ASI Coursework

Margaret Duff

30 November 2018

Analysis using Metropolis Hastings

To make life easier later we split rats into the two categories based on their status

```
split_rats <- split(rats, rats$status)
tumour=as.data.frame(split_rats[[2]]); dead=as.data.frame(split_rats[[1]])
```

Let T_i be the follow up time to tumor appearance in rat i. Assume the following probability model

$$T_i \sim \text{Weibull}(\text{shape} = 1/\sigma, \text{scale} = \exp(\eta_i)), \qquad \eta_i = \beta_0 + \beta_1 x_i$$

where

$$x_i = \begin{cases} 1 & \text{rat } i \text{ received treatment} \\ 0 & \text{rat } i \text{ received control} \end{cases}$$

Assume that $\{T_1, \ldots, T_n\}$ are independent random variables. The survival function of a Weibull with shape $a > 0$ and scale $b > 0$ is given by

$$S(t|a,b) = P(T > t|a,b) = \exp\left(-\left(\frac{t}{b}\right)^a\right), \qquad t > 0$$

We assume that censored observations (`status=0`) do not contribute to the likelihood with a factor equal to the density of T at the observed t_i but with a factor equal to the survival function evaluated at the observed t_i.

We use a random walk Metropolis-Hastings algorithm to sample from the posterior distribution of $\theta = (\beta_0, \beta_1, \log(\sigma), \log(\sigma_b))^T$ using the model above. We follow a Bayesian estimation procedure and specify the following prior distributions on the unknown parameters: β_0 β_1 and $\log(\sigma)$ are independent and following uniform (improper) priors while σ_b is also independent of β_0 β_1 and $\log(\sigma)$ and follows a prior exponential distribution with rate 5. This gives us the following log posterior function

```
log.post <- function(beta0, beta1, log_sigma,log_sigma_b,b_tumour, b_dead,  tumour, dead) {
  log_pi0_beta0=log(1);  log_pi0_beta1=  log(1);  log_pi0_log_sigma=log(1);  log_pi0_log_sigma_b= log(5*exp(-5*e
    log_pi0_b_tumour=log(dnorm(b_tumour, mean=0, sd=exp(log_sigma_b)));  log_pio_b_dead=log(dnorm(b_dead, mean=0
  log_pi0<- log_pi0_beta0+log_pi0_beta1+log_pi0_log_sigma+log_pi0_log_sigma_b+sum(log_pio_b_dead)+sum(log_pi0_b_
  log.lik<-sum(log (dweibull(tumour$time, 1/exp(log_sigma), exp(beta0+beta1*tumour$rx+b_tumour))))+sum(log (pwei
  return(log.lik+log_pi0)# now the log posterior = log likelihood +log prior
}
```

We use a normal centered on the current values as the proposal for β_0, β_1, $\log(\sigma_b)$, $\log(sigma)$ and each element of b I will also use a symmetric multivariate normal distribution centered on the current values for the values of b. I will need to tune the proposal standard deviations to get appropriate acceptance rates, aiming for about 25%.

The initial value is important since we would like to start in a region of the parameter space with high density as otherwise, that is, if the posterior density is extremely low for the initial value, it will take us a long time (a large number of generated values) to reach the area of high posterior density and therefore a long time to start sampling from the stationary distribution of the Markov chain. Thus we choose as our initial conditions the maximum likelihood estimators calculated earlier.

