hw3

jt-miller

2023-11-20

HW #3 Statistical Ecology

1. Suppose we wish to compare the density of a particular species of tree among 3 different localities. In each locality i (i = 1,2,3), we sample n_i quadrats of size 1 hector (ha), and count the number of that species of tree in it. Suppose n_1 = 6, n_2 = 8, n_3 = 7. The number of trees per quadrat in each locality are:

```
samp.1 <- c(2,7,3,2,4,6) # These are the n_1 = 6 counts for the 1st locality
samp.2 <- c(1,1,1,2,2,0,1,1) # These are the n_2 = 8 counts for the 2nd locality
samp.3 <- c(2,4,3,0,2,4,1) # These are the n_3 = 7 counts for the 3rd locality
```

Let mew_i be the true, unknown, mean number of trees per hectare in location i (i = 1,2,3). Several possible scenarious regarding the compariosn of these quantites across localities come to mind. The simplest scenario and our null hypothesis being: mew_1 = mew_2 = mew_3 = mew. We'll call this Model 0 One alternative scenario would be mew_1 =/= mew_2 =/= mew_3. We'll call this Model 1

a. Write down three other plausible alternative scenarios:

$$(\mu_1 = \mu_2) \neq \mu_3$$

$$(\mu_1 = \mu_3) \neq \mu_2$$

$$(\mu_2 = \mu_3) \neq \mu_1$$

b. For each model, including Models 0 and 1 above, propose a discrete probability model of how the data arose. After doing so, write down the likelihood function and solve for the Maximum Likelihood Estimates (MLEs) of the model parameters. Here I was both a mathematical expression and its numerical evaluation using the data given above. Use the following notation: for Model 0, denote the likelihood as

$$l(\mu)$$

for Model 1, denote it as

$$l(\mu_1, \mu_2, \mu_3)$$

See attached pdf for the further explained models: # Put pdf here

Numerical Eval in 1b

```
# For Model 0
model0_mu_hat = mean(c(samp.1, samp.2, samp.3))
# For Model 1
model1_mu_hat_1 = mean(samp.1)
model1_mu_hat_2 = mean(samp.2)
```

```r
model1_mu_hat_3 = mean(samp.3)
# For Model 2
model2_mu_hat_12 = mean(c(samp.1,samp.2))
model2_mu_hat_3 = mean(samp.3)
# For Model 3
model3_mu_hat_13 = mean(c(samp.1, samp.3))
model3_mu_hat_2 = mean(samp.2)
# For Model 4
model4_mu_hat_23 = mean(c(samp.2,samp.3))
model4_mu_hat_1 = mean(samp.1)
```

1c: For each model, evaluate the likelihood functon at the MLEs and the data at hand. That quantity is the maximized likelihood function

```r
# M0 Likelihood function
lnL0 <- function(counts1, counts2, counts3){

  all.counts <- c(counts1, counts2, counts3) # Null hyp says that all mus are equal so we can pool
  lambda.hat <- mean(all.counts)
  llikevec <- dpois(x = all.counts, lambda = lambda.hat, log = TRUE) # Log-likelihood maximized under t
  return(sum(llikevec)) # And summed up
}
# M1 Likelihood function
lnL1 <- function(counts1, counts2, counts3){
  lambda.hat1 <- mean(counts1)
  lambda.hat2 <- mean(counts2)
  lambda.hat3 <- mean(counts3)

  llike1 <- dpois(x = counts1, lambda = lambda.hat1, log = TRUE)
  llike2 <- dpois(x = counts2, lambda = lambda.hat2, log = TRUE)
  llike3 <- dpois(x = counts3, lambda = lambda.hat3, log = TRUE)

  return(sum(c(llike1, llike2, llike3)))
}
# M2 Likelihood function
lnL2 <- function(counts1, counts2, counts3){
  site12.counts <- c(counts1, counts2)
  lambda.hat12 <- mean(site12.counts)
  lambda.hat3 <- mean(counts3)

  llike12 <- dpois(x = site12.counts, lambda = lambda.hat12, log = TRUE)
  llike3 <- dpois(x = counts3, lambda = lambda.hat3, log = TRUE)

  return(sum(c(llike12, llike3)))
}
# M3 Likelihood function
lnL3 <- function(counts1, counts2, counts3){
  site13.counts <- c(counts1, counts3)
  lambda.hat13 <- mean(site13.counts)
  lambda.hat2 <- mean(counts2)

  llike13 <- dpois(x = site13.counts, lambda = lambda.hat13, log = TRUE)
  llike2 <- dpois(x = counts2, lambda = lambda.hat2, log = TRUE)
```

```
  return(sum(c(llike13, llike2)))
}
# M4 Likelihood function
lnL4 <- function(counts1, counts2, counts3){
  site23.counts <- c(counts2, counts3)
  lambda.hat23 <- mean(site23.counts)
  lambda.hat1 <- mean(counts1)

  llike23 <- dpois(x = site23.counts, lambda = lambda.hat23, log = TRUE)
  llike1 <- dpois(x = counts1, lambda = lambda.hat1, log = TRUE)

  return(sum(c(llike23, llike1)))
}
```

