**Ecol 145 Assignment 3**
**Dahl Winters**
**2/10/06**

## Questions

`(Additional R code and plots can be found at the end of the homework.)`

**(1)** Using the formula for geometric probability model given above, write down the likelihood for the bird utterance data that appear in the table above. Thus your answer should involve the actual data values shown in the table above. You do not need to simplify your expression in any way.

For n=1:8,

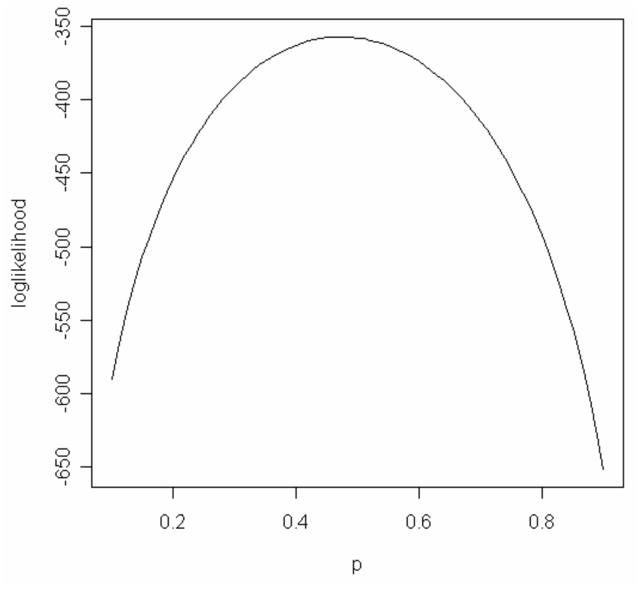
$P(X{=}n) = [p(1-p)^0]^{132} + [p(1-p)^1]^{52} + [p(1-p)^2]^{34} + [p(1-p)^3]^9 + [p(1-p)^4]^7 + [p(1-p)^5]^5 + [p(1-p)^6]^5 + [1 - \Sigma(i{=}1{:}7)[p(1-p)^{i-1}]^6\ ]$,

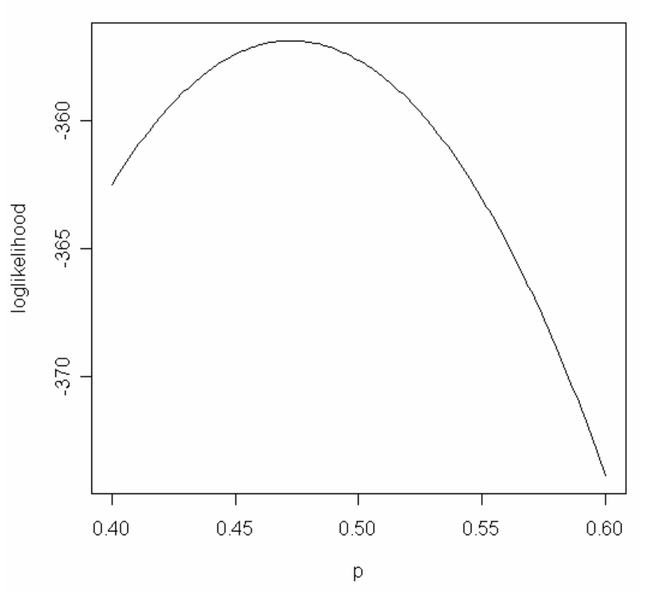
where n = # utterances/bout observed, and X is our random variable.

**(2)** Write a function in R to calculate the loglikelihood for the data. You may use either the geometric probability functions or the negative binomial probability functions. Your function should require a single argument that corresponds to the value *p*.

```
> loglik<-function(p) log(dgeom(0,p)^132) + log(dgeom(1,p)^52) +
log(dgeom(2,p)^34) + log(dgeom(3,p)^9) + log(dgeom(4,p)^7) +
log(dgeom(5,p)^5) + log(dgeom(6,p)^5) + log((1-pgeom(6,p))^6)
```

**(3)** Graph the loglikelihood function you obtained in (2) and use the graph to approximate the maximum likelihood estimate of *p*.





The value of p where the maximum occurs is the MLE, and it is approximately 0.48 judging from the 2 graphs above. The first is a wider view, and the second is a closeup view. There, the max occurs at about p=0.48 with a loglikelihood of around -355.