We choose a burn in period of 10,000 but this can always be changed later.

```
nsteps=100000 ; burn.in=10000
MH<-function(beta0_0,beta1_0, log_sigma_0, log_sigma_b0, b_tumour0, b_dead0, sigma_beta0, sigma_beta1, sigma_log
  accept <- rep(0,3) # set up locations to store values at each step
  beta0 <- rep(0,nsteps) ;  beta1=rep(0,nsteps);  log_sigma=rep(0,nsteps);  log_sigma_b=rep(0,nsteps)
  b_tumour=matrix(0,nsteps,length(tumour$rx)) ;  b_dead=matrix(0,nsteps,length(dead$rx)) # set up locations to s
  beta0[1] <- beta0_0 ;  beta1[1]=beta1_0;  log_sigma[1]=log_sigma_0;  log_sigma_b[1]=log_sigma_b0;  b_tumour[1,
lp0 <- log.post(beta0_0,beta1_0, log_sigma_0, log_sigma_b0,b_tumour0, b_dead0,tumour, dead) # calculate log post
for( i in 2:nsteps){ #MH loop
current_beta0=beta0[i-1] ;current_beta1=beta1[i-1];current_log_sigma=log_sigma[i-1];  current_log_sigma_b=log_si
proposed_log_sigma_b=current_log_sigma_b+rnorm(1,0,sigma_log_sigma_b)#update sigma_b
lp1 <- log.post(current_beta0,current_beta1, current_log_sigma, proposed_log_sigma_b,current_b_tumour, current_b
acc <- exp(min(0,lp1-lp0))
```

```
if (runif(1)>=acc| !is.finite(acc)){#reject
  b_tumour[i,] <- current_b_tumour ;  b_dead[i,]=current_b_dead;  beta0[i] <- current_beta0;  beta1[i]=current_b
  lp1<- lp0 ## Return to the 'old' log posterior
}else {#accept
  accept[1]=accept[1]+1 # keep track of number of acceptances
  log_sigma_b[i]=proposed_log_sigma_b #store found values
  lp0 <- lp1 ## uldate old log posterior to the new one
    proposed_b_tumour=current_b_tumour + rnorm( length(tumour$rx), mean=0, sd=sigma_b)#update b
    proposed_b_dead=current_b_dead+ rnorm( length(dead$rx), mean=0, sd=sigma_b)#update b
    lp1 <- log.post(current_beta0,current_beta1, current_log_sigma, proposed_log_sigma_b,proposed_b_tumour, prop
  acc <- exp(min(0,lp1-lp0))
  if (runif(1)>=acc | !is.finite(acc)){#reject
  b_tumour[i,] <- current_b_tumour ;  b_dead[i,]=current_b_dead;  beta0[i] <- current_beta0;  beta1[i]=current_b
  lp1<- lp0 ## Return to previous log posterior
}else {#accept
          accept[2]=accept[2]+1 # keep track to calculate acceptance rates
          b_tumour[i,] <- proposed_b_tumour;  b_dead[i,]=proposed_b_dead;  #store values
          lp0 <- lp1 ## update log posterior
  proposed_beta0=current_beta0+rnorm(1,0,sigma_beta0); proposed_beta1=current_beta1+rnorm(1,0,sigma_beta1)
  proposed_log_sigma=current_log_sigma+rnorm(1,0,sigma_log_sigma)#update remaining paramaters
  lp1=log.post(proposed_beta0,proposed_beta1, proposed_log_sigma, proposed_log_sigma_b,proposed_b_tumour, propos
  acc <- exp(min(0,lp1-lp0))
  if (runif(1)>=acc| !is.finite(acc)){#reject
  beta0[i] <- current_beta0 ;  beta1[i]=current_beta1;  log_sigma[i]=current_log_sigma #store values
  lp1<- lp0 ## Return to previous log posterior
}else {#accept
    accept[3]=accept[3]+1 # keep track to calculate acceptance rates
  beta0[i] <- proposed_beta0;  beta1[i]=proposed_beta1;  log_sigma[i]=proposed_log_sigma#store values
  lp0=lp1 # update log posterior
}}}}
  list(beta0=beta0, beta1=beta1, log_sigma_b=log_sigma_b, log_sigma=log_sigma,ar_outer=accept[1]/nsteps, ar_midd
}
mh=MH(5.0188886, -0.2376741, -1.3687252,-1.5976068, rep(0, length(tumour$rx)),rep(0, length(dead$rx)),0.1,0.1,0.
```

For tuning the proposal distribution we use run lengths of about 10000. 100000 would be better for the final run. The aim is for an acceptance rate of approximately 25%.

