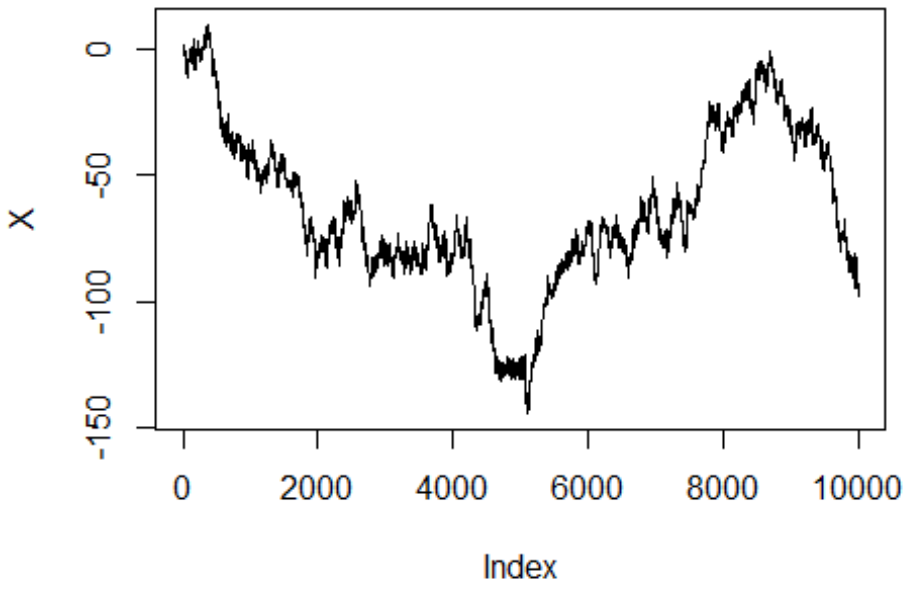
HW-4

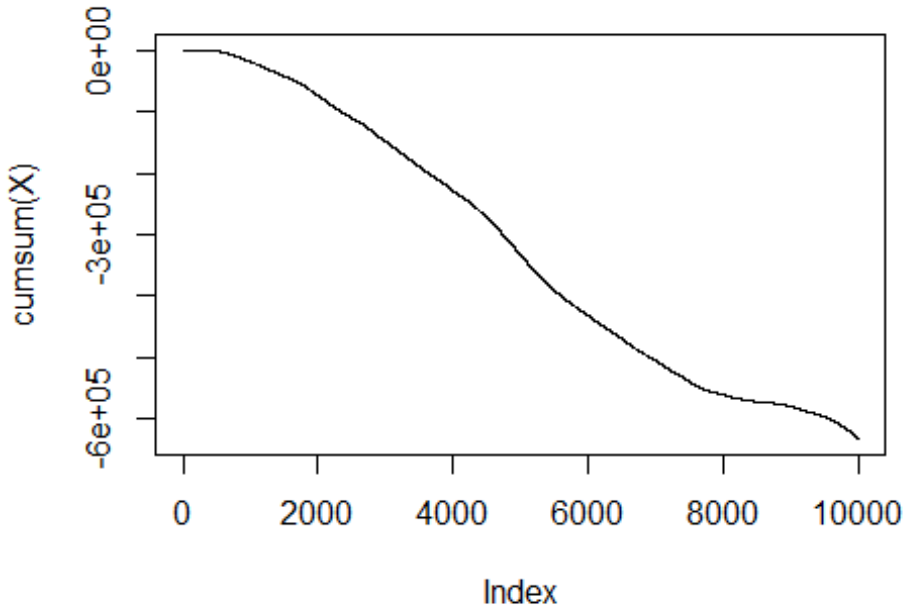
```
knitr::opts_chunk$set(echo = TRUE)
```

6.2

```
### for t = 10^4
X <- vector()
t <- 10^4
X[1] = 0
for(i in 2:t){
  X[i] <- X[i-1] + rnorm(1,0,1)
}

plot(X,type="l")
```

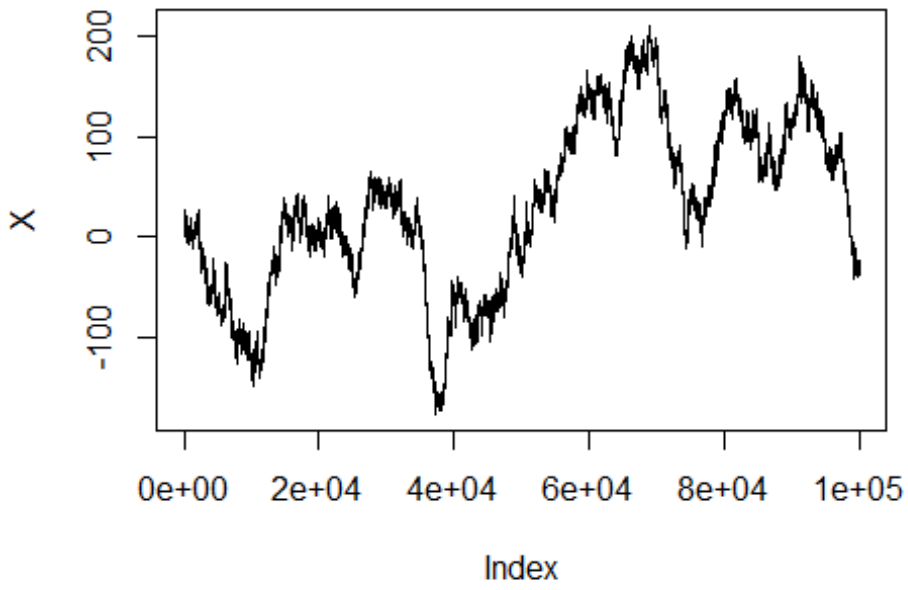



```
#Helps to see whether randomwalk is converging
plot(cumsum(X),type="l")
```

