Tarea 4

Antonio, H. F.

2022-11-02

Objetivo:

Reproducir el Ejemplo 3 del artículo de "On Bayesian Analysis of Generalized Linear Models Using Jeffreys's Prior Joseph G. Ibrahim; Purushottam W. Laud. Journal of the American Statistical Association, Vol. 86, No. 416. (Dec., 1991), pp. 981-986" utilizando los métodos siguientes:

1. Método de Tierney y Kadane
2. Metropolis-Hastings
3. Gibbs sampler

Para los dos primeros incisos son para programar en R (o cualquier otro lenguaje) se adjuntan el artículo base y otro artículo con la aplicación a regresión logítica. El tercer inciso se puede resolver con BUGS (WinBUGS, OpenBUGS, JAGS) en combinación con R o programando el Gibbs en R muestreando de las condicionales completas con Aceptación-Rechazo o Aceptación-Rechazo Adaptativo (paquete ARS).

Resultados:

Información general

El ejemplo 3, del artículo mencionado anteriormente, se basa en los datos obtenidos en el estudio de Finney (1947) del efecto de la tasa y volumen de aire inspirado sobre la constricción transitoria de los vasos en la piel de los dedos. La variable respuesta es binaria donde 1 indica la ocurrencia de la constricción y 0 lo contrario.

Datos

Se cuenta con 39 observaciones de la tasa (R), volumen (V) y respuesta (Y), como se muestra en la siguiente tabla,

Y	V [l]	R [lps]	Y	V [l]	R [lps]	Y	V [l]	R [lps]	Y	V [l]	R [lps]
1	3.7	0.825	0	0.8	0.57	0	0.4	2	1	2.7	0.75
1	3.5	1.09	0	0.55	2.75	0	0.95	1.36	0	2.35	0.03
1	1.25	2.5	0	0.6	3	0	1.35	1.35	0	1.1	1.83
1	0.75	1.5	1	1.4	2.33	0	1.5	1.36	1	1.1	2.2
1	0.8	3.2	1	0.75	3.75	1	1.6	1.78	1	1.2	2
1	0.7	3.5	1	2.3	1.64	0	0.6	1.5	1	0.8	3.33
0	0.6	0.75	1	3.2	1.6	1	1.8	1.5	0	0.95	1.9
0	1.1	1.7	1	0.85	1.415	0	0.95	1.9	0	0.75	1.9
0	0.9	0.75	0	1.7	1.06	1	1.9	0.95	1	1.3	1.625
0	0.9	0.45	1	1.8	1.8	0	1.6	0.4			

Modelo

Debido a que la variable respuesta (y_i) es binaria, se puede modelar con una distribución Bernoulli, donde la variable respuesta tiene la probabilidad π_i de tomar el valor 1, y la probabilidad $1-\pi_i$ para 0, entonces,

$$y_i|\pi_i \sim Bernoulli(\pi_i) \tag{1}$$

Bajo el MLG ligamos la $E[y_i]$ con covariables ($\mathbf{X}$) mediante una función liga, que el caso de una respuesta binomial la función es la logit, esto es,

$$\pi_i = \frac{exp\{\mathbf{x}_i'\beta\}}{1 + exp\{\mathbf{x}_i'\beta\}} \tag{2}$$

Dependiendo del enfoque utiliizado, un elemento que siempre está presente, es la función de verosimilitud, la cual se forma con (1) y (2), es decir,

$$L(\beta|\mathbf{y}, \mathbf{X}) = \prod_{i=1}^{n} \left(\frac{exp\{\mathbf{x}_i'\beta\}}{1 + exp\{\mathbf{x}_i'\beta\}} \right)^{y_i} \left(\frac{1}{1 + exp\{\mathbf{x}_i\beta\}} \right)^{1-y_i} \tag{3}$$

Método de Tierney y Kadane

Este método fue propuesto por Tierney y Kadane (1986) para aproximar esperanzas a posteri de funciones positivas. Está basado en expansión de series de segundo orden para aproximar el término de la integral. Para aproximar la función marginal de los B's se utiliza la siguiente función:

