Maximum Likelihood for Functions

Recall: Fitting Lines to Data with Least Squares

We previously learned how to fit a line to data by using the Method of Least Squares.

That is, we choose the parameters of the line to minimize the sum of the squares of the residuals/errors,

$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 = \sum_{i=1}^{n} (Y_i - [b_0 + b_1 X_i])^2.$$

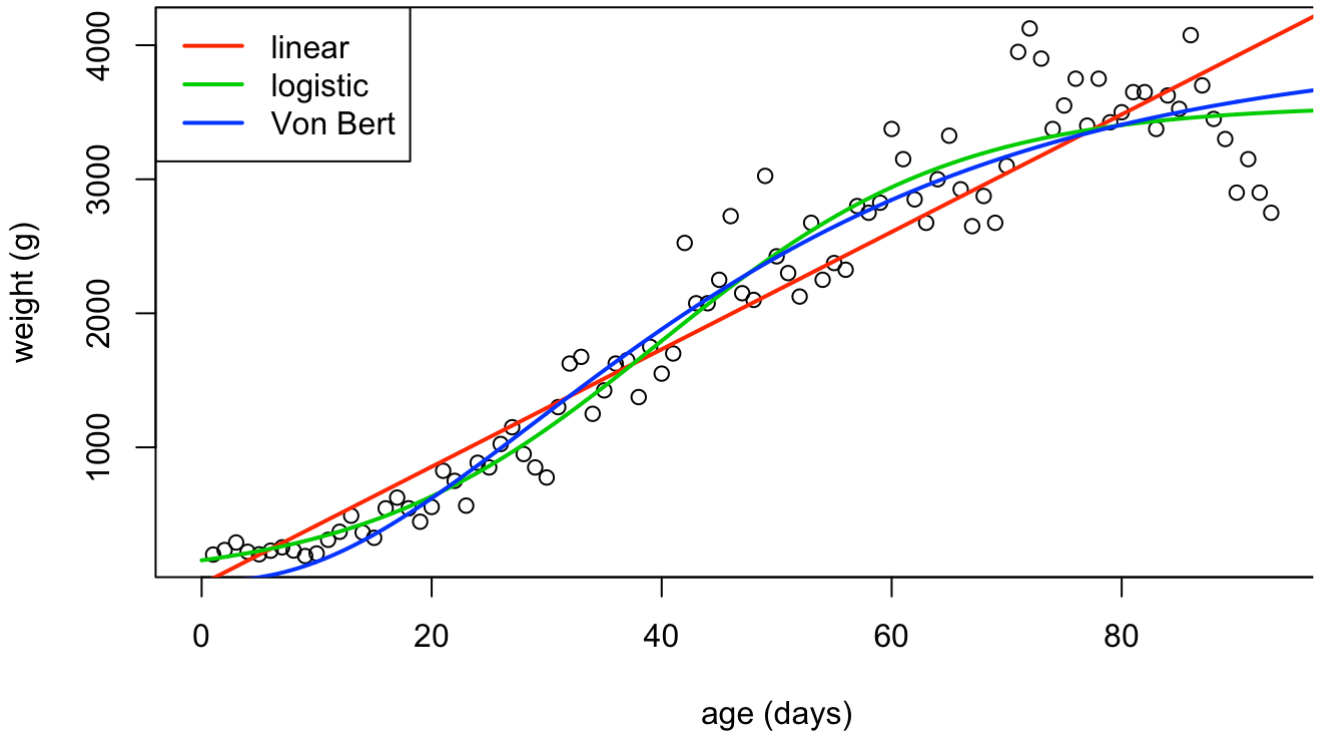
The corresponding least squares estimates for b_0 and b_1 are:

$$b_0 = \bar{Y} - b_1 \bar{X} \quad \text{and} \quad b_1 = \frac{\sum_{i=1}(X_i Y_i) - n\bar{Y}\bar{X}}{\sum_{i=1} X_i^2 - n\bar{X}^2}.$$

where $\bar{X}$ denotes the arithmetic mean of X.

