Homework 1

Ian Frankenburg

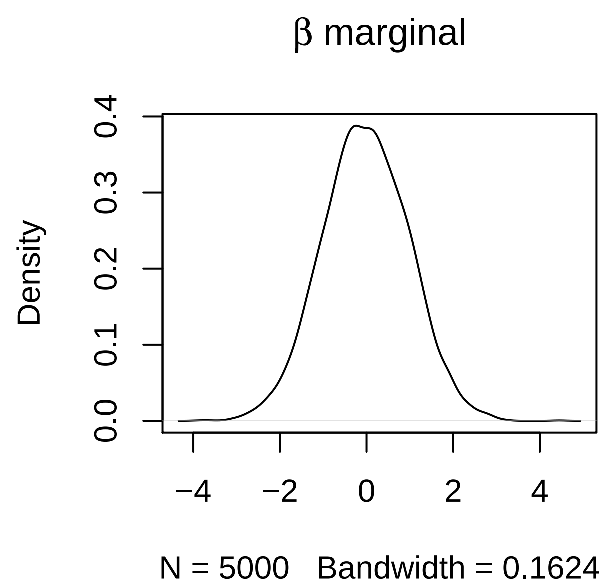
4/7/2020

Bayesian Adaptive Lasso

Part a.

Consider $p = 1$. Simulate 5,000 Monte Carlo samples from the conditional prior $\beta|\tau^2 = 1$ and obtain a plot of the density using the R function density.

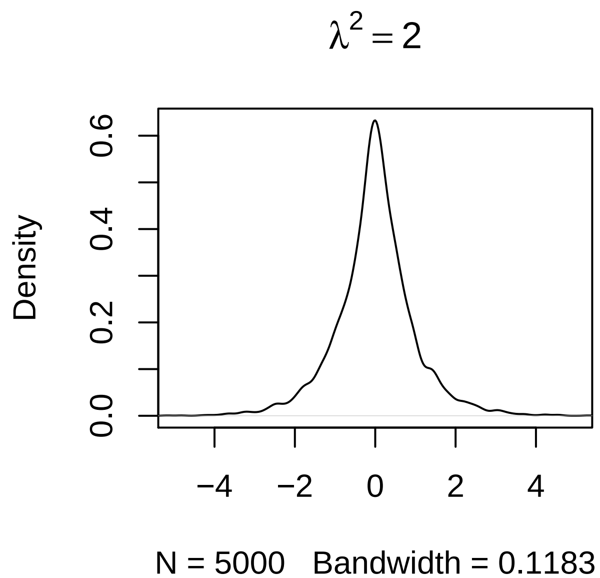
```
n <- 5000
plot(density(rnorm(n,0,1)), main=TeX(paste("$\\beta$", "marginal")))
```



Part b.

Consider $p = 1$. Simulate 5,000 Monte Carlo samples from the marginal prior $\boldsymbol{\beta}$, considering $\lambda^2 = 2$, so that $\mathbb{E}(\tau^2|\lambda) = 1$. Obtain a plot of the density as in **a.**