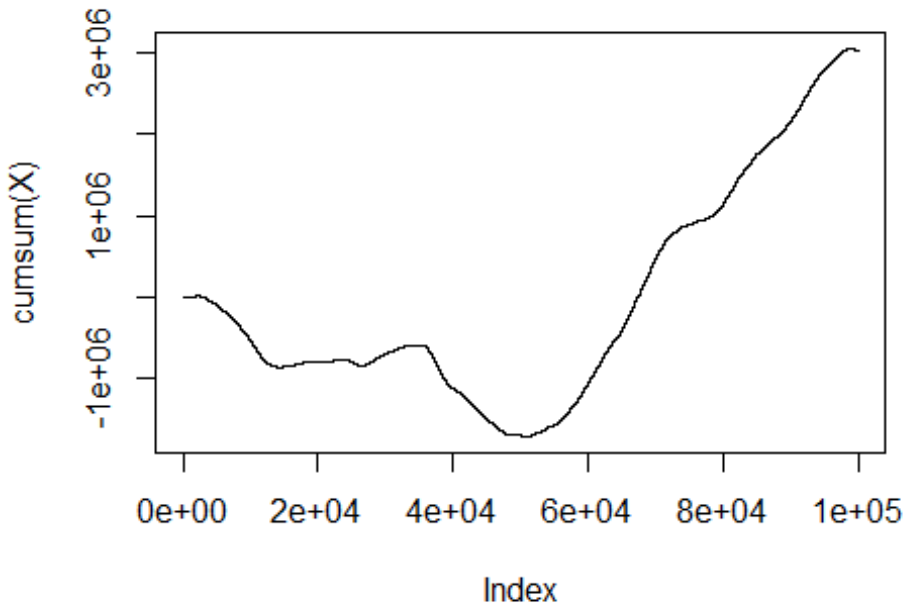



```
# A stationary probability distribution exists by construction
# for those chains; that is, there exists a probability
# distribution f such that if X(t) ~ f , then X(t + 1) ~ f

#Analyzing the last values of X we can determine whether stationary
probability
#exists for random walk

tail(X)

## [1] -98.00294 -97.61150 -98.02811 -95.28978 -93.98828 -93.37919

#By looking at the last values of X we can determine that
#random walk is not a stationary distribution

# In a limiting distribution X(t) is f for almost any initial value X(0)
# But in our current randomwalk simulation
# we reach different X(t) value with the same initial value X(0)


#By looking at the last values of X(t) we can determine that
#random walk is not a limiting distribution

### for t = 10^6
X <- vector()
t <- 10^5
X[1] = 0
```

```r
for(i in 2:t){
  X[i] <- X[i-1] + rnorm(1,0,1)
}

plot(X,type="l")
```




```r
#Helps to see whether randomwalk is converging
plot(cumsum(X),type="l")
```

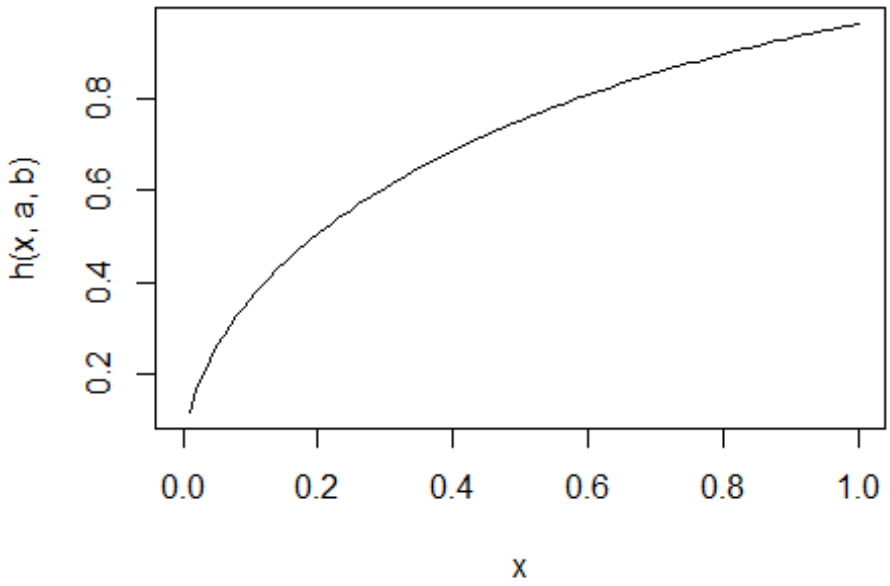



```
# A stationary probability distribution exists by construction
# for those chains; that is, there exists a probability
# distribution f such that if X(t) ~ f , then X(t + 1) ~ f

#Analyzing the last values of X we can determine whether stationary
probability
#exists for random walk

tail(X)

## [1] -24.21118 -23.99085 -23.40403 -24.58683 -25.99946 -24.11550

#By looking at the last values of X we can determine that
#random walk is not a stationary distribution

# In a limiting distribution X(t) is f for almost any initial value X(0)
# But in our current randomwalk simulation
# we reach different X(t) value with the same initial value X(0)


#By looking at the last values of X(t) we can determine that
#random walk is not a limiting distribution
```

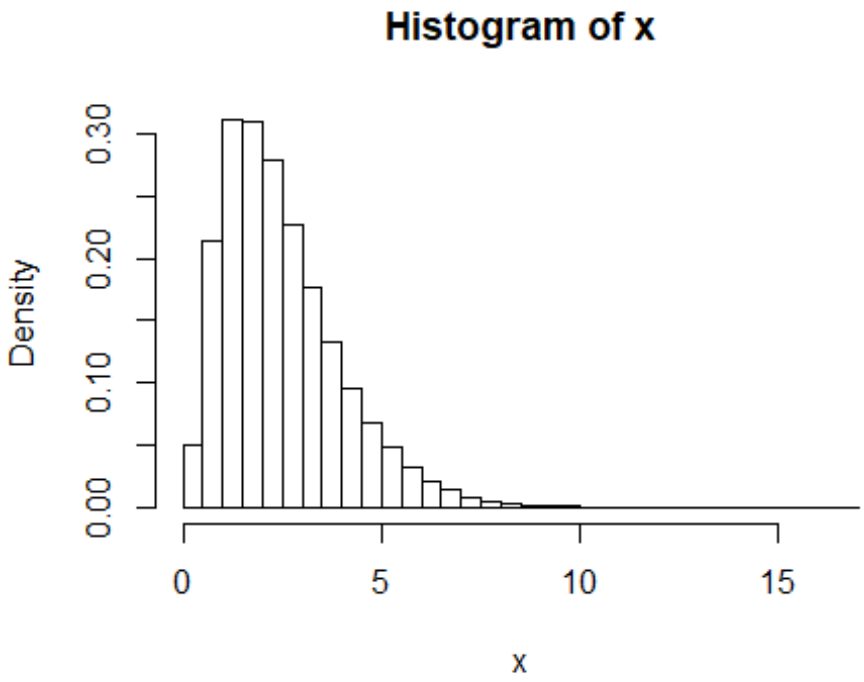
6.4 a

When b = floor(alpha)/alpha

```
h <- function(x,a,b)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(a)/a)
}

x <- seq(0,1,by=0.01)
a = 2.5
b = 1
plot(x,h(x,a,b),type = "l")
```




```
f <- function(x)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(a)/a)
}

sim <- optimize(h(x,2.5,1), interval = c(0,1) , maximum = TRUE)
sim.obj <- sim$objective
M <- 0.96
#M <- 1/sim.obj
N <- 90000
u <- runif(N,min = 0, max = M)
y <- rgamma(N,a,b)
x <- y[u < h(y,a,b)]
hist(x,breaks = 40, prob = T)
```

