Lab Report:Lab3-Group8

Udaya Shanker Mohanan Nair(udamo524), Uday Jain(udaja983)

2025-05-19

Assignment 1 : Gibbs samcol_sizeling for the logistic regression

```
##ASSIGNMENT 1

library(mvtnorm)
library(BayesLogit)

disease_data <- read.csv("Disease.csv")
x <- cbind(
  Intercept = 1,
  age_new = (disease_data$age -
               mean(disease_data$age))/sd(disease_data$age),
  gender = disease_data$gender,
  duration_new = (disease_data$duration_of_symptoms -
                    mean(disease_data$duration_of_symptoms))/sd(disease_data$duration_of_symptoms),
  dyspnoea = disease_data$dyspnoea,
  white_Blood_new = (disease_data$white_blood -
                       mean(disease_data$white_blood))/sd(disease_data$white_blood)
)

y <- disease_data$class_of_diagnosis
size <- nrow(x)
col_size <- ncol(x)

tau <- 3


gibbs_implementation <- function(x, y, n_iter=1000) {
  beta_samples <- matrix(0, n_iter, ncol(x))

  beta_value <- rep(0, ncol(x))
  size <- nrow(x)
  col_size <- ncol(x)

  prior_precision <- diag(1/tau^2, col_size)

  for (iter in 1:n_iter) {
    z <- x %*% beta_value
    poly_gamma <- rpg(size, 1, z)
    v <- solve(t(x) %*% diag(poly_gamma) %*% x + prior_precision)
    m <- v %*% t(x) %*% (y - 0.5)
    beta_value <- as.numeric(rmvnorm(1, m, v))
```

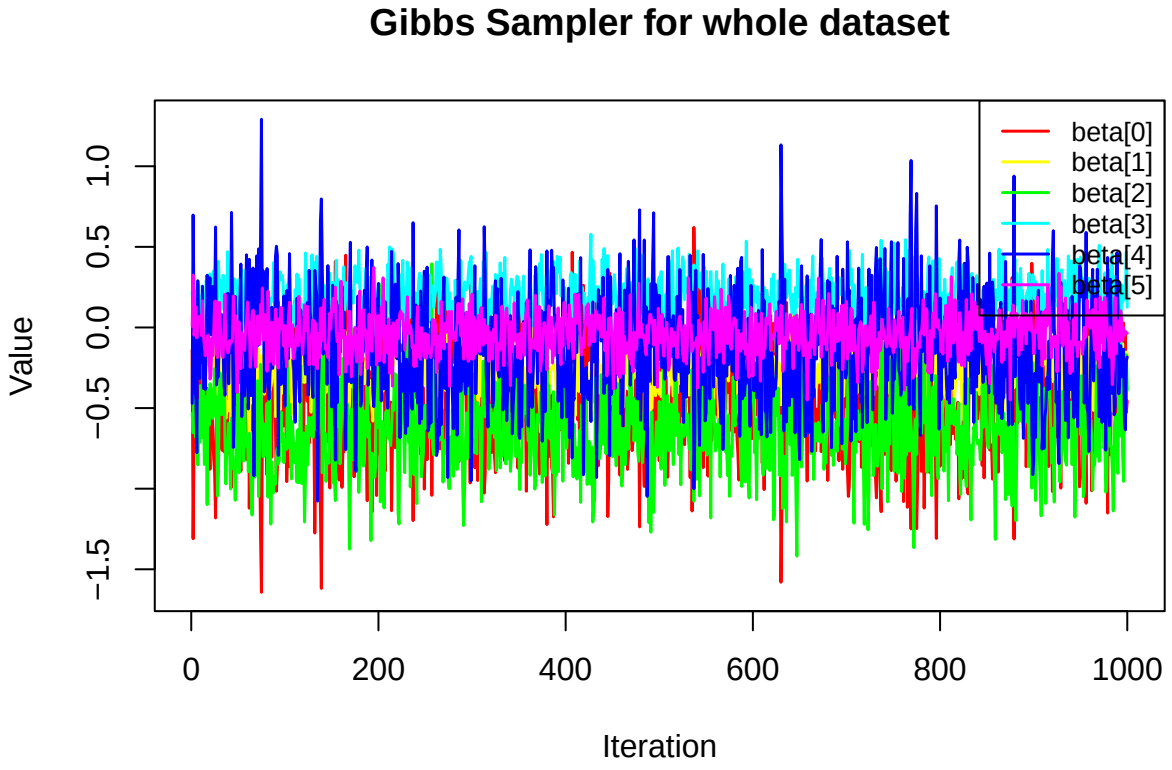
```
    beta_samples[iter,] <- beta_value
  }
  return(beta_samples)
}

set.seed(123)
results <- gibbs_implementation(x, y)

if_evaluation <- function(beta_values) {
  acf_value <- acf(beta_values, plot=FALSE, lag.max=50)$acf[,,1]
  if_value <- 1 + 2 * sum(acf_value[-1])
  return(if_value)
}

res_if <- apply(results, 2, if_evaluation)
cat("Inefficiency Factors for whole dataset:\n")
```

```
## Inefficiency Factors for whole dataset:
```

```
print(res_if)
```

```
## [1] 1.7322117 0.9684822 0.9952840 0.6659212 2.2940092 0.4340299
```

