observed value as

$$P(W_i = w_i) = p_i^{w_i}(1 - p_i)^{1-w_i}$$

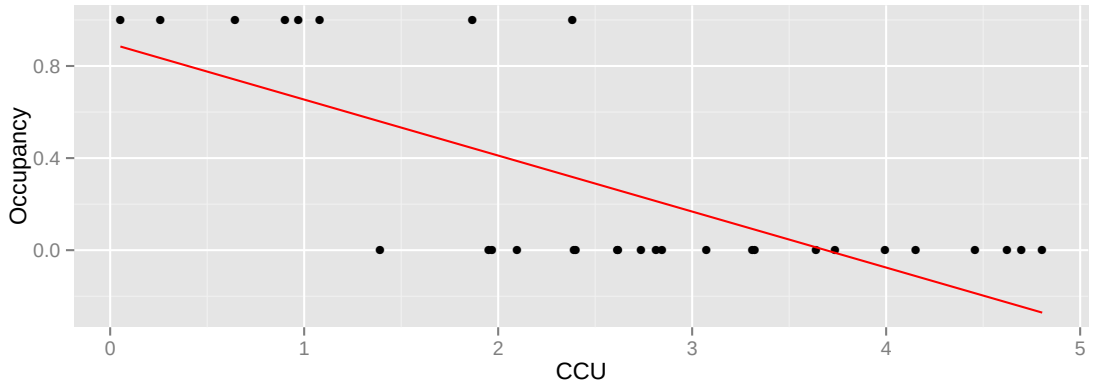
and the parameter that I wish to estimate and understand is the probability of a success p_i and usually I wish to know how my covariate data $\boldsymbol{X\beta}$ informs these probabilities.

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. Since 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. Since 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)[1] where larger values imply more metal pollution.

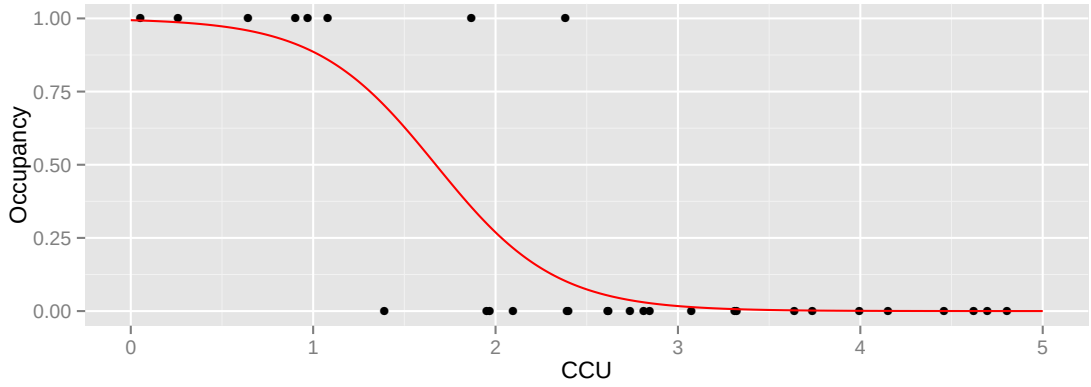


If I just fit a regular linear model to this data, we fit the following:



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:

[1]CCU is defined as the ratio of the measured metal concentration to the hardness adjusted chronic criterion concentration, and then summed across each metal.



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[2].

1. Logit 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 the probability is exactly 0.5.

Usually the difference in inferences made using these different curves is relatively small and we will usually use the logit transformation because of its representation as the log odds. 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."

[2]We use the notation $y_i = \boldsymbol{X}_{i,.}\boldsymbol{\beta}$ 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.

Since we will be using the logit transformation so often, it is useful to make the following definitions:

$$\begin{aligned} \text{logit}\,(p) &= \log\left[\frac{p}{1-p}\right] \\ \text{ilogit}\,(y) &= \frac{1}{1+e^{-y}} \end{aligned}$$

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 log-likelihood function.

$$\begin{aligned} \log\left[L\left(\boldsymbol{\beta}|\boldsymbol{w}\right)\right] &= \log\left[\prod_{i=1}^{n} L\left(\boldsymbol{\beta}|w_i\right)\right] \\ &= \sum_{i=1}^{n} \log L\left(\boldsymbol{\beta}|w_i\right) \\ &= \sum_{i=1}^{n} \log P\left(w_i|\boldsymbol{\beta}\right) \\ &= \sum_{i=1}^{n} \log\left[p_i^{wi}\left(1-p_i\right)^{1-w_i}\right] \end{aligned}$$

and we recognize that $p_i = \text{ilogit}\left(\boldsymbol{X}\boldsymbol{\beta}\right) = 1/\left(1+e^{-\boldsymbol{X}\boldsymbol{\beta}}\right)$ into the equation and simplify. Fortunately we don't have to worry about the details of maximizing the log-likelihood function as R will do it for us.

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 $n_i = 1$.

```
m1 <- glm( cbind(Occupancy, 1-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()`.

```
summary(m1)


Call:
glm(formula = cbind(Occupancy, 1 - Occupancy) ~ CCU, family = binomial,
    data = mayflies)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5574  -0.3159  -0.0655   0.0865   2.1336

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)     5.10       2.37    2.15    0.031 *
CCU            -3.05       1.21   -2.52    0.012 *
---
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: 16.65

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

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

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.

```
ilogit( predict(m1, newdata=data.frame(CCU=1)) )

    1
0.886
```

12.2 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( cbind(Occupancy, 1-Occupancy) ~ 1, data=mayflies, family=binomial )
anova(m0, m1)

Analysis of Deviance Table

Model 1: cbind(Occupancy, 1 - Occupancy) ~ 1
Model 2: cbind(Occupancy, 1 - Occupancy) ~ CCU
  Resid. Df Resid. Dev Df Deviance
1        29       34.8
2        28       12.6  1     22.1

1 - pchisq( 22.146, df=1 )

[1] 2.527e-06
```

A convenient way to get R to calculate the χ^2 p-value for you is to use the `drop1()` function.

```
drop1(m1, test='Chi')

Single term deletions

Model:
cbind(Occupancy, 1 - Occupancy) ~ CCU
       Df Deviance  AIC  LRT Pr(>Chi)
<none>        12.6 16.6
CCU     1     34.8 36.8 22.1  2.5e-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 36.79
m1  2 16.65
```

12.3 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.

$$\begin{aligned} X^2 &= \sum_{i=1}^{n} \left[\frac{(w_i - n_i\hat{p}_i)^2}{n_i\hat{p}_i} + \frac{((n_i - w_i) - n_i(1-\hat{p}_i))^2}{n_i(1-\hat{p}_i)} \right] \\ &= \sum_{i=1}^{n} \frac{(w_i - n_i\hat{p}_i)^2}{n_i\hat{p}_i(1-\hat{p}_i)} \end{aligned}$$

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] 14.92
```

Pearson's X^2 statistic is quite similar to the deviance statistic

```
deviance(m1)

[1] 12.65
```

12.4 Confidence Intervals

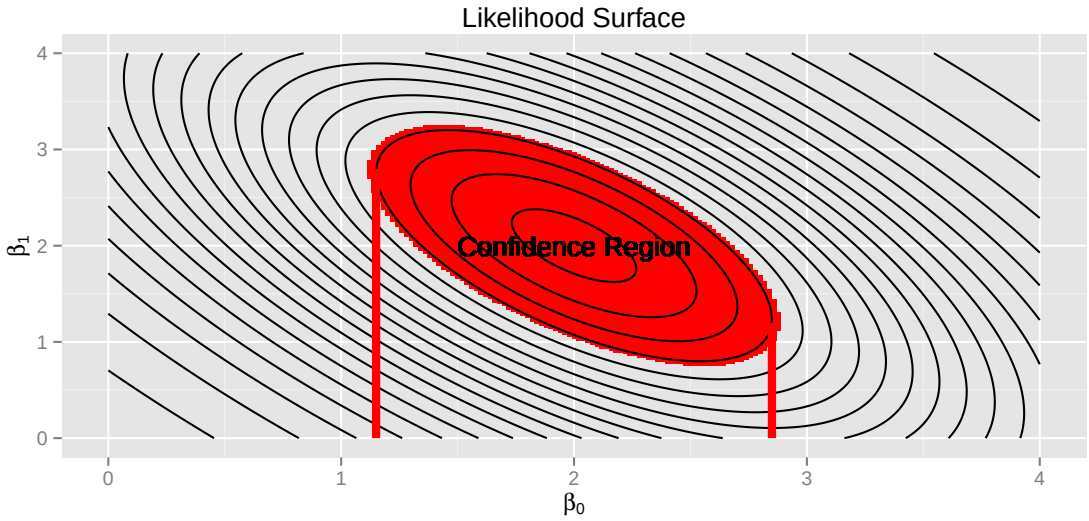
Confidence intervals for the regression could be constructed using normal approximations for the parameter estimates. An approximate $100(1-\alpha)\%$ confidence interval for β_i would be

$$\hat{\beta}_i \pm z^{1-\alpha/2} \, StdErr\left(\hat{\beta}_i\right)$$

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

$$\begin{aligned} -2\left[\log L\left(\beta_0, \beta_1\right) - \log L\left(\hat{\beta}_0, \hat{\beta}_1\right)\right] &\leq \chi^2_{df=2,0.95} \\ \log L\left(\beta_0, \beta_1\right) &\geq -2\chi^2_{2,0.95} + \log L\left(\hat{\beta}_0, \hat{\beta}_1\right) \end{aligned}$$



Looking at just the β_0 axis, this translates into a confidence interval of $(1.15, 2.85)$. 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.

```
library(MASS)
confint(m1)
```

```
             2.5 % 97.5 %
(Intercept)  1.630 11.781
CCU         -6.447 -1.304
```

12.5 Interpreting model coefficients

We first consider why we are dealing with odds $\frac{p}{1-p}$ instead of just p. The 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 spit up using odds, we 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 $.9/.1 = 9$ vs her odds are $.99/.01 = 99$. The odds ratio is $9/99 = 1/11$ and gives the same information as I calculated from the where we defined a success as my daughter not spitting up.

Given a logistic regression model with two continuous covariates, then using the `logit()` link function we have

$$\begin{aligned} \log\left(\frac{p}{1-p}\right) &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \\ \frac{p}{1-p} &= e^{\beta_0} e^{\beta_1 x_1} e^{\beta_2 x_2} \end{aligned}$$

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 (or x_2) 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)	Only Breast Milk (b)	Breast Milk + Supplement (s)
Males (M)	77/458	47/494	19/147
Females (F)	48/384	31/464	16/127

We can fit the saturated model (6 parameters to fit 6 different probabilities) as

```
data(babyfood)
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)        -1.5990     0.1249  -12.80  < 2e-16 ***
sexGirl            -0.3469     0.1985   -1.75  0.08059 .
foodBreast         -0.6534     0.1978   -3.30  0.00096 ***
foodSuppl          -0.3086     0.2758   -1.12  0.26314
sexGirl:foodBreast -0.0374     0.3123   -0.12  0.90460
sexGirl:foodSuppl   0.3176     0.4140    0.77  0.44301
---
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: 43.52

Number of Fisher Scoring iterations: 3
```

It is nice to look at the single term deletions to see if the interaction term could be dropped from the model.

```
drop1(m2, test='Chi')

Single term deletions

Model:
cbind(disease, nondisease) ~ sex * food
         Df Deviance  AIC   LRT Pr(>Chi)
<none>         0.000 43.5
sex:food  2    0.722 40.2 0.722      0.7
```

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')

Single term deletions

Model:
cbind(disease, nondisease) ~ sex + food
       Df Deviance  AIC   LRT Pr(>Chi)
<none>        0.72 40.2
sex     1     5.70 43.2  4.98    0.026 *
food    2    20.90 56.4 20.18  4.2e-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
    -1.6127     -0.3126     -0.6693     -0.1725
```

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. Similarly we can calculate the change in odds ratio for the feeding types:

```
exp( coef(m1) )

(Intercept)     sexGirl  foodBreast   foodSuppl
     0.1993      0.7316      0.5121      0.8415
```

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) )

             2.5 % 97.5 %
(Intercept) 0.1592 0.2474
sexGirl     0.5536 0.9629
foodBreast  0.3782 0.6895
foodSuppl   0.5555 1.2464
```

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.

