STAT4CI3/6CI3 Computational Methods for Inference

Assignment 3 Solutions

R code for this solution in a plain text file is also available separately

---

Q. 1 **a)** Suppose that $X \sim \text{gamma}(k, \beta)$ then the moment generating function of X can be written as

$$\begin{aligned} M_X(t) &= \int_0^\infty \mathrm{e}^{tx} \frac{1}{\Gamma(k)\beta^k} x^{k-1} \mathrm{e}^{-x/\beta}\, dx \\ &= \int_0^\infty \frac{1}{\Gamma(k)\beta^k} x^{k-1} \mathrm{e}^{-(1/\beta - t)x}\, dx \\ &= \frac{1}{\beta^k (1/\beta - t)^k} \int_0^\infty \frac{(1/\beta - t)^k}{\Gamma(k)} x^{k-1} \mathrm{e}^{-(1/\beta - t)x}\, dx \\ &= (1 - \beta t)^{-k} \qquad \text{for } t < \frac{1}{\beta} \end{aligned}$$

Now an exponential random variable with mean β has the same distribution as a gamma random variable with parameters 1 and β and so has moment generating function $(1 - \beta t)^{-1}$. Now suppose that $Y_1, \ldots, Y_k$ are independent exponential(β) random variables and let $X = \sum Y_i$ then

$$M_X(t) = \mathrm{E}\left(\mathrm{e}^{t \sum Y_i}\right) = \prod_{i=1}^{k} \mathrm{E}\left(\mathrm{e}^{tY_i}\right) = \left[M_{Y_1}(t)\right]^k = (1 - \beta t)^{-k} \qquad \text{for } t < \frac{1}{\beta}$$

and so when k is an integer we see that $X \sim \text{gamma}(k, \beta)$ can be written as a sum of independent exponential random variables with mean β. Furthermore we know from class examples that an exponential random variable with mean β can be generated as

$$Y = -\beta \log(U) \qquad \text{where } U \sim \text{uniform}(0, 1).$$

[4 marks]

Here is an R function to use this fact to generate gamma random variates.

```
rgamma.1a <- function(N,k,beta) {
#
# Function to generate N gamma random variates as a sum of
# independent exponentials which are generated from uniforms.
#
# Arguments
#   N: Sample size required
#   k: The shape parameter of the gamma which must be a
#       positive integer
#   beta: The scale parameter which must be positive
#
  if (k%%1 != 0) stop("k must be an integer")
  if (k<0 | beta<0) stop("Both parameters must be positive")
  U <- matrix(runif(N*k), ncol=k)
  Y <- -beta*log(U)
  rowSums(Y)
}
```

[3 marks]

b) To find the optimal value of b we need to maximize the acceptance probability or minimize

$$M = \sup_{x>0} \frac{f(x;\alpha,\beta)}{f(x;k,b)}$$

where f is the gamma probability density function

$$f(x;\alpha,\beta) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1}\,\mathrm{e}^{-x/\beta}$$

First we need to find M for a fixed value of b. To do that we must find the value of x which maximizes

$$\frac{f(x;\alpha,\beta)}{f(x;k,b)} = \frac{\Gamma(k)b^k}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-k} \exp\left\{-\left(\frac{1}{\beta}-\frac{1}{b}\right)x\right\}.$$

We need only worry about the parts involving x so we shall consider the function

$$\begin{aligned}
h(x) &= \log\left(x^{\alpha-k}\exp\left\{-\left(\frac{1}{\beta}-\frac{1}{b}\right)x\right\}\right)\\
&= (\alpha-k)\log x - \left(\frac{1}{\beta}-\frac{1}{b}\right)x\\
h'(x) &= \frac{\alpha-k}{x} - \frac{1}{\beta} + \frac{1}{b}\\
h''(x) &= -\frac{\alpha-k}{x^2} < 0 \quad \text{provided } \alpha > k
\end{aligned}$$

Since we are selecting $k < \alpha$ by design, $h(x)$ is maximized at $\tilde{x}$ which satisfies $h'(\tilde{x}) = 0$ which is clearly seen to be

$$\tilde{x} = \frac{\alpha-k}{\frac{1}{\beta}-\frac{1}{b}}$$

We note from this that we require $b > \beta$ in order for the accept-reject algorithm to work. If $b \leqslant \beta$ then the ratio of densities is unbounded.

Hence we have that, for fixed $b > 0$,

$$M(b) = \frac{f(\tilde{x};\alpha,\beta)}{f(\tilde{x};k,b)} = \frac{\Gamma(k)b^k}{\Gamma(\alpha)\beta^\alpha}\left(\frac{\alpha-k}{\frac{1}{\beta}-\frac{1}{b}}\right)^{\alpha-k}\exp\{-(\alpha-k)\}.$$

[4 marks]

To minimize this we need only consider the part of the $M(b)$ which depends on b.

$$\begin{aligned}
M(b) &\propto b^k \left(\frac{1}{\beta} - \frac{1}{b}\right)^{k-\alpha} \\
\log M(b) &= C + k \log b + (k-\alpha) \log \left(\frac{1}{\beta} - \frac{1}{b}\right) \\
\frac{\partial}{\partial b} \log M(b) &= \frac{k}{b} + \frac{k-\alpha}{\frac{1}{\beta} - \frac{1}{b}} \times \frac{1}{b^2} \\
&= \frac{k}{b} + \frac{\beta(k-\alpha)}{b^2 - b\beta}
\end{aligned}$$

Setting this equal to zero at b_{opt} gives

$$\begin{aligned}
\frac{k}{b_{\text{opt}}} + \frac{\beta(k-\alpha)}{b_{\text{opt}}(b_{\text{opt}} - \beta)} = 0 \quad &\iff \quad (b_{\text{opt}} - \beta)k + \beta(k-\alpha) = 0 \\
&\iff \quad b_{\text{opt}}k - \beta\alpha = 0 \\
&\iff \quad b_{\text{opt}} = \frac{\alpha\beta}{k}
\end{aligned}$$

Furthermore we can show that the second derivative of $\log M(b)$ is positive at $b = b_{\text{opt}}$ and so this point minimizes the upper bound and so maximizes the acceptance probability. Note that when $b = b_{\text{opt}}$ we have $\tilde{x} = \alpha\beta$. **[4 marks]**

The following R code will generate a sample of random size based on $N = 10000$ draws from the candidate distribution.

```
alpha <- 3.2
beta <- 2
k <- floor(alpha)
b <- alpha*beta/k
x.tilde <- alpha*beta
M <- dgamma(x.tilde,alpha,scale=beta)/dgamma(x.tilde,k,scale=b)
N <- 10000

set.seed(21032019)
Y.1b <- rgamma.1a(N,k,b)
U.1b <- runif(N)
p.1b <- dgamma(Y.1b,alpha,scale=beta)/(M*dgamma(Y.1b,k,scale=b))
accept.1b <- U.1b < p.1b
X.1b <- Y.1b[accept.1b]
```

[3 marks]

c) Since the question asks us to use the same set of candidate draws, we clearly are doing an independence Metropolis–Hastings procedure. The following code will take the random variates generated in part (b) and use them to construct the Metropolis–Hastings chain.

```
x0 <- rgamma(1,alpha,scale=beta)
X.1c <- rep(NA, N)
accept.1c <- rep(NA, N)
p.1c <- rep(NA,N)
for (i in 1:N) {
  if (i ==1) x <- x0
  else x <- X.1c[i-1]
  y <- Y.1b[i]
  p.1c[i] <- dgamma(y,alpha,scale=beta)/dgamma(y,k,scale=b)*
               dgamma(x,k,scale=b)/dgamma(x,alpha,scale=beta)
  p.1c[i] <- min(p.1c[i],1)
  accept.1c[i] <- U.2b[i]< p.1c[i]
  if (accept.1c[i]) X.1c[i] <- y
  else X.1c[i] <- x
}
```

[4 marks]

d) For the accept-reject algorithm we notice that we accept 9679 out of the 10000 generated observations which is a very high acceptance rate. We get such high acceptance since the candidate and target distributions are so close. In the Metropolis–Hastings method we accept 9807 which is a slightly better acceptance rate as guaranteed by the theory. Note that the sample size for the Metropolis–Hastings method is guaranteed to be 10,000 irrespective of the acceptance rate but for high acceptance rates, the chain is closer to a sequence of independent observations and so Monte Carlo inference is more accurate.

