Bayesian Learning

Lab 4

Emil K Svensson and Rasmus Holm

2017-05-24

Question 1

```r
bid <- read.table("../data/eBayNumberOfBidderData.dat", header = TRUE)
```

a)

```r
glm_res <- glm(nBids ~ . - 1, data = bid, family = poisson(link = "log"))
summary(glm_res)
```

```
##
## Call:
## glm(formula = nBids ~ . - 1, family = poisson(link = "log"),
##     data = bid)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5800  -0.7222  -0.0441   0.5269   2.4605
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## Const        1.07244    0.03077  34.848  < 2e-16 ***
## PowerSeller -0.02054    0.03678  -0.558   0.5765
## VerifyID    -0.39452    0.09243  -4.268 1.97e-05 ***
## Sealed       0.44384    0.05056   8.778  < 2e-16 ***
## Minblem     -0.05220    0.06020  -0.867   0.3859
## MajBlem     -0.22087    0.09144  -2.416   0.0157 *
## LargNeg      0.07067    0.05633   1.255   0.2096
## LogBook     -0.12068    0.02896  -4.166 3.09e-05 ***
## MinBidShare -1.89410    0.07124 -26.588  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 6264.01  on 1000  degrees of freedom
## Residual deviance:  867.47  on  991  degrees of freedom
## AIC: 3610.3
##
## Number of Fisher Scoring iterations: 5
```

b)

```r
library(mvtnorm)

logprior <- function(beta, mu, sigma){
    dmvnorm(beta, mean = mu, sigma = sigma, log = TRUE)
}

loglikelihood <- function(beta, X, Y){
    linear_prediction <- t(X) %*% beta
    probabilities <- Y * linear_prediction - exp(linear_prediction)
    loglike <- sum(probabilities)

    ## if (abs(loglike) == Inf)
    ##     loglike = -20000

    loglike
}

## loglikelihood <- function(beta, X, Y) {
##   linear_prediction <- t(X) %*% beta
##   probs <- dpois(Y , lambda = exp(linear_prediction), log = TRUE)
##   sum(probs)
## }

logposterior <- function(beta, X, Y, prior_mu, prior_sigma){
    loglikelihood(beta, X, Y) + logprior(beta, prior_mu, prior_sigma)
}

X <- as.matrix(bid[,-1])
Y <- as.matrix(bid[,1])

mu <- rep(0, ncol(X))
sigma <- 100 * solve(t(X) %*% X)

normal_res <- optim(par = matrix(rep(0, ncol(X)), ncol = 1),
                 fn = logposterior, method = "BFGS", hessian = TRUE,
                 X = t(X), Y = Y,
                 prior_mu= mu, prior_sigma = sigma,
                 control=list(fnscale=-1))
hessian <- normal_res$hessian
```