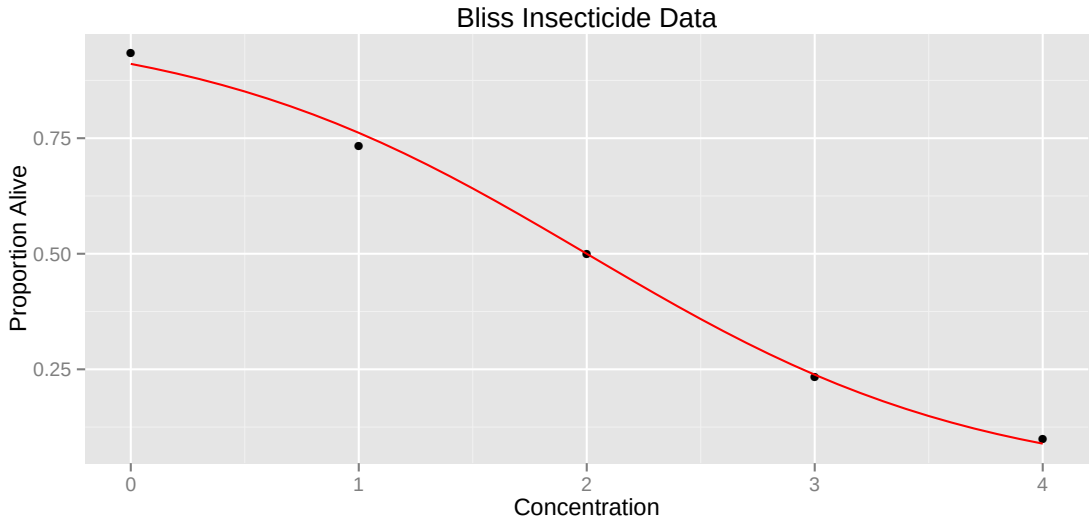
12.6 Prediction and Effective Dose Levels

To demonstrate the ideas in this section, we use a toxicology study that examined insect mortality as a function of increasing concentrations of an insecticide.

```
data(bliss)
bliss$prop.alive <- bliss$alive / (bliss$alive + bliss$dead)
bliss

  dead alive conc prop.alive
1    2    28    0     0.9333
2    8    22    1     0.7333
3   15    15    2     0.5000
4   23     7    3     0.2333
5   27     3    4     0.1000
```

We first fit the logistic regression model and plot the results

```
m1 <- glm( cbind(alive, dead) ~ conc, family=binomial, data=bliss)
probs <- data.frame(conc=seq(0,4,by=.1))
y.hat <- predict(m1, newdata=probs)  # the y values are in [-infty, infty]
probs$p.hat <- ilogit( y.hat )
ggplot(bliss, aes(x=conc)) +
  geom_point(aes(y=prop.alive)) +
  geom_line(data=probs, aes(y=p.hat), color='red') +
  opts(title='Bliss Insecticide Data') +
  xlab('Concentration') + ylab('Proportion Alive')
```



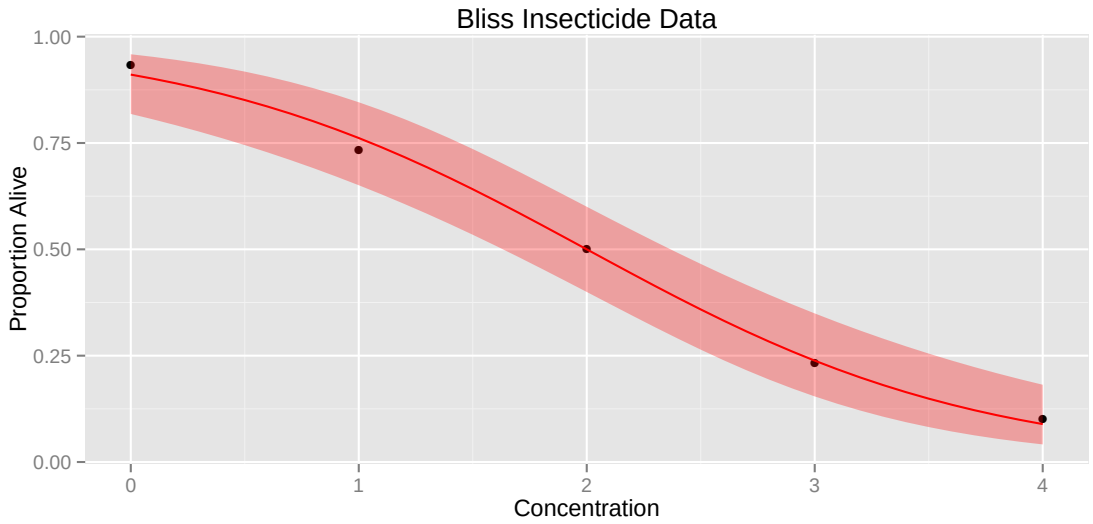
Given this, we want to develop a confidence interval for the probabilities by first calculating using the following formula. As usual, we recall that the y values live in $(-\infty, \infty)$.

$$CI_y: \quad \hat{y} \pm z^{1-\alpha/2}\, StdErr\,(\hat{y})$$

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))
y <- predict(m1, newdata=probs, se.fit=TRUE)  # two columns, fit and se.fit
probs$p.hat <- ilogit( y$fit                    )
probs$lwr   <- ilogit( y$fit - 1.96 * y$se.fit )
probs$upr   <- ilogit( y$fit + 1.96 * y$se.fit )
ggplot(bliss, aes(x=conc)) +
  geom_point(aes(y=prop.alive)) +
  geom_line(data=probs, aes(y=p.hat), color='red') +
  geom_ribbon(data=probs, aes(ymin=lwr, ymax=upr), fill='red', alpha=.3) +
  opts(title='Bliss Insecticide Data') +
  xlab('Concentration') + ylab('Proportion Alive')
```



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:

$$Var\left(g\left(\hat{\boldsymbol{\theta}}\right)\right) = g'\left(\boldsymbol{\theta}\right)^T Var\left(\boldsymbol{\theta}\right) g'\left(\boldsymbol{\theta}\right)$$

Fortunately we don't have to do these calculations by hand and can use the `dose.p()` function in the `MASS` package.

```
library(MASS)
LD <- dose.p(m1, p=c(.25, .5, .75))
LD

            Dose     SE
p = 0.25: 2.946 0.2316
p = 0.50: 2.000 0.1784
p = 0.75: 1.054 0.2316
```

and we can use these to create approximately confidence intervals for these $\hat{x}_p$ 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...
str(LD)

Class 'glm.dose'  atomic [1:3] 2.95 2 1.05
  ..- 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


# produce approximate 95% confidence intervals using
# this monstrosity of a data structure.
upr <- LD + qnorm(.975) * attr(LD,'SE')
lwr <- LD - qnorm(.975) * attr(LD,'SE')
CI <- cbind(lwr, upr)
colnames(CI) <- c('lwr','upr')
CI

              lwr   upr
p = 0.25: 2.4916 3.399
p = 0.50: 1.6503 2.350
p = 0.75: 0.6006 1.508
```

12.7 Overdispersion

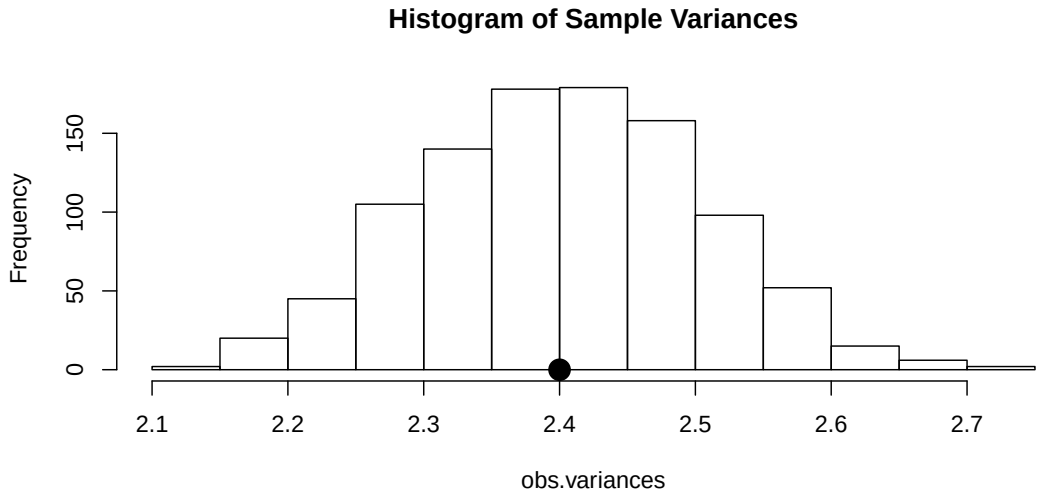
In the binomial distribution, the variance is a function of the probability of success and is

$$Var\left(W\right) = np\left(1-p\right)$$

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.

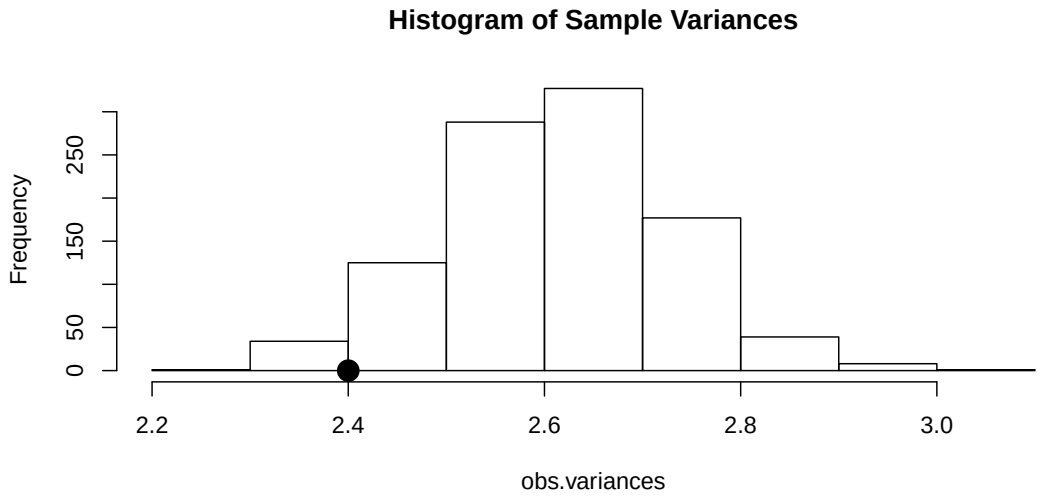
We can do a quick simulation to demonstrate that additional variability in p leads to addition variability overall.

```
N <- 1000
n <- 10
p <- .6
obs.variances <- rep(NA, N)
for( i in 1:N ){
  obs.variances[i] <- var(rbinom(N,size=n,prob=p))
}
true.var <- p*(1-p)*n
hist(obs.variances, main='Histogram of Sample Variances')
points(true.var, 0, cex=2, pch=19)
```



We see that the sample variances fall neatly about the true variance of 2.4. Next we do the same experiment, but add a small amount of random noise about the parameter p.

```
N <- 1000
n <- 10
p <- .6
p2 <- p + rnorm(N, mean=0, sd=0.05)
obs.variances <- rep(NA, N)
for( i in 1:N ){
  obs.variances[i] <- var(rbinom(N,size=n,prob=p2))
}
true.var <- p*(1-p)*n
hist(obs.variances, main='Histogram of Sample Variances')
points(true.var, 0, cex=2, pch=19)
```



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. Since 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. Since 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.