```
plot(1, type = "n",
     xlim = c(1, nrow(results)),
     ylim = range(results),
     xlab = "Iteration",
     ylab = "Value",
     main = "Gibbs Sampler for whole dataset")

cols <- rainbow(col_size)

for (i in 1:col_size) {
  lines(results[, i], col = cols[i], lwd = 1.5)
}
legend("topright",
       legend = paste0("beta[", 0:(col_size-1), "]"),
       col = cols,
       lty = 1,
       lwd = 1.5,
       cex = 0.8)
```




```
results_for_10_observations <- gibbs_implementation(x[1:10,], y[1:10])
results_for_40_observations <- gibbs_implementation(x[1:40,], y[1:40])
results_for_80_observations <- gibbs_implementation(x[1:80,], y[1:80])

res_if_10_observations <- apply(results_for_10_observations, 2, if_evaluation)
cat("Inefficiency Factors for 10 Observations:\n")
```

```
## Inefficiency Factors for 10 Observations:
```

```
print(res_if_10_observations)
```

```
## [1] 0.827587 1.420033 1.373011 2.115056 0.165961 2.412803
```

```
res_if_40_observations <- apply(results_for_40_observations, 2, if_evaluation)
cat("Inefficiency Factors for 40 Observations:\n")
```

```
## Inefficiency Factors for 40 Observations:
```

```
print(res_if_40_observations)
```

```
## [1] 1.8745666 1.0827795 1.9680445 0.2510922 1.7632683 2.3757253
```

```
res_if_80_observations <- apply(results_for_80_observations, 2, if_evaluation)
cat("Inefficiency Factors for 80 Observations:\n")
```

```
## Inefficiency Factors for 80 Observations:
```

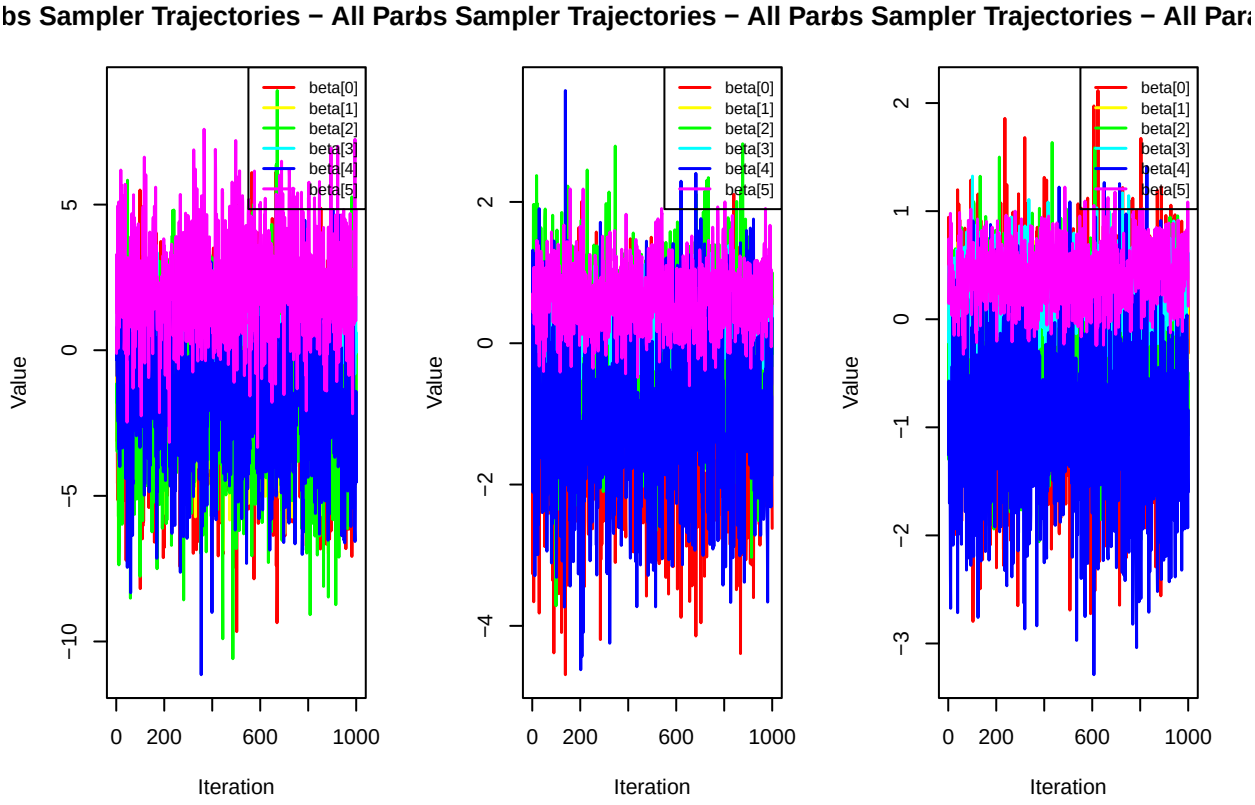
```
print(res_if_80_observations)
```

```
## [1] 1.3885638 1.0347889 1.3911514 0.9197756 1.6944566 3.0431779
```

```
par(mfrow = c(1,3))
#10 Observations
plot(1, type = "n",
```

```
     xlim = c(1, nrow(results_for_10_observations)),
     ylim = range(results_for_10_observations),
     xlab = "Iteration",
     ylab = "Value",
