title: “ASI Coursework” author: “Margaret Duff” date: “30 November 2018” output: word_document: default html_document: df_print: paged —

Carcinogenesis study on rats

Consider the following data about a drug trial on 50 litters of female rats. There were three rats per litter, one that received a drug treatment and two received a control (placebo). The variables are:

- `litter` is the litter number
- `rx` is the treatment indicator where `rx=1` indicates the treatment and `rx=0` indicates the control
- `time` contains the follow up time (in weeks) to tumor appearance
- `status` is an indicator variable of the presence of tumor `status=1` or not `status=0` which indicates censoring due to animal death before tumor appearance.

The data can be read as follows

```
rats<- read.table("http://people.bath.ac.uk/kai21/ASI/rats_data.txt")
head(rats)
```

```
##   litter rx time status
## 1      1  1  101      0
## 2      1  0   49      1
## 3      1  0  104      0
## 4      2  1  104      0
## 5      2  0  102      0
## 6      2  0  104      0
```

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

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

head(tumour)
```

```
##    litter rx time status
## 2       1  0   49      1
## 16      6  1   88      1
## 17      6  0   96      1
## 19      7  1  104      1
## 21      7  0   77      1
## 22      8  1   96      1
```

```
head(dead)
```

```
##   litter rx time status
## 1      1  1  101      0
## 3      1  0  104      0
## 4      2  1  104      0
## 5      2  0  102      0
## 6      2  0  104      0
## 7      3  1  104      0
```

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

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

You may use the functions `dweibull` and `pweibull` in R. You should 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.

5. Using the same model as in question 4, now assume 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. Use a random walk Metropolis-Hastings algorithm to sample from the posterior distribution of θ using the following points as guidelines:

```
# function to compute log posterior
log.post <- function(beta0, beta1, log_sigma,log_sigma_b,b_tumour, b_dead,  tumour, dead) {
  x_tumour=tumour$rx
  y_tumour=tumour$time
  litter_tumour=tumour$litter
  n_tumour=length(y_tumour)
  x_dead=dead$rx
  y_dead=dead$time
  n_dead=length(y_dead)
  litter_dead=dead$litter
  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*exp(log_sigma_b)))


  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, sd=exp(log_sigma_b)))

   # we add the priors since we assume they are independent
  log_pi0<- log_pi0_beta0+log_pi0_beta1+log_pi0_log_sigma+log_pi0_log_sigma_b+sum(log_pio_b_dead)+sum(l

  # Now the log likelihood
  log.lik<-sum(log (dweibull(y_tumour, 1/exp(log_sigma), exp(beta0+beta1*x_tumour+b_tumour))))+sum(log

  # now the log posterior = log likelihood +log prior
  return(log.lik+log_pi0)
}
log.post(5,-0.2,-1.3,-1.5,rep(0, length(tumour$rx)),rep(0, length(dead$rx)),tumour, dead)
```

```
## [1] -309.3973
```

- Specify clearly the proposal distributions used.

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

- Choose suitable starting values for the parameters and determine a suitable burn-in period in which all samples are discarded.

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 THIS HERE

```
nsteps=1000000
burn.in=1000

MH<-function(beta0_0,beta1_0, log_sigma_0, log_sigma_b0, b_tumour0, b_dead0, sigma_beta0, sigma_beta1, s
# 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_tumour[1,]=b_tumour0
  b_dead[1,]=b_dead0

lp0 <- log.post(beta0_0,beta1_0, log_sigma_0, log_sigma_b0,b_tumour0, b_dead0,tumour, dead)

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_sigma_b=log_sigma_b[i-1]
    current_b_tumour=b_tumour[i-1,]
    current_b_dead=b_dead[i-1,]

#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, c

    acc <- exp(min(0,lp1-lp0))

if (runif(1)>=acc){#reject
  b_tumour[i,] <- current_b_tumour
  b_dead[i,]=current_b_dead
  beta0[i] <- current_beta0
  beta1[i]=current_beta1
```

```
  log_sigma[i]=current_log_sigma
  log_sigma_b[i]=current_log_sigma_b
  lp1<- lp0 ## Keep ll0 in sync with th
}else {#accept
  #store found values
  accept[1]=accept[1]+1
            log_sigma_b[i]=proposed_log_sigma_b
          lp0 <- lp1 ## Keep ll0 in sync with th