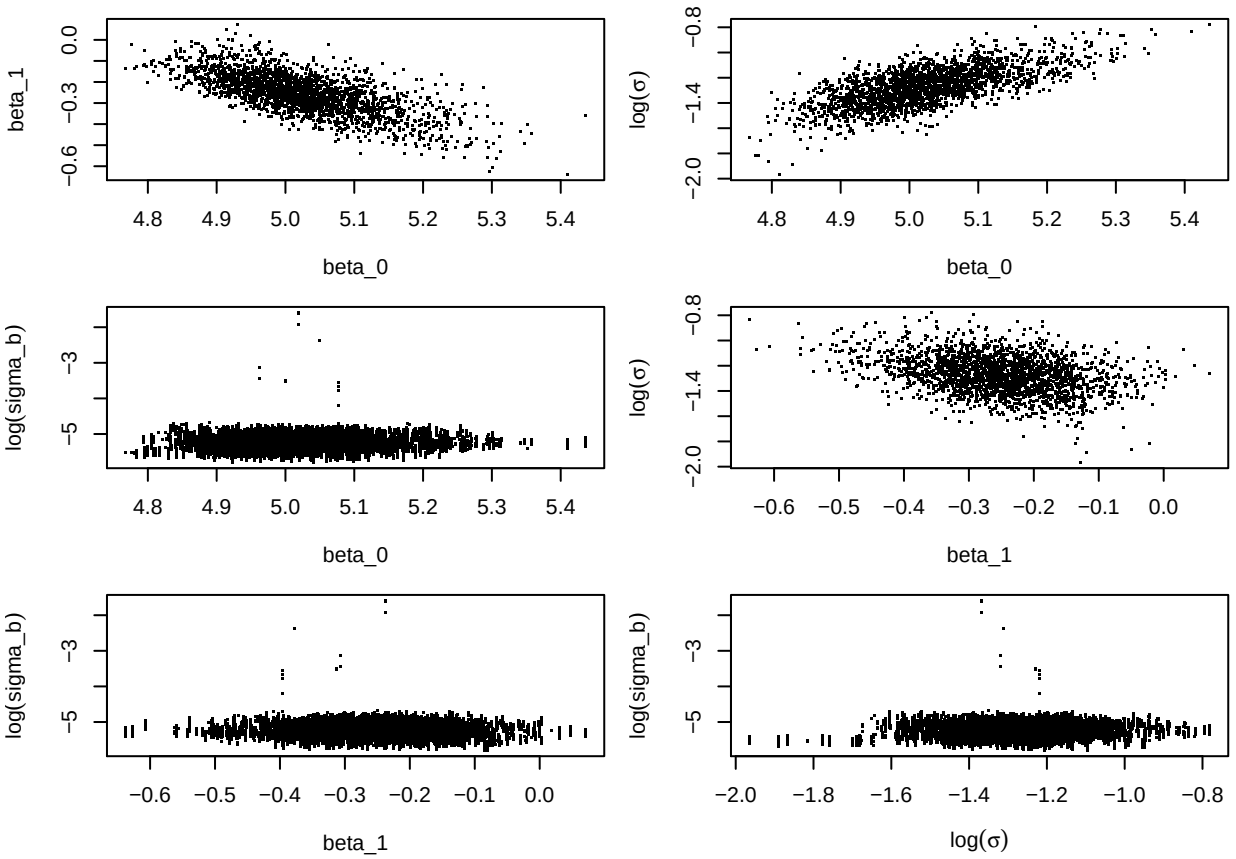
```
mh$ar_outer;mh$ar_middle;mh$ar_inner
```

```
## [1] 0.33462
```

```
## [1] 0.2376128
```

```
## [1] 0.281474
```

We check for correlation between the parameters by plotting graphs.



