Homework 3

Ian Frankenburg

6/12/2020

Bayesian Model Selection.

Part (a).

Describe and implement a Reversible Jumps MCMC algorithm exploring the model space $p(\gamma|Y)$. Report the estimated inclusion probabilities $p(\gamma_j = 1|Y)$.

I'm just going to write out the math for this part. I want to sample from $p(\gamma|Y)$. I'll do this by sampling from $p(\gamma, \beta_\gamma, \sigma^2|y)$. This will involve sampling from spaces of varying dimension since β_γ is changing. Under model γ, I know $p(\beta_\gamma, \sigma^2|y, \gamma)$ since I did part (b) first:

$$\begin{aligned} p(\sigma^2, \beta|y, \gamma) &\propto p(y|\gamma, \beta_\gamma, \sigma^2)p(\beta_\gamma|, \sigma^2, \gamma)p(\sigma^2|\gamma) \\ &N(m, M) \times IG(n/2 - 1, RSS/2), \end{aligned}$$

with the same m, M, and RSS I use in part (b). Now to sample from $p(\beta_\gamma, \gamma, \sigma^2|y)$. Using $p(\sigma^2, \beta|y\gamma)$:

$$\begin{aligned} p(\beta_\gamma, \gamma, \sigma^2|y) &\propto p(y|\beta_\gamma, \sigma^2, \gamma)p(\beta_\gamma, \sigma^2, \gamma) \\ &= p(y|\beta_\gamma, \sigma^2, \gamma)p(\beta_\gamma, \sigma^2|\gamma)p(\gamma) \\ &= N(m, M) \times IG(n/2 - 1, RSS/2) \times 2^{-q_\gamma} \end{aligned}$$

In coding this, I'd use a Metropolis-Hastings sampler to loop through these two steps, jumping around the space of models while also sampling from the corresponding parameters. Finally, I can get a monte carlo estimate for $p(\gamma_k|y)$ through

$$\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_k(\gamma_i),$$

where this is just an emperical average of the number of times a model γ_k shows up.

Part (b).

Compare your results in (a) with results obtained sampling directly from $p(\gamma|Y)$. You should be able to implement this without the need for RJ-MCMC.

I know $p(\gamma|y) \propto p(y|\gamma)p(\gamma)$, so this will come down to computing $p(y|\gamma)$. Since

$$p(y|\gamma) = \int\int p(y|\beta,\sigma^2,\gamma)p(\beta|\sigma^2,\gamma)p(\sigma^2)d\beta d\sigma^2,$$

I'll work with the integrand. I'm given that we're using a g-prior, so

$$\begin{aligned}
&p(y|\beta,\sigma^2,\gamma)p(\beta|\sigma^2,\gamma)\\
=&N(y;X_\gamma\beta,\sigma^2 I)N(\beta;\frac{\sigma^2}{g}(X_\gamma^\top X_\gamma)^{-1})\\
=&(2\pi\sigma^2)^{-n/2}\exp\big(-\frac{1}{2\sigma^2}(y-X_\gamma\beta)^\top(y-X_\gamma\beta)\big)\exp\big(-\frac{g}{2\sigma^2}\beta^\top X_\gamma^\top X_\gamma\beta\big)
\end{aligned}$$

Combinging the terms in the exponent through completing the square gives

$$\begin{aligned}
&-\frac{1}{2\sigma^2}(y-X_\gamma\beta)^\top(y-X_\gamma\beta)-\frac{g}{2\sigma^2}\beta^\top X_\gamma^\top X_\gamma\beta\\
=&-\frac{1}{2}[y^\top y/\sigma^2-2\beta^\top Xy/\sigma^2+\beta^\top X_\gamma^\top X_\gamma\beta/\sigma^2+g\beta^\top X_\gamma^\top X_\gamma\beta/\sigma^2]\\
=&-\frac{1}{2}[y^\top y/\sigma^2+(\beta-m)M^{-1}(\beta-m)+m^\top M^{-1}m]
\end{aligned}$$

where $M=\sigma^2(X_\gamma^\top X_\gamma)^{-1}/(1+g)$ and $m=MX^\top y/\sigma^2$.

