Stat 591 – Homework 06 *Solutions*

1. Consider independent data $(O_1, \ldots, O_N)$, modeled as

$$O_i \mid (E_i, \theta_i) \overset{\text{ind}}{\sim} \mathsf{Pois}(E_i\theta_i), \quad i = 1, \ldots, N,$$

where $(E_1, \ldots, E_N)$ are known constants. If we take an iid $\mathsf{Gam}(a, b)$ prior for $(\theta_1, \ldots, \theta_N)$, with b a rate parameter, then the posterior distribution of θ_i, given O_i, is $\mathsf{Gam}(a + O_i, b + E_i)$. Therefore, the Bayes estimates are

$$\mathsf{E}(\theta_i \mid O_i, E_i, a, b) = \frac{a + O_i}{b + E_i}, \quad i = 1, \ldots, N.$$

The parametric empirical Bayes estimates plug in estimates of (a, b) as opposed to the hierarchical Bayes estimates based on an assumed prior for (a, b). To estimate (a, b), there are a number of directions we could take. Here's two.

- My preferred method is maximum marginal likelihood. For this, we need the marginal distribution of $(O_1, \ldots, O_N)$ given (a, b). It's easy to see that the O_i's are independent, and their probability mass function looks like

$$\begin{aligned}
\mathsf{P}(O_i \mid a, b) &= \int e^{-E_i\theta_i} \frac{(E_i\theta_i)^{O_i}}{O_i!} \frac{b^a}{\Gamma(a)} \theta_i^{a-1} e^{-b\theta_i} \, d\theta_i \\
&= \frac{b^a}{\Gamma(a)} \frac{E_i^{O_i}}{O_i!} \int \theta_i^{a+O_i-1} e^{-(E_i+b)\theta_i} \, d\theta_i \\
&= \frac{\Gamma(a + O_i)}{\Gamma(a) O_i!} \Big(\frac{E_i}{E_i + b}\Big)^{O_i} \Big(\frac{b}{E_i + b}\Big)^{a}.
\end{aligned}$$

  In fact, this gamma mixture of Poissons is negative binomial mass function, i.e., the marginal distribution of O_i is $\mathsf{NBin}(a, p_i)$, where $p_i = E_i/(E_i + b)$. Then the marginal likelihood is the product of the mass functions above:

$$L(a, b) = \prod_{i=1}^{N} \frac{\Gamma(a + O_i)}{\Gamma(a) O_i!} \Big(\frac{E_i}{E_i + b}\Big)^{O_i} \Big(\frac{b}{E_i + b}\Big)^{a}.$$

  There's no closed-form expressions, but we can maximize this function (or its logarithm) numerically using, e.g., the `optim` function in R.

- Another approach is a sort of method of moments. Think of $R_i = O_i/E_i$ as a proxy for θ_i. Now set the sample mean and variance of R_i equal to the prior mean and variance for θ_i, i.e., set $\bar{R} : N^{-1} \sum_{i=1}^{N} R_i = a/b$ and $S_R^2 := (N-1)^{-1} \sum_{i=1}^{N} (R_i - \bar{R})^2 = a/b^2$ and solve for (a, b). That is,

$$\hat{a} = \bar{R} \quad \text{and} \quad \hat{b} = \bar{R}^2 / S_R^2.$$

In any case, once $(\hat{a}, \hat{b})$ are available, just plug them in to the Bayes estimate for given (a, b) to get the empirical Bayes estimate, that is,

$$\hat{\theta}_i^{\text{eb}} = \frac{\hat{a} + O_i}{\hat{b} + E_i}, \quad i = 1, \ldots, N.$$