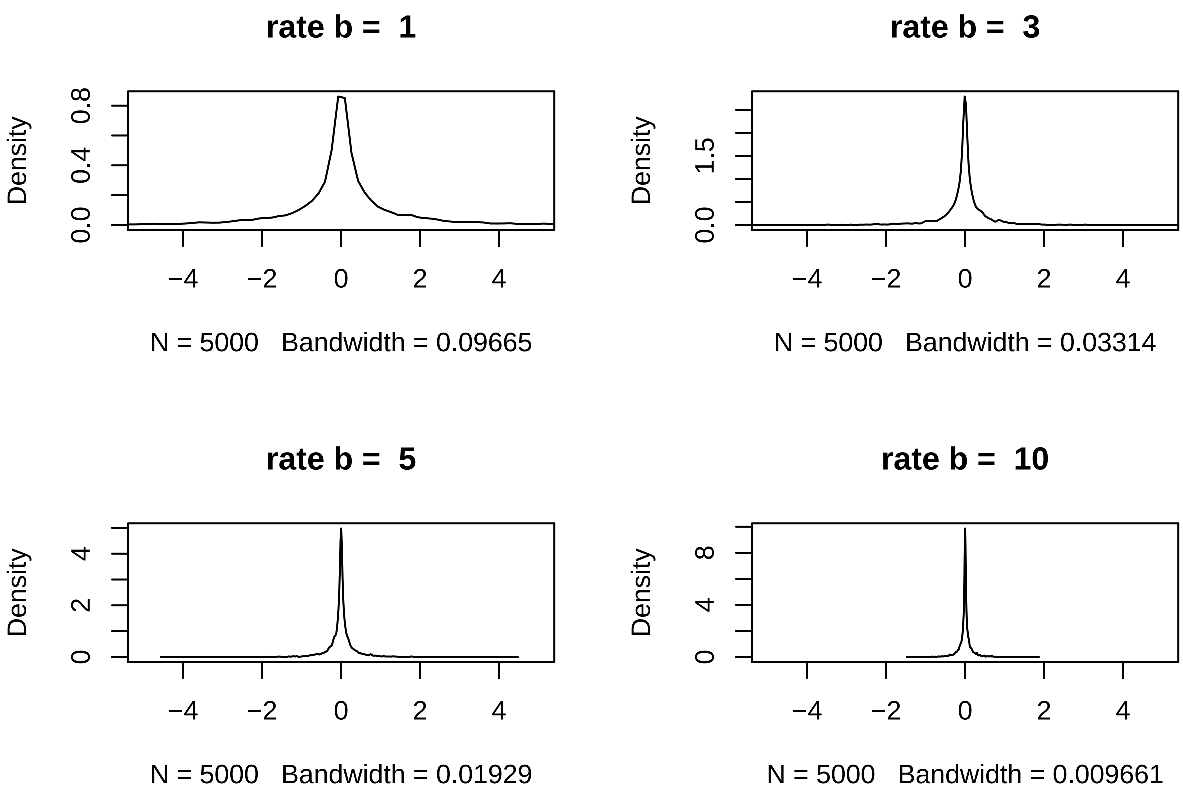
```r
lambda <- sqrt(2)
tau.sq <- rgamma(n,shape=1,rate = lambda^2/2)
beta.marginal <- rnorm(n,0,sqrt(tau.sq))
plot(density(beta.marginal), main=TeX(paste("$\\lambda^2 = 2$")), xlim=c(-5,5))
```



Part c.

Consider $p = 1$. Add a hyper prior on $\gamma = 1/\gamma \sim Gamma(a, rate = b)$. Assess how the marginal prior of $\boldsymbol{\beta}$ changes for $a = 1$ and values of $b \geq 1$.

```r
set.seed(1)
par(mfrow=c(2,2))
rates <- c(1,3,5,10)
for(b in rates){
  lambda <- 1/rgamma(n,1,b)
  tau.sq <- rgamma(n,shape=1,rate = lambda^2/2)
  beta.marginal <- rnorm(n,0,sqrt(tau.sq))
  plot(density(beta.marginal), main=paste("rate b = ",b),xlim=c(-5,5))
}
```



Part d.

Considering the hyper prior in **c.**, describe a Markov Chain Monte Carlo algorithm to sample from the posterior distribution of $\boldsymbol{\beta}$ and σ^2.

