# STA 360: Assignment 10

Michael Lin

April 7, 2015

1. See appendix for procedure and result of hypothesis testing.
2. This is Hoff 9.2. "CI.a" refers to confidence interval of parameters from part a. "CI.b" refers to confidence interval of parameters from part b (i.e. after performing model selection and averaging procedure).

| | npreg | bp | skin | bmi | ped | age | intercept |
|---|---|---|---|---|---|---|---|
| CI.a 2.5% | -1.6786677 | -0.03009734 | -0.1454174 | 0.1436408 | 3.121381 | 0.4489052 | 35.93730 |
| CI.a 97.5% | 0.3466288 | 0.43212360 | 0.5095611 | 1.1733495 | 18.239821 | 1.0743018 | 68.15189 |
| CI.b 2.5% | -1.093742 | 0 | 0 | 0.09494546 | 0 | 0.4489052 | 35.93730 |
| CI.b 97.5% | 0 | 0.3237489 | 0.2977179 | 1.17334947 | 17.56353 | 1.0743018 | 68.15189 |
| $\Pr(\beta_j \neq 0 \vert y)$ | 0.085 | 0.158 | 0.095 | 0.989 | 0.670 | 1.000 | 1.000 |

From the confidence intervals from part b, it appears that coefficients are similar between the two parts. However, coefficients for "npreg," "bp" and "skin" may be 0 since their 95% confidence intervals cover a region very close to 0. Despite having 0 in the lower bound of confidence interval for "ped", the coefficient does not appear to be 0 since the confidence interval extends relatively far away from 0.

Compute $p(H_k | x, y)$ for $k = 0, 1$ :

- $p(x_{1:n}, y_{1:m} | H_1) = p(x_{1:n} | H_1)\, p(y_{1:m} | H_1)$

$$
\begin{aligned}
p(x_{1:n}|H_1) &= \int p(x_{1:n}|\lambda_c, H_1)\, p(\lambda_c|H_1)\, d\lambda_c \\
&= \int \left(\prod_{i=1}^{n} Po(x_i|\lambda_c)\right) Ga(\lambda_c|a, b)\, d\lambda_c \\
&= \int \left(\prod_{i=1}^{n} \frac{\lambda_c^{x_i}}{x_i!} e^{-\lambda_c}\right) \frac{b^a}{\Gamma(a)} \lambda_c^{a-1} e^{-b\lambda_c}\, d\lambda_c \\
&= \int \left(\frac{\lambda_c^{\Sigma x_i}}{\prod_{i=1}^{n} x_i!} e^{-n\lambda_c}\right) \frac{b^a}{\Gamma(a)} \lambda_c^{a-1} e^{-b\lambda_c}\, d\lambda_c \\
&= \int \frac{b^a}{\left(\prod_{i=1}^{n} x_i!\right)\Gamma(a)} \lambda_c^{(a+\Sigma x_i)-1} e^{-(b+n)\lambda_c}\, d\lambda_c \\
&= \frac{b^a}{\Gamma(a)\left(\prod_{i=1}^{n} x_i!\right)} \int \lambda_c^{(a+\Sigma x_i)-1} e^{-(b+n)\lambda_c}\, d\lambda_c \\
&= \frac{b^a}{\Gamma(a)\left(\prod_{i=1}^{n} x_i!\right)} \cdot \frac{\Gamma(a+\Sigma x_i)}{(b+n)^{(a+\Sigma x_i)}} \int Ga(\lambda_c | a+\Sigma x_i, b+n)\, d\lambda_c \\
&= \frac{b^a\, \Gamma(a+\sum_{i=1}^{n} x_i)}{\Gamma(a)\left(\prod_{i=1}^{n} x_i!\right)(b+n)^{(a+\sum_{i=1}^{n} x_i)}}
\end{aligned}
$$

By symmetry,

$$
\begin{aligned}
p(y_{1:m}|H_1) &= \int p(y_{1:m}|\lambda_t, H_1)\, p(\lambda_t|H_1) \\
&= \frac{b^a\, \Gamma(a+\sum_{j=1}^{m} y_j)}{\Gamma(a)\left(\prod_{j=1}^{m} y_j!\right)(b+m)^{(a+\sum_{j=1}^{m} y_j)}}
\end{aligned}
$$

$$
p(x_{1:n}|H_1)\,p(y_{1:m}|H_1) = \frac{b^{2a}\, \Gamma(a+\sum_{i=1}^{n} x_i)\, \Gamma(a+\sum_{j=1}^{m} y_j)}{\Gamma^2(a)\left(\prod_{i=1}^{n} x_i!\right)\left(\prod_{j=1}^{m} y_j!\right)(b+n)^{(a+\Sigma x_i)}(b+m)^{(a+\Sigma y_j)}}
$$

