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.

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

```
# function to compute log posterior
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*ex
    log_pi0_b_tumour=log(dnorm(b_tumour, mean=0, sd=exp(log_sigma_b)));  log_pio_b_dead=log(dnorm(b_dead, m
  log_pi0<- log_pi0_beta0+log_pi0_beta1+log_pi0_log_sigma+log_pi0_log_sigma_b+sum(log_pio_b_dead)+sum(log_p
  log.lik<-sum(log (dweibull(tumour$time, 1/exp(log_sigma), exp(beta0+beta1*tumour$rx+b_tumour))))+sum(log
  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, sig
# set up locations to store values at each step
```

```
  accept <- rep(0,3)
  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 intial values
  beta0[1] <- beta0_0 ;  beta1[1]=beta1_0;  log_sigma[1]=log_sigma_0;  log_sigma_b[1]=log_sigma_b0;  b_tum
lp0 <- log.post(beta0_0,beta1_0, log_sigma_0, log_sigma_b0,b_tumour0, b_dead0,tumour, dead) # calculate lo
for( i in 2:nsteps){ #MH loop
# set current values
    current_beta0=beta0[i-1] ;current_beta1=beta1[i-1];current_log_sigma=log_sigma[i-1];  current_log_sigm
#update sigma_b
proposed_log_sigma_b=current_log_sigma_b+rnorm(1,0,sigma_log_sigma_b)
lp1 <- log.post(current_beta0,current_beta1, current_log_sigma, proposed_log_sigma_b,current_b_tumour, cur
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]=cur
  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
  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]=cur
  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
  #update remaining paramaters
  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)
  lp1=log.post(proposed_beta0,proposed_beta1, proposed_log_sigma, proposed_log_sigma_b,proposed_b_tumour,
  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, a
}
mh=MH(4.2 , 0.14, -1.35,-1.5, rep(0, length(tumour$rx)),rep(0, length(dead$rx)),0.1,0.1,0.1,0.2,0.001, nst
```

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
```

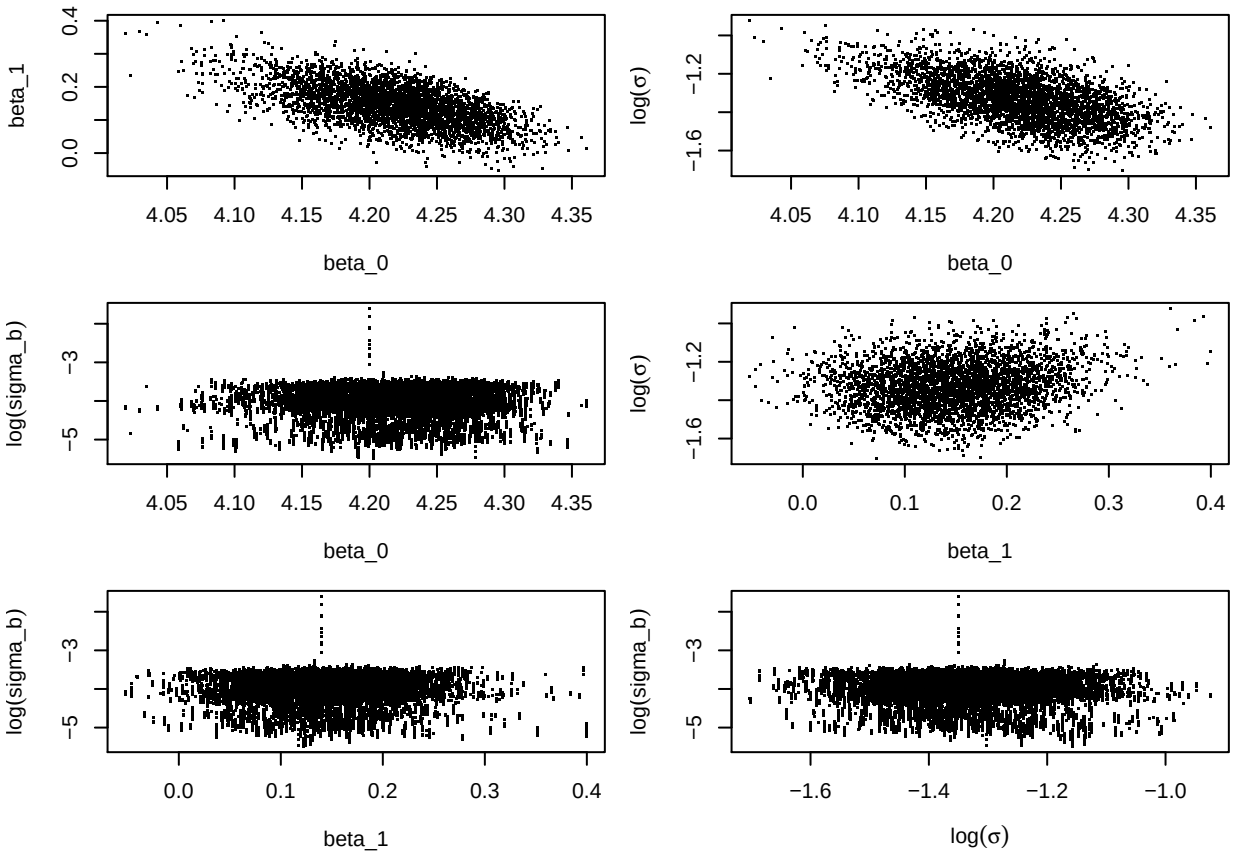
```
## [1] 0.33394
```

