Homework 5

Biology 697

10/5/2012

1 χ^2 - the function!

1.1

Write a simple function to calculate a chi-squre test based on the assumption that you should have the same observed values across all categories. Have it return the value of χ^2, the degrees of freedom, and the p-value. To test it, compare your function's output for the observed vector 1:10 versus R's chisq.test function.

```
chiFun<-function(obs){
  exp <- rep(sum(obs)/length(obs), length(obs))
  chisq <- sum((obs-exp)^2/exp)
  p <- pchisq(chisq, df=length(obs)-1, lower.tail=FALSE)

  return(list(Chisq = chisq, DF = length(obs)-1, p=p))

}

chiFun(1:10)

## $Chisq
## [1] 15
##
## $DF
## [1] 9
##
## $p
## [1] 0.09094

chisq.test(1:10)

##
##  Chi-squared test for given probabilities
##
```

```
## data:  1:10
## X-squared = 15, df = 9, p-value = 0.09094
```

1.2

Use your function to answer question 13 on page 201 of W&S. Answer all parts of the question (some will not require R - sad, I know.)

```
windowVec <- c(30, 15, 8)
chiFun(windowVec)

## $Chisq
## [1] 14.3
##
## $DF
## [1] 2
##
## $p
## [1] 0.0007841
```

2 Contingency Tables

In W&S 9.3, we are introduces to the χ^2 contingency test which seeks to evaluate association between categorical variables - such as whether a fish is infected or uninfected with a parasite v. whether or not it is eaten by birds. Using the formulae in your book write a function that will return the χ^2, DF, and p-value for a contingency table. Use the following matrix as test data (from the book) as well as the output from R's own chisq.test. Show me that your function works.

```
#we specify a matrix by giving it a vector, then
#specifying how the rows and columns are setup
eaten <- matrix(c( 1, 10, 37,
                   49, 35, 9), ncol=3, byrow=T)

#now gives names to the rows and columns
colnames(eaten) <- c("Uninfected", "Infected", "Highly Infected")
rownames(eaten) <- c("Eaten", "Not Eaten")

eaten

##           Uninfected Infected Highly Infected
## Eaten              1       10              37
## Not Eaten         49       35               9
```

```
chisq.test(eaten)

##
##  Pearson's Chi-squared test
##
## data:  eaten
## X-squared = 69.76, df = 2, p-value = 7.124e-16
```

Some helpful functions for you - `nrow, ncol, rowSums, colSums`. Also, read the whole section before writing this. In writing the function, one thing that might help would be to write out what you're going to do, step by step, in comments. Each comment should be one step. Then fill in the function with code that impelemnts what you have written in each comment. Heck, ponder adopting this as your standard workflow. It will save you a lot of time.

```
chiTableTest <- function(aMatrix){
  #degrees of freedom = (r-1)(c-1)
  df <- (nrow(aMatrix)-1) * (ncol(aMatrix)-1)

  #get the row, col, and total sums for later use
  rs <- rowSums(aMatrix)
  cs <- colSums(aMatrix)
  total <- sum(aMatrix)

  #now iterate over the entire matrix and get the (O-E)^2/E value for each cell
  #using the shortcut for calculating expected frequencies
  chisq<-0
  for(i in 1:nrow(aMatrix)){
    for(j in 1:ncol(aMatrix)){
      obs <- aMatrix[i,j]
      exp <- rs[i] * cs[j]/total
      chisq <- chisq + (obs-exp)^2/exp
    }
  }

  return(list(chisq = chisq, df = df, p = pchisq(chisq, df, lower.tail=F)))

}

chiTableTest(eaten)

## $chisq
## Eaten
## 69.76
##
```

```
## $df
## [1] 2
##
## $p
##     Eaten
## 7.124e-16
```

2.1

Show me that your function works by answering question 21 on page 230 of W&S. Note, the default behavior of R's chisq.test on the same data set is to invoke Yates' continuity correction. So, careful when checking yourself.

```
heart <- matrix (c( 229, 1534,
                    822, 18296), ncol=2, byrow=T)

rownames(heart) <- c("Without Pain", "With Pain")
colnames(heart) <- c("Died", "Lived")

chiTableTest(heart)

## $chisq
## Without Pain
##          255
##
## $df
## [1] 1
##
## $p
## Without Pain
##    2.123e-57
```

3 Student's T and Power

Have a Guiness. There are some exact formulae to calculate the power of a t-test. Let's see how they compare to a simulation based approach. Let's compare the power via simulation for a sample with a sammple size of 10, a mean of 2, and a standard deviation of 3 to the one calculated by conventional means. In R, to do this conventionally, we'd use the following for a 2-tailed one sample T-test.

```
power.t.test(n=10, delta=2, sd=3,
             alternative="two.sided", type="one.sample")
```

```
##
##      One-sample t test power calculation
##
##               n = 10
##           delta = 2
##              sd = 3
##       sig.level = 0.05
##           power = 0.4692
##     alternative = two.sided
```

How about simulation? Let's put it together using functions.

3.1 T-Test!

To start with, let's write a function for the calculation of p-values for our data and evaluate it. Write a function to conduct a one sample t-test. Run it against sample of 10 values drawn from from a normal distribution with mean 2 and sd of 3. Check the results against R's `t.test` and show me that both agree.

```
se <- function(sample) sd(sample)/sqrt(length(sample))

tTest <- function(sample, nullMu=0){
  n<-length(sample)
  pt(abs((mean(sample)-nullMu)/se(sample)), df=n-1, lower.tail=F) * 2
}

samp<-rnorm(10,2,3)
tTest(samp)

## [1] 0.2928

t.test(samp)

##
##  One Sample t-test
##
## data:  samp
## t = 1.117, df = 9, p-value = 0.2928
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -1.672  4.934
## sample estimates:
## mean of x
##     1.631
```

3.2 In which we begin simulations

Now we need to get the power of our t-test via simulation. We'll accomplish this in two steps. First, write a function that will give you a vector of p values from t-tests on simulated data. This function should accept the number of simulations, the sample size, an critical effect size (the delta from above), and a sample standard deviation as it's arguments at the very least. Use it to create a vector of 5000 simulated p values using a sample size of 10, an effect size of 2, and a sd of 3, as above.

```
pvecGen <- function(n.sims=5000, n, ybar=0, sd=1, nullMu=0){
  pv <- rep(NA, n.sims)
  for(i in 1:n.sims){
    samp<-rnorm(n, mean = ybar, sd=sd)
    pv[i] <- tTest(samp)
  }

  pv
}

set.seed(081178)
pVector<-pvecGen(n=10, ybar=2, sd=3)
```

3.3 Simulated Power!

Now that you have a vector of simulated p-values, write a function that will take such a vector, and calculate it's power given an alpha. What is the power of your test, given an alpha of 0.05? How does it compare to the power from the exact calculation above?

```
power <-function (p, alpha=0.05) 1-sum(p > alpha)/length(p)

power(pVector)

## [1] 0.468
```