I will implement a joint Gibbs and Metropolis sampler. The model is

$$\begin{aligned}
\boldsymbol{y}|\boldsymbol{\beta},\sigma^2 &\sim N(\boldsymbol{X}\boldsymbol{\beta},\sigma^2\boldsymbol{I})\\
\beta_j|\tau_j^2 &\sim N(0,\tau_j^2)\\
\tau_j^2 &\sim \text{Inverse-Gamma}(1,\frac{\lambda^2}{2})\\
\lambda &\sim \text{Inverse-Gamma}(a,1/b)\\
\sigma^2 &\sim \text{Inverse-Gamma}(0.1,0.1).
\end{aligned}$$

I need the full conditionals

$$\begin{gathered}
\{\beta_1,\dots,\beta_p|\boldsymbol{y},\sigma^2,\tau_1^2,\dots,\tau_p^2,\lambda\},\\
\{\sigma^2|\boldsymbol{y},\beta_1,\dots,\beta_p,\tau_1^2,\dots,\tau_p^2,\lambda\},\\
\{\tau_1^2,\dots,\tau_p^2|\boldsymbol{y},\beta_1,\dots,\beta_p,\sigma^2,\lambda\},\\
\{\lambda|\boldsymbol{y},\beta_1,\dots,\beta_p,\sigma^2,\tau_1^2,\dots,\tau_p^2\}
\end{gathered}$$

which are all proportional to

$$p(\boldsymbol{y}|\beta_1,\dots,\beta_p,\tau_1^2,\dots,\tau_p^2,\sigma^2,\lambda)\times p(\beta_1,\dots,\beta_p|\tau_1^2,\dots,\tau_p^2)\times p(\tau_1^2,\dots,\tau_p^2|\lambda)p(\lambda)p(\sigma^2)$$

so I'll start with the posterior

$$\begin{aligned}
p(\beta_1,\dots,\beta_p,\tau_1^2,\dots,\tau_p^2,\sigma^2,\lambda|\boldsymbol{y}) \propto\ & p(\boldsymbol{y}|\beta_1,\dots,\beta_p,\tau_1^2,\dots,\tau_p^2,\sigma^2,\lambda)\\
&\times p(\beta_1,\dots,\beta_p|\tau_1^2,\dots,\tau_p^2)\\
&\times p(\tau_1^2,\dots,\tau_p^2|\lambda)p(\lambda)p(\sigma^2).
\end{aligned}$$

As a function of just σ^2, this is proportional to

$$\begin{aligned}
& p(\boldsymbol{y}|\beta_1,\dots,\beta_p,\tau_1^2,\dots,\tau_p^2,\sigma^2,\lambda)p(\sigma^2)\\
=\ & N(\boldsymbol{X}\boldsymbol{\beta},\sigma^2\boldsymbol{I})IG(0.1,0.1).
\end{aligned}$$

Time to show this is inverse-gamma distributed.

$$\begin{aligned}
& N(\boldsymbol{y};\boldsymbol{X}\boldsymbol{\beta},\sigma^2\boldsymbol{I})IG(\sigma^2;q,r)\\
\propto\ & (\sigma^2)^{-n/2}\exp\big\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\big\}(\sigma^2)^{q-1}\exp\big\{-\frac{r}{\sigma^2}\big\}\\
=\ & (\sigma^2)^{-(n/2+q)-1}\exp\big\{-\frac{1}{\sigma^2}(r+\frac{1}{2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\big\}\\
=\ & IG(n/2+q,r+(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})/2)|_{q=0.1,r=0.1}
\end{aligned}$$

As a function of $\boldsymbol{\beta}$, the conditional is proportional to

$$\begin{aligned}
& p(\boldsymbol{y}|\beta_1,\dots,\beta_p,\tau_1^2,\dots,\tau_p^2,\sigma^2,\lambda)p(\beta_1,\dots,\beta_p|\tau_1^2,\dots,\tau_p^2)\\
=\ & N(\boldsymbol{X}\boldsymbol{\beta},\sigma^2\boldsymbol{I})\cdot\prod_{i=1}^p N(0,\tau_i^2)\\
=\ & N(\boldsymbol{X}\boldsymbol{\beta},\boldsymbol{\Sigma})\cdot N(0,\boldsymbol{\Omega}),\ \text{ where } \Omega=\text{diag}(\tau_1^2,\dots,\tau_p^2)\\
=\ & N(\boldsymbol{m},\boldsymbol{M})
\end{aligned}$$

because the posterior is determined by the quadratic form

$$(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^\top\Sigma^{-1}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})+\boldsymbol{\beta}^\top\Omega^{-1}\boldsymbol{\beta}=(\boldsymbol{\beta}-\boldsymbol{m})^\top\boldsymbol{M}^{-1}(\boldsymbol{\beta}-\boldsymbol{m}).$$