$$\hat{p}(\beta_1 \mathbf{y}, \mathbf{X}) = \left(\frac{det\Sigma * (\beta_1)}{2\pi det\Sigma} \right)^{0.5} exp\left\{ \; n[h * (\beta_1, \hat{\beta}_2^* - h(\beta_1, \hat{\beta}_2))] \right\} \tag{4}$$

```
finney <- read.table("finney.txt", quote="\"", comment.char="")
finney$V2 <- log(finney$V2)
finney$V3 <- log(finney$V3)
colnames(finney) <- c("y","v","r")
M1<-glm(y~.,data=finney,family=binomial)
y<-finney$y
X<-model.matrix(M1)
N<-nrow(X)
J<-ncol(X)

mon.names <- "LP"
parm.names <- as.parm.names(list(beta=rep(0,J)))

MyData <- list(J=J,X=X,mon.names=mon.names,parm.names=parm.names,y=y)

##Specify a model
Model<-function(parm,Data)
{
## parameters
beta<-parm[1:Data$J]
## Log(prior Densities)
beta.prior<-dnormv(beta,0,10000,log=F)
mu<-tcrossprod(Data$X,t(beta))
## Log-Likelihood
#lambda<-exp(mu)
#LL<-sum(dpois(Data$y,lambda,log=T))
pi <- exp(mu)/(1+exp(mu))
LL<-sum(dbern(Data$y,pi,log=T))
##Log-posterior
```

```r
LP<-LL+sum(beta.prior)
Modelout<-list(LP=LP,Dev=-2*LL,Monitor=LP,
yhat=rnorm(length(mu),mu),parm=parm)
return(Modelout)
}

## Initial values
Initial.Values<-c(rep(0,J))
```

```r
M2<-LaplaceApproximation(Model, Initial.Values, Data = MyData,sir=TRUE,
Iterations=5000,Method="TR")
```

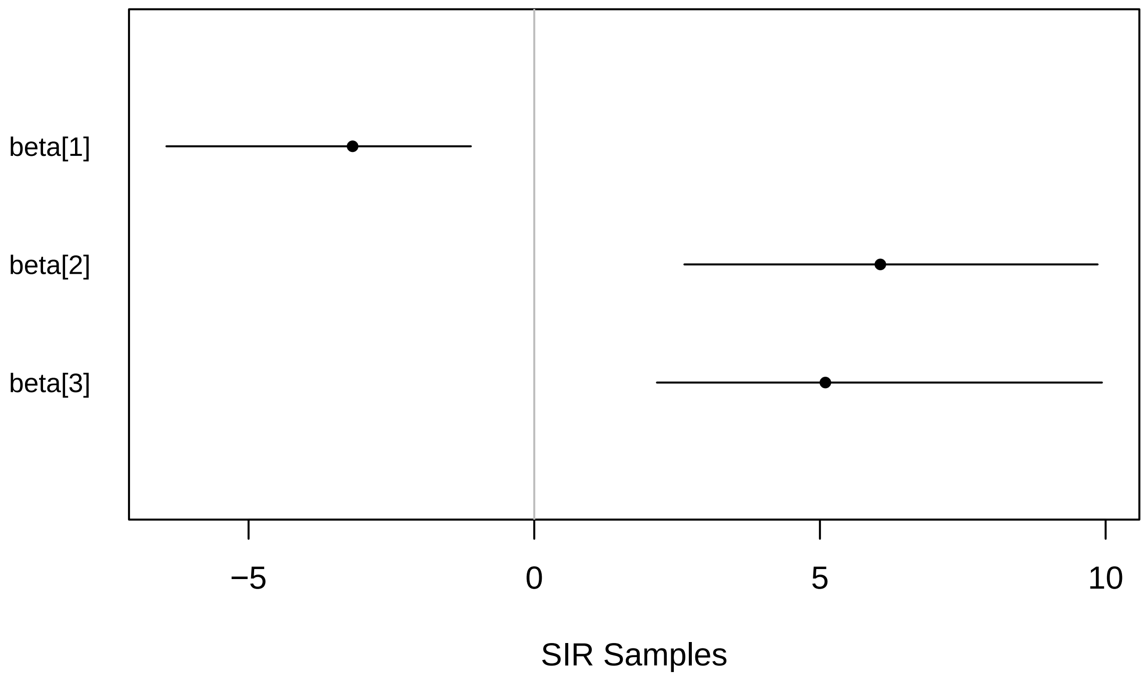
```
## Sample Size:  39
## Laplace Approximation begins...
## Estimating the Covariance Matrix
## Sampling from Posterior with Sampling Importance Resampling
## Creating Summary from Point-Estimates
## Creating Summary from Posterior Samples
## Estimating Log of the Marginal Likelihood
## Laplace Approximation is finished.
```

```r
print(M2)
```

```
##
## Call:
## LaplaceApproximation(Model = Model, parm = Initial.Values, Data = MyData,
##     Iterations = 5000, Method = "TR", sir = TRUE)
##
## Converged: TRUE
## Covariance Matrix: (NOT SHOWN HERE; diagonal shown instead)
##  beta[1]  beta[2]  beta[3]
## 1.744463 3.477610 3.378150
##
## Deviance (Final):  29.22738
## History: (NOT SHOWN HERE)
## Initial Values:
## [1] 0 0 0
##
## Iterations: 8
## Log(Marginal Likelihood): -11.8258
## Log-Posterior (Final): -14.60173
## Log-Posterior (Initial): -27.02077
## Minutes of run-time: 0.01
## Monitor: (NOT SHOWN HERE)
## Posterior: (NOT SHOWN HERE)
## Step Size (Final): [1] 7.105427e-15
## Step Size (Initial): 1
## Summary1: (SHOWN BELOW)
## Summary2: (SHOWN BELOW)
## Tolerance (Final): 2.081369e-14
## Tolerance (Stop): 1e-05
##
## Summary1:
##              Mode       SD         LB         UB
## beta[1] -2.875412 1.320781 -5.5169747 -0.2338493
```

```
## beta[2]  5.179309 1.864835  1.4496392  8.9089796
## beta[3]  4.561661 1.837974  0.8857122  8.2376098
##
## Summary2:
##                 Mode       SD        MCSE  ESS          LB     Median          UB
## beta[1]    -3.376403 1.342463 0.04245240 1000   -6.437660  -3.179361   -1.109701
## beta[2]     6.047870 1.910193 0.06040562 1000    2.627643   6.057398    9.859542
## beta[3]     5.333851 1.921675 0.06076870 1000    2.147151   5.095201    9.935112
## Deviance   32.080705 2.271290 0.07182449 1000   29.466354  31.583267   37.560683
## LP        -16.028401 1.135648 0.03591235 1000  -18.768396 -15.779691  -14.721221
```

```
caterpillar.plot(M2,Parms="beta")
```




```
Initial.Values<-as.initial.values(M2)
M3 <-LaplacesDemon(Model, Data=MyData, Initial.Values,
                   Covar=M2$Covar,Iterations=10000,Algorithm="IM",
                   Specs=list(mu=M2$Summary1[1:length(Initial.Values),1]))
```