We note the elliptical shaped graphs especially for the plots of $\beta_0, \beta_1, \log(\sigma), \log(\sigma_b)$. This suggests that the posterior density for θ is highly non independent, and independent jumps for each component will give slow mixing. We consider using a shrunken version of the co-variance, found using the results above, as the basis for proposing multivariate normal jumps in a random walk i.e. $\theta_i \sim N(\theta_{i-1}, \lambda * \Sigma)$, where $\lambda > 0$ and we can tune. To find the co-variance matrix, Σ:

```
library(mvtnorm)
mu=c(mean(mh$beta0[-(burn.in)]),mean(mh$beta1[-(burn.in)]), mean(mh$log_sigma[-(burn.in)]) )
s11=cov(mh$beta0[-(burn.in)], mh$beta0[-(burn.in)]);s12=cov(mh$beta0[-(burn.in)], mh$beta1[-(burn.in)])
s13=cov(mh$beta0[-(burn.in)], mh$log_sigma[-(burn.in)]);s22=cov(mh$beta1[-(burn.in)], mh$beta1[-(burn.in)])
s23=cov(mh$beta1[-(burn.in)], mh$log_sigma[-(burn.in)]);s33=cov(mh$log_sigma, mh$log_sigma)
covariance= matrix(c(s11,s12,s13,s12,s22,s23,s13,s23,s33), nrow=3, byrow=TRUE)
```

With this new proposal distribution for $(\beta_0, \beta_1, \log(\sigma)$ we get a new MH sampler identical to the previous one, except that when we propose new values in the inner most MH chain we use this excerpt of code:

```
proposed= rmvnorm(1, mean=c(current_beta0, current_beta1, current_log_sigma), lambda*covariance) #update remaini
proposed_beta0=proposed[1];  proposed_beta1=proposed[2];  proposed_log_sigma=proposed[3]
```

Using the new MH algorithm and tuning we get

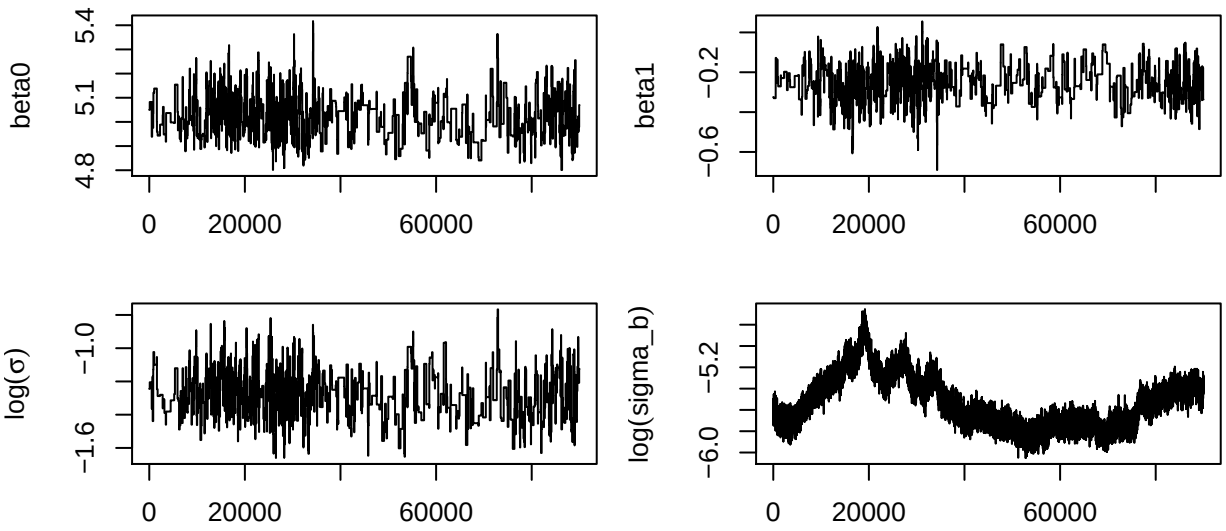
```
mh=MH2(5.0188886, -0.2376741, -1.3687252,-1.5976068, b_tumour0 =  rep(0, length(tumour$rx)),b_dead0= rep(0, leng
mh$ar_outer;mh$ar_middle;mh$ar_inner
```

```
## [1] 0.33319
```