A cross tabulation of the acceptances is

	Metropolis-Hastings		
Accept-Reject	Accept	Reject	Total
Accept	9679	0	9679
Reject	128	193	321
Total	9807	193	10000

Thus we see that whenever the Metropolis–Hastings rejects a move, the corresponding accept-reject test also rejects but for 128 occasions in which the Metropolis–Hastings accepts a move, the accept-reject still rejects the candidate. This is guaranteed since the acceptance probability for the Metropolis–Hastings is guaranteed to be greater than that for the accept-reject algorithm.

The mean and variance of the values from the accept-reject algorithm are 6.3747 and 13.0175 which are very close to the true population values of 6.4 and 12.8. For the Metropolis–Hastings algorithm the corresponding values are 6.3850 and 13.0202 which are slightly further away but still very close. It seems that, for this example, there is not much difference between the two methods at all, primarily because we are using a candidate density which is almost indistinguishable from the target.

[3 marks]

Q. 2 **a)** Here is an R function to implement the Hastings example.

```
rnorm.hastings <- function(N, x0, delta) {
#
# An implementation of the Hastings random walk algorithm
# to generate a chain of N standard normal random variates
# starting at x0 and using uniform (-delta, delta) step sizes
# for the candidate at each iteration.  The algorithm will
# return the observed chain as well as the corresponding
# acceptance chain.
#
   eps <- runif(N, -delta, delta)
   U <- runif(N)
   X <- rep(x0, N+1)
   accept <- rep(NA, N)
   for (i in 1:N) {
     y <- X[i]+eps[i]
     accept[i] <- U[i] < dnorm(y)/dnorm(X[i])
     if (accept[i]) X[i+1] <- y
     else X[i+1] <- X[i]
   }
   list(sample=X[-1], accept=accept)
}
```

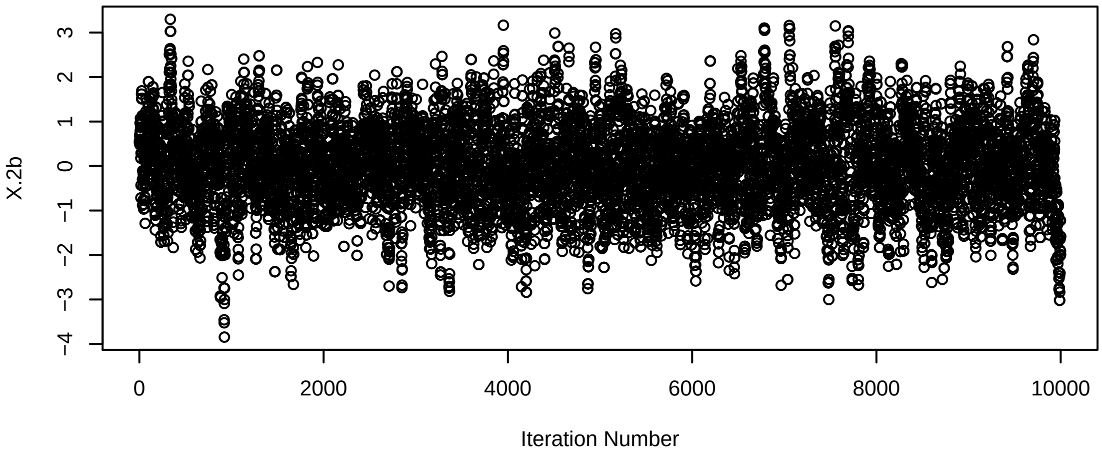
[5 marks]

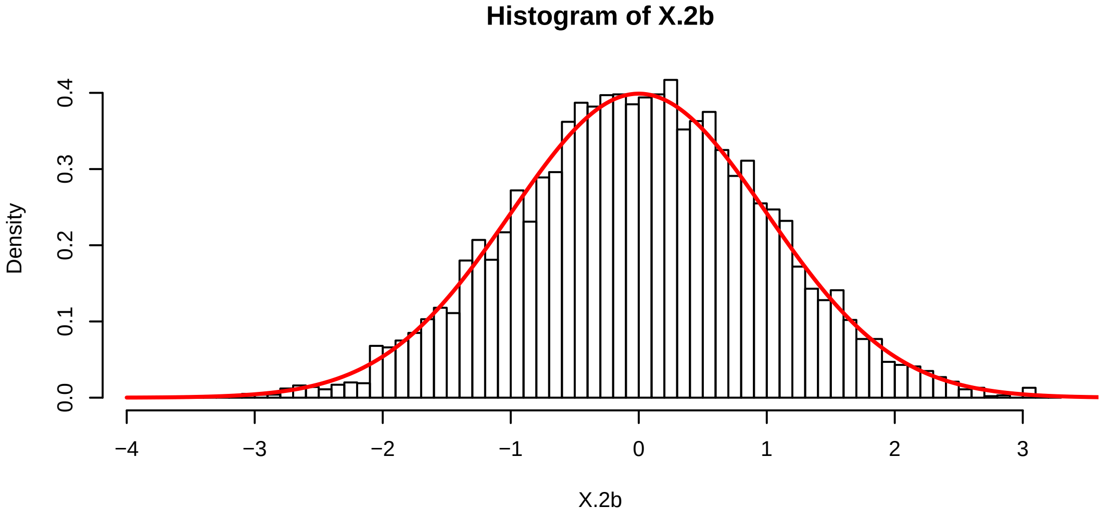
b) The plots required can be found using the following code

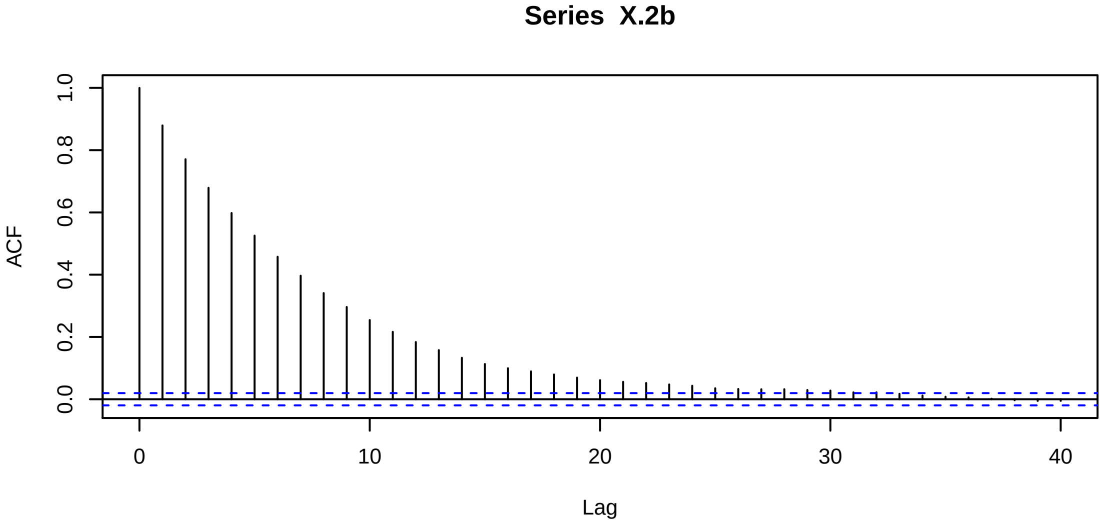
```
N <- 10000
N0 <- 5000
set.seed(18032019)
out.2b <- rnorm.hastings(N+N0, 0, 1)
X.2b <- out.2b$sample[-(1:N0)]
par(mfcol=c(3,1))
plot(X.2b, xlab="Iteration Number")
hist(X.2b, breaks=100, prob=TRUE)
lines(seq(-4,4,by=0.01), dnorm(seq(-4,4,by=0.01)), col="red", lwd=2)
acf(X.2b, lag.max=200)
```