LaplaceDemon

```
##
## Laplace's Demon was called on Fri Nov 11 04:37:49 2022
##
## Performing initial checks...
## Algorithm: Independence Metropolis
##
```

```
## Laplace's Demon is beginning to update...
## Iteration: 100,   Proposal: Multivariate,   LP: -14.9
## Iteration: 200,   Proposal: Multivariate,   LP: -15.1
## Iteration: 300,   Proposal: Multivariate,   LP: -14.9
## Iteration: 400,   Proposal: Multivariate,   LP: -14.7
## Iteration: 500,   Proposal: Multivariate,   LP: -14.8
## Iteration: 600,   Proposal: Multivariate,   LP: -14.9
## Iteration: 700,   Proposal: Multivariate,   LP: -15
## Iteration: 800,   Proposal: Multivariate,   LP: -14.9
## Iteration: 900,   Proposal: Multivariate,   LP: -15.3
## Iteration: 1000,   Proposal: Multivariate,   LP: -15.3
## Iteration: 1100,   Proposal: Multivariate,   LP: -15
## Iteration: 1200,   Proposal: Multivariate,   LP: -15.3
## Iteration: 1300,   Proposal: Multivariate,   LP: -15.3
## Iteration: 1400,   Proposal: Multivariate,   LP: -14.6
## Iteration: 1500,   Proposal: Multivariate,   LP: -14.7
## Iteration: 1600,   Proposal: Multivariate,   LP: -14.9
## Iteration: 1700,   Proposal: Multivariate,   LP: -15.1
## Iteration: 1800,   Proposal: Multivariate,   LP: -16.7
## Iteration: 1900,   Proposal: Multivariate,   LP: -14.7
## Iteration: 2000,   Proposal: Multivariate,   LP: -15.7
## Iteration: 2100,   Proposal: Multivariate,   LP: -14.7
## Iteration: 2200,   Proposal: Multivariate,   LP: -15.1
## Iteration: 2300,   Proposal: Multivariate,   LP: -15.3
## Iteration: 2400,   Proposal: Multivariate,   LP: -14.9
## Iteration: 2500,   Proposal: Multivariate,   LP: -15.1
## Iteration: 2600,   Proposal: Multivariate,   LP: -14.8
## Iteration: 2700,   Proposal: Multivariate,   LP: -15
## Iteration: 2800,   Proposal: Multivariate,   LP: -16.8
## Iteration: 2900,   Proposal: Multivariate,   LP: -14.9
## Iteration: 3000,   Proposal: Multivariate,   LP: -15.2
## Iteration: 3100,   Proposal: Multivariate,   LP: -14.8
## Iteration: 3200,   Proposal: Multivariate,   LP: -14.9
## Iteration: 3300,   Proposal: Multivariate,   LP: -15.5
## Iteration: 3400,   Proposal: Multivariate,   LP: -15.5
## Iteration: 3500,   Proposal: Multivariate,   LP: -16.6
## Iteration: 3600,   Proposal: Multivariate,   LP: -15.1
## Iteration: 3700,   Proposal: Multivariate,   LP: -14.6
## Iteration: 3800,   Proposal: Multivariate,   LP: -15
## Iteration: 3900,   Proposal: Multivariate,   LP: -14.9
## Iteration: 4000,   Proposal: Multivariate,   LP: -15.3
## Iteration: 4100,   Proposal: Multivariate,   LP: -14.9
## Iteration: 4200,   Proposal: Multivariate,   LP: -14.6
## Iteration: 4300,   Proposal: Multivariate,   LP: -15
## Iteration: 4400,   Proposal: Multivariate,   LP: -14.9
## Iteration: 4500,   Proposal: Multivariate,   LP: -15
## Iteration: 4600,   Proposal: Multivariate,   LP: -14.7
## Iteration: 4700,   Proposal: Multivariate,   LP: -14.7
## Iteration: 4800,   Proposal: Multivariate,   LP: -15.5
## Iteration: 4900,   Proposal: Multivariate,   LP: -14.9
## Iteration: 5000,   Proposal: Multivariate,   LP: -14.7
## Iteration: 5100,   Proposal: Multivariate,   LP: -14.9
## Iteration: 5200,   Proposal: Multivariate,   LP: -14.6
## Iteration: 5300,   Proposal: Multivariate,   LP: -14.8
```