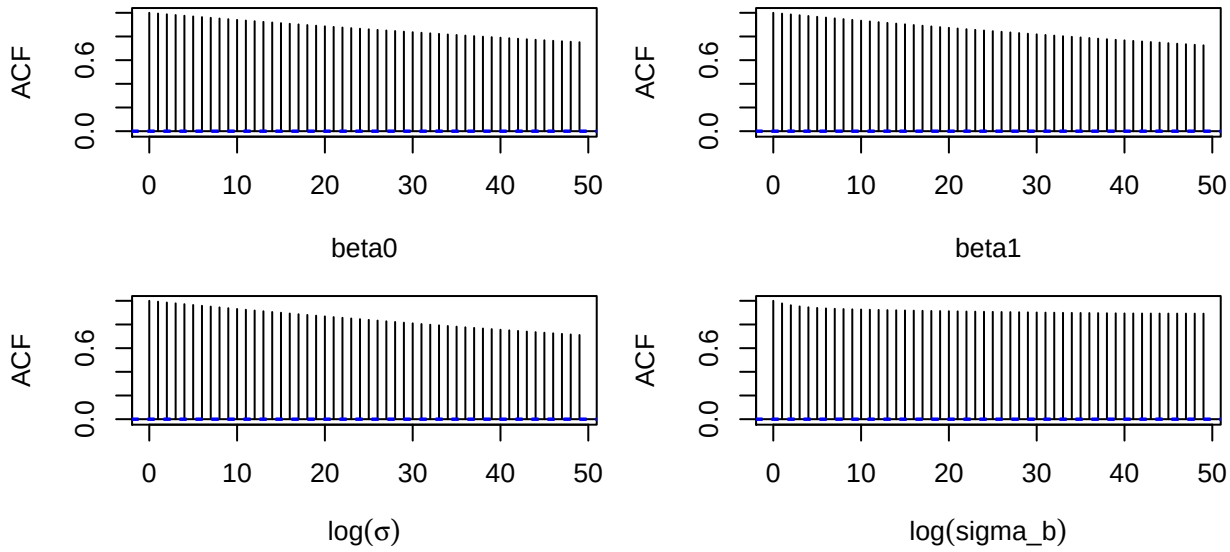
```
## [1] 0.1327771
```

```
## [1] 0.2486438
```

We now check that the Markov chain is behaving as we expect. Firstly, the trace plots show some good mixing within the parameters



The auto-correlation function plots are also decreasing suggesting that the Markov chain is converging.



We can also get some idea of the effective sample size for each of our parameters

```
n.eff <- c(0,0,0,0)
autocor <- acf(mh$beta0[-(burn.in)],plot=FALSE); t.eff <- 2*sum(autocor[[1]]) - 1; n.eff[1] <- nsteps/t.eff
autocor <- acf(mh$beta1[-(burn.in)],plot=FALSE); t.eff <- 2*sum(autocor[[1]]) - 1; n.eff[2] <- nsteps/t.eff
autocor <- acf(mh$log_sigma[-(burn.in)],plot=FALSE); t.eff <- 2*sum(autocor[[1]]) - 1;n.eff[3] <- nsteps/t.eff
autocor <- acf(mh$log_sigma_b[-(burn.in)],plot=FALSE);t.eff <- 2*sum(autocor[[1]]) - 1 ;n.eff[4] <- nsteps/t.eff
n.eff
```

```
## [1] 1166.448 1186.385 1196.091 1106.991
```

