11.2

** Starting values chosen as ML estimates. **

```
library(gtools)
# Data
x               <-  c(-0.86, -0.30, -0.05,  0.73)
y               <-  c(0, 1, 3, 5)
n               <-  rep(5,4)
# ML estimation
fit <- glm( cbind(y, n-y) ~ x, family = binomial)
fit$coefficients = round(fit$coefficients*1000)/1000
paste(paste("(",fit$coefficients[1], sep=""),
             paste(fit$coefficients[2],")", sep=""), sep=",")
```

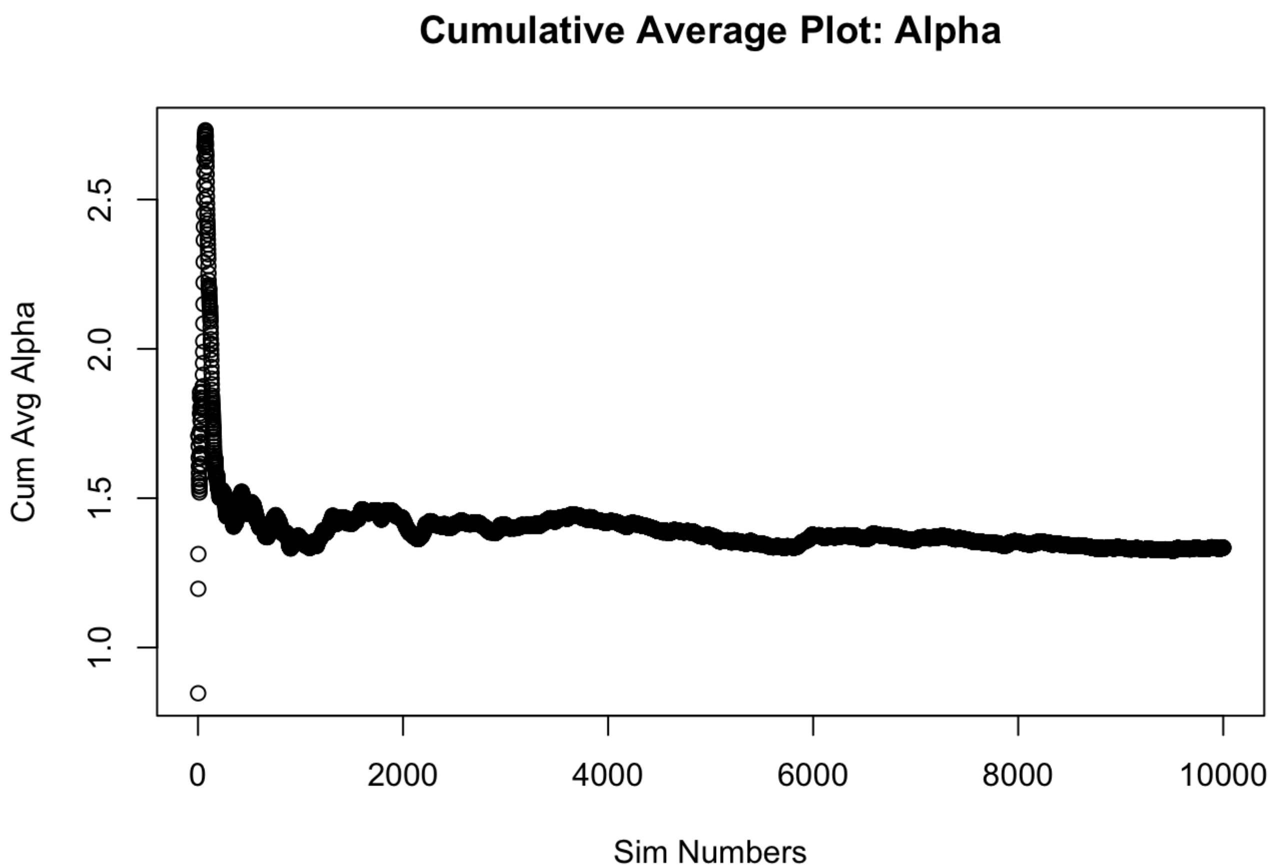
```
## [1] "(0.847,7.749)"
```

** Simulate values of α and β using the Metropolis algorithm. **

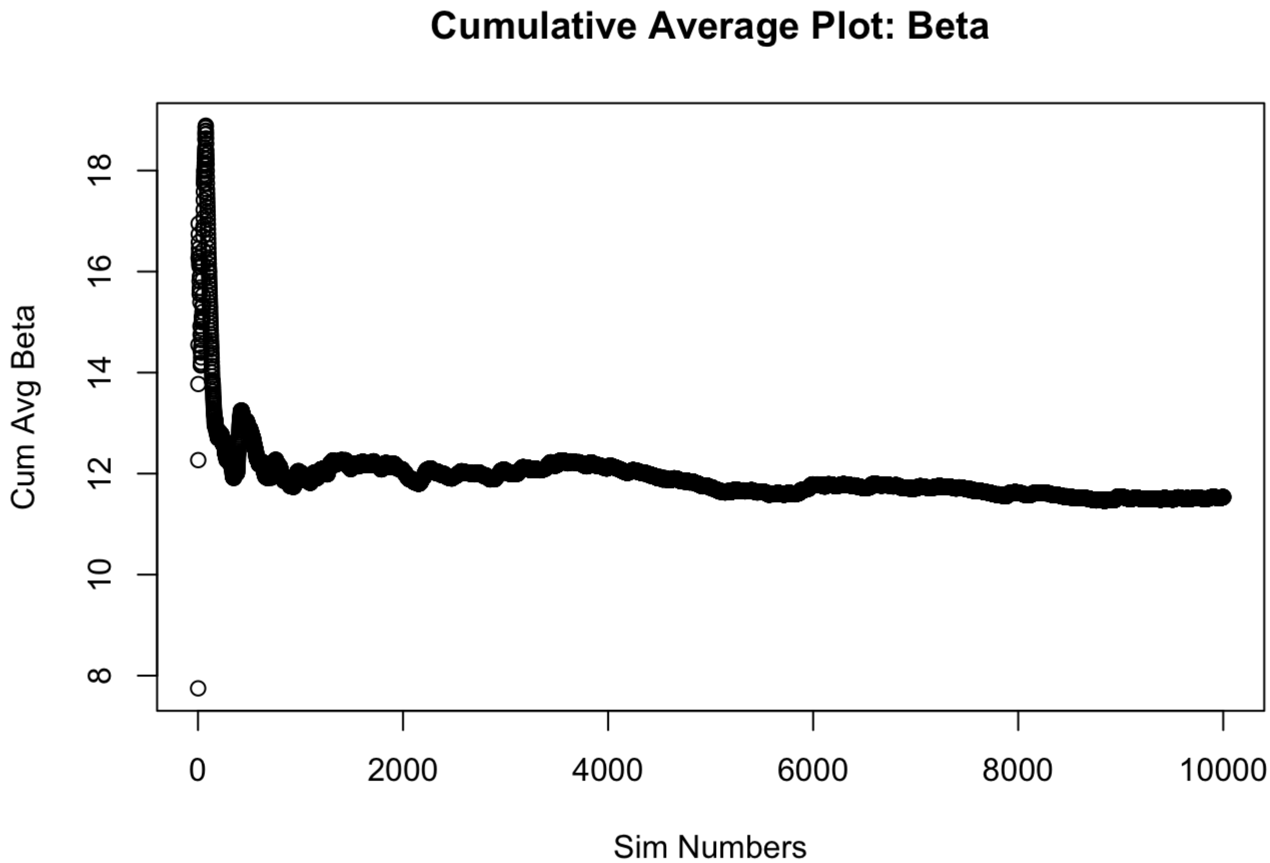
```
posterior       <-  function(theta) {
  alpha         <- theta[1]
  beta          <- theta[2]
  posterior     <-  1
  for (i in 1:length(y)) {
    posterior   <-  posterior * (
      ( ( inv.logit( alpha + beta * x[i] ) )^y[i] ) *
        ( ( 1 - inv.logit( alpha + beta * x[i] ) )^( n[i] - y[i] ) )
    )
  }
  posterior
}

# Proposal Distribution
jumping         <- function(theta){
  alpha_star <- rnorm(1, mean = theta[1], sd = 1)