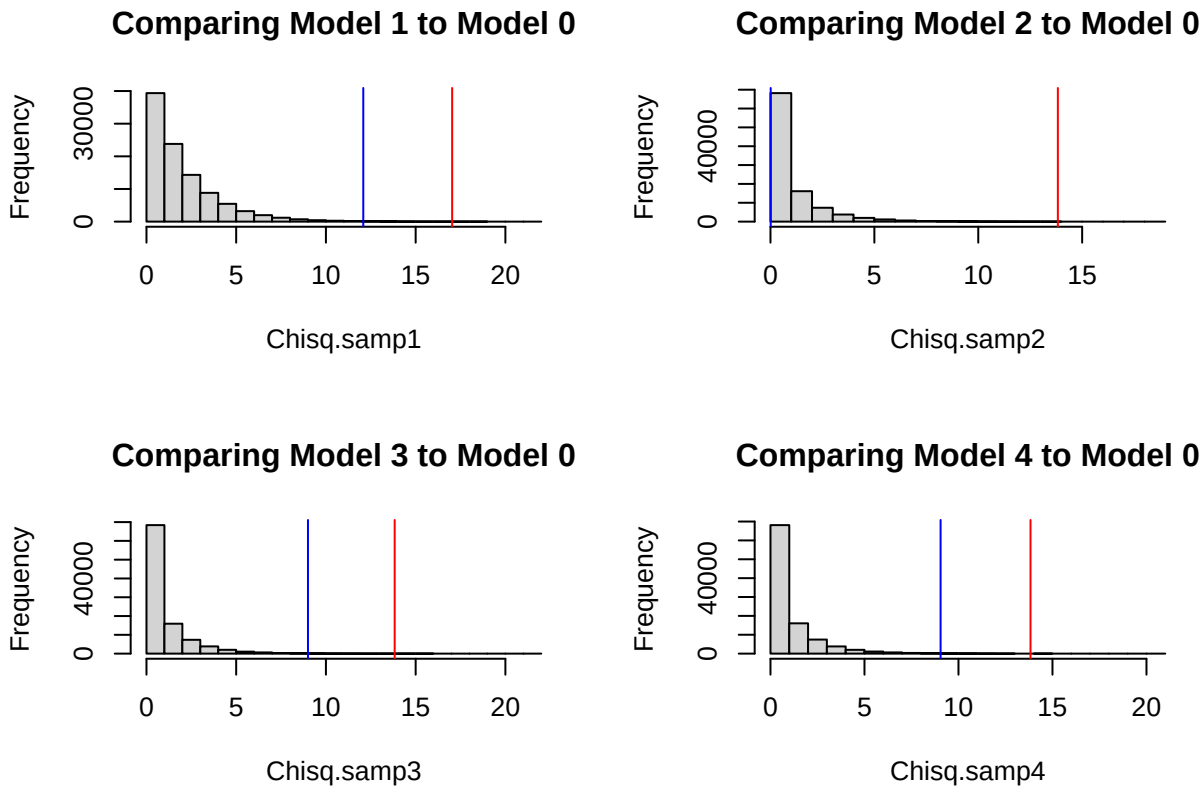
Maximized Likelihood

```
lnL0.hat <- lnL0(counts1 = samp.1, counts2=samp.2, counts3=samp.3)
lnL1.hat <- lnL1(counts1 = samp.1, counts2=samp.2, counts3=samp.3)
lnL2.hat <- lnL2(counts1 = samp.1, counts2=samp.2, counts3=samp.3)
lnL3.hat <- lnL3(counts1 = samp.1, counts2=samp.2, counts3=samp.3)
lnL4.hat <- lnL4(counts1 = samp.1, counts2=samp.2, counts3=samp.3)
```