```
## Iteration: 5400,   Proposal: Multivariate,   LP: -14.9
## Iteration: 5500,   Proposal: Multivariate,   LP: -14.9
## Iteration: 5600,   Proposal: Multivariate,   LP: -14.9
## Iteration: 5700,   Proposal: Multivariate,   LP: -15
## Iteration: 5800,   Proposal: Multivariate,   LP: -14.7
## Iteration: 5900,   Proposal: Multivariate,   LP: -14.7
## Iteration: 6000,   Proposal: Multivariate,   LP: -14.9
## Iteration: 6100,   Proposal: Multivariate,   LP: -15.5
## Iteration: 6200,   Proposal: Multivariate,   LP: -14.9
## Iteration: 6300,   Proposal: Multivariate,   LP: -15.2
## Iteration: 6400,   Proposal: Multivariate,   LP: -14.7
## Iteration: 6500,   Proposal: Multivariate,   LP: -15.2
## Iteration: 6600,   Proposal: Multivariate,   LP: -15.3
## Iteration: 6700,   Proposal: Multivariate,   LP: -14.7
## Iteration: 6800,   Proposal: Multivariate,   LP: -15
## Iteration: 6900,   Proposal: Multivariate,   LP: -14.8
## Iteration: 7000,   Proposal: Multivariate,   LP: -14.6
## Iteration: 7100,   Proposal: Multivariate,   LP: -15.2
## Iteration: 7200,   Proposal: Multivariate,   LP: -17.1
## Iteration: 7300,   Proposal: Multivariate,   LP: -14.8
## Iteration: 7400,   Proposal: Multivariate,   LP: -15.1
## Iteration: 7500,   Proposal: Multivariate,   LP: -15.5
## Iteration: 7600,   Proposal: Multivariate,   LP: -14.7
## Iteration: 7700,   Proposal: Multivariate,   LP: -14.8
## Iteration: 7800,   Proposal: Multivariate,   LP: -14.7
## Iteration: 7900,   Proposal: Multivariate,   LP: -15
## Iteration: 8000,   Proposal: Multivariate,   LP: -14.7
## Iteration: 8100,   Proposal: Multivariate,   LP: -16.1
## Iteration: 8200,   Proposal: Multivariate,   LP: -14.8
## Iteration: 8300,   Proposal: Multivariate,   LP: -14.7
## Iteration: 8400,   Proposal: Multivariate,   LP: -14.8
## Iteration: 8500,   Proposal: Multivariate,   LP: -14.7
## Iteration: 8600,   Proposal: Multivariate,   LP: -14.9
## Iteration: 8700,   Proposal: Multivariate,   LP: -14.9
## Iteration: 8800,   Proposal: Multivariate,   LP: -15.6
## Iteration: 8900,   Proposal: Multivariate,   LP: -15.5
## Iteration: 9000,   Proposal: Multivariate,   LP: -15
## Iteration: 9100,   Proposal: Multivariate,   LP: -15.1
## Iteration: 9200,   Proposal: Multivariate,   LP: -15.1
## Iteration: 9300,   Proposal: Multivariate,   LP: -15.5
## Iteration: 9400,   Proposal: Multivariate,   LP: -14.8
## Iteration: 9500,   Proposal: Multivariate,   LP: -15.2
## Iteration: 9600,   Proposal: Multivariate,   LP: -15
## Iteration: 9700,   Proposal: Multivariate,   LP: -14.8
## Iteration: 9800,   Proposal: Multivariate,   LP: -14.8
## Iteration: 9900,   Proposal: Multivariate,   LP: -15.3
## Iteration: 10000,   Proposal: Multivariate,   LP: -14.9
##
## Assessing Stationarity
## Assessing Thinning and ESS
## Creating Summaries
## Estimating Log of the Marginal Likelihood
## Creating Output
##
```

