Lisa Over
HW 8 Metropolis Hastings
March 17, 2015

CODE

Uniform Model

#printPDF function receives one vector and a filename. It prints three plots for the vector to the specified filename: ts plot, hist, acf.

```
printPDF <- function(v1,filename) {
  pdf(filename)
   #PLOTS to PDF
     par(mfrow=c(3,1)) #split plotting window into 3 rows
     ts.plot(v1,xlab="Iterations")
     hist(v1,probability=T, cex.lab=1.5, cex.axis=1.5)
     acf(v1, lag.max=1000)
  dev.off()
}
```

#check_alpha function receives three parameters "alpha.star" for the parameter probability, "param.star" for the next generated parameter, and "param0" for the previously accepted parameter.

```
check_alpha <- function(alpha.star, param.star, param0) {

    #if alpha.star is less than 1, draw a random uniform to compare
    #if random uniform draw is less than alpha star, return param.star and count=1
    #else return param0 and count=0
    if(alpha.star < 1) {
            if(runif(1,0,1) < alpha.star) {
                    results <- list("param" = param.star, "count" = 1)
                    return(results)
            }
            else {
                    results <- list("param" = param0, "count" = 0)
                    return(results)
            }
    }
    #check if alpha star for parameter is equal to one, return parameter star and count=1
    else {
            results <- list("param" = param.star, "count" = 1)
            return(results)
    }
}
```

#The metro_hastings function receives six parameters: "x0" for the initial sample value, "k" for the interval, "N" for number of independent realizations, "lag" for determining how many

Lisa Over
HW 8 Metropolis Hastings
March 17, 2015

realizations to skip between saves, and "burnin" for detemrining how many realizations to skip before starting to save.

```
metro_hastings <- function(x0,k,N,lag,burnin) {

    #Set N to be N*lag+burnin
    N <- N*lag + burnin

    #Initialize vector to hold x
    x.s <- NULL

    #store the acceptance rate for x in 'x.cnt'
    x.cnt = 0

  for(i in 1:N) {

    #generate x from uniform proposal density with (k) as the interval limits
    x.star <- runif(1,x0-k,x0+k)
    while((x.star < 0) | (x.star > 1)) {
            x.star <- runif(1,x0-k,x0+k)
    }

    #alpha star for x using uniform density as target density and hastings correction
(proposal.ratio) to make density integrate to 1
    target.ratio = (1/(2*k))/(1/(2*k))
    proposal.ratio = max((1/(2*k)), (1/(x.star+k)), (1/(1-x.star+k)))/max((1/(2*k)),
(1/(x0+k)), (1/(1-x0+k)))
    alpha.star = min(target.ratio*proposal.ratio, 1)

    #use check_alpha function to determine if x.star should be accepted
    results = check_alpha(alpha.star,x.star,x0)

    #if i is greater than burnin and if i is a multiple of the lag, store value
    #add results$count to total x.cnt--results$count will be 0 if x.star was not accepted and 1 ow
    #add accepted value to vector - x0 or x.star was returned as results$param
    #set x0 equal to whatever check_alpha returned for next iteration
  if(i > burnin) {
    if(i %% lag == 0) {
            x.cnt = x.cnt + results$count
            x.s = c(x.s, results$param)
            x0 = results$param
   }
  }


  }
  filename = sprintf("Documents/R-FILES/HW8-hCorrect-lag-U-k-%s.pdf",k)
  printPDF(x.s,filename)
```

Lisa Over
HW 8 Metropolis Hastings
March 17, 2015

```
 vector <- list("x" = x.s, "xCount" = x.cnt)
 return(vector)
}

#Set burnin=20, lag=100, x0=(random uniform value), k=0.2; run Metro with N=5000
N = 5000
lag = 100
burnin = 20
k = 0.2
x0 = runif(1,0,1)

vector = metro_hastings(x0,k,N,lag,burnin)
```