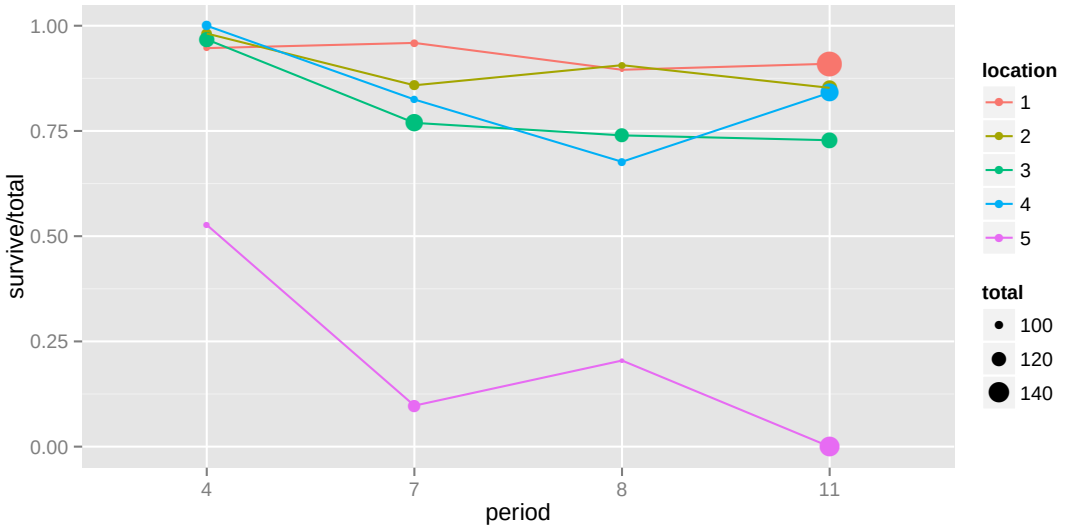
Example: We consider an experiment where at five different stream locations, four boxes of trout eggs were buried and retrieved at four different times after the original placement. The number of surviving eggs was recorded and the eggs disposed of.

```
data(troutegg)
troutegg$perish <- troutegg$total - troutegg$survive
ftable( xtabs(cbind(survive,perish)~location+period, data=troutegg) )
```

```
                survive perish
location period
1        4          89      5
         7          94      4
         8          77      9
         11        141     14
2        4         106      2
         7          91     15
         8          87      9
         11        104     18
3        4         119      4
         7         100     30
         8          88     31
         11         91     34
4        4         104      0
         7          80     17
         8          67     32
         11        111     21
5        4          49     44
         7          11    102
         8          18     70
         11          0    138
```

We can first visualize the data

```
p <- ggplot(troutegg, aes(x=period, y=survive/total, color=location)) +
  geom_point(aes(size=total)) +
  geom_line(aes(x=as.integer(period)))
print(p)
```



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:
   Min      1Q  Median      3Q     Max
-4.830  -0.365  -0.030   0.619   3.243

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    4.636      0.281   16.48  < 2e-16 ***
location2     -0.417      0.246   -1.69     0.09 .
location3     -1.242      0.219   -5.66  1.5e-08 ***
location4     -0.951      0.229   -4.16  3.2e-05 ***
location5     -4.614      0.250  -18.44  < 2e-16 ***
period7       -2.170      0.238   -9.10  < 2e-16 ***
period8       -2.326      0.243   -9.57  < 2e-16 ***
period11      -2.450      0.234  -10.47  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1021.469  on 19  degrees of freedom
Residual deviance:   64.495  on 12  degrees of freedom
AIC: 157

Number of Fisher Scoring iterations: 5
```

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] 3.372e-09
```

We therefore estimate the overdispersion parameter

```
sigma2 <- sum( residuals(m, type='pearson') ^2 ) / 12
sigma2

[1] 5.33
```

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:  F test assumes 'quasibinomial' family

Single term deletions

Model:
cbind(survive, perish) ~ location + period

scale:  5.33

         Df Deviance AIC F value  Pr(>F)
<none>            64 157
location  4      914 308    39.5 8.1e-07 ***
period    3      229 182    10.2  0.0013 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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
-4.830  -0.365  -0.030   0.619   3.243

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.636      0.649    7.14  1.2e-05 ***
location2     -0.417      0.568   -0.73  0.47732
location3     -1.242      0.507   -2.45  0.03050 *
location4     -0.951      0.528   -1.80  0.09697 .
location5     -4.614      0.578   -7.99  3.8e-06 ***
period7       -2.170      0.550   -3.94  0.00195 **
period8       -2.326      0.561   -4.15  0.00136 **
period11      -2.450      0.540   -4.53  0.00069 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasibinomial family taken to be 5.33)

    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
location  4      914    39.5 8.1e-07 ***
period    3      229    10.2  0.0013 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

While each of the time periods is different than the first, it looks like periods 7,8, and 11 are different from each other. As usual, we need to turn to `glht()` to test this.

```
coef(m2)

(Intercept)   location2   location3   location4   location5     period7
     4.6358     -0.4168     -1.2421     -0.9509     -4.6138     -2.1702
    period8    period11
    -2.3256     -2.4500

library(multcomp)
cntr <- rbind('7 vs  8' = c(0, 0,0,0,0, 1,-1, 0),
              '7 vs 11' = c(0, 0,0,0,0, 1, 0,-1),
              '8 vs 11' = c(0, 0,0,0,0, 0, 1,-1))
summary( glht(m2, linfct=cntr) )


 Simultaneous Tests for General Linear Hypotheses

Fit: glm(formula = cbind(survive, perish) ~ location + period, family = quasibinomial,
    data = troutegg)

Linear Hypotheses:
             Estimate Std. Error z value Pr(>|z|)
7 vs  8 == 0    0.155      0.406    0.38     0.92
7 vs 11 == 0    0.280      0.379    0.74     0.74
8 vs 11 == 0    0.124      0.381    0.33     0.94
(Adjusted p values reported -- single-step method)
```

Looking at this experiment, 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, so hand calculation of the scale parameter is necessary and as are many of the test statistics.

```
library(lme4)
m3 <- lmer(cbind(survive,perish) ~ period + (1|location),
          family=binomial, data=troutegg)

Warning:  calling lmer with 'family' is deprecated; please use glmer() instead

summary(m3)

Generalized linear mixed model fit by maximum likelihood ['glmerMod']
 Family: binomial ( logit )
Formula: cbind(survive, perish) ~ period + (1 | location)
   Data: troutegg

     AIC      BIC   logLik deviance
  180.31   185.29   -85.15   170.31

Random effects:
 Groups   Name        Variance Std.Dev.
 location (Intercept) 2.68     1.64
Number of obs: 20, groups: location, 5

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    3.180      0.759    4.19  2.8e-05 ***
period7       -2.155      0.237   -9.10  < 2e-16 ***
period8       -2.309      0.241   -9.57  < 2e-16 ***
period11      -2.432      0.232  -10.47  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
         (Intr) period7 period8
period7  -0.224
period8  -0.224  0.731
period11 -0.234  0.757  0.760
```

Chapter 13

Random Effects

The assumption of independent observations is often not supported. 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.

Hierarchical models are most often considered within the context of mixed effects models where we distinguish between random effects and fixed effects.

Fixed effects are 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 are 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 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.

13.1 Review of Maximum Likelihood

13.1.1 Normally distributed data

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".

For example for a single observation from a normal distribution, $y \sim N\left(\mu, \sigma^2\right)$, the probability density function is

$$P\left(y|\mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right]$$

where $\exp[u] = e^u$. If we take a sample of n independent and identically $N\left(\mu, \sigma^2\right)$ distributed observations, the joint probability function is

$$\begin{aligned}
P\left(y_1, \ldots, y_n|\mu, \sigma^2\right) &= \prod_{i=1}^{n}\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_i-\mu)^2}{2\sigma^2}\right]\right) \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left[\sum_{i=1}^{n}\left(-\frac{(y_i-\mu)^2}{2\sigma^2}\right)\right] \\
&= \frac{1}{(2\pi)^{n/2}\left(\sigma^2\right)^{n/2}} \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2\right] \\
&= \frac{1}{(2\pi)^{n/2}\left[\det\left(\sigma^2\boldsymbol{I}_n\right)\right]^{1/2}} \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{\mu})^T(\boldsymbol{y}-\boldsymbol{\mu})\right] \\
&= \frac{1}{(2\pi)^{n/2}\left[\det\left(\sigma^2\boldsymbol{I}_n\right)\right]^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T\left(\sigma^{-2}\boldsymbol{I}_n\right)(\boldsymbol{y}-\boldsymbol{\mu})\right] \\
&= \frac{1}{(2\pi)^{n/2}\left[\det\left(\boldsymbol{\Omega}\right)\right]^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T\boldsymbol{\Omega}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right]
\end{aligned}$$

where $\boldsymbol{I}_n$ is the $n \times n$ identity matrix and $\boldsymbol{\Omega}$ is the variance-covariance matrix of our observations and $\det(\boldsymbol{\Omega})$ is the determinant of $\boldsymbol{\Omega}$. If we regard this as a function of our parameters μ and σ^2 then we have defined our likelihood function

$$\mathcal{L}\left(\mu, \sigma^2|y_1, \ldots, y_n\right) = \frac{1}{(2\pi)^{n/2}\left[\det\left(\boldsymbol{\Omega}\right)\right]^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T\boldsymbol{\Omega}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right] \tag{13.1.1}$$

which we can use 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 this 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

$$\hat{\sigma}^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y})^2$$

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.

13.1.2 Likelihood Ratio Test

In general (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 maximum likelihoods functions of the two models $L(\boldsymbol{\theta}_0)$ and $L(\boldsymbol{\theta}_1)$ 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.

Since 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[1].

13.2 Estimation of Multiple Random Effects

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 $\left(\sigma^2_\epsilon\right)$.

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

$$\rho = \frac{\sigma^2_\gamma}{\sigma^2_\gamma + \sigma^2_\epsilon}$$

[1] Another alternative is to examine the bootstrap distribution of D.

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 13.1.1, 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

$$\boldsymbol{Z} = \begin{bmatrix} 1 & \cdot & \cdot \\ 1 & \cdot & \cdot \\ 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \\ \cdot & 1 & \cdot \\ \cdot & 1 & \cdot \\ \cdot & \cdot & 1 \\ \cdot & \cdot & 1 \\ \cdot & \cdot & 1 \end{bmatrix} \qquad ZZ^T = \begin{bmatrix} 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 \end{bmatrix}$$

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 suboptimal. If we knew that $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{0}$ and could use that fact and have an unbiased estimate of our variance parameters. Since $\boldsymbol{X}$ is known, we can find linear functions $\boldsymbol{k}$ of $\boldsymbol{X}$ 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

$$\boldsymbol{K}^T\boldsymbol{y} \sim N\left(\boldsymbol{K}^T\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{K}^T\boldsymbol{\Omega}\boldsymbol{K}\right) = N\left(\boldsymbol{0}, \boldsymbol{K}^T\boldsymbol{\Omega}\boldsymbol{K}\right)$$

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.