**(4)** Use one of R's numerical optimization functions to approximate the maximum likelihood estimate of *p*.

The optimize function gives a numerical estimate of the MLE as 0.47, with a value of the loglikelihood of 356.9. This agrees with what was obtained from the nlm and optim functions.

**(5)** Construct both the 95% Wald confidence interval and the 95% profile likelihood confidence interval for *p*.

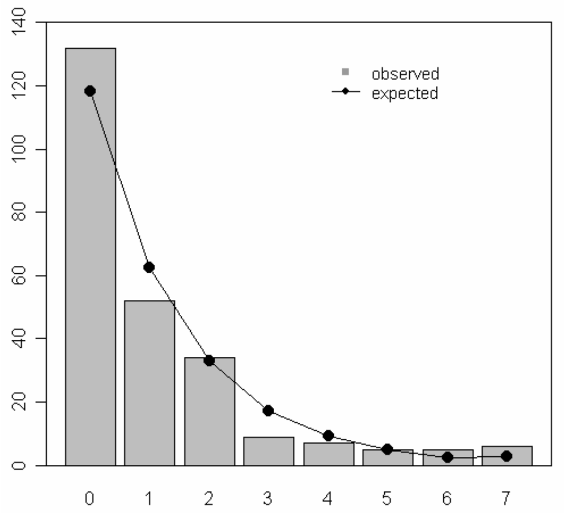
The 95% Wald confidence interval to 2 decimal places is from 0.43 to 0.52. This means we can be 95% confident that the actual value of p is within this interval.

The 95% profile likelihood confidence interval to two decimal places is from 0.43 to 0.52, which coincides with the 95% Wald confidence interval. Thus, we must have a large enough dataset, or else the two methods of calculating confidence intervals might have yielded different results.

**(6)** Calculate the expected frequencies of the #utterances/bout under the estimated geometric probability model using the maximum likelihood estimate of *p* you obtained in (4). Plot the observed and expected frequencies together in the same plot.

Expected frequencies of each #utterances/bout:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 118.217054 | 62.315967 | 32.848727 | 17.315608 | 9.127607 | 4.811452 | 2.536269 | 2.827316 |



The expected and observed frequencies both follow a downward curve with increasing values of #utterances/bout, but in several cases the observed frequencies fall short of the expected frequencies.

**(7)** Carry out a Pearson chi-squared lack-of-test for the model results. Do this two ways.

1. *By pooling cells that don't meet the minimal size criteria (if necessary) and then using the asymptotic distribution of the test statistic to assess fit.*

   The last 3 categories were pooled together (see code below), and then the test was run to give a p-value of 0.0246. This isn't as small as the p-value we obtained for our Poisson model of the aphid data, so at least we can say the geometric model is a better fit to our bird data than the Poisson was to the aphid data. The critical value for our Pearson test was 9.49 and the observed value of the test statistic was 11.18, which is more than twice the degrees of freedom, which was 4. Thus, the p-value is significant.

2. *Without pooling (except for the expected probabilities in the tail of the distribution) and then using the simulate.p.value option of the chisq.test function of R to obtain a Monte Carlo-based p-value.*

   The p-value is 0.0575, using 20000 simulated values. Since this gives us accuracy to 4 decimal places and the p-value provided is to 4 decimal places, we know the p-value must be less than 0.0575. This coincides with the result from the Pearson test with pooling, which gave a p-value of 0.0246, which is less than 0.0575 and not that far from that value. Thus, pooling didn't make much of a difference to our test of how well the geometric model fits the observed data.

Does the model fit?

According to the Pearson test, we must reject the null hypothesis that the geometric model's fit is accurate because the observed value of the test statistic, 11.18, exceeds the critical value, 9.49.

Also, the Monte Carlo method gave us a p-value of 0.0575, which is equal to 1150/20000, meaning there were 1150 simulated values of the test statistic as extreme as the observed values. An excellent fit would have close to 20000 simulated values that matched the observed values, while our model only gives 1150. So this seems to be another way of judging how far off the model is.

