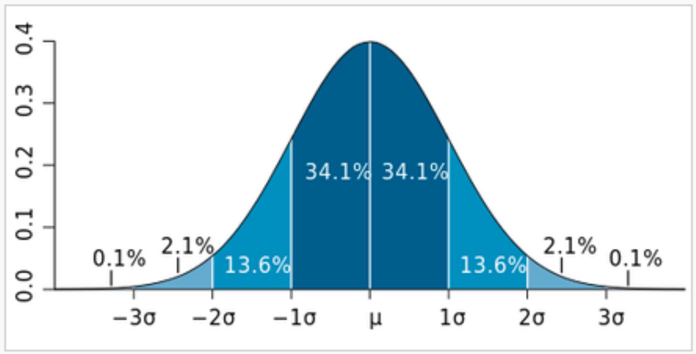
Lecture 5. Introduction to likelihood

We've spent some time introducing different kinds of deterministic model structure (linear and nonlinear functions, factors) and the different probability distributions used to model the variability around this structure. Now we need to think about how models are fit to data. Once you get beyond classic regression and Anova, the most common way to fit models to data is the method of *maximum likelihood*. This is a very general and powerful method for answering the question "what is the model that is the best fit for this data?".

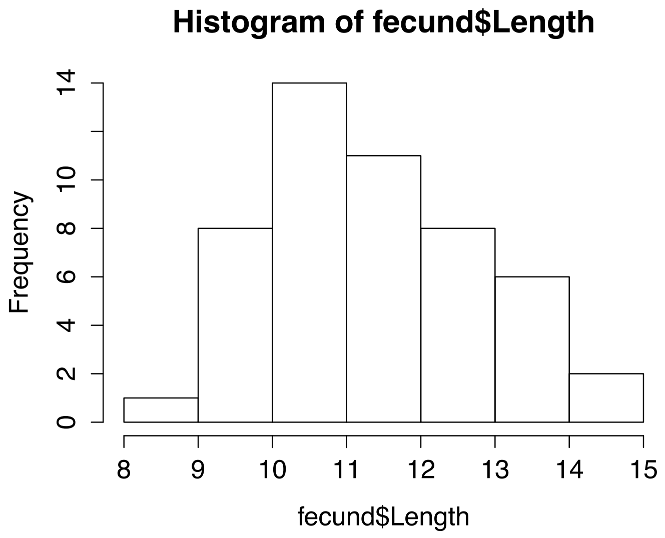
What is a likelihood?

When we talked about probability distributions, we talked about the *probability density function* (PDF; for continuous distributions) and the *probability mass function* (PMF; for discrete distributions). For example the probability density function for the normal distribution looks like this.



The probability density function (pdf) of the normal distribution, also called Gaussian or "bell curve", the most important continuous random distribution. As notated on the figure, the probabilities of intervals of values correspond to the area under the curve.

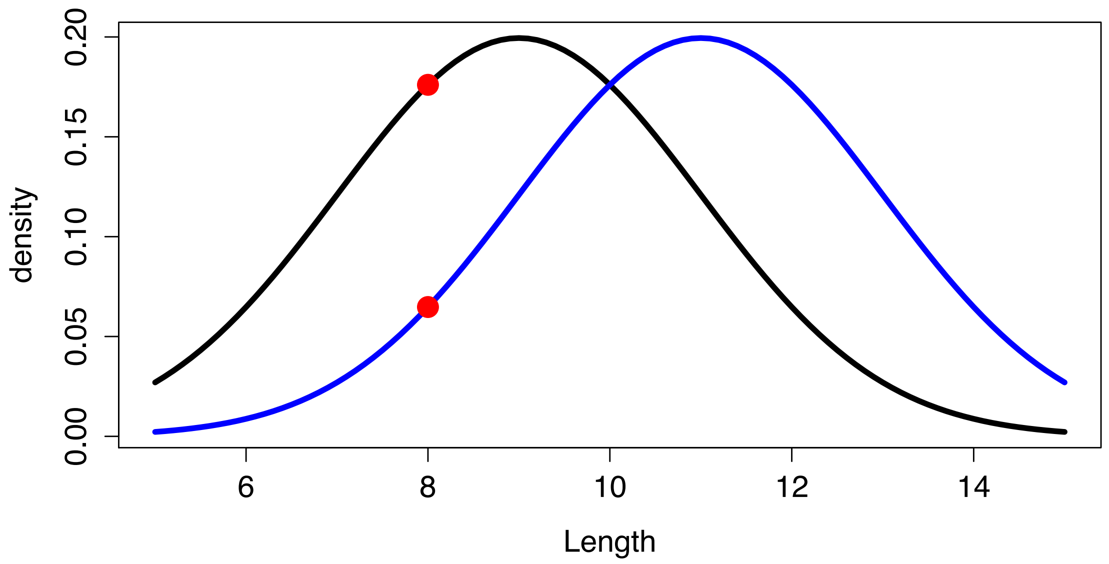
Now let's say we have some data that appear to be normally distributed, and we want to ask "what normal distribution is the closest fit to this data?". Here is a distribution of isopods lengths, which we discussed in lecture 1, sampled from one population:



One question we can ask is "what normal distribution is the best match to this data?". This is, essentially, the simplest kind of statistical model; just a distribution to model the variation, with no groups to compare and no continuous predictors. So this is where we'll start to figure how to fit models to data.

A probability distribution is defined by its parameters. The normal distribution has two parameters, the mean and the variance (or instead of the variance you could define it in terms of the standard deviation, which is just the square root of the variance). So to ask "what normal distribution is the closest fit to this data?", we need to ask "what parameters for the normal distribution give a distribution that is the closest fit to the data?". These parameters can take on a whole range of values (from –Infinity to Infinity for the mean, from 0 to Infinity for the variance), and so we can think of all possible parameter values as representing a whole family of possible models for the data, and we want to find the one model that is in some sense the best.

To answer this question, let's think about one data point at a time. Imagine we sampled a single isopod of length 8 mm. We can ask "if I go out to the coast and pick a random isopod, what is the probability of observing an isopod of length 8 mm?". The answer to this question will depend on what distribution the isopod is coming from. Here are two arbitrary example distributions, one with mean = 8 and standard deviation (SD) = 2, and one with mean = 11 and SD = 2.



The red points show the density for a length of 8 from the two distributions. So this illustrates that the answer to "what is the probability of observing an isopod of length 8 mm?" depends on the parameters of the underlying distributions. This is typically written like this:

$$P(X|\theta)$$

This mean that the probability of observing X (e.g., an isopod of length 8) depends on some parameter(s) θ (in this example, the parameters are the mean and variance of the normal distribution). The vertical line, "|", indicates a conditional probability, i.e. the probability of X is conditional on θ. Note that both X and θ can be vectors; in other words, we observe a number of isopods represented as X, and those depend on multiple parameters θ, such as the mean and variance of the normal distribution. Writing it in this way makes things cleaner to read. If the underlying probability distribution is the normal distribution, we can calculate this probability as

$$\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

Of if we use R, we can calculate it as dnorm(x, μ, σ).

If you remember back to lecture 2, I noted that a probability density function does not give you a probability *per se*, because you have to integrate under the curve to find the probability that X falls between two values. But as you can see from the plot, the density function quantifies the relative probability of seeing X when comparing different curves.

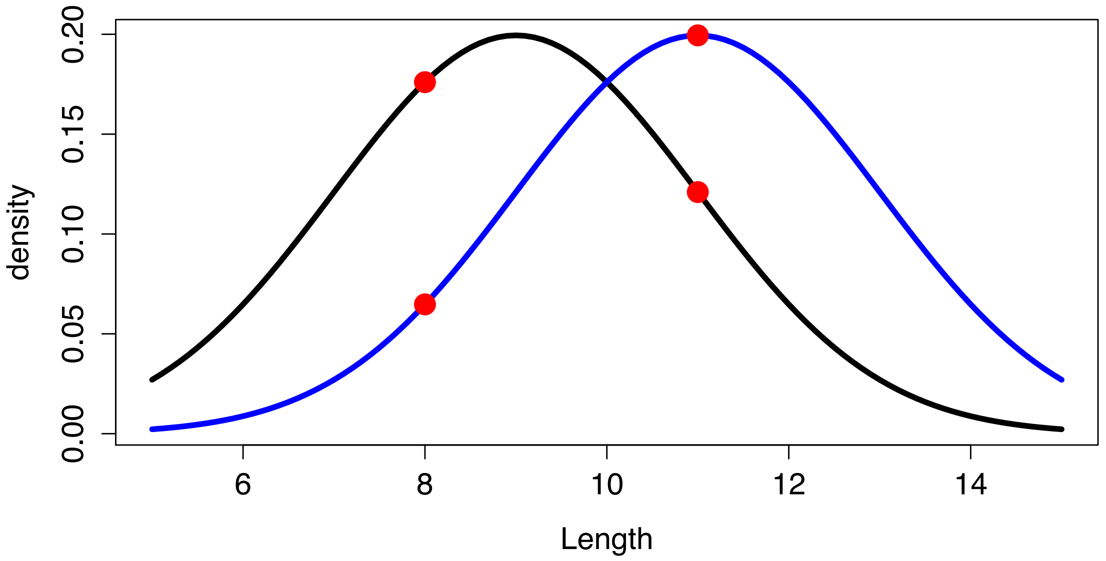
Right now we're asking the question, "what is the probability of observing an isopod of length 8 mm, assuming a particular normal distribution", but when we do statistics we are mostly interested in the reverse question, "which parameter values for the normal distribution are the best fit, given the fact that we found an isopod of length 8mm?". To answer this question, early statisticians made a simple equivalence:

$$L(\theta|X) = P(X|\theta)$$

This is read as "the likelihood of the parameters θ, given the observed data X, is equal to the probability of the data X, assuming the parameters θ are the true parameters". Or more briefly, "the likelihood of the parameters given the data is equal to the probability of the data given the parameters".

The reason for this equation is hopefully intuitive. In the plot above, the probability density for the red point (isopod of length 8) is higher under the black curve than under the blue curve. The equation $L(\theta|X) = P(X|\theta)$ is saying that the black curve is more likely to be the true distribution generating the data. More quantitatively, the equation is saying that relative likelihood of these two curves is actually equal to the relative heights of the density functions when $X = 8$.