Recall: Fitting Functions to Data with Least Squares

We also fit other functions to data, like the logistic function to the albatross data. (Code for this is in the accompanying R file.)



Fitting functions using Maximum Likelihood

An alternative to minimizing the sum of squared errors is to find parameters to the function such that the of the parameters, given the data and the model, is maximized.

Recall that we denote the pmf (pdf) as $f(Y_i)$, and it tells us the probability (density) of some yet to be observed datum Y_i given a probability distribution and its parameters.

We make many observations, $\mathbf{Y} = y_1, y_2, \ldots, y_n$, and are interested how probable it was that we obtained these data, jointly. We call this the "likelihood" of the data, and denote it as

$$\mathcal{L}(\theta; Y) = f_\theta(Y)$$

where $f_\theta(Y)$ is the pdf (or pmt) of the data
θ.

Maximum Likelihood for SLR

Recall that SLR assumes every observation in the dataset was generated by the model:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \quad \varepsilon_i \overset{\text{iid}}{\sim} \text{N}(0, \sigma^2)$$

This is a model for the conditional distribution of Y given X.

The pdf for the normal distribution is given by

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

Maximum Likelihood for SLR

In the SLR model, the conditional distribution has this distribution. That is, for any single observation, y_i

$$f(y_i|\beta_0,\beta_1,x_i) = \frac{1}{\sqrt{2\sigma^2\pi}}\exp\left(-\frac{(y_i-(\beta_0+\beta_1 x_i))^2}{2\sigma^2}\right)$$

If we interpret this function as a function of the parameters $\theta = \{\beta_0,\beta_1,\sigma\}$, then it gives us the likelihood of the i^{th} data point.

As we did for the simple binomial distribution, we can use this to estimate the parameters of the model.

Exercise: MLE for SLR

You have taken n sets of responses y_i with a covariate x_i. Thus, you have data $(X, Y) = (x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$. You want to fit the SLR model to these data. Assume that you know the variance, σ (i.e., treat it as a known constant).

1. Write down the likelihood and log-likelihood for the data.
2. Simplify the log-likelihood using the definition of the arithmetic mean of data: $\bar{x} = \frac{1}{m} \sum_{i=1}^{m} x_i$.
3. Take the derivative of the negative log-likelihood with respect to each β_0 and β_1, set the two resulting equations equal to zero, and find the MLEs of both parameters, $\hat{\beta_0}, \hat{\beta_1}$.
4. Do these equations look familiar? Where in this class have you seen these before?