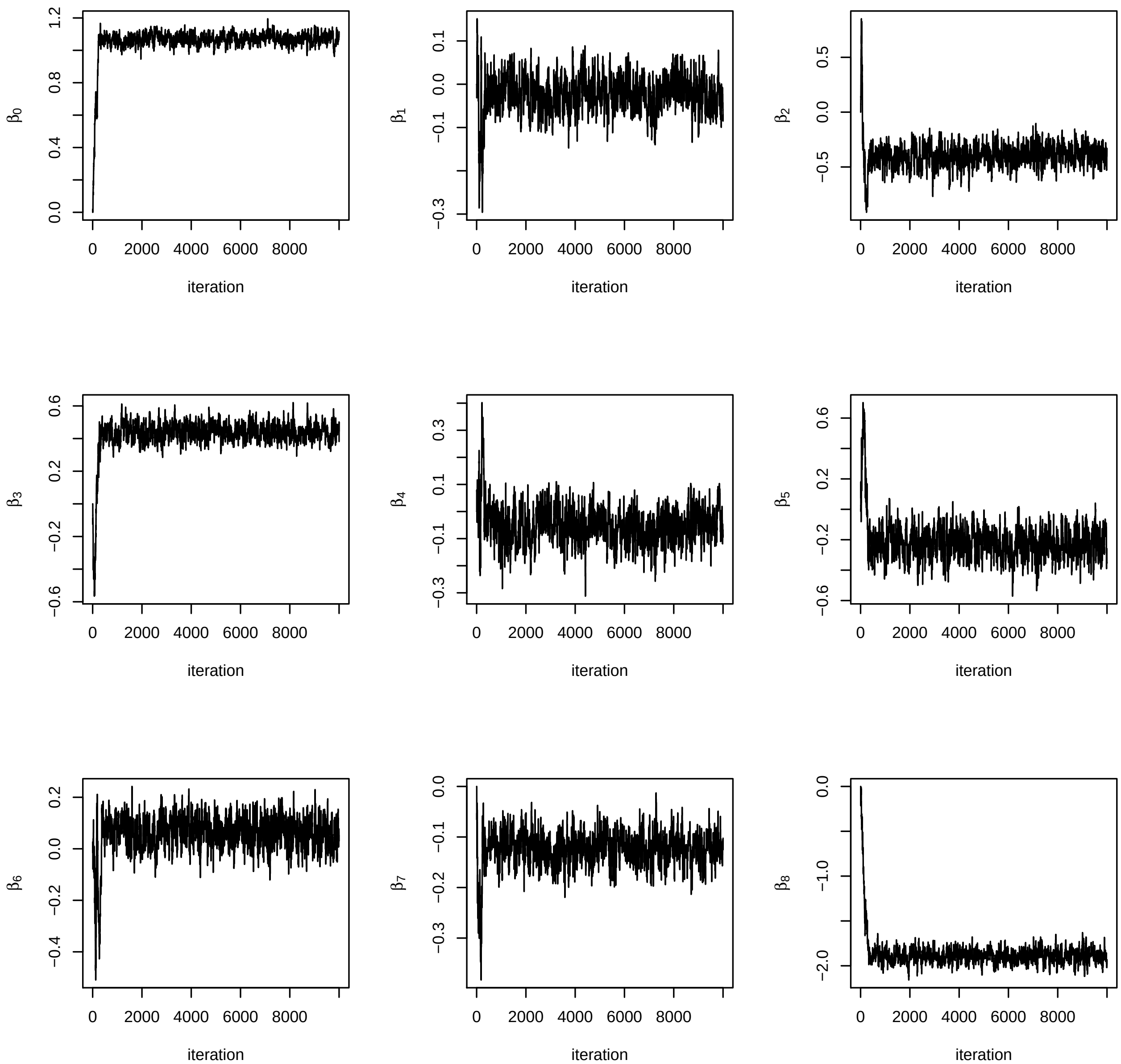
c)

```r
targetdensity <- function(theta, prior_mu, prior_sigma, X, Y, ...) {
    likelihood <- dpois(Y, lambda = exp(t(X) %*% t(theta)), log = TRUE)
    prior <- dmvnorm(theta, mean = prior_mu, sigma = prior_sigma, log = TRUE)
    sum(likelihood) + prior
}

proposaldensity <- function(theta, mu, prop_sigma, ...){
    dmvnorm(theta, mean = mu, sigma = prop_sigma, log = TRUE)
```