That means $p(y|\beta,\sigma^2,\gamma)p(\beta|\sigma^2,\gamma)$ can be written as

$$(2\pi\sigma^2)^{-n/2}\exp\big(-\frac{1}{2}[y^\top y/\sigma^2+(\beta-m)M^{-1}(\beta-m)+mM^{-1}m]\big).$$

Therefore, when I integrate $p(y|\beta,\sigma^2,\gamma)p(\beta|\sigma^2,\gamma)$, the Gaussian kernel will go away and I'll be left with

$$\begin{aligned}
&\int p(y|\beta,\sigma^2,\gamma)p(\beta|\sigma^2,\gamma)d\beta\\
=&(2\pi\sigma^2)^{-n/2}\exp(\frac{1}{2\sigma^2}y^\top y)\exp(\frac{1}{2}m^\top M^{-1}m)|2\pi M|^{1/2}\\
=&(2\pi)^{(-n+q_\gamma)/2}(\sigma^2)^{(-n+q_\gamma)/2}|(X_\gamma^\top X_\gamma)|^{-1/2}(\frac{1}{g+1})^{q_\gamma/2}\exp(-\frac{1}{2\sigma^2}y^\top y)\exp(\frac{1}{2}m^\top M^{-1}m).
\end{aligned}$$

Since

$$\begin{aligned}
\exp(-\frac{1}{2\sigma^2}y^\top y)\exp(\frac{1}{2}m^\top M^{-1}m)&=\exp\big(-\frac{1}{2\sigma^2}(y^\top y-y^\top X_\gamma(X_\gamma^\top X_\gamma)^{-1}X_\gamma^\top y/(1+g)\big)\\
&=\exp\big(-\frac{1}{2\sigma^2}y^\top(I-\frac{1}{1+g}X_\gamma(X_\gamma^\top X_\gamma)^{-1}X_\gamma^\top)y\big).
\end{aligned}$$

Since $I-\frac{1}{1+g}X_\gamma(X_\gamma^\top X_\gamma)^{-1}X_\gamma^\top$ is a projection onto the residual space, I'll call $y^\top(I-\frac{1}{1+g}X_\gamma(X_\gamma^\top X_\gamma)^{-1}X_\gamma^\top)y$ the RSS.

Now I need to compute

$$\begin{aligned}
&(2\pi)^{-(n+q_\gamma)/2}(\frac{1}{g+1})^{q_\gamma/2}|(X_\gamma^\top X_\gamma)|^{-1/2}\int(\sigma^2)^{(-n+q_\gamma)/2}\exp(-\frac{1}{2\sigma^2}RSS)p(\sigma^2)d\sigma^2\\
=&(2\pi)^{(-n+q_\gamma)/2}(\frac{1}{g+1})^{q_\gamma/2}|(X_\gamma^\top X_\gamma)|^{-1/2}\frac{\Gamma(\frac{n}{2}-1)}{(RSS/2)^{n/2}}
\end{aligned}$$

Therefore,

$$\begin{aligned} p(\gamma|y) &\propto p(y|\gamma)p(\gamma) \\ &= (2\pi)^{-(n+q_\gamma)/2}(\frac{1}{g+1})^{q_\gamma/2}|(X_\gamma^\top X_\gamma)|^{-1/2}\frac{\Gamma(\frac{n}{2}-1)}{(RSS/2)^{n/2}}p(\gamma). \end{aligned}$$

For the Gibbs sampling step, I'm going to update each γ_j component individually. Then the probability that γ_j is one is the ratio

$$p_j := \frac{P(\gamma_j = 1|y, X, \gamma_{z-j})}{P(\gamma_j = 0|y, X, \gamma_{z-j}) + P(\gamma_j = 1|y, X, \gamma_{z-j})} = \frac{X}{1+X},$$