Completing the square gives $\boldsymbol{m}=\boldsymbol{M}\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{y}$ and $\boldsymbol{M}=(\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{X}+\Omega^{-1})^{-1}$.

As a function of $\tau_1^2,\dots,\tau_p^2$, the target is proportional to

$$\begin{aligned}&p(\beta_1,\dots,\beta_p|\tau_1^2,\dots,\tau_p^2)p(\tau_1^2,\dots,\tau_p^2|\lambda)\\ =\;&\prod_{i=1}^p N(\beta_i;0,\tau_i^2)\cdot\prod_{i=1}^p IG(\tau_i^2;1,\frac{\lambda^2}{2})\end{aligned}$$

Finally, as a function of λ, the target distribution is proportional to

$$\begin{aligned}&p(\tau_1^2,\dots,\tau_p^2|\lambda)p(\lambda)\\ =\;&\prod_{i=1}^p IG(\tau_i^2;1,\frac{\lambda^2}{2})\cdot IG(\lambda;a,b)\end{aligned}$$

Now I can build an algorithm to iteratively update through these target distributions. I take the starting value of $\boldsymbol{\beta}^{(0)}$ to be the least-squares solution $\hat{\boldsymbol{\beta}}$ along with $\sigma^{2(0)}=\hat{\sigma}^2$, the MLE for σ^2.

Result: Samples from joint posterior $p(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y})$

for *s in # samples* **do**

note: extra term due in *logr* due to Jacobian of transformation

$\lambda^* \leftarrow \exp(\log(\lambda^{(s)}) + \varepsilon), \quad \varepsilon \sim N(0, \delta^2)$

$logr \leftarrow \log \pi_\lambda(\lambda^*) - \log \pi_\lambda \lambda^{(s)} + \log \lambda^* - \log \lambda^{(s)}$

if $(\log unif(0,1) < logr)$ **then**

$\lambda^{(s+1)} \leftarrow \lambda^*$

else

$\lambda^{(s+1)} \leftarrow \lambda^{(s)}$

end

for *j in 1:p* **do**

note: extra term due in *logr* due to Jacobian of transformation

$\tau_j^{2*} \leftarrow \exp(\log(\tau_j^{2(s)}) + \varepsilon), \quad \varepsilon \sim N(0, \delta^2)$

$logr \leftarrow \log \pi_{\tau_j^2}(\tau_j^{2*}) - \log \pi_{\tau_j^2}(\tau_j^{2(s)}) + \log(\tau_j^{2*}) - \log(\tau_j^{2(s)})$

if $(\log unif(0,1) < logr)$ **then**

$\tau_j^{2(s+1)} \leftarrow \tau_j^{2*}$

else

$\tau_j^{2(s+1)} \leftarrow \tau_j^{2(s)}$

end

end

$\sigma^{2(s+1)} \sim IG(n/2 + a, 2b + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{(s)})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{(s)})/2)$

$\boldsymbol{\beta}^{(s+1)} \sim N(\boldsymbol{m}, \boldsymbol{M})$, where

$\boldsymbol{M} = (\boldsymbol{X}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{X} + \boldsymbol{\Omega}^{-1})^{-1}$ and $\boldsymbol{m} = \boldsymbol{M}(\boldsymbol{X}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{y})$

$\boldsymbol{\Sigma} = \sigma^{2(s+1)}$ and $\boldsymbol{\Omega} = \mathrm{diag}(\tau_1^{2(s+1)}, \ldots, \tau_p^{2(s+1)})$

end

Part f.

