MH example 2: Poisson regression for birth rates

Patrick Graham

September 2021

The data.

This example is about regional variation across New Zealand in birth rates in the 15-19 age group. We use a Bayesian Poisson regression to investigate whether birth rates vary between major urban and other areas and between North and South Islands.

##General set-up

```
library(MASS)
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 4.0.3
```

```
library(LearnBayes)  #laplace function
library(arm)
```

```
## Loading required package: Matrix
```

```
## Loading required package: lme4
```

```
##
## arm (Version 1.11-1, built: 2020-4-27)
```

```
## Working directory is C:/Users/Patrick Graham/Documents/Patrick/Stat314-2021/Metrop
olis-Hastings/final
```

```
#setwd("~/Patrick/Stat314-2017") #Should not be needed since data created
                                  #in the code
##births by region example

#set-up the data
births <- c(64,365,149,113,30,65,38,95,117,6,12,9,9,137,36,36)
popsize <- c(5450,51620,14960,9360,1770,5640,3880,9020,17040,
             1420,1600,1290,980,18670,8290,3170)
regions <- c("Northland","Auckland","Waikato","Bay of Plenty",
             "Gisborne","Hawke's Bay","Taranaki",
             "Manawatu-Wanganui","Wellington","Tasman","Nelson",
             "Marlborough","West Coast","Canterbury","Otago",
             "Southland")

majurb <- c(0,1,0,0,0,0,0,0,1,0,0,0,0,1,1,0)
North <- c(1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0)
South <-  1- North

###place into  a data frame for easier viewing and writing out if necessary
births_reg.df <-  data.frame(regions,births,popsize,majurb,South)
###Have a look at the data

births_reg.df
```

```
##              regions births popsize majurb South
## 1          Northland     64    5450      0     0
## 2           Auckland    365   51620      1     0
## 3            Waikato    149   14960      0     0
## 4      Bay of Plenty    113    9360      0     0
## 5           Gisborne     30    1770      0     0
## 6        Hawke's Bay     65    5640      0     0
## 7           Taranaki     38    3880      0     0
## 8  Manawatu-Wanganui     95    9020      0     0
## 9         Wellington    117   17040      1     0
## 10            Tasman      6    1420      0     1
## 11            Nelson     12    1600      0     1
## 12       Marlborough      9    1290      0     1
## 13        West Coast      9     980      0     1
## 14        Canterbury    137   18670      1     1
## 15             Otago     36    8290      1     1
## 16         Southland     36    3170      0     1
```

```
rate <- births / popsize
rate
```

```
##  [1] 0.011743119 0.007070903 0.009959893 0.012072650 0.016949153 0.011524823
##  [7] 0.009793814 0.010532151 0.006866197 0.004225352 0.007500000 0.006976744
## [13] 0.009183673 0.007337975 0.004342581 0.011356467
```

There is not much to work with in these data - only two covariates. However for illustrative purposes we will work through the steps of using a Metropolis-Hastings algorithm to compute the posterior for a Bayesian Poisson regression of the birth rate against the major-urban and North/ South Island indicators

Like logistic regression, Poisson regression is an examaple of a generalised linear model (GLM).

Model specification

The Model for the data is

$$\begin{aligned}[Y_i|N_i,\theta_i] &\overset{\text{indep}}{\sim} \text{Poisson}(\theta_i N_i),\ i=1,\ldots,12\\ \log(\theta_i) &= \beta_0+\beta_1\text{majurb}_i+\beta_2\text{North}_i\ \ i=1,\ldots,12.\end{aligned}$$

Here Y denotes births, and N denotes female population size (popsize) Under the model $\theta_i = E(Y_i/N_i)$ is the expected rate, which is often referred to as the underlying rate, in contrast to the directly observed rate Y_i/N_i (births divided by female population). To complete the model we need to specify a prior for the regression parameters $p(\beta_0,\beta_1,\beta_2)$. We assume the parameters are *a priori* independent, so $p(\beta_0,\beta_1,\beta_2)=p(\beta_0)p(\beta_1)p(\beta_2)$ and adopt normal priors for each parameter. This is equivalent to a multivariate normal prior for the vector $(\beta_0,\beta_1,\beta_2)$, with mean vector equal to the concatenation of the prior means for each parameter and a diagonal variance matrix with i^{th} diagonal element equal to the prior variance for the i^{th} parameter.

The parameters are best interpreted once exponentiated: $\exp(\beta_0)$ is the expected birth rate in a non -major urban region in the South Island $\exp(\beta_1)$ is the ratio of the expected rate in major urban regions compared to non major urban regions; holding Island constant. $\exp(\beta_2)$ is the ratio of the expected rate in a North Island region compared to a South Island region, holding major urban status constant.

Since all the regression parameters are interpretable once exponentiated we can use the `mkpriorreg()` function that we defined for specifying priors for logistic regression parameters for all parameters of our Poisson glm This is in contrast to logistic regression where it was the inverse-logit of the intercept parameter that was interpretable, requiring a different function for setting the prior, makepriorb0().