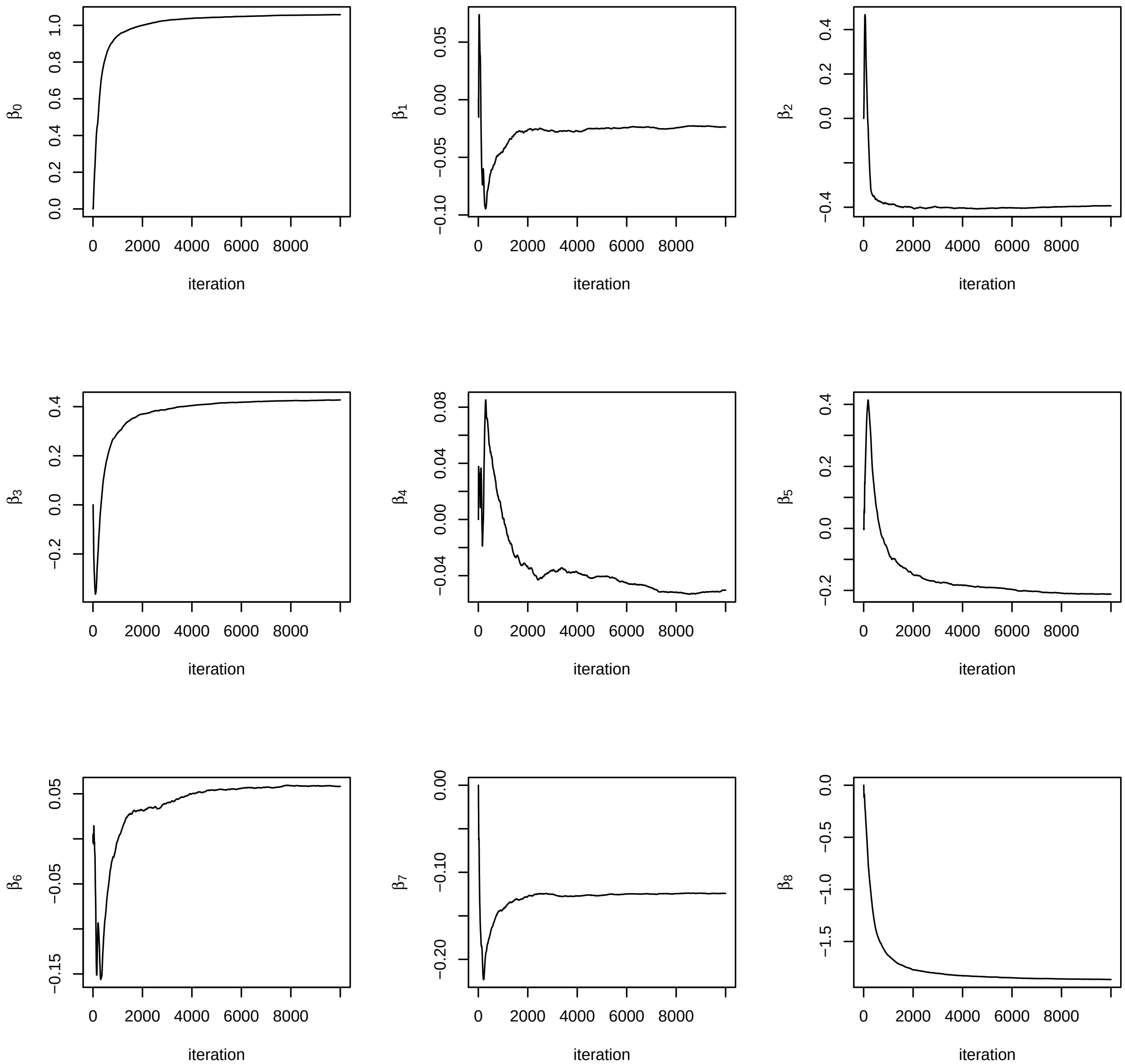
```r
}

proposalsampler <- function(mu, prop_sigma, ...){
    matrix(rmvnorm(1, mean = mu, sigma = prop_sigma), nrow = 1)
}

metropolis_hastings <- function(log_targ_post_func, log_prop_func, prop_sampler,
                                X0, iters, ...){
    x <- X0
    values <- matrix(0, ncol = length(X0), nrow = iters + 1)
    values[1,] <- X0

    alpha <- function(x, y, ...) {
        numerator <- log_targ_post_func(y, ...) + log_prop_func(x, y, ...)
        denominator <- log_targ_post_func(x, ...) + log_prop_func(y, x, ...)
        exp(numerator - denominator)
    }

    for (i in 1:iters) {
        y <- prop_sampler(x, ...)
        u <- runif(1)

        if (u < alpha(x, y, ...)) {
            x <- y
        }

        values[i+1,] <- x
    }

    values
}

iters <- 10000
X0 <- rep(0, times = ncol(X))

params <- list(
    log_targ_post_func = targetdensity,
    log_prop_func = proposaldensity,
    prop_sampler = proposalsampler,
    X0 = matrix(rep(0, times = ncol(X)), nrow = 1),
    iters = iters,
    X = t(X),
    Y = Y,
    prior_mu = rep(0, times = ncol(X)),
    prior_sigma = 100 * solve(t(X) %*% X),
    prop_sigma = 0.6 * -solve(hessian)
)

metro_res <- do.call(metropolis_hastings, params)
```

```
## Warning: package 'knitr' was built under R version 3.3.2
```

