Assignment 10.b for **STATGR6103**

submitted to Professor Andrew Gelman

Advait Rajagopal

16 November 2016

1 Question 1

Consider the following discrete-data regression model: $y_i \sim$ Poisson($e^{X_i\beta}$), i = 1,...,n, with independent Cauchy prior distributions with location 0 and scale 2.5 on the elements of β.

1.1 Part A

Write a program in R to apply the Metropolis algorithm for β given data X, y. Your program should work with any number of predictors (that is, X can be any matrix with the same number of rows as the length of y).
The following information is given to us and a full exposition of the model is useful.

$$p(\beta) \sim \text{Cauchy}(0, 2.5) \tag{1}$$
$$p(y_i|X_i, \beta) \sim \text{Poisson}(e^{X_i\beta}) \tag{2}$$
$$p(\beta|X, y) \propto p(\beta)p(y|X, \beta) \tag{3}$$

Equations 1, 2 and 3 give the prior, likelihood and posterior distributions respectively. If the number of predictors is K and the number of data points is N then the dimensions of y, β and X are as follows;

$$[y] = N \times 1$$
$$[X] = N \times K$$
$$[\beta] = K \times 1$$

The full unnormalized posterior density is given by the following expression;

$$p(\beta|X, y) \propto \prod_{j=1}^{K} \frac{1}{2.5\pi} \frac{1}{\left[1 + \frac{\beta_i}{2.5}^2\right]} \prod_{i=1}^{N} \frac{\exp(X_i\beta)^{y_i} \exp(-\exp(X_i\beta))}{y_i!}$$

In order to avoid computational overflows and underflows, I use logarithms of the posterior densities. My jumping distribution is normal with standard deviation 0.05[1].

1.2 Part B

Simulate fake data from the model for a case with 50 data points and 3 predictors and run your program. Plot the posterior simulations from multiple chains and monitor

[1]See code attached in Section 2

convergence.
I use my data generating function [2] to generate some fake data and Figure 1 shows the histogram of the posterior estimates. The coefficients β are generated randomly as is the data X. I try to ensure that β is not too large so that the exponential parameter of the Poisson distribution does not explode.

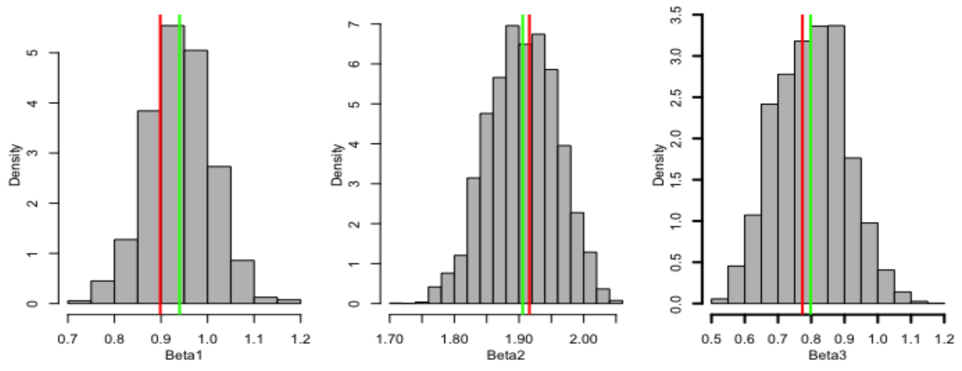

Figure 1: The histogram shows the posterior simulations from the Metropolis algorithm. The red line shows the true value of the parameter and the green line shows the mean of the posterior simulations.

Figure 2 shows the convergence of iterative simulations.

