Ecol 145 Assignment 5
Dahl Winters
2/24/06

Problem 1

The code that you used to fit the 8 new models. For the normal models that use a transformed response this should include both the code you used to fit the models as well as the code you used to construct the loglikelihoods.

1. A ZIP model in which both λ and θ are allowed to differ in the two field types

```
zip4<-function(data,p) {
        field.dummy<-as.numeric(data$field)-1   #1's will go to 0; 2's will go to 1.
        lambda<-p[1]+p[3]*field.dummy
        theta<-p[2]+p[4]*field.dummy
        zero.term<-sum(ifelse(data$slugs==0,log(theta+(1-theta)*dpois(data$slugs,lambda)),0))
        other.term<-sum(ifelse(data$slugs>0,log((1-theta)*dpois(data$slugs,lambda)),0))
        negloglike<- -(zero.term+other.term)
        negloglike
}

#getting guesses for the mean nursery value and zero fraction
tapply(slugs$slugs[slugs$slugs>0],slugs$field[slugs$slugs>0],mean)
 Nursery  Rookery
3.400000 2.935484
table(slugs$slugs,slugs$field)[1,]/40
Nursery Rookery
  0.625   0.225
```

```
nlm(function(p) zip4(slugs,p),c(3.4,.6,-.4,-.4))->out10
out10
$minimum
[1] 143.2979
$estimate
[1]  3.2709061  0.6101980 -0.5235914 -0.4382786
$gradient
[1]  2.225316e-05 -7.494805e-05 -4.939693e-05 -8.324719e-05
$code
[1] 1
$iterations
[1] 15
my.aic<-function(output) -2*(-output$minimum) + 2*length(output$estimate)
my.aic(out10)
[1] 294.5958
```

2. A normal model using a log-transformed response in which the standard deviation σ is allowed to differ in the two field types

```
norm.negloglike<-function(data,p) {
        t.y<-log(data$slugs+1)   #t is for transform. Add 1 for shifting the numbers.
        field.dummy<-as.numeric(data$field)-1   #1's will go to 0; 2's will go to 1.
        mu=p[1]
        stdev=p[2]+p[3]*field.dummy
        negloglike<- -sum(log(dnorm(t.y,mean=mu,sd=stdev)))
        negloglike
}

#getting guesses for the values of mean and sd for the nursery data
tapply(log(slugs$slugs+1),slugs$field,mean)
  Nursery   Rookery
0.4967358 0.9697582
tapply(log(slugs$slugs+1),slugs$field,sd)
  Nursery   Rookery
0.7345233 0.6776645

nlm(function(p) norm.negloglike(slugs,p),c(.5,.75,-.05))->out.norm
out.norm
$minimum
[1] 88.9453
$estimate
[1]  0.75470579  0.76979518 -0.06694717
$gradient
[1] -3.272760e-05  3.750245e-05  1.278977e-07
$code
[1] 1
$iterations
[1] 9
#can't compare this with the other AICs since it's been log transformed.

norm1<-function(data,out)
{
t.y<-log(data$slugs+1)
field.dummy<-as.numeric(data$field)-1
mu<-out$estimate[1]
my.sd<-out$estimate[2]+field.dummy*out$estimate[3]
negloglike<- -sum(log(dnorm(t.y,mean=mu, sd=my.sd)*1/(data$slugs+1)))
```

```
out<-list(negloglike,out$estimate) #making the output into a list as all the previous nlm outputs were.
names(out)<-c("minimum","estimate") #gives the same names as the nlm outputs.
out
}
```
norm1(slugs,out.norm)->out22
out22
```
$minimum
[1] 147.6051
$estimate
[1]  0.75470579  0.76979518 -0.06694717
```
my.aic(out22)
```
[1] 301.2101
```