d. Likelihood Ratio Test using Wilk's Chi-squared approximation to test the null hypothesis against each of the alternative models. With multiple hypothesis tests we'll need a Bonferroni type 1 error correction. See class notes for more details: If the number of tests we are doing is m and you want to do an overall tet with Type 1 error rate of alpha, then the Bonferroni corrected alpha_beta to be used will be equal to alpha/m

```
# Chisq observed
Gsq.obs1 <- -2*(lnL0.hat - lnL1.hat)
Gsq.obs2 <- -2*(lnL0.hat - lnL2.hat)
Gsq.obs3 <- -2*(lnL0.hat - lnL3.hat)
Gsq.obs4 <- -2*(lnL0.hat - lnL4.hat)
# Set up comparisons
alpha <- 0.001 # The alpha value (Arbitrary?)
m <- 5 # The number of tests we're doing
alpha_beta <- alpha/m # Bonferroni corrected for Type 1 error
# Degrees of Freedom are based upon the number of parameters estimated under one model - the null model
dfs.comp1 <- 3 - 1 # 3 params in model1, 1 param in model0
dfs.comp2 <- 2 - 1 # 2 params in model2, 1 param in model0
dfs.comp3 <- 2 - 1 # 2 params in model3, 1 param in model0
dfs.comp4 <- 2 - 1 # 2 params in model4, 1 param in model0
# Set up critical values
Gsq.crit1 <- qchisq(p = 1-alpha_beta, df = dfs.comp1)
Gsq.crit2 <- qchisq(p = 1-alpha_beta, df = dfs.comp2)
Gsq.crit3 <- qchisq(p = 1-alpha_beta, df = dfs.comp3)
Gsq.crit4 <- qchisq(p = 1-alpha_beta, df = dfs.comp4)
# Set up pvalues
pvalue1 <- 1 - pchisq(q = Gsq.obs1, df = dfs.comp1)
pvalue2 <- 1 - pchisq(q = Gsq.obs2, df = dfs.comp2)
pvalue3 <- 1 - pchisq(q = Gsq.obs3, df = dfs.comp3)
pvalue4 <- 1 - pchisq(q = Gsq.obs4, df = dfs.comp4)
```

```r
# Simulate
Chisq.samp1 <- rchisq(n=100000, df=dfs.comp1)
Chisq.samp2 <- rchisq(n=100000, df=dfs.comp2)
Chisq.samp3 <- rchisq(n=100000, df=dfs.comp3)
Chisq.samp4 <- rchisq(n=100000, df=dfs.comp4)

par(mfrow = c(2,2))
hist(Chisq.samp1, main = "Comparing Model 1 to Model 0")
abline(v=Gsq.obs1, col="blue")
abline(v=Gsq.crit1, col="red")

hist(Chisq.samp2, main = "Comparing Model 2 to Model 0")
abline(v=Gsq.obs2, col="blue")
abline(v=Gsq.crit2, col="red")

hist(Chisq.samp3, main = "Comparing Model 3 to Model 0")
abline(v=Gsq.obs3, col="blue")
abline(v=Gsq.crit3, col="red")

hist(Chisq.samp4, main = "Comparing Model 4 to Model 0")
abline(v=Gsq.obs4, col="blue")
abline(v=Gsq.crit4, col="red")
```



It appears in all 4 models that we should not reject the null, as the observed value is not greater than the pcrit.