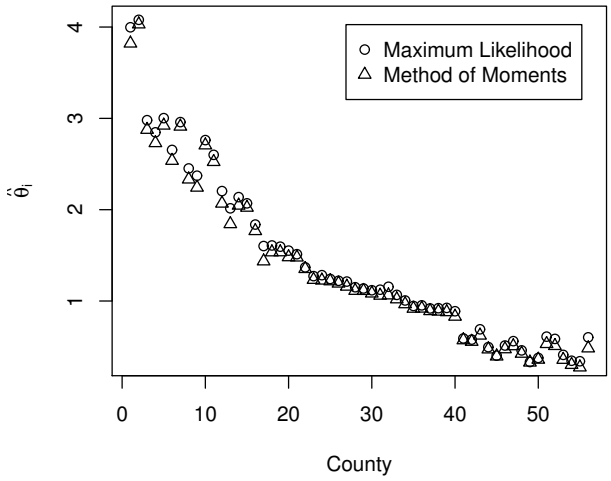

Figure 1: Plot of the two Poisson empirical Bayes estimates applied to the data in Table 10.1 in [GDS].

The data provided in Table 10.1 in [GDS] are observed and expected disease counts for $N = 56$ counties. Plots of the two Poisson empirical Bayes estimates $\hat{\theta}_i^{\text{eb}}$ are displayed in Figure 1. As you can see, despite the method of moments procedure being quite naive, it gives almost indistinguishable empirical Bayes estimates compared to the more sophisticated maximum likelihood approach.

2. For the hierarchical normal model, with the given non-informative priors, the full conditionals are similar to those derived in Homework #5 for the ANOVA model. These full conditionals are summarized below.

$$\mu = (\mu_1, \ldots, \mu_p) \mid (\sigma^2, m, v, X) \overset{\text{ind}}{\sim} \mathsf{N}\Big(\frac{n\bar{X}_j + \sigma^2 m}{nv + \sigma^2}, \frac{v\sigma^2}{nv + \sigma^2}\Big), \quad j = 1, \ldots, p$$

$$\sigma^2 \mid (\mu, m, v, X) \sim \mathsf{InvGam}\Big(\frac{np}{2}, \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{n}(X_{jk} - \mu_j)^2\Big)$$

$$m \mid (\mu, \sigma^2, v, X) \sim \mathsf{N}\Big(\frac{1}{p}\sum_{j=1}^{p}\mu_j, \frac{v}{p}\Big)$$

$$v \mid (\mu, \sigma^2, m, X) \sim \mathsf{InvGam}\Big(\frac{p}{2} - 1, \frac{1}{2}\sum_{j=1}^{p}(\mu_j - m)^2\Big).$$

Using these full conditionals, a Gibbs sampler can be constructed to produce a posterior sample:

$$\{(\mu^{(r)}, \sigma^{2(r)}, m^{(r)}, v^{(r)}) : r = 1, \ldots, R\}.$$

One way to use this sample to evaluate $\mathsf{E}(\mu_j \mid X)$ is

$$\mathsf{E}(\mu_j \mid X) \approx \frac{1}{R}\sum_{r=1}^{R}\mu_j^{(r)}.$$

The Rao–Blackwellized approach is to "integrate out" μ_j analytically first, then average the conditional expectation of μ_j given all the other parameters. That is,

$$\mathsf{E}(\mu_j \mid X) \approx \frac{1}{R}\sum_{r=1}^{R} \mathsf{E}(\mu_j \mid \sigma^{2(r)}, m^{(r)}, v^{(r)}) = \frac{1}{R}\sum_{r=1}^{R}\Big(\frac{n\bar{X}_j + \sigma^{2(r)}m^{(r)}}{nv^{(r)} + \sigma^{2(r)}}\Big).$$

This Rao–Blackwellized Monte Carlo approximation is better in the sense that it will have smaller variance than the naive Monte Carlo approximation above.

3. (a) If one can get a full posterior sample of $(\mu_1, \ldots, \mu_M, p, \sigma^2, V)$, then the exclusion probability $p_i = \mathsf{P}(\mu_i = 0 \mid x)$ can be found by simple averaging. That is, $p_i \approx R^{-1}\sum_{r=1}^{R} I\{\mu_i^{(r)} = 0\}$, the proportion of times μ_i equals zero in the posterior sample, is the Monte Carlo approximation of p_i.