[5 marks]

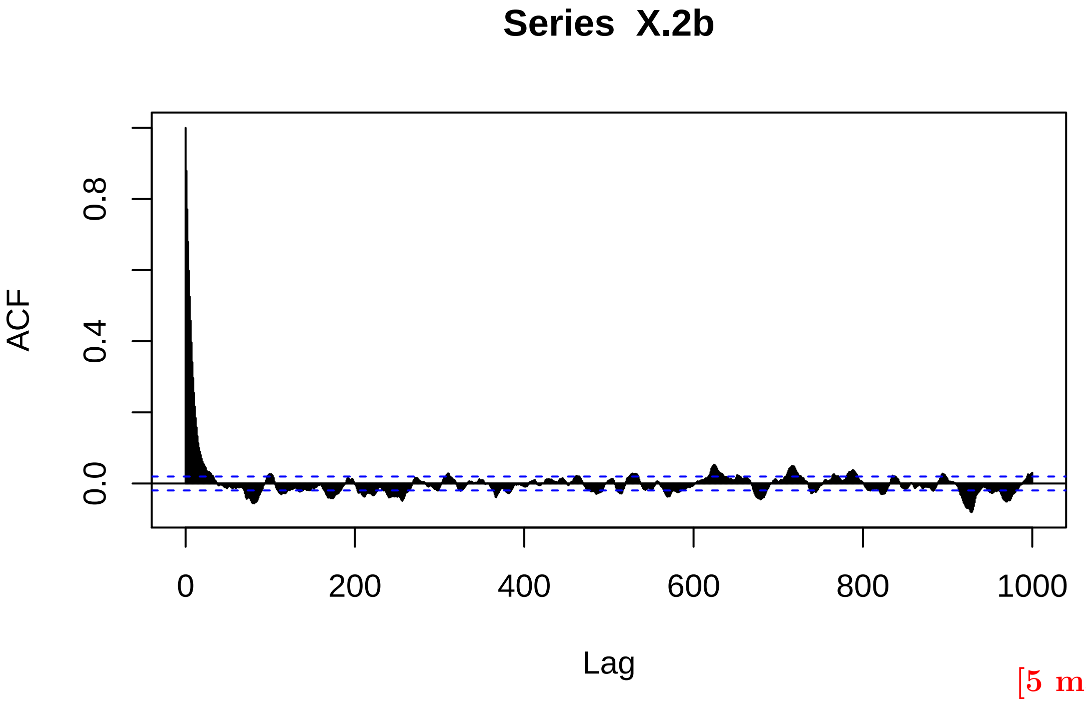
The plots show that the chain does move quite freely around the sample space and that the resulting chain looks very much like a sample from the standard normal. The auto-correlation function goes to 0 reasonably quickly and there seems to be little or no correlation for lags greater than about 25.




Histogram of X.2b


Series X.2b

The plot of the auto-correlation function, however, does not tell the whole story since it stops at a lag of 40 by default. If we examine much longer lags we see something quite different. Below is a plot with the maximum lag equal to 1000 iterations. This shows that there are small (but significantly different than 0) autocorrelations in both directions for very long lags.



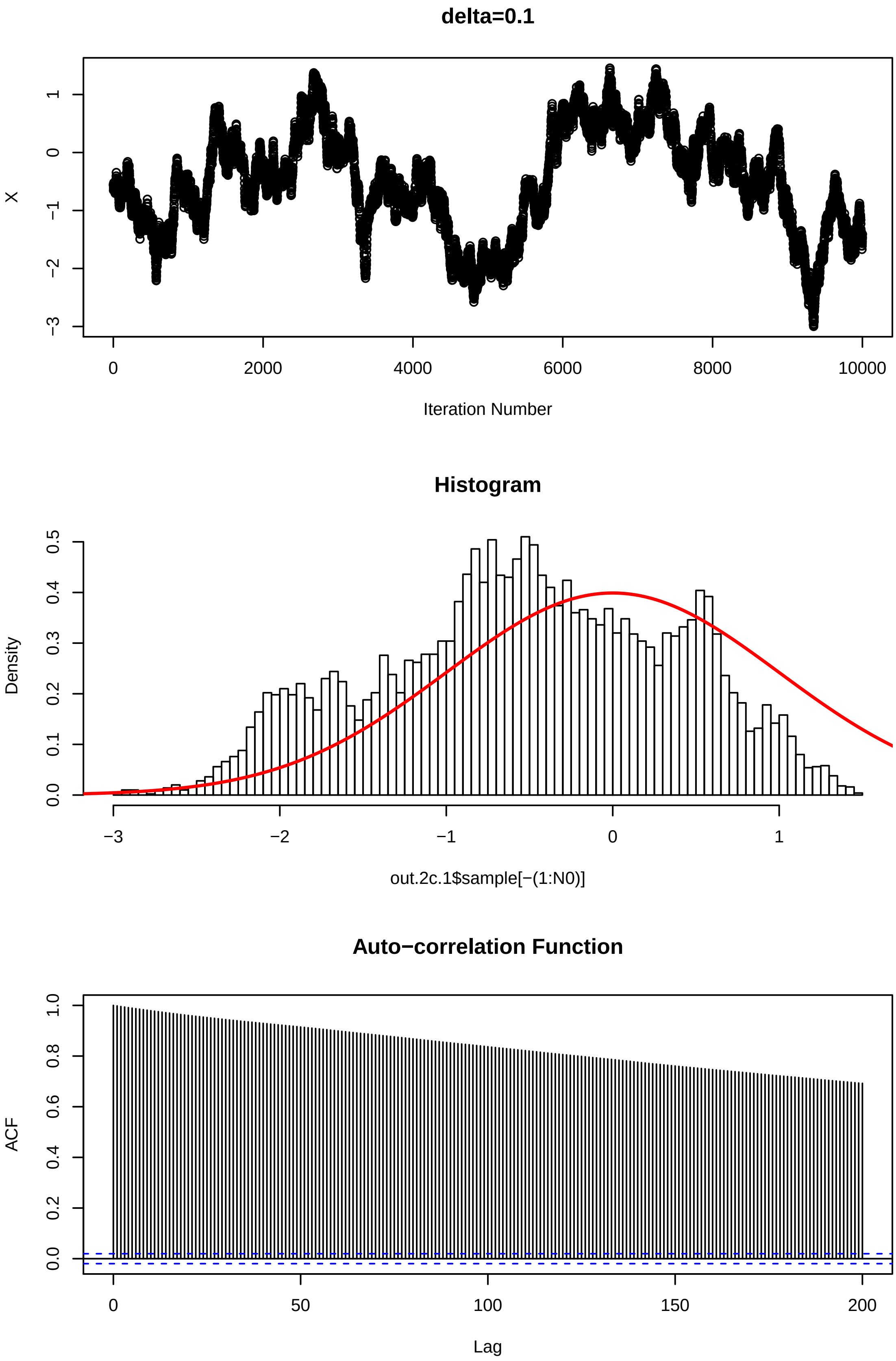
[5 marks]

