Homework 7

Michael Pena

2024-05-02

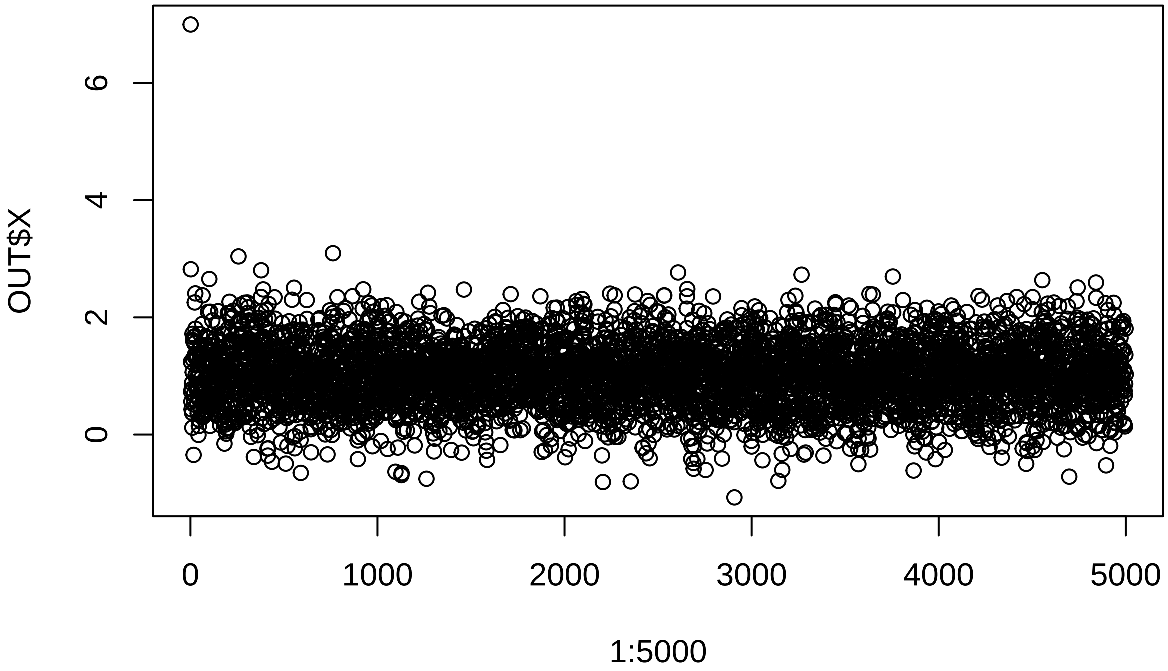
Problem 1

part (a)

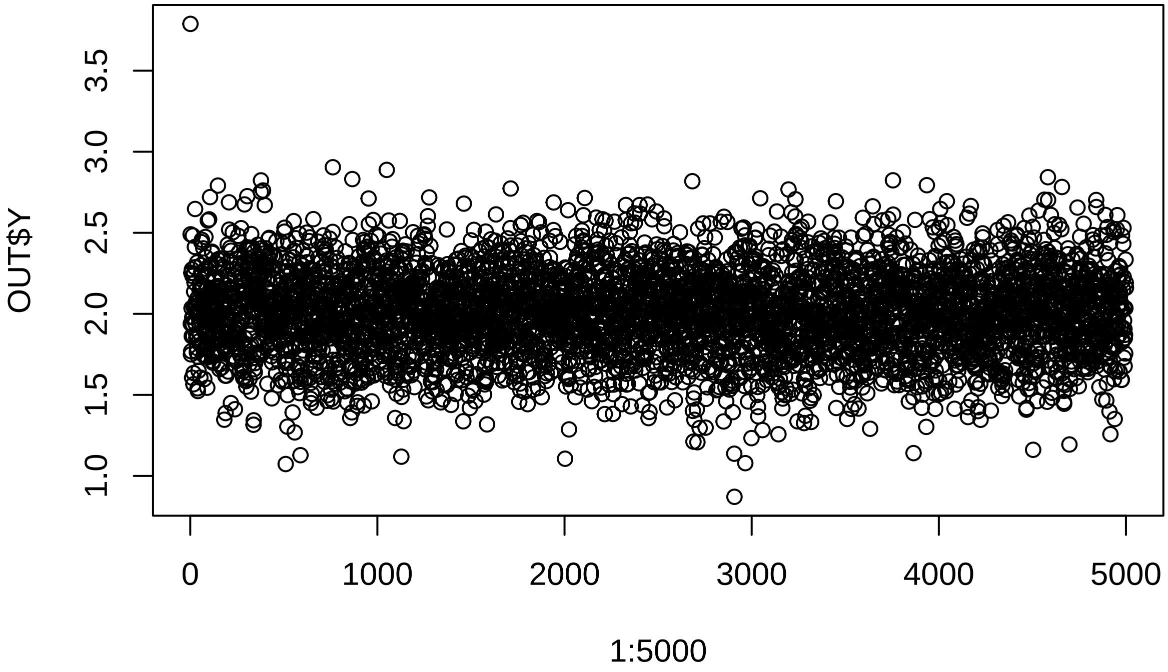
```
Gibbs <- function(N,mu,Sigma,x0,nburn,see.d){
  # initialize X and Y
  X <- rep(0,N)
  Y <- X
  X[1] = x0
  # extract from sigma
  mu1 = mu[1]
  mu2 = mu[2]
  s1 = sqrt(Sigma[1,1])
  s2 = sqrt(Sigma[2,2])
  rho = Sigma[2,1]

  #for loop to generate samples
  set.seed(see.d)
  for (i in 1:N){
    # sample from conditional rnorm

      # Sample X from conditional normal distribution
      y.mu = mu2 + rho*s2/s1*(X[i]-mu1)
      y.sd = s2^2*(1-rho)
      Y[i] <- rnorm(1,y.mu,y.sd)

      # Sample Y from conditional normal distribution
      x.mu = mu1 + rho*s1/s2*(Y[i]-mu2)
      x.sd = s1^2*(1-rho)
      X[i+1] <- rnorm(1,x.mu,x.sd)

  }
  X = X[1:N]
  # Discard burn-in samples
  return(list(X=X,Y=Y,Xburned = X[(nburn + 1):N], Yburned = Y[(nburn + 1):N]))
}
mu  = c(1,2)
S = matrix(c(1,.5,.5,.4),nrow =2, byrow =T)
Gibbs(5000,mu,S,7,1000,534) -> OUT
```

part (b)

```
plot(1:5000,OUT$X)
```

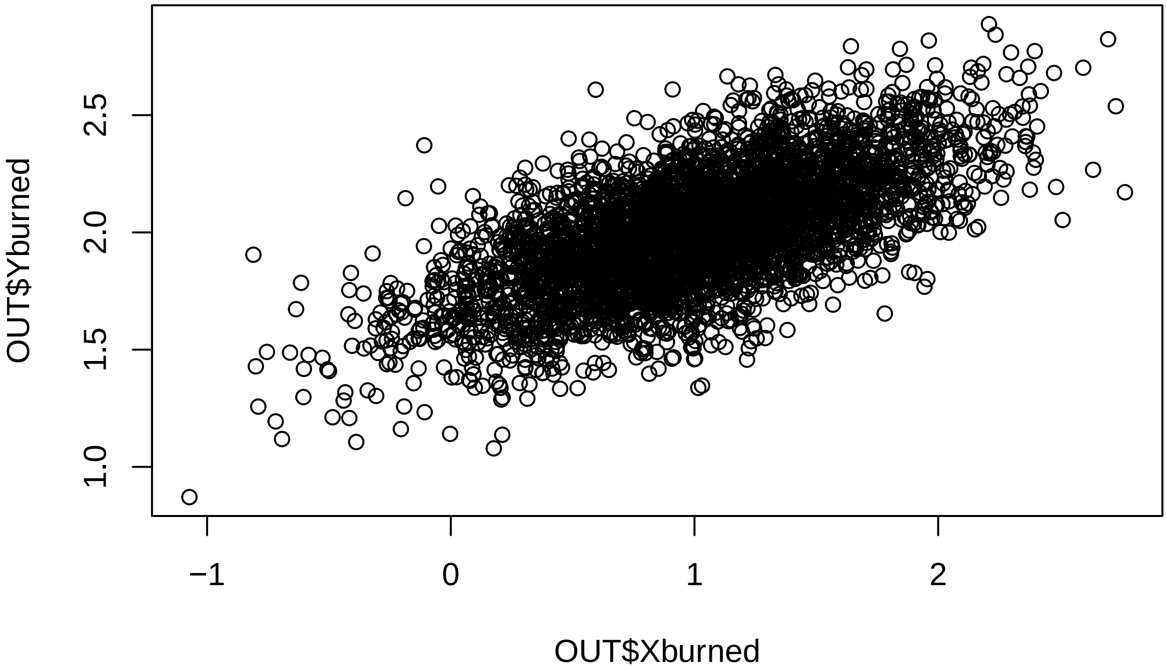


```
plot(1:5000,OUT$Y)
```



part (c)

```
plot(OUT$Xburned,OUT$Yburned)
```



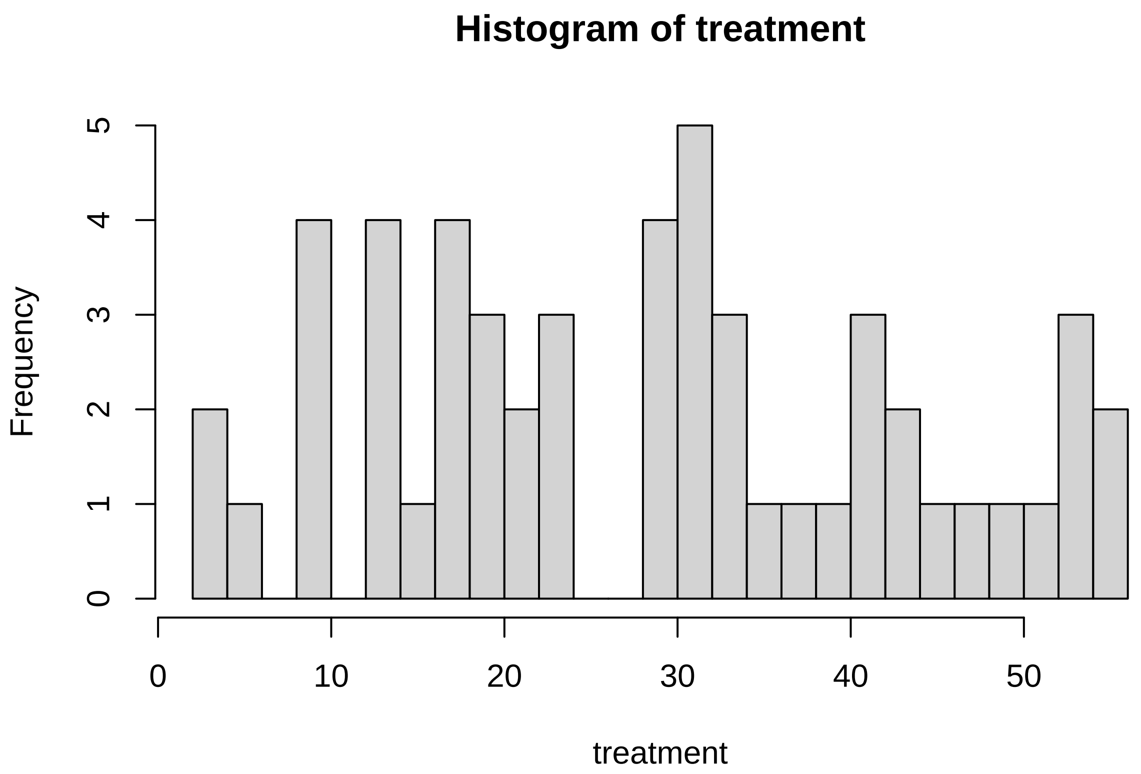
You could say this resembles an ellipses that is centered at (1,2); the shadow of a bivariate Normal Distribution. The variance of the X variables are more spread than that of the Y variables as var(Y) < var(X).

