**Likelihood Profiles**

Task: Create a likelihood profile for the quadratic coefficient of elevation for the exercise of magnolia presence-absence from last week. Does the confidence interval suggest the quadratic term is significant? (Note this can be tricky because of the tendency for parameter values to produce predicted values of the binomial probability outside [0,1]. I used a starting parameter set of 0 for the intercept and 0.7 for the linear slope. Use a small starting range for the profiled parameter, near its MLE).

First, let’s review the magnolia presence-absence exercise from last week:
Write a binomial function (using 'dbinom') for the magnolia dataset, using a deterministic model of elevation with both linear and quadratic terms (i.e., 3 parameters, including the intercept) and a binary response variable (presence-absence). What parameter values are produced by 'optim'? Plot the fitted model on a plot of presence-absence vs. elevation.

```r
>######### LOAD DATA
>dat = read.csv(file.choose()) #Importing the dataset
>dat2 = subset(dat,dat$species=="Magnolia fraseri") #Fraser's magnolia
>m.plots = unique(as.character(dat2$plotID)) #plots with mag fra
>u.plots = unique(as.character(dat$plotID)) #all plots
>nob.plots = u.plots[is.element(u.plots,m.plots)==F] #plots without mag fra
>dat3 = subset(dat,is.element(as.character(dat$plotID),nob.plots)) #dataset of no mag fra
>dat4 = subset(dat3,duplicated(dat3$plotID)==F) #one row per plot
>dat4$cover = 0 #cover of magnolia is zero in these plots
>mf = na.exclude(rbind(dat2,dat4)) #new dataframe of presences and absences

>######## WRITE THE LIKELIHOOD FUNCTION
>cover= mf$cover
>elev = mf$elev/1000 ###elevation in km (as Jason coded)
>
```

```r
>fn = function(par){
+{ -
+sum(dbinom(as.numeric(cover>0),prob=par[1]+par[2]*elev+par[3]*I(elev^2),size=1,log=T))}

>######### RUN THE OPTIMIZATION ROUTINE
>start.par = c(.1,0.0001,0)
>out1 = optim(fn,par=start.par,method="Nelder-Mead")
>out1
```

We get the following parameter estimates and negative log likelihood (NLL), with no error messages:

```
$par
 [1]  0.3164841  0.4413364 -0.3013122

$value
 [1] 619.9254

$counts
 function gradient
     374       NA

$convergence
 [1] 0

$message
NULL
```
Note, if the elev variable is left in the original units (m), rather than changed to km, we get the following output from our likelihood function, with no error messages:

```r
$par

$value
[1] 622.5487

$counts
function gradient
318       NA

$convergence
[1] 0

$message
NULL
```

>######### GRAPH THE OUTPUT
>plot1 = plot(elev, cover>1)
>x = seq(0,2,length=100) ###as Jason coded
>######### NOTE: LEAVING ELEV IN METERS, I CODED x = seq(0,2000)
>lines(x,out1$par[1]+out1$par[2]*x+out1$par[3]*x^2),col="blue",lwd=3)
Now that we have parameter estimates and the NLL, let’s create a likelihood profile for the quadratic elevation term in the magnolia presence-absence function.

```r
#******** GENERATE THE NLLs FOR MODELS WITH DIFF COEFFICIENTS FOR THE QUADRATIC TERM
> coefs = seq(-.35,-.1,length=40)  #vector of possible coefficient values around the MLE
> lik.out = numeric(length(coefs))  #holding vector (currently zeros)
> fn2 = function(par,m) {           #new simplified likelihood fn (only 2 fitted pars)
+     -sum(dbinom(as.numeric(cover>0),prob=par[1]+par[2]*elev+m*I(elev^2),size=1,log=T))
> lik.out = optim(fn2,par=c(0,0.7),m=coefs[i])$value }

When we run the above code, a solution is found with no error messages.

Note, if the elev variable is left in the original units (m) the optimization routine fails (even with various adjustments to the initial parameters) and produces the following error message:

```
Error in optim(fn2, par = c(0, 1e-04), m = coefs[i]) :  
  function cannot be evaluated at initial parameters
In addition: Warning message: 
In dbinom(x, size, prob, log) : NaNs produced
```

```r
#******** REMEMBER THAT OUR CONFIDENCE INTERVAL IS BASED ON CHI-SQUARED CRITICAL VALUE
> qchisq(0.95,1)/2
[1] 1.920729

#******** GENERATE AND PLOT VALUES FOR THE CONFIDENCE INTERVAL
> plot(coefs,lik.out,type="l"); abline(v=out1$par[3],col="blue",lwd=2)
> upper.s = coefs[which.min(lik.out):length(lik.out)]  #x values, right side
> upper.k = lik.out[which.min(lik.out):length(lik.out)]  #y values, right side
> upperCI = approx(upper.k,upper.s,xout=min(lik.out)+qchisq(0.95,1)/2)$y
> abline(v=upperCI,lty=2,col="red")
```

This plot shows the NLL values (y-axis) corresponding to various values of the quadratic parameter derived from the simulation (x-axis).

The location of the confidence boundary (dashed red line) indicates that the parameter estimate originally found for the quadratic elevation term in our model (solid blue line) is significant (does not overlap with zero).
Bayesian Analysis Using WinBUGS

Task: Rewrite the model we used in class to produce the posterior distribution of the elevation coefficient in a simple 3-parameter regression model of magnolia cover and elevation. Use the OpenBUGS manual to see the required format for loading a dataset file. (There are two acceptable formats, the easier of which is the ‘rectangular format’ that is essentially a text file of data columns, each with a header row of the variables [each followed by brackets], and with a final row containing only END). Using the write function in R (or write.csv), export a Fraser magnolia cover dataset that includes at least two columns (cover and elevation), reformat it in a text editor (or Excel), and load it into OpenBUGS.

Recall that the model we ran in class was an example of simple regression with normal error, with three parameters to be estimated:

- \( a \) = intercept
- \( b \) = slope
- \( p \) = precision (the inverse of variance)

We’re using flat priors and give WinBUGS initial values for the parameter estimates.

The first step is to reformat the tree data. I opened treedata.csv, eliminated all values not associated with Fraser magnolia, eliminated all columns except for cover and elev, and saved the results as a *.txt file like this:

```r
list (N = 400)
cover[ ] elev[
6  896.1
3  947.3
6  1027
2  461.8
5  646.7
7  1200
4  941
6  927.7
...
...
END
```

Next, we launch WinBUGS, click File -> New, and create the model to be run. We modify Jason’s template only by changing the sample size in the “for loop” from 9 to 400, and eliminating the list of data used in class:

```r
model{
  a ~ dnorm(0,1.0E-6)  #flat prior for the intercept (extremely low precision), note "~" means "is distributed as"
  b ~ dnorm(0,1.0E-6)  #flat prior for the slope (ditto)
  p ~ dgamma(0.001,0.001)  #flat gamma prior for the precision, because restricted to positive values
  for(i in 1:400) #BUGS only accepts for loops rather than vector-based functions, so we loop over each observation
  {
    mean[i] <- a + b*elev[i]  #deterministic function (linear); note equal sign in BUGS is "<-" rather than "="
    cover[i] ~ dnorm(mean[i],p)  #stochastic (data) model, here normal with precision=p
  }
}
```

list(a=0,b=0,p=1) #initial values for parameters
We click **File -> New** again and paste the data we’ve generated into the new window that appears.

Now we’re ready to run the model.

Click **Model -> Specification** in the menu at the top of the screen, opening the **Specification Tool** window.

Highlight **model** in the model window, click **check model** in the **Specification Tool** window. A message should appear at the bottom of the screen: “model is syntactically correct.”

Next, highlight the column labels (**cover**, **elev**) in the data window, click **load data** in the **Specification Tool** window. A message should appear at the bottom of the screen: “data loaded.”

Click **compile** in the **Specification Tool** window. A message should appear at the bottom of the screen: “model compiled.”

Highlight **list** in the model window and click **load inits** in the **Specification Tool** window. A message should appear at the bottom of the screen: “model is initialized.”

Now, we’re ready to run the model and generate values for the parameters.

Click **Inference -> Samples** to open the **Sample Monitor Tool** window. Type **a** in the **node** box, click **set**, and repeat for **b** and **p**. Then type **p** in the **node** box so we can view output for all of the parameters at the same time.

Click **Model -> Update** to open the **Update Tool** window.

Type **1000** in the **updates** box on the **Update Tool**, asking WinBUGS to perform a thousand iterations of the Gibbs Sampler to generate the posterior distributions of the parameters.

Click **trace** in the **Sample Monitor Tool** to see graphs for all three parameters we specified. The fitted parameter value will be on the y-axis and iteration number on the x-axis. Keep clicking **updates** on the **Update Tool** to watch WinBUGS estimate the parameter values in real time. In the **Update Tool**, WinBUGS will keep a tally of the number of iterations run.
I clicked **updates** until WinBUGs ran 50,000 iterations. I discarded the first 5,000 iterations (to account for a “burn in period” with potentially high correlation between parameter estimates and initial values) by indicating 5,001 in the **beg** box on the **Sample Monitor Tool**. Then, I clicked **density** to see the posterior distributions for each parameter:

Here we see the posterior distribution for the elevation coefficient in our model.

Clicking **stats** on the **Sample Monitor Tool** opens a new window with summary statistics and parameter estimates:

<table>
<thead>
<tr>
<th>node</th>
<th>mean</th>
<th>sd</th>
<th>MC error</th>
<th>2.5%</th>
<th>median</th>
<th>97.5%</th>
<th>start</th>
<th>sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2.06</td>
<td>0.3018</td>
<td>0.001454</td>
<td>1.47</td>
<td>2.059</td>
<td>2.653</td>
<td>5001</td>
<td>45000</td>
</tr>
<tr>
<td>b</td>
<td>0.001233</td>
<td>3.209E-4</td>
<td>1.576E-6</td>
<td>6.066E-4</td>
<td>0.001234</td>
<td>0.001862</td>
<td>5001</td>
<td>45000</td>
</tr>
<tr>
<td>p</td>
<td>0.3302</td>
<td>0.02341</td>
<td>1.192E-4</td>
<td>0.2858</td>
<td>0.3295</td>
<td>0.378</td>
<td>5001</td>
<td>45000</td>
</tr>
</tbody>
</table>

Here we see that the parameter estimate for the elevation coefficient is **0.001233**.

Clicking **history** on the **Sample Monitor Tool** opens a graph window that displays the parameter estimates fit during each of the iterations:
If we run the model for 50,000 more iterations (for a total of 100,000), again discard the first 5,000, and click density and stats on the Sample Monitor Tool, we see that the posterior distributions are slightly smoother and the coefficient of elevation is slightly different.

The parameter estimate for the elevation coefficient is 0.001235.