3. A normal model using a log-transformed response in which both the mean μ and the standard deviation σ are allowed to differ in the two field types

```
norm.negloglike<-function(data,p) {
        t.y<-log(data$slugs+1)   #t is for transform. Add 1 for shifting the numbers.
        field.dummy<-as.numeric(data$field)-1   #1's will go to 0; 2's will go to 1.
        mu=p[1]+p[3]*field.dummy
        stdev=p[2]+p[4]*field.dummy
        negloglike<- -sum(log(dnorm(t.y,mean=mu,sd=stdev)))
        negloglike
}
```

nlm(function(p) norm.negloglike(slugs,p),c(.5,.75,.5,-.05))->out.norm
out.norm
```
$minimum
[1] 84.59691
$estimate
[1]  0.49673527  0.72528319  0.47302237 -0.05614363
$gradient
[1] -8.412826e-06 -8.057555e-06 -6.806999e-06 -7.162271e-06
$code
[1] 1
$iterations
[1] 15
```

```
norm2<-function(data,out)
{
t.y<-log(data$slugs+1)
field.dummy<-as.numeric(data$field)-1
mu<-out$estimate[1]+field.dummy*out$estimate[3]
my.sd<-out$estimate[2]+field.dummy*out$estimate[4]
negloglike<- -sum(log(dnorm(t.y,mean=mu, sd=my.sd)*1/(data$slugs+1)))
out<-list(negloglike,out$estimate) #making the output into a list as all the previous nlm outputs were.
names(out)<-c("minimum","estimate") #gives the same names as the nlm outputs.
out
}
```
norm2(slugs,out.norm)->out23
out23
```
$minimum
[1] 143.2567
$estimate
[1]  0.49673527  0.72528319  0.47302237 -0.05614363
```
my.aic(out23)
```
[1] 294.5133
```

4. A common normal model using a square root transformed response

```
norm.negloglike<-function(data,p) {
        t.y<-sqrt(data$slugs+1)
        mu=p[1]
        stdev=p[2]
        negloglike<- -sum(log(dnorm(t.y,mean=mu,sd=stdev)))
        negloglike
}

#getting guesses for the mean and sd of the nursery data
tapply(sqrt(slugs$slugs+1),slugs$field,mean)
Nursery  Rookery
1.381190 1.716561
tapply(sqrt(slugs$slugs+1),slugs$field,sd)
  Nursery   Rookery
0.6137855 0.5803777

nlm(function(p) norm.negloglike(slugs,p),c(1.3,.6))->out.norm
$minimum
[1] 74.38676
$estimate
[1] 1.548875 0.613175
$gradient
[1] -2.216669e-05 -9.094947e-06
$code
[1] 1
$iterations
[1] 7

norm3<-function(data,out)  #ln back transformation, multiply by 1/2sqrt y instead of 1/y.
{
t.y<-sqrt(data$slugs+1)
mu<-out$estimate[1]
my.sd<-out$estimate[2]
negloglike<- -sum(log( dnorm(t.y,mean=mu, sd=my.sd)*1 / (2*sqrt(data$slugs+1))  ) )
out<-list(negloglike,out$estimate) #making the output into a list as all the previous nlm outputs were.
names(out)<-c("minimum","estimate") #gives the same names as the nlm outputs.
out
}
norm3(slugs,out.norm)->out24
out24
$minimum
[1] 159.1684
$estimate
[1] 1.548875 0.613175
my.aic(out24)
[1] 322.3368
```

5. A common negative binomial model

```
nbin1<-function(data,p) {
        mu<-p[1]
        theta<-p[2]
```

```
        negloglike<- -sum(log(dnbinom(data$slugs,mu=mu,size=theta)))
        negloglike
}
nlm(function(p) nbin1(slugs,p),c(1.7,.7))->out3
out3
$minimum
[1] 144.3980
$estimate
[1] 1.7749971 0.7155645
$gradient
[1] -2.544348e-05 -5.653078e-05
$code
[1] 1
$iterations
[1] 5
my.aic(out3)
[1] 292.7961
```

6. A negative binomial model in which the mean μ is allowed to differ in the two field types

```
nbin2<-function(data,p) {
        field.dummy<-as.numeric(data$field)-1
        mu<-p[1]+p[3]*field.dummy
        theta<-p[2]
        negloglike<- -sum(log(dnbinom(data$slugs,mu=mu,size=theta)))
        negloglike
}
nlm(function(p) nbin2(slugs,p),c(1,.7,1))->out4
out4
$minimum
[1] 142.6750
$estimate
[1] 1.2749992 0.7859309 0.9999995
$gradient
[1] -5.283098e-06  1.563194e-06 -3.609557e-06
$code
[1] 1
$iterations
[1] 7
my.aic(out4)
[1] 291.3500
```