Normal Model

#printPDF function receives two vectors and a filename. It prints three plots for each vector to the specified filename: ts plot, hist, acf.

```
printPDF <- function(v1,v2,alpha,lambda,filename) {
  pdf(filename)
   #PLOTS to PDF
     par(mfrow=c(3,2)) #split plotting window into 3 rows and 2 columns
     ts.plot(v1,xlab="Iterations")
     ts.plot(v2,xlab="Iterations")
     hist(v1,probability=T, cex.lab=1.5, cex.axis=1.5)
     hist(v2,probability=T, cex.lab=1.5, cex.axis=1.5)
     yy = seq(10,70,by=.001)
     alpha = (n/2)-1
     lambda = ((n-1)*s.sqr)/2
     lines(yy, ((lambda^alpha)/gamma(alpha))*((1/yy)^(alpha+1))*exp(-lambda/yy))
     acf(v1)
   acf(v2)
  dev.off()
}
```

#metro_hastings function receives seven parameters: "data" for the data values of interest, "m0" for the population mean, "s0" for the population variance, "b" for the mu interval, "c" for the sig sqr interval, "N" for number of independent random normal realizations, "lag" for determining how many realizations to skip between saves, and "burnin" for detemrining how many realizations to skip before starting to save.

```
metro_hastings <- function(data,m0,s0,b,c,N,lag,burnin) {

     #obtain mean (y.bar), variance (s.sqr), and length (n) of data
     y.bar = mean(data)
```

Lisa Over
HW 8 Metropolis Hastings
March 17, 2015

```
  s.sqr = var(data)
  n = length(data)

    #Set N to be N*lag+burnin
    N <- N*lag + burnin

    #Initialize vectors to hold mu and sigma sqr
    m.s <- NULL
    s.s <- NULL

    #store the acceptance rate for mu in 'mu.cnt' and sigma sqr in 'sig.cnt'
    mu.cnt = 0
    sig.cnt = 0

 for(i in 1:N) {

#Generate a mu star 'm.star' from the proposal density (Normal) using the population mu (m0) and (b) as the interval limits
    m.star = rnorm(1,m0,b)

#Compute alpha star for mu using mean (y.bar), variance (s.sqr), length (n) of data, and population variance (s0) -- Use target.ratio which is ratio of target density given m.star to the target density given m0 (target density is Normal)
    target.ratio = exp(-(1/(2*s0))*(n*(y.bar-m.star)^2 + m.star^2/10 - n*(y.bar-m0)^2 - m0^2/10))

#Use target.ratio*proposal.ratio to determine if s.star should be accepted
    save.param = 0
    accept.code = 0
        if(runif(1,0,1) < target.ratio) {
            save.param = m.star
            accept.code = 1
        }
        else {
            save.param = m0
            accept.code = 0
        }


#if i is greater than burnin and if i is a multiple of the lag, store value
#add accept.code to total mu.cnt - accept.code will be 0 if m.star was not accepted and 1 ow
#add accepted value - m0 or m.star as save.param
    if(i > burnin) {
    if(i %% lag == 0) {
        mu.cnt = mu.cnt + accept.code
        m.s = c(m.s, save.param)
  }
 }
```