     main = "Gibbs Sampler Trajectories - All Parameters")

cols <- rainbow(col_size)

for (i in 1:col_size) {
  lines(results_for_10_observations[, i], col = cols[i], lwd = 1.5)
}
legend("topright",
       legend = paste0("beta[", 0:(col_size-1), "]"),
       col = cols,
       lty = 1,
       lwd = 1.5,
       cex = 0.8)

#40 Observations

plot(1, type = "n",
     xlim = c(1, nrow(results_for_40_observations)),
     ylim = range(results_for_40_observations),
     xlab = "Iteration",
     ylab = "Value",
     main = "Gibbs Sampler Trajectories - All Parameters")

cols <- rainbow(col_size)

for (i in 1:col_size) {
  lines(results_for_40_observations[, i], col = cols[i], lwd = 1.5)
}
legend("topright",
       legend = paste0("beta[", 0:(col_size-1), "]"),
       col = cols,
       lty = 1,
       lwd = 1.5,
       cex = 0.8)

#80 Observations
plot(1, type = "n",
     xlim = c(1, nrow(results_for_80_observations)),
     ylim = range(results_for_80_observations),
     xlab = "Iteration",
     ylab = "Value",
     main = "Gibbs Sampler Trajectories - All Parameters")

cols <- rainbow(col_size)

for (i in 1:col_size) {
  lines(results_for_80_observations[, i], col = cols[i], lwd = 1.5)
}
legend("topright",
```

```
        legend = paste0("beta[", 0:(col_size-1), "]"),
        col = cols,
        lty = 1,
        lwd = 1.5,
        cex = 0.8)
```




```
par(mfrow = c(1,1))
```

Assignment 2 : Metropolis Random Walk for Poisson regression

```
##ASSIGNMENT 2
library(mvtnorm)
library(MASS)
data<-read.table("eBayNumberOfBidderData_2025.dat",header=TRUE)
X <- model.matrix(~ PowerSeller + VerifyID + Sealed +
                    Minblem + MajBlem + LargNeg + LogBook +
                    MinBidShare, data = data)
y <- data$nBids
################################### Part A ############################
poisson_glm<-glm(y ~ . -1,data=as.data.frame(X),family=poisson)