- $\log p(x_{1:n}, y_{1:m} | H_1) = \log p(x_{1:n} | H_1) + \log p(y_{1:m} | H_1)$

$$= \log \left[ \frac{b^a \Gamma(a + \sum_{i}^{n} x_i)}{\Gamma(a)(\prod_i x_i!)(b+n)^{(a+\sum x_i)}} \right] + \log \left[ \frac{b^a \Gamma(a + \sum_{j=1}^{m} y_j)}{\Gamma(a)(\prod_{j=1}^{m} y_j!)(b+m)^{(a+\sum y_j)}} \right]$$

$$= 2a \log b + \log \Gamma(a + \sum x_i) - 2\log \Gamma(a) - \sum_{i=1}^{n} \sum_{k=1}^{x_i} \log(k) - (a + \sum x_i) \log(b+n)$$
$$+ \log \Gamma(a + \sum y_j) - \sum_{j=1}^{m} \sum_{k=1}^{y_j} \log(k) - (a + \sum y_j) \log(b+m)$$

- $p(x_{1:n}, y_{1:m} | H_0) = \int p(x_{1:n}, y_{1:m} | \lambda, H_0)\, p(\lambda | H_0)\, d\lambda$

$$= \int \prod_{i=1}^{n} Po(x_i | \lambda) \prod_{j=1}^{m} Po(y_j | \lambda)\ Ga(\lambda | a, b)\, d\lambda$$

$$= \int \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} \prod_{j=1}^{m} \frac{\lambda^{y_j} e^{-\lambda}}{y_j!} \cdot \frac{b^a}{\Gamma(a)} \lambda^{a-1} e^{-b\lambda}\, d\lambda$$

$$= \int \frac{\lambda^{(\sum x_i + \sum y_j)} e^{-(n+m)\lambda}}{\prod_{i=1}^{n} x_i! \prod_{j=1}^{m} y_j!} \cdot \frac{b^a}{\Gamma(a)} \lambda^{a-1} e^{-b\lambda}\, d\lambda$$

$$= \frac{b^a}{(\prod_{i=1}^{n} x_i! \prod_{j=1}^{m} y_j!)\Gamma(a)} \int \lambda^{(a + \sum x_i + \sum y_j) - 1} e^{-(n+m+b)\lambda}\, d\lambda$$

$$= \frac{b^a\, \Gamma(a + \sum x_i + \sum y_j)}{(\prod_{i=1}^{n} x_i! \prod_{j=1}^{m} y_j!)\Gamma(a)(n+m+b)^{(a + \sum x_i + \sum y_j)}} \int Ga(\lambda | a + \sum x_i + \sum y_j, n+m+b)\, d\lambda$$

$$= \frac{b^a\, \Gamma(a + \sum x_i + \sum y_j)}{(\prod_{i=1}^{n} x_i!)(\prod_{j=1}^{m} y_j!)\Gamma(a)(n+m+b)^{(a + \sum x_i + \sum y_j)}}$$

- $\log p(x_{1:n}, y_{1:m} | H_0)$

$$= a \log b - (a + \sum x_i + \sum y_j) \log(n+m+b) - \sum_{i=1}^{n} \log(x_i!) - \sum_{j=1}^{m} \log(y_j!)$$
$$+ \log \Gamma(a + \sum x_i + \sum y_j) - \log \Gamma(a)$$

<u>Notes</u>:

$$\log \Gamma(a + \sum x_i) = \log[(a + \sum x_i - 1)!]$$
$$= \sum_{k=1}^{a + \sum x_i - 1} \log(k)$$

$$\log \Gamma(a + \sum y_j) = \log[(a + \sum y_j - 1)!]$$
$$= \sum_{k=1}^{a + \sum y_j - 1} \log(k)$$

- If $a=4$, $b=0.02$:

$p(H_1 | x,y) \approx 0.981898$
$p(H_0 | x,y) \approx 0.018102$
$B_{10} \approx 162.724$
Prior odds $= 0.25/0.75 = 1/3$
Post odds $= 54.241$

Before the data, the odds of different means are 3 times larger than odds of same mean.

