Metropolis Hastings\_Linear Regression

Gordon Goodwin

8/11/2021

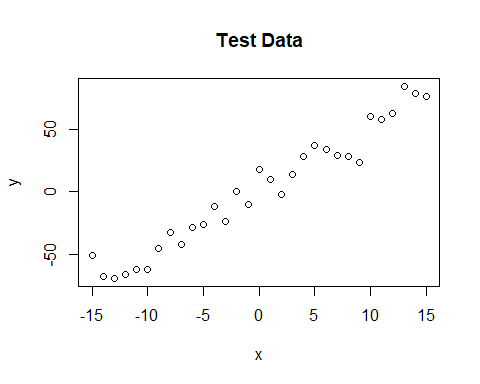
## Define True Parameters and Generate Test/Sample Data

We define the “true” underlying parameters for the linear model

* True Intercept = 0
* True Slope = 5
* Errors ~ N(0, sd)

Then we generate test data that simulates a sample from the population

true.b <- 5  
true.alpha <- 0  
true.sd <- 10  
N <- 31  
x <-(-15:15)  
  
y <- true.alpha + true.b \* x + rnorm(n=N,mean=0,sd=true.sd)  
  
plot(x,y, main="Test Data")



## Specify Generative Model and Likelihood Function

We know what the underlying population parameters are, so let’s fit the model and compare the MCMC-derived coefficient estimates based on our sample data to the true parameters

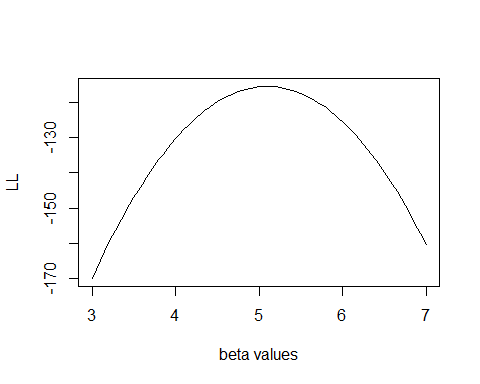
* First we need the likelihood function
* p(y | theta) -> p(y|Bx)

We model our errors as being distributed normally around zero, which is the same as modeling the actual y.i values as normally distributed around y.hat (B\*x)

* We can either use dnorm(e.i, zero, sd) or dnorm(y.i, Bx, sd)

The function below takes in a theta vector (specific set of parameter values) and outputs the log likelihood of the observed y\_i values given the param values

likelihood <- function(param){  
   
 alpha = param[1]  
 beta = param[2]  
 sd = param[3]  
   
 y.hat = alpha + beta\*x  
 ind.LL = dnorm(y, mean = y.hat, sd = sd, log = T)  
 LL = sum(ind.LL)  
 return(LL)   
}  
   
# Example: plot the likelihood profile of the slope (beta)  
beta.LLfun <- function(b){return(likelihood(c(true.alpha, b, true.sd)))}  
beta.LLres <- lapply(seq(3, 7, by=.05), beta.LLfun )  
plot (seq(3, 7, by=.05), beta.LLres , type="l", xlab = "beta values", ylab = "LL")



## Specify Prior

We will use uninformative prior distributions

* intercept ~ N(0, 1000)
* beta.1 ~ N(0, 1000)
* sd ~ invgamma(1, 1)

prior <- function(param){  
   
 alpha = param[1]  
 beta = param[2]  
 sd = param[3]  
   
   
 pr.alpha = dnorm(alpha, mean = 0, sd = 1000, log = T)  
 pr.beta = dnorm(beta, mean = 0, sd = 1000, log = T)  
 pr.sd = dinvgamma(sd, 1, 1, log = T)  
 pr.theta = sum(pr.alpha, pr.beta, pr.sd)  
 return(pr.theta)   
}

## Specifying the Posterior

Because we are using MCMC to approximate the posterior, we don’t need to use the normalized posterior, we can just sample from the joint distribution.