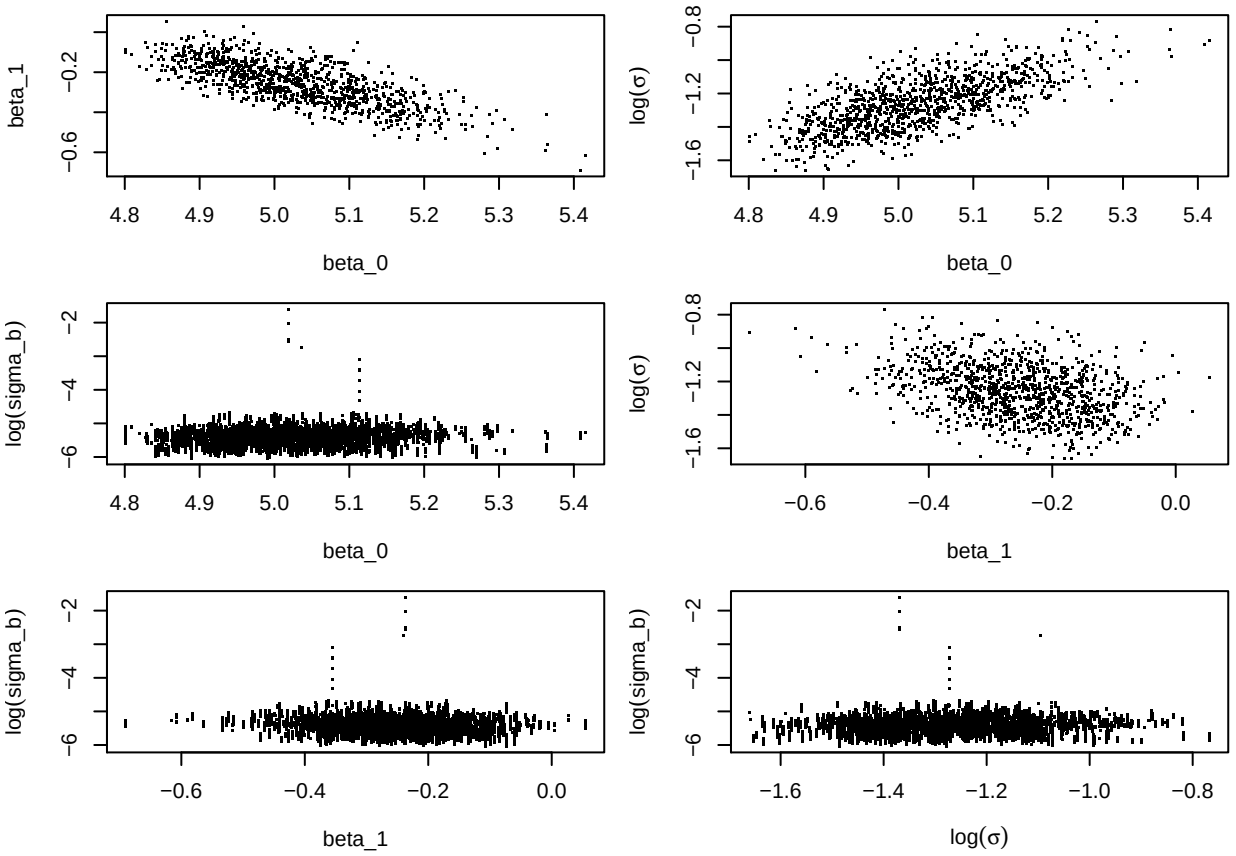
We will also obtain an independent sub-sample of our chain and then split it into two sub-samples. Using the Kolmogorov-Smirnov test we chance to see that they come from the same distribution and hence that the Markov chain has converged.

```
k1=ks.test(mh$beta0[-1000:-500], mh$beta0[-500]);
k2=ks.test(mh$beta1[-1000:-500], mh$beta1[-500]);
k3=ks.test(mh$log_sigma_b[-1000:-500], mh$log_sigma_b[-500]);
k4=ks.test(mh$log_sigma[-1000:-500], mh$log_sigma[-500]);
c(k1$p.value,k2$p.value,k3$p.value,k4$p.value)
```

```
## [1] 1.0000000 0.9990153 0.9944827 0.9836882
```