  #update b
      proposed_b_tumour=current_b_tumour + rnorm( length(tumour$rx), mean=0, sd=sigma_b)
    proposed_b_dead=current_b_dead+ rnorm( length(dead$rx), mean=0, sd=sigma_b)
    lp1 <- log.post(current_beta0,current_beta1, current_log_sigma, proposed_log_sigma_b,proposed_b_tume
    check=is.finite(lp1)
  acc <- exp(min(0,lp1-lp0))
  if (runif(1)>=acc){#reject
    #store values
  b_tumour[i,] <- current_b_tumour
  b_dead[i,]=current_b_dead
  beta0[i] <- current_beta0
  beta1[i]=current_beta1
  log_sigma[i]=current_log_sigma
  lp1<- lp0 ## Keep ll0 in sync with th
}else {#accept
    #store values
          accept[2]=accept[2]+1
          b_tumour[i,] <- proposed_b_tumour
          b_dead[i,]=proposed_b_dead
          log_sigma_b[i]=proposed_log_sigma_b
          lp0 <- lp1 ## Keep ll0 in sync with th


  #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){#reject
    #store values

  beta0[i] <- current_beta0
  beta1[i]=current_beta1
  log_sigma[i]=current_log_sigma
  lp1<- lp0 ## Keep ll0 in sync with th
}else {#accept

  #store values
    accept[3]=accept[3]+1
  beta0[i] <- proposed_beta0
  beta1[i]=proposed_beta1
```

```
  log_sigma[i]=proposed_log_sigma
  lp0=lp1

}
}

}
}
  list(beta0=beta0, beta1=beta1, log_sigma_b=log_sigma_b, log_sigma=log_sigma,ar_outer=accept[1]/nsteps
}

