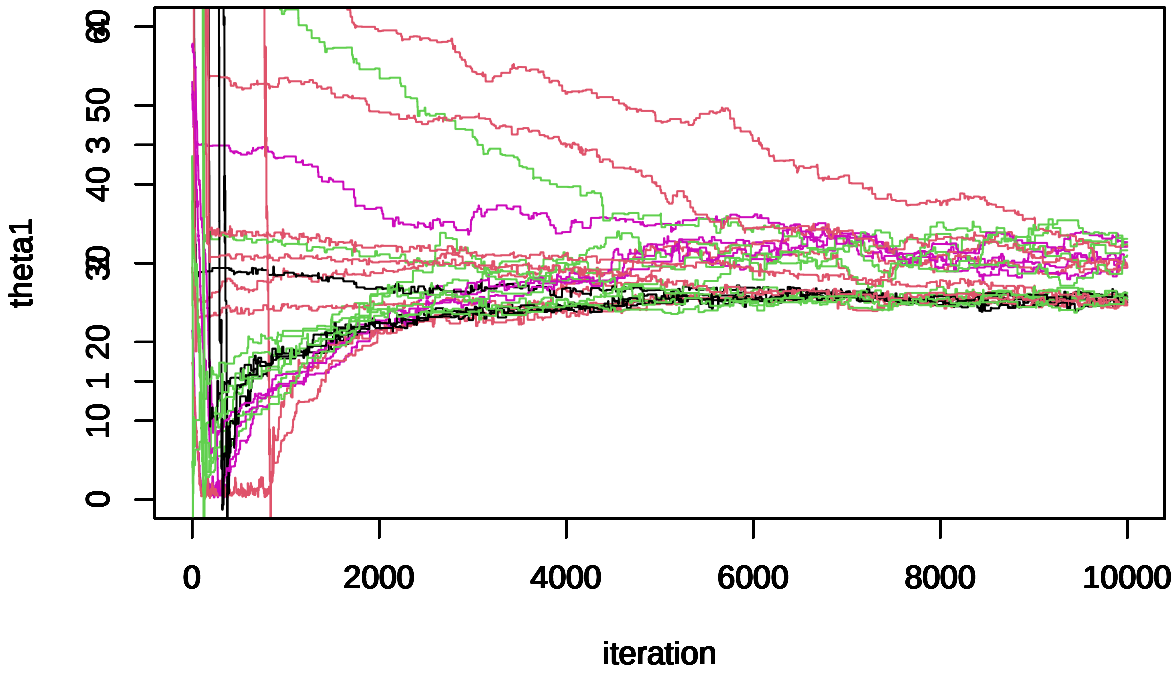
Assignment6

```
## try some tuning for each algorithm
##
library("mvtnorm")
##For the initial values, draw alpha and beta from the uniform prior for each chain.
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
##For the jumping distribution, use multivariate distribution same as last assignment.
my_data <- read.delim("spectral-1.txt",sep = " ")
eng<-my_data$Energy

cou<-my_data$Count
## I am stuck on the posterior distribution, assume as previous assignment first.

Sigma <- matrix(data=c(1,0.8,0.8,1),nrow=2,ncol=2)

metrop_alg<- function(num_iter,size){

  theta<-matrix(0,nrow=num_iter,ncol=2 )

  colnames(theta)<-c("alpha","beta")

  theta[1,]=c(runif(1,min=0,max=100),runif(1,min=0,max=100))
  ###start value from a Uniform(0,100)

for(i in 2:num_iter)
{
  # Step 2(a): sample theta_star from jumping dist.
  theta_star<-rmvnorm(1,mean=c(theta[i-1,]),sigma=Sigma*size) ##jumping

  lamudacurr<-(theta[(i-1),1])*(eng^(-theta[(i-1),2]))
  lamudaprop<-theta_star[1]*(eng^(-theta_star[2]))

  log_dpost_curr<-sum(dpois(cou,lamudacurr,log=TRUE))## p( theta^(t-1) *|y)
  log_dpost_prop<-sum(dpois(cou,lamudaprop,log=TRUE)) ## p( theta *|y)


  r<-log_dpost_prop-log_dpost_curr
  if(isTRUE(r > log(runif(1,min=0,max=1))))
  {
    theta[i,]<-theta_star
  }
  else
  {
    theta[i,]<-theta[(i-1),]
  }
}
  return(theta)
```

```
}

theta_samples1<-metrop_alg(10000,0.5)
theta_samples11<-metrop_alg(10000,0.5)
theta_samples2<-metrop_alg(10000,0.3)
theta_samples22<-metrop_alg(10000,0.3)
theta_samples3<-metrop_alg(10000,1)
theta_samples33<-metrop_alg(10000,1)