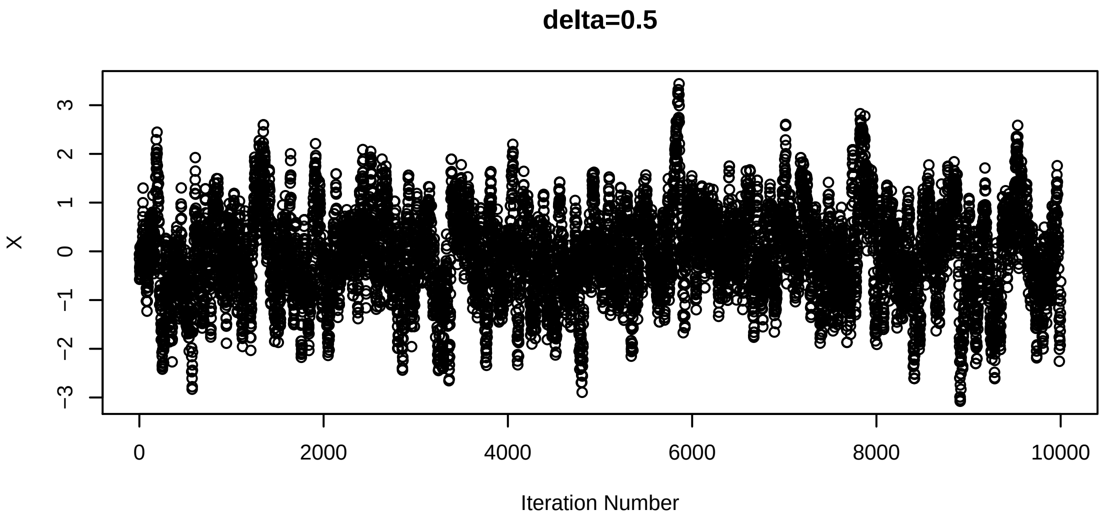
c) The code for this part of the question is exactly the same as for the previous part with different values of delta. I set the seed before each run so that the only difference is in the value of delta used. The four sets of plots are on subsequent pages. Looking at the plots of the chains against iteration number we see that the chains take a long time to vary over the entire sample space for smaller values of δ but they mix quite well for larger values. With $\delta = 10$ however the plot seems quite sparse since there are many rejections resulting in short sequences of iterations staying at the same spot. The histogram when $\delta = 0.1$ does not seem centred at the correct spot. In my case it produced many more negative than positive values. For other values of δ the histograms seem quite good. From the autocorrelation plots we see that increasing the value of δ clearly reduces the long-term autocorrelation, although, as we saw with $\delta = 1$, there remain small long-run auto-correlations. [3 marks]

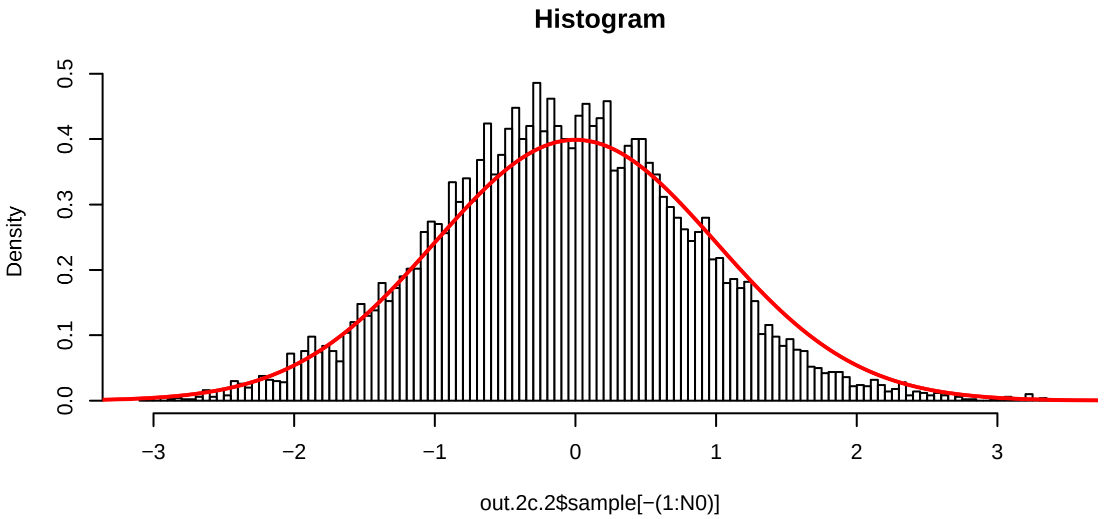
Here are some summary statistics and the proportion of acceptances over the retained iterations of each chain. These confirm that taking $\delta = 1$ or $\delta = 2$ seems to give the best results.

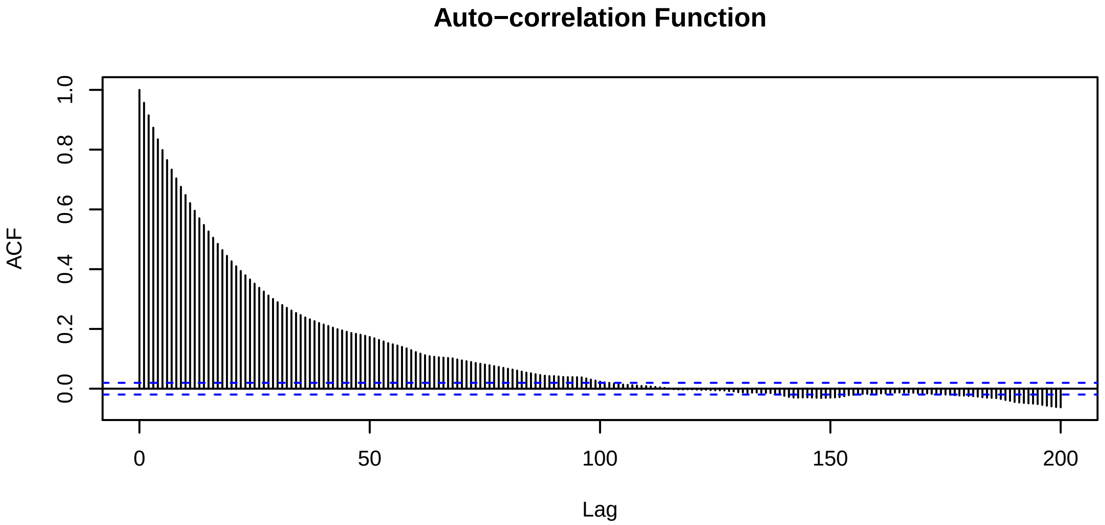
	Acceptance		Standard	
δ	Rate	Mean	Deviation	Median
0.1	0.9788	−0.5161	0.8796	−0.5030
0.5	0.9072	−0.0887	0.9166	−0.0950
1	0.8043	0.0132	0.9827	0.0184
2	0.6331	−0.0149	0.9985	−0.0112
10	0.1599	−0.03199	1.0193	0.0560

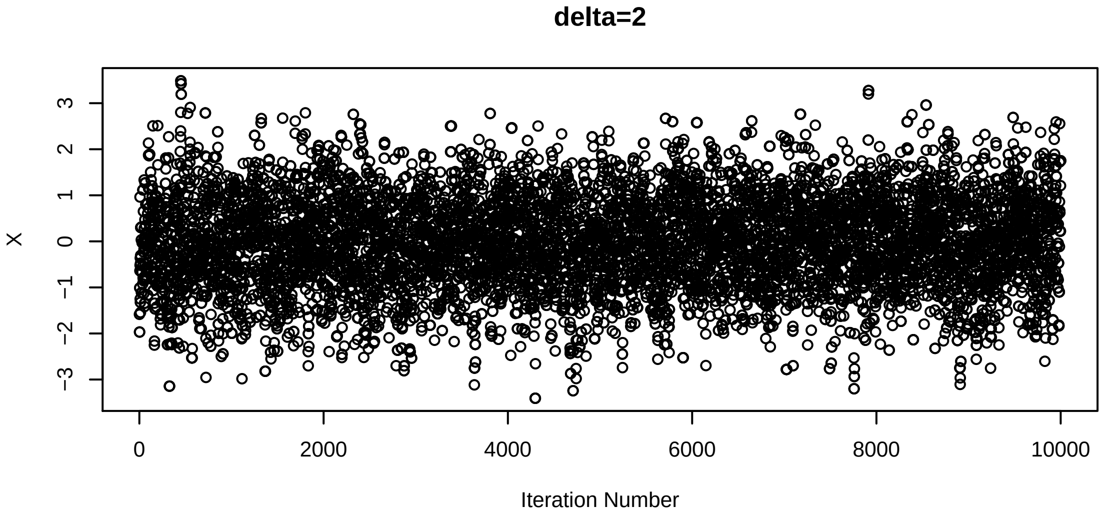
[2 marks]