Joint probability. If you want to estimate the parameters of a distribution, one isopod is not a lot to work with. How do we combine multiple data points to calculate a likelihood? For example, imagine we had two isopods:



Now we have an isopod of length 8, and an isopod of length 11. For each of the two PDFs, we want to calculate the probability of observing these two isopods. This is called a *joint probability*, and the rules for probability tell us that the probability of observing both X_1 and X_2, if they are independent samples, is equal to the product of their individual probabilities, or

$$P(X_1, X_2|\theta) = P(X_1\ |\theta)P(X_2\ |\theta)$$

Likewise, if we want to calculate the likelihood of the two curves above, now that we've measured two isopods, it is equal to this joint probability

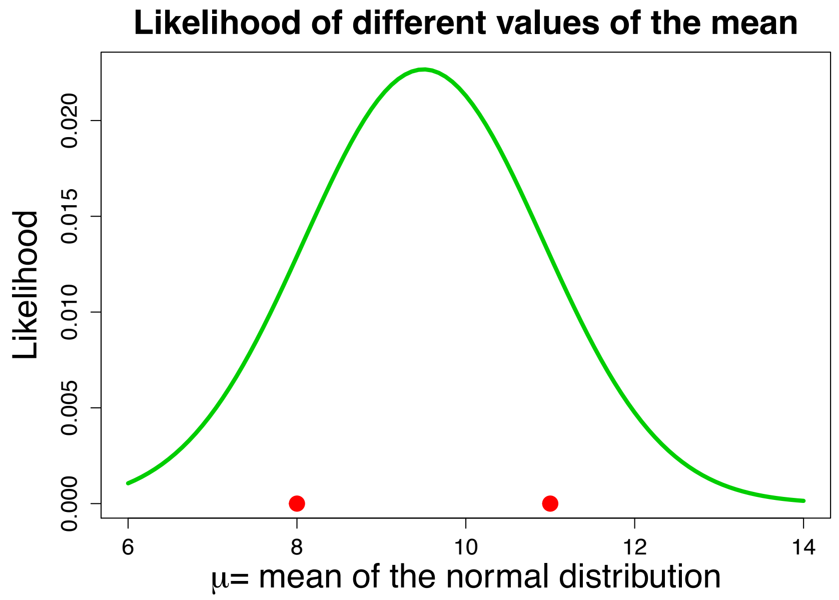
$$L(\theta|X_1, X_2) = P(X_1\ |\theta)P(X_2\ |\theta)$$

As we sample more and more isopods, we will just keep multiplying by each new probability to get the joint probability density of observing all our isopods. When we multiply many numbers together, this is abbreviated with a capital pi, Π, similar to how adding many numbers together is abbreviated with a capital sigma, Σ. So for a large number of data points we can write:

$$L(\theta|X_1, X_2, \dots) = \prod_{i=1}^{n} P(X_i\ |\theta)$$

Where *n* is the total number of samples (number of isopods measured).

Maximum likelihood. Now we've defined what a likelihood is, and how to calculate it. This lets us compare the likelihood of different parameter values, conditional on the data we observe. The final step is to ask, "what is the best choice of parameter values, given the data we observe?". Unsurprisingly, the best choice is the parameter values that maximize the likelihood. When we write $L(\theta|X)$, we can treat this as a *likelihood function*, where *X* is constant (it is the data we have already observed), and θ varies to determine the likelihood. Here is an example where I've taken the two isopods in the example above, and calculated the likelihood of different values for the mean of the normal distribution (μ):



We can see that the likelihood function peaks somewhere between 9 and 10; at the bottom I've plotted the location of the two observed isopods (at 8 and 11). Please note that although this is a bell-shaped curve like the blue and black curves shown earlier, *this is not a probability density function. This is a likelihood function that plots the likelihood of a parameter value, conditional on some observed data*.

Where is the peak of this curve, i.e. what value for the mean of the normal distribution maximizes the likelihood? In technical terms, this is called the *maximum likelihood estimate (MLE)*. For simple situations like the normal distribution, the MLE can be calculated with some simple calculus (I will not show it here). The answer is that *for the normal distribution, the maximum likelihood estimate of the mean is equal to the mean of the samples*. For our two isopod samples (8 and 11 mm), the mean is 9.5, so that is the MLE for the mean of the normal distribution from which these samples are drawn.

This seems almost too simple to be worth all this work. To reiterate, if we have a bunch of data, and we assume the data come from a normal distribution, then the best estimate for the mean of that distribution is the mean of the data. Likewise, the maximum likelihood estimate for the standard deviation of the underlying distribution is the standard deviation of the data. This example is indeed very simple, but it is merely an introduction and the principle of maximum likelihood is very valuable for more complex models.

Why is the maximum likelihood estimate the best estimate? Let's review the logic of the maximum likelihood method:

1) We have some data

2) We assume these data are generated by some model (e.g., they come from a normal distribution)

3) We want to use the data to find the best parameter estimates for the model

4) We choose the parameters that maximize the likelihood of the parameters given the data

5) This is equivalent to finding the parameters that maximize the probability of observing the data

So far I haven't mentioned why the MLE is the "best", or in what respects it is best. To answer this questions gets into some real statistical theory, but for our purposes I'll mention two properties of maximum likelihood. 1) The MLE is *consistent*: as sample size gets large, the MLE will converge on the true value of the distribution that generates the data. 2) The MLE is *efficient*: for large sample size, there is no other method that will yield an estimate closer to the true value.

You may have noticed that throughout this intro to likelihood, I often mention that we assume there is some "true" model with "true" parameters values that exists out there in the world, and which generates the data we observe. It is debatable whether this way of thinking makes sense or not, and the question gets pretty philosophical, but for now I'll note that this kind of framework is necessary for statisticians (and people who use their methods) to get anything done. In other words, you have to assume something (e.g. the data come from a normal distribution) in order to make any calculations at all. This is why it is important to check whether the assumptions are reasonable for your data, because the whole enterprise is built on them.

How to calculate a likelihood in R. Mercifully, for most kind of analyses you will not need to know how to calculate a likelihood or find a MLE, because you will use some model-fitting function that does it for you. Unmercifully, I'm going to spend some time describing how to calculate a likelihood in R, and ask you to do this a couple times for homework. The goal is to help you understand how this method actually works, practice R programming, and in addition there are definitely situations in biology where you want to fit a model that the standard functions can't fit.

The likelihood is just the probability of observing the data, assuming particular parameter values for the underlying distribution. Probability distributions in R have a number of built-in functions, and to calculate probability densities (for continuous distributions) or probability masses (for discrete distributions), we use the functions that are prefaced with 'd'. dnorm() for the normal distribution, dpois()

for the poisson distribution, etc. For a single data point *x*, the probability density for the normal distribution is just

dnorm(*x*, mean, SD)

where you specify the mean and the SD for the distribution. For example, if we're looking at isopod lengths, we can do

```
dnorm(fecund$Length[1], 9.5, 2)
```

```
## [1] 0.04659
```

Here the data frame is 'fecund', and we are finding out the likelihood of observing the first length sample, assuming that the distribution has a mean of 9.5 and a SD of 2.

When we have a whole vector of samples, the likelihood function is

$$L(\theta|X_1, X_2, \ldots) = \prod_{i=1}^{n} P(X_i\,|\theta)$$

which means we want to multiply all the probabilities together. One way to do this would be to multiply a bunch of dnorm() calls:

```
dnorm(fecund$Length[1], 9.5, 2)*dnorm(fecund$Length[2], 9.5, 2)*dnorm(f
ecund$Length[3], 9.5, 2)
```

```
## [1] 0.0009819
```

Here I've multiplied the densities of the first three lengths in the dataframe. Writing all this out is tedious, luckily there's a better way. dnorm(), like many R functions, is vectorized, which means we can feed it a vector of lengths and it will return a vector of densities:

```
dnorm(fecund$Length[1:3], 9.5, 2)
```

```
## [1] 0.04659 0.10582 0.19918
```

To get the joint density of these three values, we need to multiply them together, which can be done with prod():

```
prod(dnorm(fecund$Length[1:3], 9.5, 2))
```

```
## [1] 0.0009819
```

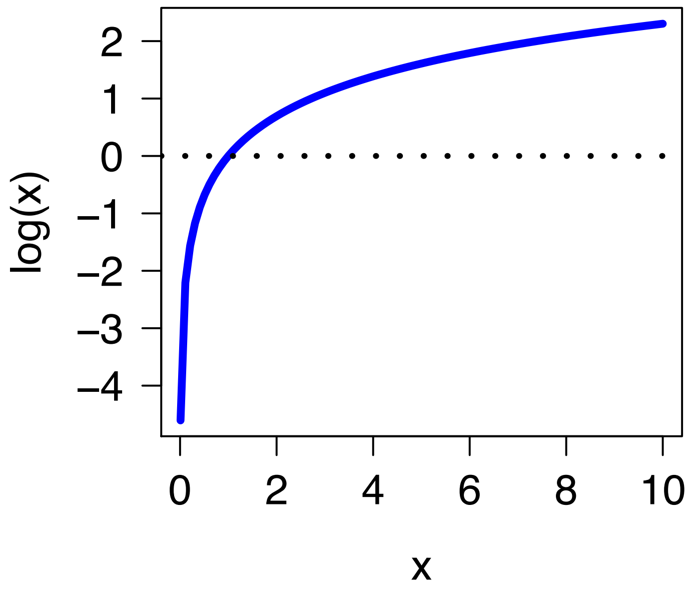
In principle this would be a fine way to calculate a likelihood with relatively little code, but there is a numerical issue with multiplying all these probability densities together. The densities are often < 1, and if we have a lot of data (could easily be thousands of samples for some analyses) then we would be multiplying a bunch of small numbers together, with a result that is *really* small.

```
prod(dnorm(fecund$Length, 9.5, 2))

## [1] 4.06914e-50
```

For all the lengths in this dataset (50 of them), the joint probability density under this model is about $4*10^{-50}$. R is smart and knows how to deal with very small (or very large) numbers, but nonetheless it is best to avoid doing calculations on computers with really small numbers, because it can sometimes lead to problems having to do with how the numbers are stored. In addition, the mathematical theory that underlies our statistical methods is a lot easier to do if we are *adding* a bunch of numbers together instead of *multiplying* them. For that reason, it is more common to work with the *log-likelihood*. Because this is so common, there is a built-in option for functions like dnorm() to return the result on a log scale:

```
dnorm(fecund$Length[1:3], 9.5, 2, log = TRUE)

## [1] -3.066 -2.246 -1.614
```

These numbers are negative because the probability densities on the original scale (`0.04659 0.10582 0.19918`) are less than one, and the log of a number less than one is negative:



Why is it useful to work with the log-likelihood? To review the properties of the logarithm one more time,

$$log\left(L(\theta|X_1, X_2, \ldots)\right) = \log\left(\prod_{i=1}^{n} P(X_i\,|\theta)\right) = \sum_{i=1}^{n} \log\,\left(P(X_i\,|\theta)\right)$$

Or in other words, the log of the product of the probabilities is equal to the sum of the log of the probabilities. So to calculate a log-likelihood, we just sum the densities on the log scale:

```
sum(dnorm(fecund$Length[1:3], 9.5, 2, log = TRUE))

## [1] -6.926
```

In addition, as you can see from the graph of log(x), as the likelihood gets very small (approaching zero), the log-likelihood will just get more and more negative, which means that the difference between 10^{-50} and 10^{-49} is the same as the difference between 10 and 100, on a log scale. This makes it easier to work with very small numbers and combine them with more intermediate size numbers.

Likelihood curves. Now that we know how to calculate a likelihood (or a log-likelihood), one thing we can do is plot a *likelihood curve*, which shows how the likelihood varies as a function of a parameter. I already did this above with the green curve. How do we code this up? This will be slightly tricky, but it is a good time to practice writing your own function.

Let's define a function that calculates the log-likelihood for the mean of a normal distribution, with the likelihood calculated based on the isopod length data I've been using:

```
log.lik = function(mean.parm) {
  sum(dnorm(fecund$Length, mean.parm, 2, log = TRUE))
}
```

Let's unpack how this function definition works. On the left is the name of the function. The part inside of function(), mean.parm, is the variable that we give to the function to do a calculation on. In this case it will be a number representing the mean of the normal distribution for which we want to calculate a likelihood. Inside of the curly brackets are the commands that the function performs. In this case, it is the same calculation of likelihood that I described above. This will calculate the probability density for each entry of the vector fecund$Length, assuming that those values come from a normal distribution with mean = mean.parm, and with a standard deviation of 2. It will calculate these probabilities on a log scale, and then sum them up with sum(), and this is the log-likelihood for the parameter mean.parm.