```
## [1] 0.6835659
```

```
## [1] 0.1637534
```

We check for correlation between the parameters by plotting graphs

```
par(mfrow=c(3,2),mar=c(4,4,1,1))
samp<-sample((1:nsteps)[-(burn.in)],nsteps/2) # For visualization purposes we take a random sample of the
plot(mh$beta0[samp],mh$beta1[samp],xlab=expression(beta_0),ylab=expression(beta_1),pch=".",cex=0.1)
plot(mh$beta0[samp],mh$log_sigma[samp],xlab=expression(beta_0),ylab=expression(log(sigma)),pch=".",cex=0.1
plot(mh$beta0[samp],mh$log_sigma_b[samp],xlab=expression(beta_0),ylab=expression(log(sigma_b)),pch=".",cex
plot(mh$beta1[samp],mh$log_sigma[samp],xlab=expression(beta_1),ylab=expression(log(sigma)),pch=".",cex=0.1
plot(mh$beta1[samp],mh$log_sigma_b[samp],xlab=expression(beta_1),ylab=expression(log(sigma_b)),pch=".",cex
plot(mh$log_sigma[samp],mh$log_sigma_b[samp],xlab=expression(log(sigma)),ylab=expression(log(sigma_b)),pch
```



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 proposal distribution we get a new MH sampler, again we tune the parameters the best we can.