Implement such algorithm in R and compare your results with estimates obtained using **glmnet()**. In particular, you should test your results on the diabetes data available from lars, (use the matrix of predictors x).

```r
## Data processing
sourceCpp("helperFunctions.cpp")
set.seed(1)
data("diabetes")
X <- cbind(rep(1,length(diabetes$x)),cbind(diabetes$x)); y <- diabetes$y;
n <- nrow(X); p <- ncol(X); samples <- 1000

## Initialize starting values
lambda <- 1
tau2 <- rep(1000,p)
beta <- solve(t(X)%*%X)%*%t(X)%*%y
sigma2 <- t(y-X%*%beta)%*%(y-X%*%beta)/n
sigma2.chain <- lambda.chain <- rep(0,samples)
beta.chain <- tau2.chain <- matrix(0,nrow=p,ncol=samples)

## MCMC
for(s in 2:samples){
  lambda <- lambdaDraw(current=lambda,tau2=tau2,a=1,b=1)
  tau2 <- tau2Draw(current=tau2,beta=beta,lambda=lambda)
  sigma2 <- sigma2Draw(beta, y, X)
  mM <- betaMeanCov(y,X,sigma2,tau2)
  beta <- t(rmvnorm(n=1,mean=mM$mean,sigma=mM$cov))
  lambda.chain[s] <- lambda
  sigma2.chain[s] <- sigma2
  beta.chain[,s] <- beta
  tau2.chain[,s] <- tau2
}
# Examine markov chains
# plot(beta.chain[1,floor(samples/4):samples],type="l")

# Plot table of coefficients from Glmnet and Bayesian Lasso
comparison <- data.frame(
  "Bayesian Lasso" = rowMeans(beta.chain[,floor(samples/4):samples]),
  "Glmnet" = matrix(coef(glmnet(y=y,x=X),alpha=1,s=1)[-2])
)
kable(comparison, "latex", booktabs = T)
```

Bayesian.Lasso	Glmnet
152.0653282	152.13348
0.0986925	0.00000
-146.5473735	-195.89915
515.9450575	522.05142
267.6553181	296.18834
-64.7222100	-101.86185
-36.1819103	0.00000
-170.9613979	-223.22347
59.2924739	0.00000
468.2759684	513.57366
50.7368620	53.86052

- I initially notice the difference in parameterization between glment's lasso and my Bayesian lasso. I'm viewing the coefficients for $\lambda = 1$ in glmnet and a value of $b = 1$ for Bayesian lasso. As I'll show later, in this implementation of Bayesian lasso, shrinkage is very sensitive to the hyperparameter b of λ.

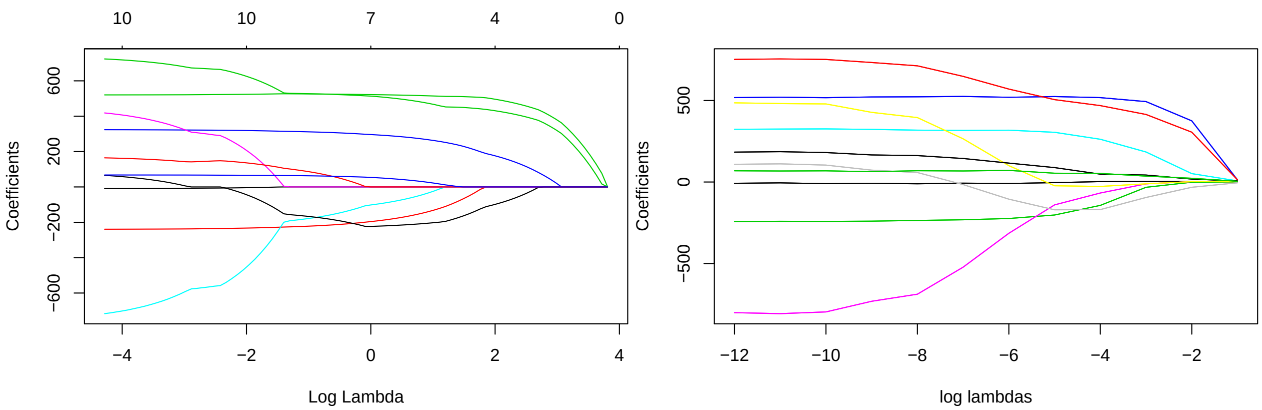
Part g.