For convenience, we re-define the `makepriorreg()` function here although we could read in the previous version using the function `source()`

```
makepriorbreg <- function(low,high,credibility) {
  #returns prior mean and standard deviation corresponding to
  ##log-transformed parameter
  require(arm)
  log_low <- log(low)
  log_high <- log(high)

  priormean <- 0.5*(log_low + log_high)

  normal_quantile <- 1 - (1-credibility)/2
  priorsd <- (log_high - priormean)/qnorm(normal_quantile)
  outlist <- c(priormean,priorsd)
}
```

Now we use this function to specify the priors

```
priorb0 <- makepriorbreg(low=0.001,high=0.1,credibility=0.95)
prior_majurb <- makepriorbreg(low=0.5,high=3,credibility=0.9)
prior_north <- makepriorbreg(low=0.1,high=10,credibility=0.99)
prior.matrix <- rbind(priorb0,prior_majurb,prior_north)
prior.matrix
```

```
##                          [,1]      [,2]
## priorb0      -4.605170e+00 1.1748099
## prior_majurb  2.027326e-01 0.5446562
## prior_north   2.220446e-16 0.8939199
```

Combine the individual means and variances into a mean vector and diagonal variance matrix that will define the multivariate normal prior.

```
prior_mean <- prior.matrix[,1]
prior_sd <- prior.matrix[,2]
prior_variance <- diag(prior_sd^2) ##setting up the prior variance matrix
#check
prior_mean
```

```
##       priorb0  prior_majurb   prior_north
## -4.605170e+00  2.027326e-01  2.220446e-16
```

```
prior_variance
```

```
##          [,1]      [,2]      [,3]
## [1,] 1.380178 0.0000000 0.0000000
## [2,] 0.000000 0.2966504 0.0000000
## [3,] 0.000000 0.0000000 0.7990928
```

Preliminary analysis

To start the ball rolling we will fit a conventional glm model, ignoring the prior. This is useful for setting up a model object which can be used in log-likelihood and log-posterior functions.

```
birthsmodel1 <- glm(births ~ majurb + North,family=poisson(link="log"),
                    offset=log(popsize) )
display(birthsmodel1)
```

```
## glm(formula = births ~ majurb + North, family = poisson(link = "log"),
##     offset = log(popsize))
##             coef.est coef.se
## (Intercept) -4.67     0.07
## majurb      -0.43     0.06
## North        0.15     0.07
## ---
##   n = 16, k = 3
##   residual deviance = 25.6, null deviance = 92.8 (difference = 67.2)
```

```
beta_mle <- coef(birthsmodel1)
```

Set-up the log-likelihood and log-posterior functions

```
loglike_poissonreg <- function(beta,model){
  ##This function is vectorised so can cope with multiple settings of beta
   if (is.vector(beta)) {beta <- matrix(beta,nrow=1,ncol=length(beta))}
  ##beta assumed to be nsim by p matrix where nsim is number of beta
  # values   (i.e number of settings of beta) to  evaluated
  ##and p is the dimension of the parameter vector in the model
  ##model is a model object which contains the model formula, the data,   the model
 matrix and so on (e.g the output from a call to glm())

  X <- model.matrix(model)  ##extracts the design matrix
  Xb <- X %*% t(beta)

  Y <- model$y  #vector

  # turn Y into a matrix conformable with beta for  some subsequent
  #calculations

  Ymatrix <- matrix(Y,nrow=length(Y),ncol=nrow(beta),byrow=FALSE)
  logN <- model$offset
  logNmatrix <-  matrix(logN,nrow=length(logN),ncol=nrow(beta),byrow=FALSE)

  expect <- exp(Xb + logN)

  logl <- Ymatrix*log(expect) - expect -lfactorial(Ymatrix)
  ##logl returns a matrix with nrow= length(Y)=length(N)
  ##so the overall log-likelihood is the sum of these components
  return(colSums(logl))
  }

###evaluate the log-likelihood at a few values

loglike_poissonreg(beta=beta_mle,model=birthsmodel1)
```

```
## [1] -58.08406
```

```
loglike_poissonreg(beta=rep(0,length(beta_mle)),model=birthsmodel1)
```

```
## [1] -146834.3
```

```
loglike_poissonreg(beta= beta_mle+0.01,model=birthsmodel1)
```

```
## [1] -58.4539
```

```
###set-up the log-posterior function

logpost_poissonreg <- function(beta,priormean,priorvariance,model) {

logpost <- loglike_poissonreg(beta,model) + dmvnorm(beta,mean=priormean,
                                                  sigma=priorvariance,log=TRUE)
return(logpost)
}

##test this out
logpost_poissonreg(beta_mle,
                   priormean=prior_mean,
                   priorvariance=prior_variance,
                   model=birthsmodel1)
```

```
## [1] -60.96331
```

Build an approximation to the posterior.