```
nsteps=100000; burn.in=1000
MH2<-function(beta0_0,beta1_0, log_sigma_0, log_sigma_b0, b_tumour0, b_dead0, lambda, sigma_log_sigma_b,
  accept <- rep(0,3) # for acceptance values of outer, middle and inner chain
  beta0 <- rep(0,nsteps) ;  beta1=rep(0,nsteps)# set up locations to store values at each step
  log_sigma=rep(0,nsteps);  log_sigma_b=rep(0,nsteps)# set up locations to store values at each step
  b_tumour=matrix(0,nsteps,length(tumour$rx)) ;  b_dead=matrix(0,nsteps,length(dead$rx)) # set up location
  beta0[1] <- beta0_0 ;  beta1[1]=beta1_0;  log_sigma[1]=log_sigma_0;  log_sigma_b[1]=log_sigma_b0;  b_tum
lp0 <- log.post(beta0_0,beta1_0, log_sigma_0, log_sigma_b0,b_tumour0, b_dead0,tumour, dead) # log posterio
for( i in 2:nsteps){ #MH loop
     current_beta0=beta0[i-1];    current_beta1=beta1[i-1]# set current values
    current_log_sigma=log_sigma[i-1];    current_log_sigma_b=log_sigma_b[i-1]# set current values
    current_b_tumour=b_tumour[i-1,];    current_b_dead=b_dead[i-1,]# set current values
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, curr
    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]=curr
  lp1<- lp0##make sure correct log posteriors are stored
}else {#accept
  accept[1]=accept[1]+1 # keep track to caluclate acceptance rates
  log_sigma_b[i]=proposed_log_sigma_b #store found values
  lp0 <- lp1 ##make sure correct log posteriors are stored