cat("Summary\n")
```

```
## Summary
```

```
print(summary(poisson_glm))
```

```
##
## Call:
## glm(formula = y ~ . - 1, family = poisson, data = as.data.frame(X))
```

```
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## `(Intercept)`  1.08534    0.03592  30.213  < 2e-16 ***
## PowerSeller   -0.02833    0.04433  -0.639  0.52280
## VerifyID      -0.34902    0.13008  -2.683  0.00729 **
## Sealed         0.50101    0.06676   7.504 6.18e-14 ***
## Minblem       -0.12189    0.08388  -1.453  0.14619
## MajBlem       -0.24884    0.09865  -2.523  0.01165 *
## LargNeg        0.03610    0.07075   0.510  0.60987
## LogBook       -0.07280    0.03505  -2.077  0.03783 *
## MinBidShare   -1.76665    0.08292 -21.306  < 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: 4131.37  on 700  degrees of freedom
## Residual deviance:  593.76  on 691  degrees of freedom
## AIC: 2529.5
##
## Number of Fisher Scoring iterations: 5
```

```
cat("\nAs observed from summary, Intercept along with VerifyID, Sealed, \n
MajBlem, LogBook and MinBidShare are significant covariates.")
```

```
##
## As observed from summary, Intercept along with VerifyID, Sealed,
##
## MajBlem, LogBook and MinBidShare are significant covariates.
```

```
################################### Part B ##############################
# Log-posterior function (with Zellner's g-prior)
log_posterior <- function(beta, X, y) {
  lambda <- exp(X %*% beta)       # exp(X ) = predicted Poisson rates
  log_lik <- sum(dpois(y,lambda,log=TRUE))    # Sum of log-likelihoods
  log_prior <- dmvnorm(beta,mean=rep(0,dim(X)[2]),
                       sigma=100*solve(t(X)%*%X),log=TRUE)
  return(log_lik + log_prior)
}
# Find posterior mode
optim_result<-optim(rep(0,dim(X)[2]),log_posterior,X=X,y=y,
                    control=list(fnscale=-1),hessian=TRUE)
beta_tilde<-optim_result$par
