Lisa Over
HW 7
March 10, 2015

CODE

Print Function

```
#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,n,s.sqr,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, lag.max=1000)
   acf(v2, lag.max=1000)
  dev.off()

}
```

Check Alpha Function

```
#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)
        }

}
```

Metropolis Function

#metro function receives eight 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 realizations, "lag" for determining how many realizations to skip between saves, "burnin" for detemrining how many realizations to skip before starting to save, and "flag" indicating the distribution to be used. Gibbs generates N independent random normal values.

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

        #obtain mean (y.bar), variance (s.sqr), and length (n) of data
        y.bar = mean(data)
      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 sigma sqr star 's.star' using the population variance (s0) and (c) as the interval
limits
        #Normal Distribution
        if(flag == "UNIFORM") {
                if(s0 < c) {
                        s.star <- runif(1,0,s0+c)
                }
```

```
		else {
			s.star <- runif(1,s0-c,s0+c)
		}
	}
	#Uniform distribution
	else if(flag == "NORMAL") {
		s.star = rnorm(1,s0,c)
			}

	#generate a mu star 'm.star' using the population mu (m0) and (b) as the interval limits
	#Uniform distribution
	if(flag == "UNIFORM") {
		m.star <- runif(1,m0-b,m0+b)
	}
	#Normal distribution
	else if(flag == "NORMAL") {
		m.star = rnorm(1,m0,b)
	}

	#alpha star for mu using mean (y.bar), variance (s.sqr), length (n) of data, and population
variance (s0)
	alpha.star = min(exp(-(n/(2*s0))*((y.bar-m.star)^2 - (y.bar-m0)^2)), 1)

	#use check_alpha function to determine if m.star should be accepted
	results = check_alpha(alpha.star,m.star,m0)
	#add results$countto total mu.cnt - results$count will be 0 if m.star was not accepted and 1
if m.star was accepted
	mu.cnt = mu.cnt + results$count
	#add accepted value - m0 or m.star was returned as results$param
	m.s = c(m.s, results$param)
	#set m0 equal to results$param for next iteration
	m0 = results$param

	#alpha star for sig sqr using mean (y.bar), variance (s.sqr), length (n) of data, population
variance (s0) and mean (m0)
	A.star = (1/s.star)^((n/2)+1)
	B.star = (1/(2*s.star))*((n - 1)*s.sqr + n*(y.bar-m.star)^2)
	A.zero = (1/s0)^((n/2)+1)
	B.zero = (1/(2*s0))*((n - 1)*s.sqr + n*(y.bar-m.star)^2)

	#alpha star for sigma sqr
	alpha.star = min(exp(log(A.star) - B.star - log(A.zero) + B.zero), 1)

	#use check_alpha function to determine if s.star should be accepted
	results = check_alpha(alpha.star,s.star,s0)
	#add results$countto total sig.cnt - results$count will be 0 if s.star was not accepted and 1 if
s.star was accepted
	sig.cnt = sig.cnt + results$count
```

Lisa Over
HW 7
March 10, 2015

```
    #add accepted value - s0 or s.star was returned as results$param
    s.s = c(s.s, results$param)
    #set s0 equal to results$param for next iteration
    s0 = results$param
 }

 if(flag == "UNIFORM") {
    filename = sprintf("Documents/R-FILES/HW7-U-b-%s-c-%s.pdf",b,c)
 }
 else if(flag == "NORMAL") {
    filename = sprintf("Documents/R-FILES/HW7-N-b-%s-c-%s.pdf",b,c)
 }
 printPDF(m.s,s.s,n,s.sqr,filename)