    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
  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]=curre
  lp1<- lp0 ##make sure correct log posteriors are stored
}else {#accept
  accept[2]=accept[2]+1# keep track to caluclate acceptance rates
  b_tumour[i,] <- proposed_b_tumour; b_dead[i,]=proposed_b_dead;log_sigma_b[i]=proposed_log_sigma_b  #stor
  lp0 <- lp1 ##make sure correct log posteriors are stored
  proposed= rmvnorm(1, mean=c(current_beta0, current_beta1, current_log_sigma), lambda*covariance) #update
  proposed_beta0=proposed[1];  proposed_beta1=proposed[2];  proposed_log_sigma=proposed[3]
  lp1=log.post(proposed_beta0,proposed_beta1, proposed_log_sigma, proposed_log_sigma_b,proposed_b_tumour, p
  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 ##make sure correct log posteriors are stored
}else {#accept
    accept[3]=accept[3]+1# keep track to caluclate acceptance rates
  beta0[i] <- proposed_beta0;  beta1[i]=proposed_beta1;  log_sigma[i]=proposed_log_sigma#store found value
  lp0=lp1##make sure correct log posteriors are stored
}}}}
  list(beta0=beta0, beta1=beta1, log_sigma_b=log_sigma_b, log_sigma=log_sigma,ar_outer=accept[1]/nsteps, ar
}
mh=MH2(4.2 , 0.14, -1.35,-1.5, rep(0, length(tumour$rx)),rep(0, length(dead$rx)),2.5,0.2,0.001, nsteps = ns
mh$ar_outer;mh$ar_middle;mh$ar_inner
```

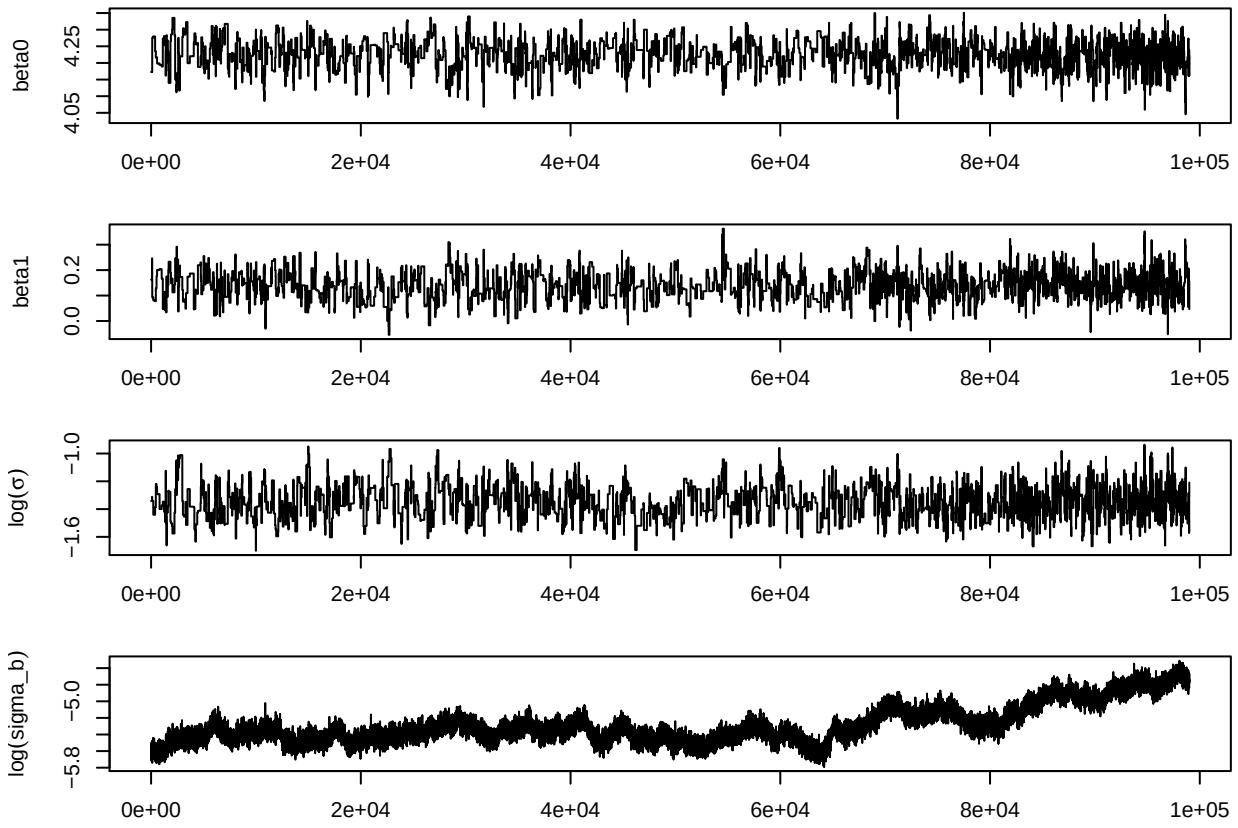
```
## [1] 0.33408
```