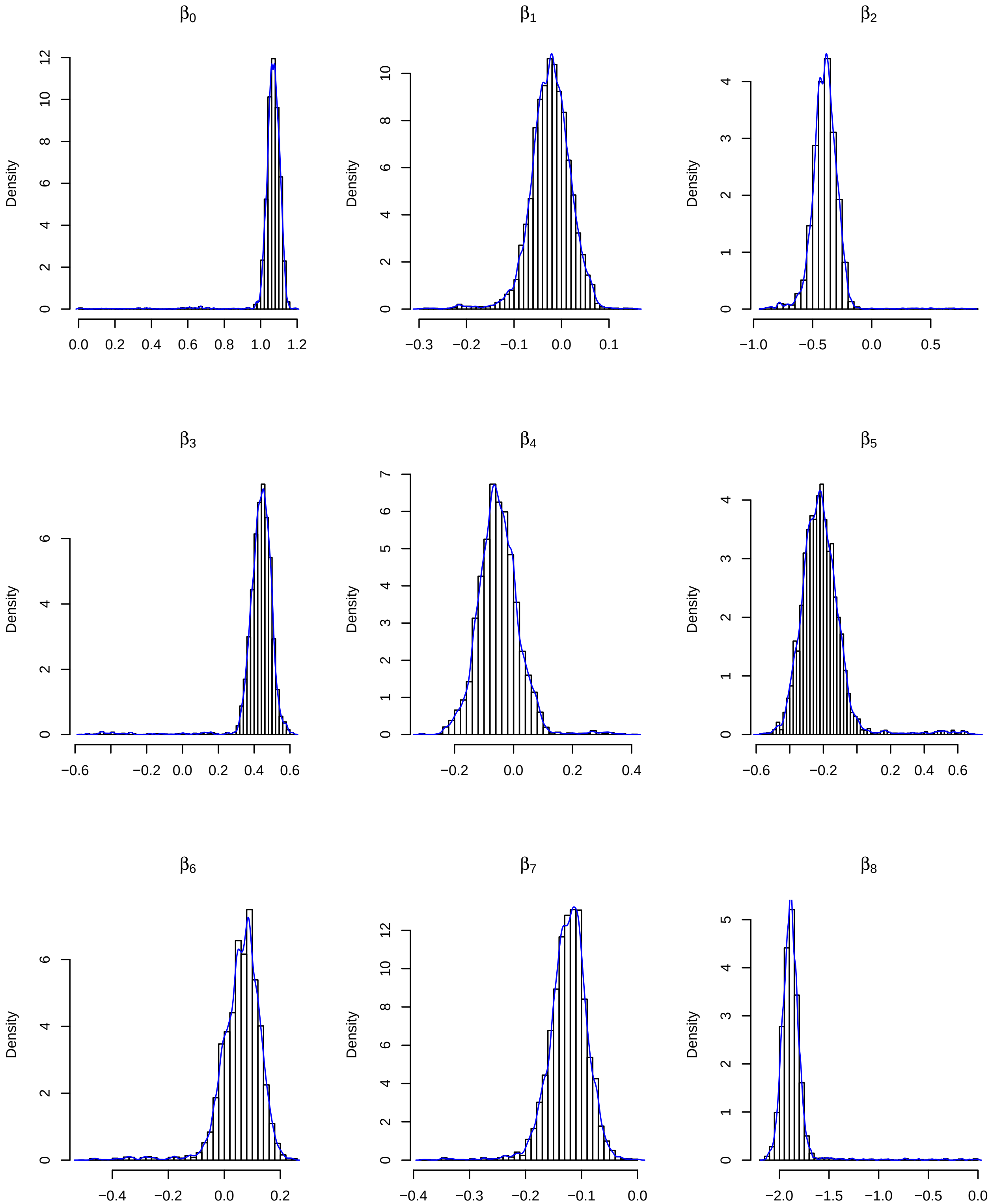
	Beta0	Beta1	Beta2	Beta3	Beta4	Beta5	Beta6	Beta7	Beta8
glm	1.072	-0.021	-0.395	0.444	-0.052	-0.221	0.071	-0.121	-1.894
normal	1.070	-0.021	-0.393	0.444	-0.052	-0.221	0.071	-0.120	-1.892
metropolis	1.058	-0.024	-0.394	0.427	-0.050	-0.212	0.058	-0.124	-1.865