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, r
mh$ar_outer
```

```
## [1] 0.33372
```

```
mh$ar_middle
```

```
## [1] 0.7318171
```

```
mh$ar_inner
```

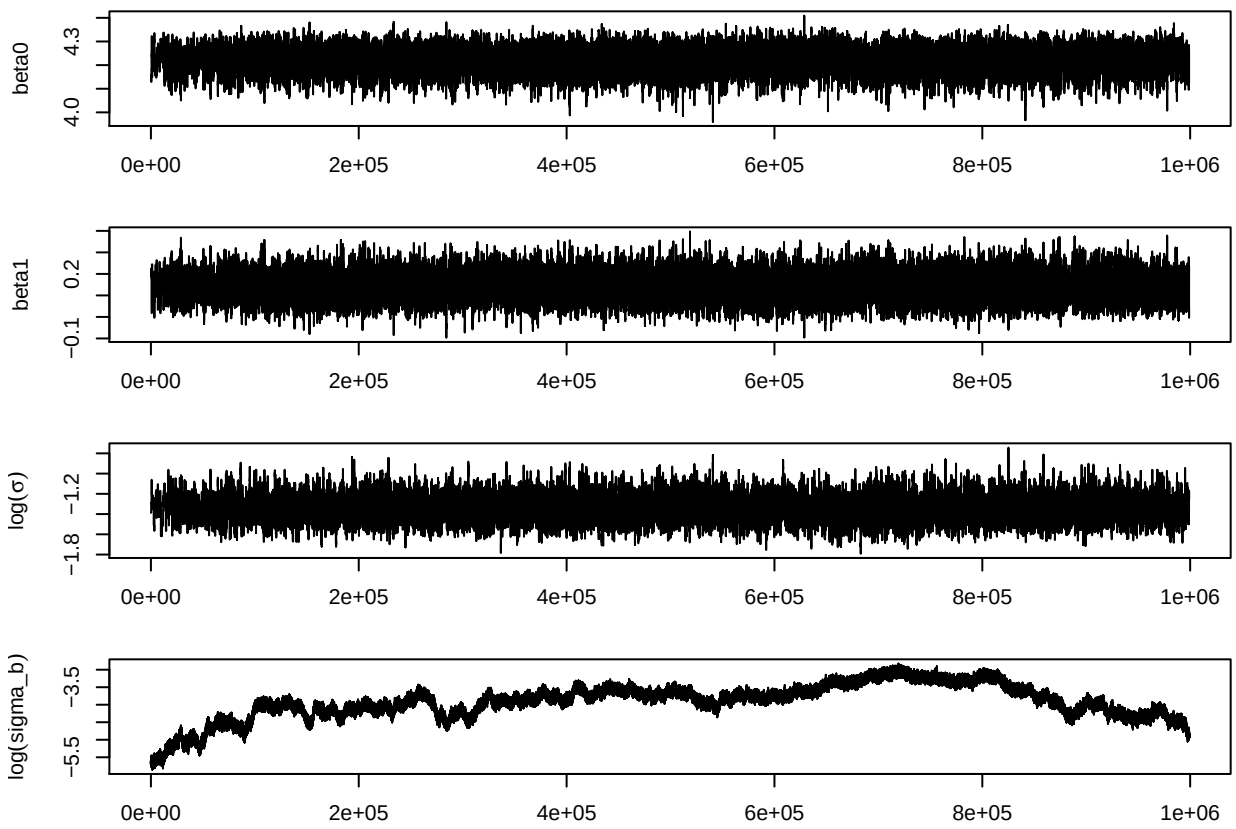
```
## [1] 0.1666107
```

- Tune the proposal standard deviations.

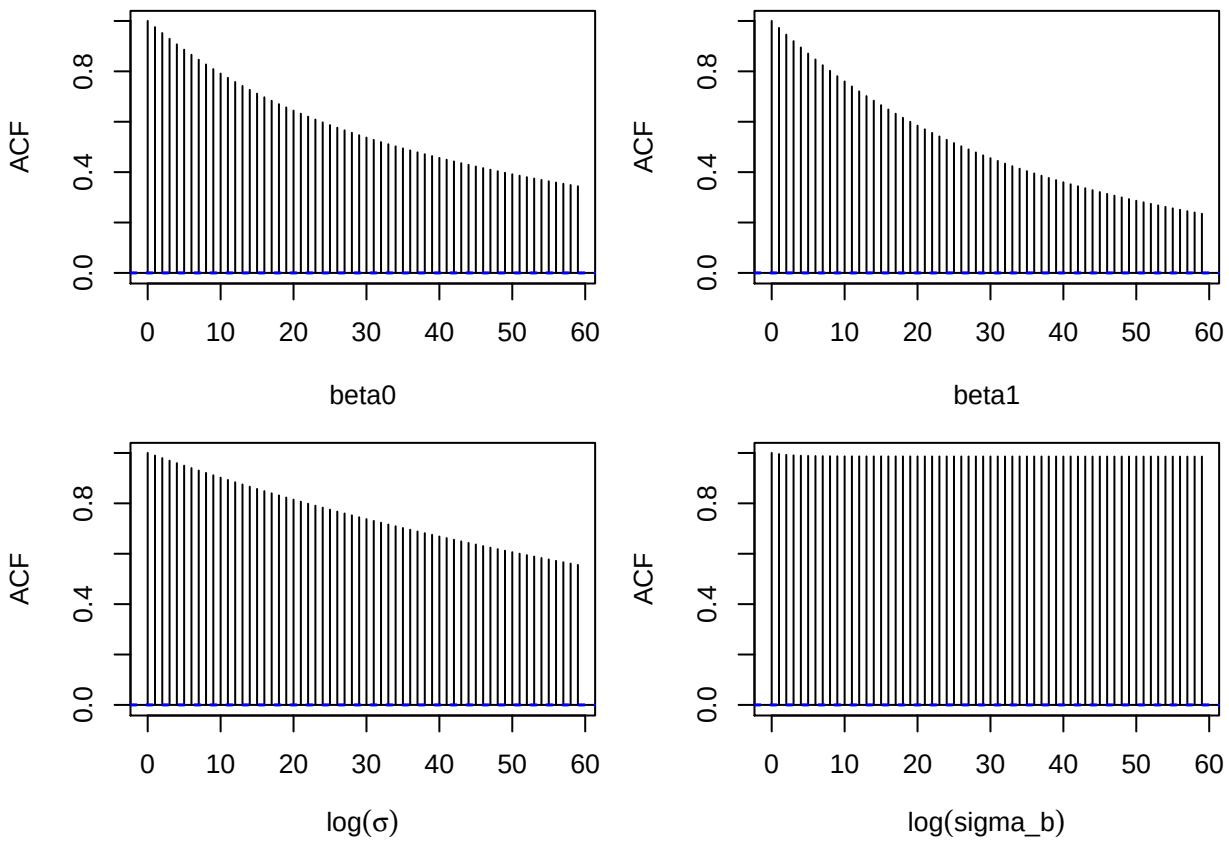
For tuning the proposal distribution we use run lengths of about 10000. 100000 would be better for the final run. The aim is for an acceptance rate of approximately 25%

- After running the tuned Metropolis-Hastings sampler check for correlation between 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)))
```




```
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)))
```



calculate the effective sample size for each paramater and see that the effective smaple size for $\log(\sigma)$ are about half the size than those for β_0, β_1 and σ_b.

```
# note we do discard burn-in iterations
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

# t.eff is the integrated autocrrelation length
n.eff
```

```
## [1] 14465.323 16496.878 11179.210  8517.103
```