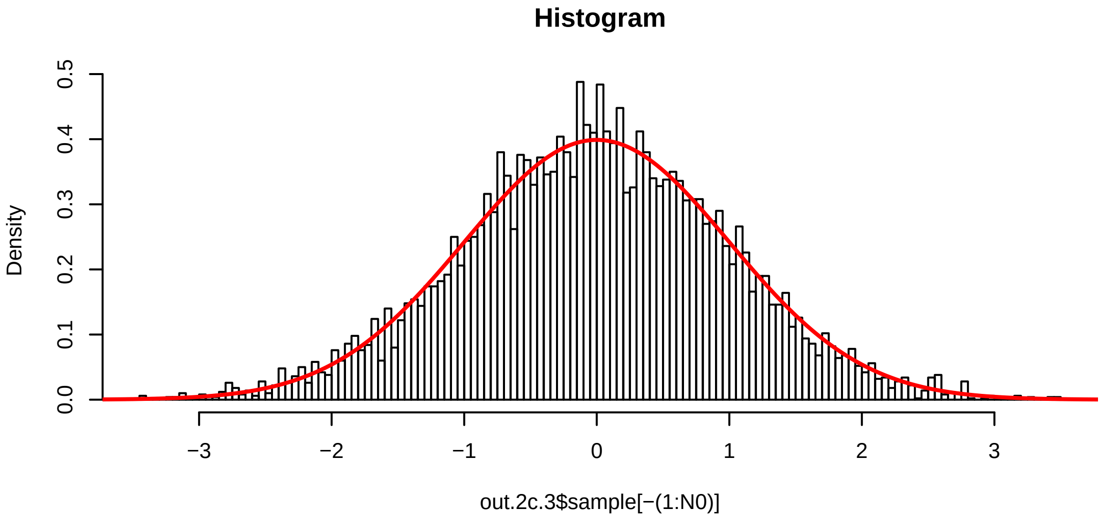

delta=0.1
X
1
0
−1
−2
−3
0
2000
4000
6000
8000
10000
Iteration Number
Histogram
Density
0.5
0.4
0.3
0.2
0.1
0.0
−3
−2
−1
0
1
out.2c.1$sample[−(1:N0)]
Auto−correlation Function
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
Lag

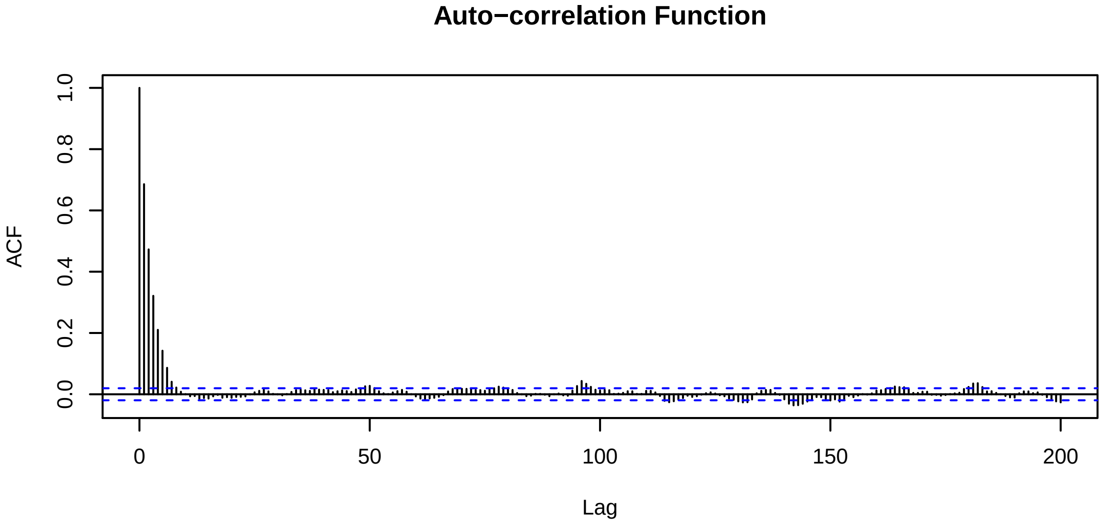

delta=0.5
X
3
2
1
0
−1
−2
−3
0
2000
4000
6000
8000
10000
Iteration Number

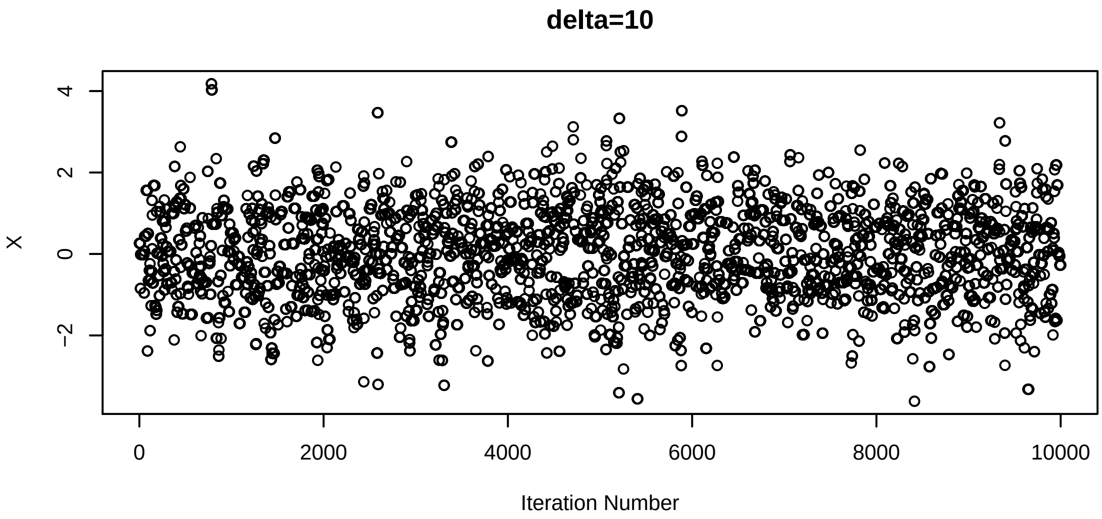

Histogram
Density
0.5
0.4
0.3
0.2
0.1
0.0
−3
−2
−1
0
1
2
3
out.2c.2$sample[−(1:N0)]

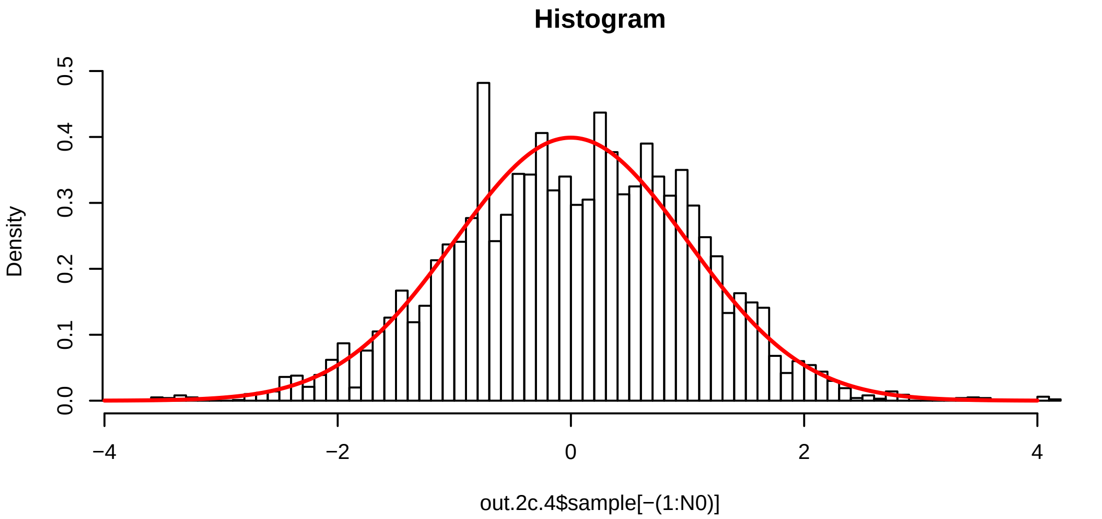

Auto−correlation Function
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
Lag