Problem 2

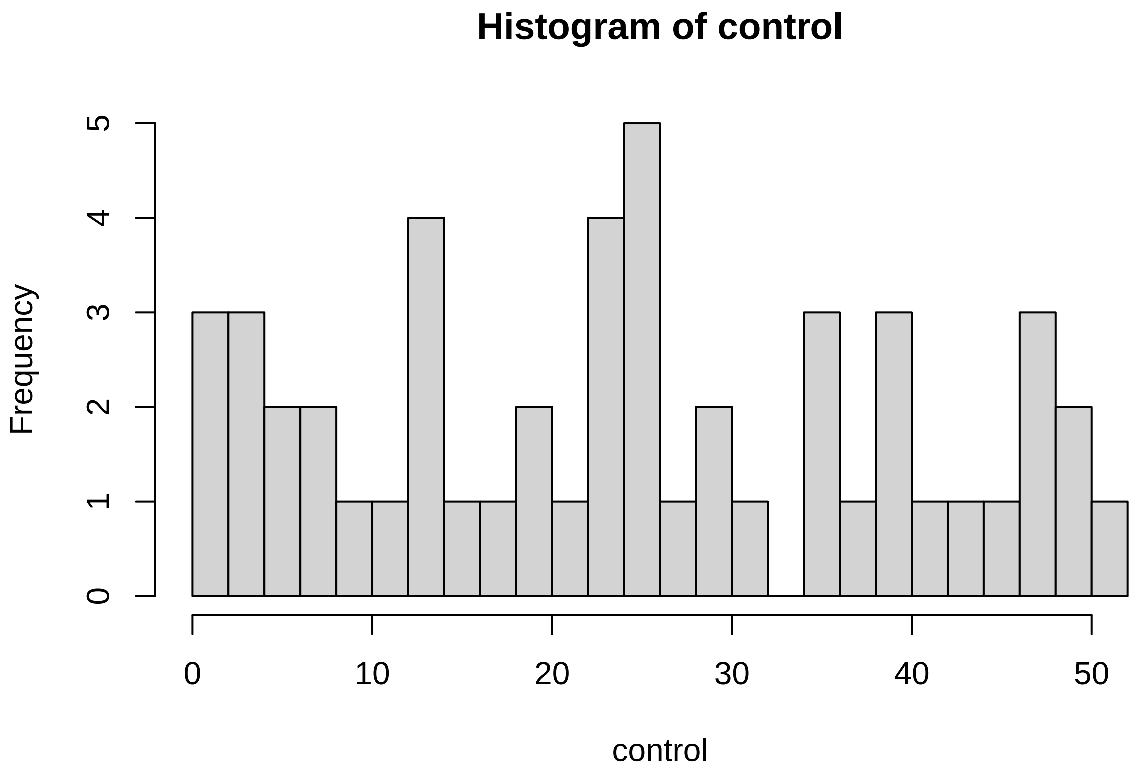
```
# load data
data <- read.table("breastcancer.dat", head =T)
```

part (a)

```
# make boxplots
treatment = data$recurtime[data$treatment == 1]
control = data$recurtime[data$treatment == 0]
hist(treatment,breaks=20)
```



```
hist(control,breaks = 20)
```



part (b)

we know that

$$L(\theta,\tau|y) * f(\theta,\tau) \propto \theta^{(\sum \delta_i^C + \sum \delta_i^H)} \tau^{(\sum \delta_i^H)} exp(-\theta \sum x_i^C - \tau\theta \sum x_i^H) \cdot \theta^a \tau^b exp(-\theta(c+d\tau))$$

which simplifies to

$$\theta^{(a+\sum \delta_i^C + \sum \delta_i^H)} \tau^{(b+\sum \delta_i^H)} exp(-\theta(c+d\tau+\tau\sum x_i^H + \sum x_i^C))$$

so when we look for everything depending on θ we get

$$f(\theta|\tau,y) = \theta^{(a+\sum \delta_i^C + \sum \delta_i^H)} exp(-\theta(c+d\tau+\tau\sum x_i^H + \sum x_i^C))$$

which is a gamma distribution $X \sim \Gamma(a+\sum\delta_i^C + \sum\delta_i^H + 1, c+d\tau+\sum x_i^H + \sum x_i^C)$
Likewise, doing to the same for τ we get

$$f(\tau|\theta,y) = \tau^{(b+\sum \delta_i^H)} exp(-\theta(d\tau+\tau\sum x_i^H))$$

which is also gamma distribution $X \sim \Gamma(b+\sum\delta_i^H + 1, \theta(d+\sum x_i^H))$

part (c)

```
GS <- function(N,data,theta,see.d){

 # set seed
 set.seed(see.d)

 # intialize sample vec
 gsvec = matrix(0,N,2)
 # hyperparams
 a = 3
 b = 1
 c = 60
 d = 120
 # get delts
 delta.c = sum(data$treatment == 0 & data$censored == 0)
 delta.h = sum(data$treatment == 1 & data$censored == 0)

 # control group
 x.c = sum(data$recurtime[data$treatment==0])
 x.h = sum(data$recurtime[data$treatment==1])

 # run loop
 for(i in 1:N){
 # tau | theta
 tau.theta = rgamma(1,shape = delta.h+b+1,rate = theta * (x.h + d))
 # theta | tau
 theta.tau = rgamma(1,shape = delta.c+delta.h+a+1,rate = (tau.theta *( x.h + d) + c + x.c))
 gsvec[i,] = c(tau.theta,theta.tau)
 theta = theta.tau
 }
 return(gsvec)
}
```

```
vec = GS(10000,data,1,534)
summary(mcmc(vec))
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean      SD  Naive SE Time-series SE
## [1,] 1.206764 0.48278 0.0048278      9.668e-03
## [2,] 0.009297 0.00266 0.0000266      5.274e-05
##
## 2. Quantiles for each variable:
##
##          2.5%      25%      50%     75%   97.5%
## var1 0.538415 0.865593 1.113011 1.44551 2.39937
## var2 0.004822 0.007404 0.009033 0.01095 0.01505
```

problem (d)

```
# render letters
tau <- vec[,1]
theta <- vec[,2]

# summary
sprintf("theta mean is %f while tau mean is %f",mean(theta),mean(tau))
```

```
## [1] "theta mean is 0.009297 while tau mean is 1.206764"
```

```
sprintf("theta CI is ( %f , %f )",quantile(theta, 0.025),quantile(theta, 0.975))
```

```
## [1] "theta CI is ( 0.004822 , 0.015055 )"
```

```
sprintf("tau CI is ( %f , %f )",quantile(tau, 0.025),quantile(tau, 0.975))
```

```
## [1] "tau CI is ( 0.538415 , 2.399374 )"
```

```
sprintf("Standard Dev. of theta : %f",sd(theta))
```

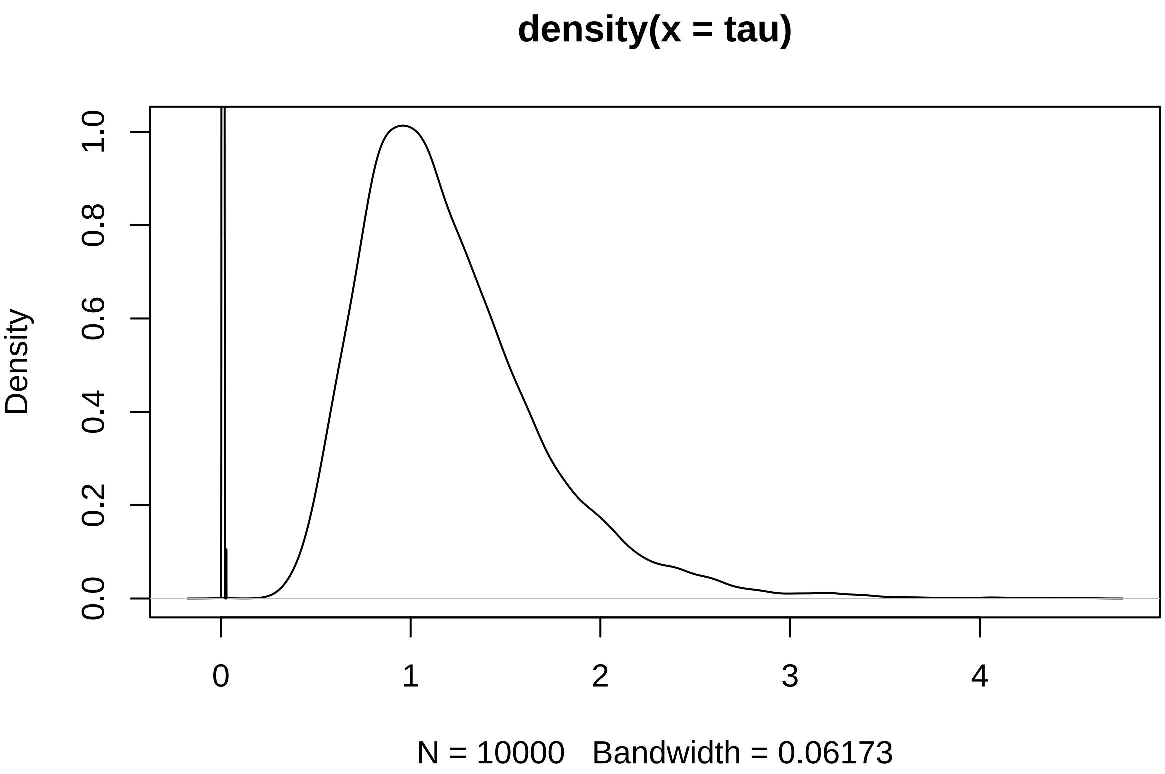
```
## [1] "Standard Dev. of theta : 0.002660"
```

```
sprintf("Standard Dev. of tau : %f",sd(tau))
```

```
## [1] "Standard Dev. of tau : 0.482782"
```

problem (e)

```
# render graphics
plot(density(tau))
lines(density(theta))
```



From our findings we can conclude that there is a 1.206764 factor difference in recurrence time between the hormone treated group and control group.