Implementing the Likelihood in R

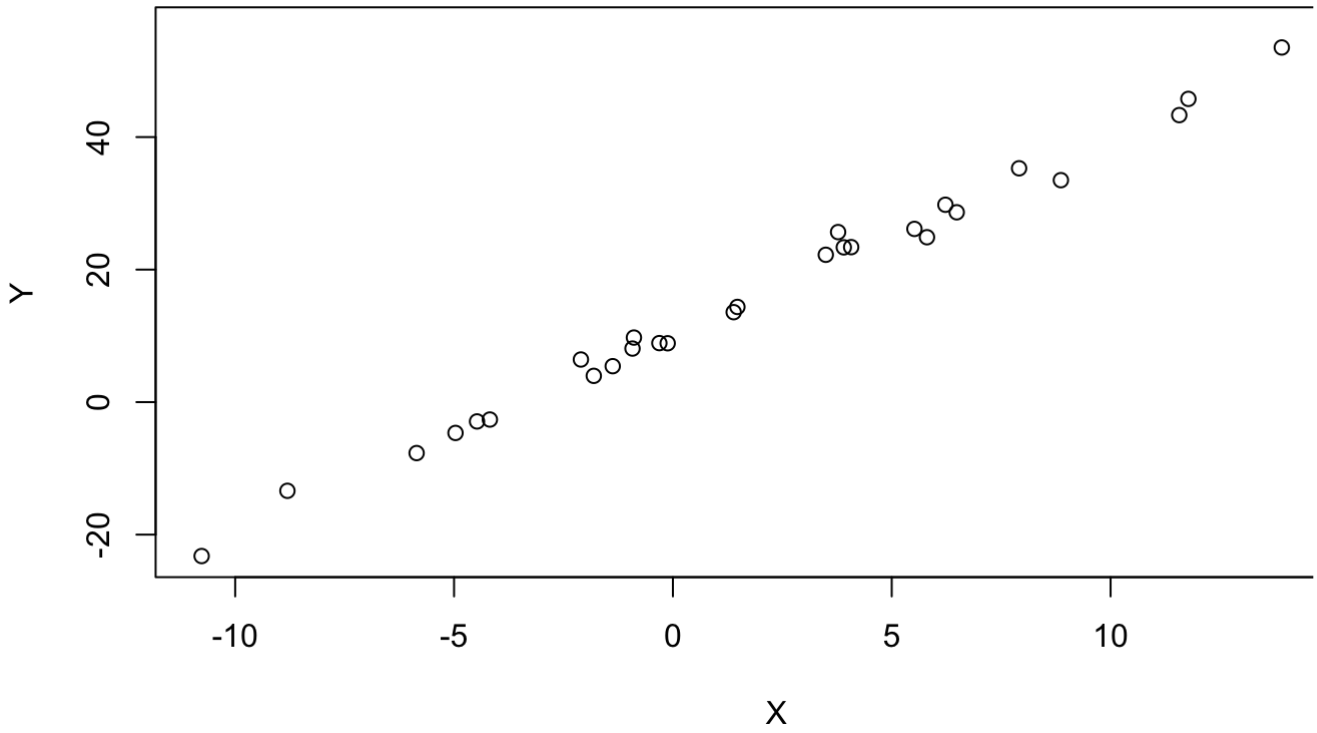
We can use a similar approach to what you used in your lab to implement the (negative log) likelihood for SLR in R. Note that I'm doing something a bit unintuitive here - I do it this way because I want to be able to use (later).

```
nll.slr<-function(par, dat, ...){
  args<-list(...)

  b0<-par[1]
  b1<-par[2]
  X<-dat$X
  Y<-dat$Y
  if(!is.na(args$sigma)){
    sigma<-args$sigma
  }else sigma<-par[3]

  mu<-b0+b1*X

  return(-sum(dnorm(Y, mean=mu, sd=sigma, log=TRUE)))

}
```

Implementing the Likelihood in R

I'm going to generate some simulated data, assuming that:

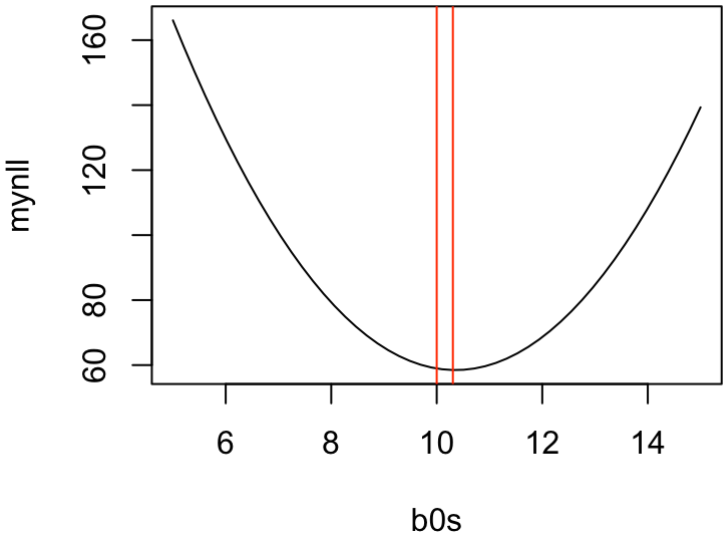
$$\beta_0 = 10, \beta_1 = 3, \text{ and } \sigma = 2.$$