Lisa Over
HW 8 Metropolis Hastings
March 17, 2015

```
  #set m0 equal to save.param for next iteration
  m0 = save.param

#Generate a sigma sqr star 's.star' from the proposal density using the population variance (s0) and (c) as the interval limits
    s.star = rnorm(1,s0,c)
    while(s.star < 0) {
        s.star = rnorm(1,s0,c)
    }

#Compute alpha star for sig sqr using mean (y.bar), variance (s.sqr), length (n) of data, population variance (s0) and mean (m0)

    #Use target.ratio which is ratio of target density given s.star to the target density given s0 (target density is inverse gamma)
    A.star = (1/s.star)^((n/2)+4)
    B.star = (1/(2*s.star))*((n - 1)*s.sqr + n*(y.bar-m0)^2 + m0^2/10 + 2)
    A.zero = (1/s0)^((n/2)+4)
    B.zero = (1/(2*s0))*((n - 1)*s.sqr + n*(y.bar-m0)^2 + m0^2/10 + 2)
    target.ratio = exp(log(A.star) - B.star - log(A.zero) + B.zero)

    #Multiply target.ratio by Hastings correction (proposal.ratio) which is the ratio of the proposal s.star density given s0 to the proposal s0 density given s.star -- proposal density is Normal -- Divide proposal densities by proportion of curve that appears after the boundary (above zero) so that the density equals one when integrated
    density.star = dnorm(s0, s.star, c)/(1-pnorm(0, s.star, c))
    density0 = dnorm(s.star, s0, c)/(1-pnorm(0, s0, c))
    proposal.ratio = density.star/density0

    #Use target.ratio*proposal.ratio to determine if s.star should be accepted
    save.param = 0
    accept.code = 0
        if(runif(1,0,1) < target.ratio*proposal.ratio) {
            save.param = s.star
            accept.code = 1
        }
        else {
            save.param = s0
            accept.code = 0
        }

#if i is greater than burnin and if i is a multiple of the lag, store value
    #add accept.code to total sig.cnt - accept.code will be 0 if s.star was not accepted and 1 ow
    #add accepted value - s0 or s.star as save.param
 if(i > burnin) {
    if(i %% lag == 0) {
        sig.cnt = sig.cnt + accept.code
        s.s = c(s.s, save.param)
```

Lisa Over
HW 8 Metropolis Hastings
March 17, 2015

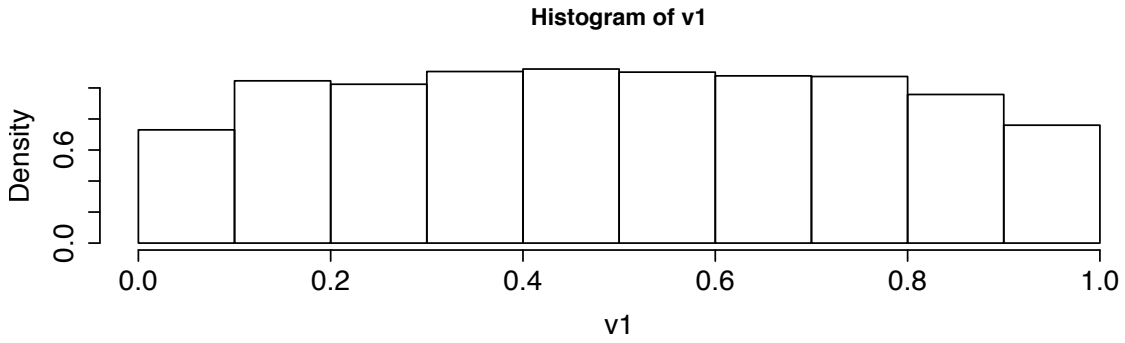
```
  }
  }
  #set s0 equal to results$param for next iteration
  s0 = save.param

  }

 #IG parameters to superpose a line over sigma sqr distribution
 alpha = length(data)/2 + 3.5
 lambda = ((length(data)-1)*var(data))/2 + 2
 #alpha = length(data)/2 - 1
 #lambda = ((length(data)-1)*var(data))/2

 filename = sprintf("Documents/R-FILES/HW8-hastings-N-b-%s-c-%s.pdf",b,c)
 printPDF(m.s,s.s,alpha,lambda,filename)

 vectors <- list("mu" = m.s, "muCount" = mu.cnt, "sigsqr" = s.s, "sigCount" = sig.cnt)
 return(vectors)

}

#Set burnin, lag, m0 of data, and s0; run metro to obtain N realizations
N = 10000
lag = 30
burnin = 0
b = .5
c = .5
m0 = mean(data2)
s0 = var(data2)

vectors = metro_hastings(data2,m0,s0,b,c,N,lag,burnin)

vectors$sigCount/N
vectors$muCount/N
```