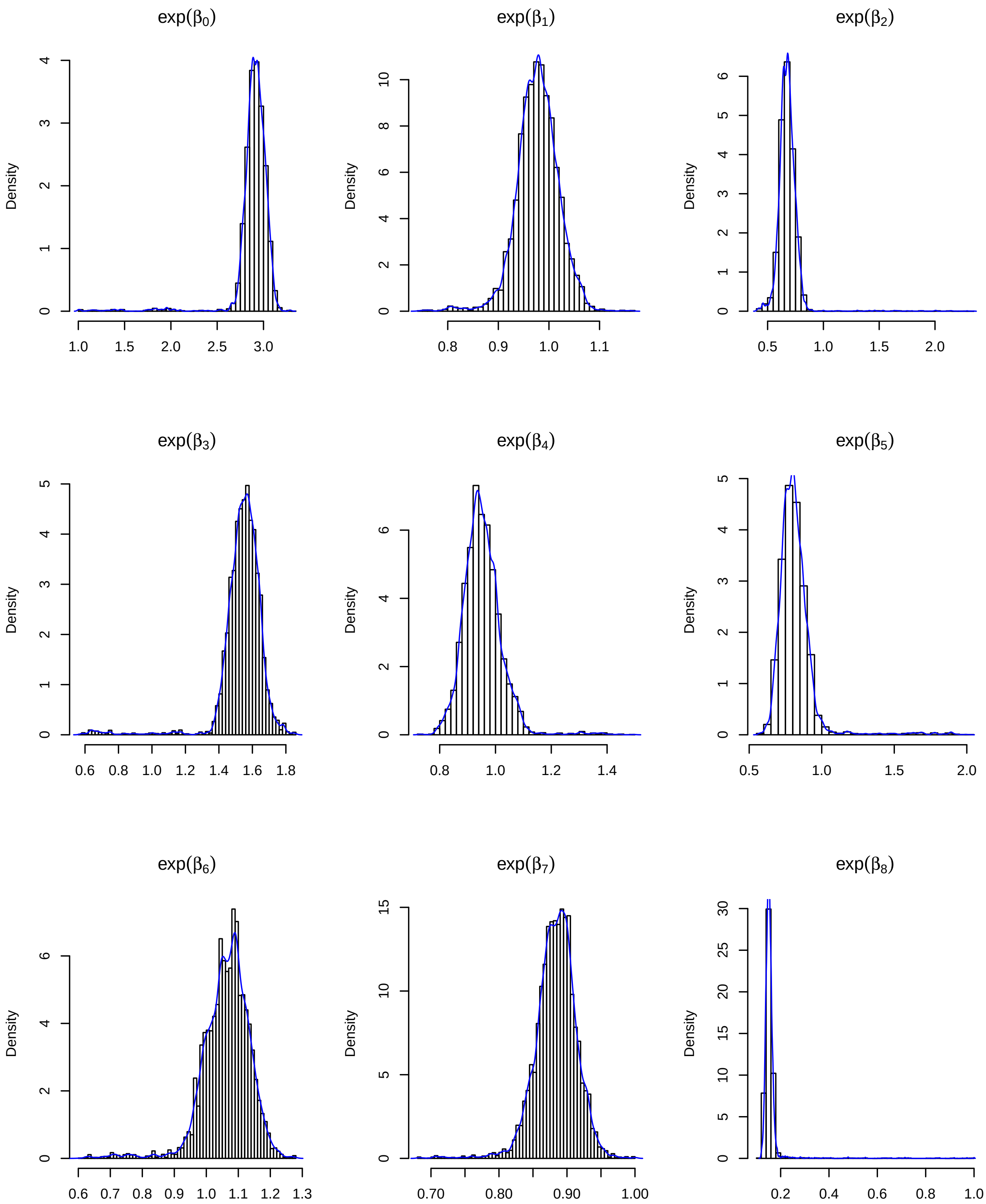
The parameters seem to converge but some of them might have some small autocorrelation left in them. Plots with cumulative means will be assessed.



The cumulative means seem to be quite stable although not completely converged. More iterations are needed to get a more stable result.