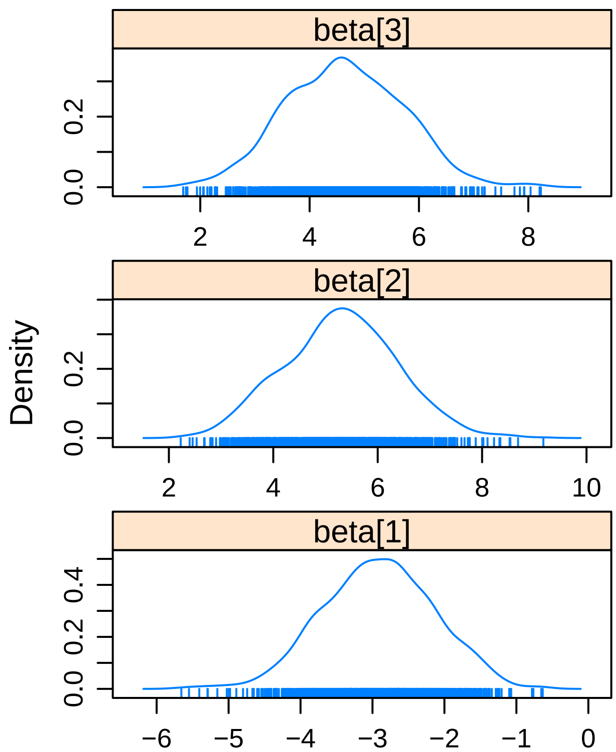
```
## Laplace's Demon has finished.
```

```
print(M3)
```

```
## Call:
## LaplacesDemon(Model = Model, Data = MyData, Initial.Values = Initial.Values,
##     Covar = M2$Covar, Iterations = 10000, Algorithm = "IM", Specs = list(mu = M2$Summary1[1:length(In
##         1]))
##
## Acceptance Rate: 0.3679
## Algorithm: Independence Metropolis
## Covariance Matrix: (NOT SHOWN HERE; diagonal shown instead)
##   beta[1]   beta[2]   beta[3]
## 0.6040177 1.1823463 1.1509729
##
## Covariance (Diagonal) History: (NOT SHOWN HERE)
## Deviance Information Criterion (DIC):
##         All Stationary
## Dbar 30.257     30.257
## pD    0.385      0.385
## DIC  30.642     30.642
## Initial Values:
## [1] -2.875412  5.179309  4.561661
##
## Iterations: 10000
## Log(Marginal Likelihood): -12.44355
## Minutes of run-time: 0.1
## Model: (NOT SHOWN HERE)
## Monitor: (NOT SHOWN HERE)
## Parameters (Number of): 3
## Posterior1: (NOT SHOWN HERE)
## Posterior2: (NOT SHOWN HERE)
## Recommended Burn-In of Thinned Samples: 0
## Recommended Burn-In of Un-thinned Samples: 0
## Recommended Thinning: 10
## Specs: (NOT SHOWN HERE)
## Status is displayed every 100 iterations
## Summary1: (SHOWN BELOW)
## Summary2: (SHOWN BELOW)
## Thinned Samples: 1000
## Thinning: 10
##
##
## Summary of All Samples
##               Mean        SD       MCSE       ESS         LB     Median
## beta[1]   -2.917983 0.7775735 0.02397016 1000.0000  -4.417718  -2.913924
## beta[2]    5.294772 1.0878955 0.03363967 1000.0000   3.220664   5.298614
## beta[3]    4.644701 1.0733677 0.03298683 1000.0000   2.645163   4.618645
## Deviance  30.257047 0.8771657 0.02958028  903.7686  29.301969  30.034485
## LP       -15.116567 0.4385826 0.01479015  903.7669 -16.216956 -15.005286
##                 UB
## beta[1]   -1.446039
## beta[2]    7.426517
## beta[3]    6.651280
## Deviance  32.457831
```

```
## LP       -14.639025
##
##
## Summary of Stationary Samples
##                 Mean        SD       MCSE       ESS          LB     Median
## beta[1]   -2.917983 0.7775735 0.02397016 1000.0000   -4.417718  -2.913924
## beta[2]    5.294772 1.0878955 0.03363967 1000.0000    3.220664   5.298614
## beta[3]    4.644701 1.0733677 0.03298683 1000.0000    2.645163   4.618645
## Deviance  30.257047 0.8771657 0.02958028  903.7686   29.301969  30.034485
## LP       -15.116567 0.4385826 0.01479015  903.7669  -16.216956 -15.005286
##                   UB
## beta[1]   -1.446039
## beta[2]    7.426517
## beta[3]    6.651280
## Deviance  32.457831
## LP       -14.639025
```

```
samples <- mcmc(M3$Posterior2)
densityplot(samples)
```




```
# Modelo Logit (MV)
####
finney <- read.table("finney.txt", quote="\"", comment.char="")
finney$V2 <- log(finney$V2)
finney$V3 <- log(finney$V3)
```

```r
colnames(finney) <- c("y","v","r")
y<-finney$y
fit_logit <- glm(y~.,data=finney,family=binomial)
summary(fit_logit)
```