   (b) It is computationally costly to produce a sample from the full $(M+3)$-dimensional posterior distribution, when $M \sim 10000$. Instead of this, Scott & Berger opt for an importance sampling strategy to evaluate p_i's. The key is that they can write p_i as an expectation of a known function $h_i(p, \sigma^2, V)$ of (p, σ^2, V) with respect to its posterior distribution, i.e., with respect to $\pi(p, \sigma^2, V \mid x)$. So the idea is to generate a sample from an approximation to the posterior $\pi(p, \sigma^2, V \mid x)$ and use that sample to compute an importance sampling estimate of p_i. The two key observations are

   - With data of size $M \sim 10000$, the posterior distribution for (p, σ^2, V), a three-dimensional quantity, ought to be approximately normal. So, a reasonable proposal distribution is a Student-t version of the normal approximation of the posterior.
   - The same sample of (p, σ^2, V) from the Student-t proposal above can be used for ALL p_i approximations. That is, one does not need separate samples of (p, σ^2, V) for each p_i, $i = 1, \ldots, M$, the importance sampling can be done once and for all.

4. Here are my solutions to all three of the computational exercises.

   (a) Scott & Berger implement an importance sampling strategy to compute the posterior inclusion probabilities, in their notation, $1 - p_j$. My R code implementation is given below. One thing I discovered that wasn't obvious from the presentation in the paper is that their definition of the "Hessian" of $\pi^\star$ is the matrix of second partial derivatives of $-\log \pi^\star$. I should have realized this sooner, given the discussion related to Equation (4.1) in [GDS].

   The exercise was to reproduce the $n = 25, 100$ rows in Scott & Berger's Table 1 in the case where the prior for p is $\pi(p) = 11p^{10}$. The example has ten observations corresponding to signals, i.e., non-zero means, and n noise observations; the noise observations are independent $\mathsf{N}(0, 1)$ samples. I used 10^5 importance samples, and my results look like the following:

	Signal Observations									
n	-5.65	-5.56	-2.62	-1.20	-1.01	-0.90	-0.15	1.65	1.94	3.57
25	0.912	0.907	0.342	0.089	0.078	0.073	0.058	0.132	0.177	0.626
100	0.993	0.992	0.274	0.056	0.048	0.045	0.034	0.085	0.118	0.673

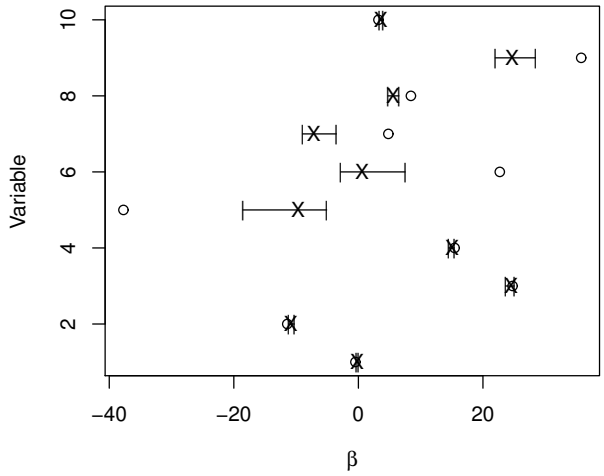

Figure 2: My version of Figure 2 in Park & Casella's paper

(b) The Bayesian lasso is a sort of Bayesian strategy for variable selection in regression, based on an interpretation of the lasso's L_1 penalty as a Laplace prior. The paper gives explicitly the full conditionals needed to construct a Gibbs sampler for posterior interference on the slope parameter β. My R code is given below.

The exercise was to reproduce Figure 2 in Park & Casella's paper. Two points are order. First, the matrix of predictor variables and the vector of response variables should be "standardized." In particular, the columns of the X matrix should have mean zero and variance unity; the response vector y has mean zero. Second, I could not figure out what "standardized coefficients" in the axis of Figure 2 was, so I kept the results in the units of the problem. My version of their figure is in Figure 2.