7. A negative binomial model in which the dispersion θ is allowed to differ in the two field types

```
nbin3<-function(data,p) {
        field.dummy<-as.numeric(data$field)-1
        mu<-p[1]
        theta<-p[2]+p[3]*field.dummy
        negloglike<- -sum(log(dnbinom(data$slugs,mu=mu,size=theta)))
        negloglike
}
nlm(function(p) nbin3(slugs,p),c(2,.2,1.6))->out5
out5
$minimum
```

```
[1] 138.2398
$estimate
[1] 2.0913799 0.2506001 1.6472437
$gradient
[1]  4.240059e-06 -1.104468e-04  9.489756e-06
$code
[1] 1
$iterations
[1] 15
```
my.aic(out5)
```
[1] 282.4796
```

8. A negative binomial model in which both the mean μ and the dispersion θ are allowed to differ in the two field types

```
nbin4<-function(data,p) {
        field.dummy<-as.numeric(data$field)-1
        mu<-p[1]+p[3]*field.dummy
        theta<-p[2]+p[4]*field.dummy
        negloglike<- -sum(log(dnbinom(data$slugs,mu=mu,size=theta)))
        negloglike
}
```
nlm(function(p) nbin4(slugs,p),c(1.2,.25,1,1.6))->out6
out6
```
$minimum
[1] 137.1253
$estimate
[1] 1.2749867 0.2840887 1.0000172 1.6449080
$gradient
[1] -3.566683e-05 -7.105427e-07  3.544126e-05  6.766300e-05
$code
[1] 1
$iterations
[1] 17
```
my.aic(out6)
```
[1] 282.2505
```

Problem 2

A final AIC table using the format recommended by Burnham & Anderson (2002) as developed in class. This table should include all 15 models, the 8 new models as well as the 7 old models.

(For the original models, see the back of the homework.)

models<-list(out1,out2,out3,out4,out5,out6,out7,out8,out9,out10,out20,out21,out22,out23,out24)
models.names<-
c('Pois.common','Pois.mean','NB.common','NB.mean','NB.disp','NB.both','ZIP.common','ZIP.lambda','ZIP.theta','ZIP.both','LN.common','LN.mean','LN.sd','LN.both','N.sqrt')

AIC.func<-function(model.list,n,modelnames) { #n = # observations in our dataset.
#for all models, pull out the likelihood, #parameters, then calculate AIC and AICc, then store in an output.
        output<-NULL
        for (i in 1:length(model.list)) {
                cur.model<-model.list[[i]] #lists must be accessed by double brackets.

```
            LL<- -cur.model$minimum
            K<-length(cur.model$estimate)
            AIC<- -2*LL+2*K
            AICc<-AIC + 2*K*(K+1)/(n-K+1)
            output<-rbind(output,c(LL,K,AIC,AICc))
        }
        colnames(output)<-c('LogL','K','AIC','AICc')
        minAICc<-min(output[,"AICc"])
        deltai<-output[,"AICc"]-minAICc
        rel.like<-exp(-deltai/2)
        wi<-round(rel.like/sum(rel.like),3)          #Akaike weights
        out<-data.frame(modelnames,output,deltai,wi)
        out
}
```

AIC.func(models,80,models.names)

	modelnames	LogL	K	AIC	AICc	deltai	wi
1	Pois.common	-176.8383	1	355.6766	355.7266	72.95659331	0.000
2	Pois.mean	-171.1275	2	346.2551	346.4070	63.63693240	0.000
3	NB.common	-144.3980	2	292.7961	292.9480	10.17796270	0.003
4	NB.mean	-142.6750	3	291.3500	291.6577	8.88763204	0.006
5	**NB.disp**	**-138.2398**	**3**	**282.4796**	**282.7873**	**0.01722825**	**0.490**
6	**NB.both**	**-137.1253**	**4**	**282.2505**	**282.7700**	**0.00000000**	**0.494**
7	ZIP.common	-150.4711	2	304.9422	305.0941	22.32407768	0.000
8	ZIP.lambda	-150.4209	3	306.8417	307.1494	24.37938513	0.000
9	ZIP.theta	-143.7118	3	293.4236	293.7313	10.96130016	0.002
10	ZIP.both	-143.2979	4	294.5958	295.1153	12.34522757	0.001
11	LN.common	-147.7365	2	299.4730	299.6249	16.85483350	0.000
12	LN.mean	-143.3864	3	292.7727	293.0804	10.31039170	0.003
13	LN.sd	-147.6051	3	301.2101	301.5178	18.74779855	0.000
14	LN.both	-143.2567	4	294.5133	295.0328	12.26280308	0.001
15	N.sqrt	-159.1684	2	322.3368	322.4887	39.71871029	0.000

Problem 3

An explanation of what the table tells you. In particular address the following things.