1e. Parametric Bootstrap Likelihood Ratio Tests (PBLRT): Often and especially with samll sample sizes the

Wilk's chi-square approximation for the distribution of the likelihood ratio (if the null is true) does not hold very well. A way to get around this is to approximate the distribution of the likelihood ratio computationally. If we ewant to test a null model H0 vs an alternative model of H1, the PBLRT is accomplihsed by following the steps listed below. Compare the Model 0 vs Model 1 between the Wilk's chi-sq approx and the PBLRT

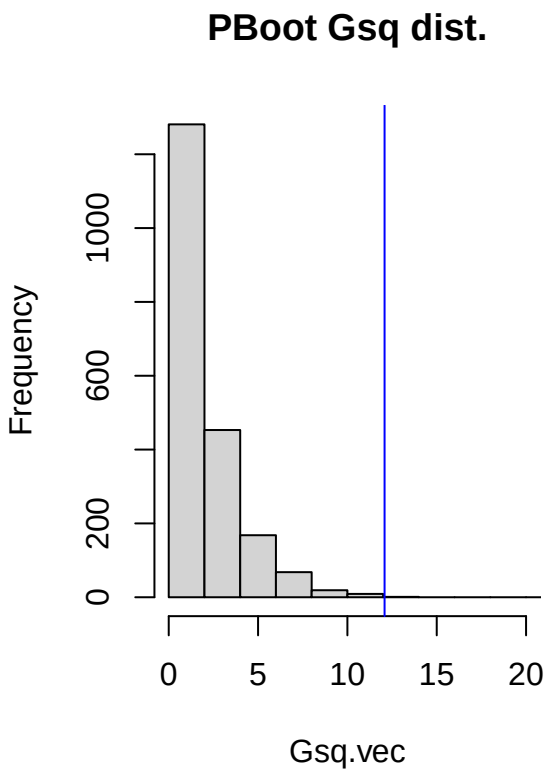
```
# Step 1: Compute the maximum likelihood model parameter(s) under the null for the given data set. Labe
## For M0 there is only one parameter to estimate
all.samps <- c(samp.1, samp.2, samp.3) # Pool all samples for the null model
theta_hat_0 <-  mean(all.samps) # The MLE under the M0 (H0)
## For M1 there are 3 parameters to estimate
theta_hat_1.1 <- mean(samp.1)
theta_hat_1.2 <- mean(samp.2)
theta_hat_1.3 <- mean(samp.3)

# Step 2: Plug theta_hat_0 and theta_hat_1 into their respective likelihood functions and compute the o

# Step 3: Simulate with bootstrap
B <- 2000
Gsq.vec <- rep(0,B) # Empty gsq vector
n1 <- length(samp.1)
n2 <- length(samp.2)
n3 <- length(samp.3)

for(i in 1:B){
    # Simulate data like the one observed but under the Null hypothesis

    boot.data1 <- rpois(n=n1, lambda = theta_hat_0) # dpois under the null, pretend that the Null Hypot
    boot.data2 <- rpois(n=n2, lambda = theta_hat_0) # "" for second experimental condition
    boot.data3 <- rpois(n=n3, lambda = theta_hat_0) # "" for third experimental condition

    lnL0.boot <- lnL0(counts1 = boot.data1, counts2=boot.data2, counts3=boot.data3) # Compute the likel
    lnL1.boot <- lnL1(counts1 = boot.data1, counts2=boot.data2, counts3=boot.data3)

    Gsq.vec[i] <- -2*(lnL0.boot-lnL1.boot) # Compute the gsq per particular case

}
boot.pval <- sum(Gsq.vec > Gsq.obs1)/2000 # Proportion of boostrap Gsq's that are bigger than the obser
print(boot.pval) # compare to Chisquare pvalue
```

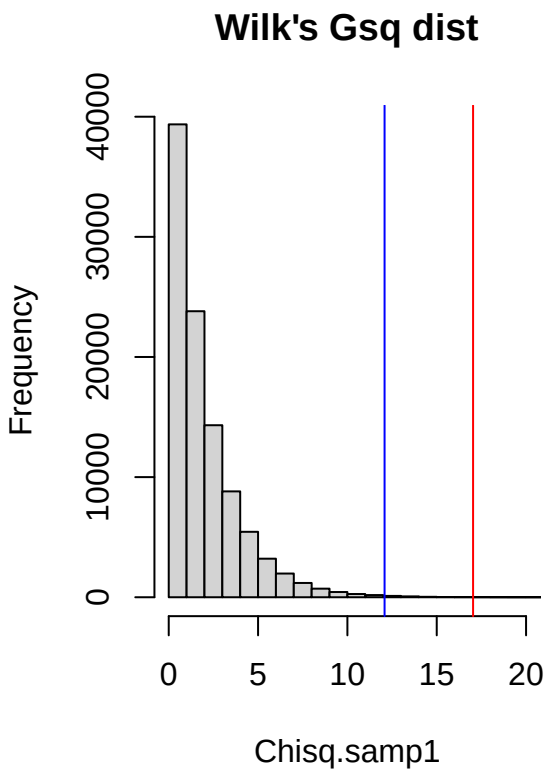
```
## [1] 5e-04
```

```
print(pvalue1)
```

```
## [1] 0.002380668
```

```
par(mfrow=c(1,2))
hist(Gsq.vec, main = "PBoot Gsq dist.", xlim= c(0,20))
abline(v = Gsq.obs1, col = "blue")
hist(Chisq.samp1, main = "Wilk's Gsq dist", xlim=c(0,20))
abline(v=Gsq.obs1, col="blue")
abline(v=Gsq.crit1, col="red")
```