Given the data, the odds of different means are 54 times larger than odds of same mean.

The data provided strong evidence that the means are different.

- The prior seems slightly unreasonable because it implies the mean of $X \cup Y$ is around 200 when the data shows the mean is higher. On the other hand, the prior is too informative because it seems to be too similar to distribution of $x, y$.

| | $a=5$ $b=0.02$ | $a=2$ $b=0.005$ | $a=2.5$ $b=0.01$ |
|---|---|---|---|
| $p(H_1 \mid x,y)$ | 0.98332 | 0.96598 | 0.55112 |
| $p(H_0 \mid x,y)$ | 0.01668 | 0.0340 | 0.44888 |
| $B_{10}$ | 176.861 | 85.174 | 3.6832 |
| Post odds | 58.954 | 28.391 | 1.2277 |

A less informative prior lowers the posterior probabilities and odds.

R code for hypothesis testing:

```r
## Load data ##
x = c(204, 215, 182, 225, 207, 188, 205, 227, 190, 211, 196, 203)
y = c(211, 233, 244, 241, 195, 252, 238, 249, 220, 213)

## Define parameters ##
a = 2.5     #(shape)
b = 0.01  #(rate)
n = length(x)
m = length(y)

## Define priors ##
H1.prior = 1/4
H0.prior = 3/4

## Marginal ##
w = rep(0, n) #w[i] = log(x[i]!)
for(i in 1:n){
  w[i] = sum(log(seq(1,x[i],1)))
}

v = rep(0, m)
for(i in 1:m){
  v[i] = sum(log(seq(1,y[i],1)))
}

f = sum(log(seq(1,a+sum(x)-1,1))) #f = log(gamma(a+sum(x)))
g = sum(log(seq(1,a+sum(y)-1,1))) #g = log(gamma(a+sum(y)))
h = sum(log(seq(1,a+sum(x)+sum(y)-1,1))) #h = log(gamma(a+sum(x)+sum(y)))

H1.loglik = 2*a*log(b) + f - 2*log(gamma(a)) - sum(w) -
  (a+sum(x))*log(b+n) + g - sum(v) - (a+sum(y))*log(b+m)
H0.loglik = a*log(b) + h - (a+sum(x)+sum(y))*log(n+m+b) - sum(w) - sum(v) - log(gamma(a))

## Posterior ##
H1.post = exp(H1.loglik)*H1.prior/(exp(H1.loglik)*H1.prior+exp(H0.loglik)*H0.prior)
H0.post = exp(H0.loglik)*H0.prior/(exp(H1.loglik)*H1.prior+exp(H0.loglik)*H0.prior)

## Bayes Factor ##
B10 = exp(H1.loglik - H0.loglik)

## Odds ##
odds.prior = H1.prior/H0.prior
odds.post = H1.post/H0.post
```

