Stat 536 Homework 7

Due: 10/27/08

1. Suppose you collect the following data from 12 subpopulations.

Subpopulation	A_1A_1	A_1A_2	A_2A_2	$\hat{q}$
1	**208**	**484**	**311**	**0.44865**
2	**202**	**513**	**288**	**0.45713**
3	**705**	**305**	**29**	**0.82531**
4	**114**	**456**	**473**	**0.32790**
5	**772**	**207**	**10**	**0.88524**
6	**95**	**446**	**471**	**0.31423**
7	**791**	**252**	**28**	**0.85621**
8	**520**	**358**	**82**	**0.72813**
9	**279**	**489**	**229**	**0.52508**
10	**395**	**465**	**165**	**0.61220**
11	**686**	**249**	**23**	**0.84603**
12	**434**	**459**	**127**	**0.65049**
		$\hat{p} = 0.62232$		

where $\hat{q}$ and $\hat{p}$ are explained below. Use this data to estimate F_{IS}, F_{ST}, and F_{IT} for the population, following these steps.

(a) (10 pts) Assume that q across subpopulations are distributed as independent truncated-normal random variables as discussed in class. Specify the distributions f and g in the following formula for the likelihood of the table data $n = \{n_{ij} : i = 1, 2, \cdots, 12, j = 1, 2, 3\}$.

$$L(n \mid F_{IS}, p, c) = \int_q g(n; F_{IS}, q) f(q; p, c) dq \tag{1}$$

where g is the density of n and involves parameters F_{IS} and q and f is the density of q, involving parameters p and c.

Solution:

Assume F_{IS} is constant for all populations. $q = (q_1, \ldots, q_{12})$ is a vector of length 12. g is a product of multinomial distributions, namely

$$\begin{aligned} g(n; F_{IS}, q_1, \ldots, q_{12}) &= g_1(n_{1,\cdot}; F_{IS}, q_1) \times \cdots \times g_{12}(n_{12,\cdot}; F_{IS}, q_{12}) \\ g_i(n_{i,\cdot}; F_{IS}, q_i) &= \frac{(\sum_j n_{ij})!}{\prod_j n_{ij}!} \left[q_i^2 + F_{IS} q_i (1 - q_i)\right]^{n_{i1}} \left[2 q_i (1 - q_i)(1 - F_{IS})\right]^{n_{i2}} \left[(1 - q_i)^2 + q_i (1 - q_i) F_{IS}\right]^{n_{i3}} \end{aligned}$$

f is a product of truncated (perhaps better described as censored) normal distributions

$$\begin{aligned} f(q; p, c) &= \prod_{i=1}^{12} f_i(q_i; p, c) \\ f_i(q_i; p, c) &= \begin{cases} \Phi_{p, cp(1-p)}(0) & q_i = 0 \\ \phi_{p, cp(1-p)}(q_i) & 0 < q_i < 1 \\ 1 - \Phi_{p, cp(1-p)}(1) & q_i = 1 \end{cases} \end{aligned}$$

where $\phi_{p,cp(1-p)}$ is the pdf of a normal distribution with mean $\mu = p$ and variance $\sigma^2 = cp(1-p)$ and $\Phi_{p,cp(1-p)}$ is the corresponding CDF.

(b) (20 pts) The preceding likelihood derives from the law of total probability, where the integration is over all the possible outcomes of the subpopulation allele frequencies q. In general, it is difficult to compute the integral in eq. (1), however, there are a lot of data in the table, so $q = (q_1, \ldots, q_{12})$ and p may be well-estimated by the sample proportions $\hat{q} = (\hat{q}_1, \ldots, \hat{q}_{12})$ and $\hat{p}$ (shown in table). Under this assumption, the log likelihood becomes approximately

$$\ln L(n \mid F_{IS}, c) \approx \ln g(n; F_{IS}, \hat{q}) + \ln f(\hat{q}; \hat{p}, c) \tag{2}$$

now a function of only two unknown population parameters (F_{IS}, c). Write a function to compute the right-hand side of eq. (2), and use the `R` function `optim` to find the MLEs $\hat{F}_{IS}$ and $\hat{c}$.