We will use a multivariate normal approximation based on the posterior mode and an approximation to the posterior variance based on the curvature of the unnormalised log-posterior at the mode (negative inverse of the second derivative). We use the laplace function from the LearnBayes package to find the posterior mode and approximate variance.

```
logpost1 <- laplace(logpost_poissonreg,mode=beta_mle,
                    priormean=prior_mean,
                    priorvariance=prior_variance,
                    model=birthsmodel1)

str(logpost1)
```

```
## List of 4
##  $ mode    : Named num [1:3] -4.671 -0.419 0.15
##   ..- attr(*, "names")= chr [1:3] "(Intercept)" "majurb" "North"
##  $ var     : num [1:3, 1:3] 0.00548 -0.00214 -0.00447 -0.00214 0.00317 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "(Intercept)" "majurb" "North"
##   .. ..$ : chr [1:3] "(Intercept)" "majurb" "North"
##  $ int     : num -67.3
##  $ converge: logi TRUE
```

```
mean_approx <- logpost1$mode
var_approx <- logpost1$var
```

The multivariate normal approximation to the variance is defined by mean_approx and var_approx.

Setting up the Metropolis-Hastings algorithm.

We need to define the variance for the staring distribution which is an over-dispersed (i.e bigger variance) version of our approximation to the posterior

```
var_start <- 1.5*var_approx
```

We also need to define the jumping distribution. We will use a normal jumping density, centred on the current value with variance computed according to the usual Gelman et al recommendation. Since we are using a normal jumping density (which is symmetric) this is really a Metropolis algorithm.

```
var_jump <- ((2.4^2)/length(mean_approx)) * var_approx

##specify number of chains, length of chains and burn-in period
nchains <- 5
nsim <- 2000
burnin <- 1000

##structures for storing output
store_beta <- array(dim=c(length(mean_approx),nchains,nsim))
store_accept <- matrix(nrow=nsim,ncol=nchains)
```

Run the Metropolis algorithm

```
for (j in 1:nchains){ ##loop over chains
  ##generate a starting value

  old_beta <- rmvnorm(n=1,mean=mean_approx,sigma=var_start)
 for (i in 1:nsim) { ##loop over simulations
   ###generate proposal
   prop_beta <- rmvnorm(n=1,mean=old_beta,sigma=var_jump)
  ##compare log unnormalised posterior at proposed and old value
  logpost_prop <-  logpost_poissonreg(prop_beta,
                              priormean=prior_mean,
                              priorvariance=prior_variance,
                              model=birthsmodel1)

  logpost_old      <-  logpost_poissonreg(old_beta,
                                     priormean=prior_mean,
                                     priorvariance=prior_variance,
                                     model=birthsmodel1)

  logrMH <- logpost_prop - logpost_old

  ###decide on acceptance or rejection
  logU <- log(runif(1))
  if (logU <= logrMH)  {
         old_beta <- prop_beta
         store_accept[i,j] <- 1
     } else {
       store_accept[i,j] <- 0
        }

  #store current value of beta
  store_beta[,j,i] <- old_beta

 } #end loop over simulations
} #end loop over chains
```

Check acceptance rates:

```
colMeans(store_accept)
```

```
## [1] 0.3360 0.3140 0.3045 0.3215 0.3235
```

Check convergence

```
npos <- nsim - burnin  ##size of  posterior sample in each chain

n1 <-  round(npos/2) # size of chunk1

#set up vectors for storing
rhat <- vector(mode="numeric",length=length(mean_approx))
neff  <- vector(mode="numeric",length=length(mean_approx))
#for storing the statistics for each parameter.
k <- 1
for (k in 1:length(mean_approx)) {  #looping over parameters

  #subset out post burn-in sample for  kth parameter
  #set this up  as npos by nchains matrix

  betak <- t(store_beta[k,1:nchains,((burnin+1):nsim)] )

  ##chunk each chain into two pieces and reassemble the matrix -
  ## for  convergence checking

  chunk1 <- betak[1:n1,]
  chunk2 <- betak[(n1+1):npos,]

  betak_chunked <- cbind(chunk1,chunk2)
  #str(betak_chunked)

  chainmeans <- colMeans(betak_chunked)
  withinsd <- apply(betak_chunked,MARGIN=2,FUN="sd")
  betweensd <- sd(chainmeans)
  B = (npos/2)*betweensd^2
  W = mean(withinsd^2)
  varplus <- ((npos-1)/npos) * W + (1/npos)*B
  rhat[k] <- sqrt(varplus/W)

  #set-up betak_chunked  as  MCMC object

  betak.mcmc <-  coda::mcmc(betak_chunked)
  #str(betak.mcmc)

  neff[k] <- sum(coda::effectiveSize(betak.mcmc))
}
rhat
```