problem (g)

```
GS.2 <- function(f,N,data,theta,see.d){

 # set seed
 set.seed(see.d)

 # intialize sample vec
 gsvec = matrix(0,N,2)
 # hyperparams
 a = 3*f
 b = 1*f
 c = 60*f
 d = 120*f
 # get delts
 delta.c = sum(data$treatment == 0 & data$censored == 0)
 delta.h = sum(data$treatment == 1 & data$censored == 0)

 # control group
 x.c = sum(data$recurtime[data$treatment==0])
 x.h = sum(data$recurtime[data$treatment==1])

 # run loop
 for(i in 1:N){
 # tau | theta
 tau.theta = rgamma(1,shape = delta.h+b+1,rate = theta * (x.h + d))
 # theta | tau
 theta.tau = rgamma(1,shape = delta.c+delta.h+a+1,rate = (tau.theta *( x.h + d) + c + x.c))
 gsvec[i,] = c(tau.theta,theta.tau)
 theta = theta.tau
 }
 return(gsvec)
}

vec.half = GS.2(1/2,10000,data,1,534)
summary(mcmc(vec.half))
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD  Naive SE Time-series SE
## [1,] 1.308366 0.544328 5.443e-03      1.127e-02
## [2,] 0.008713 0.002608 2.608e-05      5.246e-05
##
## 2. Quantiles for each variable:
##
##          2.5%      25%      50%    75%   97.5%
## var1 0.570576 0.924121 1.197060 1.5776 2.64827
## var2 0.004354 0.006865 0.008448 0.0103 0.01441
```

```
GS.2 <- function(f,N,data,theta,see.d){

 # set seed
 set.seed(see.d)

 # intialize sample vec
 gsvec = matrix(0,N,2)
 # hyperparams
 a = 3*f
 b = 1*f
 c = 60*f
 d = 120*f
 # get delts
 delta.c = sum(data$treatment == 0 & data$censored == 0)
 delta.h = sum(data$treatment == 1 & data$censored == 0)

 # control group
 x.c = sum(data$recurtime[data$treatment==0])
 x.h = sum(data$recurtime[data$treatment==1])

 # run loop
 for(i in 1:N){
 # tau | theta
 tau.theta = rgamma(1,shape = delta.h+b+1,rate = theta * (x.h + d))
 # theta | tau
 theta.tau = rgamma(1,shape = delta.c+delta.h+a+1,rate = (tau.theta *( x.h + d) + c + x.c))
 gsvec[i,] = c(tau.theta,theta.tau)
 theta = theta.tau
 }
 return(gsvec)
}

vec.double = GS.2(2,10000,data,1,534)
summary(mcmc(vec.double))
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean       SD  Naive SE Time-series SE
## [1,] 1.06297 0.404979 4.050e-03      7.869e-03
## [2,] 0.01032 0.002733 2.733e-05      5.304e-05
##
## 2. Quantiles for each variable:
##
##          2.5%      25%     50%     75%   97.5%
## var1 0.497220 0.778352 0.98547 1.26821 2.05331
## var2 0.005601 0.008386 0.01009 0.01205 0.01613
```

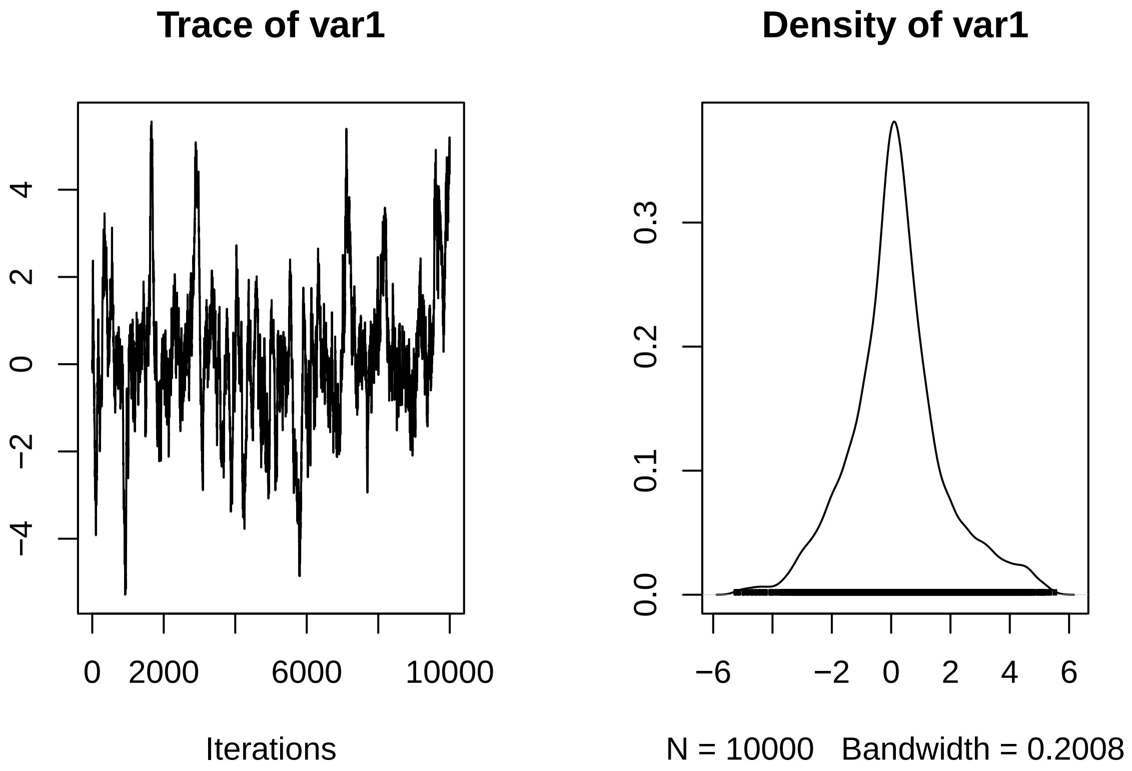
The Tau mean value seemed to not change as much as when the hyperparameters to doubled.

Problem 3

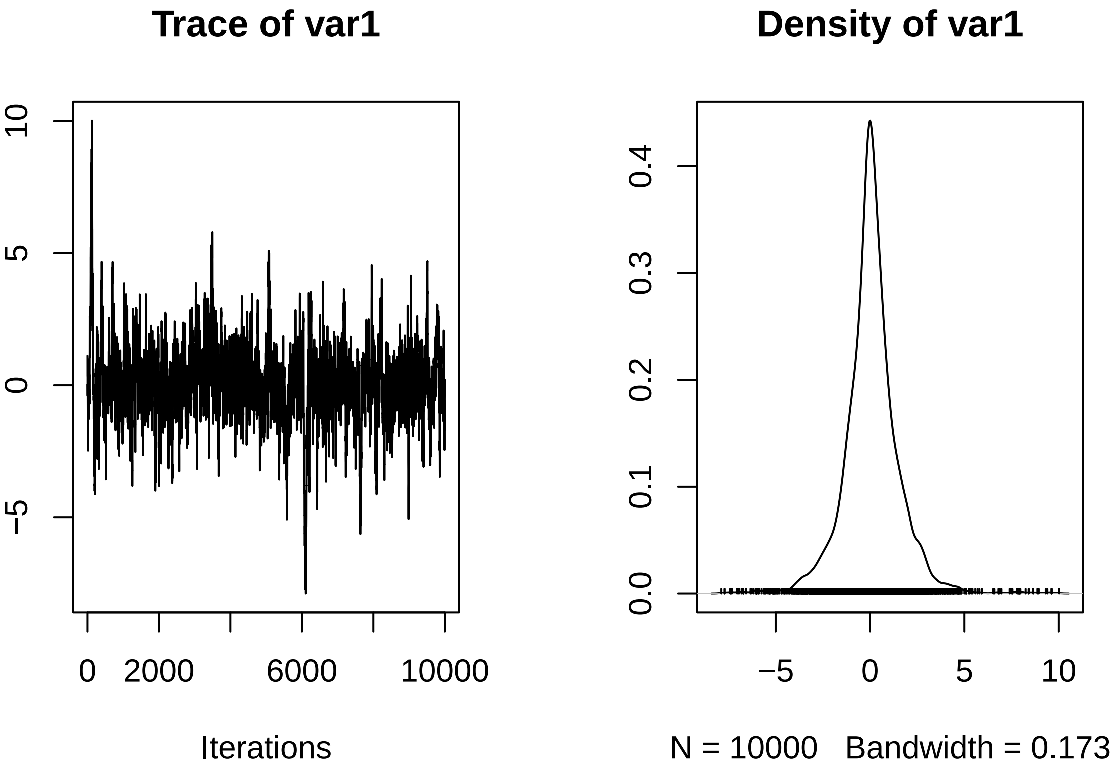
```
set.seed(534)
# render pmf
P <- function(X){0.5*exp(-abs(X))}
# make looping algorithm
MH <- function(N,var){
  # make a sample vector
  vec <- rep(0,N)
  vec_bar = vec
  s <- sqrt(var)
  vec[1] <- rnorm(1,0,s)
  accept = 0
  # loop
  for(i in 2:N){
    # sample from norm
    rnorm(1,vec[i-1],s) -> x1
    runif(1) -> u
    P(x1)/P(vec[i-1]) -> alpha
    if(u < alpha){
      vec[i] = x1
      accept = accept+1
      #x1 -> x0
    } else {
      vec[i] = vec[i-1]
    }
    vec_bar[i] <- mean(vec[1:i])
  }
  # return sample vector
  list(vec = vec, means = vec_bar,ratio = accept/N)
}
```

```
VARvec <- c(.05,.5,1,3,100)
n = 10000
for(i in 1:5){
  chain = MH(n,VARvec[i])
  print(chain$ratio)
  summary(mcmc(chain$vec))
  plot(mcmc(chain$vec))
  }
```