(c) For the simple many-normal-means example, with the given prior, the posterior distribution is proportional to

$$\omega^{an-1} \prod_{i=1}^{n} \Big\{ \omega e^{-X_i^2/2} \delta_0(\theta_i) + (1-\omega) e^{-(X_i-\theta_i)^2/2} \Big\}.$$

First, if ω were known, then it's clear that the posterior distribution of $\theta = (\theta_1, \ldots, \theta_n)$ is such that the components are independent and each is either 0 or distributed as $\mathsf{N}(X_i, 1)$ based the outcome of a coin flip. The probability of heads on the coin is determined by the coefficients on the two terms in the posterior above. Specifically,

$$\theta_i \mid (X, \omega) \sim \begin{cases} \delta_0 & \text{with prob. } \propto \omega e^{-X_i^2/2} \\ \mathsf{N}(X_i, 1), & \text{with prob. } \propto 1-\omega. \end{cases}$$

Here, by $\propto$, I mean that the probability of taking $\theta_i = 0$ is

$$\frac{\tilde{p}_i}{\tilde{p}_i + (1-\omega)}, \quad \text{where} \quad \tilde{p}_i = \omega e^{-X_i^2/2}.$$

Second, if I know θ, then I know D, the number of θ_i's equal 0. The data don't contribute anything to the conditional posterior of ω, given θ, and, moreover, $D \sim \mathsf{Bin}(n, \omega)$. Since the prior is a beta, conjugacy determines that the (conditional) posterior of ω is

$$\omega \mid (X, \theta) \sim \mathsf{Beta}(an + D, n - D + 1).$$

This determines the full conditionals and a Gibbs sampler is easily constructed. Once the posterior samples of $\theta = (\theta_1, \ldots, \theta_n)$ are available, take the coordinate-wise average to get the posterior mean estimator. My R code is below.

For this exercise, the goal is to implement the above Bayes method in the simulation example in Castillo & van der Vaart's paper. This one has $n = 200$ and the "true" θ is determined by taking $s = s_n$ values of magnitude A and the remaining $n - s$ values equal to 0. Nine different combinations of (s, A) are considered. The following table shows the Monte Carlo approximations of the mean square error $\mathsf{E}_\theta \|\hat{\theta}_n - \theta\|^2$ of the posterior mean estimator described above. This is based on only 100 simulated data sets (same as Castillo & van der Vaart use).

	Signal size A		
s	3	4	5
25	144	140	137
50	161	163	161
100	186	176	179

R code for the Poisson parametric empirical Bayes example.

```
poisson.eb <- function(O, E) {

  loglik <- function(p) {

    a <- exp(p[1])
    b <- exp(p[2])
    t1 <- sum(lgamma(a + O) - lgamma(a))
    t2 <- sum(O * log(E / (E + b)) + a * log(b / (E + b)))
    return(t1 + t2)

  }
  g <- function(p) -loglik(p)
  o <- optim(par=c(0,0), fn=g, method="BFGS")
  a <- exp(o$par[1])
  b <- exp(o$par[2])
  peb1 <- (O + a) / (E + b)
  a0 <- mean(O / E)
  b0 <- a0**2 / var(O / E)
  peb2 <- (O + a0) / (E + b0)
  return(cbind(peb1, peb2))

}

peb.example <- function() {

  O <- c(9, 39, 11, 9, 15, 8, 26, 7, 6, 20, 13, 5, 3, 8, 17, 9, 2, 7, 9, 7,
         16, 31, 11, 7, 19, 15, 7, 10, 16, 11, 5, 3, 7, 8, 11, 9, 11, 8, 6,
         4, 10, 8, 2, 6, 19, 3, 2, 3, 28, 6, 1, 1, 1, 1, 0, 0)
  E <- c(1.4, 8.7, 3.0, 2.5, 4.3, 2.4, 8.1, 2.3, 2.0, 6.6, 4.4, 1.8, 1.1, 3.3, 7.8,
         4.6, 1.1, 4.2, 5.5, 4.4, 10.5, 22.7, 8.8, 5.6, 15.5, 12.5, 6.0, 9.0, 14.4,
         10.2, 4.8, 2.9, 7.0, 8.5, 12.3, 10.1, 12.7, 9.4, 7.2, 5.3, 18.8, 15.8, 4.3,
         14.6, 50.7, 8.2, 5.6, 9.3, 88.7, 19.6, 3.4, 3.6, 5.7, 7.0, 4.2, 1.8)
  out <- poisson.eb(O, E)
  plot(out[,1], pch=1, xlab="County", ylab=expression(hat(theta)[i]))
  points(out[,2], pch=2)
  legend(x="topright", inset=0.05, pch=1:2, c("Maximum Likelihood", "Method of Moments"))
  return(out)

}
```