Its about the same, therefore our conclusions shouldnt change(?).

2. See poster for distribution mapping.

3. Suppose a biologist goes out to a lodgepine forest and counts the number of pines present in each of 100 quadrats of size a and then counts the number of quadrats with 0,1,2,3,4,5,6 and >= 7 trees. The obtained counts are summarized in the first two columns of table 1 (see sheet).

a. Calc the ML estimate of lambda_a for the lodgepole pines example. Once you do that, calculate the expected frequencies of the number of pines in a sample size of q (fill in the third column of table 1) and graph the observed vs expected frequencies.

```
### This bit is Jose-Miguels code found in LodgepolePinesGsq.R
# Count freq
yy <- c(7,16,20,24,17,9,5,2); # meaning that 7 quadrats contained 0 trees, 16 quadrats contained 1 tree

#  Initial parameter value is calculated from the approximate sample mean.
kk=length(yy);
nn=sum(yy);
xx=0:(kk-1);
lambda0=sum(xx*yy)/nn;

# ML objective function "negloglike.ml" is negative of log-likelihood; the optimization routine in R, "

negloglike.ml=function(theta,ys)
{
```

```
   lambda=exp(theta);      #  Constrains 0 < lambda, otherwise NaNs are produced. # Mean number of trees
   k=length(ys);
   x=0:(k-1);
   x1=x[1:(k-1)];
   p=rep(0,k);
   p[1:(k-1)]=exp(-lambda+x1*log(lambda)-lfactorial(x1)); # The log(Poisson Probability) same as dpois(
   p[k]=1-sum(p[1:(k-1)]);
   ofn=-sum(ys*log(p));      #  No need to calculate all the factorials.
   return(ofn);
}

# The ML estimate.
MULTML=optim(par=log(lambda0),
   negloglike.ml,NULL,method="BFGS",ys=yy);  #  Nelder-Mead algorithm is not
                                             #  reliable for 1-D problems.
reslts=c(exp(MULTML$par[1]),-MULTML$val); # Take the theta, and then transform via exponentiation to a
lambda.ml=reslts[1];            # This is the ML estimate.
nn=sum(yy);
loglike.ml=reslts[2]+lfactorial(nn)-sum(lfactorial(yy)); #  Log-likelihood.

# Calculate expected values, LR statistic, etc.
xx1=xx[1:(kk-1)];
pp=rep(0,kk);
pp[1:(kk-1)]=exp(-lambda.ml+xx1*log(lambda.ml)-lfactorial(xx1));
pp[kk]=1-sum(pp[1:(kk-1)]);
EE=nn*pp;
y1=yy;
y1[y1==0]=1;                # Guard against log(0) in G-squared.
Gsq=2*sum(yy*log(y1/EE));  # G-squared goodness of fit statistic.
pvalG=1-pchisq(Gsq,8-1-1);     # p-value (chisquare distribution) for G-squared.
Xsq=sum((yy-EE)^2/EE);      # Pearson goodness of fit statistic.
pvalX=1-pchisq(Xsq,8-1-1);     # p-value (chisquare distribution) for Pearson

#  Print the results.
lambda.ml;
```

```
## [1] 2.859631
```

```
loglike.ml;
```

```
## [1] -13.97714
```

```
Gsq;
```

```
## [1] 1.276895
```

```
pvalG;
```

```
## [1] 0.9729164
```

```
Xsq;
```

```
## [1] 1.260605
```