## green Chain converged the best in the first 10000 iteration.
## Assume the larger jumping size is better here for random-walk algorithm

plot(x=c(1:10000),y=theta_samples1[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=6,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples11[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=6,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples2[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=2,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples22[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=2,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples3[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=3,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples33[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=3,ylim=c(0,60))
par(new=TRUE)



plot(x=c(1:10000),y=theta_samples1[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=1,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples11[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=1,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples2[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=2,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples22[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=2,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples3[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=3,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples33[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=3,ylim=c(0,4))
par(new=TRUE)
```

```r
metrop_single_alg<- function(num_iter,size){

  theta<-matrix(0,nrow=num_iter,ncol=2)
  colnames(theta)<-c("alpha","beta")
  theta[1,1]=runif(1,min=0,max=100) ###start value from a Uniform(0,100)
  theta[1,2]=runif(1,min=0,max=100)
  theta_star<-matrix(0,1,2)
for(i in 2:num_iter)
{
  # Step 2(a): sample theta_star from jumping dist.
  theta_star[1,1]<-rnorm(1,mean=theta[i-1,1],sqrt(Sigma)*size) ##jumping
  theta_star[1,2]<-rnorm(1,mean=c(theta[i-1,2]),sqrt(Sigma)*size)

  lamudaprop1<-theta_star[1]*(eng^(-theta[i-1,2]))
  lamudacurr1<-(theta[i-1,1])*(eng^(-theta[i-1,2]))

  log_dpost_prop1<-sum(dpois(cou,lamudaprop1,log=TRUE))
  log_dpost_curr1<-sum(dpois(cou,lamudacurr1,log=TRUE))

  r1<-log_dpost_prop1-log_dpost_curr1

  if(isTRUE(r1 > log(runif(1,min=0,max=1))))
  {
    theta[i,1]<-theta_star [1]
  }
  else
  {
    theta[i,1]<-theta[i-1,1]
  }

  lamudaprop2<-(theta[i,1])*(eng^(-theta_star[2]))
  lamudacurr2<-(theta[i,1])*(eng^(-theta[i-1,2]))

  log_dpost_prop2<-sum(dpois(cou,lamudaprop2,log=TRUE))
  log_dpost_curr2<-sum(dpois(cou,lamudacurr2,log=TRUE))

  r2<-log_dpost_prop2-log_dpost_curr2

  if(isTRUE(r2 > log(runif(1,min=0,max=1))))
  {
    theta[i,2]<-theta_star [2]
  }
  else
  {
    theta[i,2]<-theta[i-1,2]
  }


}
  return(theta)
}

#####  I will conclude the one with 0.5 step size did the best based on the
```

```r
##  Gelman-Rubin R, but I did not use this step size in my assignment

theta_samples10<-metrop_alg(10000,0.5)
theta_samples110<-metrop_alg(10000,0.5)
theta_samples20<-metrop_alg(10000,0.3)
theta_samples220<-metrop_alg(10000,0.3)
theta_samples30<-metrop_alg(10000,1)
theta_samples330<-metrop_alg(10000,1)

plot(x=c(1:10000),y=theta_samples10[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=6,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples110[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=6,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples20[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=2,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples220[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=2,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples30[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=3,ylim=c(0,60))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples330[1:10000,1],type="l",xlab="iteration",
     ylab="theta1",col=3,ylim=c(0,60))
par(new=TRUE)


plot(x=c(1:10000),y=theta_samples10[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=1,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples110[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=1,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples20[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=2,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples220[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=2,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples30[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=3,ylim=c(0,4))
par(new=TRUE)
plot(x=c(1:10000),y=theta_samples330[1:10000,2],type="l",xlab="iteration",
     ylab="theta1",col=3,ylim=c(0,4))
```




```
par(new=TRUE)

mc1=as.mcmc(theta_samples11[100000:130000])
mc2=as.mcmc(theta_samples22[100000:130000])
mc3=as.mcmc(theta_samples33[100000:130000])
mc4=as.mcmc(theta_samples44[100000:130000])
mc5=as.mcmc(theta_samples55[100000:130000])
mclist = mcmc.list(mc1,mc2,mc3,mc4,mc5)
gelman.diag(mclist)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]         NA         NA
```

```
mc10=as.mcmc(theta_samples10[1:10000])
mc110=as.mcmc(theta_samples110[1:10000])

mclist1 = mcmc.list(mc10,mc110)
gelman.diag(mclist1)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       2.38       4.87
```

```
mc20=as.mcmc(theta_samples20[1:10000])
mc220=as.mcmc(theta_samples220[1:10000])
mclist2 = mcmc.list(mc20,mc220)
gelman.diag(mclist2)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
```

```
## [1,]       2.08       5.98
```

```
mc30=as.mcmc(theta_samples30[1:10000])
mc330=as.mcmc(theta_samples330[1:10000])
mclist3 = mcmc.list(mc30,mc330)
gelman.diag(mclist3)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.44       2.98
```