# R Code

## Problem 1

```
#   Generating the dataset bird.data from the bird utterance data given
> frequency<-c(132,52,34,9,7,5,5,6)
> sum(frequency)
[1] 250
> bird.data<-rep(1:8,frequency)
> bird.data
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [35] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [69] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[103] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
[137] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[171] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[205] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6
[239] 6 7 7 7 7 7 8 8 8 8 8 8
```

## Problem 2

```
#   Coming up with the loglikelihood function

> loglik<-function(p) log(dgeom(0,p)^132) + log(dgeom(1,p)^52) +
log(dgeom(2,p)^34) + log(dgeom(3,p)^9) + log(dgeom(4,p)^7) +
log(dgeom(5,p)^5) + log(dgeom(6,p)^5) + log((1-pgeom(6,p))^6)
```

## Problem 3

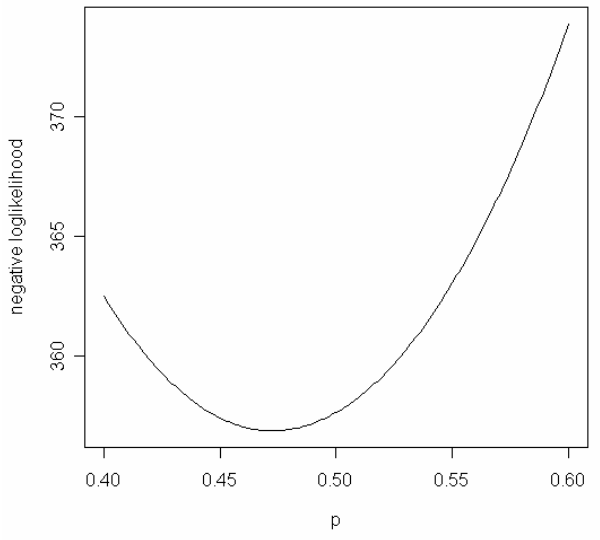
```
#   Graphing the loglikelihood to estimate the value of the MLE for p.
> plot(seq(.1,.9,.01), sapply(seq(.1,.9,.01), loglik), type='l',
xlab=expression(p), ylab='loglikelihood')

#   Zooming in on the maximum
> plot(seq(.4,.6,.001), sapply(seq(.4,.6,.001), loglik), type='l',
xlab=expression(p), ylab='loglikelihood')
```

## Problem 4

```
#   Getting a function for the
negative loglikelihood so R can
perform minimizations

> negloglik=function(p) -loglik(p)
> negloglik(.2) #to check if my
function worked
[1] 453.3979
> plot(seq(.4,.6,.001),
sapply(seq(.4,.6,.001),negloglik),
type='l', xlab=expression(p),
ylab='negative loglikelihood') #to see
if there is a minimum as I expected
there to be, and there is.
```



```
> nlm(negloglik,.48)
$minimum
[1] 356.9039

$estimate
[1] 0.4728682

$gradient
[1] -1.875833e-08

$code
[1] 1

$iterations
[1] 3

There were 18 warnings (use warnings() to see them)

> optim(0.48,negloglik)
$par
[1] 0.472875

$value
[1] 356.9039

$counts
function gradient
      22       NA

$convergence
[1] 0

$message
NULL

Warning message:
one-diml optimization by Nelder-Mead is unreliable: use optimize in:
optim(0.48, negloglik)

#Because of the warning message, I decided to use the optimize function
instead.

> optimize(negloglik,c(.4,.5))
$minimum
[1] 0.4728819

$objective
[1] 356.9039
```

## Problem 5

```
#   Calculating the Wald confidence interval

> out<-nlm(negloglik,.48,hessian=TRUE)
There were 18 warnings (use warnings() to see them)
```

```
> out
$minimum
[1] 356.9039

$estimate
[1] 0.4728682

$gradient
[1] -1.875833e-08

$hessian
         [,1]
[1,] 2070.006

$code
[1] 1

$iterations
[1] 3

> out$estimate-qnorm(.975)*sqrt(1/out$hessian) #lower bound
          [,1]
[1,] 0.4297895
> out$estimate+qnorm(.975)*sqrt(1/out$hessian) #upper bound
          [,1]
[1,] 0.5159469

#   Calculating the profile likelihood confidence interval
> library(Bhat)
> x.in<-list(label='p',est=.47, low=.43, up=.52)
> plkhci(x.in,negloglik,'p')
[1] 0.4300000 0.5160958
```