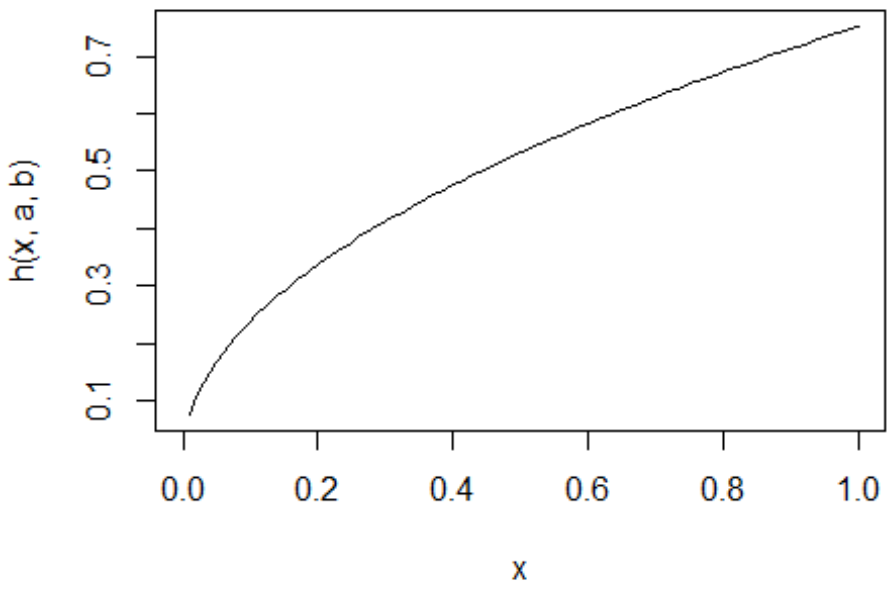



```
#Acceptance rate for with b = floor(a)/a
length(x)/N
```

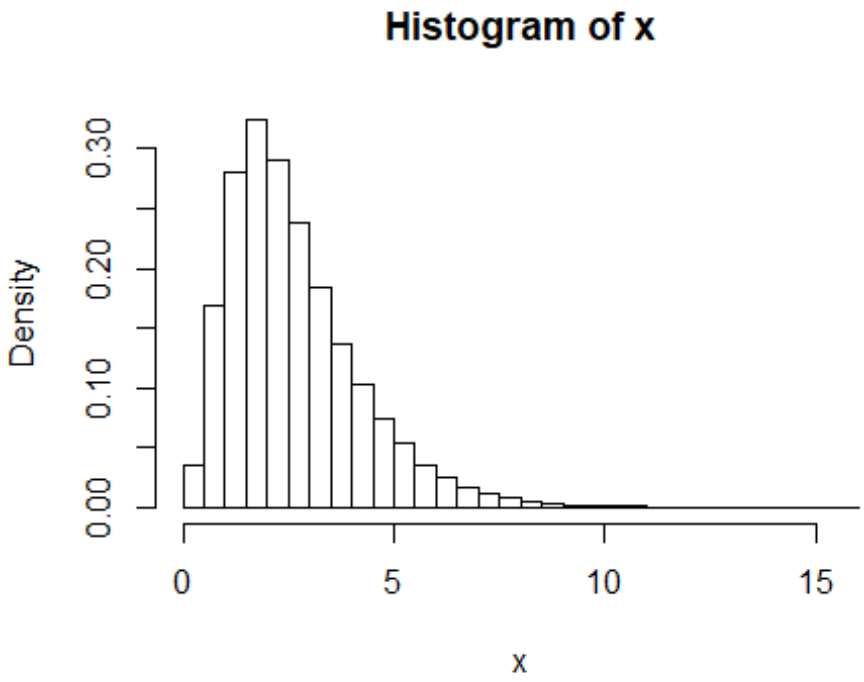
```
## [1] 0.9683333
```

Without b = floor(alpha)/alpha

```
h <- function(x,a,b)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(b))
}

x <- seq(0,1,by=0.01)
a = 2.5
b = 1
plot(x,h(x,a,b),type = "l")
```




```
f <- function(x)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(b))
}

sim <- optimize(h(x,2.5,1), interval = c(0,1) , maximum = TRUE)
sim.obj <- sim$objective
M <- 0.96
#M <- 1/sim.obj
N <- 90000
u <- runif(N,min = 0, max = M)
y <- rgamma(N,a,b)
x <- y[u < h(y,a,b)]
hist(x,breaks = 40, prob = T)
```

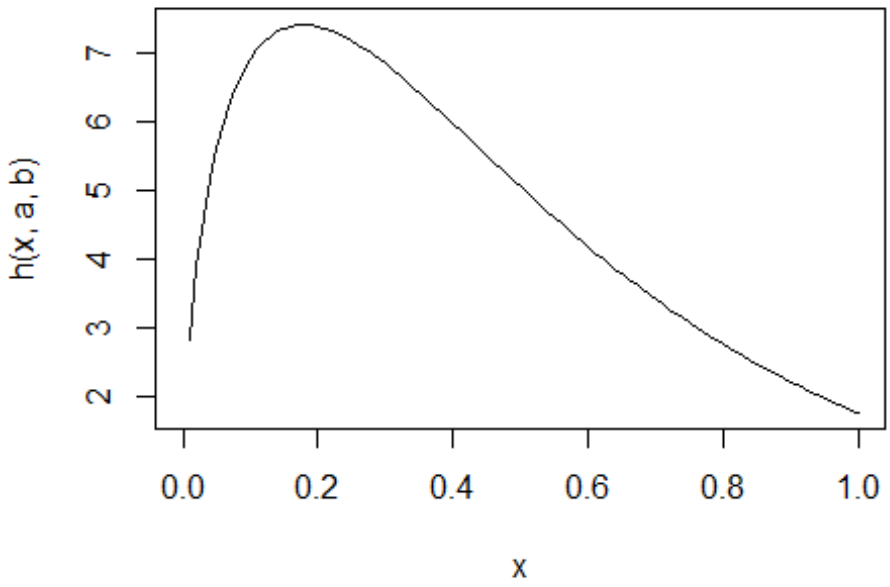



```
#Acceptance rate for b != floor(a)/a
length(x)/N
```