- Does the table reveal that there is a single best model?

The AIC_cs range from 355.7266 for the common Poisson model (Pois.common) to 282.7700 for the negative binomial model where both the mean and dispersion are allowed to vary (NB.both). All 4 of the negative binomial models have low AIC_cs (from 282-293). Other models with similar AIC_cs to these (between 293 and 295) are the ZIP model where theta is allowed to vary, the lognormal where the mean is allowed to vary, and the ZIP and lognormal where both parameters are allowed to vary. Both Poisson models are the worst, as expected, and the square root transformed normal model is also just as bad.

Judging from the Akaike weights of the negative binomial models, only NB.both and NB.disp have significant weights. NB.both is the best model because it has the lowest AIC_c (and the lowest AIC), and the largest Akaike weight. However, NB.disp has an Akaike weight of 0.490, just 0.004 different from the best model. This means it should also be considered as an equally valid model for describing the data, since 49% of the time it will come out to be the best model and 49.4% of the time NB.both will be the best model. Also, the value of Δi for NB.disp is 0.017,

which is within the 0-2 range indicating “excellent” evidence according to Burnham and Anderson. Thus, I conclude the table reveals two equally best models, one that is only marginally better than the other.

- Explain what the column of Akaike weights tell you about the various models.

The Akaike weight represents the probability that the model is the best model if we were to obtain more data and refit all the models again. Because it can be interpreted as a probability, an Akaike weight close to 1 means that the data strongly support the use of that model in describing the data. All of the models except two have an Akaike weight of 0.006 or lower, which means that of every 1000 times we fit all the models, these models will be the best only 6 or fewer times. Thus, all these models can be dismissed as good models for describing the data. In sharp contrast to the 0.006 or lower Akaike weights of these models, the NB.both and NB.disp models have weights of 0.494 and 0.490 respectively. These values allow me to conclude that of my current choices, I should use either NB.both or NB.disp to describe the data.

Problem 4

A goodness of fit test of your choice for what you deem to be the "best" model. Note: if there are a couple of models that are really close in terms of AIC_c you may wish to choose the more parsimonious model as your "best" model even if it isn't the winner in terms of AIC_c. In addition to having philosophical appeal, choosing simpler models will buy you extra degrees of freedom if you use the parametric version of the Pearson chi-squared test as your lack of fit test.

```
#Fitting the nbin4 (NB.both) model for only the nursery slugs
nbin4.nursery<-function(data,p) {
        field.dummy<-as.numeric(data$field)-1
        mu<-p[1]
        theta<-p[2]
        negloglike<- -sum(log(dnbinom(data$slugs[field.dummy==0],mu=mu,size=theta)))
        negloglike
}
nlm(function(p) nbin4.nursery(slugs,p),c(1.3,.3))->outN
outN
$minimum
[1] 57.74456
$estimate
[1] 1.2749986 0.2840896
$gradient
[1] -4.592062e-06  2.970069e-06
$code
[1] 1
$iterations
[1] 6

#Fitting the model for only the rookery slugs
nbin4.rookery<-function(data,p) {
        field.dummy<-as.numeric(data$field)-1
        mu<-p[1]+p[3]
        theta<-p[2]+p[4]
        negloglike<- -sum(log(dnbinom(data$slugs[field.dummy==1],mu=mu,size=theta)))
        negloglike
```

```
}
```
nlm(function(p) nbin4.rookery(slugs,p),c(2,1,.16,.73))->outR
outR
```
$minimum
[1] 79.38071
$estimate
[1] 2.0574981 1.0994738 0.2175015 0.8294739
$gradient
[1] 4.910779e-06 4.497949e-06 6.394885e-07 4.433787e-06
$code
[1] 1
$iterations
[1] 8
```