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

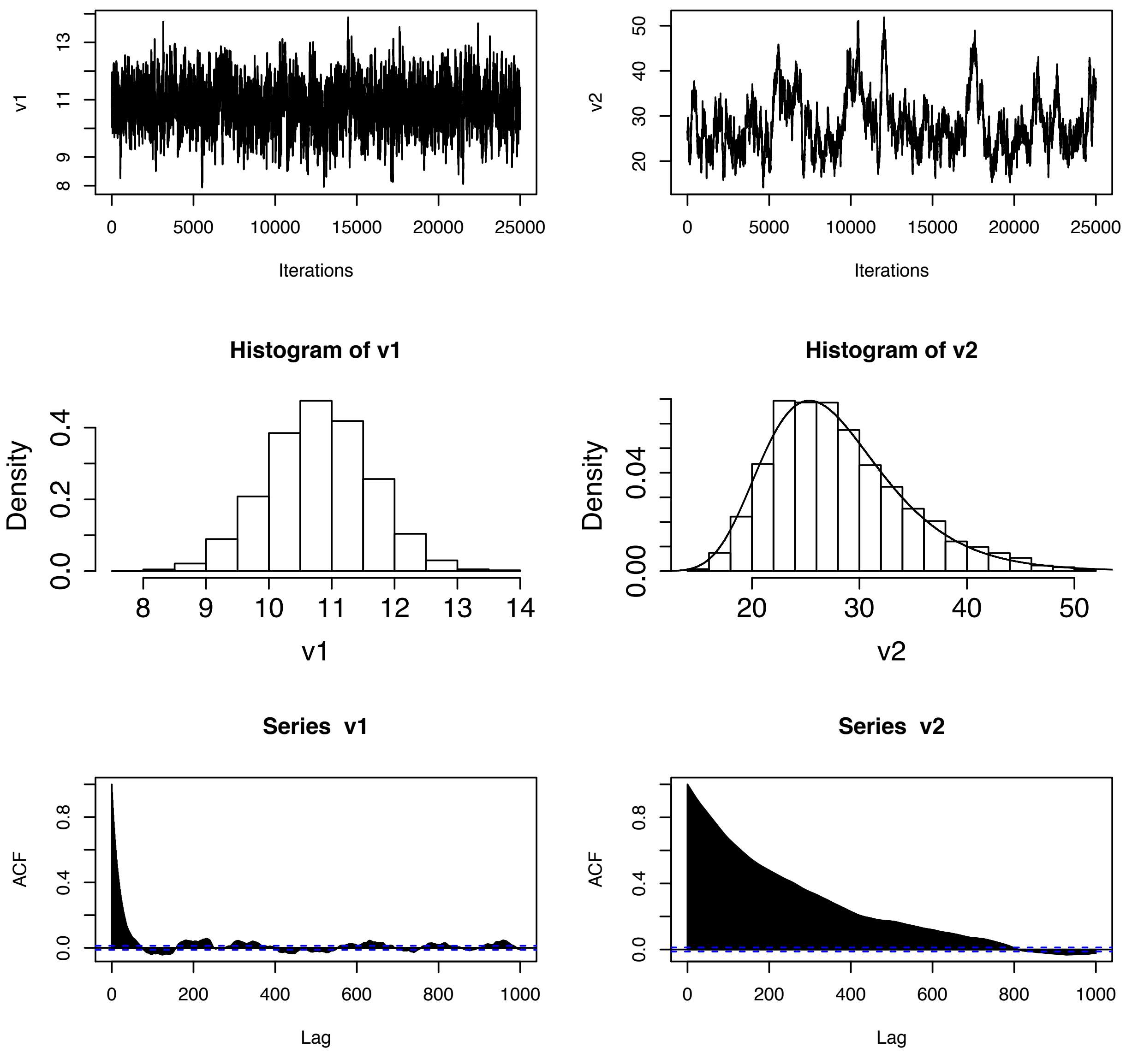
Attach File, Run Metro, and Obtain Results

```
data = read.table(file.choose(), header=F)
attach(data)
data = data$V1

#Set burnin=0, lag=1, m0=mean of data, and s0=var of data; run Metro with N=25000
#Run four scenarios: (1) b=.5, c=1,  flag="UNIFORM" (2) b=2, c=4,  flag="UNIFORM" (3) b=.5,
c=1,  flag="NORMAL" (4) b=2, c=4,  flag="NORMAL"
N = 25000
lag = 1
burnin = 0
```
b =0.5
c = 1
flag = "UNIFORM"
```
m0 = mean(data)
s0 = var(data)

vectors = metro(data,m0,s0,b,c,N,lag,burnin,flag)

#Acceptance Probabilities
vectors$sigCount/N
vectors$muCount/N
```