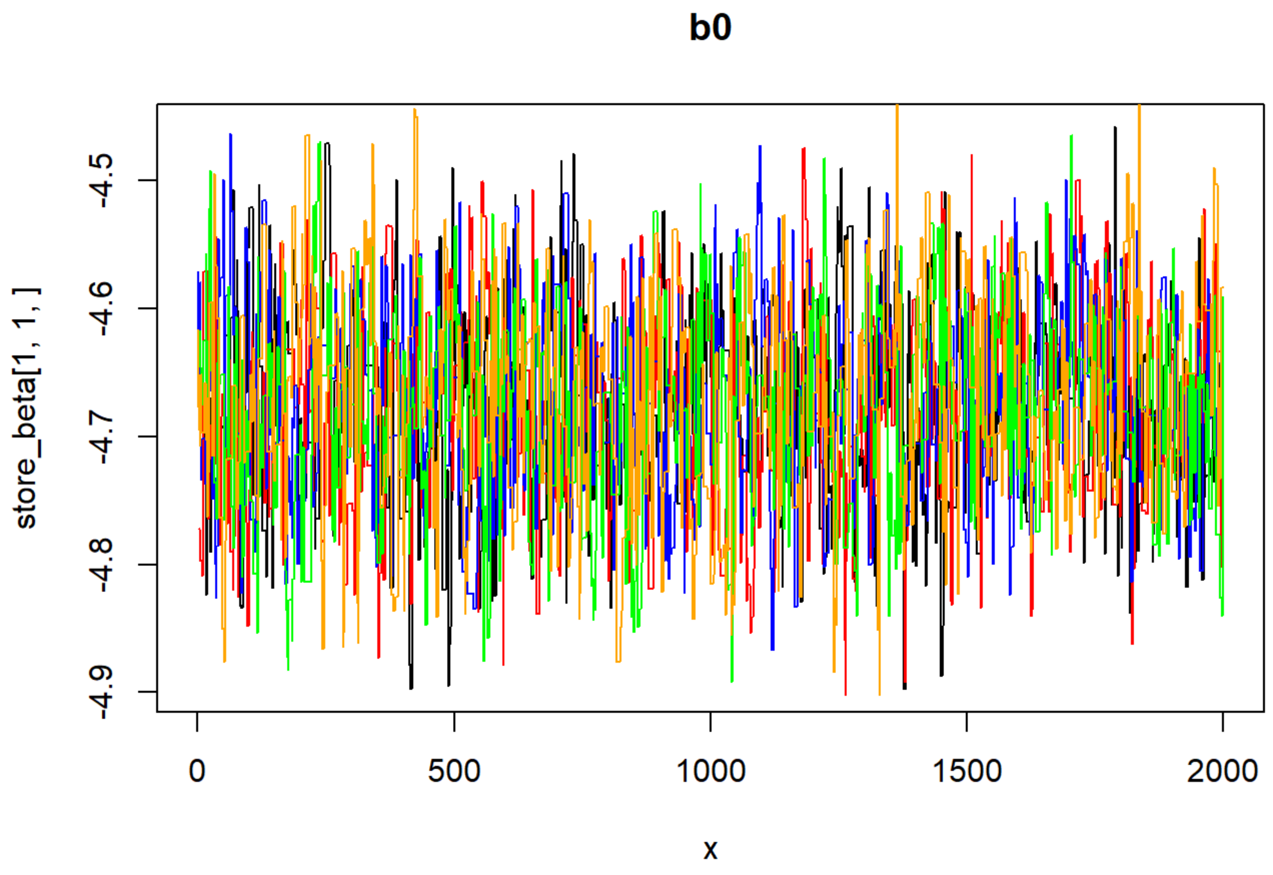
```
## [1] 1.004361 1.003792 1.007180
```

```
neff
```