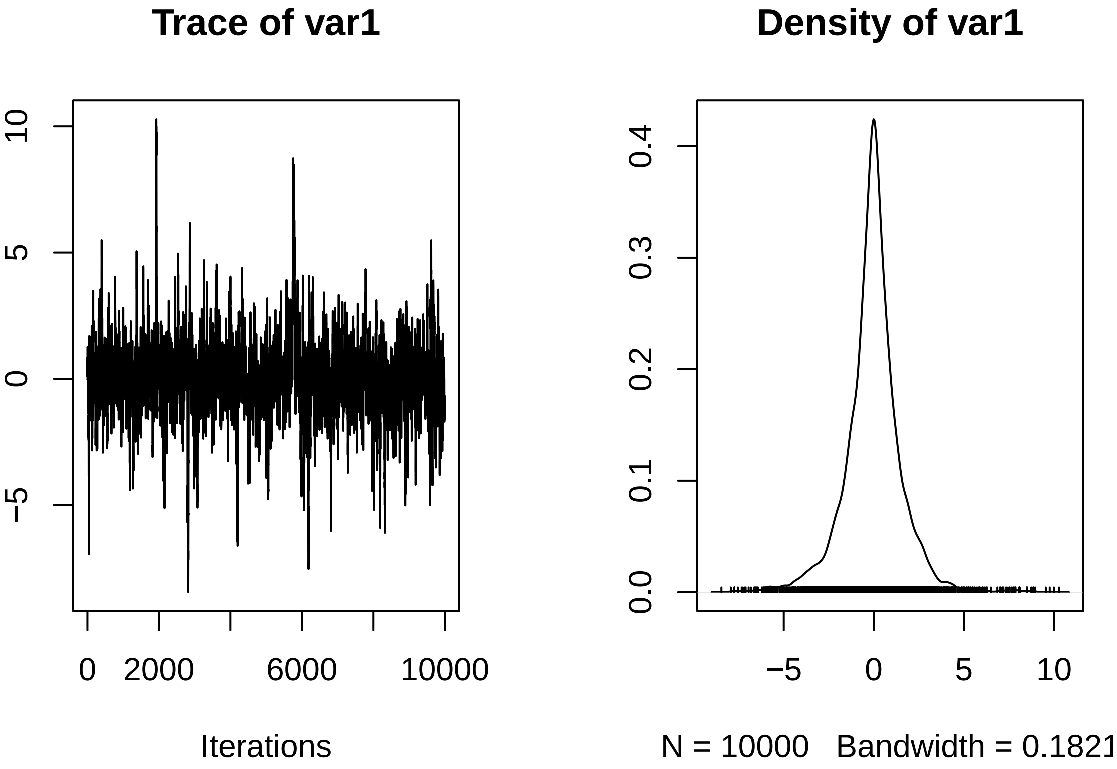
```
## [1] 0.921
```



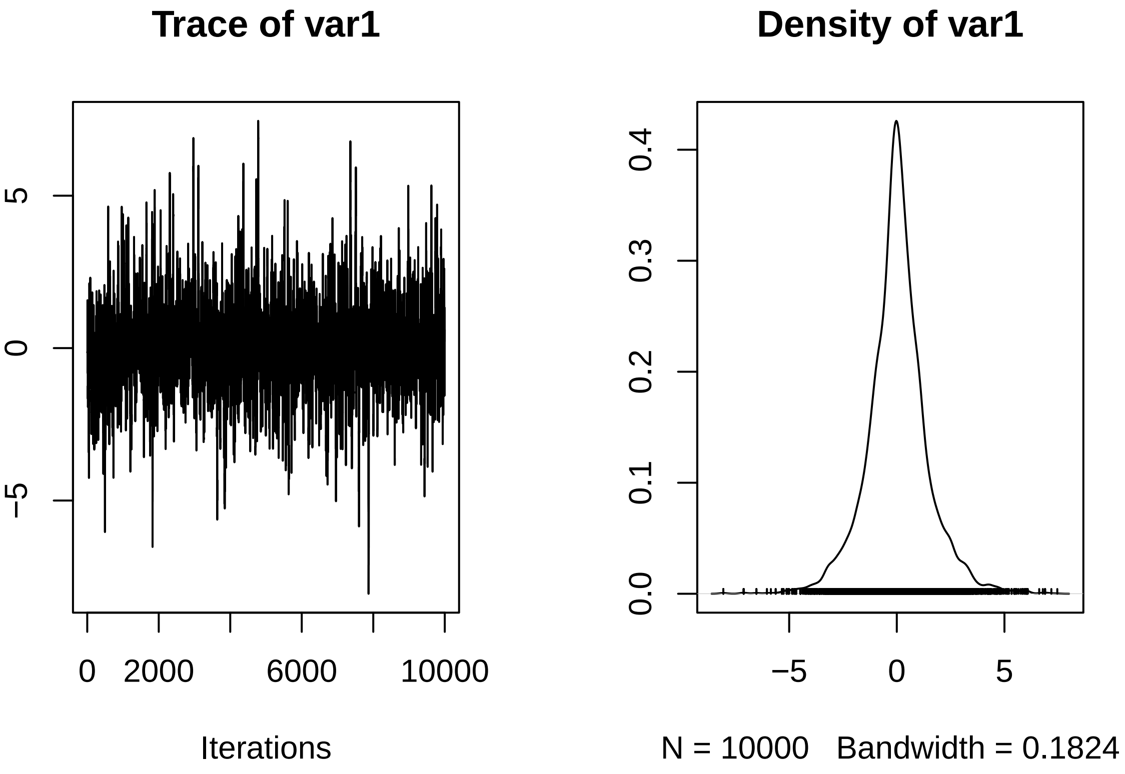
```
## [1] 0.7741
```


Trace of var1
10
5
0
−5
0
2000
6000
10000
Iterations
Density of var1
0.4
0.3
0.2
0.1
0.0
−5
0
5
10
N = 10000   Bandwidth = 0.173

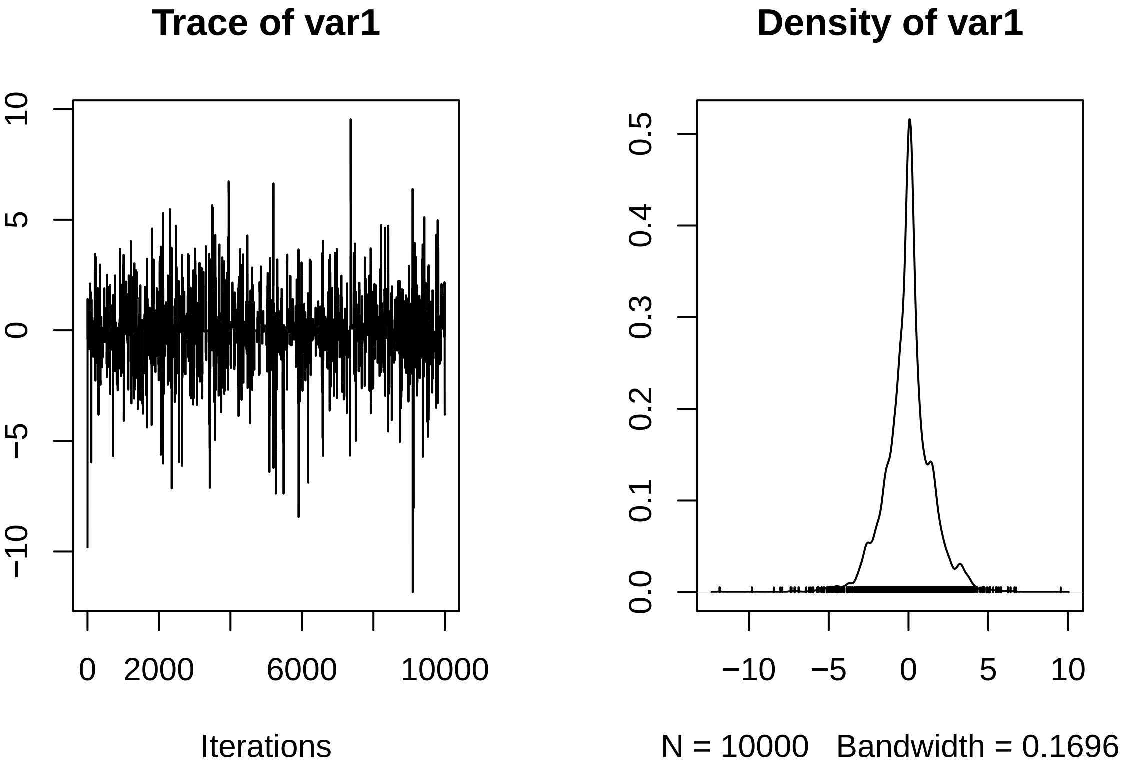
```
## [1] 0.7
```


Trace of var1
10
5
0
−5
0
2000
6000
10000
Iterations
Density of var1
0.4
0.3
0.2
0.1
0.0
−5
0
5
10
N = 10000   Bandwidth = 0.1821

```
## [1] 0.5632
```



```
## [1] 0.1563
```



```
set.seed(534)
 P <- function(X){
  if(X >= 0){exp(-X)}
  else{0}
```

```
    }   # render pmf
# make looping algorithm
MHexp <- function(N,var){
  # make a sample vector
  vec <- rep(0,N)
  vec_bar = vec
  s <- sqrt(var)
  vec[1] <- rnorm(1,0,s)
  accept = 0
  # loop
  for(i in 2:N){
    # sample from norm
    rnorm(1,vec[i-1],s) -> x1
   # while(x1 < 0){rnorm(1,vec[i-1],s) -> x1}
    runif(1) -> u
    P(x1)/P(vec[i-1]) -> alpha
    if(u < alpha){
      vec[i] = x1
      accept = accept+1
      #x1 -> x0
    } else {
      vec[i] = vec[i-1]
    }
    vec_bar[i] <- mean(vec[1:i])
  }
  # return sample vector
  list(vec = vec, means = vec_bar,ratio = accept/N)
}
```

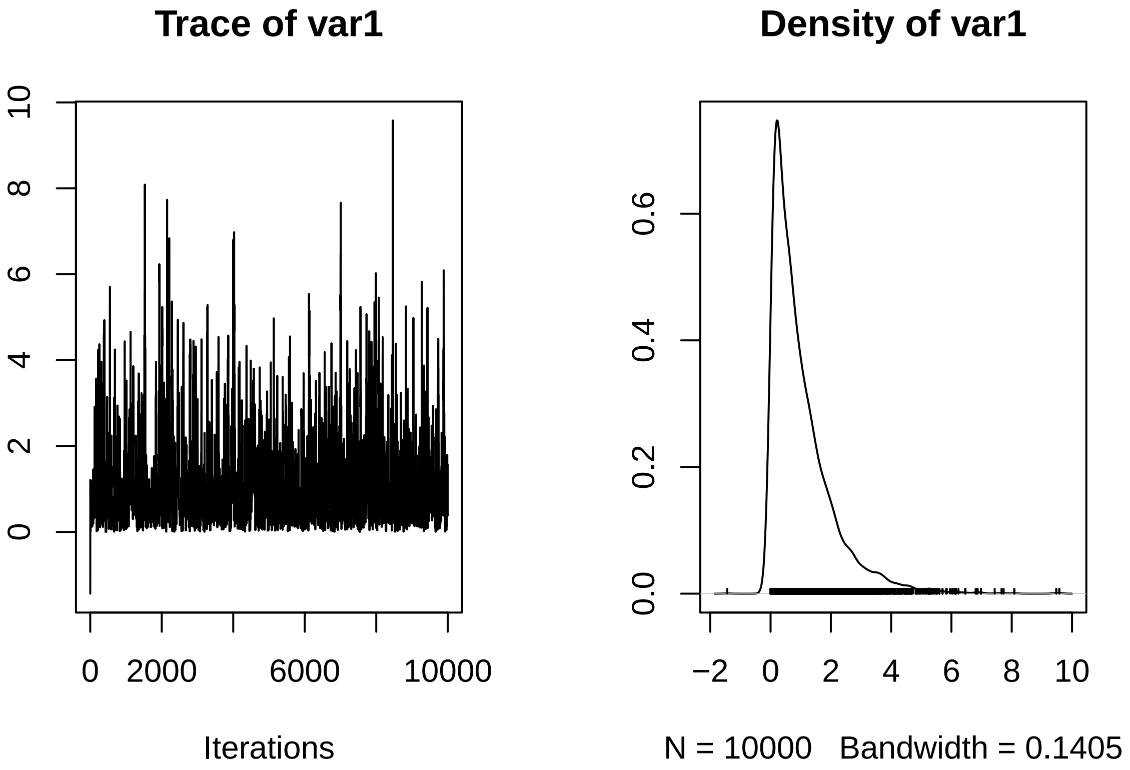
```
  chain = MHexp(10000,7)
  sprintf("Acceptance rate is %f",chain$ratio)
```

```
## [1] "Acceptance rate is 0.270900"
```