A couple things to note here. The function is defined to use fecund$Length; so this is a function that will only work for that data, because I've stated that's what dnorm uses. In addition, I've assumed the standard deviation is 2. In reality, we don't know what it is, and the full maximum likelihood method will estimate the mean and the standard deviation at the same time. But for not let's just imagine that the mean of the distribution is the only thing we're trying to estimate.

Now that we have a function that calculates the log-likelihood we can try it on different parameter values for the mean. For example,

```
log.lik(9)

## [1] -126.7
```

This says the log-likelihood for mean=9 is -126.7. What if we want to calculate the log-likelihood for more than one parameter value? We can try putting a vector of parameter values into the function:

```
log.lik(c(9,11))

## [1] -112.3
```

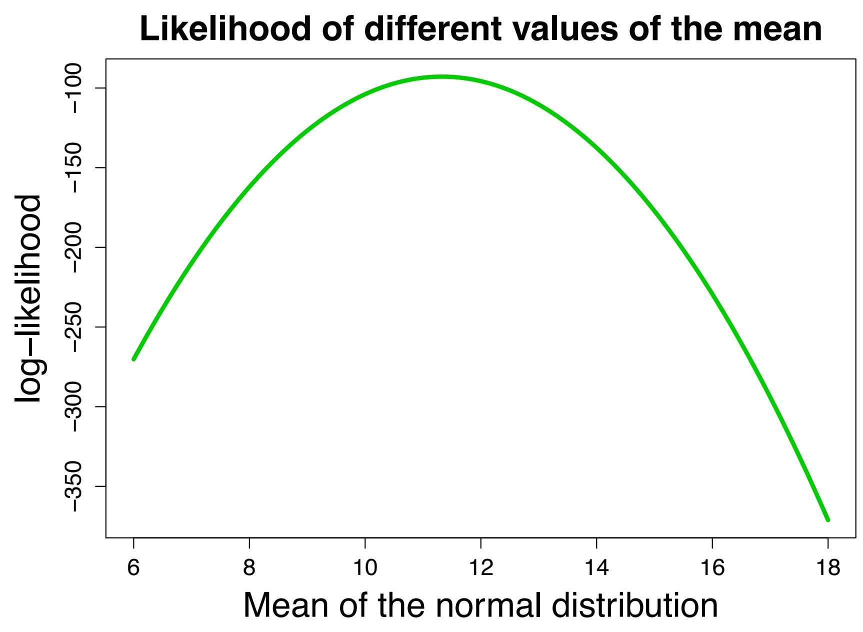
Hmm, it doesn't look like this worked. I gave it two parameter values (9 and 11), but it only returned one number. The problem is that this function is not *vectorized*, i.e. it isn't set up to take a whole vector and return a corresponding vector. But that's OK, because we can use sapply() instead:

```
sapply(c(9,11), log.lik)

## [1] -126.71  -93.53
```

This sapply() command is saying, "take the vector c(9, 11), and for each element of that vector, apply the function log.lik". And we get our two log-likelihoods as the output.

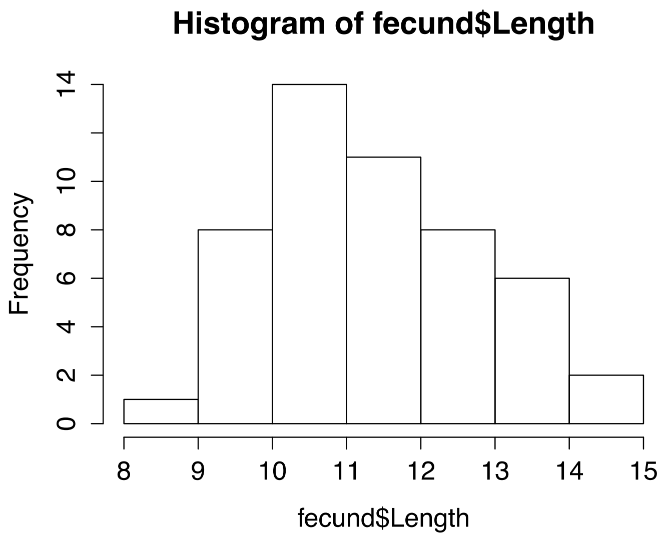
Now we're ready to plot a likelihood curve. We can use our log.lik function, in combination with sapply, as an input for the curve() function.

```
curve(sapply(x, log.lik), from = 6, to = 18, lwd = 4, col = 'green3', y
lab = 'Likelihood', xlab = 'Mean of the normal distribution', main = 'L
ikelihood of different values of the mean', cex.lab = 1.5, cex.main =
1.5)
```



How does this work? I fed curve() the function `sapply(x, log.lik)`. As we know, curve takes 'x' as the dependent variable of a function to plot. What curve does behind the scenes is it generates a vector of values for x, and then it calculates the corresponding y-values using the function you supply, and then it plots them as a continuous curve. So curve wants a vectorized function that can take a vector of x's and return a vector of y's. The sapply() command does just that.

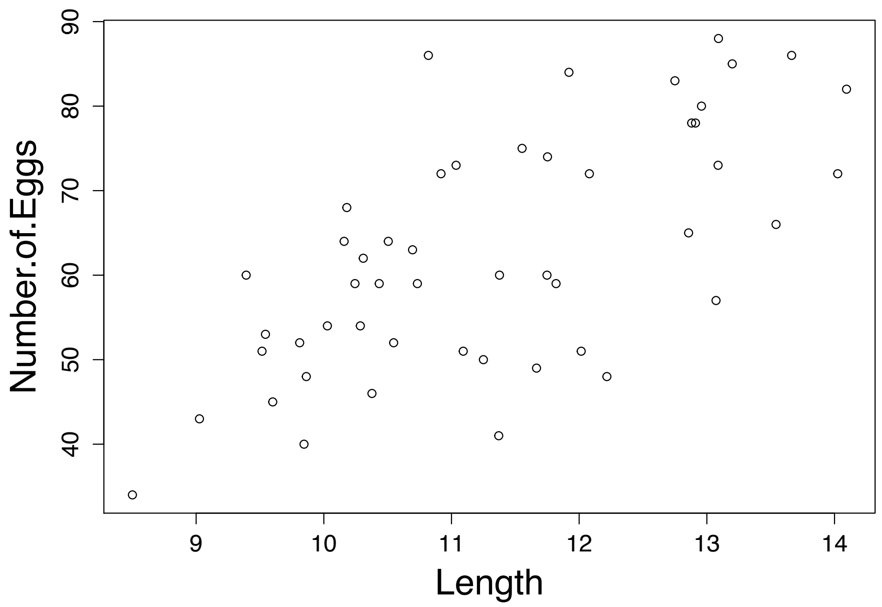
If we look at the plot, we can see that the peak of the likelihood curve is around 11. That makes sense if you go back to the histogram of the data that I showed at the beginning of the lecture:



A mean around 11 seems reasonable. How do we think about the shape of the curve? We will get to that shortly.

Maximum likelihood for a linear model

One reason maximum likelihood is such a powerful approach is because you can apply the method to *any* statistical model for which you can calculate a likelihood (as we will see later, for very complex models it can actually be challenging to calculate the likelihood). Let's revisit linear models from the maximum likelihood perspective. To stick with the isopod data, let's look at fecundity as a function of length for one population:



We went to estimate the relationship between fecundity (number of eggs) and length. Based on a preliminary look at the data, it seems reasonable to use a linear relationship with a normal error distribution. The model can be written as

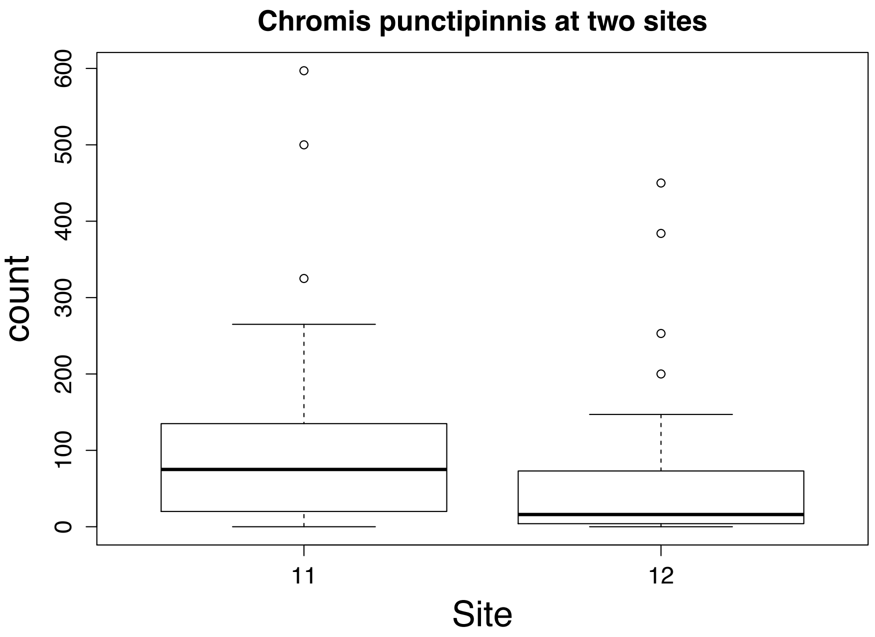
$$\mu_i = a + bL_i$$
$$Y_i \sim \text{Normal}(\mu_i, \sigma)$$

So this is just the model for a standard linear regression. The parameters of a linear model are classically estimated using the *least squares* method. That means that the "best" parameter estimates are those that minimize the (squared) distance between y-value predicted by the line and the observed y-values.

If we wanted to estimate the parameters of the regression line using maximum likelihood instead, how would the results differ? In fact, the results would be the same, because for linear models with normally distributed error the maximum likelihood estimate (MLE) for the parameters is equivalent to the least squares estimate. Another way of saying this is that *least squares is a special case of maximum likelihood*. It is a special case that applies for linear models with normally distributed error.

A maximum likelihood example with the Poisson distribution

With this example we'll get into an area where maximum likelihood offers something new. Let's say we have some survey data on the abundance of the fish *Chromis punctipinnis* at two sites over a number of years. (This is from the Channel Islands kelp forest surveys I used earlier).