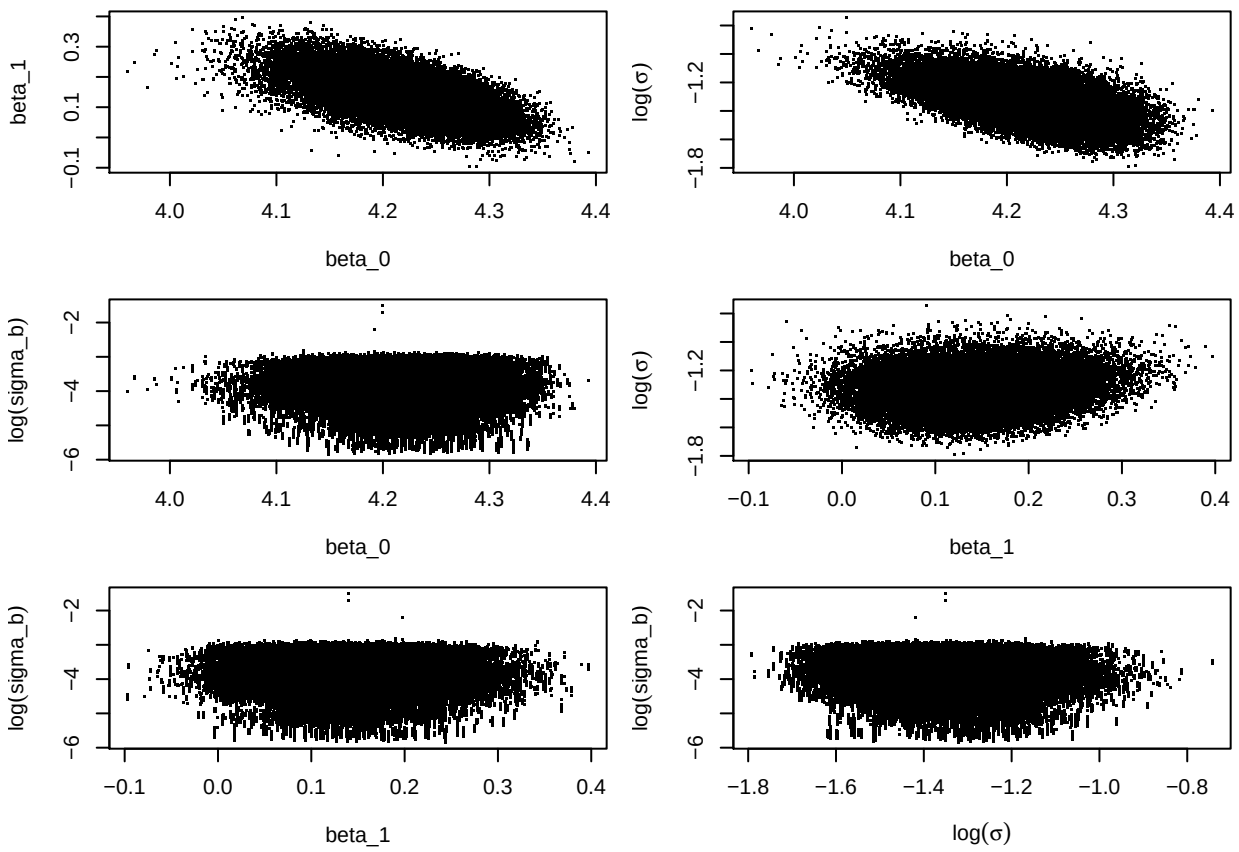
We check for correlation between the parameters by plotting graphs...

```r
par(mfrow=c(3,2),mar=c(4,4,1,1))
# For visualization purposes we take a random sample
# of the iterations retained (after discarding burn-in)
samp<-sample((1:nsteps)[-(burn.in)],nsteps/2)

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=

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



- Write a new Metropolis-Hastings sampler that makes use of any posterior correlation between the parameters (you may find the `mvtnorm` package in R helpful for this). See section 6.5.1 of *Core statistics*.

Fit a multivariate normal distribution to the paramaters from thw inner metropolis hastings step

```r
mu=c(mean(mh$beta0[-(burn.in)]),mean(mh$beta1[-(burn.in)]), mean(mh$log_sigma[-(burn.in)]) )
s11=cov(mh$beta0, mh$beta0)
s12=cov(mh$beta0, mh$beta1)
s13=cov(mh$beta0, mh$log_sigma)
s22=cov(mh$beta1, mh$beta1)
s23=cov(mh$beta1, mh$log_sigma)
s33=cov(mh$log_sigma, mh$log_sigma)
```