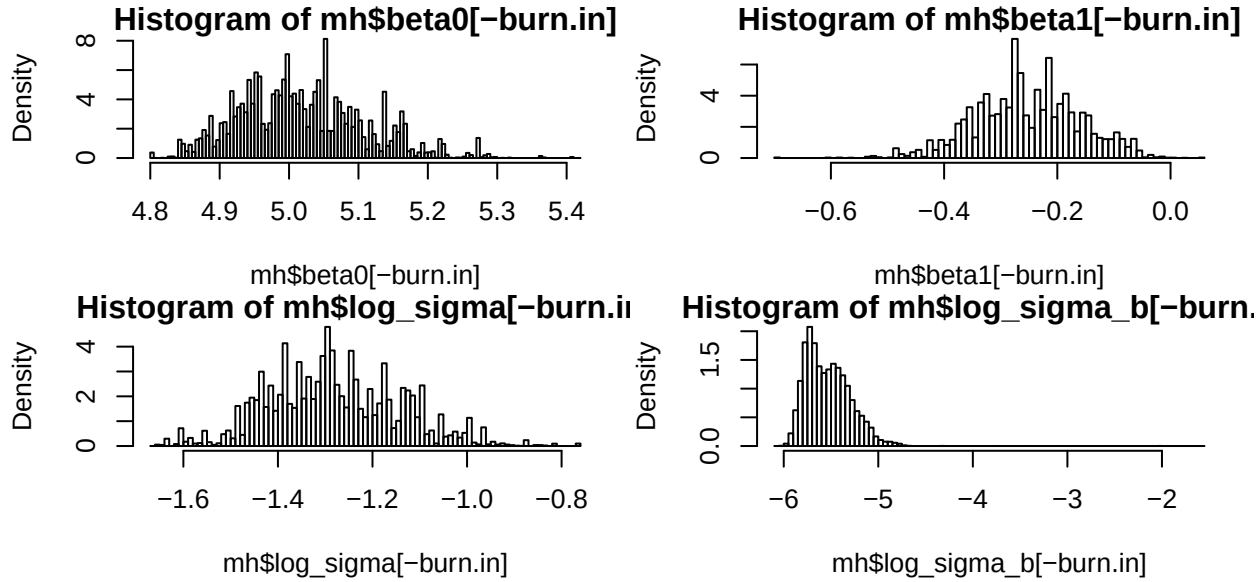
We see that in all cases the p-values are large, and thus there is not significant evidence that the two sub-samples are from different distributions. Hence we conclude that the Markov chain has converged.

Again, we investigate the correlation between the parameters by plotting a random sample of the iterations retained after discarding the burn in period .



It looks like we need a more sophisticated proposal to deal with the correlation of $\log(\sigma_b)$ and the other 3 parameters.

We also investigate the shape of the marginal posterior densities of the parameters by plotting histograms of the retained values



The long non symmetric tails on some of the marginal posterior densities suggesting such as on the $\log(\sigma_b)$ graph again suggest a more sophisticated proposal would be useful. We compute a 95% posterior probability interval for the intervention effect β_1.

```
quantile(mh$beta1[-(burn.in)], c(.025, 0.975))#95% condfidence interval for beta_1
```

```
##        2.5%       97.5%
## -0.42987051 -0.07346982
```

We conclude that 0 is not contained in the 95% posterior probability interval for the intervention effect, suggesting that the intervention has a statistically significant effect.AS per the analysis in part 2,the confidence values for β_1 being greater than zero suggest that the treatment is having a poitive effect. This is in contrast to previous results and thus perhaps suggests that more

investigation is required.