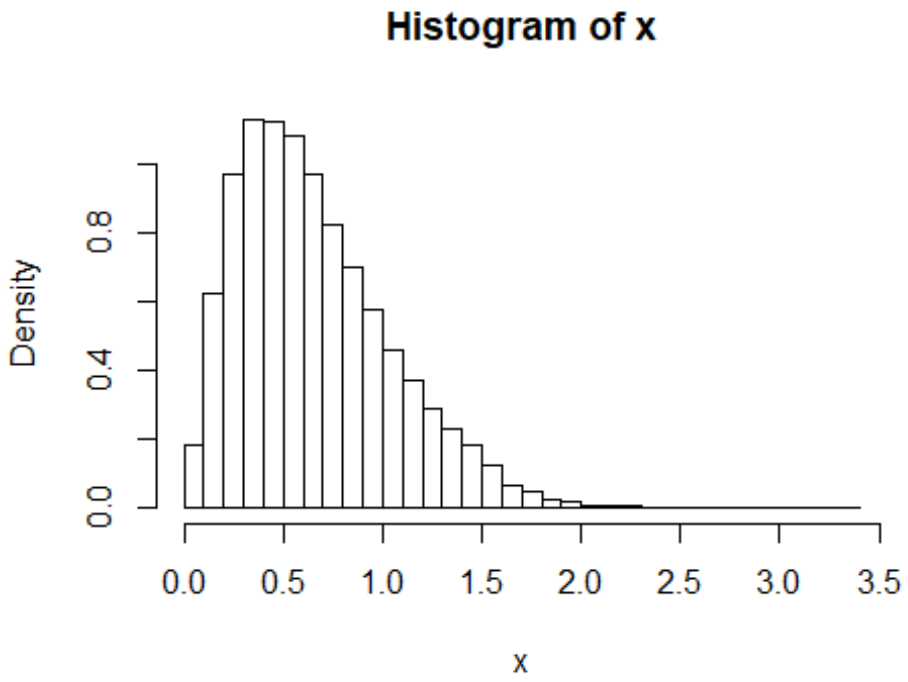
```
## [1] 0.9236
```

When b = floor(alpha)/alpha and Beta not equal to 1

```
h <- function(x,a,b)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(a)/a)
}

x <- seq(0,1,by=0.01)
a = 2.5
b = 3.6
plot(x,h(x,a,b),type = "l")
```




```r
f <- function(x)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(a)/a)
}

sim <- optimize(h(x,2.5,3.6), interval = c(0,1) , maximum = TRUE)
sim.obj <- sim$objective
M <- 0.56
#M <- 1/sim.obj
N <- 90000
u <- runif(N,min = 0, max = M)
y <- rgamma(N,a,b)
x <- y[u < h(y,a,b)]
hist(x,breaks = 40, prob = T)
```

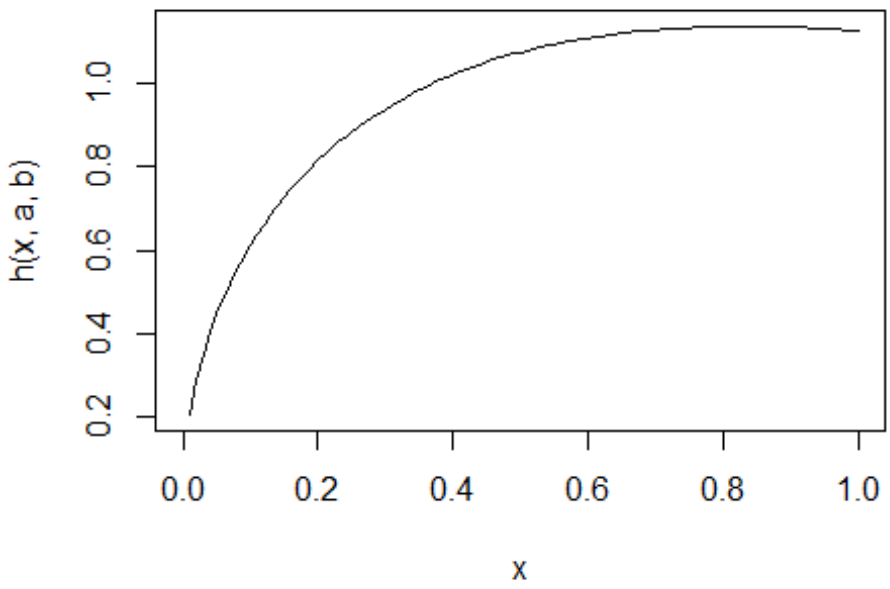



```
#Acceptance rate for with b = floor(a)/a when Beta not equal to 1
length(x)/N
```

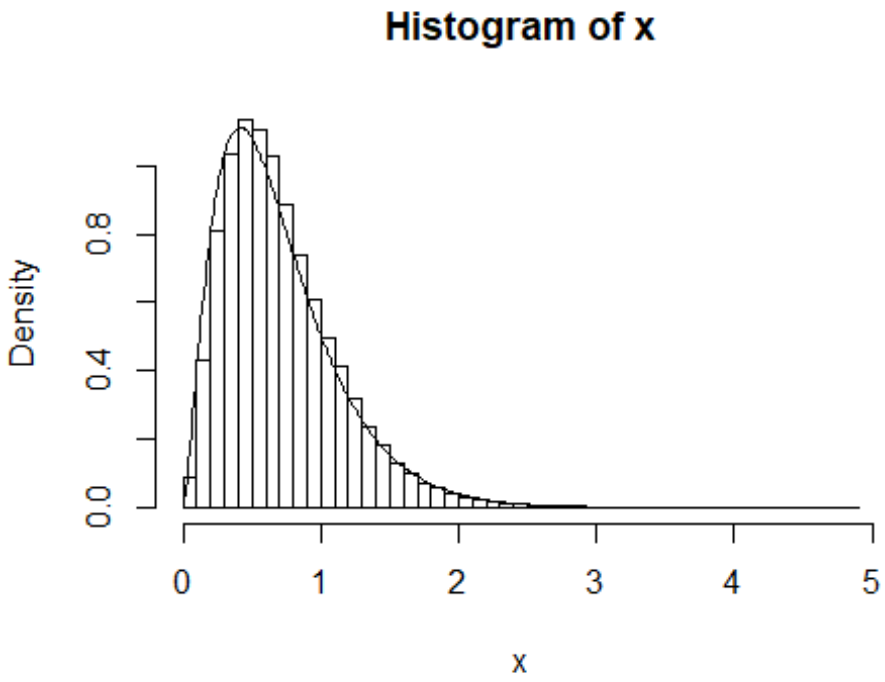
```
## [1] 0.9730667
```

Without b = floor(alpha)/alpha

```
h <- function(x,a,b)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(b))
}

x <- seq(0,1,by=0.01)
a = 2.5
b = 3.6
plot(x,h(x,a,b),type = "l")
```




```r
f <- function(x)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(b))
}

sim <- optimize(h(x,2.5,3.6), interval = c(0,1) , maximum = TRUE)
sim.obj <- sim$objective
M <- 1.13
#M <- 1/sim.obj
N <- 90000
u <- runif(N,min = 0, max = M)
y <- rgamma(N,a,b)
x <- y[u < h(y,a,b)]
hist(x,breaks = 40, prob = T)
c <- seq(0,3,0.01)
lines(c,dgamma(c,a,b),type = "l")
```