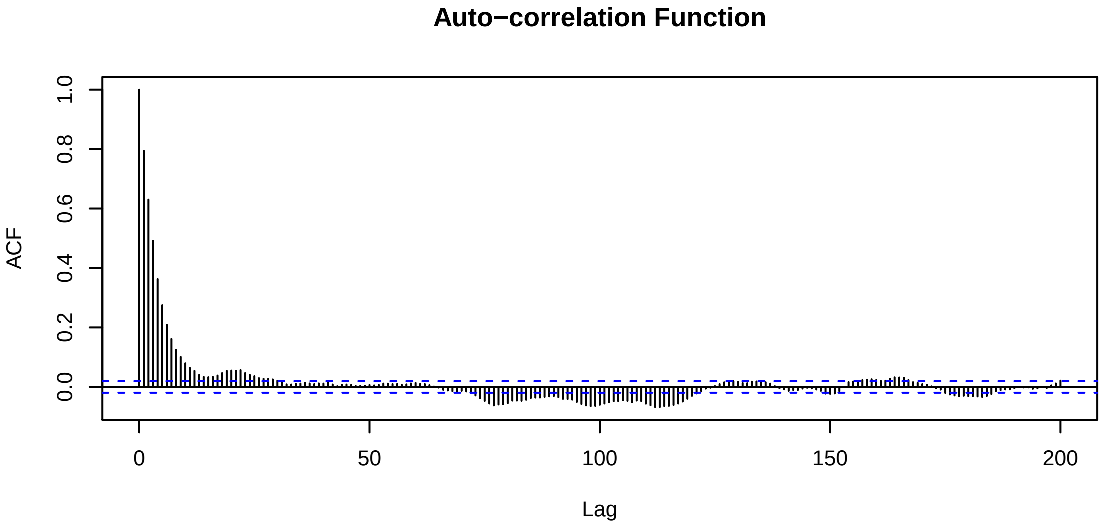

delta=2
X
3
2
1
0
−1
−2
−3
0
2000
4000
6000
8000
10000
Iteration Number


Histogram
Density
0.0
0.1
0.2
0.3
0.4
0.5
−3
−2
−1
0
1
2
3
out.2c.3$sample[−(1:N0)]


Auto−correlation Function
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
Lag


delta=10
X
4
2
0
−2
0
2000
4000
6000
8000
10000
Iteration Number


Histogram
Density
0.5
0.4
0.3
0.2
0.1
0.0
−4
−2
0
2
4
out.2c.4$sample[−(1:N0)]


Auto−correlation Function
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
Lag

[5 marks]

Q. 3 **a)** The following code will generate the bivariate chain. Note that the constant is not required in the bivariate density since it will cancel in calculating the acceptance probabilities.

```
rbinorm.mh <- function(N, mu1, mu2, s1, s2, rho) {
#
# Function to run a Metropolis-Hastings algorithm to
# generate bivariate normals using a random walk with
# independent standard normal innovations.
#
# Arguments:
#  N - length of the chain
#  mu1, mu2 - marginal means
#  s1, s2 - marginal standard deviations
#  rho - correlation
#
   dbinorm <- function(x, mu1, mu2, s1, s2, rho) {
   # Local function to evaluate the bivariate normal
   # density up to the constant which will cancel in
   # the acceptance probabilities anyway.
       z1 <- (x[1]-mu1)/s1
       z2 <- (x[2]-mu2)/s2
       exp(-1/(2*(1-rho^2))*(z1^2-2*rho*z1*z2+z2^2))
   }

   x0 <- rnorm(2, c(mu1, mu2), c(s1,s2))
   eps <- cbind(rnorm(N,0,s1), rnorm(N,0,s2))
   U <- runif(N)
   X <- matrix(NA, nrow=N+1, ncol=2)
   X[1,] <- x0
   for (i in 1:N) {
      y <- X[i,]+eps[i,]
      p <- dbinorm(y,mu1,mu2,s1,s2,rho)/
             dbinorm(X[i,],mu1,mu2,s1,s2,rho)
      if (U[i] < p) X[i+1,] <- y
      else X[i+1,] <- X[i,]
   }
   X[-1,]
}
```

[5 marks]

b) The conditional density of $X \mid Y = y$ is proportional to the joint density. Since y is considered fixed and known we only need to consider the parts that depend on x. Hence we get

$$
\begin{aligned}
f_{X|Y}(x \mid y) &\propto \exp\left\{-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_1)^2}{\sigma_1^2} - \frac{2\rho(x-\mu_1)(y-\mu_2)}{\sigma_1\sigma_2}\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2(1-\rho^2)}\left[\frac{x^2-2\mu_1 x}{\sigma_1^2} - \frac{2\rho x(y-\mu_2)}{\sigma_1\sigma_2}\right]\right\} \\
&= \exp\left\{-\frac{1}{2\sigma_1^2(1-\rho^2)}\left[x^2 - 2\left(\mu_1 + \frac{\rho\sigma_1}{\sigma_2}(y-\mu_2)\right)x\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2\sigma_1^2(1-\rho^2)}\left[x - \left(\mu_1 + \frac{\rho\sigma_1}{\sigma_2}(y-\mu_2)\right)\right]^2\right\}
\end{aligned}
$$

Since this is exactly of the form of the univariate normal except for a constant that does not depend on x we see that

$$X \mid Y = y \sim \text{normal}\left(\mu_1 + \frac{\rho\sigma_1}{\sigma_2}(y-\mu_2), (1-\rho^2)\sigma_1^2\right)$$

By symmetry in the bivariate density we can also deduce that

$$Y \mid X = x \sim \text{normal}\left(\mu_2 + \frac{\rho\sigma_2}{\sigma_1}(x-\mu_1), (1-\rho^2)\sigma_2^2\right)$$

[8 marks]

c) Using the results from (b) we can implement the Gibbs sampler.

```
rbinorm.gibbs <- function(N, mu1, mu2, s1, s2, rho) {
#
# Function to run a Gibbs sampling algorithm to
# generate bivariate normals.
#
# Arguments:
#  N - length of the chain
#  mu1, mu2 - marginal means
#  s1, s2 - marginal standard deviations
#  rho - correlation
#
   x0 <- rnorm(2, c(mu1, mu2), c(s1,s2))
   # calculate the conditional standard deviations
   s1c <- s1*sqrt(1-rho^2)
   s2c <- s2*sqrt(1-rho^2)
   X <- matrix(NA, nrow=N+1, ncol=2)
   X[1,] <- x0
   for (i in 1:N) {
      X[i+1,1] <- rnorm(1, mu1+rho*s1/s2*(X[i,2]-mu2), s1c)
      X[i+1,2] <- rnorm(1, mu2+rho*s2/s1*(X[i+1,1]-mu1), s2c)
   }
   X[-1,]
}
```

[5 marks]

d) To run these two methods and examine the output we run the following code.

```
set.seed(19032019)
X.q3.mh <- rbinorm.mh(10000, -1, 1, 1, 2, -0.5)
set.seed(19032019)
X.q3.gibbs <- rbinorm.gibbs(10000, -1, 1, 1, 2, -0.5)
colMeans(X.q3.mh)
apply(X.q3.mh,2,sd)
cor(X.q3.mh[,1], X.q3.mh[,2])
colMeans(X.q3.gibbs)
apply(X.q3.gibbs,2,sd)
cor(X.q3.gibbs[,1], X.q3.gibbs[,2])
```