Lisa Over
HW 7
March 10, 2015

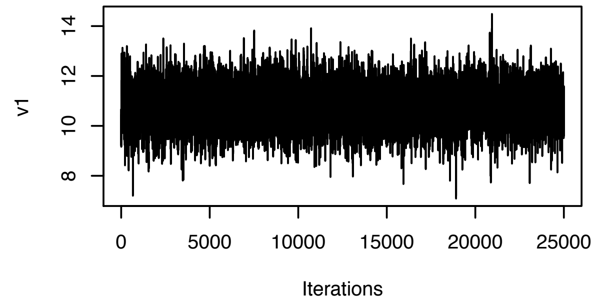
RESULTS

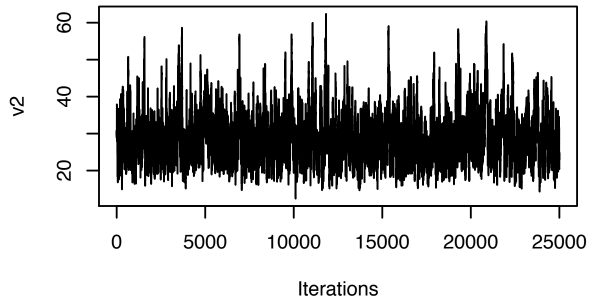
Plots for b=0.5, c=1, UNIFORM

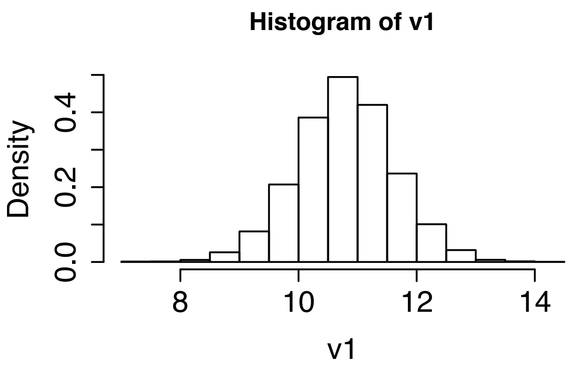


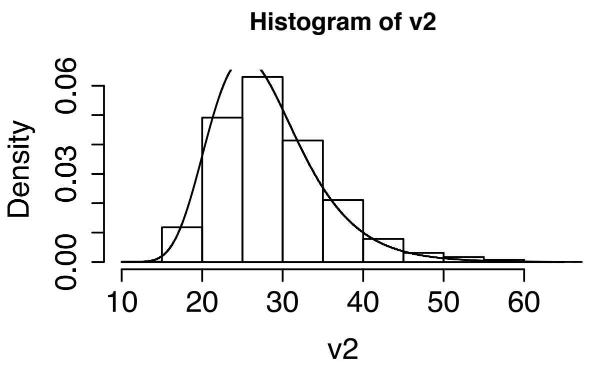
Lisa Over
HW 7
March 10, 2015

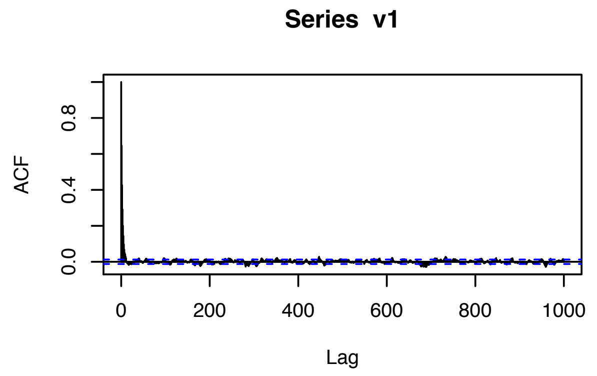
Plots for b=2, c=4, UNIFORM

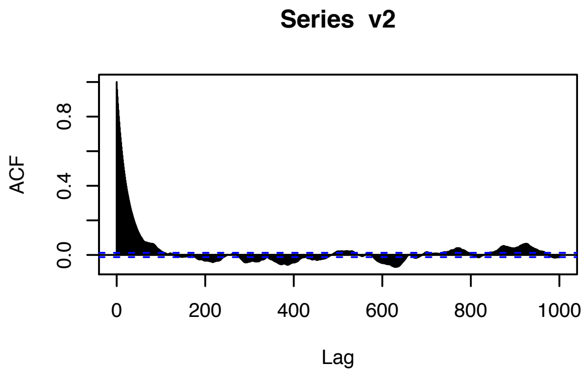






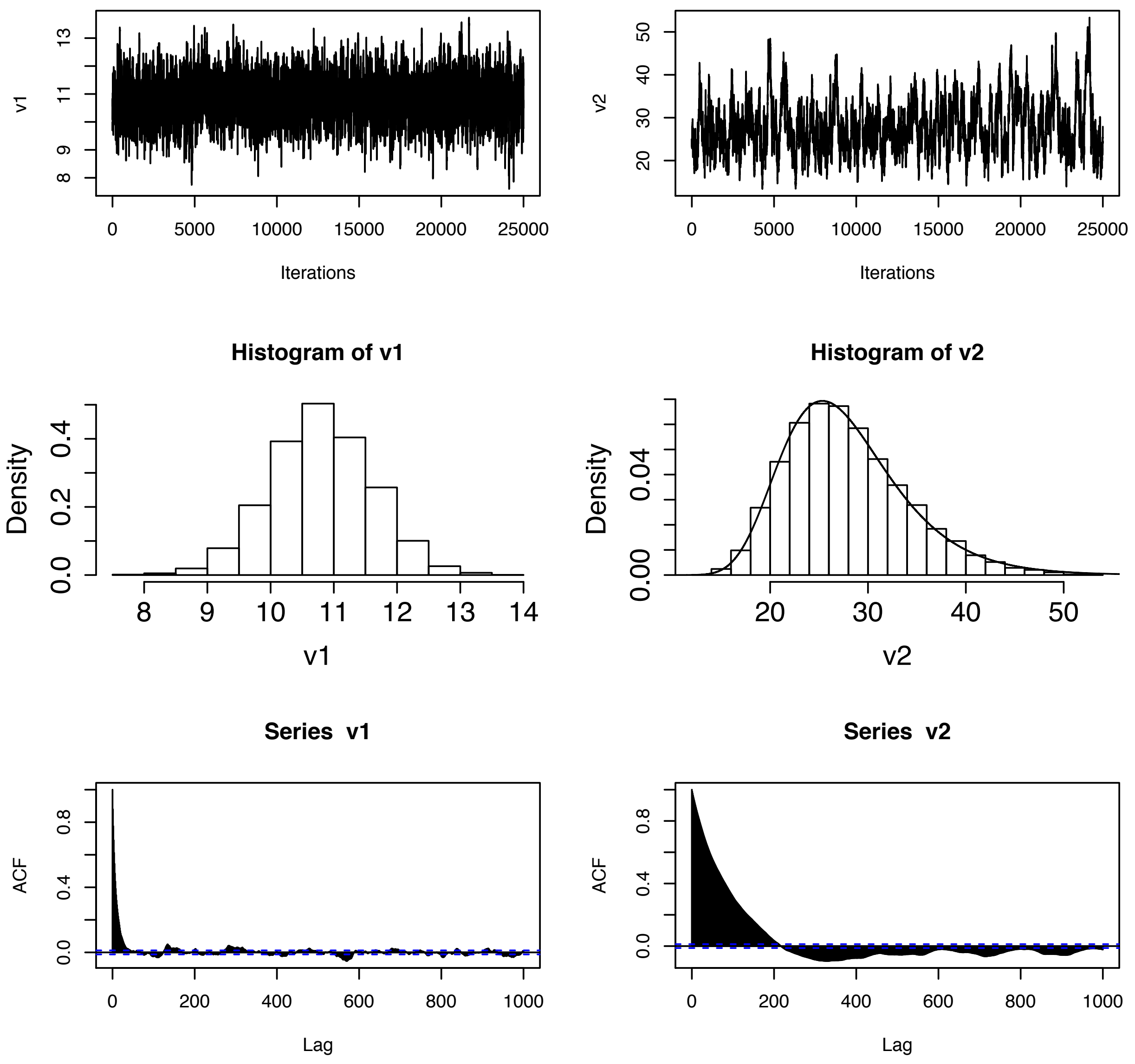






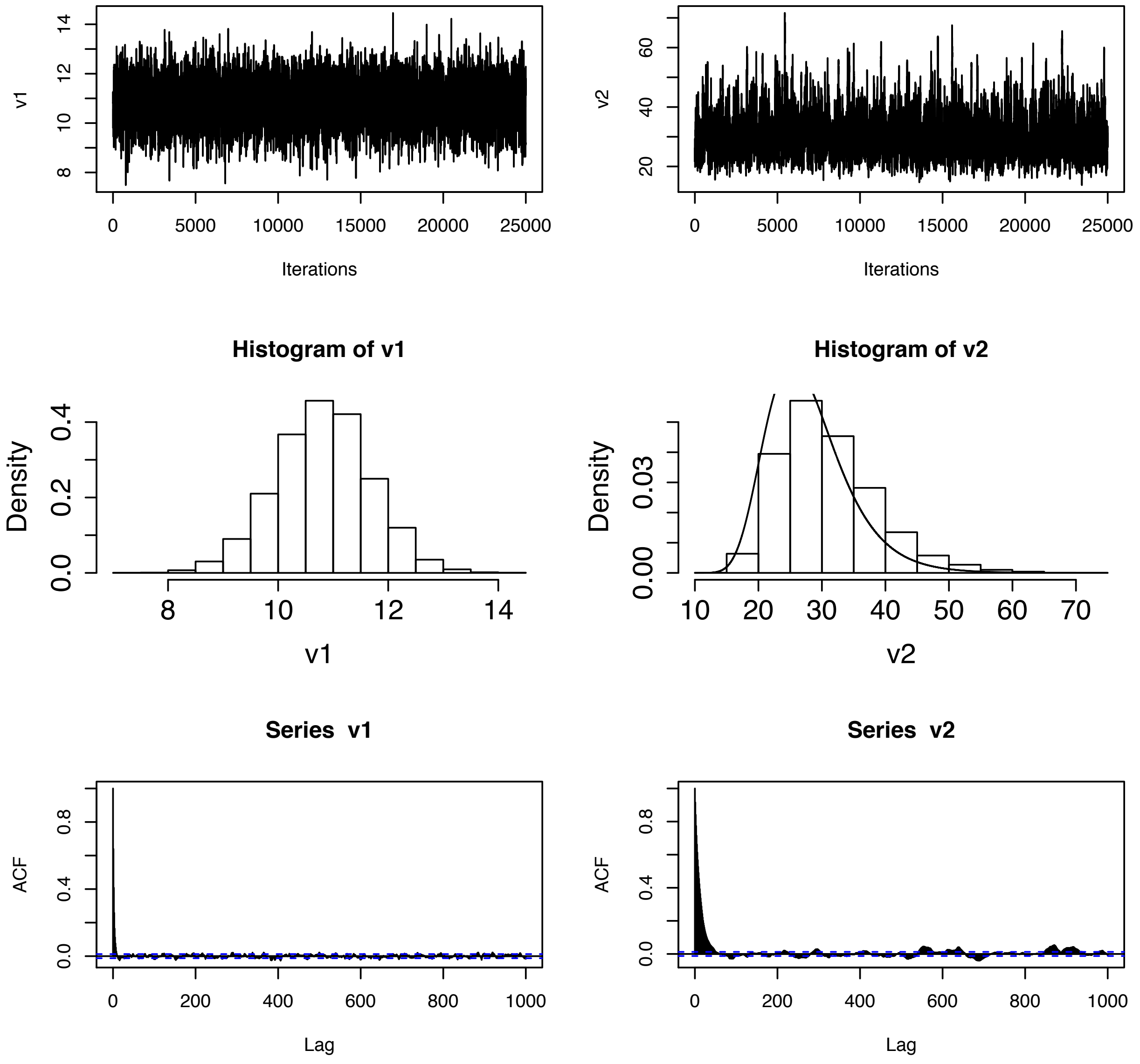
Lisa Over
HW 7
March 10, 2015

Plots for b=0.5, c=1, NORMAL



Lisa Over
HW 7
March 10, 2015

Plots for b=2, c=4, NORMAL



Lisa Over
HW 7
March 10, 2015

Acceptance Probabilities and Lags

	b=0.5, c=1, UNIFORM	**b=2, c=4, UNIFORM**	**b=0.5, c=1, NORMAL**	**b=2, c=4, NORMAL**