```
covariance= matrix(c(s11,s12,s13,s12,s22,s23,s13,s23,s33), nrow=3, byrow=TRUE)

mu
```

```
## [1]  4.2221022  0.1413442 -1.3494325
```

```
covariance
```

```
##              [,1]          [,2]          [,3]
## [1,]  0.002143978 -0.0016074944 -0.0032918785
## [2,] -0.001607494  0.0031860614  0.0009122283
## [3,] -0.003291879  0.0009122283  0.0146852584
```

```
library(mvtnorm)

new=rmvnorm(1, mean=mu, sigma= covariance)
print(new)
```

```
##          [,1]       [,2]      [,3]
## [1,] 4.269677 0.07788341 -1.523029
```

```
dmvnorm( new, mu, covariance)
```

```
## [1] 66.9377
```

Not try writing a new MH algorithm using $\theta|y \sim N(\mu, \Sigma)$.

This we now have the case of an independence sampler. We peopose values of θ using the fixed distribution $\theta' \sim N(\theta_{j-1}, \Sigma)$.

$$\alpha(\theta'|\theta_{j-1}) = \min\{1, \frac{\pi(\theta')q(\theta_{j-1}|\theta')}{\pi(\theta_{j-1})q(\theta'|\theta_{j-1})}\}$$

```
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
# 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_tumour[1,]=b_tumour0
  b_dead[1,]=b_dead0

lp0 <- log.post(beta0_0,beta1_0, log_sigma_0, log_sigma_b0,b_tumour0, b_dead0,tumour, dead)

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_sigma_b=log_sigma_b[i-1]
    current_b_tumour=b_tumour[i-1,]
    current_b_dead=b_dead[i-1,]

#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, c

    acc <- exp(min(0,lp1-lp0))

if (runif(1)>=acc){#reject
  b_tumour[i,] <- current_b_tumour
  b_dead[i,]=current_b_dead
  beta0[i] <- current_beta0
  beta1[i]=current_beta1
  log_sigma[i]=current_log_sigma
  log_sigma_b[i]=current_log_sigma_b
  lp1<- lp0 ## Keep ll0 in sync with th
}else {#accept
  #store found values
  accept[1]=accept[1]+1
            log_sigma_b[i]=proposed_log_sigma_b
          lp0 <- lp1 ## Keep ll0 in sync with th

  #update b
      proposed_b_tumour=current_b_tumour + rnorm( length(tumour$rx), mean=0, sd=sigma_b)
    proposed_b_dead=current_b_dead+ rnorm( length(dead$rx), mean=0, sd=sigma_b)
    lp1 <- log.post(current_beta0,current_beta1, current_log_sigma, proposed_log_sigma_b,proposed_b_tumo
    check=is.finite(lp1)
  acc <- exp(min(0,lp1-lp0))
  if (runif(1)>=acc){#reject
    #store values
  b_tumour[i,] <- current_b_tumour
  b_dead[i,]=current_b_dead
  beta0[i] <- current_beta0
  beta1[i]=current_beta1
  log_sigma[i]=current_log_sigma
  lp1<- lp0 ## Keep ll0 in sync with th
}else {#accept
    #store values
          accept[2]=accept[2]+1
          b_tumour[i,] <- proposed_b_tumour
          b_dead[i,]=proposed_b_dead
          log_sigma_b[i]=proposed_log_sigma_b
          lp0 <- lp1 ## Keep ll0 in sync with th


  #update remaining paramaters
  proposed= rmvnorm(1, mean=c(current_beta0, current_beta1, current_log_sigma), lambda*covariance)
```

```
  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_tumou
  factor=log(dmvnorm(c(current_beta0, current_beta1, current_log_sigma_b), proposed, lambda*covariance)

  acc <- exp(min(0,lp1-lp0+factor))

  if (runif(1)>=acc){#reject
    #store values

  beta0[i] <- current_beta0
  beta1[i]=current_beta1
  log_sigma[i]=current_log_sigma
  lp1<- lp0 ## Keep ll0 in sync with th
}else {#accept

  #store values
    accept[3]=accept[3]+1
  beta0[i] <- proposed_beta0
  beta1[i]=proposed_beta1
  log_sigma[i]=proposed_log_sigma
  lp0=lp1

}}}}
  list(beta0=beta0, beta1=beta1, log_sigma_b=log_sigma_b, log_sigma=log_sigma,ar_outer=accept[1]/nsteps
}