Now let's say we want to test if there is a significant difference in mean abundance between these sites. This is count data, so we'll assume a Poisson distribution is appropriate (actually there is an issue because the data are *overdispersed*, which we'll discuss at a later date). This is a pretty simple model:

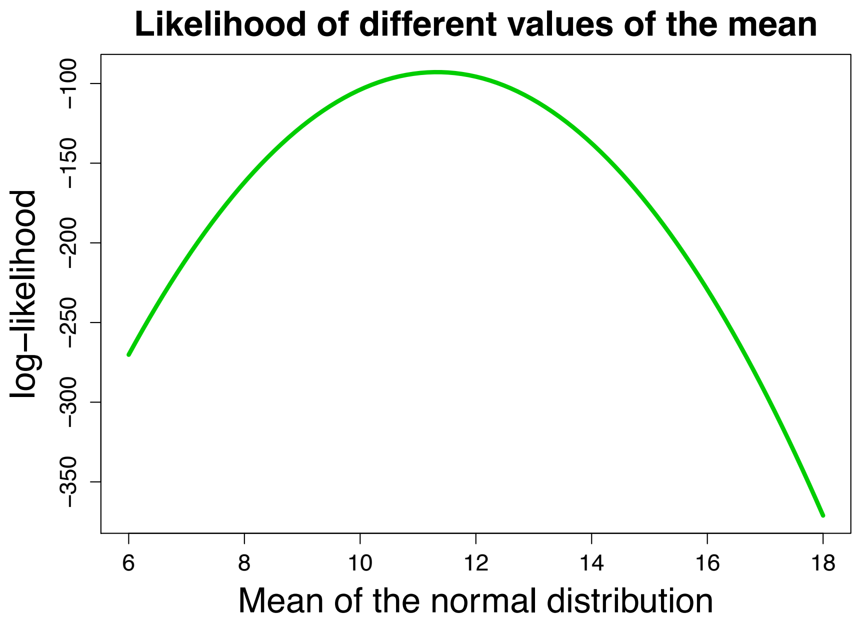
$$\mu_i = \exp(intercept + sitediff * X_i)$$
$$Y_i \sim \text{Poisson}(\mu_i)$$

Where Y_i is the observed count data, μ_i is the expected value for each count, *intercept* and *sitediff* are the two parameters of the model, and *X* is an indicator variable for the two sites. Here I've coded up the two sites in the same way that I described for factors in lecture 1. The indicator variable *X* will be 0 if the data are from Site 11, while *X* will be 1 if the data are from Site 12. That means the model will estimate the mean for Site 11 as exp(*intercept*), and the mean for Site 12 as exp(*intercept* + *sitediff*), which is equivalent to exp(*intercept*)*exp(*sitediff*) [please review the rules for exponents if you don't understand that last part]. In other words, the model assumes that the sites differ by a factor of exp(*sitediff*), and so we want to estimate the parameter *sitediff* and see whether it's different from zero. The deterministic model has the form of a linear model inside of an exponential function, which is the default for working with Poisson data (this is a generalized linear model, which we will discuss more later).

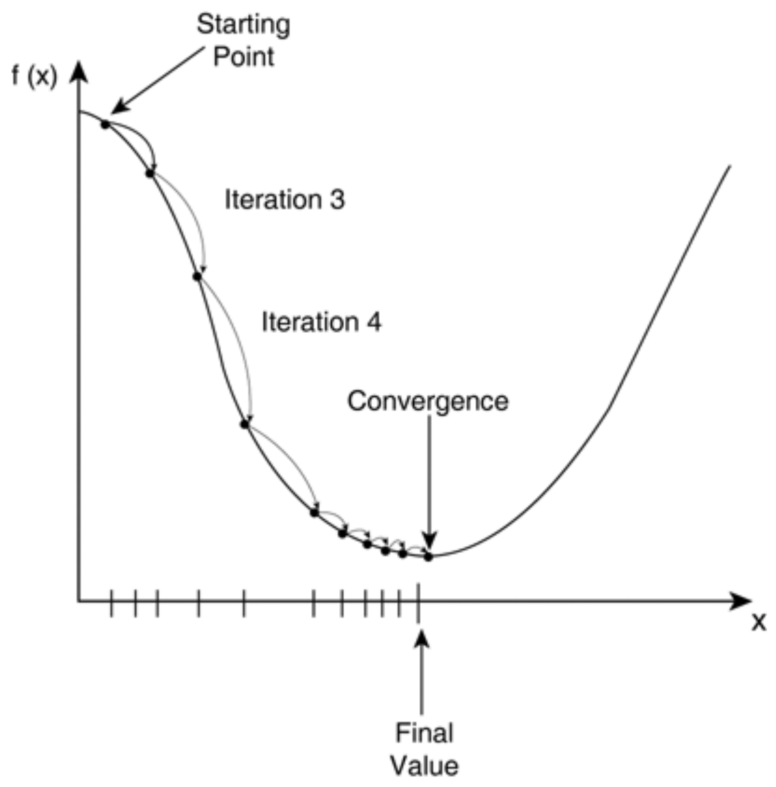
So we have two parameters, and we want to get their maximum likelihood estimates. The least squares method doesn't apply here, because that is for normally distributed data. What's the alternative? It is easy enough to calculate the likelihood for different parameter values, but we want to know which parameter values maximize the likelihood. Maybe some clever mathematician has derived a formula that tells us what the maximum likelihood estimates are for this kind of situation. Actually, it turns out that there is no such formula. I.e., for this kind of model (a generalized linear model with Poisson distribution) it is not possible to write down a formula that tells you what the MLE are.

Optimization algorithms.

Most applications of maximum likelihood fall into this category, where there is no formula to tell us what the maximum likelihood parameter estimates are. Instead the MLEs are found using some kind of optimization algorithm. Let's go back to the likelihood curve I plotted earlier, for the mean of the normal distribution of isopod lengths:



An optimization algorithm is designed to find the peak of this curve. There are many different kinds of optimization algorithms, and we won't get into them here. Here's an example of a very simple kind of algorithm:



For some reason I've never really understood, these algorithms are designed to find the *lowest* value of some function. For maximum likelihood we want the highest value, but we can just multiple the log-likelihood by -1 to get the same curve upside down, and then what we want to find is the minimum of that curve. For the algorithm plotted above, you start as some starting point. This is an initial guess for what the minimum of the curve might be. Then the algorithm says "What is the slope of this curve? Is it negative? If so, let's take a step down the curve? Is it positive? If so, let's take a step up the curve? And when we take a step, take a big step if the slope is steep, and take a small step if the slope is shallow." Eventually the algorithm gets near the bottom of the curve, where the slope is very shallow, and it takes smaller and smaller steps. At some point it says "The next step I'm supposed to take is so small, I must be very close to the bottom." And then the algorithm stops, and where it stops is the maximum likelihood estimate. This kind of algorithm is called *gradient descent*, because it moves down the curve, and the size of the steps is proportional to the slope of the curve (in multivariate calculus the slope of a surface is called the gradient).

There are many such algorithms for situations of different complexity, but they all have a similar structure. It has to start somewhere on the likelihood curve, and then it has a procedure for finding the bottom, and some criterion for when it should stop searching.

Likelihood surfaces. Let's go back to our example with the fish *Chromis* at two sites. I said there wasn't a formula to find the MLE for the two parameters of the model, *intercept* and *sitediff*. So we need to use some optimization algorithm instead to find the MLEs. The easiest way to do this is with the glm() function, which uses an algorithm designed for generalized linear models. But at the moment let's think about just what that algorithm needs to do.

The examples I've shown so far focus on one parameter at a time (e.g., the mean of the normal distribution). However, in most statistical models there is more than one parameter, and when we fit a model to some data we want to simultaneously find the values of all the parameters that maximize the likelihood. If we go back to the basic definition of likelihood,

$$L(\theta|X) = P(X|\theta)$$

both θ and X are typically vectors. So we're looking for a vector of parameters that jointly maximize the likelihood function. To better understand this we can visualize an example with two parameters, such as the fish example, where we need to estimate *intercept* and *sitediff*. What we want to visualize is the probability of the data, as a function of different values for these two parameters. So we need to define a likelihood function that calculates this probability, using the model I described above:

$$\mu_i = \exp(intercept + sitediff * X_i)$$
$$Y_i \sim \text{Poisson}(\mu_i)$$

Here is some code to do it:

```
log.lik = function(intercept, site.diff) {

  sum(dpois(fish.sub$count, lambda = c(exp(intercept),
exp(intercept+site.diff))[fish.sub$Site], log = TRUE))

}
```

Let's unpack this a little. I'm using dpois() because we are assuming Poisson-distributed count data, and I'm using sum() and log=TRUE in order to get the log-likelihood. The Poisson distribution only has one parameter, lambda, which is the mean of the distribution. This mean will depend on which site the fish are counted at, or at least that is what our model is testing. So I've made a vector of two different lambdas, `c(exp(intercept), exp(intercept+site.diff))`. The first element of the vector is the predicted mean abundance at Site 11, exp(intercept), and the second element of the vector is the predicted mean abundance at Site 12, exp(intercept+site.diff). Finally, I use the Site factor, `fish.sub$Site`, to index these two values. If the count was taken from the first site (Site 11), then dpois() will use lambda = exp(intercept), and if the count was taken from the second site (Site 12), then dpois() will use lambda = exp(intercept+site.diff).

Why does this indexing with the factor Site work? As we discussed in the previous lecture, factors are stored in R as integer vectors with associated levels. So if we look at the factor it looks like this

```
fish.sub$Site

##  [1] 11 11 12 11 12 11 12 11 11 12 12 11 12 11 11 11 11 11 11 12 12 12 12
## [24] 11 12 12 12 11 12 11 12 11 12 12 12 12 12 12 11 12 11 12 12 12 11 12
## [47] 11 11 11 11 12 12 12 12 11 11 11 12 11 12 12 12 11 12 11 12 11 12 11
## [70] 11 11 11 12 11 11 12 11 12 11 12 12 11 11 11 11 11 12 11 12 11 12 12
## [93] 11 11 12
## Levels: 11 12
```

It has two levels, '11' and '12'. We can look at the integer version also

```
unclass(fish.sub$Site)

##  [1] 1 1 2 1 2 1 2 1 1 2 2 1 2 1 1 1 1 1 1 2 2 2 2 1 2 2 2 1 2 1 2 1 2 2 2
## [36] 2 2 2 1 2 1 2 2 2 1 2 1 1 1 1 2 2 2 2 1 1 1 2 1 2 2 2 1 2 1 2 1 2 1 1
## [71] 1 1 2 1 1 2 1 2 1 2 2 1 1 1 1 1 2 1 2 1 2 2 1 1 2
## attr(,"levels")
## [1] "11" "12"
```

So site '11' is stored as a 1, and site '12' is stored as a 2. We could have already inferred this, based on the order of the levels of the factor. So if we make an imaginary vector c('a', 'b'), and index it with the Site factor we get this

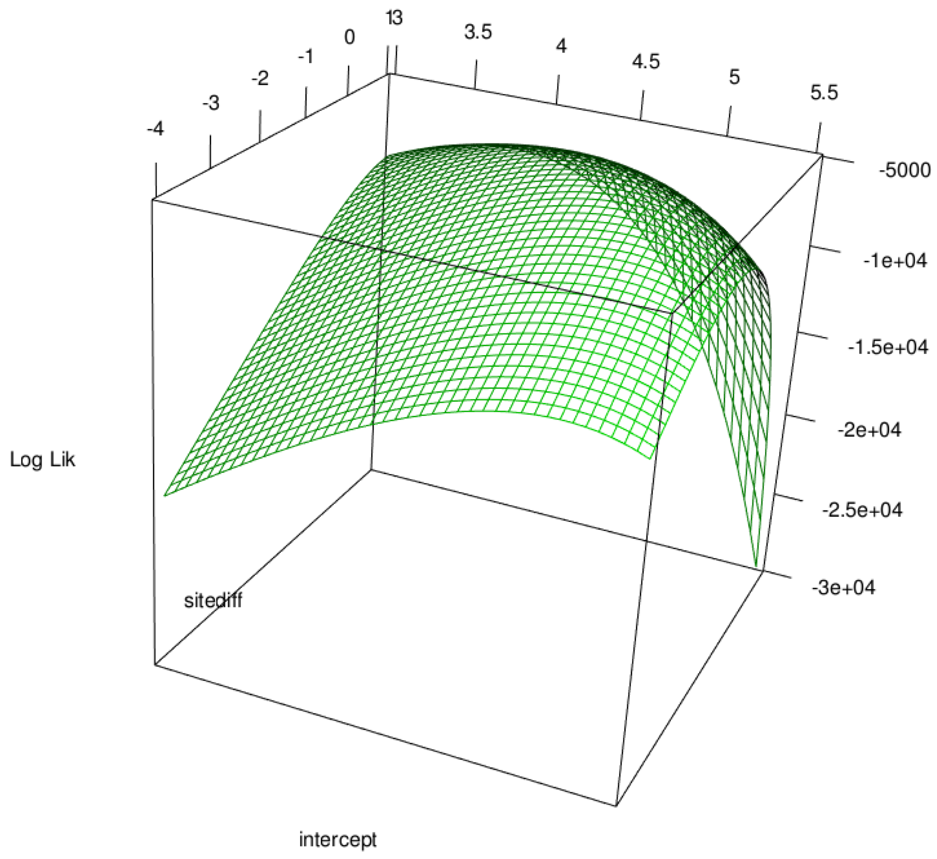
```
c("mean1", "mean2")[fish.sub$Site]

##  [1] "mean1" "mean1" "mean2" "mean1" "mean2" "mean1" "mean2" "mean1"
##  [9] "mean1" "mean2" "mean2" "mean1" "mean2" "mean1" "mean1" "mean1"
## [17] "mean1" "mean1" "mean1" "mean2" "mean2" "mean2" "mean2" "mean1"
…
```

It pulls out mean1 when Site is the first level of the factor, and it pulls out mean2 when Site is the second level of the factor.