Likelihood profile in R

For now, let's assume that I know what β_1 is. Let's build a likelihood profile for my simulated data:

```
N<-50
b0s<-seq(5, 15, length=N)
mynll<-rep(NA, length=50)
for(i in 1:N){
  mynll[i]<- nll.slr(par=c(b0s[i],b1), dat=dat, sigma=sigma)
}
```

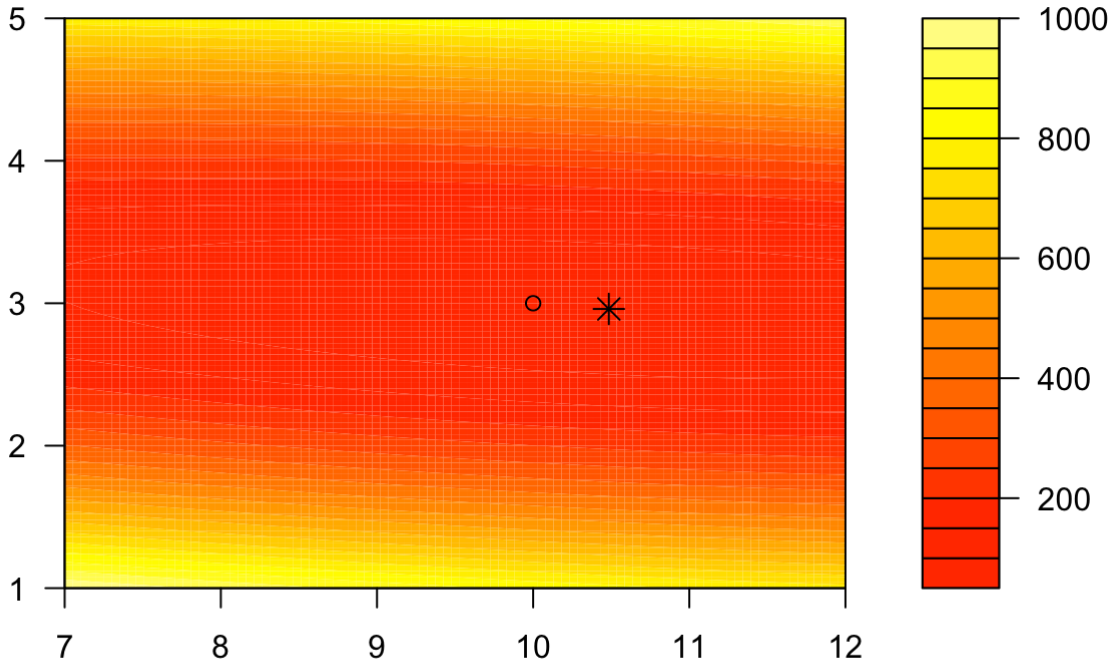


Likelihood surface in R

If we wanted to estimate both β_0 and β_1 the simplest approach is to do a **grid search** to find the maximum likelihood estimators.

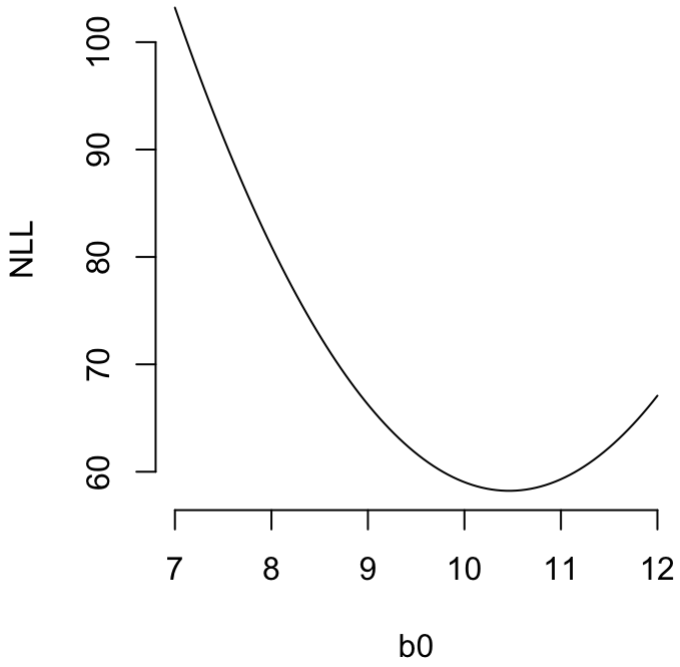
```
N0=100
N1=101
b0s<-seq(7,12, length=N0)
b1s<-seq(1,5, length=N1)

mynll<-matrix(NA, nrow=N0, ncol=N1)
for(i in 1:N0){
  for(j in 1:N1) mynll[i,j]<-nll.slr(par=c(b0s[i],b1s[j]), dat=dat, sigma
}


ww<-which(mynll==min(mynll), arr.ind=TRUE)

b0.est<-b0s[ww[1]]
b1.est<-b1s[ww[2]]
rbind(c(b0, b1), c(b0.est, b1.est))


##          [,1] [,2]
## [1,] 10.00000 3.00
## [2,] 10.48485 2.96
```