This gives us the following results

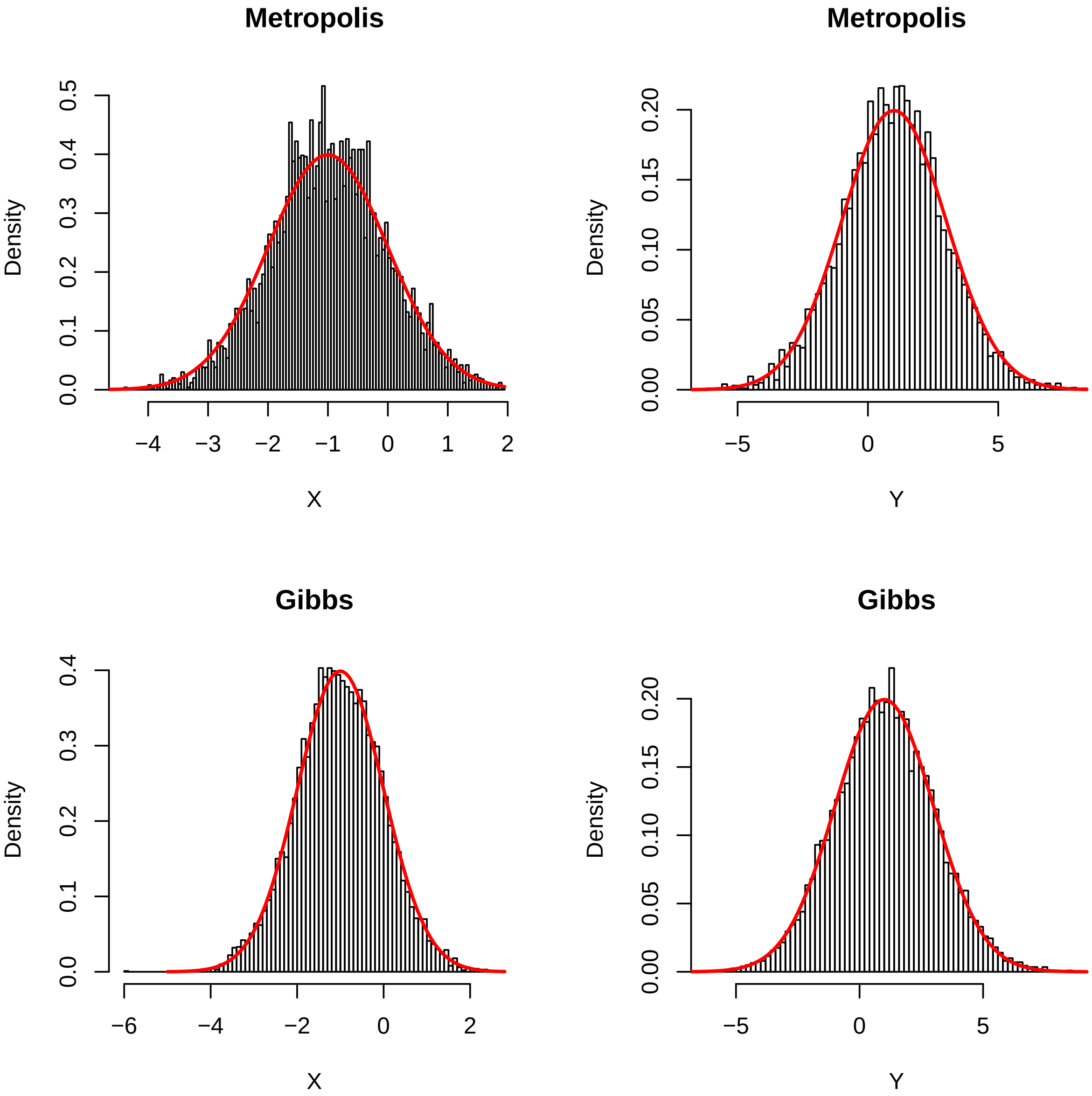
	μ_1	μ_2	σ_1	σ_2	ρ
True Values	-1	1	1	2	-0.5
Metropolis–Hastings	-0.9848	0.9880	0.9664	1.9509	-0.4801
Gibbs Sampler	-1.0177	1.0348	0.9934	2.0141	-0.4916

[3 marks]

The four histograms can be plotted by the following code

```
par(mfrow=c(2,2))
hist(X.q3.mh[,1], breaks=100, prob=TRUE, main="Metropolis", xlab="X")
curve(dnorm(x,-1,1), c(-5,3), col="red", lwd=2, add=TRUE)
hist(X.q3.mh[,2], breaks=100, prob=TRUE, main="Metropolis", xlab="Y")
curve(dnorm(x,1,2), c(-7,9), col="red", lwd=2, add=TRUE)
hist(X.q3.gibbs[,1], breaks=100, prob=TRUE, main="Gibbs", xlab="X")
curve(dnorm(x,-1,1), c(-5,3), col="red", lwd=2, add=TRUE)
hist(X.q3.gibbs[,2], breaks=100, prob=TRUE, main="Gibbs", xlab="Y")
curve(dnorm(x,1,2), c(-7,9), col="red", lwd=2, add=TRUE)
```

The plots are on the next page.



[2 marks]

It appears from these plots that the Gibbs sampler results in samples that are closer to the marginal densities than does the Metropolis-Hastings algorithm. The deviations in the Metropolis-Hastings algorithm typically take the form of large spikes at certain values and these are likely caused by rejections of moves which results in higher autocorrelation in the chains. We can verify that this is the case using the `acf` function which goes to zero much quicker for the marginal Gibbs sampler chains than for the marginal Metropolis-Hastings chain. I have included code in the `R` file for this but that was not required for the question.

[2 marks]

Q. 4 **a)** The likelihood for given values of n and θ is

$$\mathrm{P}(X = x \mid n, \theta) \;=\; \binom{n}{x}\theta^x(1-\theta)^{n-x}$$

and the prior distribution for the parameter vector is

$$\pi(n,\theta) \;=\; \pi_n(n)\pi_\theta(\theta) \;=\; \frac{\mu^n\,\mathrm{e}^{-\mu}}{n!} \times \frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

Hence the posterior density for (n,θ) is

$$\begin{aligned}
\pi(n,\theta \mid x) &\propto \mathrm{P}(X = x \mid n,\theta)\pi(n,\theta) \\
&= \binom{n}{x}\theta^x(1-\theta)^{n-x} \times \frac{\mu^n\,\mathrm{e}^{-\mu}}{n!} \times \frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\
&\propto \frac{1}{(n-x)!}\theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1}\mu^n
\end{aligned}$$

The support of this distribution is then $n = x, x+1, \ldots$ and $\theta \in (0,1)$. **[4 marks]**

b) The conditional posterior for n given the value of θ is then

$$\begin{aligned}
\pi(n \mid \theta, x) &\propto \pi(n,\theta \mid x) \\
&\propto \frac{\mu^n}{(n-x)!}\theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1} \\
&\propto \frac{\mu^n}{(n-x)!}(1-\theta)^{n-x+\beta-1} \\
&\propto \frac{\mu^{n-x}}{(n-x)!}(1-\theta)^{n-x} \\
&= \frac{\big(\mu(1-\theta)\big)^(n-x)}{(n-x)!} \qquad n = x, x+1, \ldots
\end{aligned}$$

[3 marks]

Now suppose that the random variable $Y = n - x$ then the conditional distribution of Y for a given x and θ is

$$\begin{aligned}
\mathrm{P}(Y = y \mid \theta, x) &= \mathrm{P}(n = x + y \mid \theta, x) \\
&= \pi(x + y \mid x, \theta) \\
&\propto \frac{\big(\mu(1-\theta)\big)^y}{y!} \qquad y = 0, 1, \ldots \\
&\propto \frac{\mathrm{e}^{\mu(1-\theta)}\,\big(\mu(1-\theta)\big)^y}{y!} \qquad y = 0, 1, \ldots
\end{aligned}$$