OK after that digression on indexing with factors, let's go back to our likelihood function. I've defined a likelihood function that calculated the likelihood using the Poisson distribution, as a function of two parameters. If we have a function where there are 2 or more independent variables, that creates a surface, so let's visualize what that surface looks like:

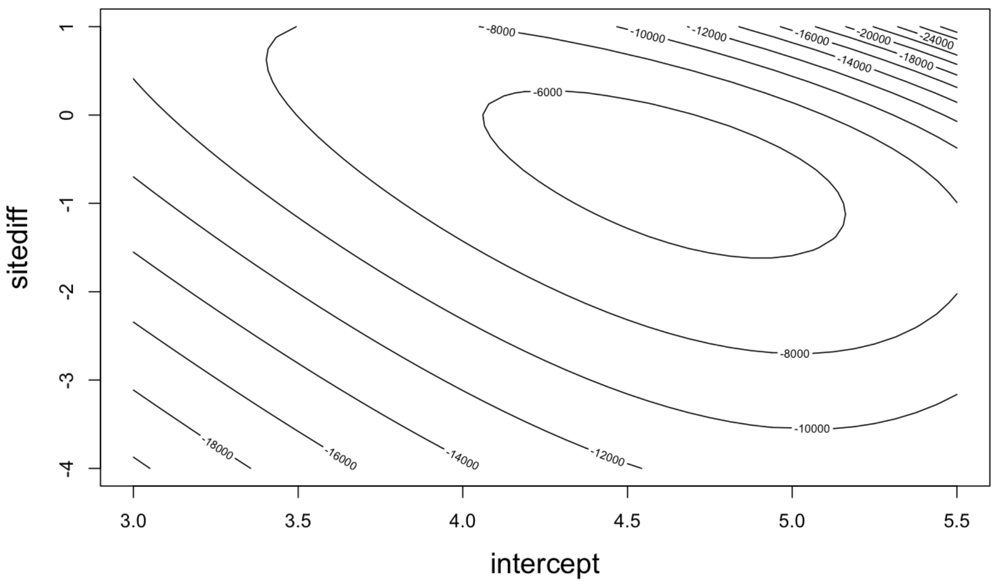
```
library(emdbook)

curve3d(log.lik(x,y), from = c(3,-4), to = c(5.5,1), sys3d = 'rgl', col
 = 'green', front = "lines", back = "lines", xlab = 'intercept', ylab =
 'sitediff', cex.lab = 1.5, zlab = "Log Lik")
```

Here I've used a nice function curve3d() that Ben Bolker created in his emdbook package. This function is nice because it works like curve(), but in 3 dimensions. So instead of just specifying the x-variable with 'x', you also specify the y-variable with 'y', and then the response variable is plotted on the z axis. This function actually calls a variety of other functions, depending on the options you set. Above I've set sys3d = 'rgl', so it will use the rgl package to make an interactive 3d plot. The rgl package is nice because you can plot something in 3d and then spin the plot around to look at it. Here is a screencap of what is looks like:



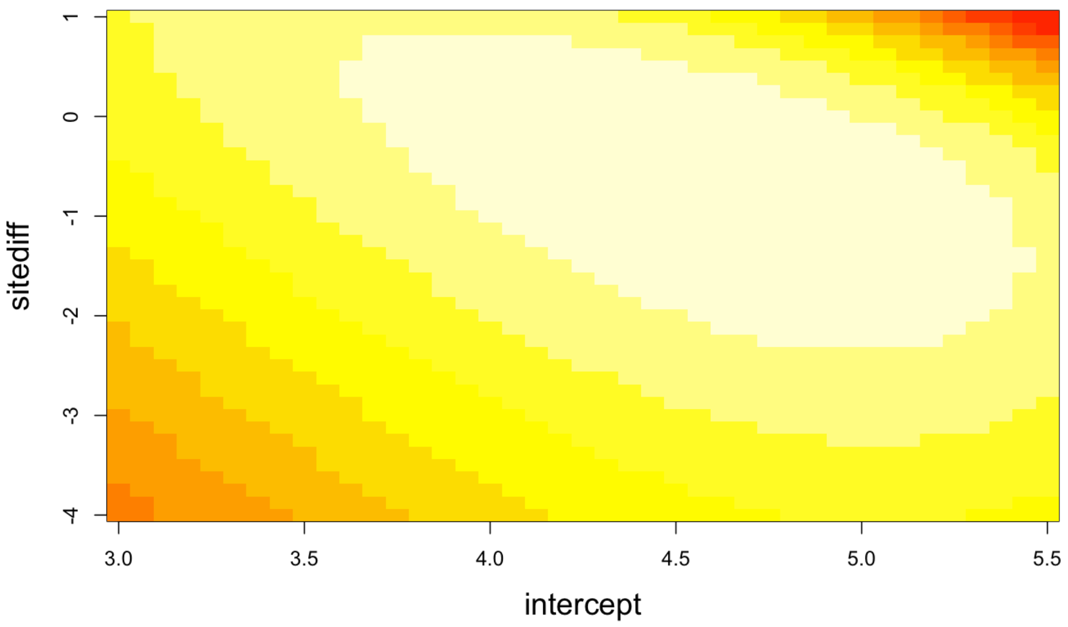
Since I can't put an interactive plot in this file, let's look at the same surface with a contour plot, by setting sys3d = 'contour'.

```r
curve3d(log.lik(x,y), from = c(3,-4), to = c(5.5,1), sys3d = 'contour',
 col = 'black',xlab = 'intercept', ylab = 'sitediff', cex.lab = 1.5, zl
ab = "Log Lik")
```



It looks like the highest values of the log-likelihood occur around intercept = 4.5 and sitediff = -0.5. As we move away from those values, the log-likelihood declines. So this surface is like a smooth mountain, and the peak of the mountain is the values of the parameters that maximize the likelihood. Thos are the values we are searching for, and a function like glm() will use an optimization algorithm that climbs to the top of this mountain. You could also look at this with a heatmap, if that is more intuitive:

```r
curve3d(log.lik(x,y), from = c(3,-4), to = c(5.5,1), sys3d = 'image',
xlab = 'intercept', ylab = 'sitediff', cex.lab = 1.5, zlab = "Log Lik")
```



The whiter areas have higher values by default. Both of the above images could be created with the contour() and image() functions, respectively, but when plotting a smooth surface the curve3d() function is a little more convenient.

Likelihood ratio test

We now know that to find the parameters of a model that provide the best fit to the data, we find the parameter values that maximize the likelihood. One we get the MLEs, how do we use this information to test hypotheses we're interested in? The most important test is the *likelihood ratio test*.

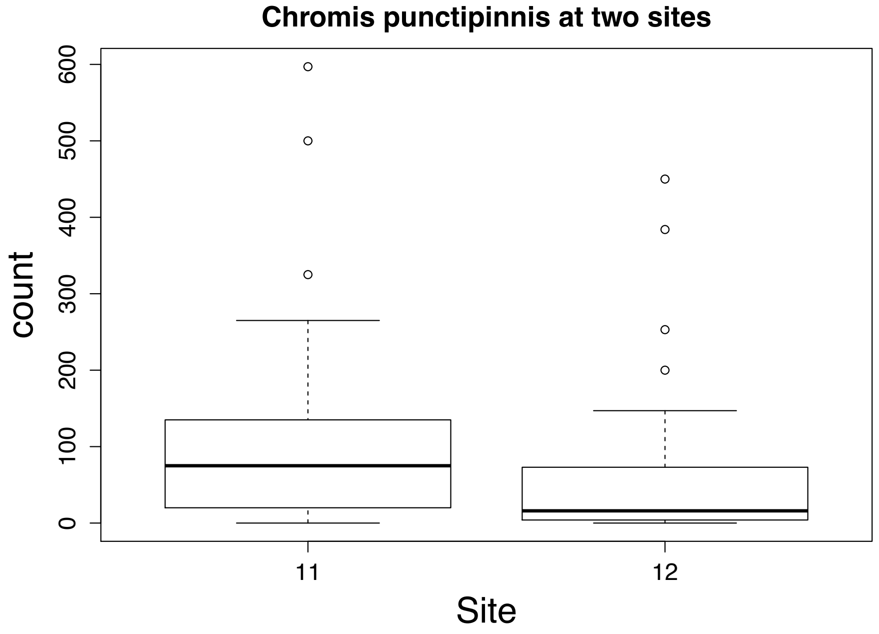
The premise of the likelihood ratio test is fairly simple. We fit a model, and we want to compare it to some null model where one of the terms is removed. This is the classis null hypothesis testing framework: the model with our term of interest is the "alternative hypothesis", and the model with that term removed is the "null hypothesis". It's often written like this:

$$H_0: \theta = \theta_0$$

$$H_1: \theta = \theta_1$$

Where H_0 is the null hypothesis, and H_1 is the alternative hypothesis. Usually our null hypothesis is that some term is absent, which is equivalent to setting one or more parameters to zero, so we could write the null hypothesis as H_0: $\theta = 0$. And the alternative hypothesis is that θ equals its maximum likelihood estimate, or H_1: $\theta = \theta_{\text{MLE}}$.

Previously I used the example of count data for a fish at two sites:



We can use this model to test whether abundance differs between these sites:

$$\mu_i = \exp(intercept + sitediff * X_i)$$
$$Y_i \sim \text{Poisson}(\mu_i)$$

The parameter we're interested in is *sitediff*, because this quantifies the difference in abundance between the sites. So we will compare this model to a model where *sitediff* is removed (you could also say that *sitediff* is set to 0). When *sitediff* is removed, the model is saying that both sites have the same mean abundance, quantified with the one parameter *intercept*.