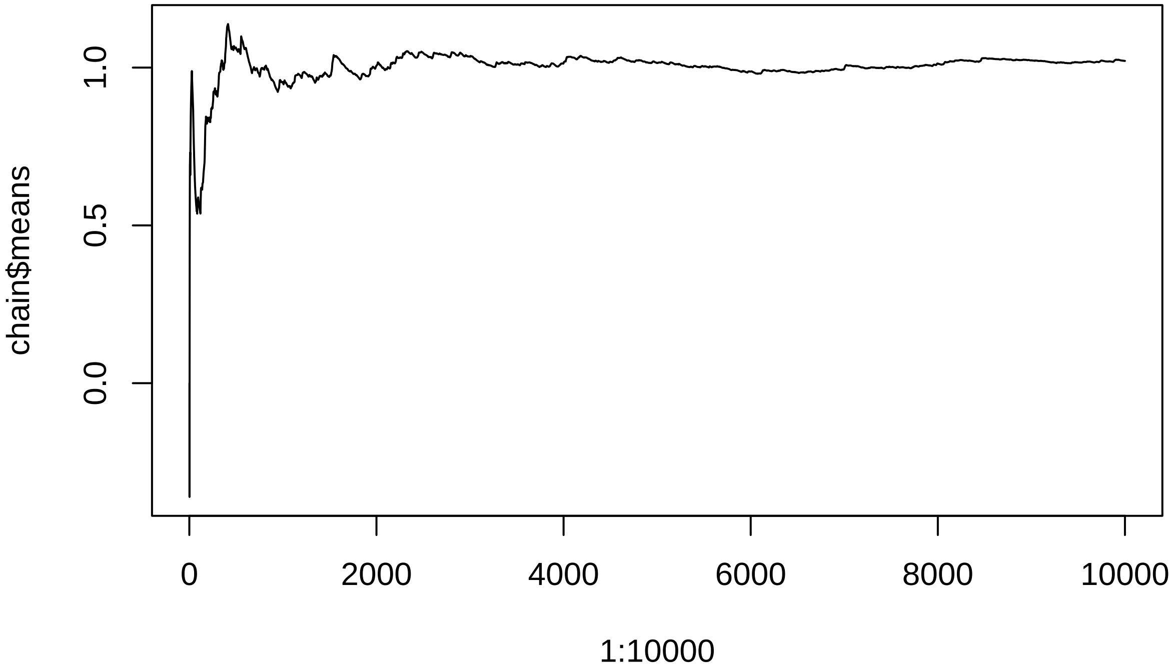
```
  summary(mcmc(chain$vec))
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##        1.02114        1.02972        0.01030        0.03533
##
## 2. Quantiles for each variable:
##
##    2.5%     25%     50%     75%   97.5%
## 0.03154 0.28735 0.70831 1.40771 3.80582
```

```
plot(mcmc(chain$vec))
```

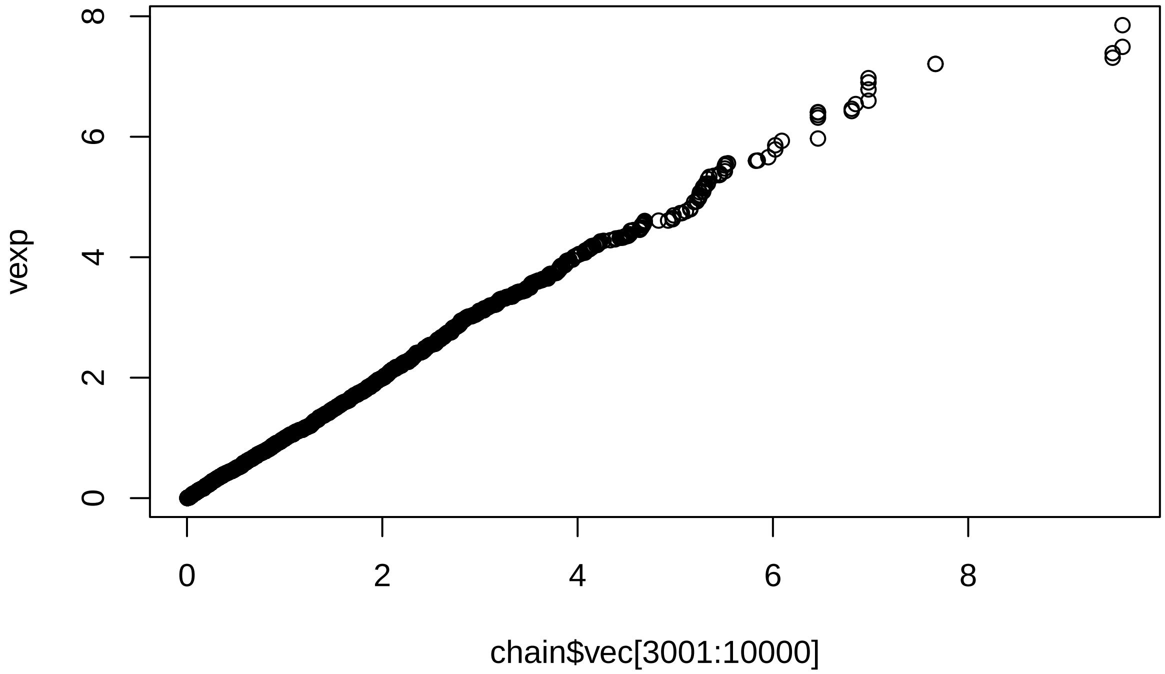


```
plot(1:10000,chain$means,type = 'l')
```



I decided to burn 3000.

```
vexp <- rexp(7000,1)
qqplot(chain$vec[3001:10000],vexp)
```

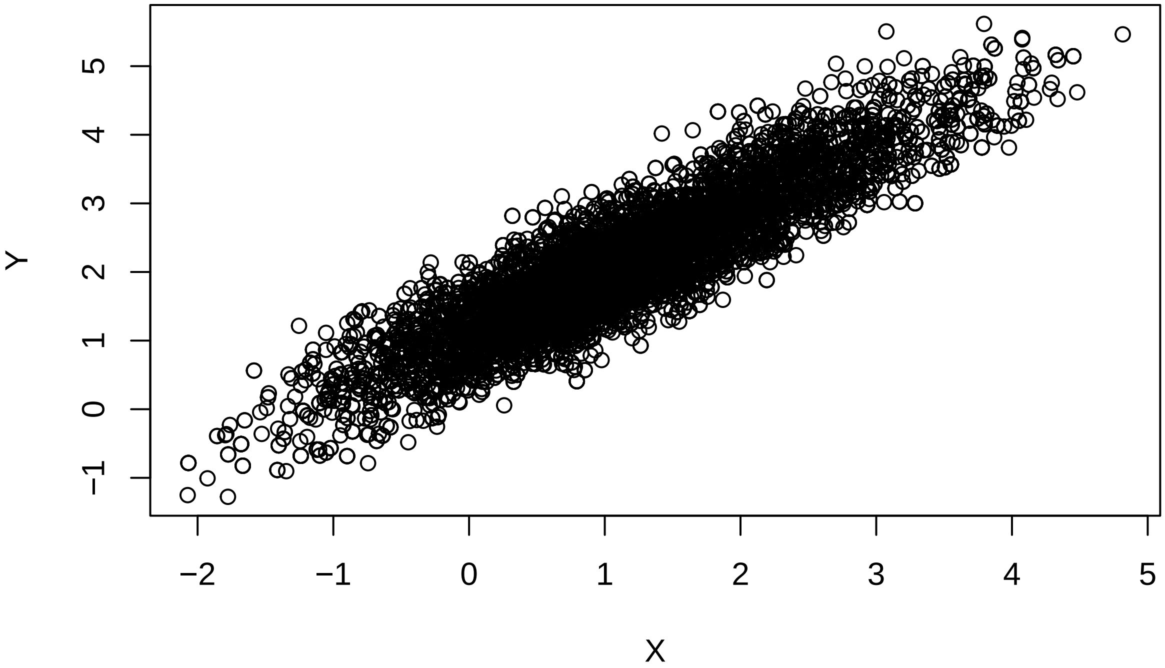


Problem (5).