## Problem 6

```
> phat<-out$estimate
> exp<-c( dgeom(0:6,phat),1-pgeom(6,phat) )*250
> exp
# These are the expected frequencies of #utterances/bout expected under the
estimated geometric probability model using the MLE of p, which is 0.47.

[1] 118.217054  62.315967  32.848727  17.315608   9.127607   4.811452
2.536269  [8]   2.827316

> max(exp) #the maximum expected value
[1] 118.2171
> max(table(bird.data)) #the maximum observed value, which is greater than
the maximum expected value – the y limit of the graph should be above this.
[1] 132

> coords<-barplot(table(bird.data), ylim=c(0,140))
> points(coords,exp,pch=16,cex=1.5)
> lines(coords,exp)
> legend(coords[5],130, c('observed','expected'), pch=c(15,16),
col=c('gray60',1), lty=c(NA,1), cex=c(.9,.9), bty='n')
> box()
```

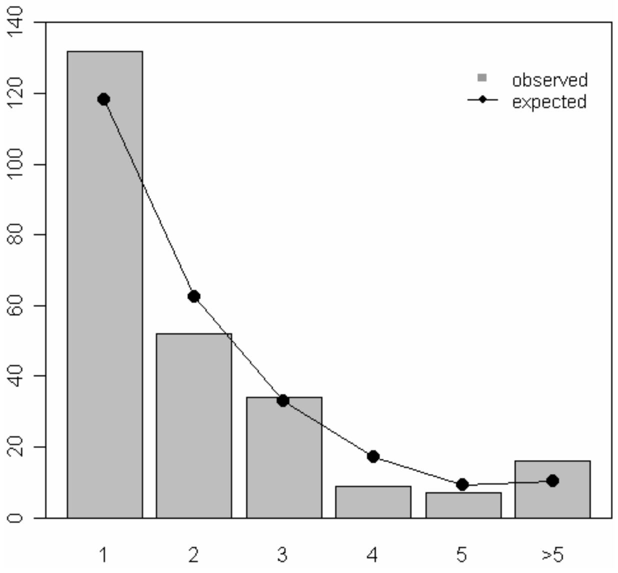
## Problem 7

```
#   Doing the Pearson chi-squared test with pooling

> exp<-c( dgeom(0:6,phat),1-pgeom(6,phat) )*250
> exp
[1] 118.217054  62.315967  32.848727  17.315608   9.127607   4.811452
2.536269   2.827316

#   The last 3 frequencies are less than our cut-off of 5.  These can all be
pooled together.

> expected<-c( dgeom(0:4,phat),1-pgeom(4,phat) )*250
> expected
[1] 118.217054  62.315967  32.848727  17.315608   9.127607  10.175037
> observed<-c(rawcounts[0:5], sum(rawcounts[6:length(rawcounts)]))
> observed
  0   1   2   3   4
132  52  34   9   7  16
> names(observed)<-c(1:5,'>5') #naming the categories correctly
> observed
  1   2   3   4   5  >5
132  52  34   9   7  16
> coords<-barplot(observed, ylim=c(0,140))
> points(coords,expected,pch=16,cex=1.5)
> lines(coords,expected)
> legend(coords[5],130, c('observed','expected'), pch=c(15,16),
col=c('gray60',1), lty=c(NA,1), cex=c(.9,.9), bty='n')
> box()
```



```
> pearson<-sum((observed-
expected)^2/expected)
> pearson
[1] 11.17910
> df<-length(observed)-1-1
> df
[1] 4
> p.val<-1-pchisq(pearson,df)
> p.val
[1] 0.02462328
> qchisq(.95,df)
[1] 9.487729

#   Doing the Pearson chi-squared test
without pooling

> expprob<-c(dgeom(0:6,phat),1-
pgeom(6,phat))
> sum(expprob)
[1] 1
> chisq.test(rawcounts,p=expprob,simulate.p.value=TRUE,B=20000)
        Chi-squared test for given probabilities with simulated p-value
(based on 20000 replicates)
data:  rawcounts
X-squared = 13.8053, df = NA, p-value = 0.0575
```