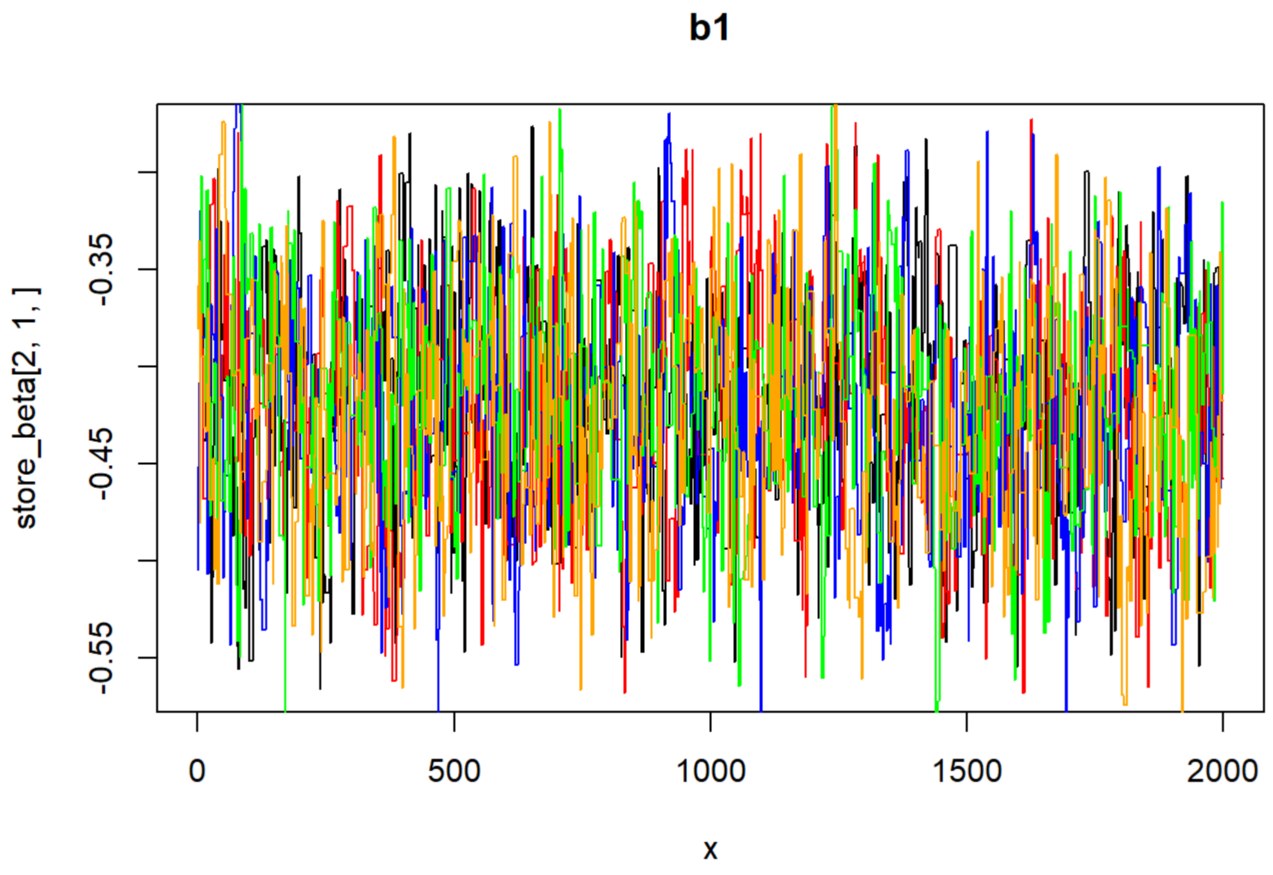
```
## [1] 565.2899 510.2379 520.3874
```

We will now look at the traceplots. We could just plot the post burn-in samples, but will examine the full iteration history.

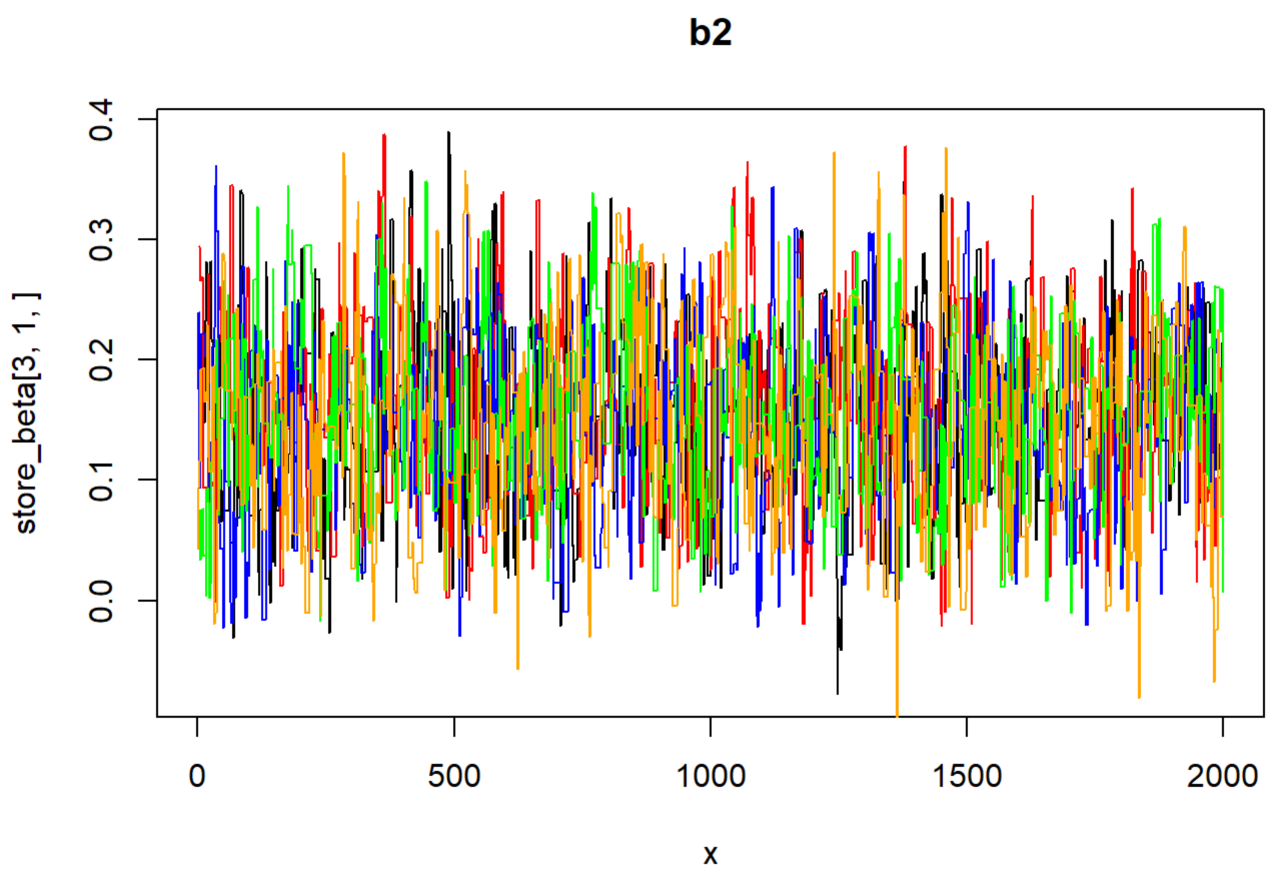
```
x <- seq(from=1,to=nsim,by=1)
#plots for intercept
plot(x,store_beta[1,1,],type="l",main= "b0") #parameter 1, chain1 all simulations

lines(x,store_beta[1,2,],col="red")
lines(x,store_beta[1,3,],col="blue")
lines(x,store_beta[1,4,],col="green")
lines(x,store_beta[1,5,],col="orange")
```



```
#plots for beta1 - effects of major urban regions
plot(x,store_beta[2,1,],type="l",main= "b1")
lines(x,store_beta[2,2,],col="red")
lines(x,store_beta[2,3,],col="blue")
lines(x,store_beta[2,4,],col="green")
lines(x,store_beta[2,5,],col="orange")
```



```
#plots for beta2 - effects of North Island
plot(x,store_beta[3,1,],type="l",main= "b2")
lines(x,store_beta[3,2,],col="red")
lines(x,store_beta[3,3,],col="blue")
lines(x,store_beta[3,4,],col="green")
lines(x,store_beta[3,5,],col="orange")
```