R code for 9.2:

```
library(MCMCpack)
set.seed(1)
## load data ##
data = read.table("http://www.stat.washington.edu/people/pdhoff/Book/Data/hwdata/azdiabetes.dat", se
data = data[,c(1:7)]
data$intercept = rep(1,nrow(data))
y = data[,2]
X = as.matrix(data[,-2])

## initialize parameters ##
g = length(y); nu0 = 2; s20 = 1;
S = 1000;
n = dim(X)[1]; p = dim(X)[2];

## compute beta ##
Hg = (g/(g+1)) * X%*%solve(t(X)%*%X)%*%t(X)
SSRg = t(y)%*%( diag(1, nrow=n) - Hg) %*%y

s2 = 1/rgamma(S, (nu0+n)/2, (nu0*s20+SSRg)/2)
Vb = g*solve(t(X)%*%X)/(g+1)
Eb = Vb%*%t(X)%*%y

E = matrix(rnorm(S*p, 0, sqrt(s2)), S, p)
beta = t( t(E%*%chol(Vb)) + c(Eb))

## posterior (beta) confidence interval ##
beta.npreg.ci = quantile(beta[,1], c(0.025, 0.975))
beta.bp.ci = quantile(beta[,2], c(0.025, 0.975))
beta.skin.ci = quantile(beta[,3], c(0.025, 0.975))
beta.bmi.ci = quantile(beta[,4], c(0.025, 0.975))
beta.ped.ci = quantile(beta[,5], c(0.025, 0.975))
beta.age.ci = quantile(beta[,6], c(0.025, 0.975))
beta.int.ci = quantile(beta[,7], c(0.025, 0.975))

## function: compute marginal probability ##
lpy.X = function(y, X, g=length(y), nu0=1, s20=try(summary(lm(y~-1+X))$sigma^2, silent=T)){
  n = dim(X)[1]; p = dim(X)[2];
  if(p==0){
    Hg = 0
    s20 = mean(y^2)
  }
  if(p>0){
    Hg = (g/(g+1)) * X%*%solve(t(X)%*%X)%*%t(X)
  }
  SSRg = t(y)%*%( diag(1, nrow=n) - Hg) %*%y

  -0.5*(n*log(pi)+p*log(1+g)+(nu0+n)*log(nu0*s20+SSRg)-nu0*log(nu0*s20)) +
    lgamma((nu0+n)/2) - lgamma(nu0/2)
}

## MCMC setup ##
z = rep(1, dim(X)[2])
lpy.c = lpy.X(y, X[, z==1, drop=F])
```

```
S = 1000
Z = matrix(NA, S, dim(X)[2])

## Gibbs sampler ##
for(s in 1:S){
  for(j in sample(1:dim(X)[2])){
    zp = z
    zp[j] = 1-zp[j]
    lpy.p = lpy.X(y, X[, zp==1, drop=F])
    r = (lpy.p - lpy.c)*(-1)^(zp[j]==0)
    z[j] = rbinom(1, 1, 1/(1+exp(-r)))
    if(z[j]==zp[j]){
      lpy.c = lpy.p
    }
  }

  Z[s,] = z
}


## find new beta ##
BETA = matrix(0, S, p)
for(s in 1:S){
  z = Z[s,]
  X.z = NULL
  for(i in 1:p){
    if(z[i]!=0){
      X.z = cbind(X.z, X[,i])
    }
  }

  Hg.new = (g/(g+1)) * X.z%*%solve(t(X.z)%*%X.z)%*%t(X.z)
  SSRg.new = t(y)%*%( diag(1, nrow=n) - Hg.new) %*%y

  s2.new = 1/rgamma(1, (nu0+n)/2, (nu0*s20+SSRg.new)/2)
  Vb.new = g*solve(t(X.z)%*%X.z)/(g+1)
  Eb.new = Vb.new%*%t(X.z)%*%y

  E.new = matrix(rnorm(1*ncol(X.z), 0, sqrt(s2.new)), 1, ncol(X.z))
  beta.new = t( t(E.new%*%chol(Vb.new)) + c(Eb.new))

  for(j in 1:sum(z)){
    BETA[s, which(z!=0)[j] ] = beta.new[j]
  }

}

## Pr(beta!=0|y) ##
Z.mean = rep(0,p)

for(i in 1:p){
  Z.mean[i] = mean(Z[,i])
}

```

```
## new posterior(BETA) confidence interval ##
BETA.npreg.ci = quantile(BETA[,1], c(0.025, 0.975))
BETA.bp.ci = quantile(BETA[,2], c(0.025, 0.975))
BETA.skin.ci = quantile(BETA[,3], c(0.025, 0.975))
BETA.bmi.ci = quantile(BETA[,4], c(0.025, 0.975))
BETA.ped.ci = quantile(BETA[,5], c(0.025, 0.975))
BETA.age.ci = quantile(BETA[,6], c(0.025, 0.975))
BETA.int.ci = quantile(BETA[,7], c(0.025, 0.975))

## new BETA1 ##
BETA1 = beta*Z
quantile(BETA1[,1], c(0.025, 0.975))
quantile(BETA1[,2], c(0.025, 0.975))
quantile(BETA1[,3], c(0.025, 0.975))
quantile(BETA1[,4], c(0.025, 0.975))
quantile(BETA1[,5], c(0.025, 0.975))
quantile(BETA1[,6], c(0.025, 0.975))
quantile(BETA1[,7], c(0.025, 0.975))
```