J_inv<-solve(-optim_result$hessian)  # Posterior covariance
names(beta_tilde)<-colnames(X)
colnames(J_inv)<-rownames(J_inv)<-colnames(X)
cat("\nBeta_tilde:\n")
```

```
##
## Beta_tilde:
```

```
print(beta_tilde)
```

```
##  (Intercept)  PowerSeller     VerifyID       Sealed      Minblem      MajBlem
##  1.083356669 -0.007280161 -0.024583323  0.503016907 -0.043645091 -0.105898988
```

```
##      LargNeg      LogBook  MinBidShare
##  0.214272038 -0.066582591 -1.652945142
```

```
cat("\nCovariance Matrix (Jy(-1) beta_tilde)\n")
```

```
##
## Covariance Matrix (Jy(-1) beta_tilde)
```

```
print(J_inv)
```

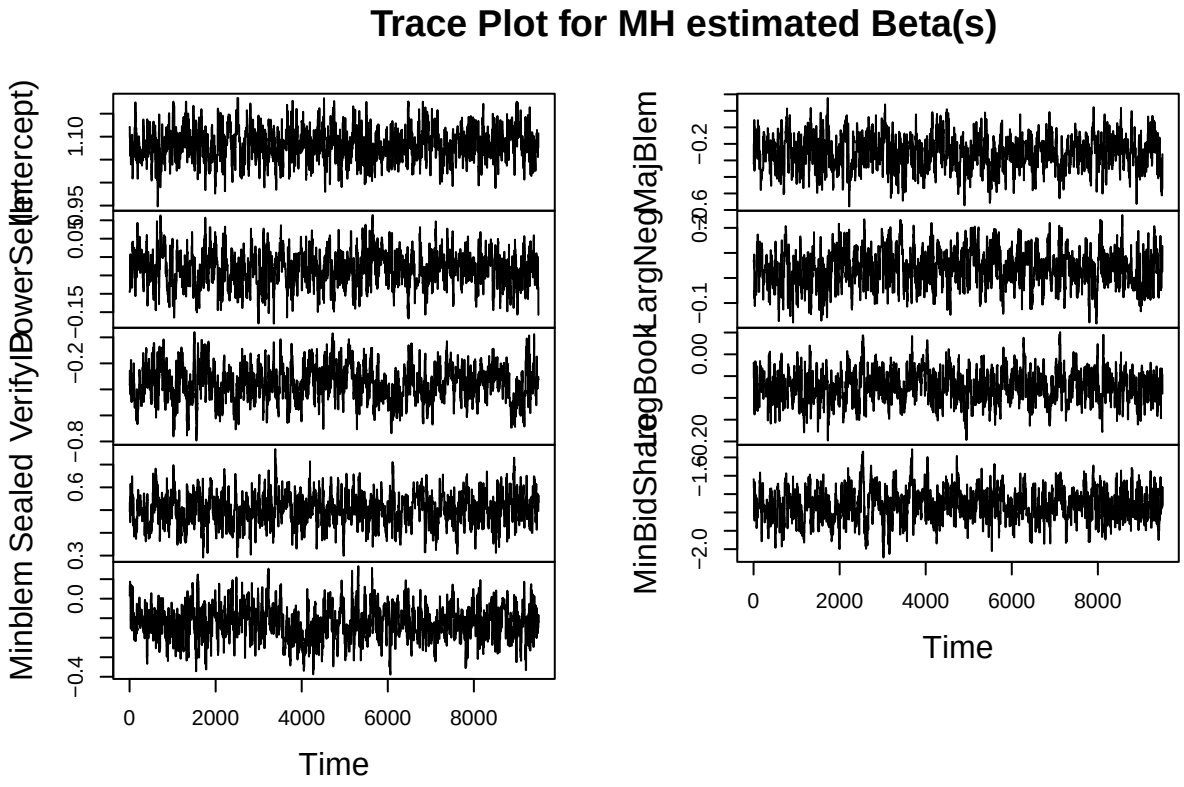
```
##                (Intercept)   PowerSeller      VerifyID        Sealed
## (Intercept)  1.260971e-03 -0.0010014926 -0.0002880175 -0.0005185488
## PowerSeller -1.001493e-03  0.0019142227 -0.0001124431 -0.0001786808
## VerifyID    -2.880175e-04 -0.0001124431  0.0124865323 -0.0013641746
## Sealed      -5.185488e-04 -0.0001786808 -0.0013641746  0.0043005309
## Minblem     -6.064828e-04  0.0000841006  0.0001045665  0.0005164962
## MajBlem     -4.138960e-04 -0.0002287781  0.0003588459  0.0005360809
## LargNeg     -6.857255e-04  0.0003631365  0.0003997369  0.0004405196
## LogBook      3.154009e-05  0.0002124258 -0.0002575469 -0.0000768578
## MinBidShare  1.325391e-03 -0.0007370076 -0.0001983139 -0.0003699649
##                   Minblem       MajBlem       LargNeg       LogBook
## (Intercept) -6.064828e-04 -4.138960e-04 -6.857255e-04  3.154009e-05
## PowerSeller  8.410060e-05 -2.287781e-04  3.631365e-04  2.124258e-04
## VerifyID     1.045665e-04  3.588459e-04  3.997369e-04 -2.575469e-04
## Sealed       5.164962e-04  5.360809e-04  4.405196e-04 -7.685780e-05
## Minblem      6.430545e-03  4.962350e-04  8.833409e-05 -1.109472e-04
## MajBlem      4.962350e-04  8.650177e-03  5.783662e-04 -9.383533e-05
## LargNeg      8.833409e-05  5.783662e-04  4.405579e-03 -4.005484e-04
## LogBook     -1.109472e-04 -9.383533e-05 -4.005484e-04  1.186365e-03
## MinBidShare -3.698609e-04  3.474073e-04 -6.705869e-05  1.330474e-03
##               MinBidShare
## (Intercept)  1.325391e-03
## PowerSeller -7.370076e-04
## VerifyID    -1.983139e-04
## Sealed      -3.699649e-04
## Minblem     -3.698609e-04
## MajBlem      3.474073e-04
## LargNeg     -6.705869e-05
## LogBook      1.330474e-03
## MinBidShare  6.373720e-03
```

```
################################### Part C ##############################
RWMSampler <- function(logPostFunc,theta_init,cov_prop,n_iter,burn_in,c,...) {
  # logPostFunc: Function to compute the log-posterior density. First argument
  # must be `theta`.
  # theta_init: Initial parameter vector.
  # cov_prop Proposal covariance matrix.
  # n_iter Total number of iterations.
  # burn_in Burn-in samples.
  # c Step size scaling factor.  # Tune for 25-30% acceptance
  p<-length(theta_init)
  theta<-matrix(NA, n_iter, p)
  theta[1, ]<-theta_init
  log_post_current<-logPostFunc(theta[1, ], ...)
  n_accept<-0
  for (i in 2:n_iter) {
```

```
    # Propose new theta
    theta_prop<-MASS::mvrnorm(1,mu=theta[i-1, ],Sigma=c*cov_prop)
    # Compute log-posterior at proposal
    # Log acceptance probability (avoid numerical overflow)
    log_alpha<-logPostFunc(theta_prop,...)-logPostFunc(theta[i-1, ], ...)
    if(log(runif(1)) < log_alpha){
      theta[i, ]<-theta_prop
      n_accept<-n_accept+1
    } else{
      theta[i, ]<-theta[i-1, ]
    }
  }
  theta<-theta[(burn_in+1):n_iter, ]
  acceptance_rate<-n_accept/n_iter
  cat("Acceptance rate:", acceptance_rate,"\n")
  return(theta)
}

LogPostPoisson <- function(theta, X, y) {
  lambda<-exp(X%*%theta)
  log_lik<-sum(dpois(y,lambda,log=TRUE))
  log_prior<-mvtnorm::dmvnorm(theta,mean=rep(0,ncol(X)),