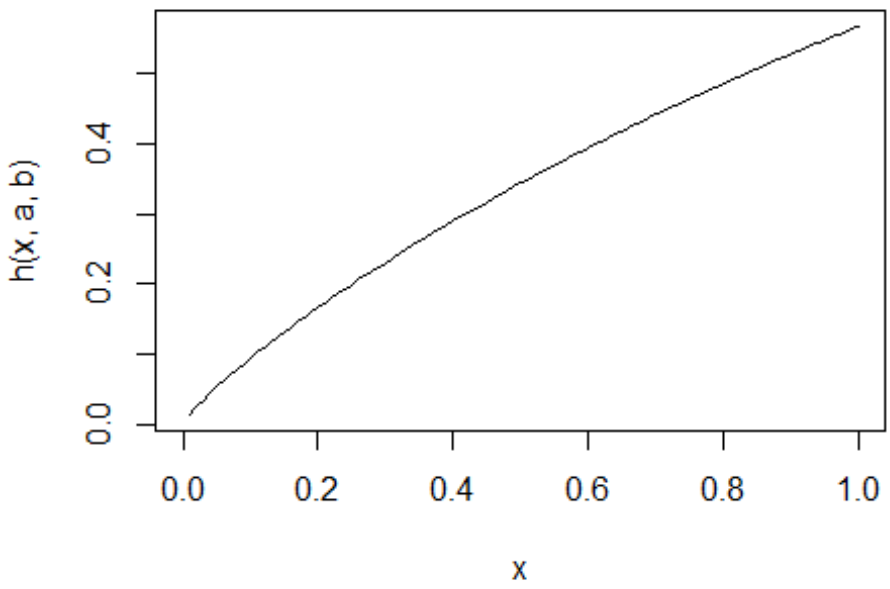

```
#Acceptance rate for b != floor(a)/a
length(x)/N
```

```
## [1] 0.9065222
```

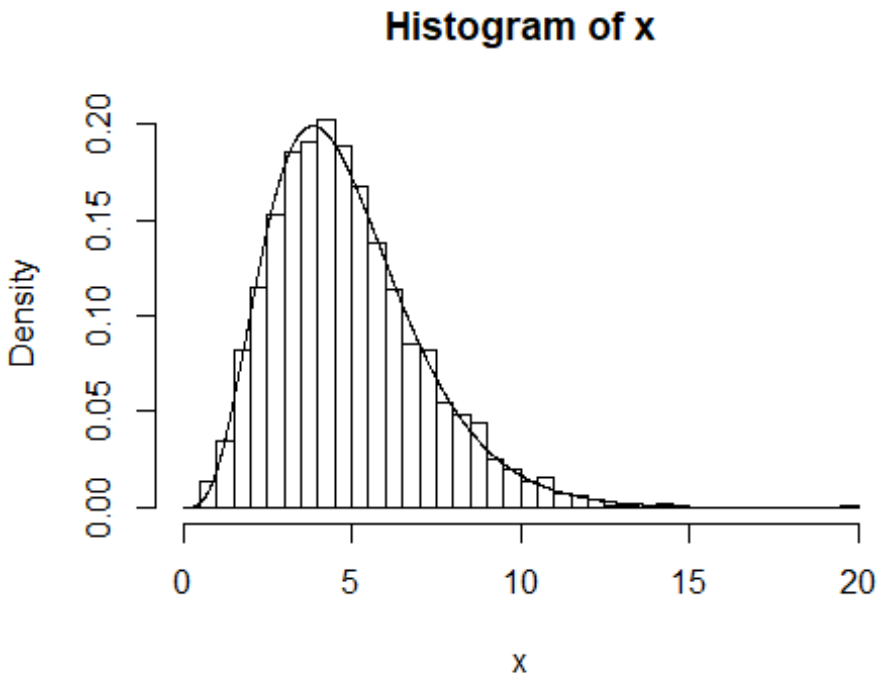
The Acceptance rate is at 96.7% when b = floor(a)/a is used and 92.2% when not used. The acceptance rate is dropped to 0.11% when beta is not equal to 1 for b = floor(a)/a and 90.8% for b equal to floor(beta)

6.4 b

```
h <- function(x,a,b)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(a)/a)
}


x <- seq(0,1,by=0.01)
a = 4.85
b = 1
plot(x,h(x,a,b),type = "l")
```




```r
f <- function(x)
{
  dgamma(x,a,b) / dgamma(x,floor(a),floor(a)/a)
}

sim <- optimize(h(x,4.85,1), interval = c(0,1) , maximum = TRUE)
sim.obj <- sim$objective
M <- 0.56
N <- 5000
u <- runif(N,min = 0, max = M)
y <- rgamma(N,4.85,1)
#x <- y[u < h(y,4.85,1)]
x <- y[u < rgamma(y,4.85,1)]
hist(x,breaks = 40, prob = T)
c <- seq(0,15,by = 0.01)
lines(c,dgamma(c,4.85,1),type = "l")
```




```
length(x)/N
```

```
## [1] 1
```

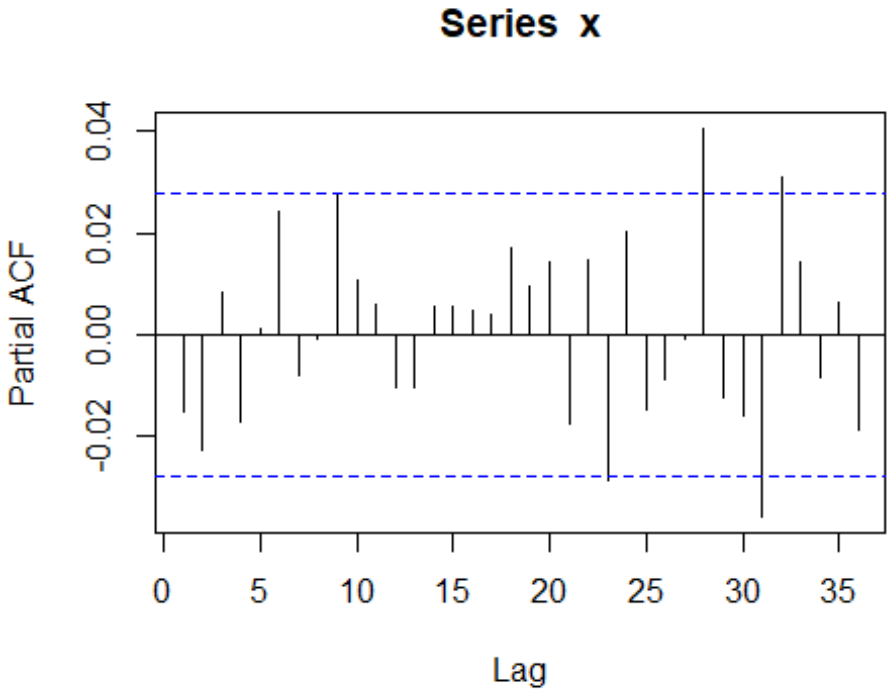
```
mean(x)
```

```
## [1] 4.887669
```