```
## [1] 0.2328484
```

```
## [1] 0.2577452
```

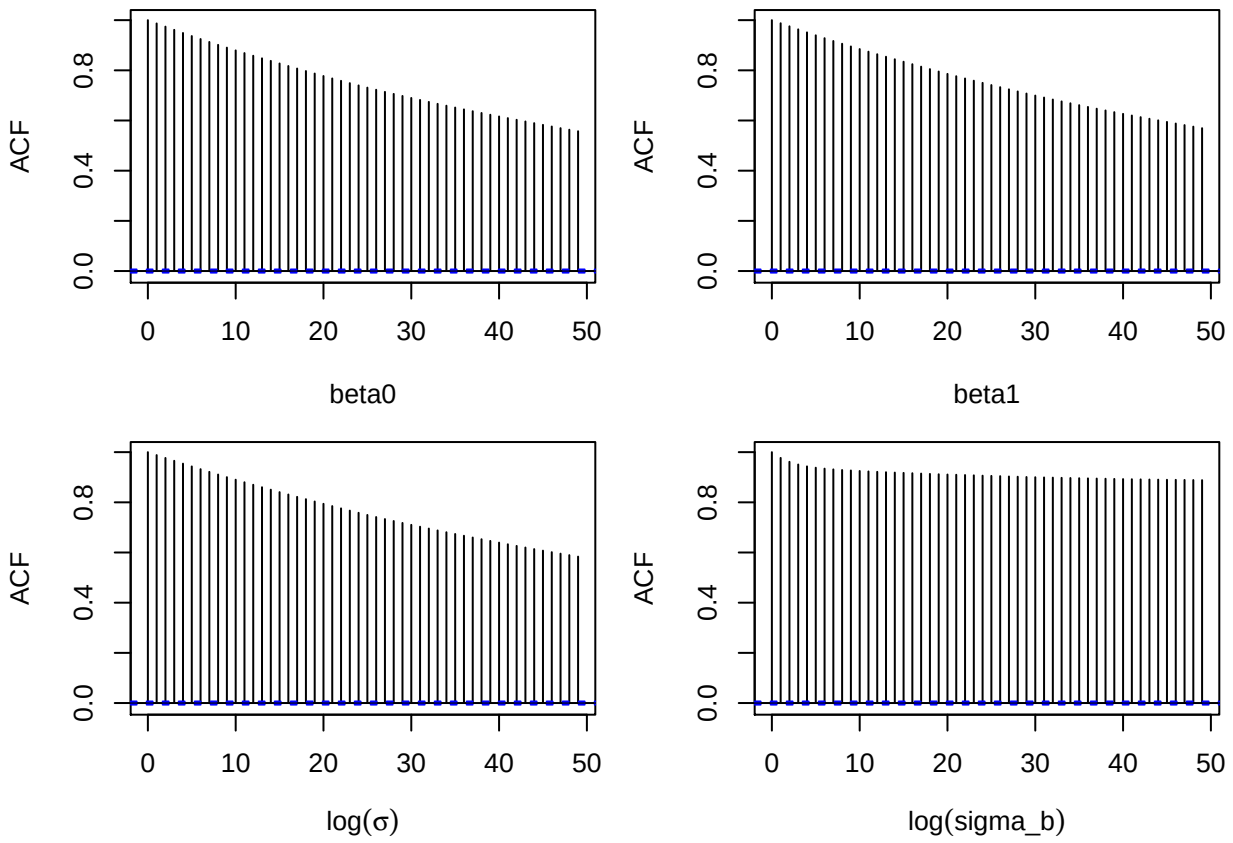
We now check that the Markov chain is behaving as we expect. Firstly, the trace plots show some good mixing within the parameters