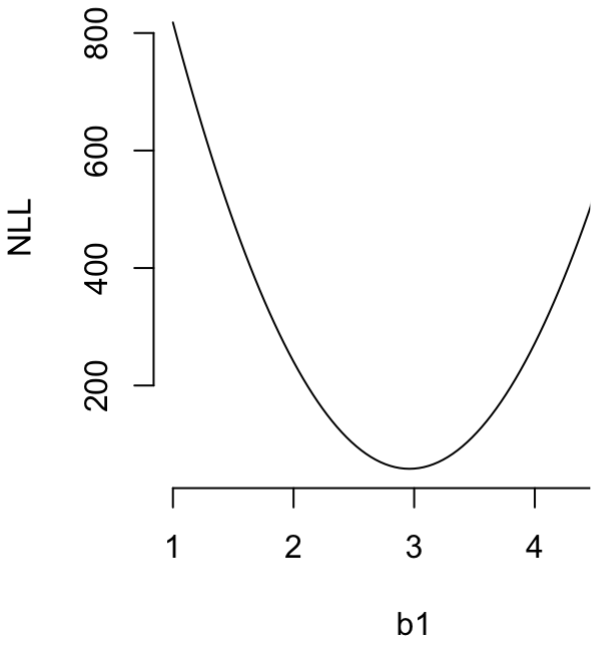
```
filled.contour(x = b0s,
               y = b1s,
               z= mynll,
               col=heat.colors(21),
               plot.axes = {axis(1); axis(2); points(b0,b1, pch=21);
                 points(b0.est, b1.est, pch=8, cex=1.5);
                 xlab="b0"; ylab="b1"})
```



Conditional Likelihood

We can also look at the conditional surfaces (i.e., we look at the slice around whatever the best estimate is for the other parameter):





Alternatives to Grid Search

There are many alternative methods to grid searches. Since we are seeking to minimize an arbitrary function (the negative log likelihood) we typically use a descent method to perform general optimization.

There are lots of options implemented in the function in R. We won't go into the details of these methods, due to time constraints. However, I typically use:

- Brent's method: for 1-D search within a bounding box, only
- L-BFGS-B (limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm with bounding box constraints): a quasi-Newton method, used for higher dimensions, when you want to be able to put simple limits on your search area.

Maximum likelihood using

The first argument is the function that you want to minimize, and the second is a vector of starting values for your parameters. After the main arguments, you can add what you need to evaluate your function (e.g.).