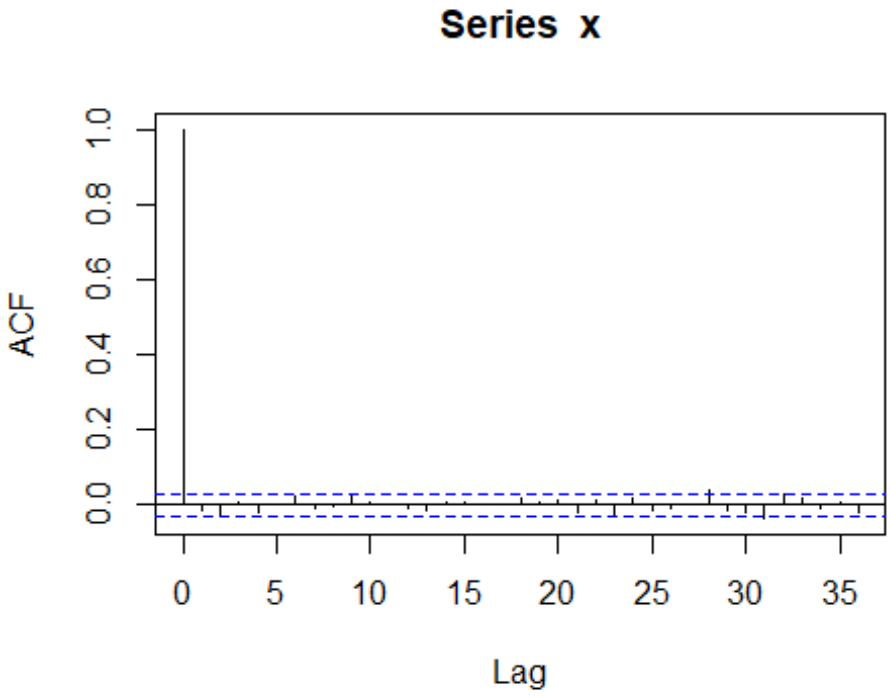
```
var(x)
```

```
## [1] 5.031518
```

```
pacf(x)
```

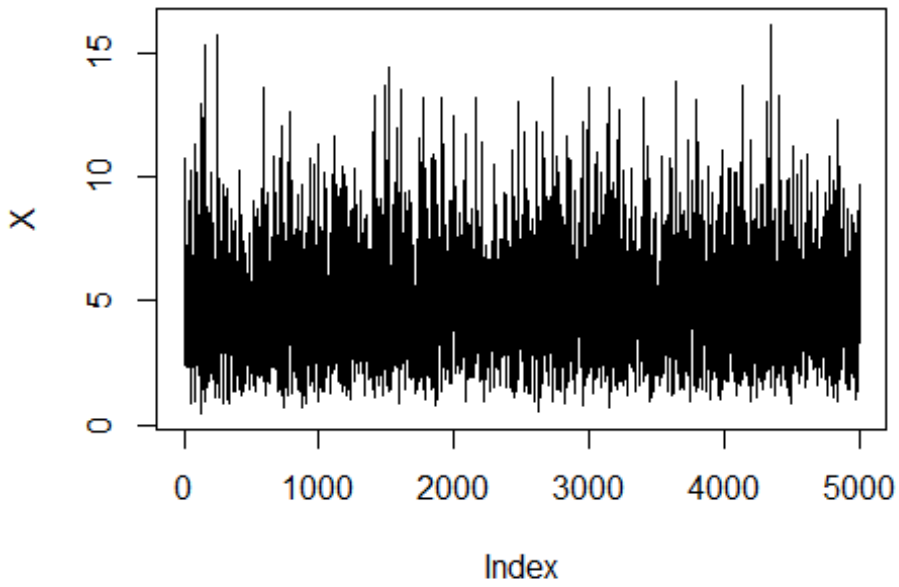

Series x
Partial ACF
0.04
0.02
0.00
-0.02
0
5
10
15
20
25
30
35
Lag

```
acf(x)
```