  beta_star <- rnorm(1, mean = theta[2], sd = 4)
  return(c(alpha_star,beta_star))
}

sims <-  10000
theta <-  matrix(0,nrow = sims, ncol = 2)
theta[1,] <-  fit$coefficients

# Compute Acceptance Ratio and sample parameters
for (i in 2:sims)
{
  theta.star <-  jumping(theta[i-1,]) # new sample
  # Acceptance Ratio
  r <-  min( exp(
    ( log( posterior(theta.star) ) ) -
      ( log( posterior(theta[i-1,])) )
  ), 1)

  #print(r)
  if ( r >= runif(1) )
  {
    theta[i,]     <- theta.star # Accept
  }
  else
  {
    theta[i,]     <- theta[i-1,] # Reject
  }

}
```

** Convergence Diagnostics using Cumulative Average Plots of Parameters **

```
plot(cumsum(theta[ ,1])/c(1:sims), main = "Cumulative Average Plot: Alpha", xlab = "Sim
 Numbers", ylab = "Cum Avg Alpha")
```




```
plot(cumsum(theta[ ,2])/c(1:sims), main = "Cumulative Average Plot: Beta", xlab = "Sim N
umbers", ylab = "Cum Avg Beta")
```


Cumulative Average Plot: Beta
Cum Avg Beta
8
10
12
14
16
18
0
2000
4000
6000
8000
10000
Sim Numbers