Lisa Over
HW 8 Metropolis Hastings
March 17, 2015

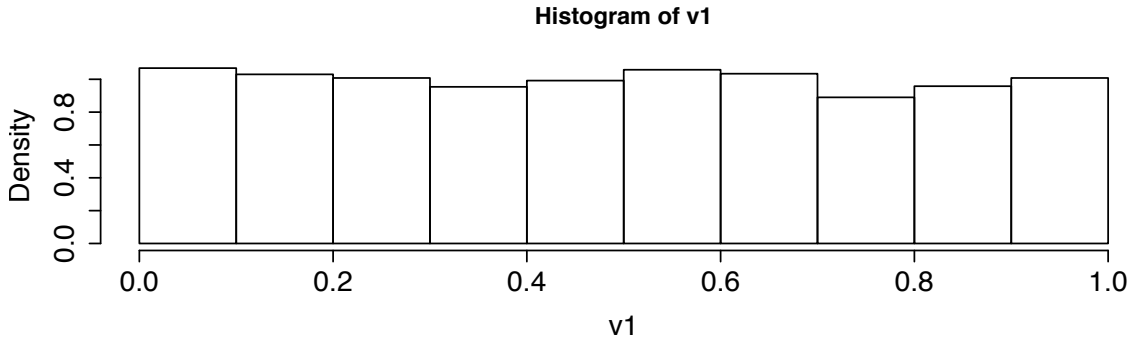
RESULTS

Uniform Model

Method 1



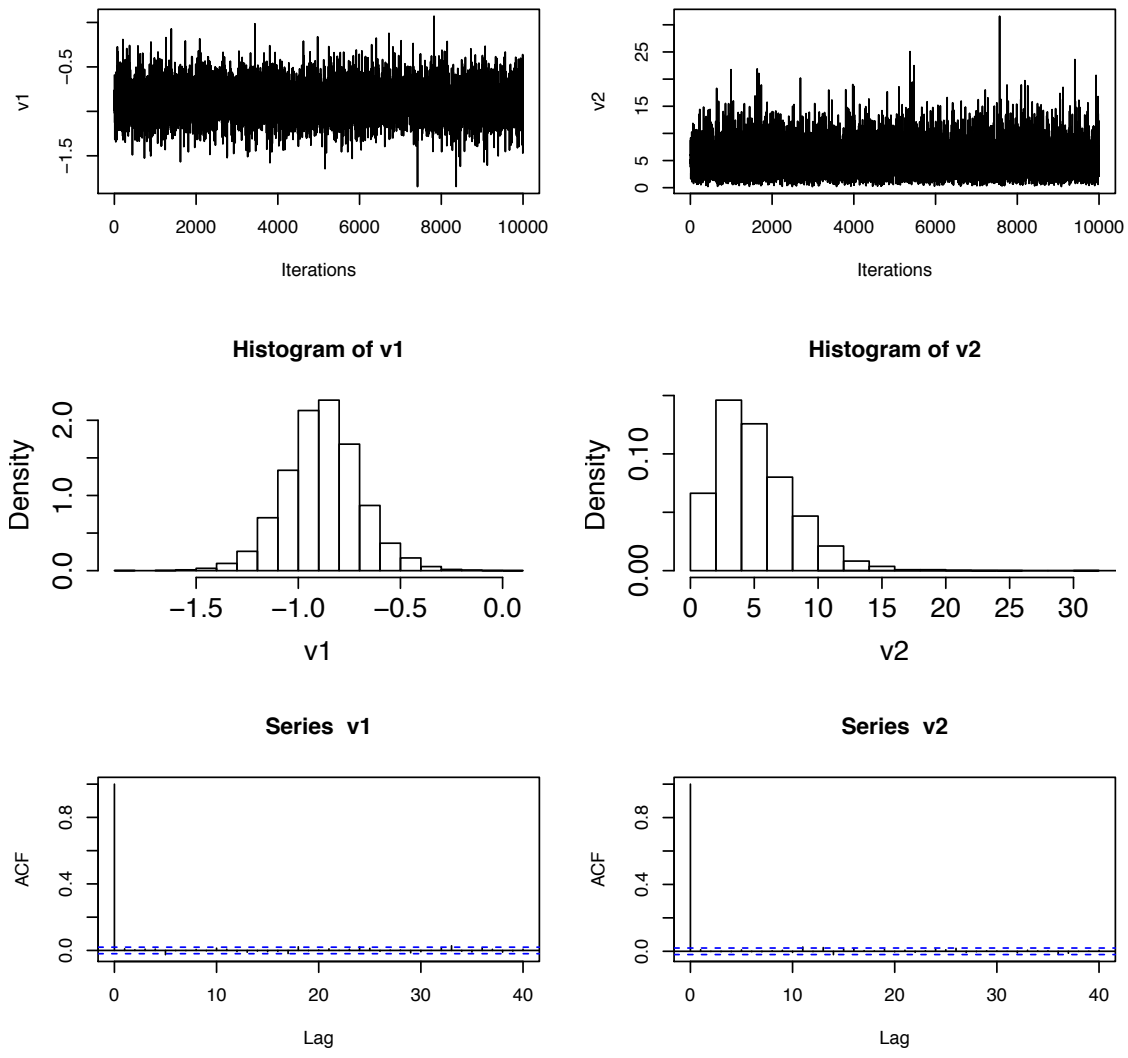
Method 2



The lag used to obtain independent draws was 100. The overall acceptance probability for method 1 (without the Hastings correction) was 1, and the overall acceptance probability for method 2 (with the Hastings correction) was 0.9462. The histogram for method 1 is not uniform. It rises slightly and then falls so that there is a small hump in the middle. The histogram for method 2 appears uniformly distributed. The histogram for model 1 is not uniform because when x.star is proposed from the interval (x0-k, x0+k) such that x.star < 0 or x.star > 1, it is rejected and another is drawn until one is found within the uniform interval (0, 1). This means that values close to the boundaries of 0 and 1 are not as

likely to be selected as those in the middle. To make the selection of values within the interval (0, 1) random, we use the Hastings correction to adjust the acceptance probability when near the parameter bounds. This makes values near the bounds as likely to be selected as those in the middle.

Normal Model



The lag used to obtain independent draws was 30. The overall acceptance probability for μ was 0.1099, and the overall acceptance probability for σ was 0.4287 when b=2 and c=4. The code does not run consistently so it was not possible to find the best b and c.