Metropolis-Hastings

```
##
## Call:
## glm(formula = y ~ ., family = binomial, data = finney)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4527  -0.6110   0.1001   0.6181   2.2775
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -2.875      1.321  -2.177  0.02946 *
## v              5.179      1.865   2.778  0.00547 **
## r              4.562      1.838   2.482  0.01306 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 54.040  on 38  degrees of freedom
## Residual deviance: 29.227  on 36  degrees of freedom
## AIC: 35.227
##
## Number of Fisher Scoring iterations: 6
```

```r
X<-model.matrix(fit_logit)
```

```r

# Log-verosimlitud del modelo de regresión logística
loglik <- function(beta, y, X) {
eta <- c(X %*% beta)
sum(y * eta - log(1 + exp(eta)))
}
# Log-a posteriori
logpost <- function(beta, y, X) {
beta.prior<-dnormv(beta,0,10000,log=F)
loglik(beta, y, X) + sum(beta.prior)
}
```

Log-Verosimiltud

```r

# R:= número de muestras
# burn_in:= número de muestras elimindas
# S:=Matriz de covarianzas de la propuesta normal

RMH <- function(R, burn_in, y, X, S) {
p <- ncol(X)
out <- matrix(0, R, p) # Inicializar una matriz vacia para almacer la salida
```

```
beta <- rep(0, p) # Valores iniciales
logp <- logpost(beta, y, X)
# Eigen-descomposición (puede ser Cholesky)
eig <- eigen(S, symmetric = TRUE)
A1 <- t(eig$vectors) * sqrt(eig$values)
# Inicio del Gibbs sampler
for (r in 1:(burn_in + R)) {
beta_new <- beta + c(matrix(rnorm(p), 1, p) %*% A1)
logp_new <- logpost(beta_new, y, X)
alpha <- min(1, exp(logp_new - logp))
if (runif(1) < alpha) {
logp <- logp_new
beta <- beta_new # Aceptar el valor
}
# Almacenar los valores después del periodo burn-in
if (r > burn_in) {
out[r - burn_in, ] <- beta
}
}
out
}
```

MH con caminata aleatoria (MRW)

```
library(coda)
R <- 30000 # Numero de muestras retenidas
burn_in <- 30000 # Burn-in
set.seed(123)
# Matriz de covarianzas de la propuesta
S <- diag(1e-3, ncol(X))
# Correr el MCMC
start.time <- Sys.time()
fit_MCMC <- as.mcmc(RMH(R, burn_in, y, X, S)) # Convertir la matriz en un objeto "coda"
end.time <- Sys.time()
time_in_sec <- as.numeric(end.time - start.time)
time_in_sec
```

MH preliminar

```
## [1] 5.914857
```

```
# Diagnóstico
summary(effectiveSize(fit_MCMC)) # Tamaño de muestra efectivo
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.219   3.772   4.325   4.348   4.913   5.501
```

```
summary(R / effectiveSize(fit_MCMC)) # Tiempo de autocorrelación integrado
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5454    6195    6937    7236    8128    9318
```

```
summary(1 - rejectionRate(fit_MCMC)) # Tasa de aceptación
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9709  0.9709  0.9709  0.9709  0.9709  0.9709
```

```r
set.seed(123)
# Correr el MCMC
start.time <- Sys.time()
# Se utiliza la aproximación de Laplace de la matriz de covarianzas
fit_logit <- glm(y~.,data=finney,family=binomial)
p <- ncol(X)
S <- 2.38^2 * vcov(fit_logit) / p
# MCMC
fit_MCMC <- as.mcmc(RMH(R, burn_in, y, X, S)) # Convertit la matriz en un objeto coda
end.time <- Sys.time()
time_in_sec <- as.numeric(end.time - start.time)
time_in_sec
```