where

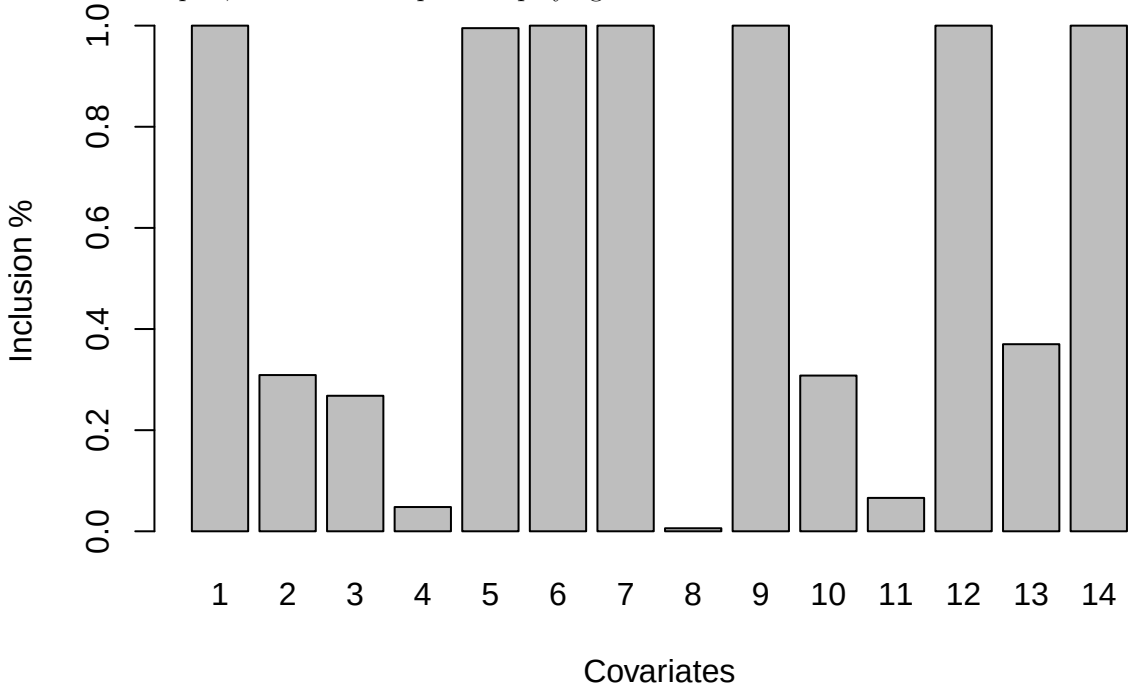
$$X := \frac{P(\gamma_j = 1|y, X, \gamma_{z-j})}{P(\gamma_j = 0|y, X, \gamma_{z-j})} = \frac{P(\gamma_j = 1)}{P(\gamma_j = 0)} \times \frac{p(y|X, \gamma_{z-j}.\gamma_j = 1)}{p(y|X, \gamma_{z-j}, \gamma_j = 0)},$$

So I'll just have to draw from a Bernoulli(p_j).

```
data("BostonHousing")
y <- BostonHousing$medv
X <- cbind(1,BostonHousing[, -which(colnames(BostonHousing) == "medv")])
X[, "chas"] <- as.numeric(X[, "chas"])
X <- as.matrix(X)
lpy.X <- function(y,X, g=0.001) {
   n <- nrow(X)
   p <- ncol(X)
   if(is.null(p)){X=as.matrix(X);n=dim(X)[1] ; p=dim(X)[2]}
   H <- 0
   if(p>1){
      H <- 1/(g+1) * X%*%solve(t(X)%*%X)%*%t(X)
   }
   RSS<- t(y)%*%(diag(n)  - H) %*%y
   return((-n+p)/2*log(2*pi)-p/2*log(g+1)-
            1/2*log(det(t(X)%*%X))-lgamma(n/2-1)-n/2*log(RSS/2))
}
p <- ncol(X)
z <- rep(1,p)
lpy.current <- lpy.proposed <- lpy.X(y,X[, z==1])
S <- 1000
Z <- matrix(0, S, ncol(X))
for(s in 1:S){
   for(j in sample(2:p)){
      z.propose <- z
      # if model includes covariate j, propose removing
      # else if model doesn't include, propose including
      z.propose[j] <- 1 - z.propose[j]
      # log-likelihood of proposed
      lpy.proposed <- lpy.X(y, X[, z.propose==1])
      # log-likelihood will always be less if proposing to include
      # so correct order by multiplying by -1
      r <- (lpy.proposed - lpy.current)*(-1)^(z.propose[j]==0)
      # sample from bernoulli with probability p=exp(-r)
      # to adjust for log scale
      z[j] <- rbernoulli(1,1/(1+exp(-r)))
      if(z[j] == z.propose[j]){
         lpy.current <- lpy.proposed
      }
```

```
   }
   Z[s,] <- z
   }
inclusion_probs <- colMeans(Z[1:(1*S),])
```

For 1000 samples, here's a barplot displaying the number of times a covariate was included.