R code for the Scott–Berger example.

```
sb.lmpost <- function(u, x) {

  v <- u[1]
  sig2 <- u[2]; sig <- sqrt(sig2); vs2 <- sig2 + v; vs <- sqrt(vs2)
  p <- u[3]
  o1 <- sum(log(p * dnorm(x, 0, sig) + (1 - p) * dnorm(x, 0, vs)))
  o2 <- 10 * log(p)
  o3 <- -2 * log(vs2)
  return(o1 + o2 + o3)

}

sb.h <- function(u, x) {

  v <- u[1]
  sig2 <- u[2]
  p <- u[3]
  vs2 <- v + sig2
  o <- 1 + (1 - p) * sqrt(sig2 / vs2) * exp(v * x**2 / (2 * sig2 * vs2)) / p
  return(1 / o)

}

ldmt <- function(x, loc, inv.scale, df) {

  qform <- drop(t(x - loc) %*% inv.scale %*% (x - loc))
  return(-(df + length(loc)) * log(1 + qform / df) / 2)

}

rmt <- function(loc, ch.scale, df) {

  d <- length(loc)
  return(loc + drop(ch.scale %*% rnorm(d)) / sqrt(rchisq(1, df) / df))

}

sb.inclprob <- function(x, nmc) {

  if(!exists("ginv")) library(MASS)
  f <- function(z) {

    u <- c(exp(z[1:2], 1 / (1 + exp(-z[3]))))
    return(-sb.lmpost(u, x) - sum(z) + 2 * log(1 + exp(z[3])))

  }
  o <- optim(fn=f, par=c(0, 0, log(9)), method="BFGS", hessian=TRUE)
  loc <- o$par
  scale <- 5 * ginv(o$hessian) / 3
  inv.scale <- 3 * o$hessian / 5
  ch.scale <- chol(scale)
  df <- 3
  logw <- function(z) -f(z) - ldmt(z, loc, inv.scale, df)
  hh <- function(z) {
```

```
    u <- c(exp(z[1:2], 1 / (1 + exp(-z[3]))))
    return(sb.h(u, x))

  }
  num <- den <- 0
  for(i in 1:nmc) {

    Z <- rmt(loc, ch.scale, df)
    w <- exp(logw(Z))
    num <- num + hh(Z) * w
    den <- den + w

  }
  return(1 - num / den)

}

sb.example <- function(n, nmc) {

  #set.seed(7)
  x <- c(-5.65, -5.56, -2.62, -1.20, -1.01, -0.90, -0.15, 1.65, 1.94, 3.57)
  x <- c(x, rnorm(n))
  pr <- sb.inclprob(x, nmc)
  return(pr[1:10])

}
```