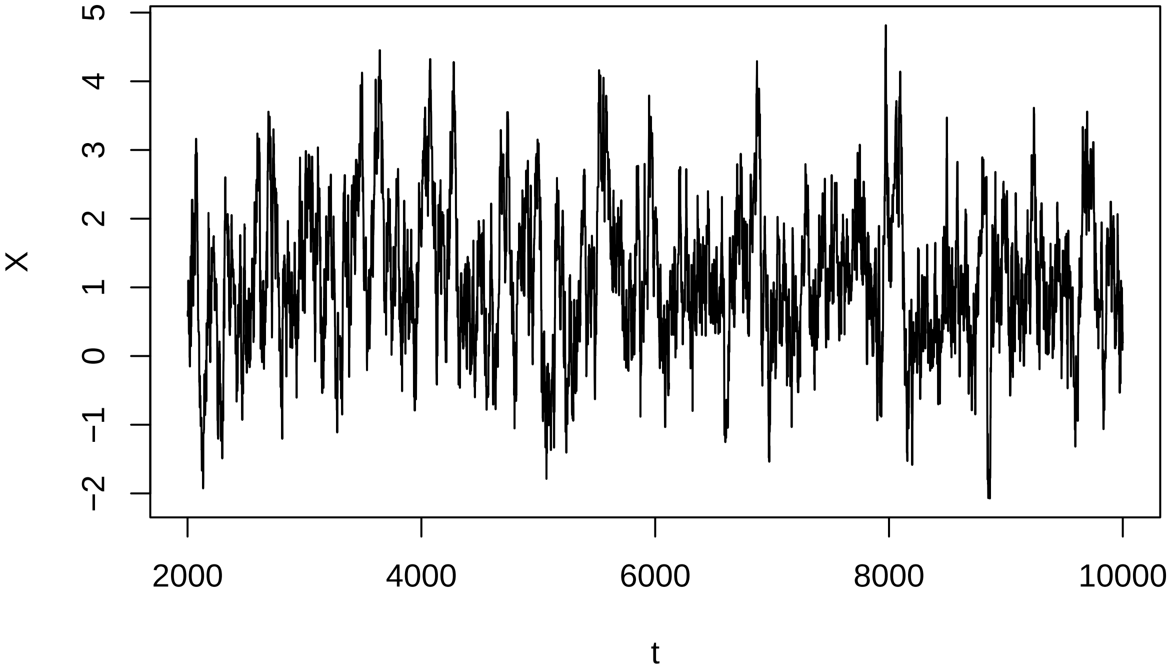
system 1

```
# render bivariate normal function
P <- function(v){
  v[1] -> X
  v[2] -> Y
  x = matrix(c(X,Y),nrow=2)
  mu = matrix(c(1,2),nrow=2)
  S = matrix(c(1,.9,.9,1),nrow=2,byrow=T)
  1/(2*pi*sqrt(det(S)))*exp(-.5*t(x-mu)%*%solve(S)%*%(x-mu))
}
# make looping algorithm
WALK.BVU <- function(N,nburn){
  # make a sample vector
  vec <- matrix(0,N,2)
  vec_bar = vec
  vec[1,1] <- runif(1,-.75,.75)
  vec[1,2] <- runif(1,-1,1)
  accept = 0
  # loop
  for(i in 2:N){
    # render z
    z = c(runif(1,-.75,.75),runif(1,-1,1))
    vecprime = z + vec[i-1,]
   # while(x1 < 0){rnorm(1,vec[i-1],s) -> x1}
    runif(1) -> u
   min(P(vecprime)/P(vec[i-1,]),1) -> alpha
    if(u < alpha){
      vec[i,] = vecprime
      accept = accept+1
      #x1 -> x0
    } else {
      vec[i,] = vec[i-1,]
    }
    vec_bar[i,] <- mean(vec[1:i,])
  }
  # burn
  vec = vec[(nburn+1):N,]
  vec_bar = vec_bar[(nburn+1):N]
  # return sample vector
  list(vec = vec, means = vec_bar,ratio = accept/N)
}
```

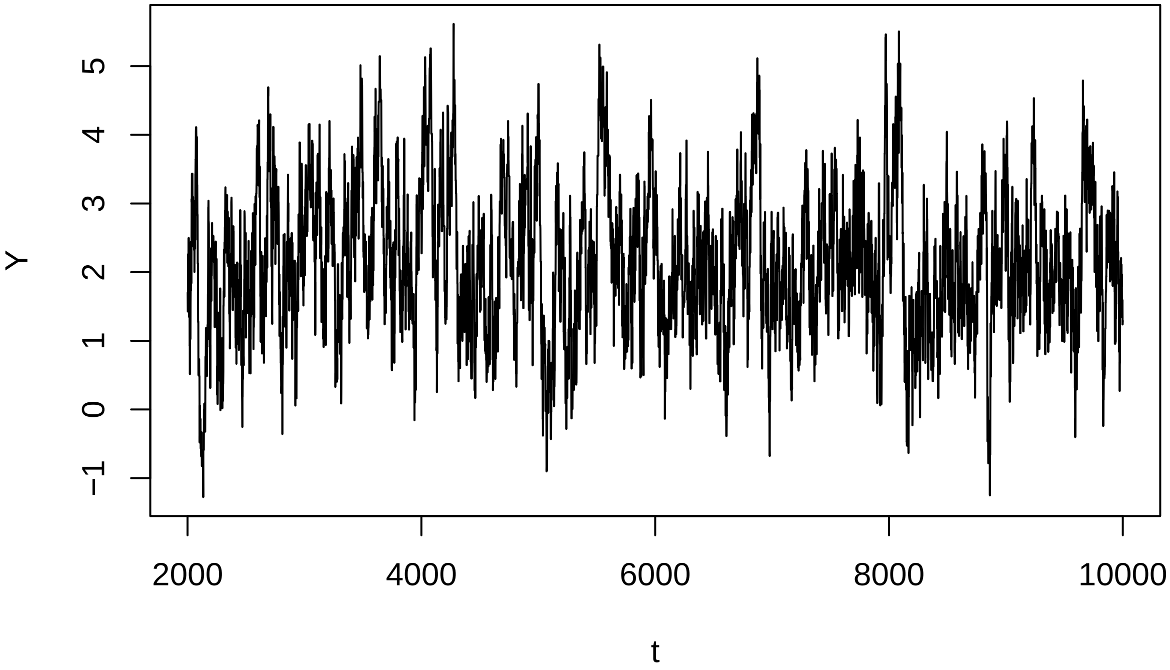
```
WALK.BVU(10000,2000) -> chain
X = chain$vec[,1]
Y = chain$vec[,2]
plot(X,Y)
```



```
t = 2001:10000
plot(t,X,type = 'l')
```



```r
plot(t,Y,type = "l")
```



```r
sprintf("Acceptance rate is for system 1 is %f",chain$ratio)
```

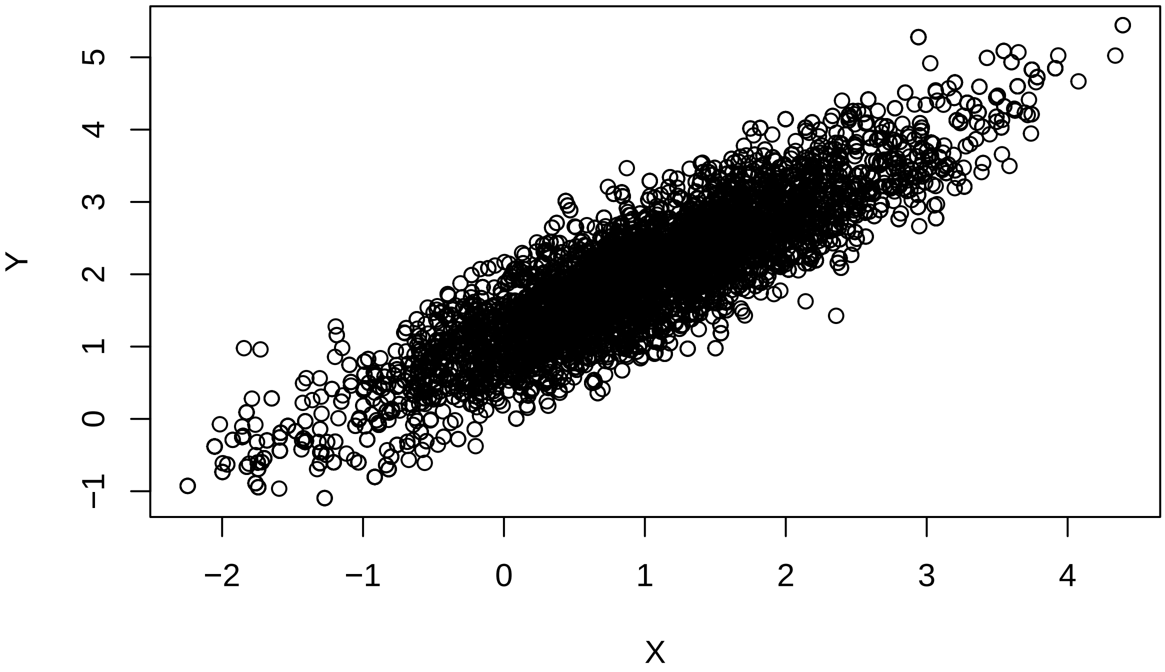
```
## [1] "Acceptance rate is for system 1 is 0.515000"
```

This is not an acceptable acceptance rate because it is just above the 40-50% range in the article.

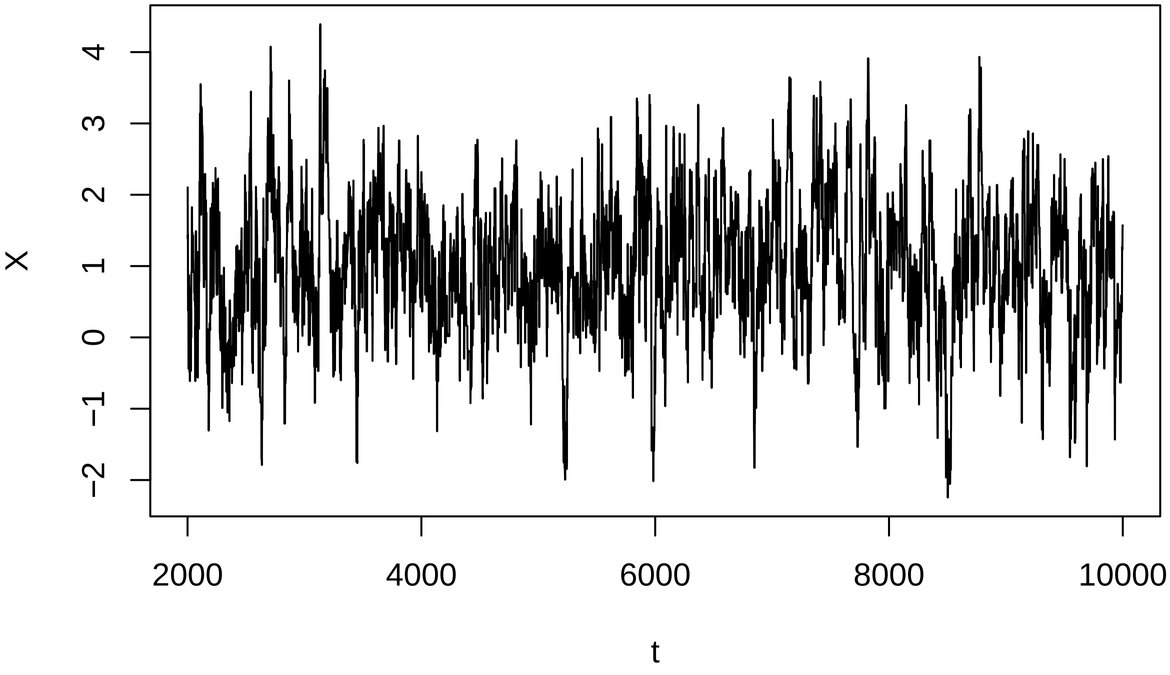
system 2

```r
# make looping algorithm
WALK.N2 <- function(N,nburn){
  # make a sample vector
  vec <- matrix(0,N,2)
  vec_bar = vec
  vec[1,] <- mvrnorm(1,c(0,0),diag(c(.6,.4)))
  accept = 0
  # loop
  for(i in 2:N){
    # render z
    z = mvrnorm(1,c(0,0),diag(c(.6,.4)))
    vecprime = z + vec[i-1,]
   # while(x1 < 0){rnorm(1,vec[i-1],s) -> x1}
    runif(1) -> u
   min(P(vecprime)/P(vec[i-1,]),1) -> alpha
    if(u < alpha){
      vec[i,] = vecprime
      accept = accept+1
      #x1 -> x0
    } else {
      vec[i,] = vec[i-1,]
    }
    vec_bar[i,] <- mean(vec[1:i,])
  }
  # burn
  vec = vec[(nburn+1):N,]
  vec_bar = vec_bar[(nburn+1):N]
  # return sample vector
  list(vec = vec, means = vec_bar,ratio = accept/N)
}
```