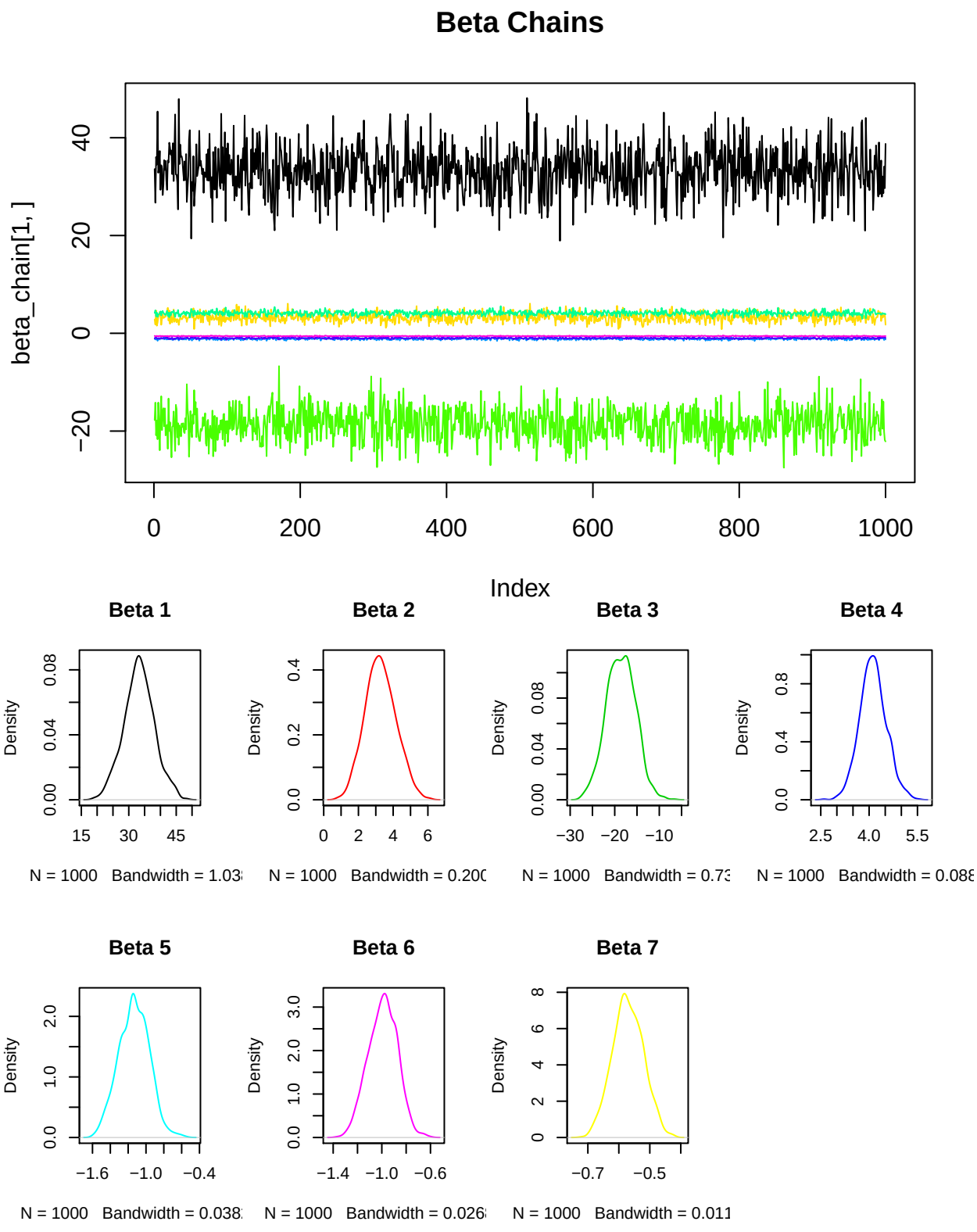


Part (c).