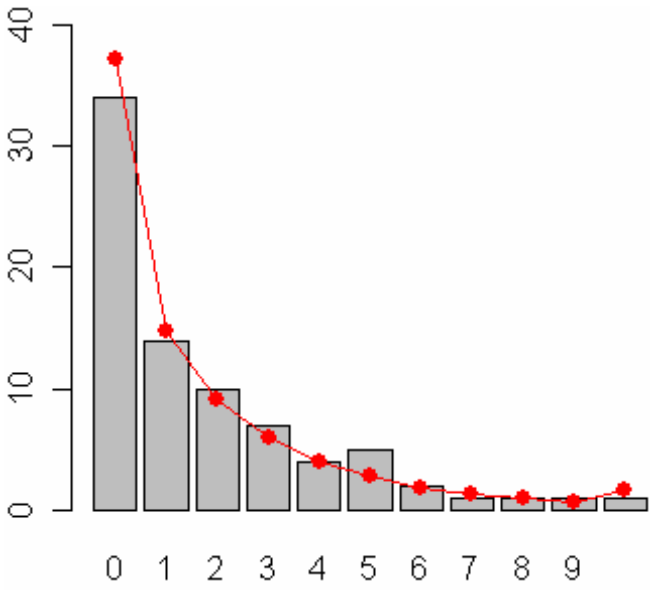
#Coming up with separate expected values for the nursery and rookery slugs
pN<-c(dnbinom(0:9,mu=outN$estimate[1], size=outN$estimate[2]), 1-pnbinom(9, mu=outN$estimate[1], size=outN$estimate[2]))
pR<-c(dnbinom(0:9,mu=outR$estimate[1], size=outR$estimate[2]), 1-pnbinom(9, mu=outR$estimate[1], size=outR$estimate[2]))

pN*40->**Ei.N**
pR*40->**Ei.R**
Ei<-Ei.N+Ei.R #combines the separate expected values into a single dataset

Ei.N
```
 [1] 24.6603805  5.7292024  3.0081433  1.8729640  1.2575447
 [6]  0.8811523  0.6346130  0.4658995  0.3469098  0.2611307
[11]  0.8820598
```
Ei.R
```
 [1] 12.5431502  8.9879415  6.1490699  4.1404245  2.7655513
 [6]  1.8382552  1.2179112  0.8050308  0.5311877  0.3500181
[11]  0.6714597
```
Ei
```
 [1] 37.2035306 14.7171439  9.1572132  6.0133884  4.0230960
 [6]  2.7194075  1.8525242  1.2709303  0.8780975  0.6111488
[11]  1.5535196
```

#Generating the observed values
table(slugs$slugs)->**Oi**
names(Oi)[length(Oi)]<-"10+"
Oi
```
  0  1  2  3  4  5  6  7  8  9 10+
 34 14 10  7  4  5  2  1  1  1   1
```

#Plotting the graph
coords<-barplot(Oi,ylim=c(0,40))
points(coords,Ei,col=2,pch=16,cex=1.2)
lines(coords,Ei,col=2)



Right: the barplot of the best model (red line) and the observed data showing a rather good fit.

#Performing the goodness of fit test
expprobs<-(pN+pR)/2
expprobs
```
 [1] 0.46504413 0.18396430 0.11446516 0.07516736 0.05028870
 [6] 0.03399259 0.02315655 0.01588663 0.01097622 0.00763936
[11] 0.01941899
```

chisq.test(Oi,p=expprobs,simulate.p.value=TRUE)
Chi-squared test for given probabilities with simulated p-value (based on 2000 replicates)
data: Oi
X-squared = 2.994, df = NA, **p-value = 0.9845**

The randomization test gives a high p-value of almost 1, which indicates that the NB.both model has a very good fit. This supports the visibly good fit of the model to the observed data that I can see from the barplot above.