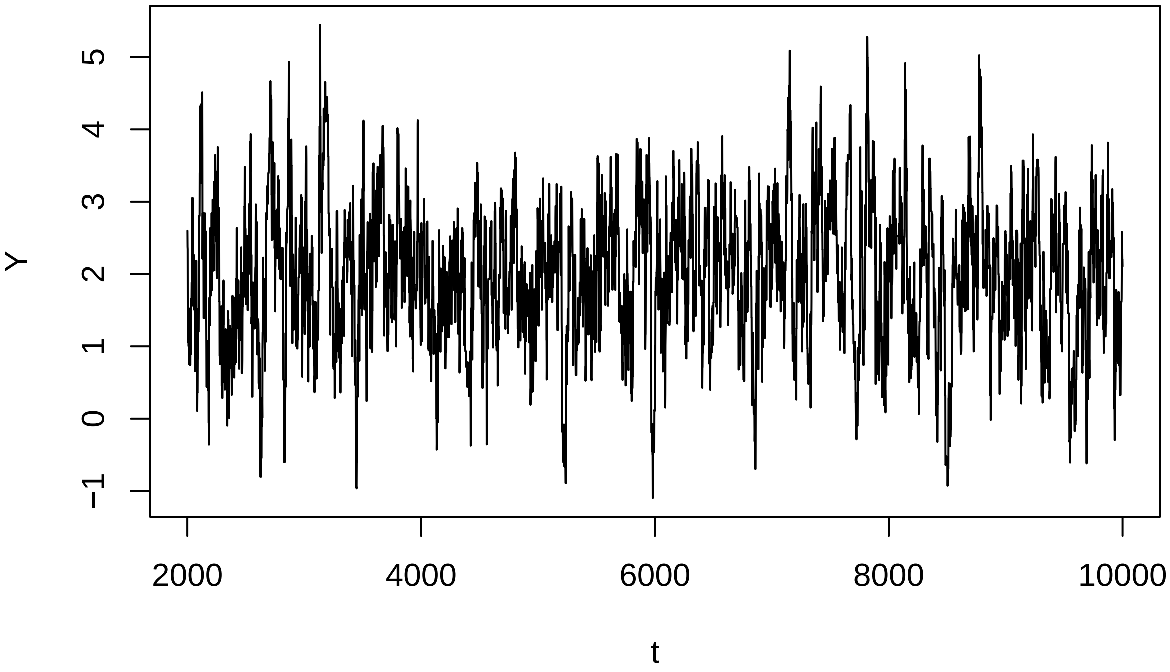
```
WALK.N2(10000,2000) -> chain
X = chain$vec[,1]
Y = chain$vec[,2]
plot(X,Y)
```



```
t = 2001:10000
plot(t,X,type = 'l')
```



```
plot(t,Y,type = "l")
```



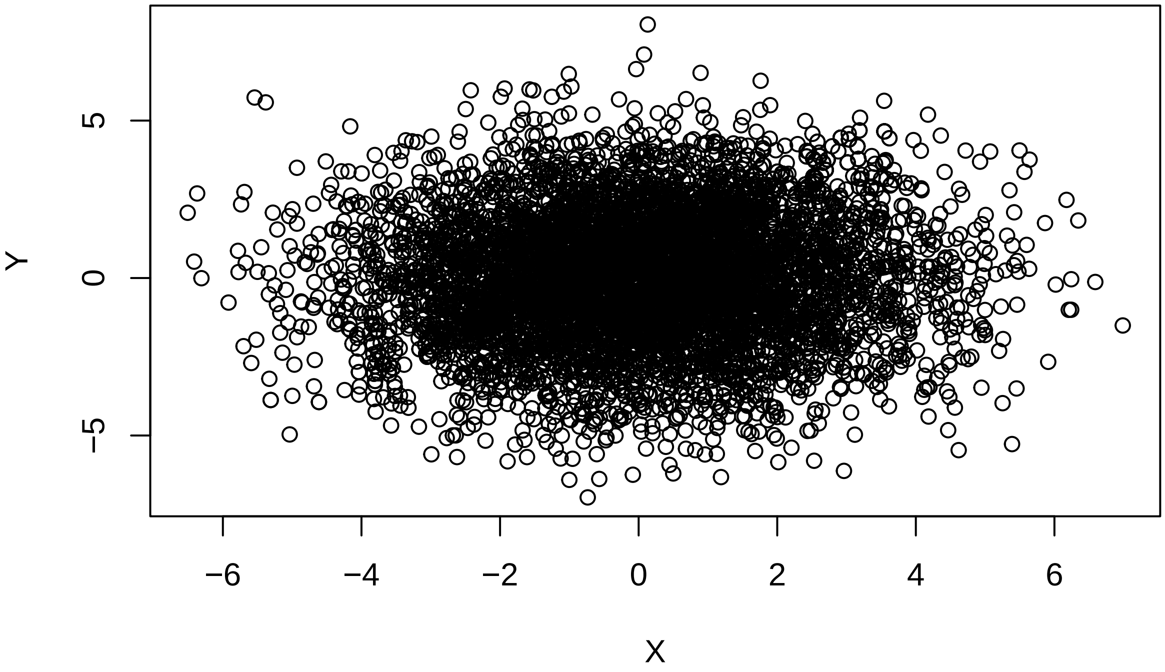
```
sprintf("Acceptance rate is for system 2 is %f",chain$ratio)
```

```
## [1] "Acceptance rate is for system 2 is 0.439300"
```

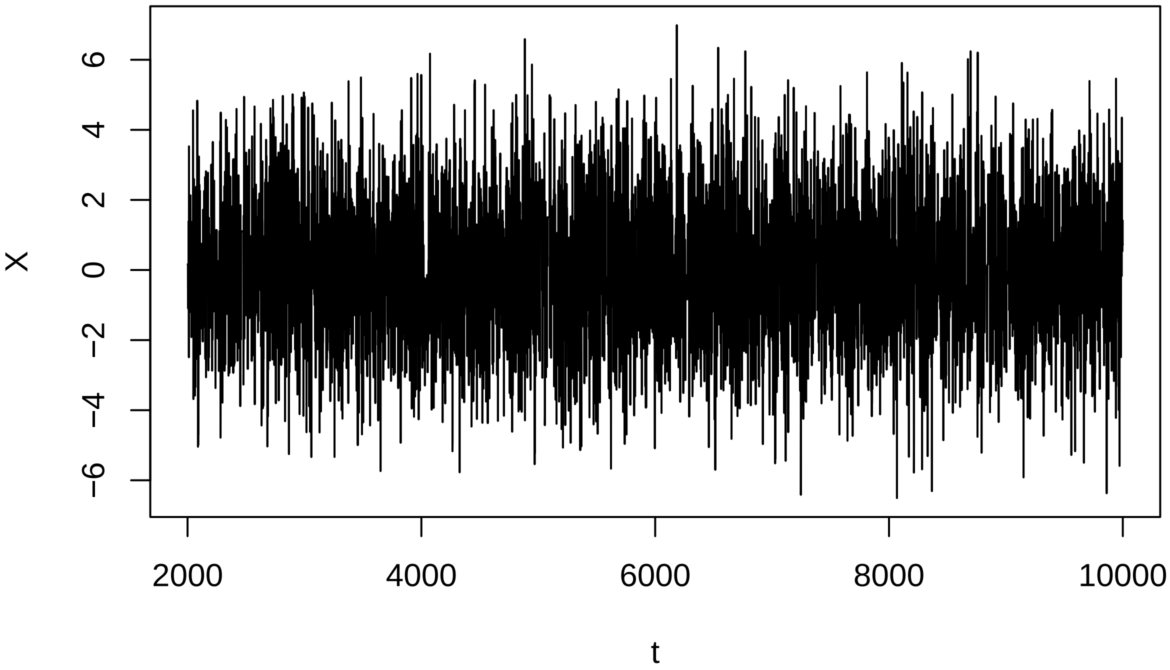
system 3

```
# render bivariate normal function
cH <- function(v){
  v[1] -> X
  v[2] -> Y
  x = matrix(c(X,Y),nrow=2)
  mu = matrix(c(1,2),nrow=2)
  S = diag(c(2,2))
  0.9/(2*pi*sqrt(det(S)))*exp(-.5*t(x-mu)%*%solve(S)%*%(x-mu))
}
# make looping algorithm
WALK.P3 <- function(N,nburn){
  # make a sample vector
  vec <- matrix(0,N,2)
  vec_bar = vec
  #vec[1,] <- mvrnorm(1,c(0,0),diag(2))
  vec[1,] <- c(rnorm(1,0,2),rnorm(1,0,2))
  accept = 0
  # loop
  for(i in 2:N){
    # render vecprime
    vecprime <- mvrnorm(1,c(0,0),diag(2))
    c(rnorm(1,0,2),rnorm(1,0,2)) -> vecprime
    runif(1) -> u
    # render alphas
      if(P(vec[i-1,]) < cH(vec[i-1,])){ #C1
       alpha = 1
      } else if(P(vec[i-1,]) >= cH(vec[i-1,]) && P(vecprime) < cH(vecprime)){ #C2
       alpha = cH(vec[i-1,])/P(vec[i-1,])
      } else{ #C3
       alpha = min(1,(P(vecprime)*cH(vec[i-1,]))/(cH(vecprime)*P(vec[i-1,])))
      }
    # accept-reject
    if(u < alpha){
      vec[i,] = vecprime
      accept = accept+1

    } else {
      vec[i,] = vec[i-1,]
    }
    vec_bar[i,] <- mean(vec[1:i,])
  }
  # burn
  vec = vec[(nburn+1):N,]
  vec_bar = vec_bar[(nburn+1):N]
  # return sample vector
  list(vec = vec, means = vec_bar,ratio = accept/N)
}
```