Aproximación de la matriz de covarianzas a posteriori

```
## [1] 4.896195
```

```r
# Diagnóstico
summary(effectiveSize(fit_MCMC)) # Tamaño de muestra efectivo
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2361    2419    2476    2458    2507    2537
```

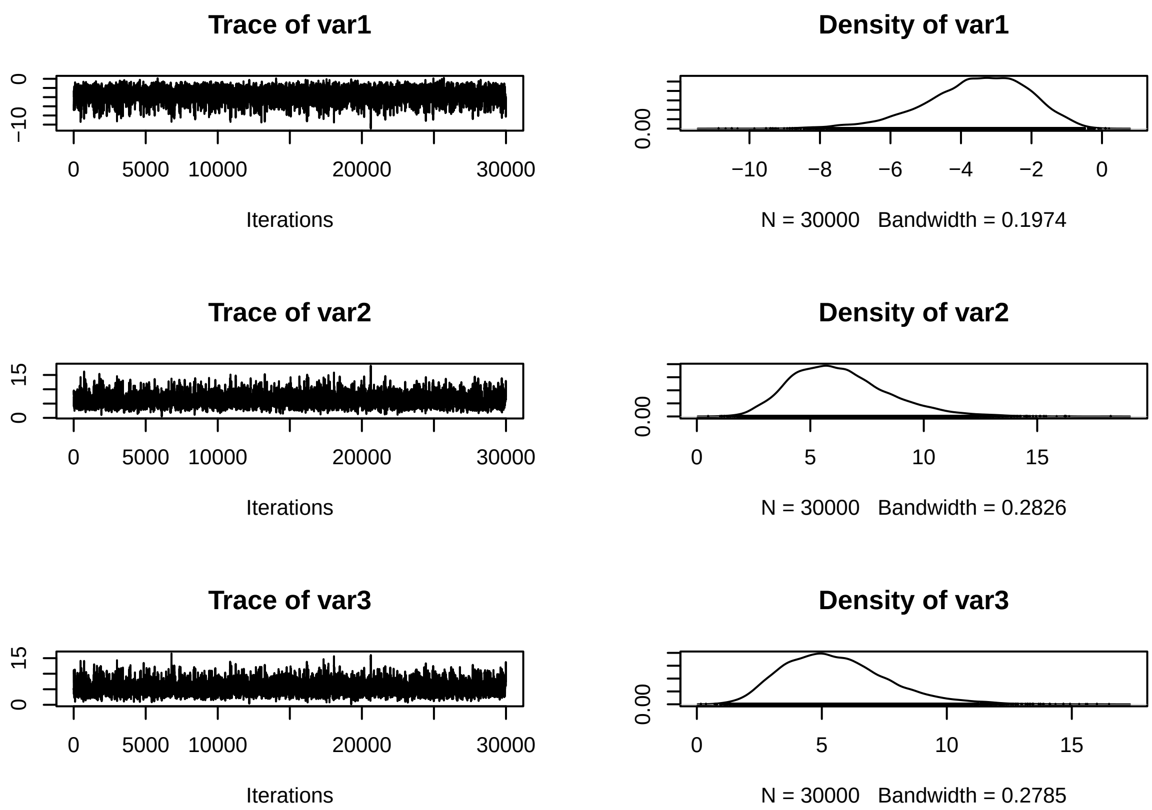
```r
summary(R / effectiveSize(fit_MCMC)) # Tiempo de autocorrelación integrado
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.82   11.97   12.12   12.22   12.41   12.71
```

```r
summary(1 - rejectionRate(fit_MCMC)) # Tasa de aceptación
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3367  0.3367  0.3367  0.3367  0.3367  0.3367
```

```r
# Trazas
plot(fit_MCMC[, 1:3])
```



Gibbs sampler

Para este método se utilizó JAGS (Just Another Gibss Sampler). Este método va generando muestras de las distribuciones condicionales a posteriori de la distribución de interes, en este caso el vector de parámetros β.

```
finney.dat  <-read.table("finney.txt",header=F)
colnames(finney.dat) <- c("y","v","r")
x1 <- finney.dat$v
x2 <- finney.dat$r
y <- finney.dat$y
n <- length(y)
datos <- list(n=n,y=finney.dat$y,v=finney.dat$v,r=finney.dat$r)
mlefit <- glm(y~log(x1)+log(x2),family=binomial(link = "logit"))
mean.as <-summary(mlefit)$coef[,1]
sd.as <- summary(mlefit)$coef[,2]
logis <-"logis.txt"

iniciales <- list(b=mean.as)
jagsmod <- jags.model(logis,d=datos,inits=iniciales)
```