Example 1. Our example comes from an experiment to test the paper brightness as affected by the shift operator. The data has 20 observations with 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).

```
library(faraway)
data(pulp)
# 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   0.447     4.2  0.023 *
Residuals   16   1.70   0.106
---
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 $\texttt{operator4} = -(-0.16 - 0.34 + 0.22) = 0.28$

To fit the random effects model we will use the package `lme4` which stands for Linear Mixed Effects and the 4 represents that the package is written in the S4 dialect of R (which means a few things will be new to us)

```
library(lme4)
m2 <- lmer( bright ~ 1 + (1|operator), data=pulp )
summary(m2)

Linear mixed model fit by REML ['lmerMod']
Formula: bright ~ 1 + (1 | operator)
   Data: pulp

REML criterion at convergence: 18.63

Random effects:
 Groups   Name        Variance Std.Dev.
 operator (Intercept) 0.0681   0.261
 Residual             0.1062   0.326
Number of obs: 20, groups: operator, 4

Fixed effects:
            Estimate Std. Error t value
(Intercept)   60.400      0.149     404
```

Notice that the estimate of the fixed effect (the overall mean is the same in the fixed-effects ANOVA and in the mixed model, but 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

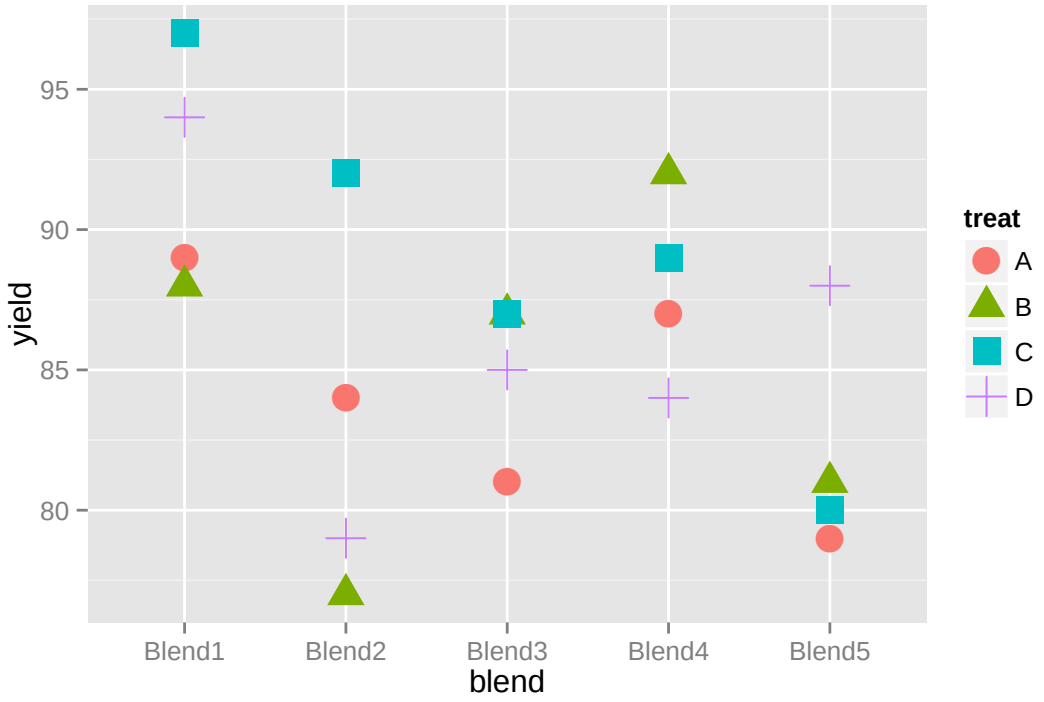
```
ranef(m2)

$operator
  (Intercept)
a     -0.1219
b     -0.2591
c      0.1677
d      0.2134

attr(,"class")
[1] "ranef.mer"
```

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.

13.3 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. We illustrate this with a manufacturing example from the pharmaceutical industry. We are interested in 4 different processes 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)
library(ggplot2)
ggplot(penicillin, aes(y=yield, x=blend, color=treat, shape=treat)) +
  geom_point(size=5)
```



We can analyze this data using mixed models with the blend as a random effect and treatment as a fixed effect.

```
m1 <- lmer( yield ~ treat + (1|blend), data=penicillin )
summary(m1)

Linear mixed model fit by REML ['lmerMod']
Formula: yield ~ treat + (1 | blend)
   Data: penicillin

REML criterion at convergence: 103.8

Random effects:
 Groups   Name        Variance Std.Dev.
 blend    (Intercept) 11.8     3.43
 Residual             18.8     4.34
Number of obs: 20, groups: blend, 5

Fixed effects:
            Estimate Std. Error t value
(Intercept)    84.00       2.48    33.9
treatB          1.00       2.75     0.4
treatC          5.00       2.75     1.8
treatD          2.00       2.75     0.7

Correlation of Fixed Effects:
       (Intr) treatB treatC
treatB -0.555
treatC -0.555  0.500
treatD -0.555  0.500  0.500
```

As we see, the variability from the different blends $\left(\sigma^2_\gamma\right)$ is a similar size as the pure error $\left(\sigma^2_\epsilon\right)$. This implies that if we could reduce the variability in the blend of nutrients, we would substantially decrease the variability in yield.

The fixed effects in the model show that the various treatments (the different methods of biosynthesis) shows that method `C` seems to be the best while method `A` appears to be the worst. However we notice that the t-value comparing `C` to `A` is quite small (smaller than 2). However, there is no p-value to compare. The reason for this is that in mixed models it is not always clear what the appropriate degrees of freedom are.[2]

To test if the fixed effects are significant, we will use the Likelihood Ratio Test. However, since the fixed effects differ between the two models, we should not use the REML method when fitting the models (since REML essentially removes the fixed effects before maximization).

[2]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. A previous version of this software (package `nlme`) will produce p-values but at the cost of flexibility in the types of mixed models that can be fit.

```
m0.ml<-lmer(yield~      (1|blend),data=penicillin,REML=FALSE) # Must use ML for fixed
m1.ml<-lmer(yield~treat+(1|blend),data=penicillin,REML=FALSE) # effect comparisons!!!
anova(m0.ml, m1.ml)

Data: penicillin
Models:
m0.ml: yield ~ (1 | blend)
m1.ml: yield ~ treat + (1 | blend)
      Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)
m0.ml  3 127 130  -60.7      121
m1.ml  6 129 135  -58.6      117  4.05      3       0.26
```

This shows that the treatment effect (ie biosynthesis method) is not significant and none of the methods provide a statistically significant improvement.

13.4 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.

For example, 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 subsamples and measures the fat content twice within a subsample. So our hierarchy is that observations are nested within subsamples 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 subsamples are nested within the technicians and we denote them as S_{ijk}. Finally our "pure" error is the two measurements from the same subsample. 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 subsamples and 48 "pure" random effects.

```
data(eggs)
str(eggs)

'data.frame': 48 obs. of  4 variables:
 $ Fat       : num  0.62 0.55 0.34 0.24 0.8 0.68 0.76 0.65 0.3 0.4 ...
 $ Lab       : Factor w/ 6 levels "I","II","III",..: 1 1 1 1 1 1 1 1 2 2 ...
 $ Technician: Factor w/ 2 levels "one","two": 1 1 1 1 2 2 2 2 1 1 ...
 $ Sample    : Factor w/ 2 levels "G","H": 1 1 2 2 1 1 2 2 1 1 ...

eggs$Lab : eggs$Technician

 [1] I:one   I:one   I:one   I:one   I:two   I:two   I:two   I:two
 [9] II:one  II:one  II:one  II:one  II:two  II:two  II:two  II:two
[17] III:one III:one III:one III:one III:two III:two III:two III:two
[25] IV:one  IV:one  IV:one  IV:one  IV:two  IV:two  IV:two  IV:two
[33] V:one   V:one   V:one   V:one   V:two   V:two   V:two   V:two
[41] VI:one  VI:one  VI:one  VI:one  VI:two  VI:two  VI:two  VI:two
12 Levels: I:one I:two II:one II:two III:one III:two IV:one ... VI:two
```

Now that we know how to create the interaction, we could do the same for the subsample.

```
m2 <- lmer( Fat ~ 1 + (1|Lab) + (1|Lab:Technician) +
                (1|Lab:Technician:Sample), data=eggs)
summary(m2)

Linear mixed model fit by REML ['lmerMod']
Formula: Fat ~ 1 + (1 | Lab) + (1 | Lab:Technician) + (1 | Lab:Technician:Sample)
   Data: eggs

REML criterion at convergence: -64.24

Random effects:
 Groups                Name        Variance Std.Dev.
 Lab:Technician:Sample (Intercept) 0.00306  0.0554
 Lab:Technician        (Intercept) 0.00698  0.0835
 Lab                   (Intercept) 0.00592  0.0769
 Residual                          0.00720  0.0848
Number of obs: 48, groups: Lab:Technician:Sample, 24; Lab:Technician, 12; Lab, 6

Fixed effects:
            Estimate Std. Error t value
(Intercept)    0.388      0.043    9.02
```

Our random effects are all approximately the same size. We might consider if we could simplify our model by removing the subsample random effect. Since the fixed effects are the same for both models, we can still use the REML method for fitting the estimates.

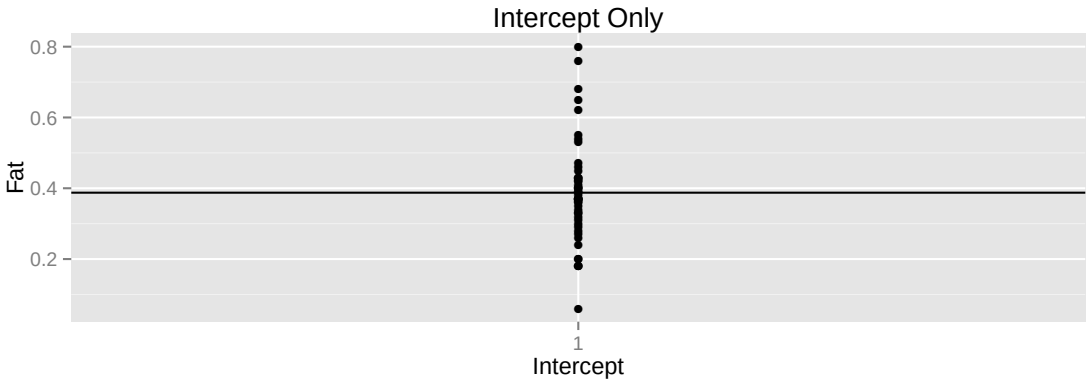
```
m2 <- lmer( Fat ~ 1 + (1|Lab) + (1|Lab:Technician) +
                (1|Lab:Technician:Sample), data=eggs)
m1 <- lmer( Fat ~ 1 + (1|Lab) + (1|Lab:Technician), data=eggs)
summary(m1)

Linear mixed model fit by REML ['lmerMod']
Formula: Fat ~ 1 + (1 | Lab) + (1 | Lab:Technician)
   Data: eggs

REML criterion at convergence: -62.63

Random effects:
 Groups         Name        Variance Std.Dev.
 Lab:Technician (Intercept) 0.00800  0.0895
 Lab            (Intercept) 0.00592  0.0769
 Residual                   0.00924  0.0961
Number of obs: 48, groups: Lab:Technician, 12; Lab, 6

Fixed effects:
            Estimate Std. Error t value
(Intercept)    0.388      0.043    9.02

anova(m1, m2)

Data: eggs
Models:
m1: Fat ~ 1 + (1 | Lab) + (1 | Lab:Technician)
m2: Fat ~ 1 + (1 | Lab) + (1 | Lab:Technician) + (1 | Lab:Technician:Sample)
   Df   AIC   BIC logLik deviance Chisq Chi Df Pr(>Chisq)
m1  4 -59.2 -51.7   33.6    -67.2
m2  5 -58.8 -49.4   34.4    -68.8   1.6      1       0.21
```

Notice the random effects in this smaller model are larger and the subsample random effect is being subsumed into other random effects. The LRT suggests that we cannot reject the null hypothesis that $H_0 : \sigma_S^2 = 0$. Finally we see if we can remove the technician random effect.