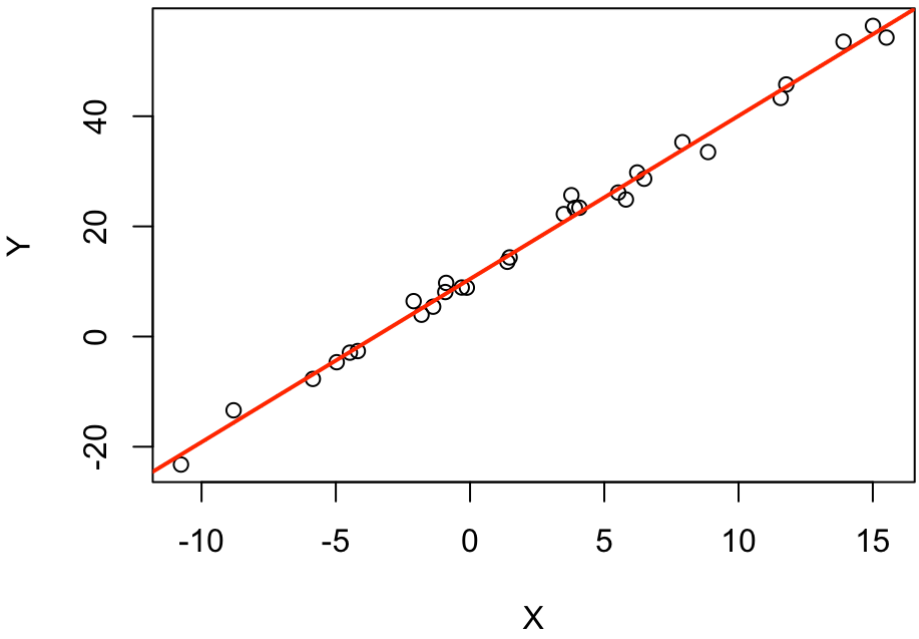
```
fit <- optim(nll.slr, par=c(2, 1), method="L-BFGS-B", ## this is a n-D me
             lower=-Inf, upper=Inf, dat=dat, sigma=sigma)

fit
```

```
## $par
## [1] 10.458935  2.961704
##
## $value
## [1] 58.22473
##
## $counts
## function gradient
##       12       12
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

I can also fit sigma as the same time, if I want:

```
fit <- optim(nll.slr, par=c(2, 1, 5), method="L-BFGS-B", ## this is a n-D
             lower=c(-Inf, -Inf, 0.1), upper=Inf, dat=dat, sigma=NA)

fit$par
```

```
## [1] 10.458945  2.961704  1.621689
```



Confidence intervals

The joint distribution of the MLEs are asymptotically Normally distributed. Given this, if you are minimizing the negative log likelihood (NLL) then the covariance matrix of the estimates is (asymptotically) the inverse of the Hessian matrix. The Hessian matrix evalutes the second derivatives of the NLL (numerically here), which gives us information about the curvature the likelihood. Thus we can use the Hessian to estimate confidence intervals:

```
fit <- optim(nll.slr, par=c(2, 1), method="L-BFGS-B", hessian=TRUE,
             lower=-Inf, upper=Inf, dat=dat, sigma=sigma)

fisher_info<-solve(fit$hessian)
est_sigma<-sqrt(diag(fisher_info))
upper<-fit$par+1.96*est_sigma
lower<-fit$par-1.96*est_sigma
interval<-data.frame(value=fit$par, upper=upper, lower=lower)
interval
```

```
##       value     upper    lower
## 1 10.458935 11.228565 9.689305
## 2  2.961704  3.067705 2.855704
```

Compare to fitting with

We can, of course, simply fit the model using the function:

```
lmfit<-lm(Y~X)

summary(lmfit)$coeff
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 10.458936 0.32957007 31.73509 1.699822e-23
## X            2.961704 0.04539126 65.24834 3.874555e-32
```

The estimates we get using are almost identical to the estimates that we obtain here, and the standard errors on the intercept and slope are very similar to those we calculated from the Hessian (est_sigma= 0.3926683, 0.0540817).

Exercise: MLEs for simple trait data

For this exercise you will use the same data + function that you used to practice fitting curves using non-linear least squares methods.

1. Using the function as an example, write a function that calculates the negative log likelihood as a function of the parameters describing your trait and any additional parameters you need for an appropriate noise distribution (e.g., σ if you have normal noise).
2. For at least one of your parameters plot a likelihood profile given your data, with the other parametes fixed.
3. Use the function to find the MLE of the same parameter and indicate this on your likelihood profile.
4. Obtain a confidence interval for your estimate.