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 =
mh$ar_outer
```

```
## [1] 0.33122
```

```
mh$ar_middle
```

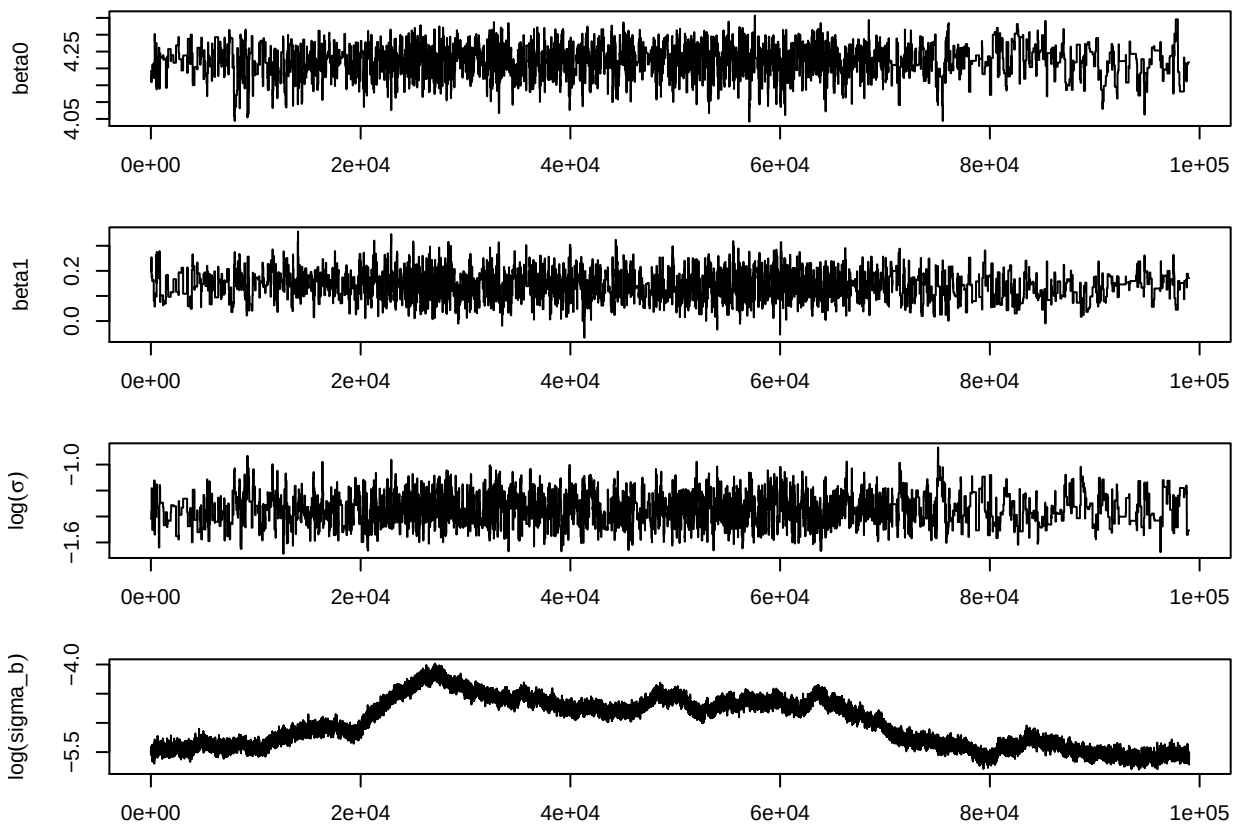
```
## [1] 0.3587948
```