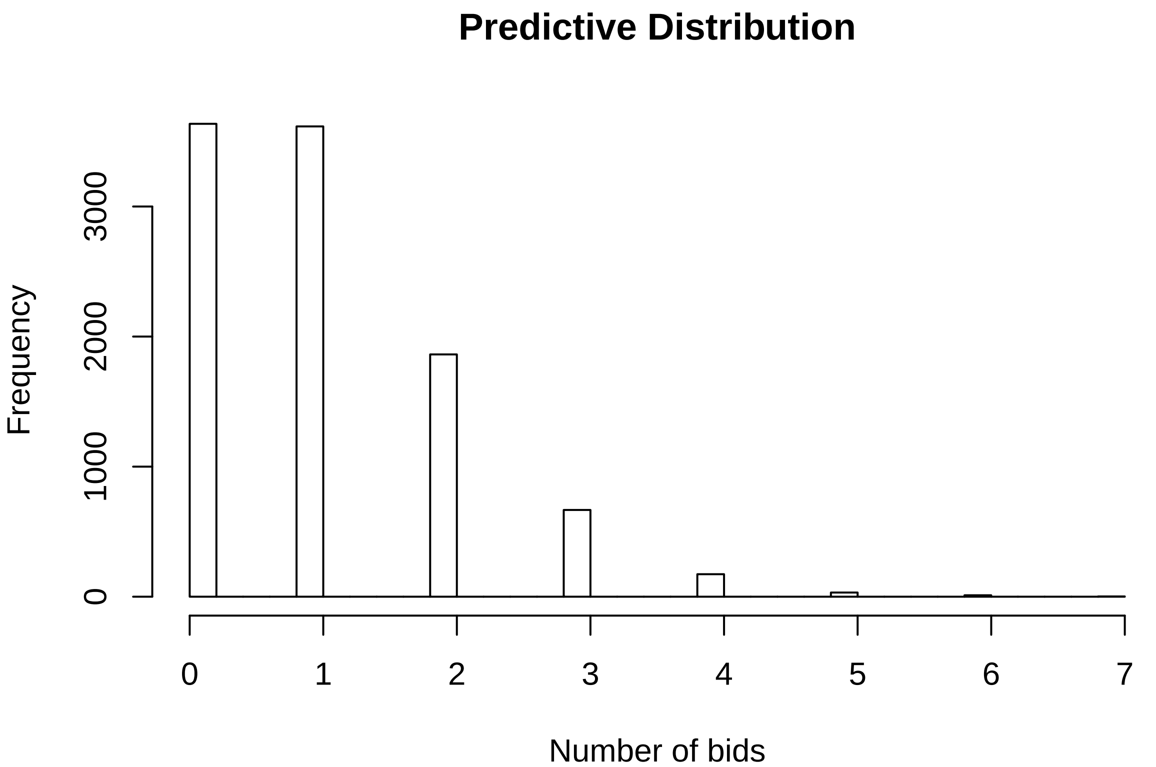
β_0
Density
0 2 4 6 8 10 12
0.0 0.2 0.4 0.6 0.8 1.0 1.2
β_1
Density
0 2 4 6 8 10
−0.3 −0.2 −0.1 0.0 0.1
β_2
Density
0 1 2 3 4
−1.0 −0.5 0.0 0.5
β_3
Density
0 2 4 6
−0.6 −0.2 0.0 0.2 0.4 0.6
β_4
Density
0 1 2 3 4 5 6 7
−0.2 0.0 0.2 0.4
β_5
Density
0 1 2 3 4
−0.6 −0.2 0.2 0.4 0.6
β_6
Density
0 2 4 6
−0.4 −0.2 0.0 0.2
β_7
Density
0 2 4 6 8 10 12
−0.4 −0.3 −0.2 −0.1 0.0
β_8
Density
0 1 2 3 4 5
−2.0 −1.5 −1.0 −0.5 0.0



d

```
Xpred <- matrix(c(1, 1, 1, 1, 0, 0, 0, 1, 0.5), nrow = 1)
predsamples <- rpois(10000, lambda = exp(Xpred %*% t(metro_res)))
```

```r
hist(predsamples, breaks = 50, main="Predictive Distribution", xlab="Number of bids")
```




```r
paste("The probability that there are no bidders", mean(predsamples == 0))
```

```
## [1] "The probability that there are no bidders 0.3636"
```