Homework5

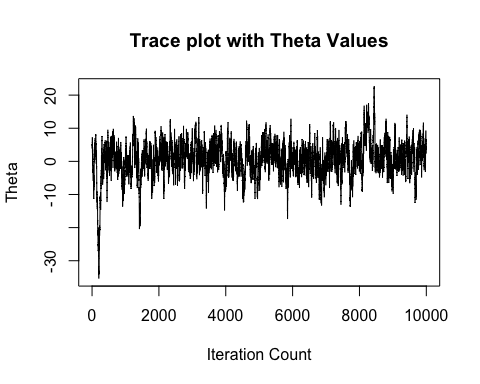
Tyler Poelking

2/22/2018

# function for calculating the acceptance probability  
calculate.alpha.unimodal <- function(value.start, value.end) {  
 theta.start.alpha <- ((1 + (1/5) \* (((value.start - 1)/4))^2)^-3)  
 theta.end.alpha <- ((1 + (1/5) \* (((value.end - 1)/4))^2)^-3)  
 alpha <- theta.end.alpha/theta.start.alpha  
 return(min(1, alpha))  
}  
   
#set theta starting.value  
starting.val = 4  
  
#set propostal standard dev  
proposal.sd = 2  
  
# K is the number of samples  
K <- 10000   
  
  
#Initialize vector for holding thetas   
thetas <- rep(0, K)  
  
#store initial value in first position of thetas  
thetas[1] <-starting.val  
  
#Initialize vector for holding accept and reject values  
success\_rej <- rep(0, K)  
  
#store initial value in first position of success\_rej  
success\_rej[1] <- starting.val  
  
#Run game for k iterations  
for(k in 2:K) {  
   
 #generate new theta value and store  
 theta.new <- rnorm(1, starting.val, proposal.sd)  
   
   
 p\_Reject <- calculate.alpha.unimodal(starting.val, theta.new)  
   
 #If probability of a rejection > than random number, replace initial theta and Store success w/ replacement or failure w/ no replacement  
 if(runif(1, 0, 1) < p\_Reject) {  
 starting.val <- theta.new  
 success\_rej[k] <- 1  
 }  
 else {  
 success\_rej[k] <- 0  
 }  
 thetas[k] <- starting.val  
}

1. Provide the following details about your algorithms: the starting value, the variance of proposal distribution (i.e., the tuning parameter), the number of iterations you ran your algorithm, how many interations you threw away to allow the algorithm to “burn-in,”" and the percentage of times your algorithm accepted a proposal after burn-in was achieved.```{r pressure.
2. Provide a trace plot demonstrating that your algorithm has converged. Explain why you think convergence has been achieved based on the plot.

#create data frame of data generated  
df <- data.frame(theta = thetas, Iterations = 1:K)  
  
#plot  
plot(df$Iterations, df$theta, type = "l", ylab = "Theta", xlab = "Iteration Count", main = "Trace plot with Theta Values")



#set burn in length  
burn.len <- 2600

The starting value was 4. The Variance was 2^2 = 4. The number of iterations was 10,000. The number of iterations thrown away to allow algorithm burn in was aproximately 25,000 The percentage of times algorithm accepted proposed theta: 0.67

The patterns in the trace plot are fairly consistent over a long number of iterations and no new patterns are forming. This provides evidence that the Metropolis algorithm has converged.

1. Using your posterior samples, estimate E[θ|y] and var[θ|y].

#E[θ|y]  
e.val <- mean(thetas[burn.len:K])  
e.val

## [1] 1.146068

#var[θ|y]  
v.val = var(thetas[burn.len:K])  
v.val

## [1] 22.00483

1. Using your posterior samples, estimate the posterior probability that θ>3.

p\_greaterThan3 <- mean(thetas[burn.len:K] > 3)  
p\_greaterThan3

## [1] 0.3309012