```
mh$ar_inner
```

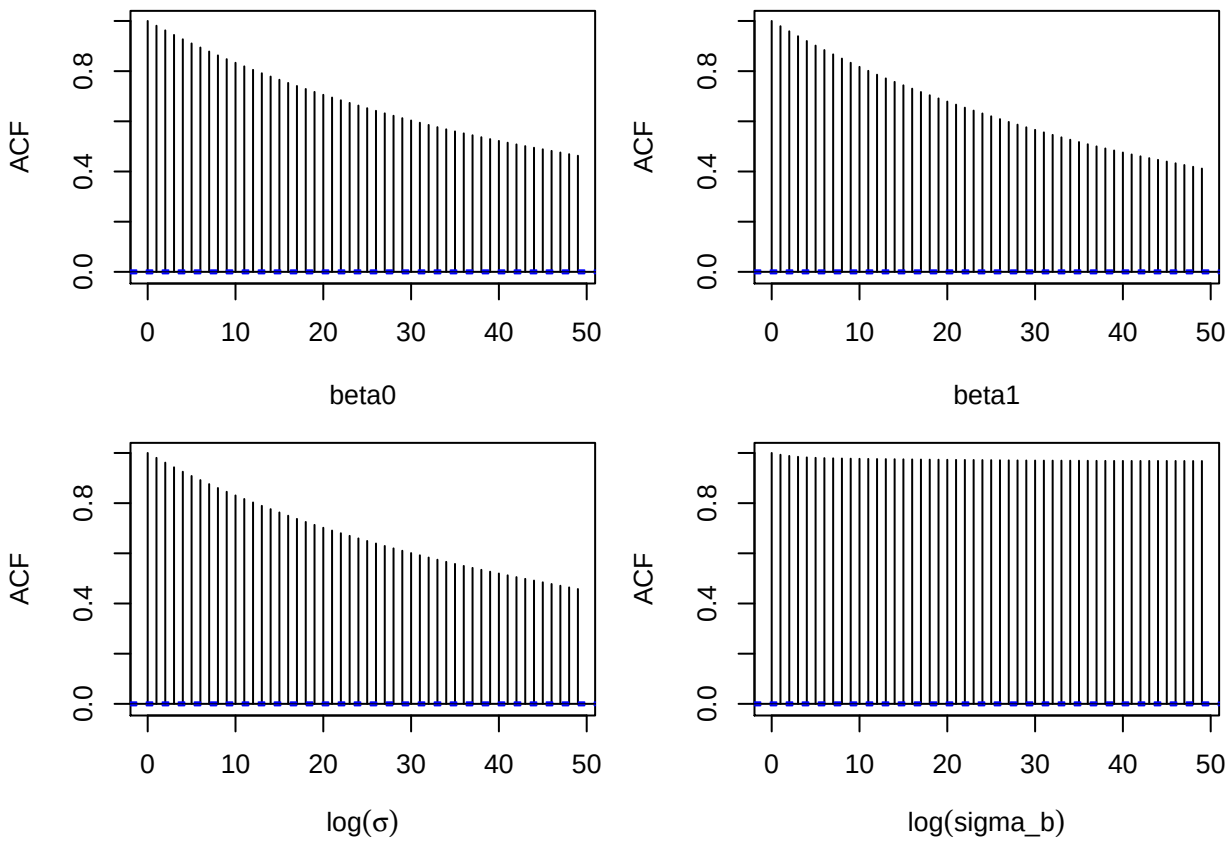
```
## [1] 0.259172
```

- After running the tuned Metropolis-Hastings sampler check for correlation between 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)))
```




```
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)))
```



calculate the effective sample size for each paramater and see that the effective smaple size for $\log(\sigma)$ are about half the size than those for β_0, β_1 and σ_b.

```
# note we do discard burn-in iterations
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

# t.eff is the integrated autocrrelation length
n.eff
```

```
## [1] 1488.496 1558.702 1494.653 1037.836
```

We check for correlation between the parameters by plotting graphs...

- Tune your new Metropolis-Hastings sampler by tuning the appropriate parameter in the proposal distribution.
- In general, you should give substantial empirical evidence that the Markov chain constructed has converged to its stationary limiting distribution.

We will obtain an independent subsample of our chain and then split it into two subsamples. Using the Kolmogoroc-Smirnov test we chanck to see that they come from the same distribution and hence that the Markov chain has converged.

```
ks.test(mh$beta0[-1000:-500], mh$beta0[-500])
```

```
## Warning in ks.test(mh$beta0[-1000:-500], mh$beta0[-500]): p-value will be
## approximate in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  mh$beta0[-1000:-500] and mh$beta0[-500]
## D = 0.0027283, p-value = 0.8517
## alternative hypothesis: two-sided
```

```
ks.test(mh$beta1[-1000:-500], mh$beta1[-500])
```

```
## Warning in ks.test(mh$beta1[-1000:-500], mh$beta1[-500]): p-value will be
## approximate in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  mh$beta1[-1000:-500] and mh$beta1[-500]
## D = 0.0020569, p-value = 0.9842
## alternative hypothesis: two-sided
```

```
ks.test(mh$log_sigma_b[-1000:-500], mh$log_sigma_b[-500])
```

```
## Warning in ks.test(mh$log_sigma_b[-1000:-500], mh$log_sigma_b[-500]): p-
## value will be approximate in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  mh$log_sigma_b[-1000:-500] and mh$log_sigma_b[-500]
## D = 0.0036654, p-value = 0.5143
## alternative hypothesis: two-sided
```

```
ks.test(mh$log_sigma[-1000:-500], mh$log_sigma[-500])
```

```
## Warning in ks.test(mh$log_sigma[-1000:-500], mh$log_sigma[-500]): p-value
## will be approximate in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  mh$log_sigma[-1000:-500] and mh$log_sigma[-500]
## D = 0.0020367, p-value = 0.9858
## alternative hypothesis: two-sided
```