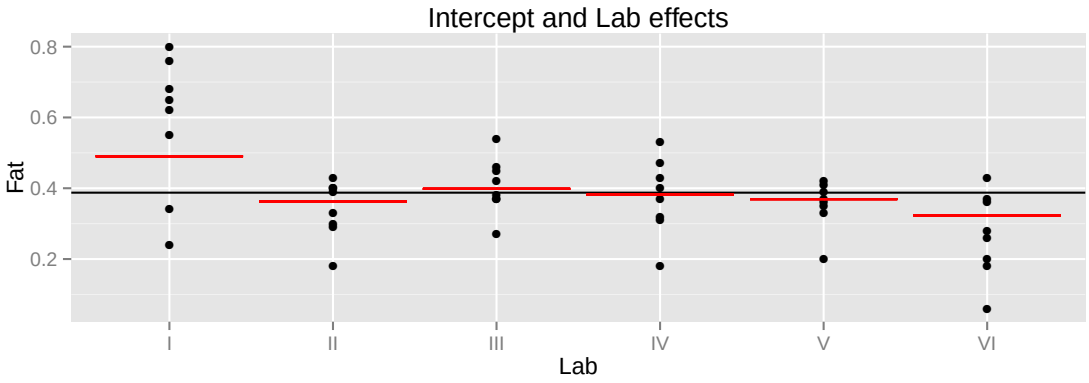
```
m0 <- lmer( Fat ~ 1 + (1|Lab), data=eggs)
anova(m0, m1)

Data: eggs
Models:
m0: Fat ~ 1 + (1 | Lab)
m1: Fat ~ 1 + (1 | Lab) + (1 | Lab:Technician)
   Df   AIC   BIC logLik deviance Chisq Chi Df Pr(>Chisq)
m0  3 -53.3 -47.7   29.6    -59.3
m1  4 -59.2 -51.7   33.6    -67.2  7.91      1     0.0049 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on these p-values, the technician random effect is significant and we conclude that `m1` is the simplest model we should work with.

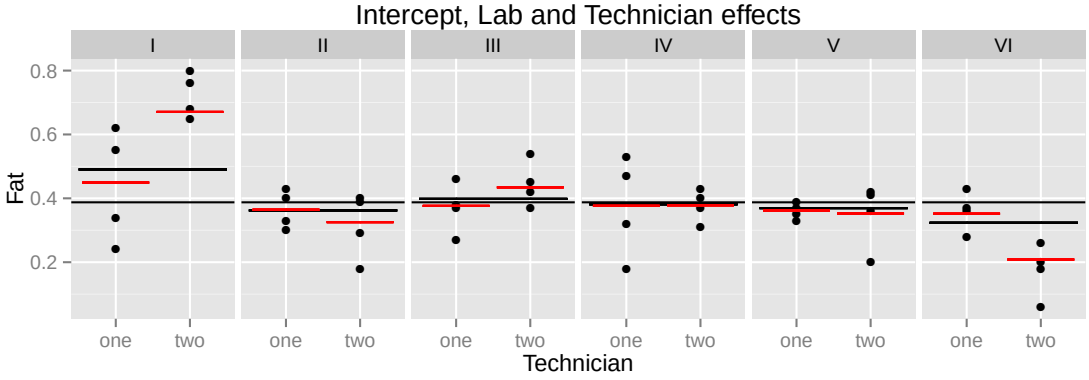
To help visualize the effects present in the model we consider the following. First we examine just the intercept model.



Next we split the observations into their respective labs and display the lab effect.



Finally we separate out the individual technicians and display their effects.



13.5 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. Recall that we had a machine to do the abrasion test with four positions and we did 4

different machine runs. 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)
m <- lmer( wear ~ material + (1|run) + (1|position), data=abrasion)
summary(m)

Linear mixed model fit by REML ['lmerMod']
Formula: wear ~ material + (1 | run) + (1 | position)
   Data: abrasion

REML criterion at convergence: 100.3

Random effects:
 Groups   Name        Variance Std.Dev.
 run      (Intercept)  66.9     8.18
 position (Intercept) 107.1    10.35
 Residual              61.2     7.83
Number of obs: 16, groups: run, 4; position, 4

Fixed effects:
            Estimate Std. Error t value
(Intercept)   265.75       7.67    34.7
materialB     -45.75       5.53    -8.3
materialC     -24.00       5.53    -4.3
materialD     -35.25       5.53    -6.4

Correlation of Fixed Effects:
          (Intr) matrlB matrlC
materialB -0.361
materialC -0.361  0.500
materialD -0.361  0.500  0.500
```

We might ask how to calculate the other contrasts of interest, for example `B` vs `C`.

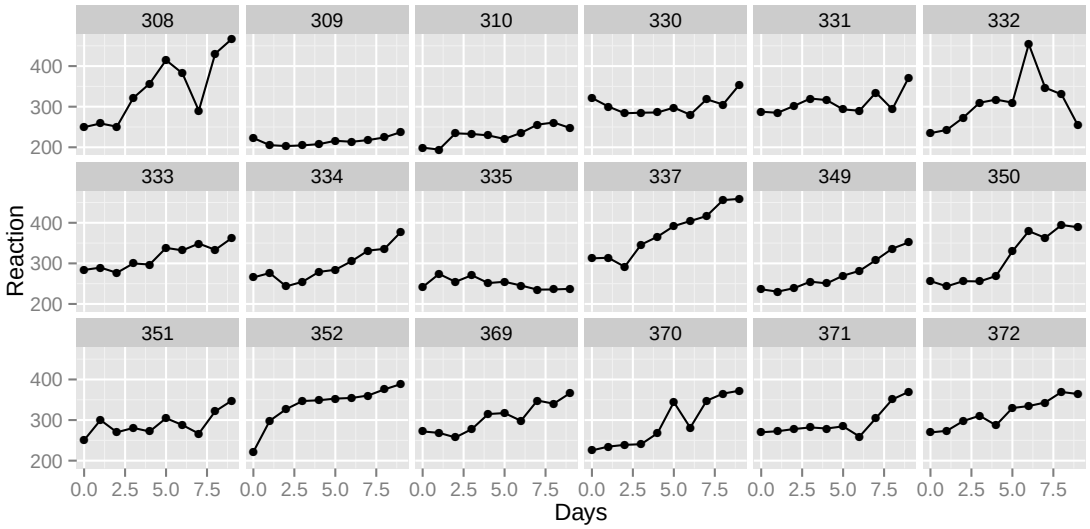
```
library(multcomp)
contr <- rbind('B - A' = c(0,  1,  0,  0),
               'C - A' = c(0,  0,  1,  0),
               'D - A' = c(0,  0,  0,  1),
               'C - B' = c(0, -1,  1,  0),
               'D - B' = c(0, -1,  0,  1),
               'D - C' = c(0,  0, -1,  1))
test <- glht(m, linfct=contr)
summary(test)


 Simultaneous Tests for General Linear Hypotheses

Fit: lmer(formula = wear ~ material + (1 | run) + (1 | position),
    data = abrasion)

Linear Hypotheses:
           Estimate Std. Error z value Pr(>|z|)
B - A == 0   -45.75       5.53   -8.27   <0.001 ***
C - A == 0   -24.00       5.53   -4.34   <0.001 ***
D - A == 0   -35.25       5.53   -6.37   <0.001 ***
C - B == 0    21.75       5.53    3.93   <0.001 ***
D - B == 0    10.50       5.53    1.90     0.23
D - C == 0   -11.25       5.53   -2.03     0.18
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

13.6 Repeated Measures and Longitudinal Data

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.

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)
p <- ggplot(sleepstudy, aes(y=Reaction, x=Days)) +
    facet_wrap(~ Subject, ncol=6) +
    geom_point() +
    geom_line()
print(p)
```

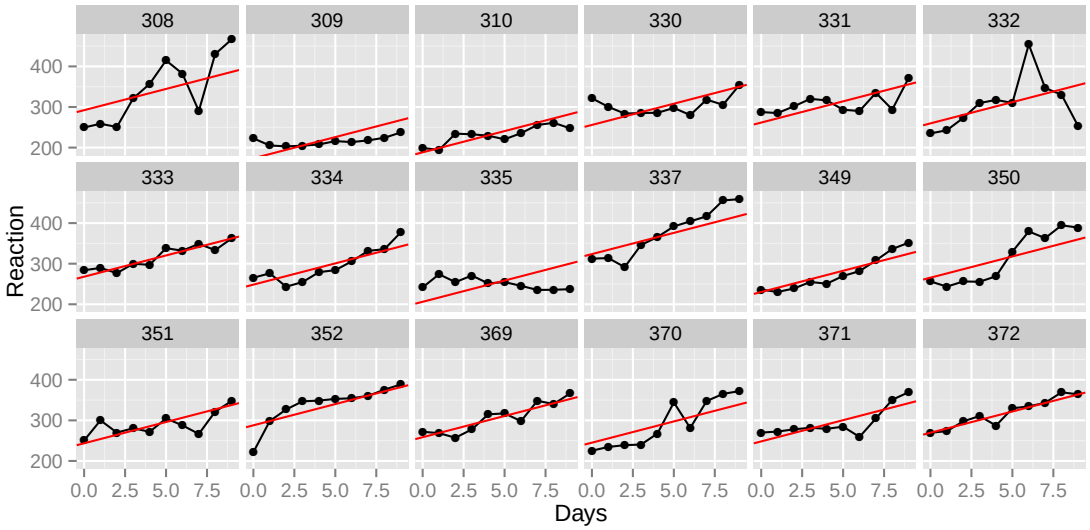


We want to fit a line to these data, but how should we do this? First we notice that each subject has their own baseline for reaction time and the subsequent measurements are relative to this, so it is clear that we should fit a model with a random intercept.

```
m1 <- lmer( Reaction ~ Days + (1|Subject), data=sleepstudy)
summary(m1)

Linear mixed model fit by REML ['lmerMod']
Formula: Reaction ~ Days + (1 | Subject)
   Data: sleepstudy

REML criterion at convergence: 1786

Random effects:
 Groups   Name        Variance Std.Dev.
 Subject  (Intercept) 1378     37.1
 Residual              960     31.0
Number of obs: 180, groups: Subject, 18

Fixed effects:
            Estimate Std. Error t value
(Intercept)  251.405      9.747    25.8
Days          10.467      0.804    13.0

Correlation of Fixed Effects:
     (Intr)
Days -0.371

fixef(m1)

(Intercept)        Days
     251.41       10.47

ranef(m1)

$Subject
    (Intercept)
308      40.784
309     -77.850
310     -63.109
330       4.406
331      10.216
332       8.221
333      16.500
334      -2.997
335     -45.282
337      72.183
349     -21.196
350      14.111
351      -7.862
352      36.378
369       7.036
370      -6.363
371      -3.294
372      18.116

attr(,"class")
[1] "ranef.mer"
```

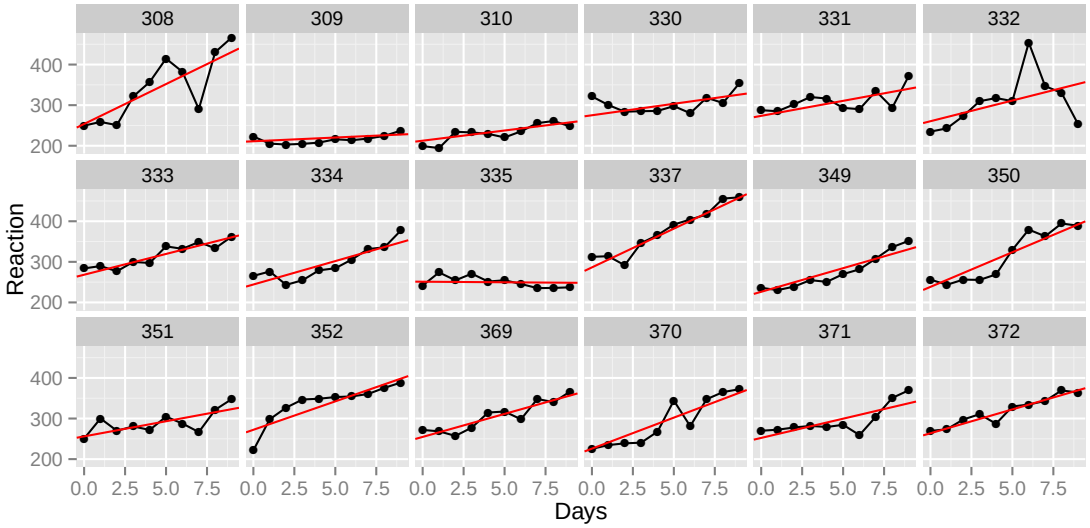
To visualize how well this model fits our data, we will plot the predicted values which are lines with y-intercepts that are equal to the sum of the fixed effect of intercept and the random intercept per subject. The slope for each patient is assumed to be the same and is approximately 10.4.

```
yhats1 <- data.frame(
    Subject = factor(rownames( ranef(m1)$Subject )),
    intercept = fixef(m1)[1] + ranef(m1)$Subject[,1],
    slope = fixef(m1)[2] )

Warning:  row names were found from a short variable and have been discarded

p1 <- p + geom_abline(data=yhats1, aes(intercept=intercept, slope=slope),
                      color='red')
print(p1)
```