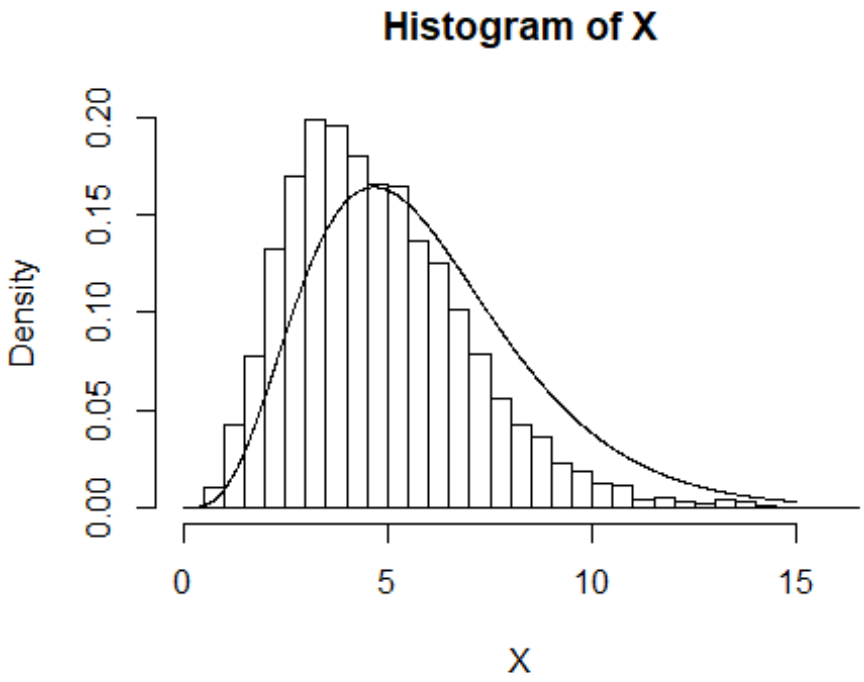

Series x
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
Lag

6.4 c

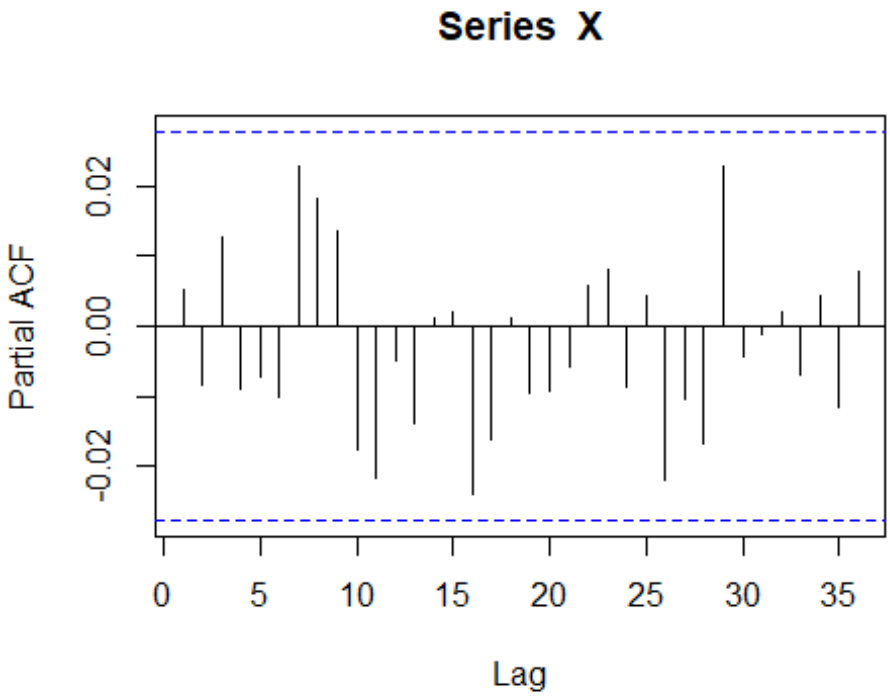
```r
N <- 5000
X<-rep(0,N)
X[1]<-rgamma(1,4.85,1)
for (i in 2:5000){
  Y<-rgamma(1,4,4/4.85)
  rho=(dgamma(X[i-
1],4,4/4.85)*dgamma(Y,4.85,1))/(dgamma(Y,4,4/4.85)*dgamma(X[i-1],4.85,1))
  X[i]=X[i-1]+(Y-X[i-1])*(runif(1)<rho)
}

length(unique(X))/N

## [1] 0.9358

mean(X)

## [1] 4.799567

var(X)

## [1] 4.795891

plot(X,type = "l")
```



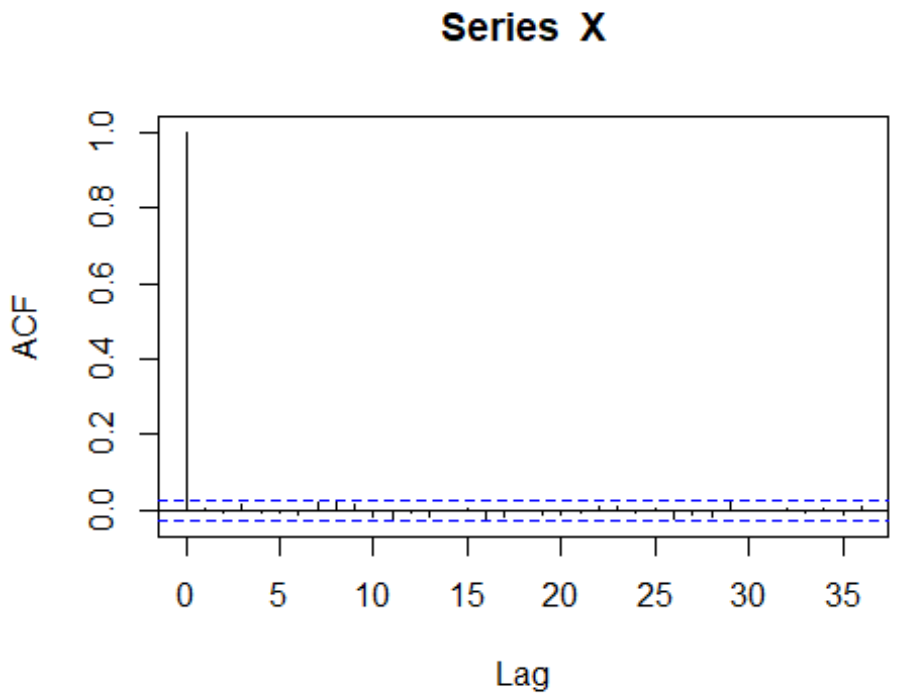

```r
x<-seq(0,15, by=.01)
y<-dgamma(x,4.85,4/4.85)
hist(X,breaks = 40, prob = T)
lines(x,y, type = "l")
```


Histogram of X
Density
0.00
0.05
0.10
0.15
0.20
0
5
10
15
X

```
pacf(X)
```


Series X
Partial ACF
-0.02
0.00
0.02
0
5
10
15
20
25
30
35
Lag

```
acf(X)
```



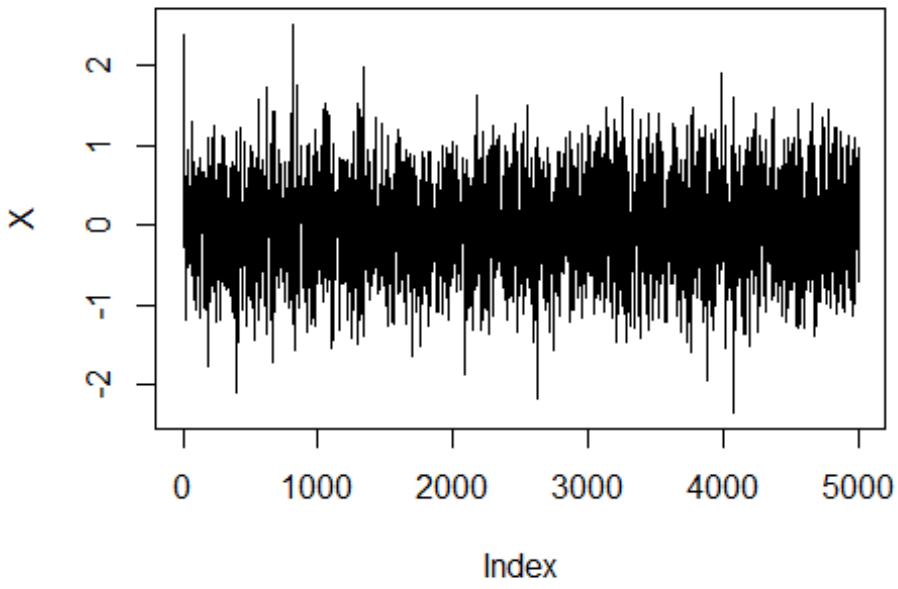
6.4 d

i) The acceptance rate for accept reject algorithm is 99% and there is no acceptance rate for Metropolis-Hastings algorithm since it is based on morkov-chain (random walk) that is if a value is rejected because of the condition previous value is copied to present value. But effeciency rate is around 93.38%