```
show.plot<- (burn.in):nsteps
par(mfrow=c(4,1),mar=c(3,4,1,1))
plot(mh$beta0[show.plot],type="l",ylab=expression(beta0))
plot(mh$beta1[show.plot],type="l",ylab=expression(beta1))
plot(mh$log_sigma[show.plot],type="l",ylab=expression(log(sigma)))
plot(mh$log_sigma_b[show.plot],type="l",ylab=expression(log(sigma_b)))
```



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

```
par(mfrow=c(2,2),mar=c(4,4,1,1))
acf(mh$beta0[-burn.in],xlab=expression(beta0))
acf(mh$beta1[-burn.in],xlab=expression(beta1))
acf(mh$log_sigma[-burn.in],xlab=expression(log(sigma)))
acf(mh$log_sigma_b[-burn.in],xlab=expression(log(sigma_b)))
```



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

```
## [1] 1349.518 1335.088 1319.696 1108.604
```

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)
```

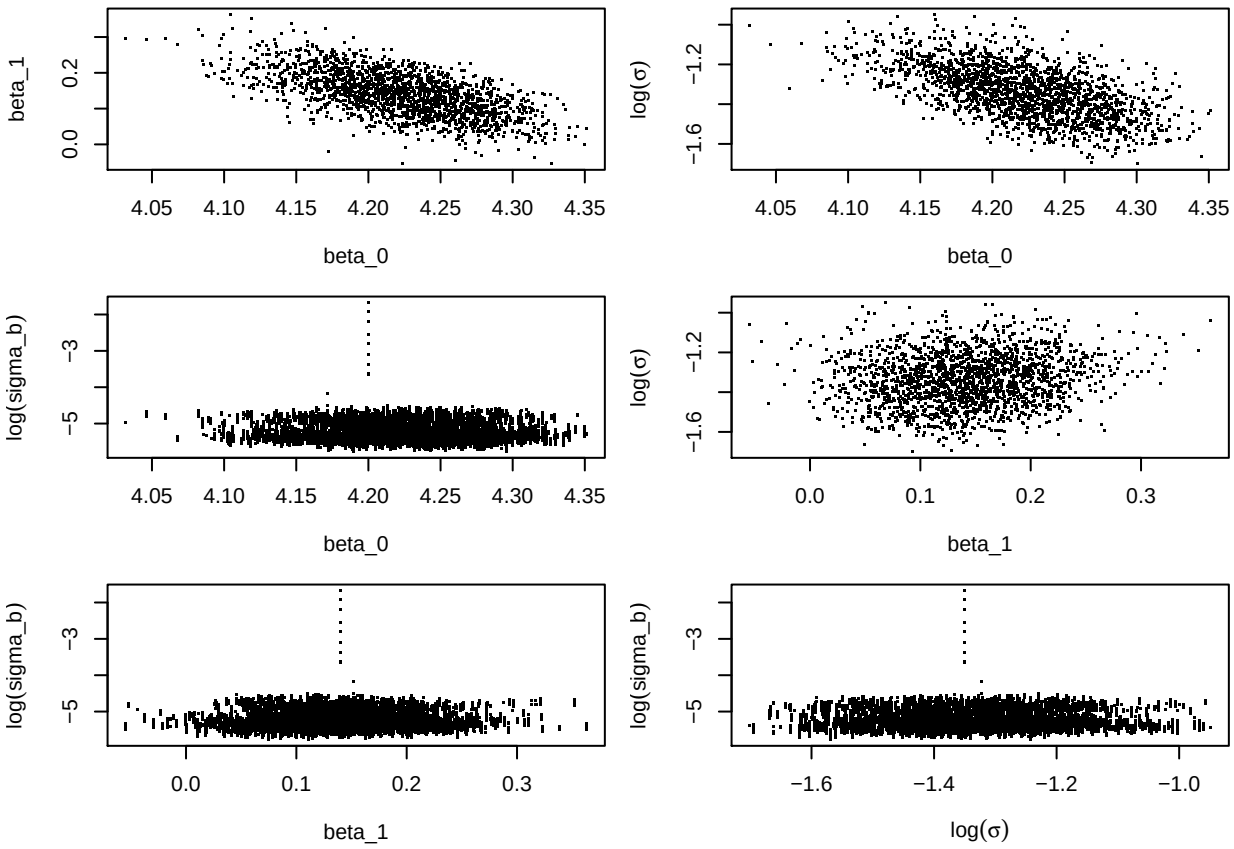
```
## [1] 0.9944232 0.9934642 0.3212377 0.9301483
```

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 .

```
par(mfrow=c(3,2),mar=c(4,4,1,1))
samp<-sample((1:nsteps)[-(burn.in)],nsteps/2)# For visualization purposes we take a random sample of the i
plot(mh$beta0[samp],mh$beta1[samp],xlab=expression(beta_0),ylab=expression(beta_1),pch=".",cex=0.1)
```

```
plot(mh$beta0[samp],mh$log_sigma[samp],xlab=expression(beta_0),ylab=expression(log(sigma)),pch=".",cex=0.1
plot(mh$beta0[samp],mh$log_sigma_b[samp],xlab=expression(beta_0),ylab=expression(log(sigma_b)),pch=".",cex
plot(mh$beta1[samp],mh$log_sigma[samp],xlab=expression(beta_1),ylab=expression(log(sigma)),pch=".",cex=0.1
plot(mh$beta1[samp],mh$log_sigma_b[samp],xlab=expression(beta_1),ylab=expression(log(sigma_b)),pch=".",cex
plot(mh$log_sigma[samp],mh$log_sigma_b[samp],xlab=expression(log(sigma)),ylab=expression(log(sigma_b)),pch
```



It looks like we need a more sophisticated proposal to deal with the correlation of $\log(\sigma_b)$ and the other 3 parameters. 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 or beta_1
```

```
##       2.5%      97.5%
## 0.03026908 0.25058503
```

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.