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)
yhats2 <- data.frame(
    Subject = factor(rownames( ranef(m2)$Subject )),
    intercept = fixef(m2)[1] + ranef(m2)$Subject[,1],
    slope = fixef(m2)[2] + ranef(m2)$Subject[,2])
p2 <- p + geom_abline(data=yhats2, aes(intercept=intercept, slope=slope),
                      color='red')
print(p2)
```



Now we can do a test to see if this is better than the random intercept model. Notice because the fixed effects are identical between the two models, we can continue to use REML estimates.

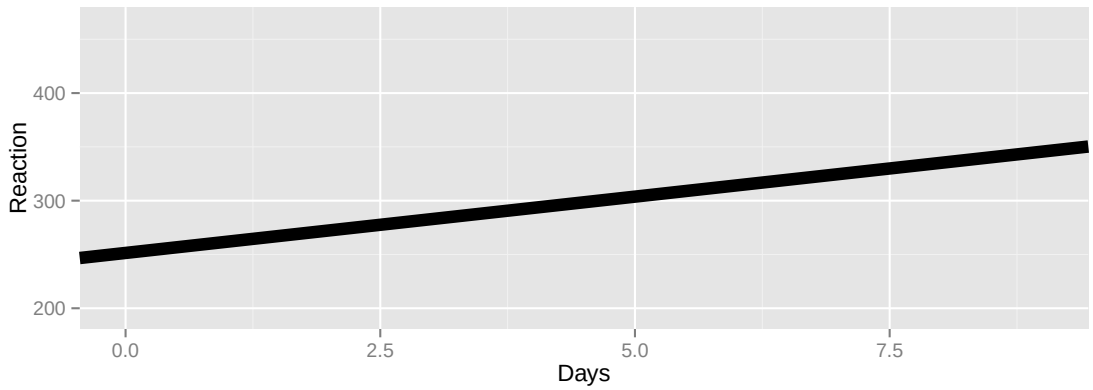
```
m1 <- lmer( Reaction ~ Days + (1     |Subject), data=sleepstudy)
m2 <- lmer( Reaction ~ Days + (1+Days|Subject), data=sleepstudy)
anova(m1, m2)

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 1815   -897     1794
m2  6 1764 1783   -876     1752  42.1      2    7.1e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This looks much better and we definitively reject the null model in favor of the complex. Finally we examine the summary of our chosen model.

```
summary(m2)

Linear mixed model fit by REML ['lmerMod']
Formula: Reaction ~ Days + (1 + Days | Subject)
   Data: sleepstudy

REML criterion at convergence: 1744

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 Subject  (Intercept) 612.1    24.74
          Days         35.1     5.92    0.07
 Residual             654.9    25.59
Number of obs: 180, groups: Subject, 18

Fixed effects:
            Estimate Std. Error t value
(Intercept)   251.41       6.82    36.8
Days           10.47       1.55     6.8

Correlation of Fixed Effects:
     (Intr)
Days -0.138
```

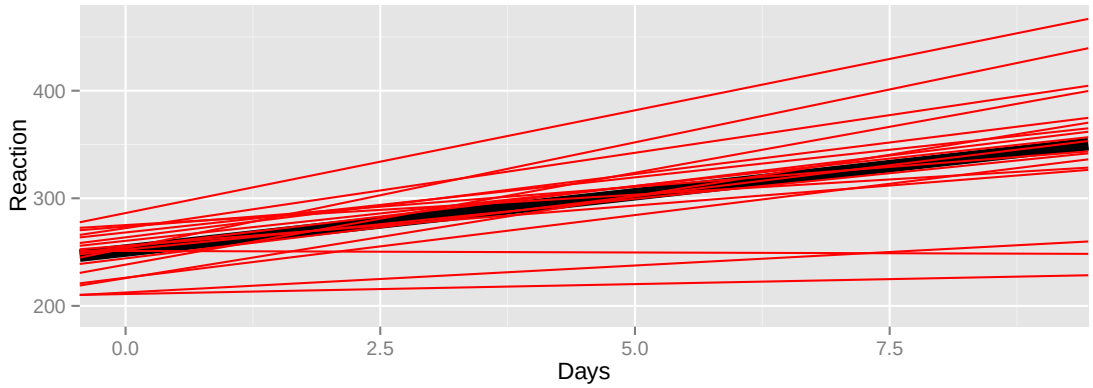
It is instructive to look at this example from the top down. First we plot the population regression line.

```
ggplot(sleepstudy, aes(x=Days, y=Reaction)) +
   geom_point(size=0) +  # do not show the raw data...
   geom_abline(intercept=fixef(m2)[1], slope=fixef(m2)[2], size=3)
```



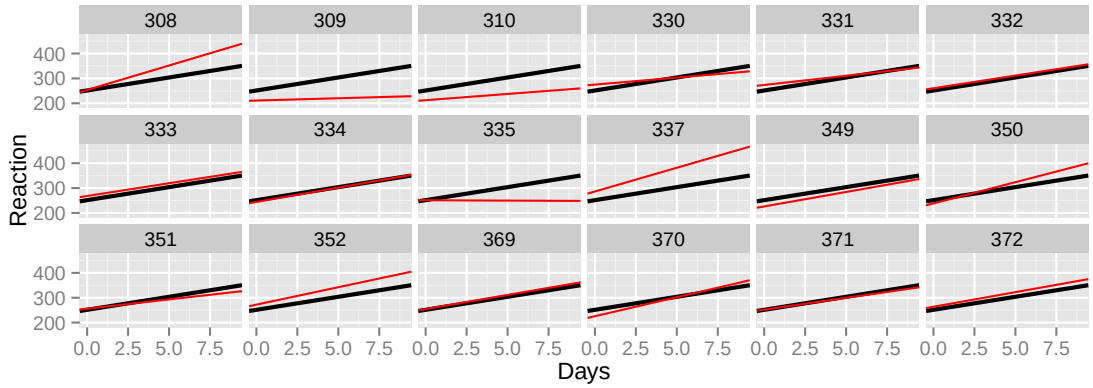
Next we add the random individual offsets from that population line...

```
ggplot(sleepstudy, aes(x=Days, y=Reaction)) +
    geom_point(size=0) +  # do not show the raw data...
    geom_abline(intercept=fixef(m2)[1], slope=fixef(m2)[2], size=3) +
    geom_abline(data=yhats2, aes(intercept=intercept, slope=slope), color='red')
```



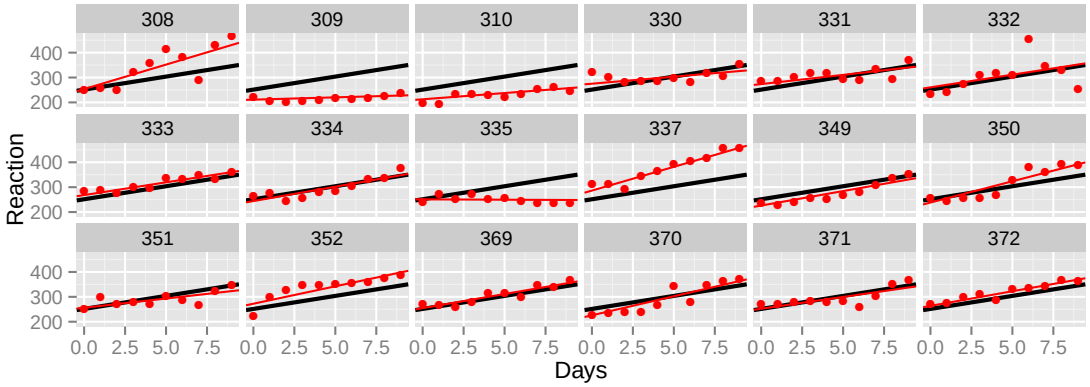
Separate them to their own individual graphs...

```
ggplot(sleepstudy, aes(x=Days, y=Reaction)) +
    geom_point(size=0) +  # do not show the raw data...
    geom_abline(intercept=fixef(m2)[1], slope=fixef(m2)[2], size=1) +
    geom_abline(data=yhats2, aes(intercept=intercept, slope=slope), color='red') +
    facet_wrap( ~ Subject, ncol=6)
```



And add the individual random observations about the subject's random regression line.

```
ggplot(sleepstudy, aes(x=Days, y=Reaction)) +
    geom_abline(intercept=fixef(m2)[1], slope=fixef(m2)[2], size=1) +
    geom_abline(data=yhats2, aes(intercept=intercept, slope=slope), color='red') +
    geom_point(color='red') +
    facet_wrap( ~ Subject, ncol=6)
```



Chapter 14

Bootstrapping

Recall that a confidence interval is really attempting assess how much the sample mean varies from sample to sample. Once we know how much it varies, we then will infer that the true mean is within some distance of the sample mean based on that variability.

Traditional statistical methods use the normality assumption to calculate these confidence intervals. In the case of estimating a population mean μ we calculate

$$\bar{x} \pm \underbrace{z^{1-\alpha/2}\left(\frac{\sigma}{\sqrt{n}}\right)}_{how\ \bar{x}\ varies}$$

and the z quantile controls the confidence level of the interval.

14.1 The Parametric Bootstrap: What to do if the error distribution is known

To illustrate the parametric bootstrap method, we consider the case of estimating the population mean μ from a sample of n independent and identically distributed observations

$$x_i = \mu + \epsilon_i$$

where ϵ_i has some known distribution G (which we can sample from). We will use the sample mean $\bar{x}$ to estimate the population mean μ, but we are left with the question of "How much does the sample mean vary from sample to sample?" One answer to that question is to take a sample of n observations from the distribution G (which I'll call ϵ_i^*) and calculate a new set of data

$$x_i^* = \bar{x} + \epsilon_i^*$$

and the mean of this new set of data $\bar{x}^*$ will vary about $\bar{x}$ in the same way that $\bar{x}$ varies about μ. Since I could create thousands of these datasets (and thus sample means $\bar{x}^*$), then I could create a distribution of how these means vary about $\bar{x}$. The middle 95% of that distribution is an excellent confidence interval for the population mean μ.

Example

For this example we pick an error distribution has extremely heavy tails, the Cauchy distribution[1].

[1]The Cauchy distribution is a t-distribution with 1 degree of freedom. It has extremely heavy tails and is, in general, a very extreme distribution.

```
n  <- 12
mu <- 35
x  <- mu + rcauchy(n)
x.bar <- mean(x)
obs.means <- rep(NA, 1000)
for( i in 1:1000 ){
  x.star <- x.bar + rcauchy(n)
  obs.means[i] <- mean(x.star)
}
quantile( obs.means, c(0.025, 0.975) )  # 95% CI - percentile method
```

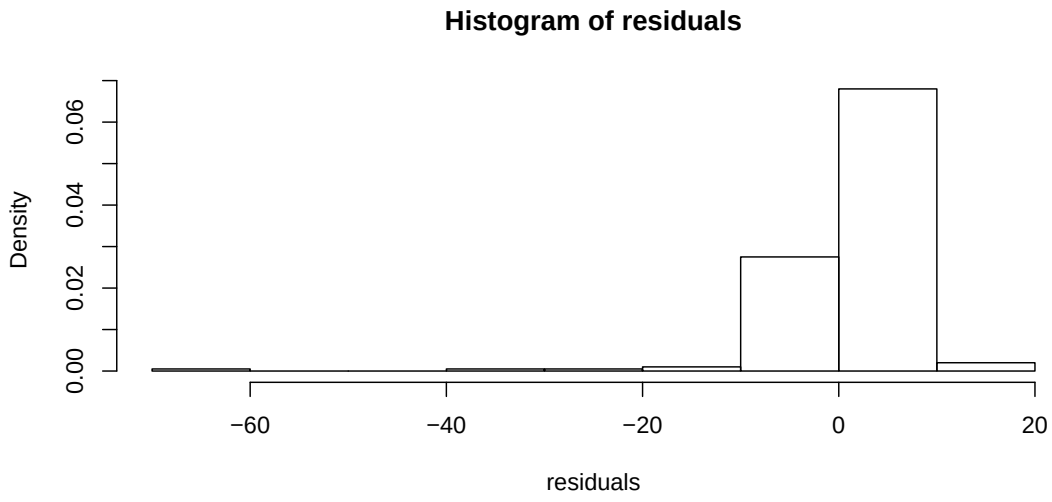
```
 2.5% 97.5%
23.07 48.72
```

This gives us a very convenient numerical way of calculating a confidence interval for μ even when there isn't a good theoretical interval.