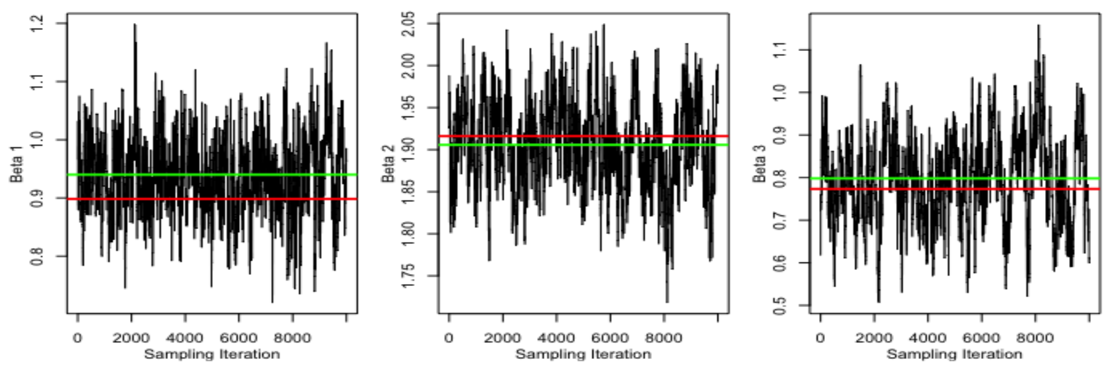

Figure 2: The plot shows the convergence of iterative simulations of the sampling chains. The red line shows the true value and the green line shows the mean value as estimated by the Metropolis algorithm.

1.3 Part C

Fit the model in Stan and check that you get the same results.
I fit a model to the data in Stan and Figure 3 shows the posterior simulations as computed by Stan and my Metropolis algorithm. The two methods yield very similar results.

[2] See code attached in Section 2

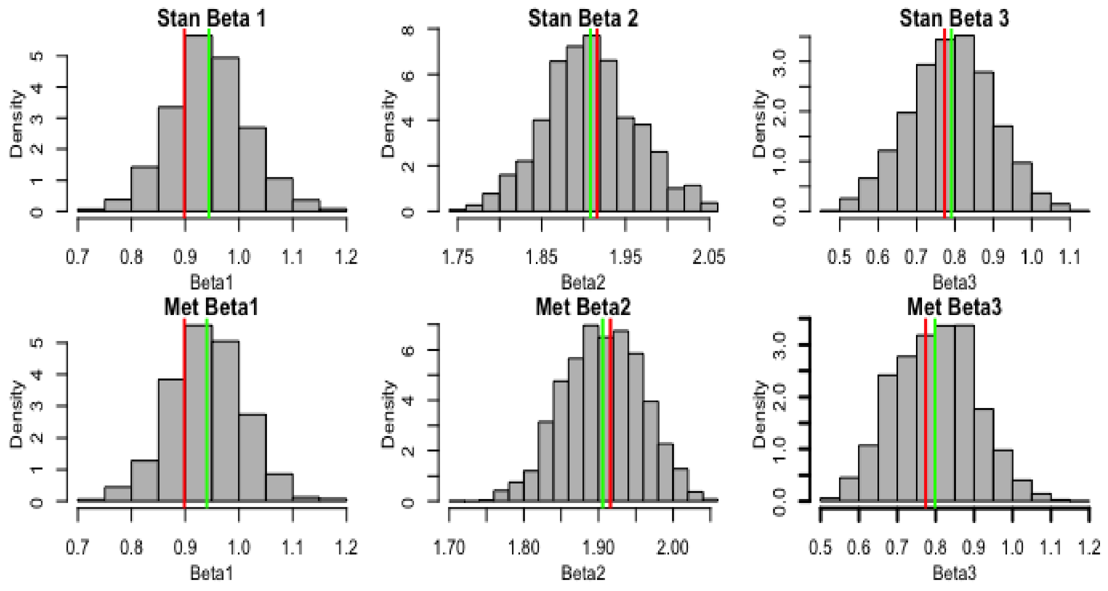

Figure 3: The Metropolis algorithm and Stan estimate the β's very close to each other.