Based on the Rhat statistics and the traceplots, the chains certainly seem to have converged. However the effective Monte Carlo sample sizes are not spectacular. Longer chains may be required to get highly accurate tail probabilities

Posterior inference

Since the chains have converged we will move to inference

```
 #Subset out the post burn-in sample.
postsample <- store_beta[,,(burnin+1):nsim]

###since all parameters are interpretable once exponentiated
## we exponentiate the entire postsample

exppost <- exp(postsample)

quantile(exppost[1,,],probs=c(0.025,0.5,0.975))
```

```
##        2.5%         50%       97.5% 
## 0.008119144 0.009329523 0.010716539
```

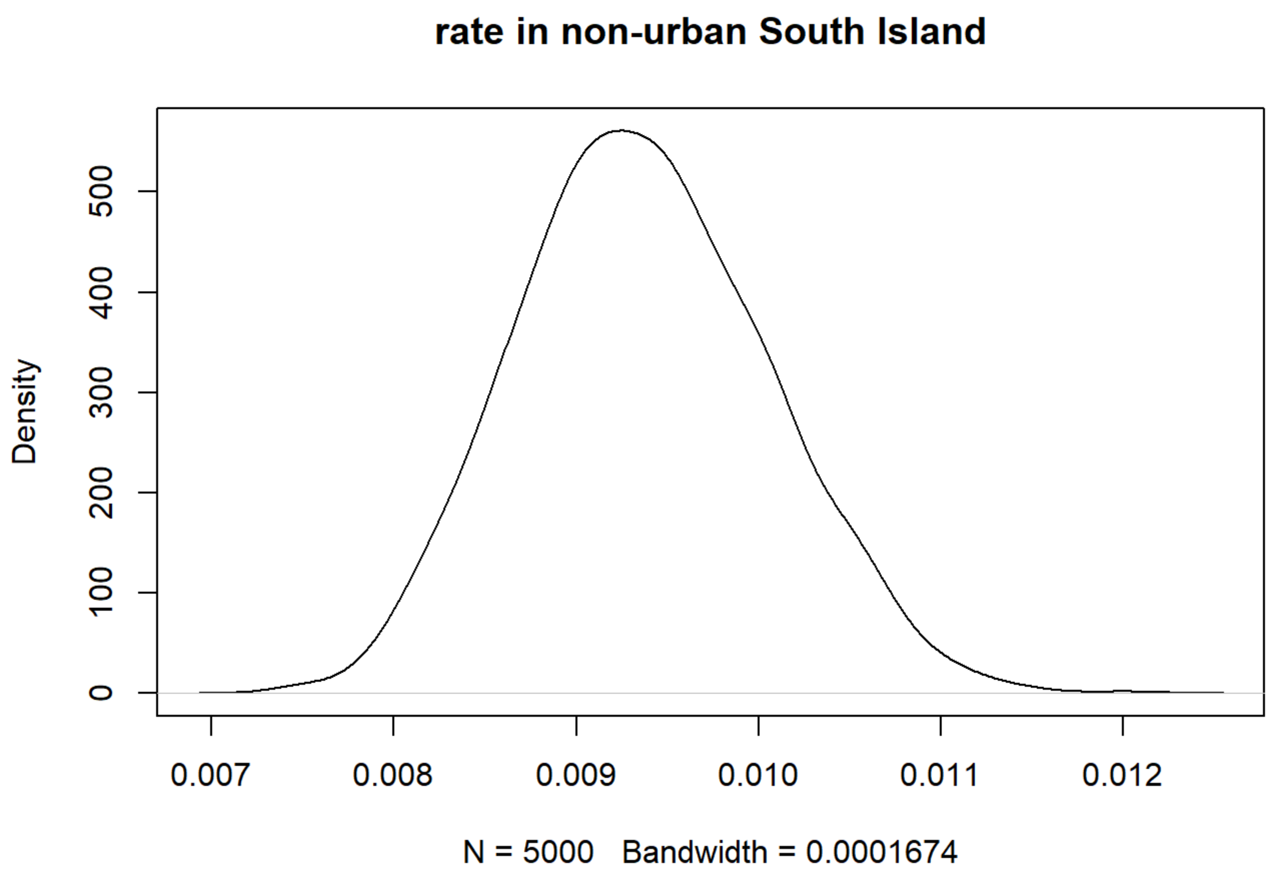
```
quantile(exppost[2,,],probs=c(0.025,0.5,0.975))
```

```
##      2.5%       50%     97.5% 
## 0.5891598 0.6554885 0.7301577
```