Free λ and carry out a sensitivity analysis assessing the behavior of the posterior distribution of $\boldsymbol{\beta}$ and σ^2, as hyper parameters a and b are changed. Explain clearly the rationale you use to assess sensitivity and provide recommendations for the analysis of the diabetes data.

```
# Sequence of lambdas for comparison with glmnet
lambdas <- seq(from=-12, to = -1, length.out = 12)

# Keep track of posterior mean of beta for each fixed lambda
post.means <- matrix(NA,nrow=p,ncol=length(lambdas))

for(i in 1:length(lambdas)){
  for(s in 2:samples){
    lambda <- exp(lambdas[i])
    tau2 <- tau2Draw(current=tau2,beta=beta,lambda=lambda)
    sigma2 <- sigma2Draw(beta, y, X)
    mM <- betaMeanCov(y,X,sigma2,tau2)
    beta <- t(rmvnorm(n=1,mean=mM$mean,sigma=mM$cov))
    sigma2.chain[s] <- sigma2
    beta.chain[,s] <- beta
    tau2.chain[,s] <- tau2
  }
  post.means[,i] <- matrix(rowMeans(beta.chain[,floor(samples/4):samples]),nrow = p)
}
```



- Glmnet is on the left and Bayesian lasso on the right. These regularization paths look very similar.

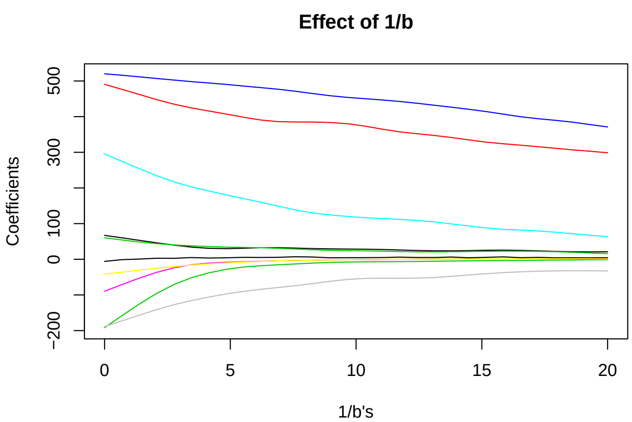
Part g.

Free λ and carry out a sensitivity analysis assessing the behavior of the posterior distribution of $\boldsymbol{\beta}$ and σ^2, as hyper parameters a and b are changed. Explain clearly the rationale you use to assess sensitivity and provide recommendations for the analysis of the diabetes data.

```r
## Sequence of b's that define a path of hyperparameters for lambda
bs <- c(seq(from=1e-5,to=20,length.out = 30))
betasB <- matrix(0,nrow=p,ncol=length(bs))

## Keep track of posterior means for each b value whith lambda free
post.means <- matrix(NA,nrow=p,ncol=length(bs))

for(j in 1:length(bs)){
  for(s in 2:samples){
    lambda <- lambdaDraw(current=lambda,tau2=tau2,a=1,b=1/bs[j])
    tau2 <- tau2Draw(current=tau2,beta=beta, lambda=lambda)
    sigma2 <- sigma2Draw(beta, y, X)
    mM <- betaMeanCov(y,X,sigma2,tau2)
    beta <- t(rmvnorm(n=1,mean=mM$mean,sigma=mM$cov))
    beta.chain[,s] <- beta
  }
betasB[,j] <- rowMeans(beta.chain[,floor(samples/4):samples])
}
```



- Though I didn't include the plots, the shrinkage of coefficients seemed very robust to changes in a for fixed b, so I chose to fix $a = 1$ and focus on varying b. From the plot, I notice as $1/b$ approaches zero, the coefficients approach the least-squares estimates. As 1β increases, there's an increasing amount of shrinkage towards zero. This matches behavior of the regularization paths in part g as *lambda* is fixed and increasing.