14.2 The Nonparametric Residual Bootstrap: What to do if the error distribution is not known

Unfortunately we are almost never in the situation where we know what the error distribution is so sampling is not easily done. Instead what we will do is use an *estimate* of the error distribution and sample from that. But where do we get an estimate of the distribution?

```
n  <- 200
mu <- 35
x  <- mu + rcauchy(n)
x.bar <- mean(x)
residuals <- x - x.bar
hist(residuals, freq=FALSE)
```



Since we have the observed residuals, we can *use those as the estimated distribution*. This seems disturbingly simple and feels like it is cheating, but this actually works beautifully.

From a practical standpoint, how should I sample from this observed distribution? I need to generate n error terms, and I have only n residuals. To do this, we will sample *with replacement* from the residuals. What this means is that the first error term is selected at random from the n residuals and then the second is also selected at random from the same set of n residuals. This means that we most likely will have some residuals sampled twice (or more than twice).

To do the sampling in R, we will use the `sample()` function. It takes a vector of values, the number of elements to sample, and an optional argument that specifies if we should sample with or without replacement.

```
sample(1:5, 3)

[1] 3 4 5

sample(1:5, 5)

[1] 1 4 3 2 5

sample(1:5, 5, replace=TRUE)

[1] 2 3 4 3 3
```

We can use these as a set of indices.

```
example.vector <- c(-.25, .43, .1, -.18, -.1)
index <- sample(1:5, 5, replace=TRUE)
index

[1] 3 4 5 4 4

example.vector[index]

[1]  0.10 -0.18 -0.10 -0.18 -0.18
```

We now use this resampling of the observed residuals as a sample estimated error distribution.

```
n  <- 12
mu <- 35
x  <- mu + rcauchy(n)
x.bar <- mean(x)
residuals <- x - x.bar
obs.means <- rep(NA, 1000)
for( i in 1:1000 ){
  index <- sample(1:n, n, replace=TRUE)
  x.star <- x.bar + residuals[index]
  obs.means[i] <- mean(x.star)
}
quantile( obs.means, c(0.025, 0.975) )  # 95% CI -percentile method

 2.5% 97.5%
34.54 36.05
```

Example - ANOVA

Suppose that we wish to do an ANOVA using bootstrapping. In particular, we are interested in creating bootstrap confidence intervals for the group means.

```
n  <- 8              # number of observations in each group
mu <- c(20, 24, 28)  # group means
y  <- rep(mu, each=n) + rnorm(3*n)
groups <- factor(rep(c('A','B','C'), each=n))
```

Next we have to calculate the group means. I will do this by calling the `analysis of variance function aov()` and get the coefficients.

```
model <- aov( y ~ -1+groups)
fitted <- fitted(model)
residuals   <- resid(model)
```

We now have calculated the group means and the residuals so we can now calculate our bootstrap replicates and calculate the group means for each replicated data set.

```
replicate.means <- array(NA, dim=c(1000, 3))
for(i in 1:1000){
  index <- sample( 1:24, 24, replace=TRUE )
  y.star <- fitted + residuals[index]
  model <- aov( y.star ~ -1+groups )
  replicate.means[i,] <- coef(model)
}
str(replicate.means)

 num [1:1000, 1:3] 20.1 20.4 20 20.5 20.5 ...
```

Now that we have the replicated group means calculated, we just need to calculate the $(0.025, 0.975)$ quantiles for each column, giving us the bootstrap confidence intervals for the group means.

```
quantile(replicate.means[,1], probs=c(0.025,0.975))

 2.5% 97.5%
19.54 20.94

quantile(replicate.means[,2], probs=c(0.025,0.975))

 2.5% 97.5%
23.09 24.54

quantile(replicate.means[,3], probs=c(0.025,0.975))

 2.5% 97.5%
26.95 28.35
```

Instead of calling the quantile function for each column, I could have used the `apply()` function which will apply a given function to each row or column of a matrix. Instead of the prior code, I could have used:

```
apply(replicate.means, MARGIN=2, FUN=quantile, probs=c(0.025, 0.975))

       [,1]  [,2]  [,3]
2.5%  19.54 23.09 26.95
97.5% 20.94 24.54 28.35
```

14.3 Different methods for calculating CIs

Suppose that we are interested in estimating θ with some estimator $\hat{\theta}$. The usual confidence interval of

$$\hat{\theta} \pm z^{1-\alpha/2} \, StdDev\left(\hat{\theta}\right)$$

requires knowing the standard deviation of your estimator and knowing that

$$\frac{\hat{\theta} - \theta}{StdDev\left(\hat{\theta}\right)} \sim N(0,1)$$

or is approximately normally distributed. The bootstrap could be used to address either one (or both) of these requirements. As such, there are a several different methods for calculating confidence intervals. We will outline several methods for producing confidence intervals (in the order of most assumptions to fewest).

14.3.1 Normal intervals

This confidence interval assumes the sampling distribution of $\hat{\theta}$ is normal. We can use the bootstrap replicate samples to get an estimate of the standard error of the statistic of interest by just calculating the sample standard deviation of the replicated statistics.

Let $\theta()$ be the statistic of interest and $\hat{\theta}$ be the value of that statistic calculated from the observed data. Define $\hat{SE}$ as the sample standard deviation of statistic replicates.

Our first guess as to a confidence interval is

$$\hat{\theta} \pm z^{1-\alpha/2} \hat{SE}$$

which, because the standard normal distribution is symmetric and $z^{\alpha/2}$ is negative, I could write the interval as

$$\left[\hat{\theta} + z^{\alpha/2} \hat{SE}, \quad \hat{\theta} + z^{1-\alpha/2} \hat{SE}\right] \tag{14.3.1}$$

```
n <- 15
x <- rpois(n, lambda=5)
theta.hat <- mean(x)
resids <- x - theta.hat
theta.hat.star <- rep(NA, 1000)
for(i in 1:1000){
  index <- sample(1:n, n, replace=TRUE)
  x.star <- theta.hat + resids[index]
  theta.hat.star[i] <- mean(x.star)
}
boot.StdDev <- sd(theta.hat.star)
lwr <- theta.hat - qnorm(.975) * boot.StdDev
upr <- theta.hat + qnorm(.975) * boot.StdDev
CI <- c(lwr, upr)
names(CI) <- c('2.5%','97.5%')
CI

 2.5% 97.5%
3.334 6.132
```

14.3.2 Percentile intervals

The percentile interval doesn't assume normality but it does assume that the bootstrap distribution is symmetric and unbiased for the population value. This is the method we used to calculate confidences intervals in at the start of this chapter is perhaps the easiest to calculate and understand. This method only uses $\hat{\theta}^*$, and is

$$\left[\hat{\theta}^*_{\alpha/2}\ ,\ \ \hat{\theta}^*_{1-\alpha/2}\right]$$

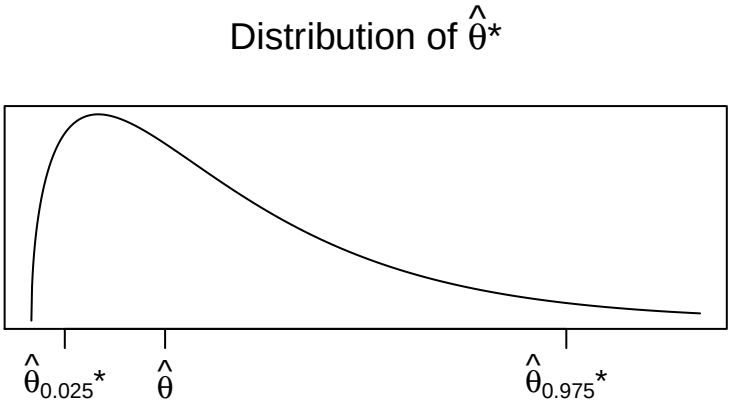
14.3.3 Basic intervals

Unlike the percentile bootstrap interval, the basic interval does not assume the bootstrap distribution is symmetric but does assume that $\hat{\theta}$ is an unbiased estimate for θ.

To address this, we will using the observed distribution of our replicates $\hat{\theta}^*$. Let $\hat{\theta}^*_{\alpha/2}$ and $\hat{\theta}^*_{1-\alpha/2}$ be the $\alpha/2$ and $1-\alpha/2$ quantiles of the replicates $\hat{\theta}^*$. Then another way to form a confidence interval would be

$$\left[\hat{\theta} - \left(\hat{\theta}^*_{1-\alpha/2} - \hat{\theta}\right),\quad \hat{\theta} - \left(\hat{\theta}^*_{\alpha/2} - \hat{\theta}\right)\right]$$

where the minus sign on the upper limit is because $\left(\hat{\theta}^*_{\alpha/2} - \hat{\theta}\right)$ is already negative. The idea behind this interval is that the sampling variability of $\hat{\theta}$ from θ is the same as the sampling variability of the replicates $\hat{\theta}^*$ from $\hat{\theta}$, and that the distribution of $\hat{\theta}$ is possibly skewed, so we can't add/subtract the same amounts. Suppose we observe the distribution of $\hat{\theta}^*$ as



Then any particular value of $\hat{\theta}^*$ could be *much* larger than $\hat{\theta}$. Therefore $\hat{\theta}$ could be *much* larger than θ. Therefore our confidence interval should be $\Big[\hat{\theta} - \text{big},\ \hat{\theta} + \text{small}\Big]$.

This formula can be simplified to

$$\Big[2\hat{\theta} - \hat{\theta}^*_{1-\alpha/2}\ ,\ \ 2\hat{\theta} - \hat{\theta}^*_{\alpha/2}\Big]$$

The following R code is an example of this calculation.

```
n <- 15
x <- rpois(n, lambda=5)
theta.hat <- mean(x)
resids <- x - theta.hat
theta.hat.star <- rep(NA, 1000)
for(i in 1:1000){
 index <- sample(1:n, n, replace=TRUE)
 x.star <- theta.hat + resids[index]
 theta.hat.star[i] <- mean(x.star)
}
lwr <- 2*theta.hat - quantile(theta.hat.star, .975)
upr <- 2*theta.hat - quantile(theta.hat.star, .025)
CI <- c(lwr, upr)
names(CI) <- c('2.5%','97.5%')
CI

 2.5% 97.5%
4.067 6.468
```

14.3.4 Studentized intervals (a.k.a bootstrap-t limits)

The studentized intervals does not assume normality or symmetry of the sampling distribution of $\hat{\theta}$, nor does it assume that $\hat{\theta}$ is an unbiased estimate of θ.