Hence we have that the conditional posterior distribution of n is such that

$$n \mid \theta, x \;\overset{d}{=}\; x + Y \qquad \text{where } Y \sim \text{Poisson}\big(\mu(1-\theta)\big)$$

[3 marks]

c) Similarly we can get the conditional posterior for θ given n and x.

$$\begin{aligned} \pi(\theta \mid n, x) &\propto \pi(n, \theta \mid x) \\ &\propto \frac{\mu^n}{(n-x)!}\theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1} \\ &\propto \theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1} \qquad \theta \in (0,1) \end{aligned}$$

and we recognize this as the kernel of a beta density with parameters $x+\alpha$ and $n-x+\beta$ and so we have

$$\theta \mid n, x \;\sim\; \text{Beta}(x+\alpha, n-x+\beta)$$

[3 marks]

d) Given the results of the previous two parts we have the full conditional distributions required to implement the Gibbs Sampler. Here is the code to do that

```
N <- 10000
N0 <- 2000
mu <- 16
alpha <- 2
beta <- 4
x <- 2

Gibbs.Q4d <- matrix(NA, nrow=N+N0, ncol=2)
set.seed(21032019)
n <- rpois(1,mu)
th <- rbeta(1, alpha, beta)
for (i in 1:(N+N0)) {
  n <- x+rpois(1,mu*(1-th))
  th <- rbeta(1, x+alpha, n-x+beta)
  Gibbs.Q4d[i,] <- c(n, th)
}
Gibbs.Q4d <- Gibbs.Q4d[-(1:N0),]
```

[3 marks]

The Monte Carlo estimates of the marginal posterior means and variances for my simulation are

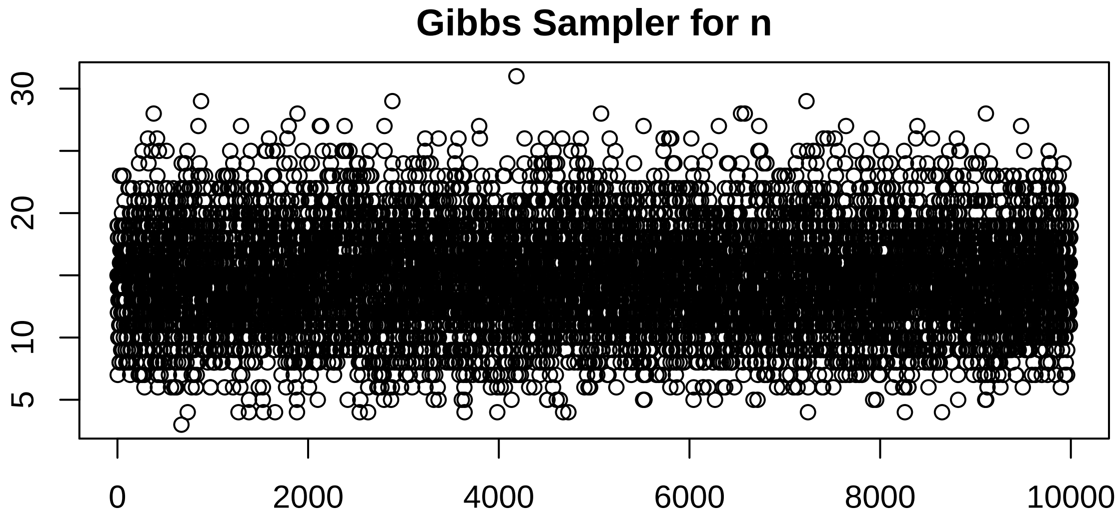
$$\begin{aligned} \widehat{\text{E}(n \mid x)} &= 14.7635, \qquad & \widehat{\text{Var}(n \mid x)} &= 14.9469 \\ \widehat{\text{E}(\theta \mid x)} &= 0.2000, \qquad & \widehat{\text{Var}(\theta \mid x)} &= 0.0092 \end{aligned}$$

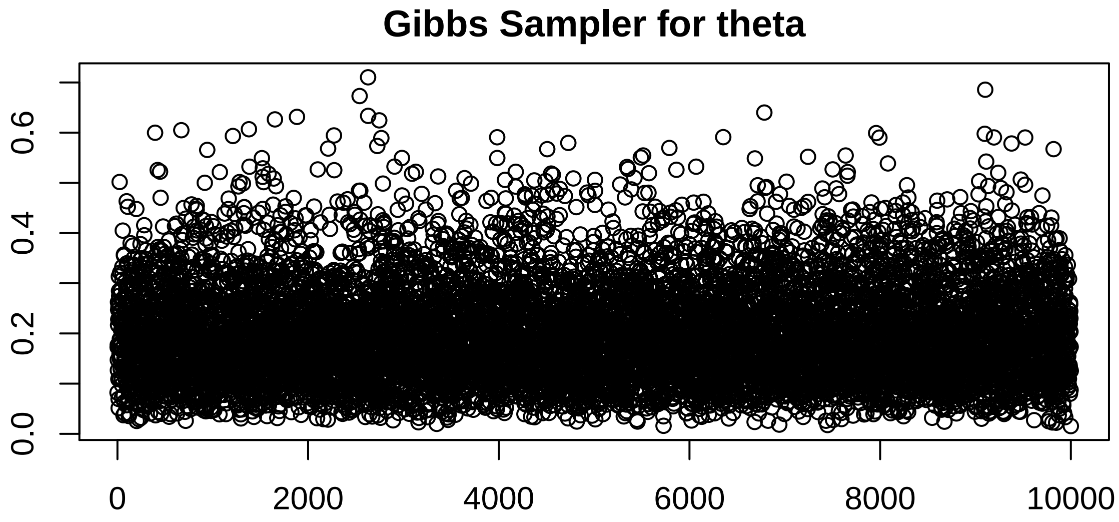
and the Monte Carlo estimate of the posterior correlation is the sample correlation between the n chain and the θ chain

$$\widehat{\mathrm{Cor}(n, \theta \mid x)} \;=\; -\,0.3921$$

Note that they are negatively correlated which is what we would expect since we always observe $x = 2$ so a larger value of n would necessarily mean that the success probability θ is reduced. **[3 marks]**

The plots of the chain iterations are shown here





[2 marks]

e) For this we note that we already have two of the full conditionals as they are no different than in part (d). The third full conditional for X given n and θ is simply the binomial we specified as the model for the observed data earlier. Hence we simply need to add one extra bit to the code to generate from this binomial distribution. Thus the code is as follows

```
Gibbs.Q4e <- matrix(NA, nrow=N+N0, ncol=3)
set.seed(21032019)
n <- rpois(1,mu)
th <- rbeta(1, alpha, beta)
for (i in 1:(N+N0)) {
  x <- rbinom(1, n, th)
  n <- x+rpois(1,mu*(1-th))
  th <- rbeta(1, x+alpha, n-x+beta)
  Gibbs.Q4e[i,] <- c(x, n, th)
}
Gibbs.Q4e <- Gibbs.Q4e[-(1:N0),]
```

[2 marks]

Based on my chain I estimate the marginal mean and variance of X to be

$$\widehat{\mathrm{E}(X)} \;=\; 5.4783 \qquad \widehat{\mathrm{Var}(X)} \;=\; 14.2648$$

[1 mark]

As for n and θ now we see

$$\begin{aligned} \widehat{\mathrm{E}(n)} &= 16.0354, & \widehat{\mathrm{Var}(n)} &= 15.8599 \\ \widehat{\mathrm{E}(\theta)} &= 0.3390, & \widehat{\mathrm{Var}(\theta)} &= 0.0334 \end{aligned}$$

and $\widehat{\mathrm{Cor}(n,\theta)} \;=\; 0.0236$. We note that these are (other than Monte Carlo variability) the moments specified by the prior distributions as we would expect if we do not condition on an observed x. **[2 marks]**