R code for Park–Casella's Bayesian lasso

```
rmnorm <- function(m, V) {

  p <- length(m)
  cV <- chol(V)
  return(drop(m + cV %*% rnorm(p)))

}

rinvgam <- function(a, b) 1 / rgamma(1, a, b)

rinvgauss <- function(mm, ll) {

  p <- length(mm)
  F <- rnorm(p)**2
  X <- mm + mm**2 * F / 2 / ll - mm * sqrt(4 * mm * ll * F + mm**2 * F**2) / 2 / ll
  U <- runif(p)
  o <- (U <= mm / (mm + X)) * X + (U > mm / (mm + X)) * mm**2 / X
  return(o)

}

bayes.lasso <- function(X, Y, M, B, r, d) {

  n <- nrow(X)
  p <- ncol(X)
  XtX <- t(X) %*% X
  tXY <- t(X) %*% Y
  Beta <- InvTau2 <- matrix(0, nrow=M + B, ncol=p)
  Sig2 <- Lambda <- numeric(M + B)
  beta <- solve(XtX, tXY)
  res <- drop(Y - X %*% beta)
  sig2 <- mean(res**2)
  invtau2 <- 1 / beta**2
  lambda <- p * sqrt(sig2) / sum(abs(beta))
  for(m in 1:(M+B)) {

    invD <- diag(invtau2)
    invA <- solve(XtX + invD)
    loc <- invA %*% tXY
    scale <- sig2 * invA
    beta <- rmnorm(loc, scale)
    Beta[m,] <- beta
    res <- drop(Y - X %*% beta)
    sc <- (sum(res**2) + sum(beta**2 * invtau2)) / 2
    sh <- (n + p - 1) / 2
    sig2 <- rinvgam(sh, 1 / sc)
    Sig2[m] <- sig2
    mm <- sqrt(lambda**2 + sig2 / beta**2)
    invtau2 <- rinvgauss(mm, lambda**2)
    InvTau2[m,] <- invtau2
    sh <- r + p / 2
    sc <- d + sum(1 / invtau2) / 2
    lambda <- rgamma(1, sh, 1 / sc)
    Lambda[m] <- lambda
```

```
  }
  o <- list(beta=Beta[-(1:B),], sig2=Sig2[-(1:B)], invtau2=InvTau2[-(1:B),],
            lambda=Lambda[-(1:B)])
  return(o)

}

pc.diabetes.example <- function(M, B) {

  r <- 1
  delta <- 1.78
  file <- "http://homepages.math.uic.edu/~rgmartin/Teaching/Stat591/Bayes/Code/diabetes.txt"
  data <- read.table(file=file, header=TRUE, sep="")
  names <- colnames(data)[-11]
  X <- drop(scale(data[,1:10]))
  Y <- drop(scale(data[,11], scale=FALSE))
  o <- bayes.lasso(X, Y, M, B, r, delta)
  beta <- o$beta
  b <- solve(t(X) %*% X, t(X) %*% Y)
  bb <- apply(beta, 2, median)
  get.ci <- function(v) drop(quantile(v, c(0.025, 0.975)))
  ci <- t(apply(beta, 2, get.ci))
  plot(x=b, y=1:10, ylab="Variable", xlab=expression(beta), xlim=range(b, ci))
  for(i in 1:10) {

    lines(x=ci[i,], y=rep(i, 2))
    points(x=ci[i,], y=rep(i, 2), pch="|")

  }
  points(x=bb, y=1:10, pch="X")
  return(list(gibbs.out=beta, b.ls=b, b.ba=bb, ci=ci))

}
```

R code for Gibbs sampler in Castillo–van der Vaart's example.

```
mnm.gibbs <- function(X, a, M) {

  n <- length(X)
  theta <- matrix(0, ncol=n, nrow=M)
  W <- numeric(M)
  for(m in 1:M) {

    if(m == 1) D <- sum(abs(X) <= sqrt(2 * log(n))) else D <- sum(theta[m-1,] == 0)
    w <- rbeta(1, a * n + D, 1 + n - D)
    W[m] <- w
    u <- runif(n)
    prob <- w * exp(-X**2 / 2)
    p <- prob / (prob + 1 - w)
    tt <- (u > p) * rnorm(n, X, 1)
    theta[m, ] <- tt

  }
  return(list(W=W, theta=theta))

}

cv.sim <- function(n, s, A, reps, a, M) {

  mse <- 0 * outer(s, A)
  for(i in seq_along(s)) {

    for(j in seq_along(A)) {

      true <- c(rep(A[j], s[i]), rep(0, n-s[i]))
      for(r in 1:reps) {

        X <- true + rnorm(n)
        est <- apply(mnm.gibbs(X, a, M)$theta, 2, mean)
        mse[i,j] <- mse[i,j] + sum((est - true)**2) / reps

      }

    }

  }
  rownames(mse) <- s
  colnames(mse) <- A
  return(mse)

}
```