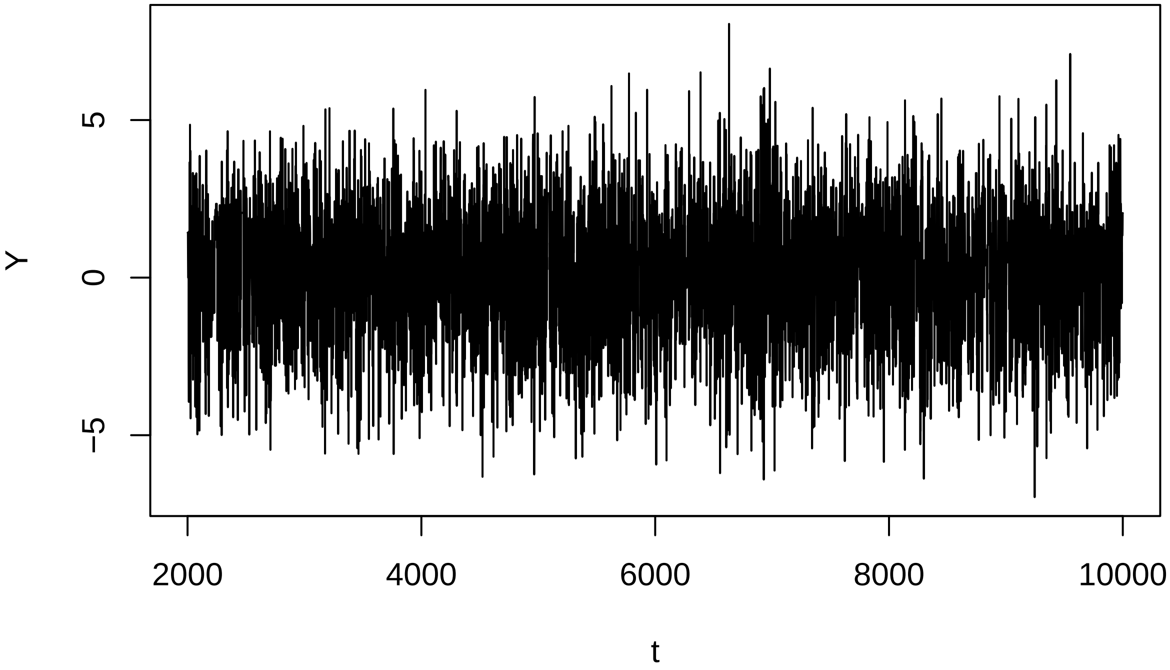
```
WALK.P3(10000,2000) -> chain
X = chain$vec[,1]
Y = chain$vec[,2]
plot(X,Y)
```



```
t = 2001:10000
plot(t,X,type = 'l')
```



```
plot(t,Y,type = "l")
```



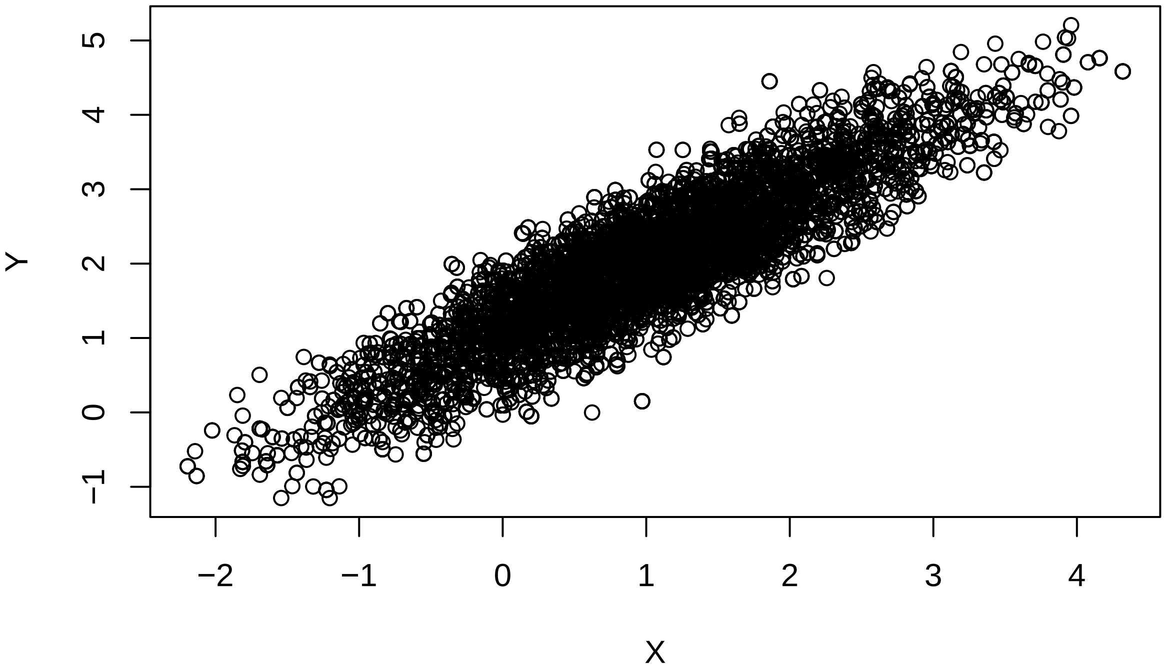
```
sprintf("Acceptance rate is for system 3 is %f",chain$ratio)
```

```
## [1] "Acceptance rate is for system 3 is 0.744300"
```

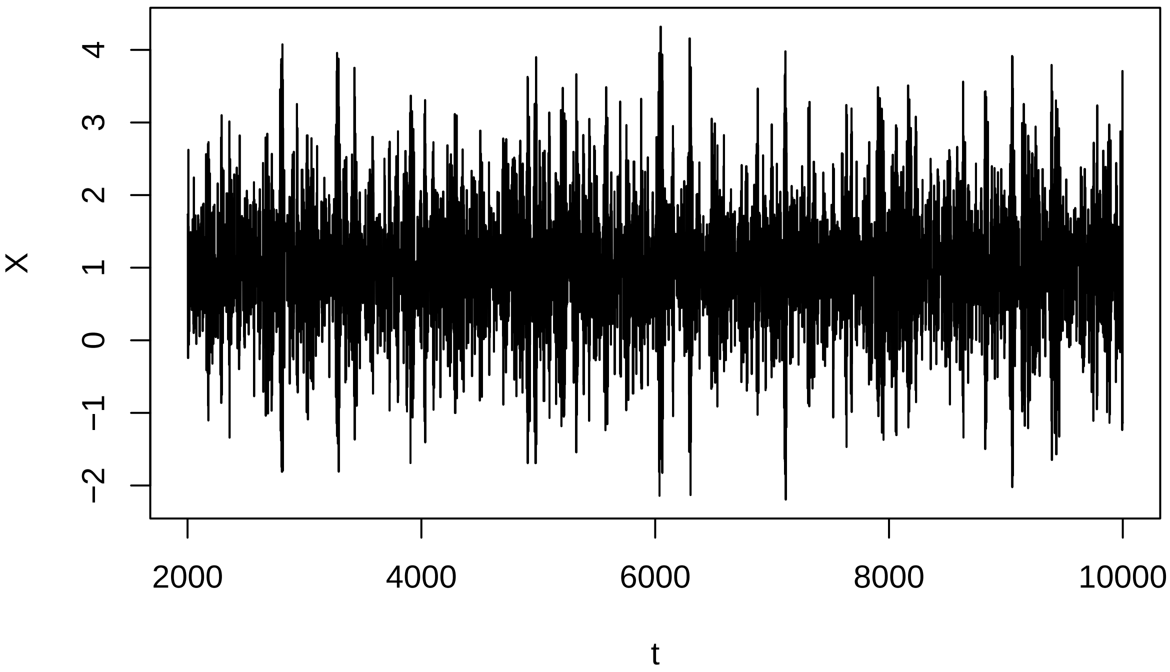
system 4

```
# make looping algorithm
WALK.BVU4 <- function(N,nburn){
  # make a sample vector
  vec <- matrix(0,N,2)
  vec_bar = vec
  vec[1,1] <- runif(1,-.75,.75)
  vec[1,2] <- runif(1,-1,1)
  accept = 0
  mu = c(1,2)
  # loop
  for(i in 2:N){
    # render z
    z = c(runif(1,-1,1),runif(1,-1,1))

    # walk
    vecprime = z + 2*mu - vec[i-1,]
    runif(1) -> u
   min(P(vecprime)/P(vec[i-1,]),1) -> alpha
    if(u < alpha){
      vec[i,] = vecprime
      accept = accept+1
      #x1 -> x0
    } else {
      vec[i,] = vec[i-1,]
    }
    vec_bar[i,] <- mean(vec[1:i,])
  }
  # burn
  vec = vec[(nburn+1):N,]
  vec_bar = vec_bar[(nburn+1):N]
  # return sample vector
  list(vec = vec, means = vec_bar,ratio = accept/N)
}
```

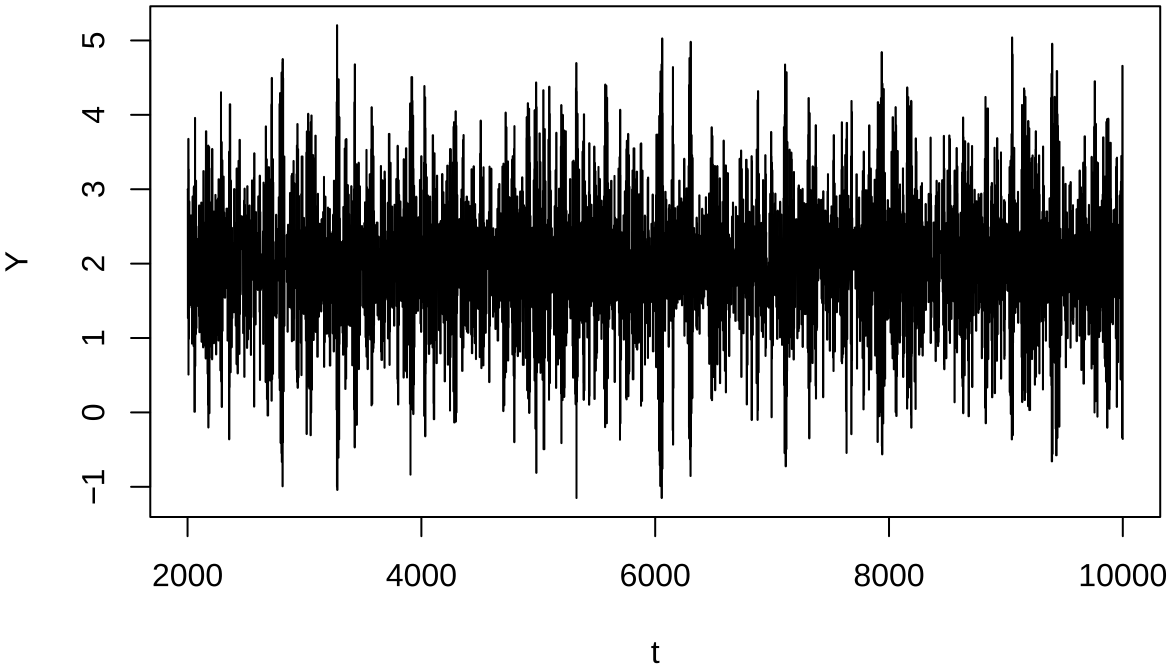
```
WALK.BVU4(10000,2000) -> chain
X = chain$vec[,1]
Y = chain$vec[,2]
plot(X,Y)
```



```
t = 2001:10000
plot(t,X,type = 'l')
```



```
plot(t,Y,type = "l")
```



```
sprintf("Acceptance rate is for system 4 is %f",chain$ratio)
```

```
## [1] "Acceptance rate is for system 4 is 0.462700"
```