Metropolis-Hasting MCMC algorithm for mu (normal) and tau (gamma)

```r
rm(list=ls())
y = scan("y data.txt", what=double())
```

```r
means <- function(chain) {
    return(c(mean(chain[,1]), mean(chain[,2])))
}
```

```r
cred_int <- function(chain) {
    mu_95 <- quantile(chain[,1], 0.95); mu_05 <- quantile(chain[,1], 0.05)
    tau_95 <- quantile(chain[,2], 0.95); tau_05 <- quantile(chain[,2], 0.05)

    intervals <- matrix(0,2,2)
    intervals[1,1] <- mu_05; intervals[1,2] <- mu_95;
    intervals[2,1] <- tau_05; intervals[2,2] <- tau_95;

    return(intervals)
}
```

Returning the likelihood of the y__i data given a mu and tau.

```r
likelihood <- function(param){
    mu = param[1]
    tau = param[2]

    singlelikelihoods = dnorm(y, mean = mu, sd = 1/(sqrt(tau)), log = T)
    sumll = sum(singlelikelihoods)
    return(sumll)
}
```

Defining the prior

```r
# Prior distribution
prior <- function(param){
    tau = param[2]
    prior = log(1/(tau))
    return(prior)
}
```

The posterior

```
posterior <- function(param){
   return (likelihood(param) + prior(param))
}
```

The MCMC

```
######## Metropolis algorithm ################

proposalfunction <- function(param){
    mu_c = param[1]; tau_c = param[2]
    tau_n = rgamma(n = 1, shape = (5*tau_c), rate = 5)
    mu_n = rnorm(n = 1, mean = mu_c, sd = sqrt(tau_n))

    return(c(mu_n, tau_n))
    }

run_metropolis_MCMC <- function(startvalue, iterations){
    chain = array(dim = c(iterations+1,2))
    chain[1,] = startvalue
    for (i in 1:iterations){
        proposal = proposalfunction(chain[i,])

        probab = exp(posterior(proposal) - posterior(chain[i,]))
        if (runif(1) < probab){
            chain[i+1,] = proposal
        }else{
            chain[i+1,] = chain[i,]
        }
    }
    return(chain)
}

set.seed(1)
nreps = 10000; burnIn = 1000;
startvalue1 = c(3, 1);
startvalue2 = c(10,5);

chain1 = run_metropolis_MCMC(startvalue1, nreps)
chain2 = run_metropolis_MCMC(startvalue2, nreps)

(mean_chain1 <- means(chain1))
```

```
## [1] 5.0751991 0.2402314
```

```
(mean_chain2 <- means(chain2))
```

```
## [1] 5.0829774 0.2535314
```

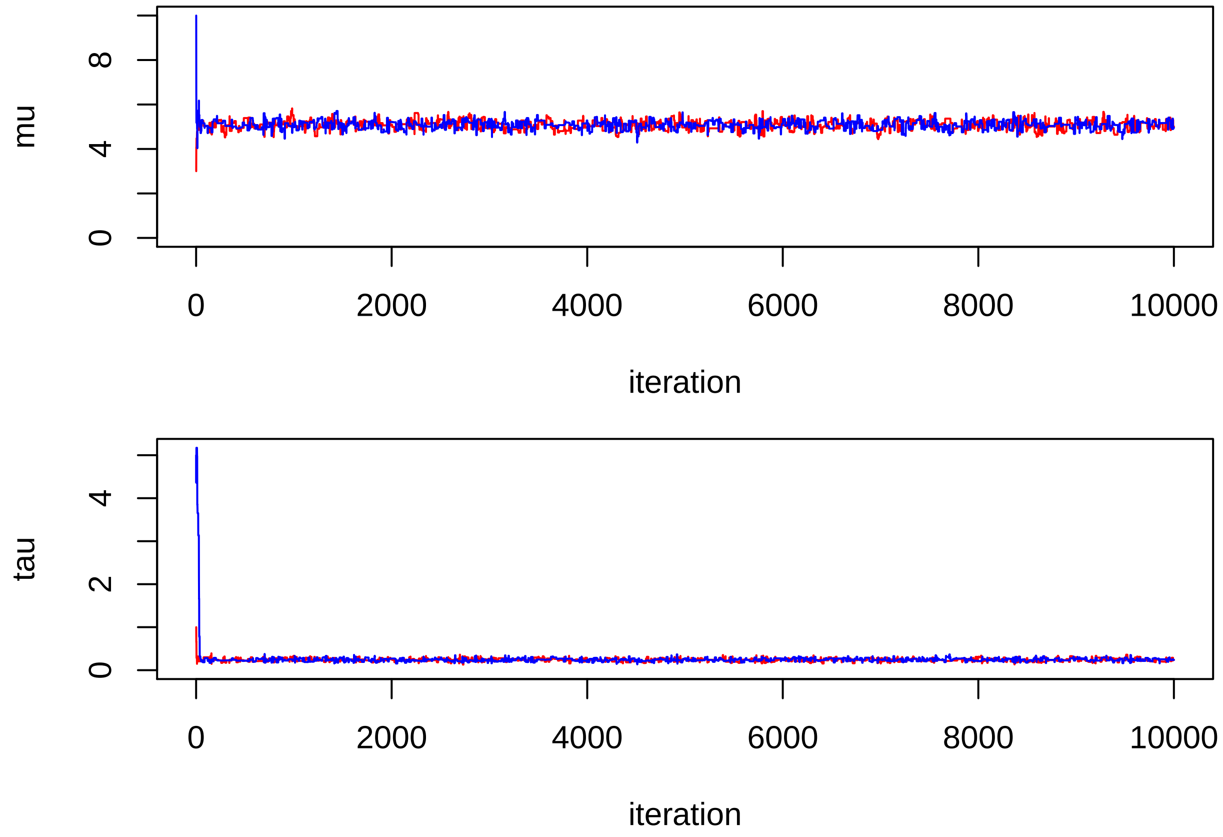
```r
(cred_chain1 <- cred_int(chain1))
```

```
##            [,1]      [,2]
## [1,] 4.7398422 5.4161318
## [2,] 0.1844582 0.3008101
```