μ Acceptance Probability	0.8713	0.5624	0.8047	0.4408
σ^2 Acceptance Probability	0.9634	0.8598	0.9412	0.7866
Lag	850	150	220	75

The samples with b=2 and c=4 perform the best with respect to the acceptance probabilities not being too high. Both the sample drawn from the uniform distribution and the sample drawn from the Normal distribution, when b=2 and c=4, have lower acceptance probabilities than the two samples when b=0.5 and c=1.

The acceptance probability of 0.56 for the mean, μ, of the sample drawn from a uniform distribution is fairly close to the desired 66% rate. The acceptance probability for the standard deviation, σ^2, from the same sample is high at 0.86.

The acceptance probability of 0.78 for the standard deviation, σ^2, of the sample drawn from a Normal distribution is fairly close to the desired 66% rate. The acceptance probability for the mean, μ, from the same sample is low at 0.44.

With respect to acceptance probabilities, the sample that performs the best is the one drawn from the Normal distribution when b=2 and c=4.

The sample drawn from a Normal distribution when b=2, c=4 performs the best with respect to the lag not being too high. The samples drawn from a uniform distribution have a significantly higher lag than do the Normal sample counterparts.

Taking acceptance probabilities and lag into consideration, the sample that performs the best is the sample drawn from a Normal distribution with b=2 and c=4.