Solution:

The following code is not the best example of how to maximize the log likelihood. Because I have imposed constraints on the parameters $F_{IS} \in (-1, 1)$ and $c \in (0, 100)$, `optim` defaults to the L-BFGS-B algorithm, but this algorithm computes numerical derivatives and thus requires finite values returned from the log likelihood function. However, the log likelihood is not always finite, so I return a very small (negative) number whenever this happens, to indicate that optim should avoid these parameters choices. However, the large number leads to incorrect estimation of derivatives by L-BFGS-B. For example, if it calls `ll.fxn` at multiple parameter values that lead to `Inf` log likelihood, then it will estimate the derivative to be flat. The end result is it does not always converge. A more stable approach is described in HW8.

```
mle <- function(n, F.IS, c, p, q, small=1e-6) {
    optim(c(F.IS,c), ll.fxn, n=n, q.hat=q, p.hat=p, lower=c(-1,0,0)+small, upper=c(1-small, 100, 1-small))$par
}# mle

lprob.n <- function(n, q, F.IS) {
    # compute current genotype probabilities
    P <- matrix(c(q^2+F.IS*q*(1-q), 2*q*(1-q)*(1-F.IS), (1-q)^2+F.IS*q*(1-q)), ncol=3, byrow=F)
    # check for invalid parameter choices
    if(min(P)<0 | max(P)>1) return(-Inf)
    return(sum(n*log(P)))
}

lprob.q <- function(q, p, c) {
    sd = sqrt(c*p*(1-p))
    prob.0 <- pnorm(0, mean=p, sd=sd)
    prob.1 <- 1 - pnorm(1, mean=p, sd=sd)
    prob.q <- ifelse(q<0, prob.0, ifelse(q>1, prob.1, dnorm(q, mean=p, sd=sd)))
    if(min(prob.q) <= 0) return(-Inf)
    return(sum(log(prob.q)))
}

ll.fxn <- function(param, n, q.hat, p.hat=NA, max.ll=1000000) {
    F.IS <- param[1]
    c <- param[2]
    if(is.na(p.hat)) p <- param[3]  # estimate p too
    else p <- p.hat
    lp <- lprob.n(n, q.hat, F.IS)
    qp <- lprob.q(q.hat, p, c)
    tmp <- - (lp + qp)
```

```
    cat(c("ll.fxn:", F.IS, c, p, ":", tmp, "\n"))
    # optim method requires finite value
    if(is.infinite(tmp) || is.nan(tmp)) return(max.ll)
    tmp
}

# enter and print data
n <- matrix(c(208, 484, 311, 202, 513, 288, 705, 305,  29, 114, 456, 473, 772, 207, 10, 95, 446, 471, 791, 252,  28, 520
print(n)

# compute approximate mles for q & p
q.hat <- (2*n[,1]+n[,2])/rowSums(n)/2
p.hat <- sum(2*n[,1] + n[,2])/sum(n)/2

# pick initial values for F.IS and c
init <- c(runif(n=1, min=-1, max=1), runif(n=1))

# estimate and print mle (you should run multiple times)
est <- mle(n=n, F.IS=init[1], c=init[2], p=p.hat, q=q.hat)
print(paste("c", est[2]))
```

Successful output looks like the following

```
      [,1] [,2] [,3]
 [1,]  208  484  311
 [2,]  202  513  288
 [3,]  705  305   29
 [4,]  114  456  473
 [5,]  772  207   10
 [6,]   95  446  471
 [7,]  791  252   28
 [8,]  520  358   82
 [9,]  279  489  229
[10,]  395  465  165
[11,]  686  249   23
[12,]  434  459  127
ll.fxn: 0.090877506416291 0.162943334784359 0.622318481848185 : 10695.6920997967
ll.fxn: 0.091877506416291 0.162943334784359 0.622318481848185 : 10696.5937139732
ll.fxn: 0.089877506416291 0.162943334784359 0.622318481848185 : 10694.8009448112
ll.fxn: 0.090877506416291 0.163943334784359 0.622318481848185 : 10695.6909291997
ll.fxn: 0.090877506416291 0.161943334784359 0.622318481848185 : 10695.6935122316
ll.fxn: -0.999999 1.45445931892435 0.622318481848185 : Inf
ll.fxn: -0.998999 1.45445931892435 0.622318481848185 : Inf
ll.fxn: -0.999999 1.45445931892435 0.622318481848185 : Inf
ll.fxn: -0.999999 1.45545931892435 0.622318481848185 : Inf
ll.fxn: -0.999999 1.45345931892435 0.622318481848185 : Inf
... # Luckily it gets past this point; unsuccessful runs might stop here # ...
ll.fxn: 0.00922210021030553 0.172091320060812 0.622318481848185 : 10658.3117335938
ll.fxn: 0.00922210021030553 0.170091320060812 0.622318481848185 : 10658.3107360318
ll.fxn: 0.00911835474881214 0.168476877715218 0.622318481848185 : 10658.3104745588
ll.fxn: 0.0101183547488121 0.168476877715218 0.622318481848185 : 10658.3162462192
ll.fxn: 0.00811835474881214 0.168476877715218 0.622318481848185 : 10658.3165323440
ll.fxn: 0.00911835474881214 0.169476877715218 0.622318481848185 : 10658.3105421786
ll.fxn: 0.00911835474881214 0.167476877715218 0.622318481848185 : 10658.3106187781
ll.fxn: 0.00913066017710836 0.168662836344682 0.622318481848185 : 10658.3104704526
ll.fxn: 0.0101306601771084 0.168662836344682 0.622318481848185 : 10658.3163874741
ll.fxn: 0.00813066017710836 0.168662836344682 0.622318481848185 : 10658.3163824699
ll.fxn: 0.00913066017710836 0.169662836344682 0.622318481848185 : 10658.3105769148
ll.fxn: 0.00913066017710836 0.167662836344682 0.622318481848185 : 10658.3105748970
Warning message:
In optim(c(F.IS, c), ll.fxn, n = n, q.hat = q, p.hat = p, lower = c(-1,  :
  bounds can only be used with method L-BFGS-B
[1] 0.00913066 0.16866284
```