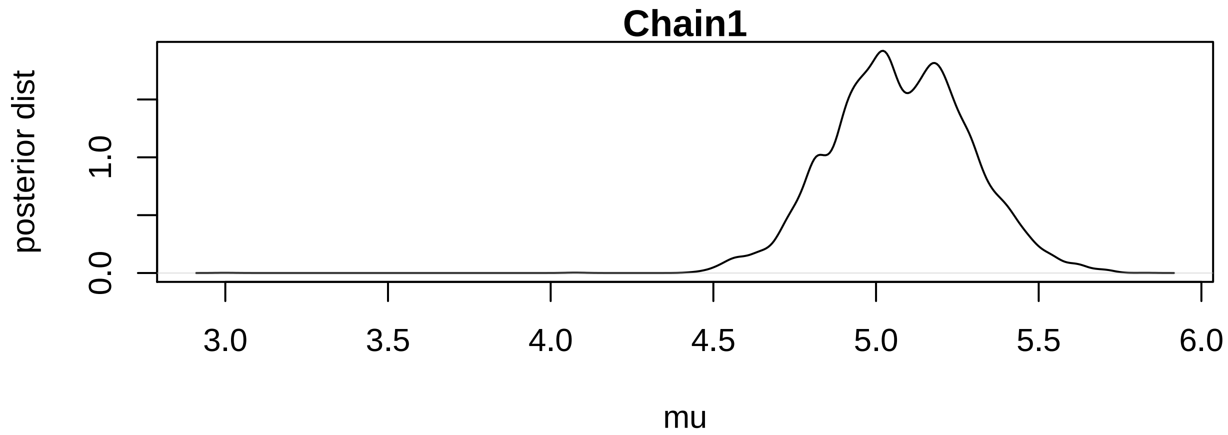
```r
(cred_chain2 <- cred_int(chain2))
```

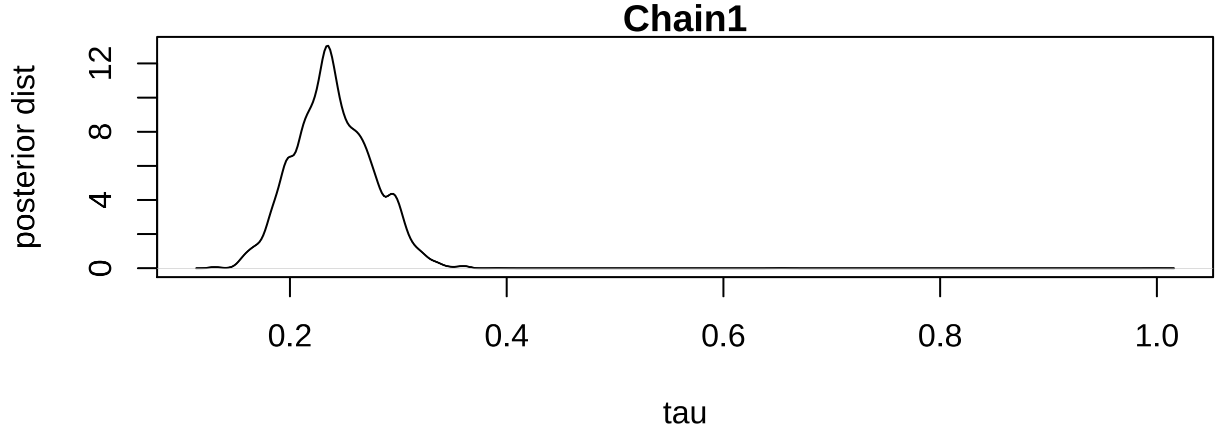
```
##            [,1]      [,2]
## [1,] 4.7566523 5.3918818
## [2,] 0.1889075 0.2971658
```

```r
par(mfrow=c(2,1), mar=c(4,4,1,1))

plot(1:(nreps+1), chain1[,1], type="l", col="red", ylim = c(0, max(chain1[,1], chain2[,1])), xlab = "ite
points(1:(nreps+1), chain2[,1], type="l", col="blue")

plot(1:(nreps+1), chain1[,2], type="l", col="red", ylim = c(0, max(chain1[,2], chain2[,2])), xlab = "ite
points(1:(nreps+1), chain2[,2], type="l", col="blue")
```