* p(theta | y) -> proportional to -> p(y|theta) \* p(theta)

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

## MCMC Metropolis Hastings Proposal Distribution

Sets the proposal distribution from which to draw candidate parameter values

* normal distribution centered at current param values

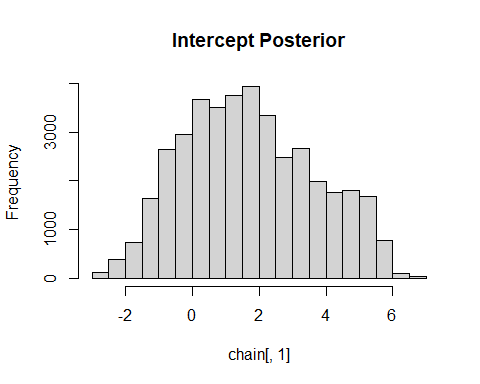
propose <- function(param){  
 return(rnorm(3,mean = param, sd= c(0.1,0.5,0.3)))  
}

## MCMC Metropolis Hastings

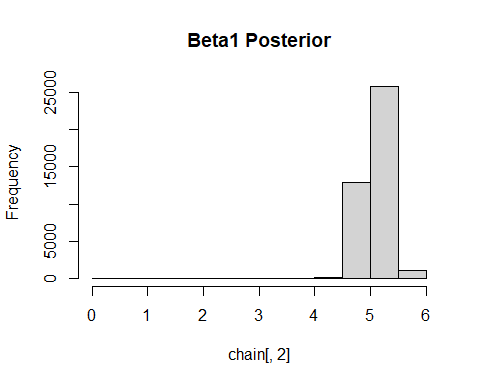
Now the MH algo puts it all together

* Draw candidate values from proposal distribution
* Accept/reject candidate values based on ratio of [p(y|theta)\*p(theta)]

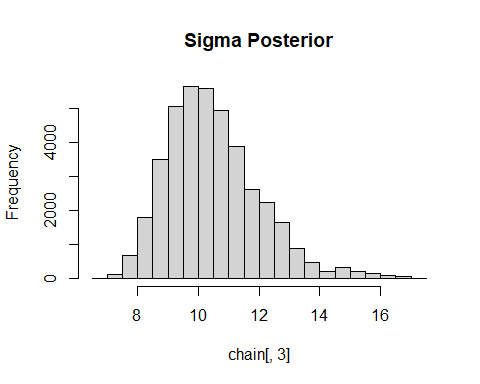
MCMC\_MH <- function(startval, iterations){  
 chain = array(dim = c(iterations+1,3)) # Structure of Chain Array  
 chain[1,] = startval # Establish Loop  
 for (i in 1:iterations){   
 proposal = propose(chain[i,]) # Propose theta[i+1] based on theta[i]  
   
 accept = exp(posterior(proposal) - posterior(chain[i,]))   
 if (runif(1) < accept){ #AR <-Post(theta[i+1]) to Post(theta[i])  
 chain[i+1,] = proposal   
 }else{  
 chain[i+1,] = chain[i,]  
 }  
 }  
 return(chain)  
}  
   
startval = c(4,0,10)  
chain = MCMC\_MH(startval, 40000)  
   
burnIn = 1000  
acceptance = 1-mean(duplicated(chain[-(1:burnIn),])) # Dupes = rejected params  
  
  
hist(chain[,1], main = "Intercept Posterior")



hist(chain[,2], main = "Beta1 Posterior")



hist(chain[,3], main = "Sigma Posterior")



## Plotting the Results

par(mfrow = c(2,3))  
hist(chain[-(1:burnIn),1],nclass=30, main="Intercept Posterior", xlab="True = red")  
abline(v = mean(chain[-(1:burnIn),1]))  
abline(v = true.alpha, col="red" )  
hist(chain[-(1:burnIn),2],nclass=30, main="Posterior Beta.1", xlab="True = red")  
abline(v = mean(chain[-(1:burnIn),2]))  
abline(v = true.b, col="red" )  
hist(chain[-(1:burnIn),3],nclass=30, main="Posterior Sigma", xlab="True = red")  
abline(v = mean(chain[-(1:burnIn),3]) )  
abline(v = true.sd, col="red" )  
plot(chain[-(1:burnIn),1], type = "l", xlab="True = red" , main = "Chain values of alpha", )  
abline(h = true.alpha, col="red" )  
plot(chain[-(1:burnIn),2], type = "l", xlab="True = red" , main = "Chain values of beta1", )  
abline(h = true.b, col="red" )  
plot(chain[-(1:burnIn),3], type = "l", xlab="True = red" , main = "Chain values of sd", )  
abline(h = true.sd, col="red" )