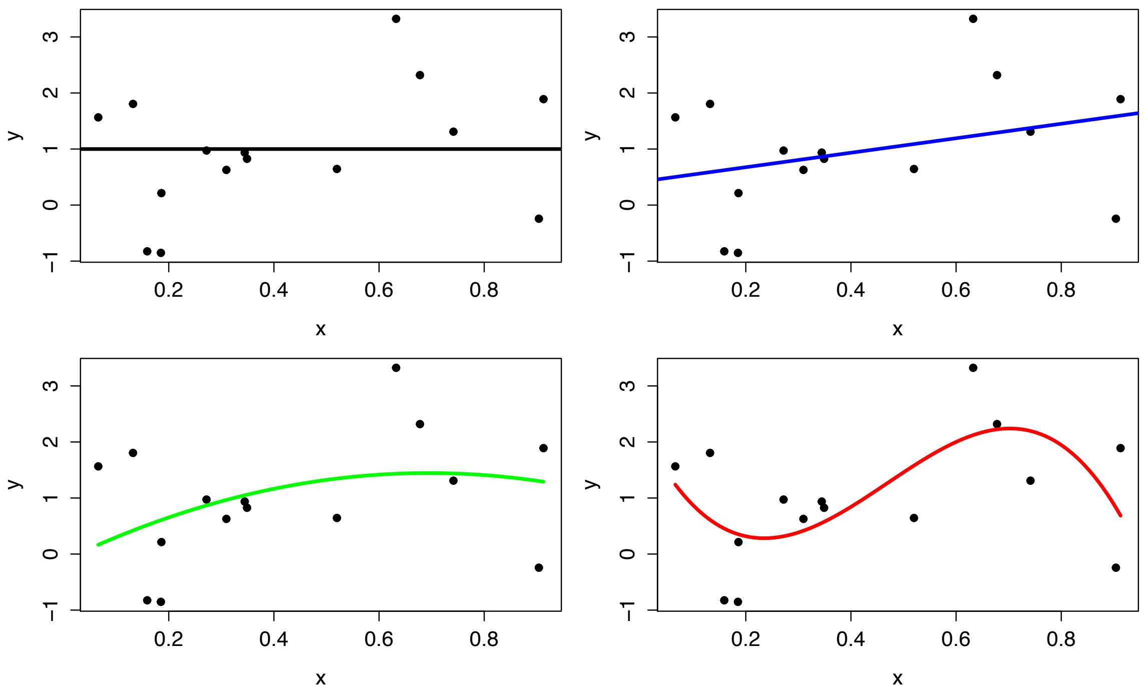
In the likelihood framework, we can compare the two hypotheses using the likelihood of two models that represent those hypotheses. In practice, we fit our original model, and then we fit a second model that removes the term of interest, which is equivalent to fitting a model where $\theta = 0$. These models are often referred to as the "full model" and the "restricted model", because when we set $\theta = 0$ we are restricting a parameter from the full model. We calculate the likelihood of each model, which is equal to the probability of the data under the model's parameter values:

$$L_1 = L(\theta = \theta_{\text{MLE}}|X) = \text{P}(X|\theta = \theta_{\text{MLE}})$$
$$L_0 = L(\theta = 0|X) = \text{P}(X|\theta = 0)$$

OK now we have two likelihoods, so how do we use them to figure out if θ is different from zero? We can quantify the difference in likelihood between the two models using their ratio: $\frac{L_1}{L_0}$, and in practice we will use the log of the ratio, $\log\left(\frac{L_1}{L_0}\right) = \log(L_1) - \log(L_0)$. The likelihood ratio is always going to be greater than one (which means the log of the ratio will always be greater than zero), i.e. *the full model will always have a greater likelihood than the restricted model*. Why is that? The restricted model is a special case of the full model, where a parameter is no longer free to vary (because $\theta = 0$). Because the full model has an extra free parameter, it will always fit the data a little better, because it has more flexibility in fitting the data. Even if the true value for θ is 0, letting θ be a freely estimated parameter will result in a higher likelihood. Let's take a quick look at why that is.

Free parameters and fitting the noise

This is a digression related to the point that the full model always has a greater likelihood than the restricted model. This actually relates to some pretty fundamental issues about how model-fitting and hypothesis testing work. We'll talk about this in more detail when we get to AIC.



I generated the above plot by selecting 15 x-values randomly from a uniform distribution, and then selecting 15 y-values randomly from a normal distribution with mean 1. So there is no real relationship here, and any hint of a relationship is just due to randomness. The black line shows the mean of the y-values (1). This is the true model: these are just 15 values with a mean of 1. The blue, green, and red lines are respectively a linear, quadratic, and cubic polynomial fit to the same data. So now we have four models:

y = a
y = a + b*x
y = a + b*x + c*x^2
y = a + b*x + c*x^2 + d*x^3

The first model is the true one (with a = 1), and each successive model adds a new parameter that is zero in the underlying true model. However, the log-likelihoods for these models are respectively -22.7, -21.9, -21.6, and -19. So each successive model has a higher likelihood, even though each successive model is more false. Why is that? In essense, the extra terms of the model allow the model to fit any subtle patterns in the noise. By fitting those subtle random patterns, the likelihood increases because observing this exact pattern in the data is more probable under that model. However, those subtle patterns aren't real and if we sampled 15 more points there would be different random patterns.

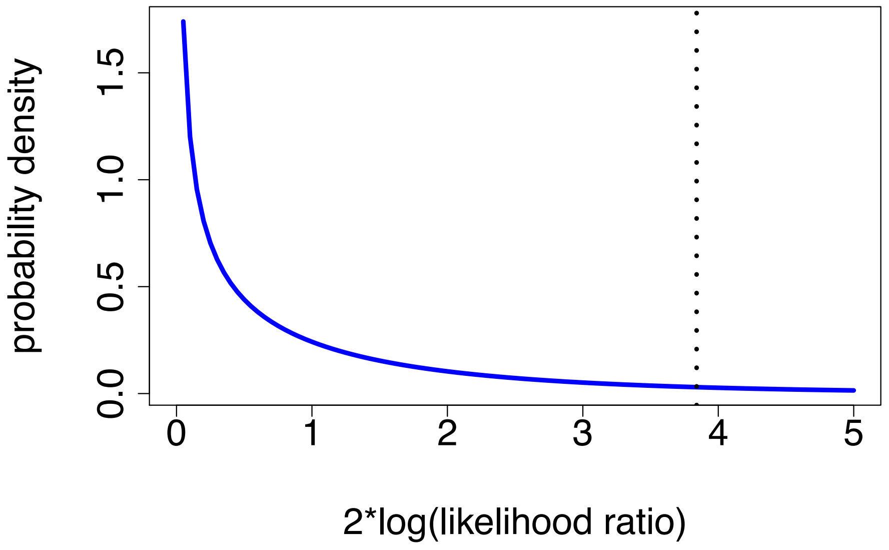
The fact that a model will find any subtle patterns in the data and fit a relationship to them is called *overfitting*, and you can think of the likelihood ratio test as saying "OK, the full model has more parameters than the restricted model. But given that there will be some overfitting, does the full model explain even more variation thatn we expect?".

The likelihood ratio test

So the question we want to ask is, *how large does the likelihood ratio have to be to indicate that the full model is a better fit than the restricted model?* The answer to this is given by a theorem that shows that the log likelihood ratio, multiplied by 2, has an approximate chi-squared distribution if the restricted model is true, i.e. if the true value of θ is 0 in our example.

$$LR = 2 * \log\left(\frac{L_1}{L_0}\right) \sim \chi^2_{df}$$

I will abbreviate 2 times the log likelihood ratio as *LR*. The chi-squared distribution is define by the degrees of freedom parameter *df*, and for the likelihood ratio the degrees of freedom is equal to the difference in the number of parameters between the full model and the restricted model. In our example where there is one parameter θ that differs between the models, *df* = 1. Finally, we use the chi-squared distribution to get a p-value, which tests whether the two models are significantly different, i.e. whether the full model is significantly different from the null hypothesis. This is the likelihood ratio test.



The plot above shows the chi-squared distribution with *df* = 1. This means that to test a single parameter, *LR* needs to be greater than ~3.84 in order for $p < 0.05$, because the area under the curve to the right of the dotted line equals 0.05.

This whole procedure takes a bit of space to explain, but in practice it is pretty straightforward.

1. Fit your model (the full model), and fit a model with one term removed (the restricted model).

2. Calculate the likelihood ratio for the two models (and calculate 2 * the log of this ratio).

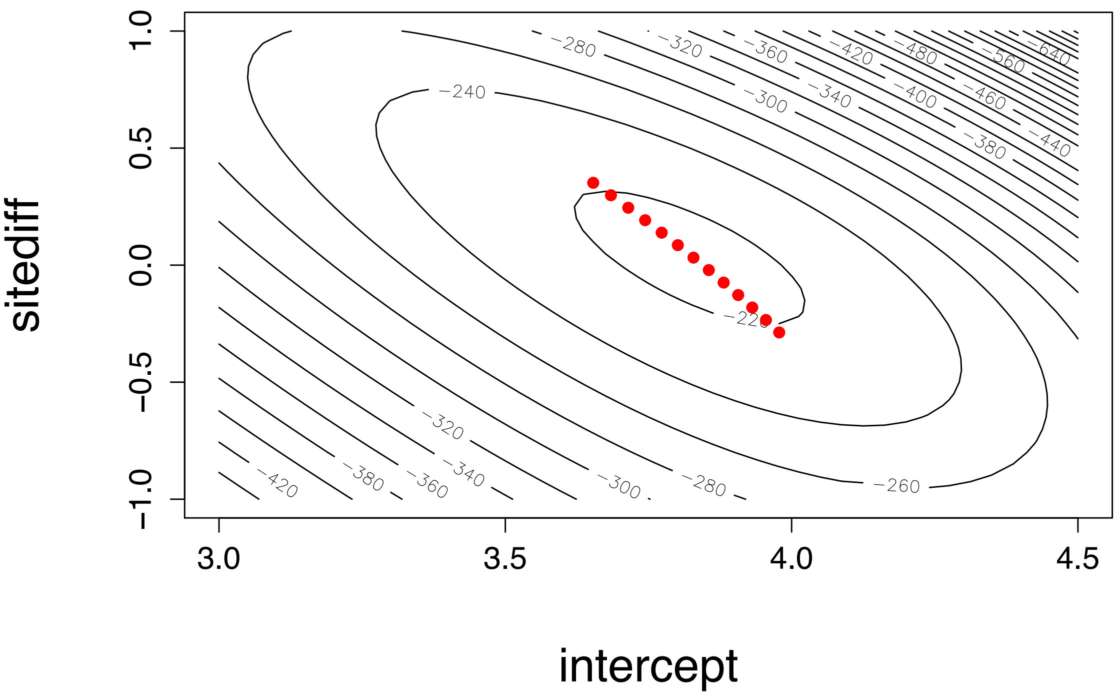
3. Compare *LR* to a chi-squared distribution with *df* = the number of parameters in the term you removed.

4. If the p-value from the chi-squared distribution is small enough, then you can reject the null hypothesis (i.e., you can reject that $\theta = 0$).

An when you do this in R it will be even easier, because the built-in functions will help you do this in two steps.