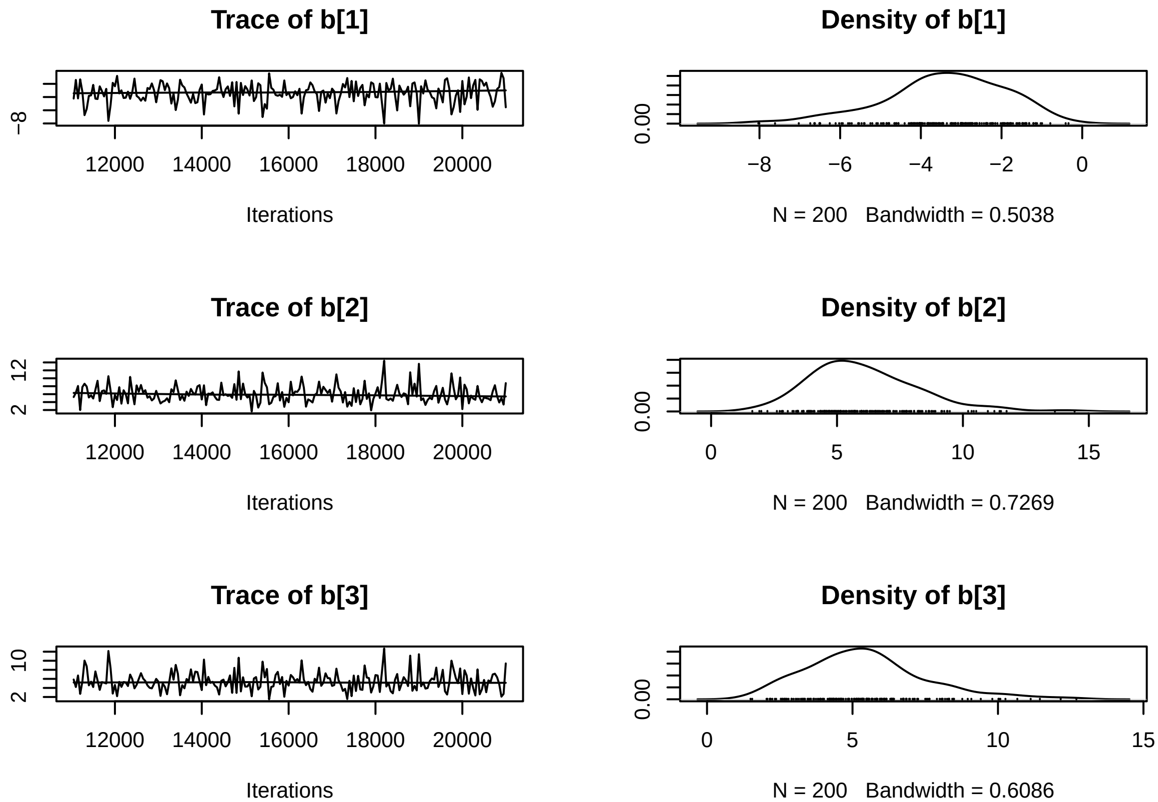
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 39
##    Unobserved stochastic nodes: 3
##    Total graph size: 302
```

```
##
## Initializing model
```

```
update(jagsmod, 10000)
cadena=coda.samples(jagsmod, "b",
                    n.iter=10000,thin=50)
summary(cadena)
```

```
##
## Iterations = 11050:21000
## Thinning interval = 50
## Number of chains = 1
## Sample size per chain = 200
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##        Mean    SD Naive SE Time-series SE
## b[1] -3.410 1.503   0.1063        0.06531
## b[2]  6.067 2.155   0.1524        0.15237
## b[3]  5.457 2.087   0.1475        0.14755
##
## 2. Quantiles for each variable:
##
##        2.5%    25%    50%    75%  97.5%
## b[1] -6.644 -4.167 -3.237 -2.329 -1.118
## b[2]  2.722  4.586  5.725  7.237 11.253
## b[3]  2.170  4.180  5.315  6.400 10.266
```

```
plot(cadena)
```


Trace of b[1]
−8
12000
14000
16000
18000
20000
Iterations
Density of b[1]
0.00
−8
−6
−4
−2
0
N = 200   Bandwidth = 0.5038
Trace of b[2]
12
2
12000
14000
16000
18000
20000
Iterations
Density of b[2]
0.00
0
5
10
15
N = 200   Bandwidth = 0.7269
Trace of b[3]
10
2
12000
14000
16000
18000
20000
Iterations
Density of b[3]
0.00
0
5
10
15
N = 200   Bandwidth = 0.6086