For the MCMC implementations in (a) and (b), produce plots for the posterior distribution of the regression coefficients, when the median model is selected (select covariates with marginal inclusion probability above 0.5), and when we average over all explored models.

```
median_model <- which(inclusion_probs >= 0.5)
X_med <- X[, median_model]
X_medtX_med_inv <- solve(t(X_med)%*%X_med)
XtXinv <- solve(t(X)%*%X)
n <- length(y)
S <- 1000
beta_chain <- matrix(0, nrow = ncol(X_med), ncol = S)
beta_chain_avg <- matrix(0, nrow = ncol(X), ncol = S)
g = 0.001
for(i in 1:S){
   #Sampling sigma2
   sigma2 <- rinvgamma(n = 1, shape = n/2 - 1,
                       rate = 1/2*t(y)%*%
                        (diag(n) - 1/(1 + g)*X_med%*%X_medtX_med_inv%*%
                            t(X_med))%*%y)
   M = sigma2/(1 + g)*X_medtX_med_inv
   m = 1/sigma2*t(X_med)%*%y
   beta_chain[, i] <- t(as.matrix(rmvnorm(n = 1, mean = M%*%m, sigma = M)))

   sigma2 <- rinvgamma(n = 1, shape = n/2 - 1,
                       rate = 1/2*t(y)%*%
                        (diag(n) - 1/(1 + g)*X%*%XtXinv%*%
                            t(X))%*%y)
   M <- sigma2/(1 + g)*XtXinv
   m <- 1/sigma2*t(X)%*%y
   beta_chain_avg[, i] <- t(as.matrix(rmvnorm(n = 1, mean = M%*%m, sigma = M)))
}
```



It's really interesting how the median model in stochastic variable selection leaves out essentially every small coefficient. I've made a table comparing the median model along with the set of averaged coefficients and LASSO.

	Median	Average	LASSO
(Intercept)	33.258	33.722	32.187
crim	NA	-0.108	-0.101
zn	NA	0.046	0.042
indus	NA	0.018	0.000
chas	3.277	2.659	2.694
nox	-18.602	-17.637	-16.563
rm	4.122	3.804	3.852
age	NA	0.000	0.000
dis	-1.135	-1.476	-1.419
rad	NA	0.303	0.264
tax	NA	-0.012	-0.010
ptratio	-0.992	-0.954	-0.934
b	NA	0.009	0.009
lstat	-0.570	-0.524	-0.523

Part (d).

Split the data into a training $D_0 = (Y_0, X_0)$ set and a test set $D_1 = (Y_1, X_1)$ (including approximately one third of the sample). Train your model on D_0 and produce a Monte Carlo sample from the predictive distribution $p(Y_1|X_1)$ associated with covariate information in the test set X_1. Describe how $p(Y_1|X_1)$ when related to the test data Y_1, may be used to provide formal probabilistic understanding of predictive performance in terms of accuracy and uncertainty.

For this part, I'll sample from the posterior predictive distribution for each testing data point through:

1. Sample a model from $p(\gamma|y)$
2. Sample $\sigma^2|y$ and $\beta|y$ given γ
3. Sample $\hat{y} = X_{test}\beta$.

This gives me a way to quantify uncertainty about each prediction in that I'm accounting for both model uncertainty and parameter uncertainty. For an ultimate point estimate, I can use BMA. just need to average over the sampled β values and take $\hat{y} = X_{\text{test}}\beta_{\text{bma}}$.

```
train_ind = sample(seq_len(nrow(X)),size = floor(0.75*nrow(X)) )
trainX = X[train_ind,]; testX = X[-train_ind,]
trainy = y[train_ind]; testy = y[-train_ind]
S <- 500
beta_chain <- matrix(0, nrow = p, ncol = S)
beta <- rep(0, p)
predictions <- matrix(0, nrow = length(testy), ncol = S)
for(j in 1:S){
   # sample gamma
   gamma <- Z[j, ]
   X <- trainX[, which(gamma == 1)]
   y <- trainy
   sigma2 <- rinvgamma(n = 1, shape = nrow(X)/2 - 1,
                       rate = 1/2*t(y)%*%
                        (diag(nrow(X)) - 1/(1 + g)*X%*%solve(t(X)%*%X)%*%
                            t(X))%*%y)
   M <- sigma2/(1 + g)*solve(t(X)%*%X)
   m <- 1/sigma2*t(X)%*%y
   beta[which(gamma==1)] <- t(as.matrix(rmvnorm(n = 1, mean = M%*%m, sigma = M)))
   beta_chain[, j] <- beta
   predictions[, j] <- testX %*% beta
}
```

BMA.RMSE	Lasso.RMSE
4.666	4.253

It's expected that the BMA estimate has worse predictive capicity, since the highest probability models have 5 less covariates than the cross-validated lasso estimate, which was optimizing for predictive performance.