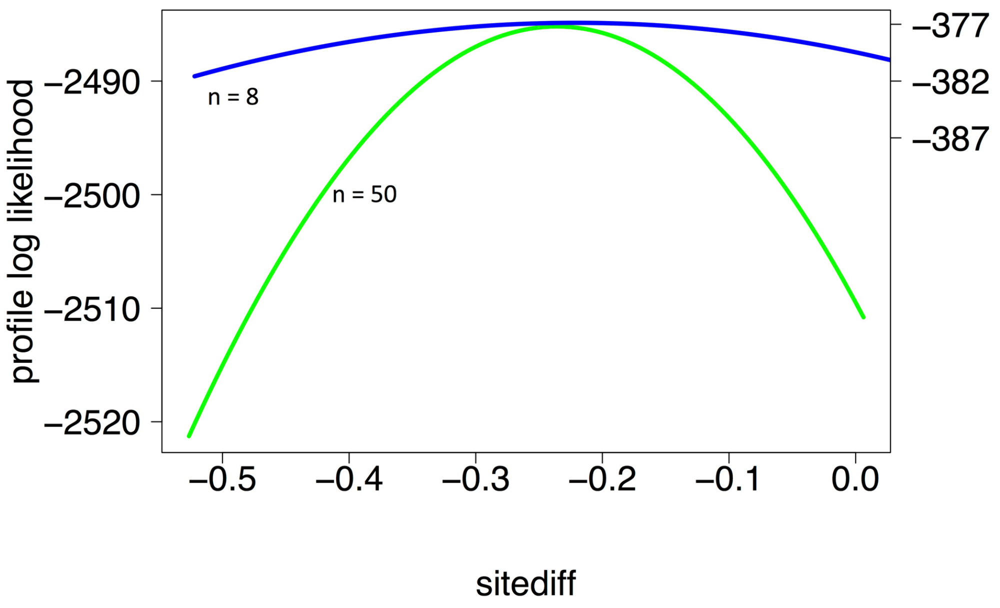
Likelihood profile and confidence intervals

In the last lecture I plotted some examples of a *likelihood curve*. In order to properly think about uncertainty in a parameter like *sitediff*, we need to use the *likelihood profile*. A likelihood curve and a likelihood profile are similar but not identical. A likelihood surface says "how does the likelihood vary as I vary some parameter θ, holding any other parameters in the model constant?". This can be useful for exploring and visualizing how likelihood works. In contrast, to generate a likelihood profile we vary some parameter θ, and for each value of θ we re-fit the model while holding θ at our chosen value. For example, in the fish data from two sites the model has two parameters, *intercept* and *sitediff*. When we fit the model we find the value of these two parameters that maximizes the likelihood. When we want to do hypothesis tests on *sitediff*, we need to say "what is the likelihood of the model when we force *sitediff* to have some value?". To do this, we need to refit the model because the MLE for *intercept* will change if the value of *sitediff* changes. We can see this from this contour plot of the likelihood surface:



The contour plot shows the log-likelihood as a function of the two model parameters. As I'll explain shortly, to do a likelihood ratio test or get a confidence interval for *sitediff*, we'll vary it over some range of values and then refit the model. The red points show how the MLE for *intercept* changes as we change *sitediff*. This is the likelihood profile. Why does the MLE for *intercept* change as we change *sitediff*? [finish this]

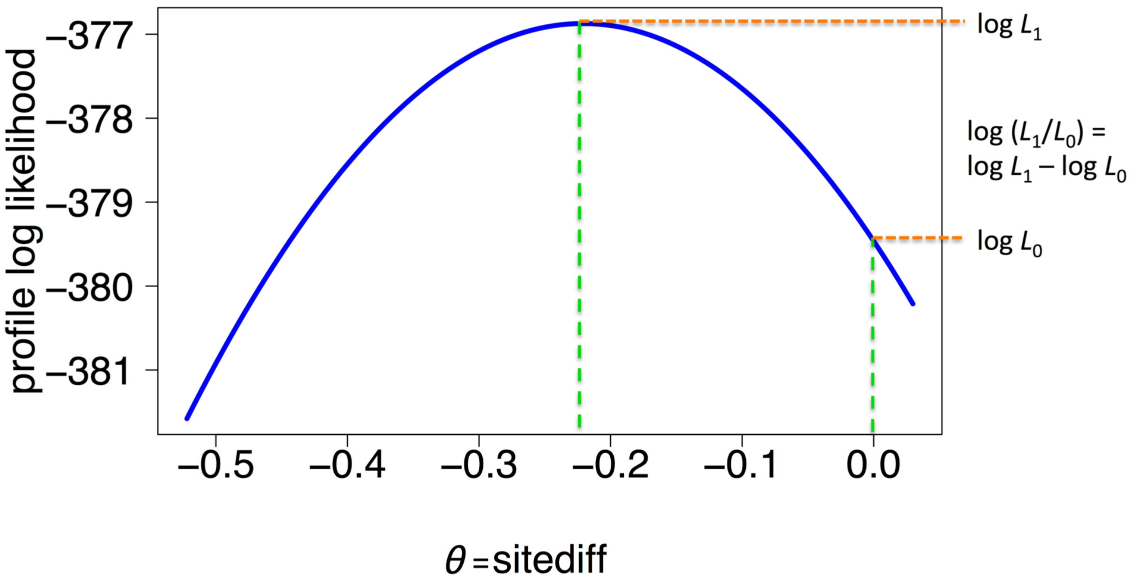
The next plot shows how the likelihood profile for *sitediff* changes when we use different amounts of data to fit the model:



I varied the parameter *sitediff* on the x-axis, and on the y-axis I calculated the log-likelihood of the model for that parameter value. The blue profile uses only 8 data points (4 per site), while the other uses 50. How do the likelihood profiles differ? The most noticeable change is that the curvature increases as the sample size increases. The y-axes are not the same (the axis for the blue profile is on the right), but the range of the y-axis is the same for the two curves. That means the curve for n = 50 shows a change in log-likelihood of ~25 from the peak of the curve to *sitediff* = 0. In contrast the curve for n = 8 only changes by ~3 units for a similar change in the parameter.

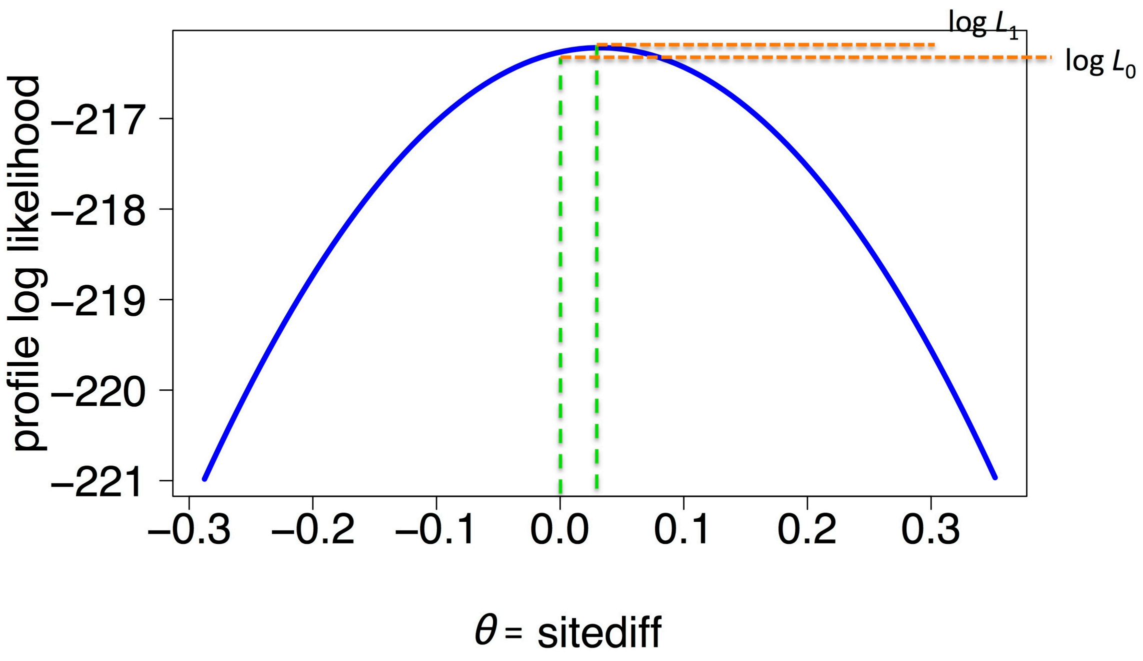
The curvature of the likelihood profile is essentially telling us how confident we are in the maximum likelihood estimate. For a shallow profile like n=8, a range of different parameter values have similar likelihoods; in other words, the limited data are not telling us as much about which parameter values are likely to be the true ones.

The logic of the likelihood ratio test can also be visualized using a likelihood profile:



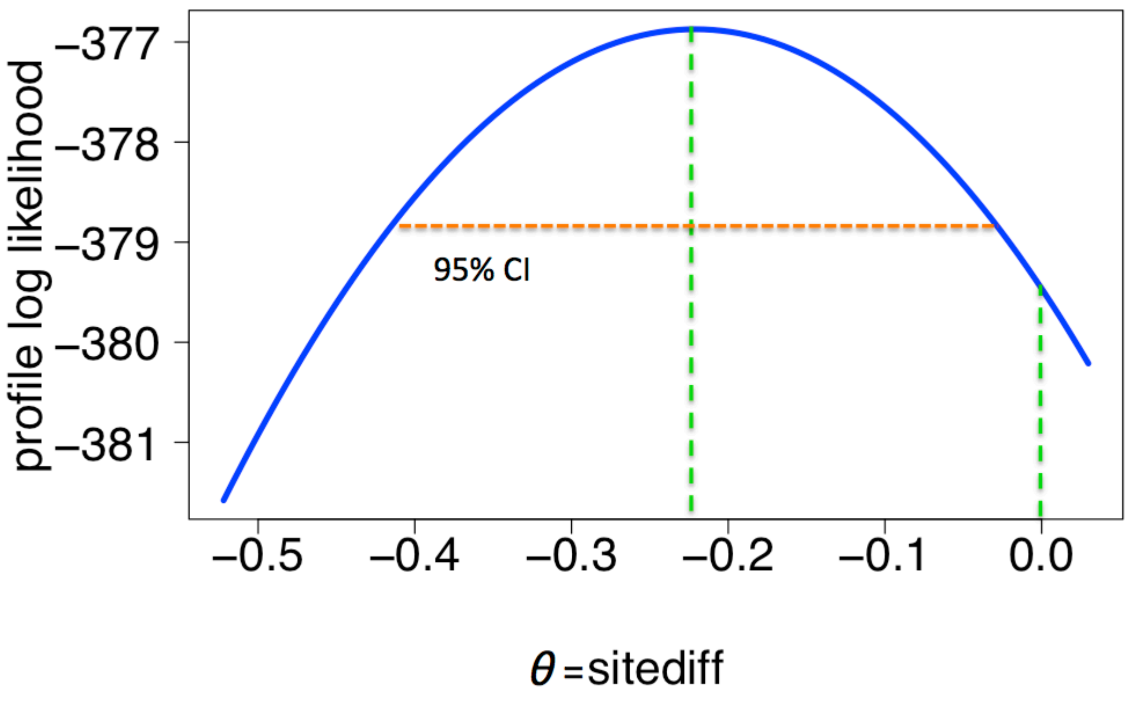
This plot zooms in on the blue curve from the previous plot (n = 8). The peak of the curve is the maximum likelihood estimate for the parameter. We want to test whether this is different from zero. So we calculate the log-likelihood at the peak, and also at the location on the curve where sitediff = 0. Then we use those log-likelihoods to compare *LR* to a chi-square distribution with df = 1. In this case p = 0.023, so we can reject the null hypothesis that sitediff = 0.