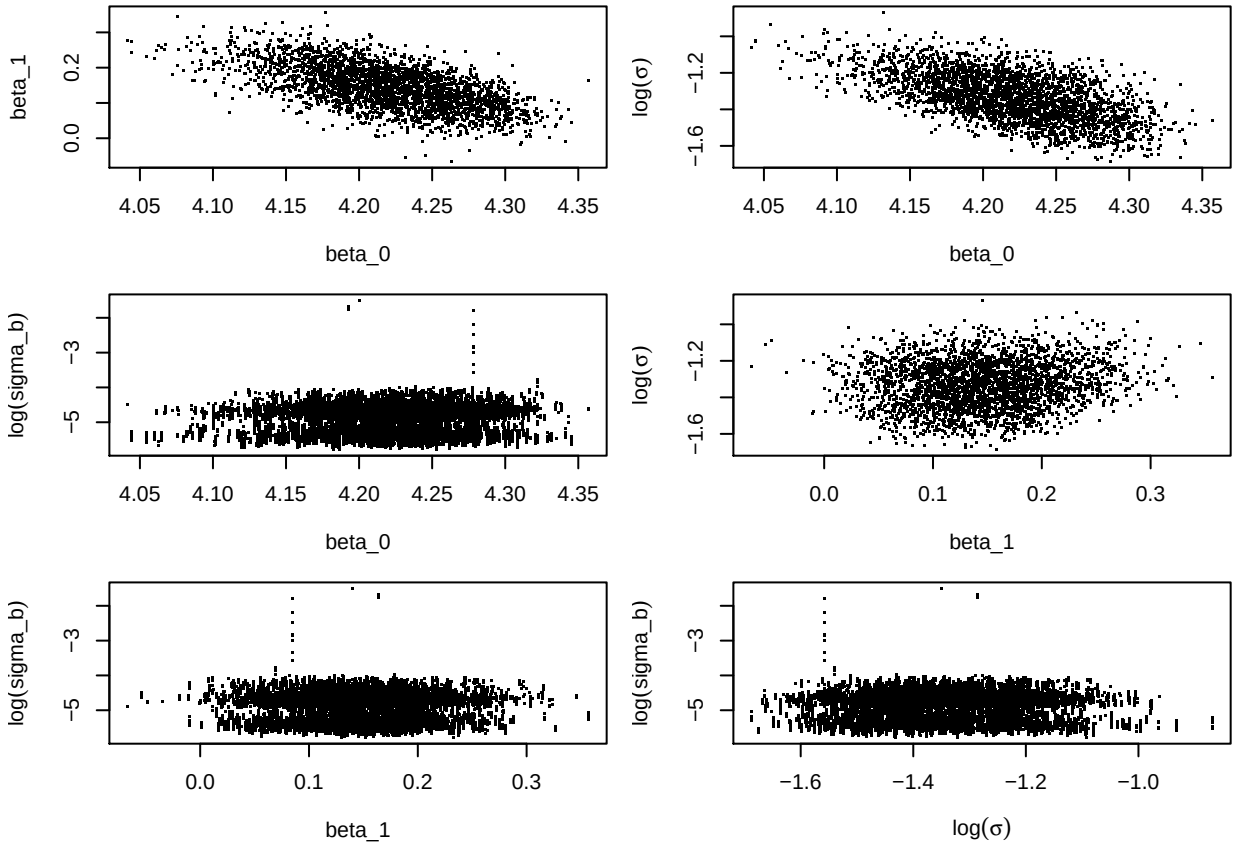
- Produce plots to learn about the shape of the marginal posterior densities of the parameters. You should also investigate the correlation between parameters.

```r
par(mfrow=c(3,2),mar=c(4,4,1,1))
# For visualization purposes we take a random sample
# of the iterations retained (after discarding burn-in)
samp<-sample((1:nsteps)[-(burn.in)],nsteps/2)

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=

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



* Compute a 95% posterior probability interval for the intervention effect β_1. What conclusions can you draw?

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

```
##       2.5%      97.5%
## 0.03553161 0.25385524
```

We note that 0 is not contained in the 95% posterior probablity interval for the intervention effect, suggesting that the intervention has a statistially significant effect.