Lisa Over
HW 8 Metropolis Hastings
March 17, 2015

i. $\pi(\mu,\sigma|\vec{y}) = \frac{1}{\Gamma(2.5)\sqrt{20\pi}} \times (\frac{1}{\sigma^2})^{\frac{n}{2}+4} \times e^{-\frac{1}{2\sigma^2}[(n-1)s^2+n(\bar{y}-\mu)^2+\frac{\mu^2}{10}+2}$

ii. $Marginal\ posterior\ for\ \sigma^2 \propto (\frac{1}{\sigma^2})^{\frac{n}{2}+3.5} \times e^{\frac{-(n-1)s^2+2}{2\sigma^2}}$, which is

$\sim IG(\frac{n}{2}+3.5, \frac{(n-1)s^2+2}{2})$

iii. The variance, σ^2, requires the Hastings correction because σ^2 cannot be negative. Values near the boundary will not be accepted if the interval associated with the proposed σ^{2*} is below the 0 bound. To make σ^{2*} random, the Hastings correction adjusts the acceptance probability when near the 0 bound.

iv. $$\frac{q(\Theta_c^2|\Theta^{2*})}{q(\Theta^{2*}|\Theta_c^2)} = \frac{\frac{1}{\sqrt{2\pi v}}\times e^{-\frac{1}{2v}(\Theta_c^2-\Theta^{2*})^2}}{\frac{1}{\sqrt{2\pi v}}\times e^{-\frac{1}{2v}(\Theta^{2*}-\Theta_c^2)^2}} \times \frac{\frac{1}{\int_0^\infty \frac{1}{\sqrt{2\pi v}} e^{-\frac{1}{2v}(\Theta^{2*}-\Theta_c^2)^2} d\Theta_c^2}}{\frac{1}{\int_0^\infty \frac{1}{\sqrt{2\pi v}} e^{-\frac{1}{2v}(\Theta_c^2-\Theta^{2*})^2} d\Theta^{2*}}}$$

v. The σ^2 of the data used in homework 7 was large enough that any proposed σ^{2*} would be far away from the 0 bound. All possible σ^{2*} were equally likely to be selected.