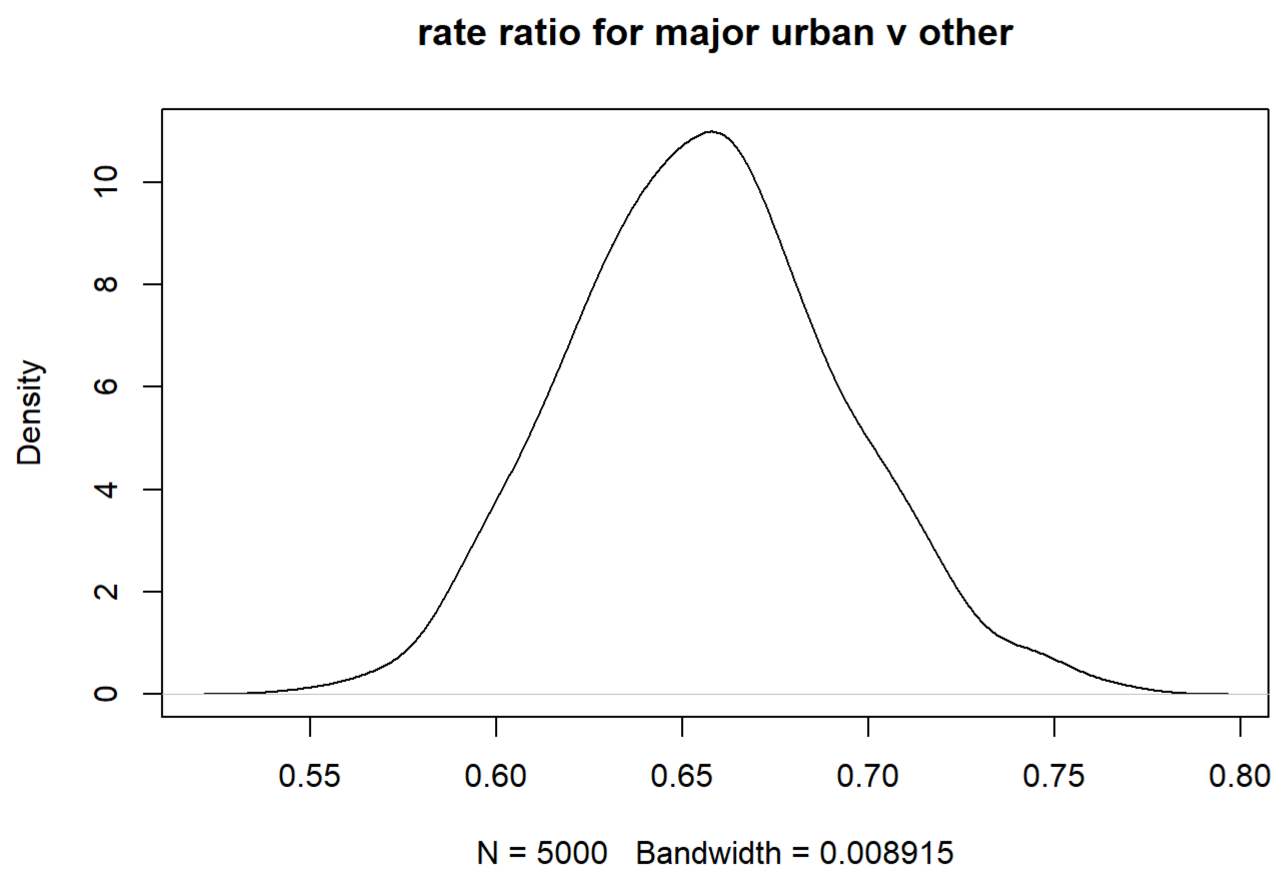
```
quantile(exppost[3,,],probs=c(0.025,0.5,0.975))
```

```
##      2.5%       50%     97.5% 
## 1.013194 1.170524 1.336541
```