Part e.

C++ Helper functions.

```
#include <cmath>
#include <math.h>
#include <random>
#include <RcppArmadillo.h>
using namespace Rcpp;
using namespace std;

// [[Rcpp::depends(RcppArmadillo)]]

double loglambdaTarget(double lambda, vector<double> tau2, double a, double b) {
  return((-a-1)*log(lambda)-(pow(lambda,2)/2*accumulate(tau2.begin(),tau2.end(),0))-1/(b*lambda));
}

double logtau2jTarget(double tau2j, double lambda, double betaj){
  return(-log(sqrt(tau2j)) - 1.0/2.0*(1.0/tau2j*pow(betaj,2) + pow(lambda,2)*tau2j));
}

// [[Rcpp::export]]
double lambdaDraw(double current, vector<double> tau2, double a, double b){
  std::random_device rd;
  std::mt19937 mt(rd());
  std::uniform_real_distribution<double> dist(0, 1.0);
  std::normal_distribution<double> norm(0, current);
  double proposed = exp(log(current)+norm(mt));
  double logr = loglambdaTarget(proposed,tau2,1,b)-
    loglambdaTarget(current,tau2,1,b)+log(proposed)-
    log(current);
  if(log(dist(mt))<logr){
    current = proposed;
  }
  return current;
}

// [[Rcpp::export]]
vector<double> tau2Draw(vector<double> current, vector<double> beta, double lambda){
  std::random_device rd;
  std::mt19937 mt(rd());
  std::uniform_real_distribution<double> dist(0, 1.0);
  std::normal_distribution<double> norm(0, 1);
  for(int j=0;j < current.size(); j++){
    double tau2j_proposed = exp(log(current[j])+norm(mt));
    double logr =
      logtau2jTarget(tau2j_proposed, lambda, beta[j]) -
      logtau2jTarget(current[j], lambda, beta[j]) +
      log(tau2j_proposed)-log(current[j]);
    if(log(dist(mt)) < logr){
      current[j] = tau2j_proposed;
    }
  }
  return current;
```

```
}

// [[Rcpp::export]]
double sigma2Draw(const arma::vec & beta, const arma::vec & y, const arma::mat & X){
  std::random_device rd;
  std::mt19937 mt(rd());
  int n = X.n_rows;
  arma::colvec coef = arma::solve(X, y);
  arma::colvec resid = y - X*coef;
  double rss = arma::as_scalar(arma::trans(resid)*resid);
  std::gamma_distribution<double> gamma(n/2.0+0.1,1.0/(rss/2.0+0.1));
  return 1.0/gamma(mt);
}

// [[Rcpp::export]]
List betaMeanCov(const arma::vec & y, const arma::mat & X, double sigma2, const arma::vec & tau2){
  std::random_device rd;
  std::mt19937 mt(rd());
  int p = X.n_cols;
  arma::mat I = arma::eye(p,p);
  arma::mat tau2_inv = arma::inv(arma::diagmat(tau2));
  arma::mat M = arma::inv(X.t()*X/sigma2+tau2_inv);
  arma::colvec m = M*X.t()*y/sigma2;
  return List::create(Named("cov") = M, Named("mean")= m);
}


// List mcmc(double lambda, double sigma2, const arma::vec & tau2, const arma::vec & beta2){
//
//   return List::create(Named("beta.chain") = M, Named("sigma2.chain")= m,
//                       Named("tau2.chain") = M, Named("lambda.chain") = M);
// }
```