                              sigma=100*solve(t(X)%*%X),log=TRUE)
  return(log_lik+log_prior)
}
samplesMH<-RWMSampler(logPostFunc=LogPostPoisson,theta_init=beta_tilde,
                      cov_prop=J_inv,n_iter=10000,burn_in=500,c=0.7,X=X,y=y)
```

```
## Acceptance rate: 0.2605
```

```
colnames(samplesMH)<-colnames(X)
plot.ts(samplesMH, main = "Trace Plot for MH estimated Beta(s)")
```




```
cat("\nSummary Results\n")
```

```
##
## Summary Results
```

```
print(data.frame(Covariate=colnames(samplesMH),
           Mean=round(colMeans(samplesMH),4),
           Std=round(apply(samplesMH,2,sd),4)),row.names=FALSE)
```

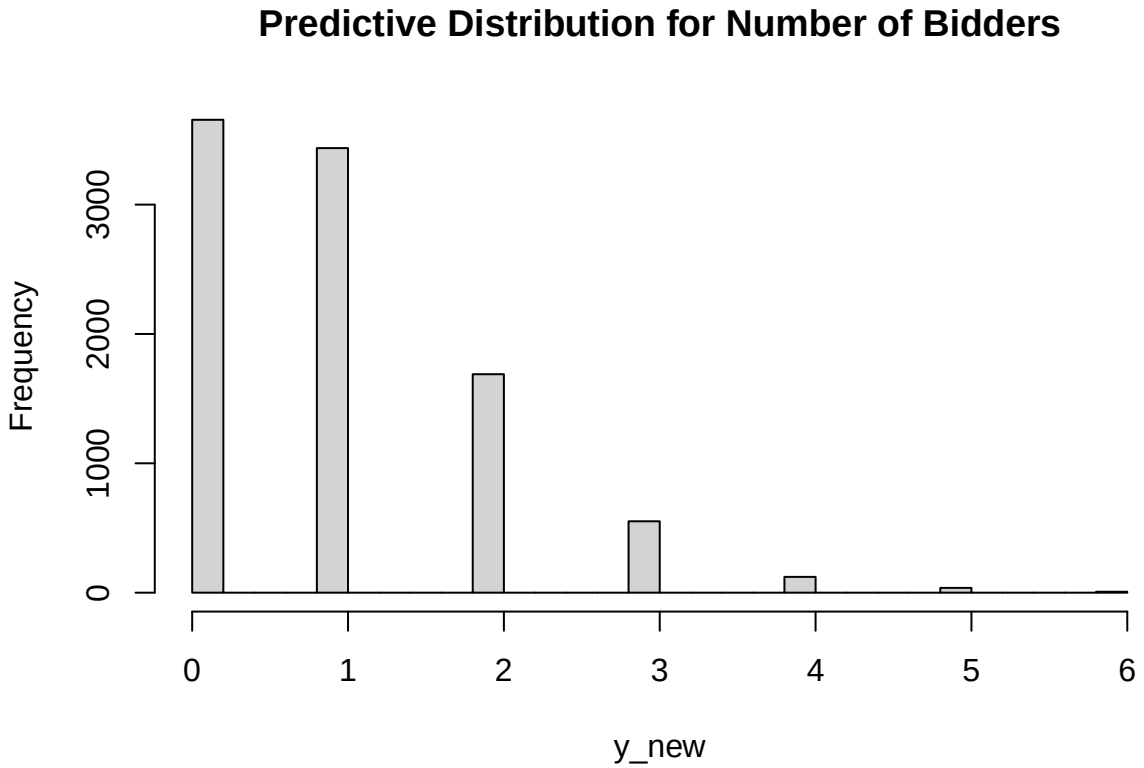
```
##    Covariate    Mean    Std
##  (Intercept)  1.0831 0.0351
##  PowerSeller -0.0332 0.0457
##     VerifyID -0.3545 0.1350
##       Sealed  0.5028 0.0685
##      Minblem -0.1231 0.0855
##      MajBlem -0.2560 0.0996
##      LargNeg  0.0350 0.0719
##      LogBook -0.0736 0.0353
##  MinBidShare -1.7604 0.0841
```

```
cat("\nComparing the values with those in part B, we find while values for
    Intercept, Sealed , LogBook and MinBidShare are similar, other covariates
    show significant differences in values\n")
```

```
##
## Comparing the values with those in part B, we find while values for
##     Intercept, Sealed , LogBook and MinBidShare are similar, other covariates
##     show significant differences in values
```

```
################################## Part D ############################
x_new<-c(1,1,0,1,0,1,0,1.3,0.7)
```

```r
lambda_new<-exp(samplesMH %*% x_new)
y_new<-rpois(nrow(samplesMH),lambda_new)
hist(y_new,breaks=30,main="Predictive Distribution for Number of Bidders")
```




```r
cat("\nProbability of no bidders (y_new = 0):",mean(y_new==0),"\n")
```

```
##
## Probability of no bidders (y_new = 0): 0.3848421
```