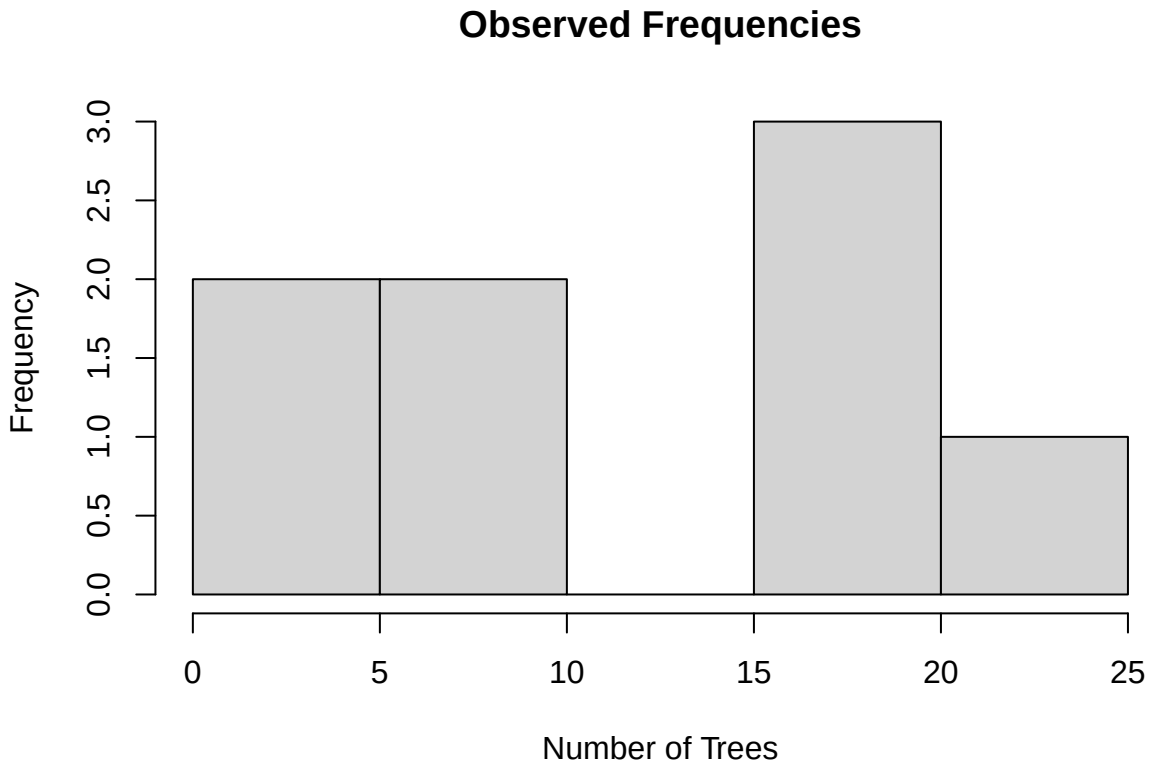
```
pvalX;
```

```
## [1] 0.9737855
```