Original Models (used in the table for Problem 2)

1. Common means Poisson model

```
poi.1<-function(data,p) -sum(log(dpois(data$slugs,lambda=p)))
nlm(function(p) poi.1(slugs,p),2)->out1
my.aic(out1)
[1] 355.6766
```

2. Separate means Poisson model

```
poi.2<-function(data,p) {
field.dummy<-as.numeric(data$field)-1
mylambda<-p[1]+p[2]*field.dummy
negloglike<- -sum(log(dpois(data$slugs,lambda=mylambda)))
negloglike
}
nlm(function(p) poi.2(slugs,p),c(1.2,1))->out2
my.aic(out2)
[1] 346.2551
```

3. Common mean and zero fraction ZIP model

```
zip1<-function(data,p) {
lambda<-p[1]
theta<-p[2]
zero.term<-sum(log(theta+(1-theta)*
  dpois(data$slugs[data$slugs==0], lambda)))
nonzero.term<-sum(log((1- theta)* dpois(data$slugs[data$slugs>0],
  lambda)))
negloglike<- -(zero.term+nonzero.term)
negloglike
}
nlm(function(p) zip1(slugs,p),c(3,.4))->out7
my.aic(out7)
[1] 304.9422
```

4. Separate means and common zero fraction ZIP model

```
zip2<-function(data,p) {
field.dummy<-as.numeric(slugs$field)-1
mylambda<-p[1]+p[3]*field.dummy
theta<-p[2]
zero.term<-sum(ifelse(data$slugs==0,log(theta+(1-theta)*
  dpois(data$slugs,lambda=mylambda)),0))
nonzero.term<-sum(ifelse(data$slugs>0,log((1-theta)*
  dpois(data$slugs,lambda=mylambda)),0))
negloglike<- -(zero.term+nonzero.term)
negloglike
}
nlm(function(p) zip2(slugs,p),c(3.4,.42,-.4))->out8
my.aic(out8)
[1] 306.8417
```

5. Separate zero fraction and common mean ZIP model

```
zip3<-function(data,p)
{
field.dummy<-as.numeric(data$field)-1
mylambda<-p[1]
theta<-p[2]+p[3]*field.dummy
zero.term<-sum(ifelse(data$slugs==0,log(theta+(1-theta)*
   dpois(data$slugs,lambda=mylambda)),0))
nonzero.term<-sum(ifelse(data$slugs>0,log((1-theta)*
   dpois(data$slugs,lambda=mylambda)),0))
negloglike<- -(zero.term+nonzero.term)
negloglike
}
nlm(function(p) zip3(slugs,p),c(3,.6,-.4))->out9
my.aic(out9)
[1] 293.4236
```

6. Common mean and variance log-transformed normal model

```
norm.neglike<-function(data,p) {
t.y<-log(data$slugs+1)
mu<-p[1]
my.sd<-p[2]
negloglike<- -sum(log(dnorm(t.y,mean=mu,sd=my.sd)))
negloglike
}
nlm(function(p) norm.neglike(slugs,p),c(.73,.74))->out.norm

norm.like<-function(data,out)
{
t.y<-log(data$slugs+1)
mu<-out$estimate[1]
my.sd<-out$estimate[2]
negloglike<- -sum(log(dnorm(t.y,mean=mu,
  sd=my.sd)*1/(data$slugs+1)))
out<-list(negloglike,out$estimate)
names(out)<-c("minimum","estimate")
```

```
out
}

norm.like(slugs,out.norm)->out20
my.aic(out20)
[1] 299.4730
```

7. Separate mean and common variance log-transformed normal model

```
norm.neglike2<-function(data,p)
{
t.y<-log(data$slugs+1)
field.dummy<-as.numeric(data$field)-1
mu<-p[1]+field.dummy*p[3]
my.sd<-p[2]
negloglike<- -sum(log(dnorm(t.y,mean=mu,sd=my.sd)))
negloglike
}
nlm(function(p) norm.neglike2(slugs,p),c(.5,.7,.5))->outnorm2

norm.like2<-function(data,out)
{
t.y<-log(data$slugs+1)
field.dummy<-as.numeric(data$field)-1
mu<-out$estimate[1]+field.dummy*out$estimate[3]
my.sd<-out$estimate[2]
negloglike<- -sum(log(dnorm(t.y,mean=mu,
  sd=my.sd)*1/(data$slugs+1)))
out<-list(negloglike,out$estimate)
names(out)<-c("minimum","estimate")
out
}

norm.like2(slugs,outnorm2)->out21
my.aic(out21)
[1] 292.7727
```