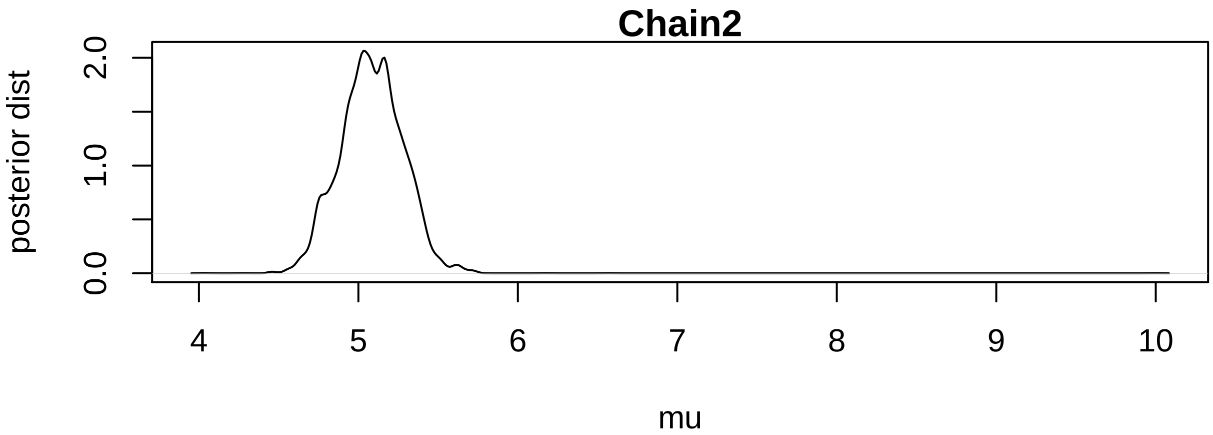


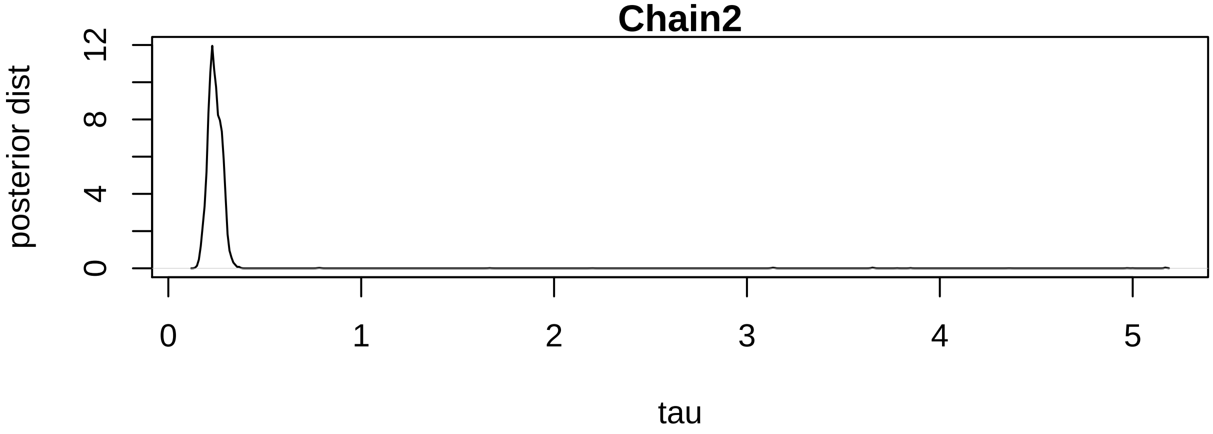
```r
par(mfrow=c(2,1), mar=c(4,4,1,1))

plot(density(chain1[,1]), ylab="posterior dist", xlab="mu", main="Chain1")
plot(density(chain1[,2]), ylab="posterior dist", xlab="tau", main="Chain1")
```





```
plot(density(chain2[,1]), ylab="posterior dist", xlab="mu", main="Chain2")
plot(density(chain2[,2]), ylab="posterior dist", xlab="tau", main="Chain2")
```


Chain2
posterior dist
0.0
1.0
2.0
4
5
6
7
8
9
10
mu


Chain2
posterior dist
0
4
8
12
0
1
2
3
4
5
tau