2 Code

Listing 1: R code

```r
##PART A
###
metropolis.func <- function(n.pred, length.data){
  #generate data
  generate <- function(n.pred, length.data){
    beta.mat <<- matrix(data = rnorm(3,1), nrow = n.pred, ncol = 1)
    X.mat <<- matrix(data = NA, nrow = length.data ,ncol = n.pred)
    for(i in 1:length.data){
      for(j in 1:n.pred){
        X.mat[i,j] <<- (rnorm(1))
      }
    }
    y <<- rpois(length.data, exp(X.mat %*% beta.mat))
  }
 #Generate some data
generate(n.pred, length.data)
 ####Prior
prior <- function(param){
  prb <- NULL
  for(i in 1:length(param)){
    prb[i] <- dcauchy(param[i],
                      location = 0,
                      scale = 2.5,
                      log = TRUE)
  }
  return(sum(prb))
}
####Likelihood
likelihood <- function(param){
  pred = exp(X.mat %*% param)
  lk = dpois(y, pred, log = T)
  return(sum(lk))
}
####Posterior
posterior <- function(param){
 return (likelihood(param) + prior(param))
}
####Proposal function
proposalfunction <- function(param){
  return(rnorm(length(param),
               mean = param, sd = rep(0.05,length(param))))
}
```

Listing 2: R code contd.

```
####Metropolis algorithm
run_metropolis_MCMC <- function(startvalue, iterations){
  chain = array(dim = c(iterations+1,length(startvalue)))
  chain[1,] = startvalue
  for (i in 1:iterations){
    proposal = proposalfunction(chain[i,])

    probab = exp(posterior(proposal) - posterior(chain[i,]))
    if (runif(1) < probab){
      chain[i+1,] = proposal
    }else{
      chain[i+1,] = chain[i,]
    }
  }
  return(chain)
}
metropolis.func.fit <<- (run_metropolis_MCMC(
rep(mean(c(beta.mat)),
length(c(beta.mat))), 20000))
warmup <<- 10000
acceptance.rate <- 1 - mean(duplicated(
metropolis.func.fit[-(1:warmup),]))
return(paste("acceptance␣rate␣=",
round(acceptance.rate, digits = 3)))
}

###PART B
metropolis.func(3,50)

#Histograms of parameter estimates
par(mfrow = c(1, 3),
    mar = c(3, 3, 1, 1),
    oma = c(.5, .5, .5, .5),
    mgp = c(2,1,0))
parameters <- c(beta.mat)

hist((metropolis.func.fit[-(1:warmup),1]),
     freq = FALSE,
     xlab = "Beta1",
     col = "grey",
     main = NULL)
abline(v = parameters[1],
       col = "red",
       lwd = 2)
```

Listing 3: R code contd.

```
abline(v = mean(metropolis.func.fit[-(1:warmup),1]),
       col = "green",
       lwd = 2)

hist((metropolis.func.fit[-(1:warmup),2]),
     freq = FALSE,
     xlab = "Beta2",
     col = "grey",
     main = NULL)
abline(v = parameters[2],
       col = "red",
       lwd = 2)
abline(v = mean(metropolis.func.fit[-(1:warmup),2]),
       col = "green",
       lwd = 2)
hist((metropolis.func.fit[-(1:warmup),3]),
     freq = FALSE,
     xlab = "Beta3",
     col = "grey",
     main = NULL,
     lwd = 2)
abline(v = parameters[3],
       col = "red",
       lwd = 2)
abline(v = mean(metropolis.func.fit[-(1:warmup),3]),
       col = "green",
       lwd = 2)
#Check convergence
par(mfrow = c(1, 3),
    mar = c(3, 3, 1, 1),
    oma = c(.5, .5, .5, .5),
    mgp = c(2,1,0))

plot(metropolis.func.fit[-(1:warmup), 1],
     type = "l",
     xlab = "Sampling␣Iteration",
     ylab = "Beta␣1",
     main = NULL)
abline(h = parameters[1],
       col = "red",
       lwd = 2)
abline(h = mean(metropolis.func.fit[-(1:warmup),1]),
       col = "green",
       lwd = 2)
```