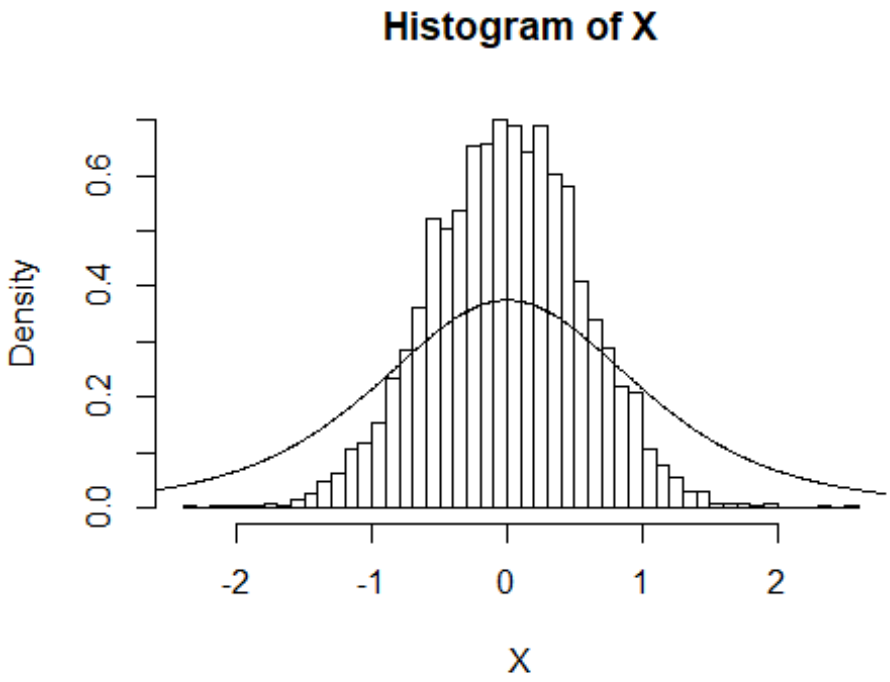
ii)Mean and Variance of accept-reject are as follows - 4.79 and 4.76 Mean and variance of Mtropolis - Hastings algorithm are - 4.9 and 4.9 With variance from metropolis we can deduct that values are vary with in a close range. Looking at a pacf and acf of metropolis we can observe the values generated from metropolis-hastings algorithm is correlated.

6.10 a

```
N= 5000
X=rep(rnorm(1,0,1), N)
n = 4
for(i in 2:N){
  Y=rnorm(1, 0, 1)
  rho=(dt(Y,df=n)/dt(X[i-1], df= n))*(dnorm(Y, 0,1)/dnorm(X[i-1],0,1))
  X[i]=X[i-1] + (Y-X[i-1])*(runif(1)<rho)
}

plot(X,type = "l")
```




```
hist(X,breaks = 40, probability = TRUE)
x<-seq(-3,3, by=.01)
y<-dt(x,df=4)
lines(x,y)
```



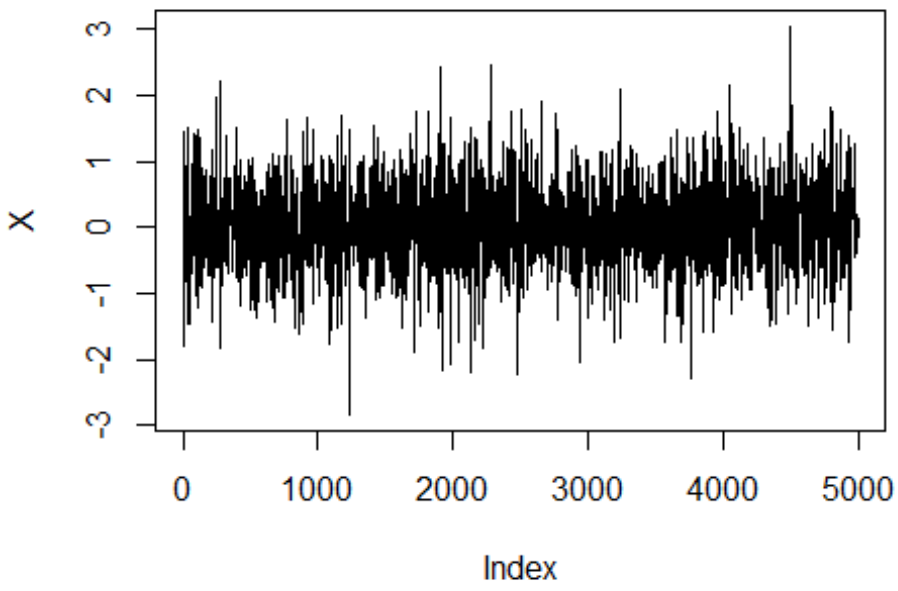

```
mean(X)
```

```
## [1] -0.004149037
```

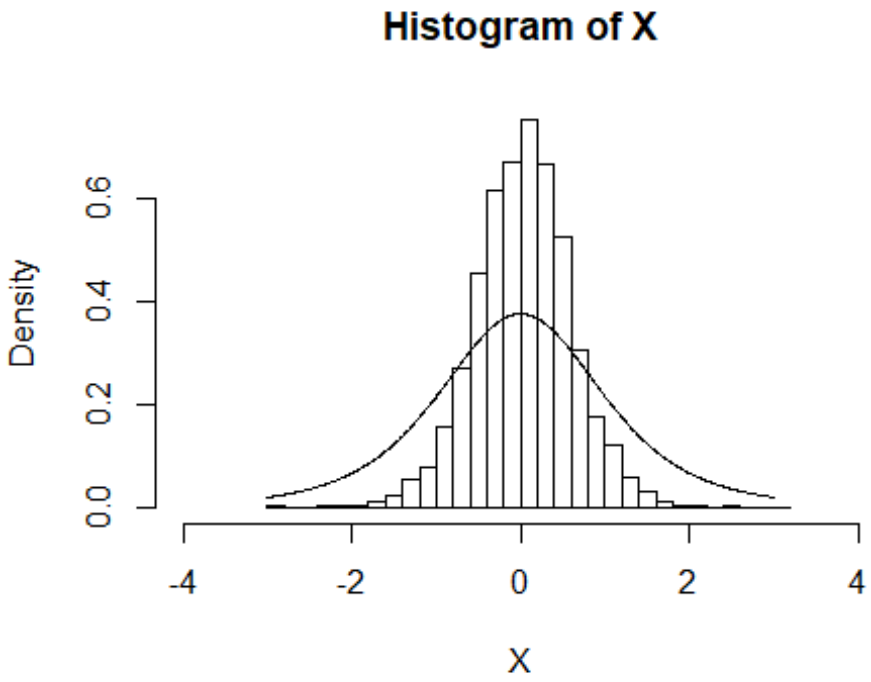
```
#Mean of t-distibution is 0, oTherwise undefined. When calculating mean of
all X, or accepted values
#with candidate density N(0,1) for a target t-distribution with df = 4 mean
is 0.007
```

6.10 b

```
N= 5000
X=rep(rt(1,2), N)
n = 4
for(i in 2:N){
  Y=rt(1,2)
  rho=(dt(Y,df=n)/dt(X[i-1], df= n))*(dt(Y,2)/dt(X[i-1],2))
  X[i]=X[i-1] + (Y-X[i-1])*(runif(1)<rho)
}

plot(X,type = "l")
```




```
hist(X,breaks = 40, probability = TRUE, xlim = c(-4,4))
x<-seq(-3,3, by=.01)
y<-dt(x,df=4)
lines(x,y)
```




```r
mean(X)
```

```
## [1] 0.03382485
```

```r
#Mean of t-distibution is 0, otherwise undefined. When calculating mean of
all X, or accepted values
#with candidate denstity t and df = 2 for target t distribution and df = 4
mean is 0.117
```