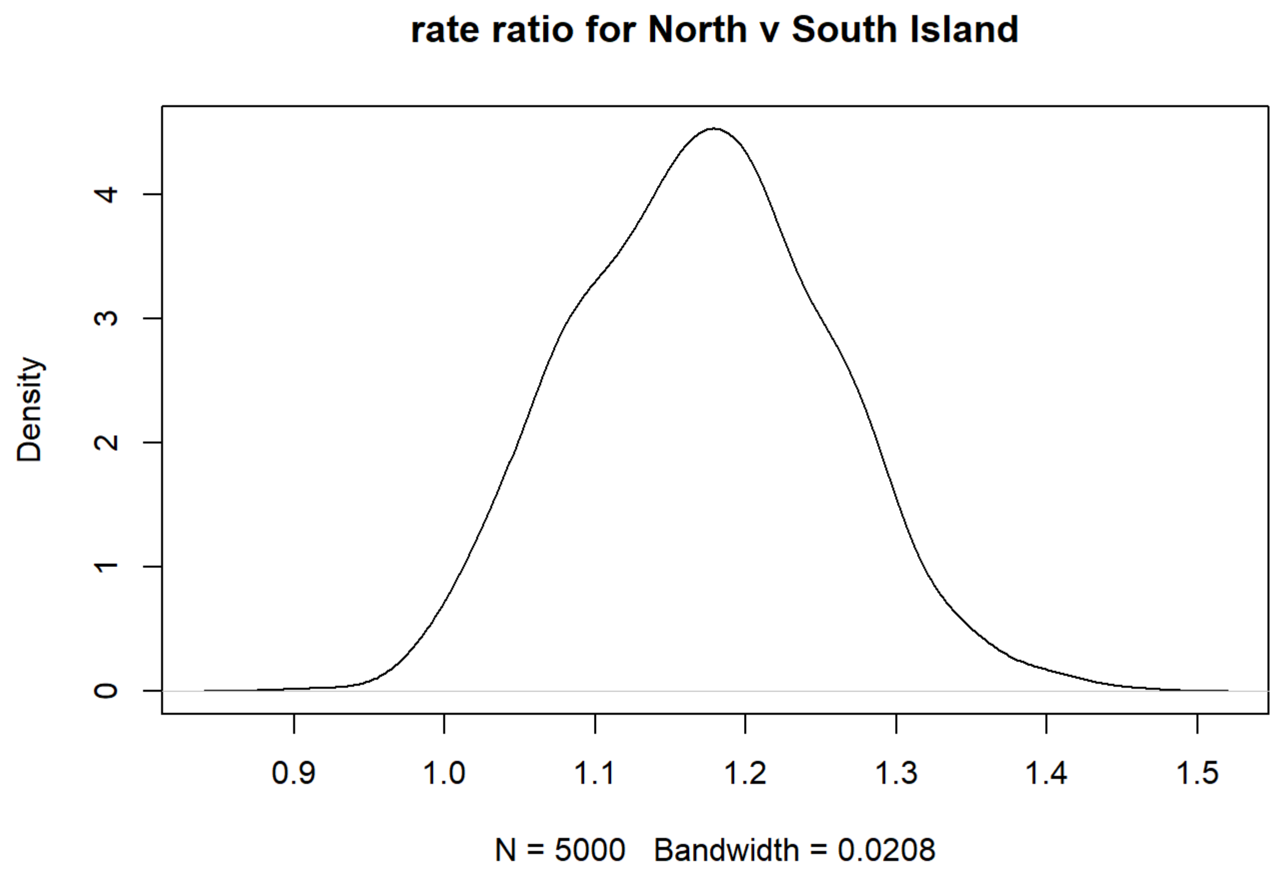
```
##density plots
plot(density(exppost[1,,],adjust=1.5),main="rate in non-urban South Island")
```



```
plot(density(exppost[2,,],adjust=1.5),main="rate ratio for major urban v other")
```



```
plot(density(exppost[3,,],adjust=1.5),main="rate ratio for North v South Island")
```



Overall it looks like there is good evidence that birth rates are markedly lower in major urban areas. There is also evidence that birth rates are higher in the North Island.