Listing 4: R code contd.

```
plot(metropolis.func.fit[-(1:warmup), 2],
     type = "l",
     xlab = "Sampling␣Iteration",
     ylab = "Beta␣2",
     main = NULL)
abline(h = parameters[2],
       col = "red",
       lwd = 2)
abline(h = mean(metropolis.func.fit[-(1:warmup),2]),
       col = "green",
       lwd = 2)
plot(metropolis.func.fit[-(1:warmup), 3],
     type = "l",
     xlab = "Sampling␣Iteration",
     ylab = "Beta␣3",
     main = NULL)
abline(h = parameters[3],
       col = "red",
       lwd = 2)
abline(h = mean(metropolis.func.fit[-(1:warmup),3]),
       col = "green",
       lwd = 2)
###
##PART C
Xmat <- X.mat
y
nr <- 50
nc <- 3

stanc("10b.stan")
fit <- stan("10b.stan",
            data = list("Xmat", "y"),
            iter = 1000,
            chains = 3)
print(fit)
extract(fit)
par(mfrow = c(2, 3),
    mar = c(3, 3, 1, 1),
    oma = c(.5, .5, .5, .5),
    mgp = c(2,1,0))
```

Listing 5: R code contd.

```
hist(extract(fit)$betas[,1],
      freq = FALSE,
      xlab = "Beta1",
      col = "grey",
      main = "Stan␣Beta␣1")
abline(v = parameters[1],
       col = "red",
       lwd = 2)
abline(v = mean(extract(fit)$betas[,1]),
       col = "green",
       lwd = 2)
hist(extract(fit)$betas[,2],
     freq = FALSE,
     xlab = "Beta2",
     col = "grey",
     main = "Stan␣Beta␣2")
abline(v = parameters[2],
       col = "red",
       lwd = 2)
abline(v = mean(extract(fit)$betas[,2]),
       col = "green",
       lwd = 2)
hist(extract(fit)$betas[,3],
     freq = FALSE,
     xlab = "Beta3",
     col = "grey",
     main = "Stan␣Beta␣3")
abline(v = parameters[3],
       col = "red",
       lwd = 2)
abline(v = mean(extract(fit)$betas[,3]),
       col = "green",
       lwd = 2)

####
hist((metropolis.func.fit[-(1:warmup),1]),
     freq = FALSE,
     xlab = "Beta1",
     col = "grey",
     main = "Met␣Beta1")
abline(v = parameters[1],
       col = "red",
       lwd = 2)
```

Listing 6: R code contd.

```
abline(v = mean(metropolis.func.fit[-(1:warmup),1]),
       col = "green",
       lwd = 2)
hist((metropolis.func.fit[-(1:warmup),2]),
     freq = FALSE,
     xlab = "Beta2",
     col = "grey",
     main = "Met␣Beta2")
abline(v = parameters[2],
       col = "red",
       lwd = 2)
abline(v = mean(metropolis.func.fit[-(1:warmup),2]),
       col = "green",
       lwd = 2)
hist((metropolis.func.fit[-(1:warmup),3]),
     freq = FALSE,
     xlab = "Beta3",
     col = "grey",
     main = "Met␣Beta3",
     lwd = 2)
abline(v = parameters[3],
       col = "red",
       lwd = 2)
abline(v = mean(metropolis.func.fit[-(1:warmup),3]),
       col = "green",
       lwd = 2)
```

2.1 Stan Code

Listing 7: Stan code.

```
data{
int nr;
int nc;
matrix[nr, nc] Xmat;
int y[nr];
}
parameters{
vector[nc] betas;
}
model{
 y ~ poisson(exp(Xmat*betas));
  for(i in 1:nc){
    betas[i] ~ cauchy(0, 2.5);
  }
}
```