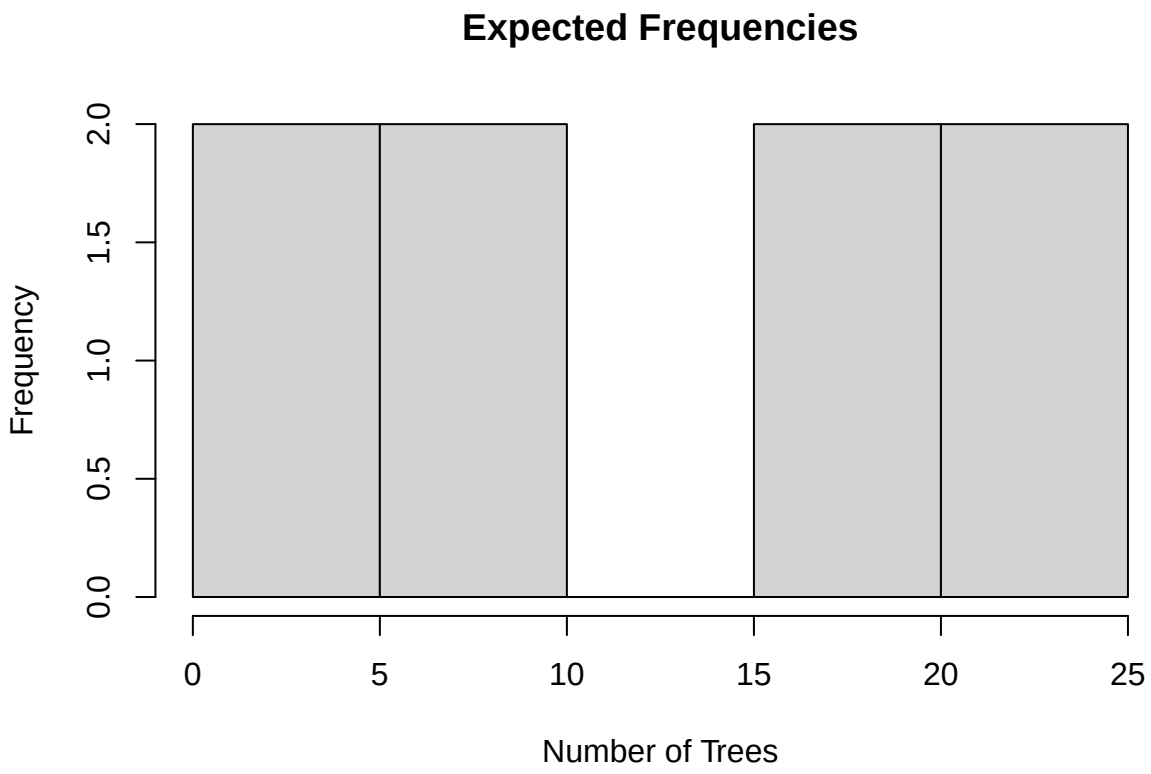
```
cbind(EE,yy);
```

```
##             EE yy
## [1,]  5.728991  7
## [2,] 16.382799 16
## [3,] 23.424378 20
## [4,] 22.328357 24
## [5,] 15.962714 17
## [6,]  9.129494  9
## [7,]  4.351164  5
## [8,]  2.692104  2
```

```
### Graph the observed vs expected frequencies
hist(yy, main = "Observed Frequencies", xlab = "Number of Trees", ylab = "Frequency")
```



```r
hist(EE, main = "Expected Frequencies",xlab = "Number of Trees", ylab = "Frequency")
```



The Pval of the Pearson goodness of fit statistic indicates the poisson does pretty well (97.3% explained)

Repeat for the negative binomial.

```r
negloglike.ml_negbinom=function(theta,ys)
{
   lambda=exp(theta);      #  Constrains 0 < lambda, otherwise NaNs are produced. # Mean number of trees
   size = exp(theta[2])
   k=length(ys);
   x=0:(k-1);
   x1=x[1:(k-1)];
   p= dnbinom(x1, size = size, mu = lambda, log = TRUE)
   ofn=-sum(ys*p);      #  No need to calculate all the factorials.
   return(ofn);
}

# The ML estimate.
#MULTML=optim(par=log(lambda0),
   #negloglike.ml_negbinom,NULL,method="BFGS",ys=yy);  #  Nelder-Mead algorithm is not
                                                       #  reliable for 1-D problems.
#reslts=c(exp(MULTML$par[1]),-MULTML$val); # Take the theta, and then transform via exponentiation to a
#lambda.ml=reslts[1];             # This is the ML estimate.
#nn=sum(yy);
#loglike.ml=reslts[2]+lfactorial(nn)-sum(lfactorial(yy)); #  Log-likelihood.
```

4. See attached pdf # Put pdf here

5. TRUE/FALSE

a. The significance level of a statistical test is equal to the probability that the null hypothesis is true. FALSE, the significance level of a statitiscal test is equal to alpha, while the probability of not regecting the null hypothesis is 1 - alpha.

b. The likelihood ratio test is a random variable TRUE,

c. A type 2 error is more serious than a type 1 error FALSE, type 2 error is typically more safe since it does not conclude that we should reject the null, however; its also dependent on how we frame the test for whether one has 'serious' implications.

d. A type 1 error occurs when the test statistic falls into the rejection region of the test. TRUE

e. If a test rejects at the significance of 0.06, then the pvalue is less than or equal to 0.06 TRUE

f. The pvalue of a test is the probability that the Null hypothesis is correct FALSE, the pvalue is the probability given the null hyp is true of observing a test statistic as extreme as the sample taken.