For each bootstrap sample $\mathbf{b}$, we calculate

$$t^* = \frac{\hat{\theta}^*(\mathbf{b}) - \hat{\theta}}{\hat{SE}^*(\mathbf{b})}$$

where $\hat{\theta}^*(\mathbf{b})$ is the statistic of interest calculated from the replicate data and $\hat{SE}^*(\mathbf{b})$ is the estimated standard error calculated from that replicated data set, which is estimated via bootstrap. In other words, for each bootstrap replicate, we will perform a bootstrap on the bootstrap to get the estimated standard error. Once the t^* replicates have been defined, we will use the $\alpha/2$ and $1-\alpha/2$ quantiles of those t^* in

$$\left[\hat{\theta} - t^*_{1-\alpha/2}\,\hat{SE}\,,\ \ \hat{\theta} - t^*_{\alpha/2}\,\hat{SE}\right]$$

To clarify the procedure, we will return to the Poisson example:

```
n <- 15
x <- rpois(n, lambda=5)
theta.hat <- mean(x)
resids <- x - theta.hat
theta.hat.star <- rep(NA, 1000)
t.star <- rep(NA, 1000)
for(i in 1:1000){
  index <- sample(1:n, n, replace=TRUE)
  x.star <- theta.hat + resids[index]
  theta.hat.star[i] <- mean(x.star)
  resids.star <- x.star - theta.hat.star[i]
  theta.hat.star.star <- rep(NA, 100)
  for(j in 1:100){
    index.index <- sample(1:n, n, replace=TRUE)
    x.star.star <- theta.hat.star[i] + resids.star[index.index]
    theta.hat.star.star[j] <- mean(x.star.star)
  }
  t.star[i] <- (theta.hat.star[i] - theta.hat) /
sd(theta.hat.star.star)
}
SE.hat <- sd(theta.hat.star)
lwr <- theta.hat - quantile(t.star, .975) * SE.hat
upr <- theta.hat - quantile(t.star, .025) * SE.hat
CI <- c(lwr, upr)
names(CI) <- c('2.5%','97.5%')
CI


 2.5% 97.5%
3.872 6.601
```

14.3.5 Towards bias-corrected and accelerated intervals (BCa)

The studentized intervals shows that the schemes for creating confidence intervals can get quite complicated. There is a thriving research community investigating different ways of creating intervals and which are better in what instances. The BCa interval is the most general of the bootstrap intervals and makes the fewest assumptions. Unfortunately is can sometimes fail to converge. The details of this method are too complicated to be presented here but can be found in texts such as chapter 12 in Efron and Tibshirani's book *An Introduction to the Bootstrap* (1998).

14.4 Bootstrapping other data information

In the simple ANOVA example where we created replicate data sets by taking the fitted values (i.e. the group means) and adding randomly selected residuals. The resulting dataset always had the same number of observations in each group as the original dataset. If your experiment was designed with a set number per group, then the replicate datasets should also have the same number. However, if the number of observations per group was random... then the replicate datasets should also have a random number of observations. The idea is that your replicate dataset should have the same experimental design as your actual data and whatever was randomly selected should change in the replicated datasets.

Example - Regression

Suppose that we have data for a simple regression $\{x_i, y_i\}$ and we believe the data came from the model

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

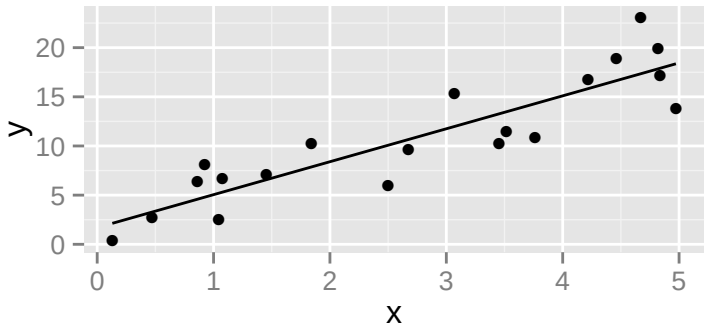
and we are interested in estimating R, Pearson's correlation coefficient, and obtaining a bootstrap confidence interval for it. Furthermore the x-values were randomly observed and there was no experimental design to force a nice even spread of x-values.

We could follow the same procedure we used for the ANOVA example in 14.2 and create bootstrap samples using the observed residuals and add them to randomly selected $\{x_i, \hat{y}_i\}$ pairs.

This first block of code is simply to generate some data

```
n  <- 20              # number of observations
beta.0 <- 2           # y-intercept
beta.1 <- 3           # slope
x <- runif(n, 0, 5)
y <- beta.0 + beta.1 * x + 2*rt(n, df=6)
model <- lm(y~x)
fitted <- fitted(model)
residuals <- resid(model)
R <- cor(x, y, method='pearson') # calculate Pearsons Correlation
R

[1] 0.8904
```

```
library(ggplot2)
ggplot(data.frame(x=x,y=y,yhat=fitted), aes(x=x)) +
  geom_point(aes(y=y)) +
  geom_line(aes(y=yhat))
```



Now that we have the correlation coefficient, we should create bootstrap samples. As usual, we will calculate the residuals and sampling from those, but now we will also resample from the x and $\hat{y}$ values as well.

```
N <- 1000
R.rep <- rep(NA, N)
for(i in 1:N){
  index <- sample(1:n, n, replace=TRUE)
  x.star <- x[index]
  y.star <- fitted[index] + residuals[index]
  R.rep[i] <- cor(x.star, y.star, method='pearson')
}
```

After resampling the data and storing the statistic of interest, we can create the percentile bootstrap confidence interval by examining the appropriate quantiles of `R.rep`.

```
quantile(R.rep, c(0.025, 0.975))

  2.5%  97.5%
0.8132 0.9503
```

We could have (and probably should have) calculated the confidence interval using the bootstrap t-distribution instead. We will skip that tedious programing exercise and examine how to get R to do those calculations.

14.5 Package 'boot' in R

It is clear that bootstrapping is a relatively tedious process when we start using more complicated methods of calculating confidence intervals. Fortunately R has a package to make these calculations easy. The package `boot` has two useful functions:

- `boot(data, statistic, R)` will do the necessary looping. It expects three arguments

  - `data` is an object that contains the data you wish to use. Usually this will be a data frame that contains all the information you need to calculate the desired statistic.
  - `statistic` is a function that has two arguments, the first is the original data, and the second is vector corresponding to the indices that are the random element of creating a replicated data set. It should return the statistic of interest and (optionally) an estimate of its standard deviation if we want to calculate the studentized confidence interval.
  - `R` is the number of bootstrap replicates to compute

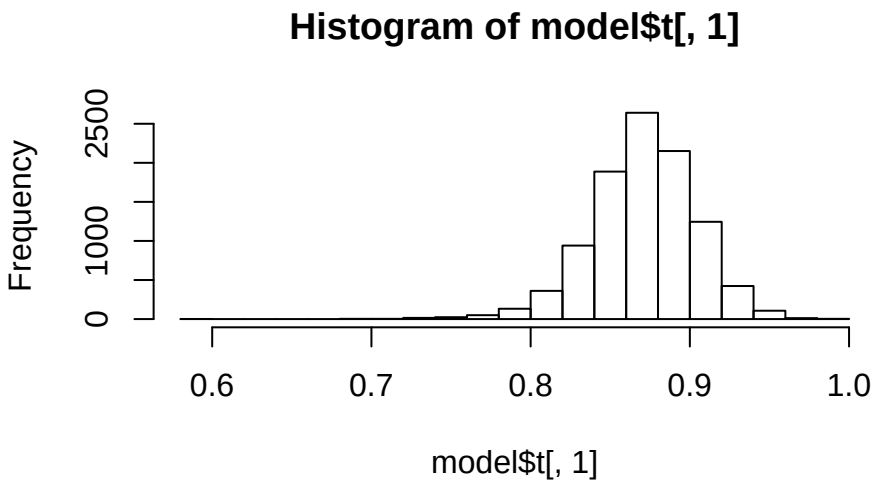
- `boot.ci(model)` calculates different forms of the confidence interval using the output `model` from the function `boot()`.

To demonstrate how these function work, we will recreate the regression analysis and calculate a bootstrap confidence interval for the correlation coefficient.

```
library(boot)        # attach the 'boot' library
n  <- 20             # number of observations in each group
beta.0 <- 2          # y-intercept
beta.1 <- 3          # slope
x <- seq(0, 5, length=n)
y <- beta.0 + beta.1 * x + 2*rt(n, df=6)
model <- lm(y~x)
residuals <- resid(model)
fitted <- fitted(model)
my.data <- data.frame(x=x, y=y, yhat=fitted, residuals=residuals)
stat <- function(data, indices){
  x.star <- data$x[indices]
  y.star <- data$yhat[indices] + data$residuals[indices]
  cor.star <- cor( x.star, y.star, method='pearson' )
  return( cor.star )
}
model <- boot(my.data, stat, R=10000)
model


ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = my.data, statistic = stat, R = 10000)


Bootstrap Statistics :
    original   bias    std. error
t1*   0.8644 0.006427     0.03272
```

We first notice that based on our bootstrap replicates, the bias of our estimator is quite small. Looking at a histogram of $\hat{\theta}^*$, we see

```
hist(model$t[,1])
```



and conclude that there might be some issue with the distribution being skewed. With this in mind, the basic interval might be the most appropriate. Next we ask R to calculate the various bootstrap confidence intervals for us.

```
boot.ci(model)

Warning:  bootstrap variances needed for studentized intervals

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = model)

Intervals :
Level      Normal              Basic
95%   ( 0.7939,  0.9221 )   ( 0.7982,  0.9273 )

Level     Percentile            BCa
95%   ( 0.8016,  0.9307 )   ( 0.7683,  0.9148 )
Calculations and Intervals on Original Scale
```

The warning message is that the studentized interval requires an estimate of the variance of $\hat{\theta}^*$. While the basic interval is likely our best interval type, for completeness we will also calculate the studentized interval. To do so we need an estimate of the variance of $\hat{\theta}^*$ we need to create an estimate of that, which we will bootstrap our bootstrap.

To do this, we will create another function `stat2()` that will calculate and return the correlation coefficient, but will also call the bootstrap using my old function `stat()`.[2]

[2]We can bootstrap multiple parameters and return them, but if we want the studentized interval, our statistic function can only return the one statistic and the estimated variance.

```
stat2 <- function(data, indices){
  x.star <- data$x[indices]
  y.star <- data$yhat[indices] + data$residuals[indices]
  cor.coef <- cor(x.star, y.star, method='pearson')
  inner.boot <- boot(data[indices,], stat, 100)
  cor.var <- var(inner.boot$t[,1])
  return(c(cor.coef, cor.var))
}
model <- boot(my.data, stat2, R=1000)
model


ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = my.data, statistic = stat2, R = 1000)


Bootstrap Statistics :
    original    bias    std. error
t1* 0.864445 0.0057246    0.031926
t2* 0.001401 0.0003027    0.001481

boot.ci(model)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = model)

Intervals :
Level      Normal              Basic              Studentized
95%   ( 0.7961,  0.9213 )   ( 0.7970,  0.9264 )   ( 0.7646,  0.9063 )

Level     Percentile            BCa
95%   ( 0.8025,  0.9319 )   ( 0.7859,  0.9191 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
```

14.6 Conclusion

As your sample size increases, the bootstrap method is asymptotically consistent, but there is no small sample guarantee of correctness. In general, bootstrap methods tend to be a little liberal (too small of confidence intervals). Finally, bootstrapping is not assumption free and the different types of intervals make different number of assumptions and all of the methods assume independence of the data. The assumptions are:

1. Normal Intervals: The sampling distribution of $\hat{\theta}$ is normal and therefore is symmetric and $\hat{\theta}$ is an unbiased estimate of θ.

2. Percentile Intervals: The distribution of $\hat{\theta}$ is symmetric and $\hat{\theta}$ is an unbiased estimate of θ.
3. Basic Intervals: That $\hat{\theta}$ is an unbiased estimate of θ.
4. Studentized Intervals: That we can estimate the bias and variance of $\hat{\theta}$, possibly by bootstrapping the bootstrap. Bootstrapping the bootstrap is computationally intensive.
5. BCa: That the bias and variance of $\hat{\theta}$ can be estimated. The "acceleration" part of this algorithm can sometimes numerically unstable.

Picking which interval type is not always straight forward. One method is to start with the method with the most assumptions decide if the assumptions are invalid and move down the list of methods until we arrive at a set of assumptions we can live with.

In practice the bootstrap is used when:

1. The sampling distribution of the statistic of interest is extremely complicated or unknown.
2. The sample size is too small for asymptotic distributional assumptions to be appropriate.
3. Power calculations must be made and pilot sample data is available.