Here's an example to show what this looks like when the null hypothesis is true. I've reshuffled the underyling data so that the two sites actually have the same mean fish abundance. Then I plotted the likelihood profile for the same model as before:

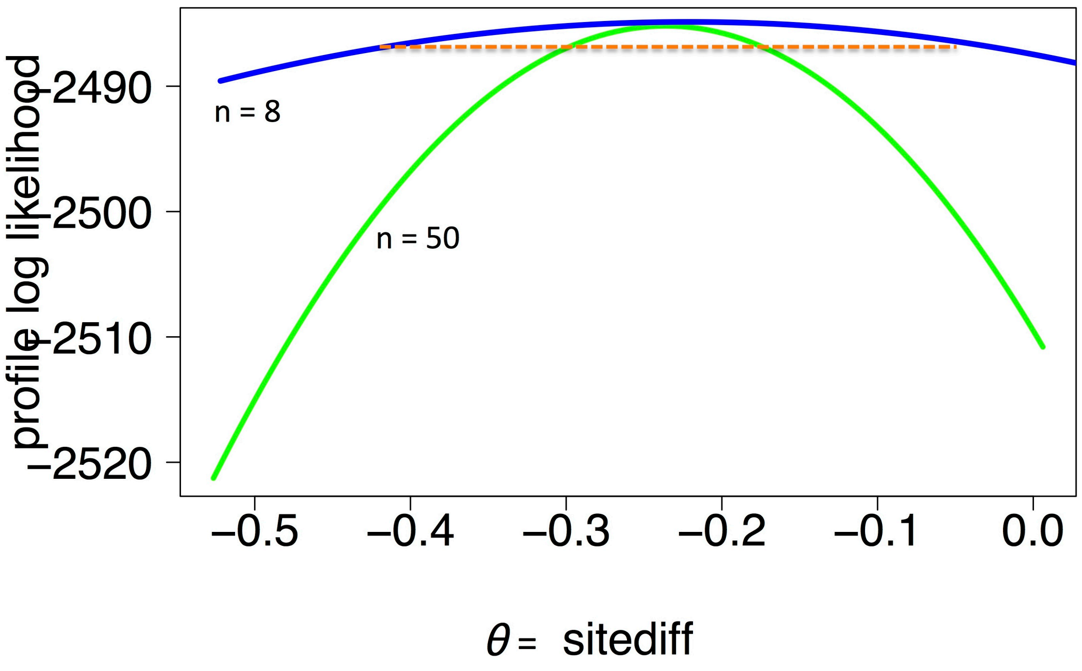


So now the true value of θ is zero. The MLE for θ is not zero, because the data are a sample (by definition) and therefore the parameter is estimated with some error (this of course is the essence of statistics). Note that the likelihood for the full model, L_1, is slightly larger than the likelihood for the null model, L_0. This is the case even though in this instance the null model is true. The reason is that the θ parameter is free to vary, and therefore noise in the data will be estimated as some difference between the two sites in fish abundance. This is why we need to compare the likelihood ratio to the chi-square distribution: even if a term in the full model is truly zero, it will have some non-zero estimate due to sampling error, and because this fits the data better this will yield a higher likelihood. The chi-squared distribution tells us when the difference in likelihoods is large enough that it's probably not due to sampling error.

The likelihood profile is used, in combination with the likelihood ratio test, to calculate confidence intervals for parameters. For a 95% confidence interval, we ask where on the likelihood profile would we reject the null hypothesis at a p = 0.05 threshold:



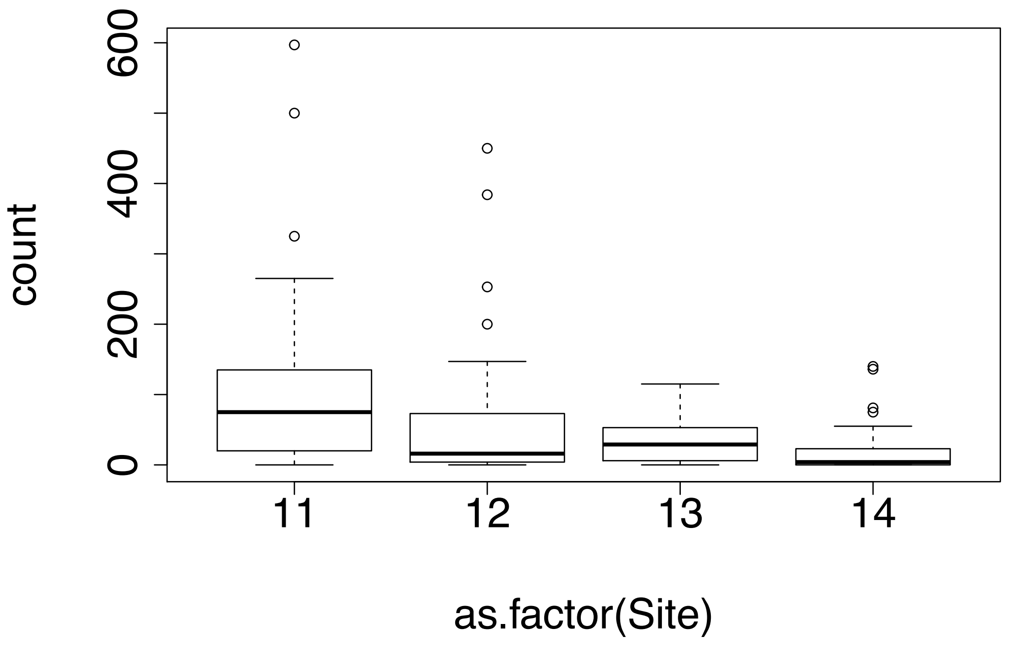
We can also compare the profile confidence intervals for the two subsets of the data:



The profile for n=50 is much steeper, because now we have much more confidence about the value of this parameter. The dotted orange line intersects both curves at their 95% confidence interval. The interval for n=50 is much narrower.

Likelihood ratio tests vs. t-tests and F-tests

I am outlining likelihood methods here so that we can start using generalized linear models and other methods where likelihood is the preferred way to the fit a model and test hypotheses with it. In the example I used above, we were interested in testing a single parameter that quantified whether two sites had different mean abundances. The same approach is used when we are interested in testing factors with >2 levels. For example, if we were comparing fish abundance at four sites:



then we would use a version of the same model:

$$\mu_i = \exp(intercept + sitediff2 * X_{1i} + sitediff3 * X_{2i} + sitediff4 * X_{3i})$$
$$Y_i \sim \text{Poisson}(\mu_i)$$

Where the *intercept* tells us the mean abundance at the first site, *sitediff2* tells us the difference in abundance between site two and site one, *sitediff3* tells us the difference in abundance between site three and site one, etc. And the *X*'s are indicator variables to code which site a particular sample comes from.

If we want to test whether there is significant variation among sites, we just need to compare this full model with a restricted model where all of the *sitediff* parameters are set to zero. The resticted model only has the *intercept* parameter, and so that model would be saying 'all sites have the same mean, quantified by the *intercept* coefficient'. We would compare the full and restricted models using a likelihood ratio test, and we would use a chi-squared distribution with *df* = 3, because the restricted model has three parameters set to zero, compared to the full model. Now let's consider how testing hypotheses is similar and different between standard linear models with normally distributed error, and the more general case of the likelihood approach. Recall from last lecture that linear models are fit using the least-squares method, and this method is a special case of maximum likelihood. So the parameter estimates you get with lm() are maximum likelihood estimates. Once we fit the model, how do we test hypotheses? Depending on the situation, either a t-test or an F-test is appropriate.

Let's look at the output from summary() for a model where the abundance of fish is modeled with a Site variable (4 sites) and a Year variable, to test whether there is an overall increase in abundance over time.

```
mod = lm(count ~ Site + Year, data = fish.sub)

summary(mod)

##
## Call:
## lm(formula = count ~ Site + Year, data = fish.sub)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -105.9  -38.1  -15.7   17.3  491.6
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  650.180   1617.498    0.40    0.688
## Site12       -49.263     16.369   -3.01    0.003 **
## Site13       -71.048     16.200   -4.39  1.9e-05 ***
## Site14       -86.787     16.376   -5.30  3.3e-07 ***
## Year          -0.273      0.810   -0.34    0.737
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 79.8 on 186 degrees of freedom
## Multiple R-squared:  0.148,  Adjusted R-squared:  0.129
## F-statistic: 8.06 on 4 and 186 DF,  p-value: 5.16e-06
```

When R reports the coefficients, we get the coefficient estimate, a standard error, a t-statistic, and a p-value from the corresponding t-test. This t-test is testing whether the coefficient is different from zero. For the special case of a linear model with normal error, the coefficient estimates follow a t-distribution with 1 degree of freedom, so this is a good test. However, for the Site variable we are usually interested not in the individual coefficients, but in whether there is significant variation across all sites. So these t-tests aren't useful. In contrast, for the Year coefficient we do want to know whether it is different from zero, so the t-test is the appropriate test.

To test a factor like site we use an F-test, which tests whether the variation accounted for by Site is bigger than expected based on how much residual variation there is. How about the F-test that is returned by summary(), at the bottom? This is an F-test for the *whole model*, i.e. it is testing whether all the terms collectively explain significant variation. This usually isn't the test we're interested in, so let's ignore that.

R has a built-in anova() function. This function is fine for a one-way anova, i.e. the only term in the model is a single factor. Problems arise with this function when you are interested in tests on a model with multiple terms. When a model has multiple terms (Site, Year), you want to test a particular term (Site) after accounting for any variation explained by the other terms (in this case, Year). This is called a marginal test, and anova() does not necessarily return marginal tests. I recommend not using it for this kind of situation (F-tests on a model with multiple terms). It is possible to get appropriate marginal tests with the drop1() function, but there are still some cases where you have to be careful to get sensible results. Instead, I recommend using the Anova() function in the "car" package, which always gives marginal tests.

Let's do Anova() on the model:

```
Anova(mod)
```

```
## Anova Table (Type II tests)
##
## Response: count
##            Sum Sq  Df F value  Pr(>F)
## Site       204728   3   10.73 1.6e-06 ***
## Year          721   1    0.11    0.74
## Residuals 1183366 186
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We get a test for Site, and a test for Year. We wanted a test for Site so that's good. What's up with the test for Year? We already tested that earlier with a t-test, but now there's an F-test. It turns out that these are equivalent (note the same p-value). *When testing a single parameter, the F-statistic is equal to the t-statistic squared.* So you could use either one.

One more point about F-tests. F-tests are used in linear models to compare the variance explained by some term to the residual variance. You can also use an F-test to test whether any two variances are equal (this is called var.test in R). I rarely see this used, but in any case don't confuse it with the F-test that is returned by Anova.

This section is a digression to review how terms are tested in linear models, to make sure we know how to do this but also to compare to the likelihood ratio tests described earlier. Here's the punchline: *likelihood ratio tests are analogous to F-tests, but they apply to any model that can be fit with maximum likelihood*. To test the importance of a factor we can use a likelihood ratio test that removes that factor from the model, or for linear models we can use a (marginal) F-test. To test the importance of a single parameter (e.g. a slope parameter) we can use a likelihood ratio test, or for linear models we can use a t-test (which is equivalent to an F-test).

In fact, F-tests can be derived as a special kind of likelihood ratio test. Earlier I explained how the log likelihood ratio, multiplied by 2, is approximately distributed as a chi-squared distribution. This approximation is derived assuming a large sample size. The likelihood ratio test is used for small sample sizes as well, because we usually don't have a better (convenient) option. For the special case of linear models with normal error, *the F-test is a better likelihood ratio test that accounts for sample size*. So really all the common tests are likelihood ratio tests, and linear models have a special kind of likelihood ratio test that is more realistic about what to expect with limited data.