(c) (10 pts) Convert your MLEs into estimates of the three F statistics. Recall that F_{ST} is related to the variance of the distribution of q. To obtain the variance of the truncated normal, you may use a numerical

approach.

Solution:

We have $\hat{F}_{IS} = 0.009$. Now, recall F_{ST} and F_{IT} are given as functions of F_{IS}, p, and c, so their MLEs are functions of the MLEs.

$$\begin{aligned} \hat{F}_{ST} &= \frac{\widehat{\mathrm{Var}}(q)}{\hat{p}(1-\hat{p})} \\ \hat{F}_{IT} &= (1-\hat{F}_{ST})(1-\hat{F}_{IS}) \end{aligned}$$

Here, $\hat{p} = 0.6223$ is assumed fixed at our original estimate and $\widehat{\mathrm{Var}}(q)$ is the variance of the censored normal with pdf $\phi_{\hat{p},\hat{c}\hat{p}(1-\hat{p})}$. To estimate this variance I use simple Monte Carlo sampling.

```
# numeric estimation of the variance of a censored normal distribution
cnorm <- function(p, c, samples=10000) {
   x <- rnorm(n=samples, mean=p, sd=sqrt(c*p*(1-p)))
   # censor non-probability values
   x[x<0] <- 0
   x[x>1] <- 1
   list(mean=mean(x), var=var(x))
}
F.IS <- est[1]
c.hat <- est[2]
F.ST <- cnorm(p=p.hat, c=c.hat)$var/p.hat/(1-p.hat)
F.IT <- 1-(1-F.IS)*(1-F.ST)
print(paste(c("F.IS", "F.ST", "F.IT"), c(F.IS, F.ST, F.IT)))
```

with output

```
[1] "F.IS 0.00913045416586168" "F.ST 0.161719913796534"
[3] F.IT 0.164567622723696"
```

There is little evidence of deviation from HWE within the subpopulations, but there is substantial differentiation between subpopulations ($F_{ST} = 0.16$) and therefore overall deviation from HWE, expressed as correlation of alleles within individuals sampled from the total population, $F_{IT} = 0.83$.

This result agrees nicely with the conditions used to simulate this data, as shown below

```
simulate <- function(n.spop=12, mean.sample.size=1000, p.popn=0.6, cparam=0.2, F.IS=0.0) {
   n <- matrix(NA, nrow=n.spop, ncol=3)
   for(i in 1:n.spop) { # for each population
      # generate its allele frequency from censored normal
      q <- rnorm(n=1, mean=p.popn, sd=sqrt(cparam*p.popn*(1-p.popn)))
      if(q<0) q <- 0
      else if(q>1) q <- 1
      # compute genotype proportions
      P <- c(q^2 + F.IS*q*(1-q), 2*q*(1-q)*(1-F.IS), (1-q)^2 + F.IS*q*(1-q))
      # allow for some variation in sample size per population
      n.sample <- rpois(n=1, lambda=mean.sample.size)
      # conditional on F.IS and q, the data are multinomial
      n[i,] <- t(rmultinom(n=1, size=n.sample, prob=P))
```

```
   }
   n
}# end simulate
```