Computational Statistics Computer Lab 4 (Group 7)

Qinyuan Qi(qinqi464) Satya Sai Naga Jaya Koushik Pilla (satpi345)

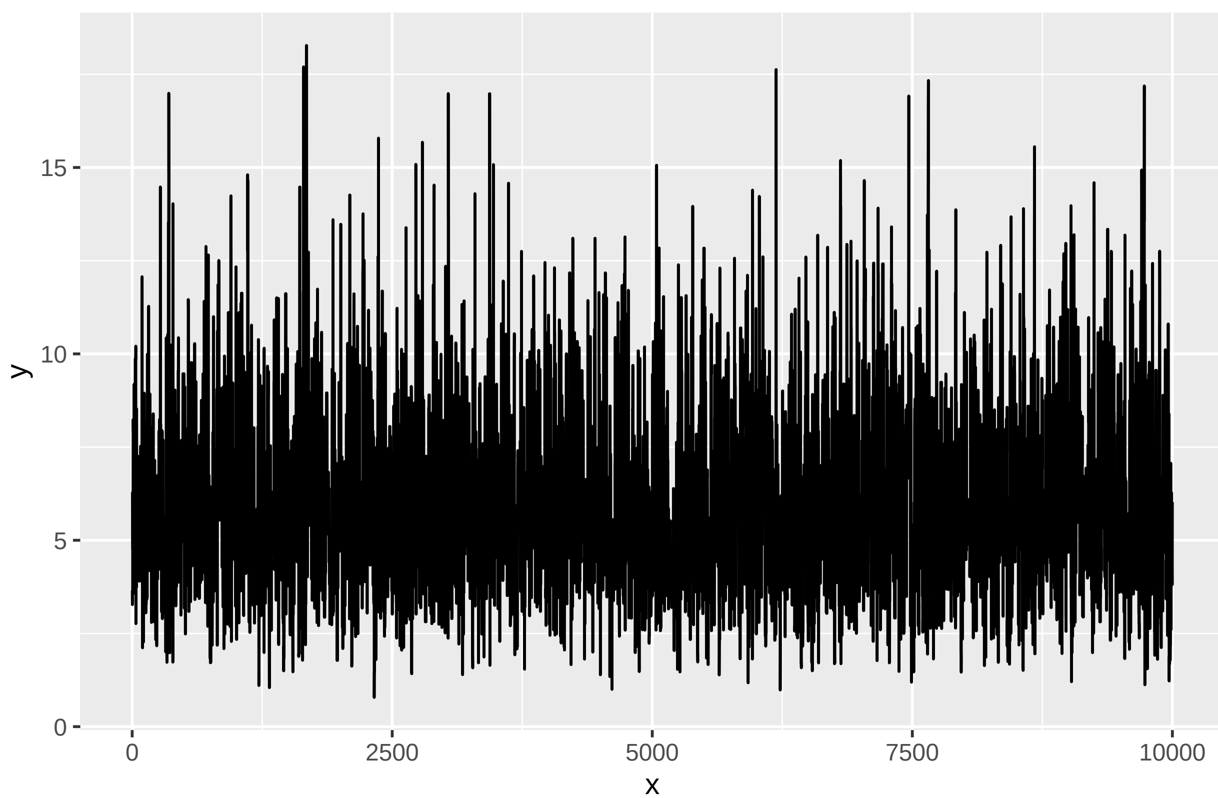
2023-11-28

Question 1: Computations with Metropolis–Hastings (Solved by Qinyuan Qi)

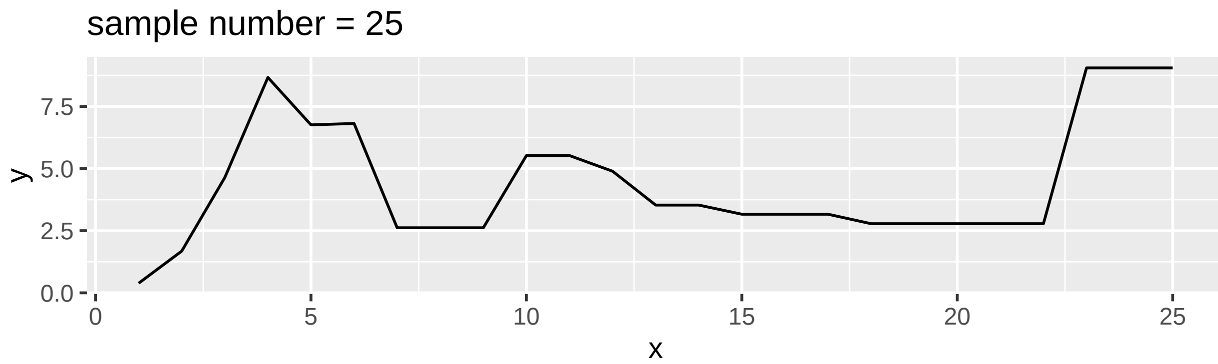
Answer:

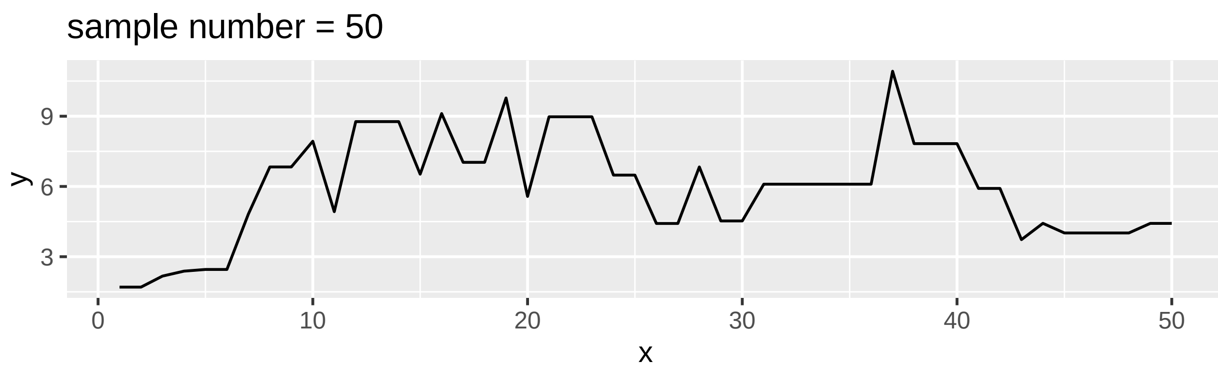
(1.a)

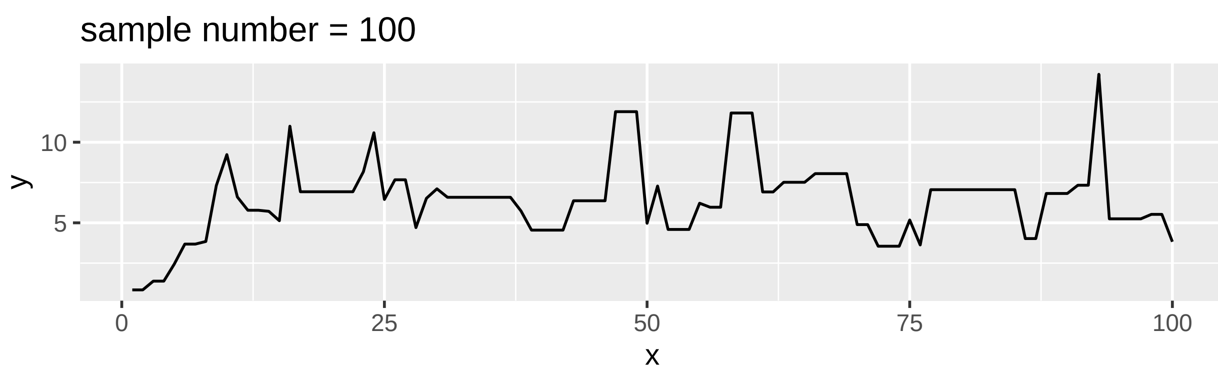
The following is generated plot. And according to the result, the convergence of the chain seems not converge to a fixed value. But fluctuates up and down around value 6. Mean is 6.009954.



If we set sample_number to 25,50 and 100 respectively , we will find the burn-in period around 10.

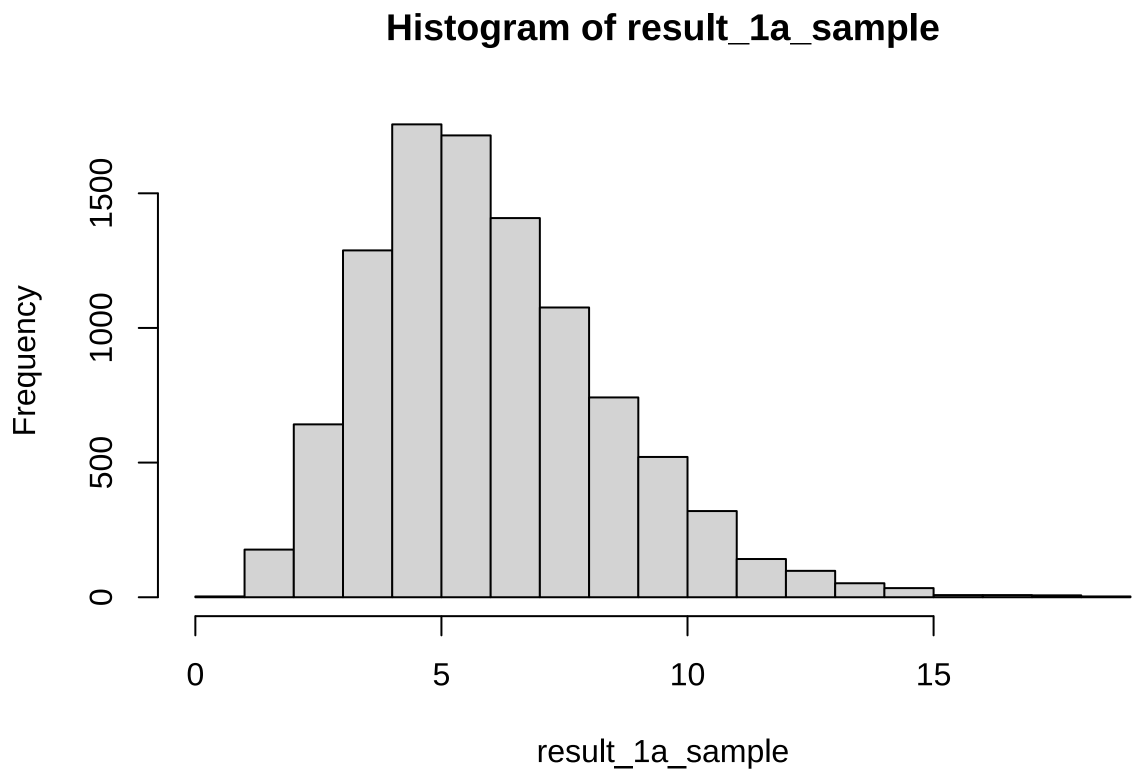






Acceptance rate of 10000 sample_number is 0.445.

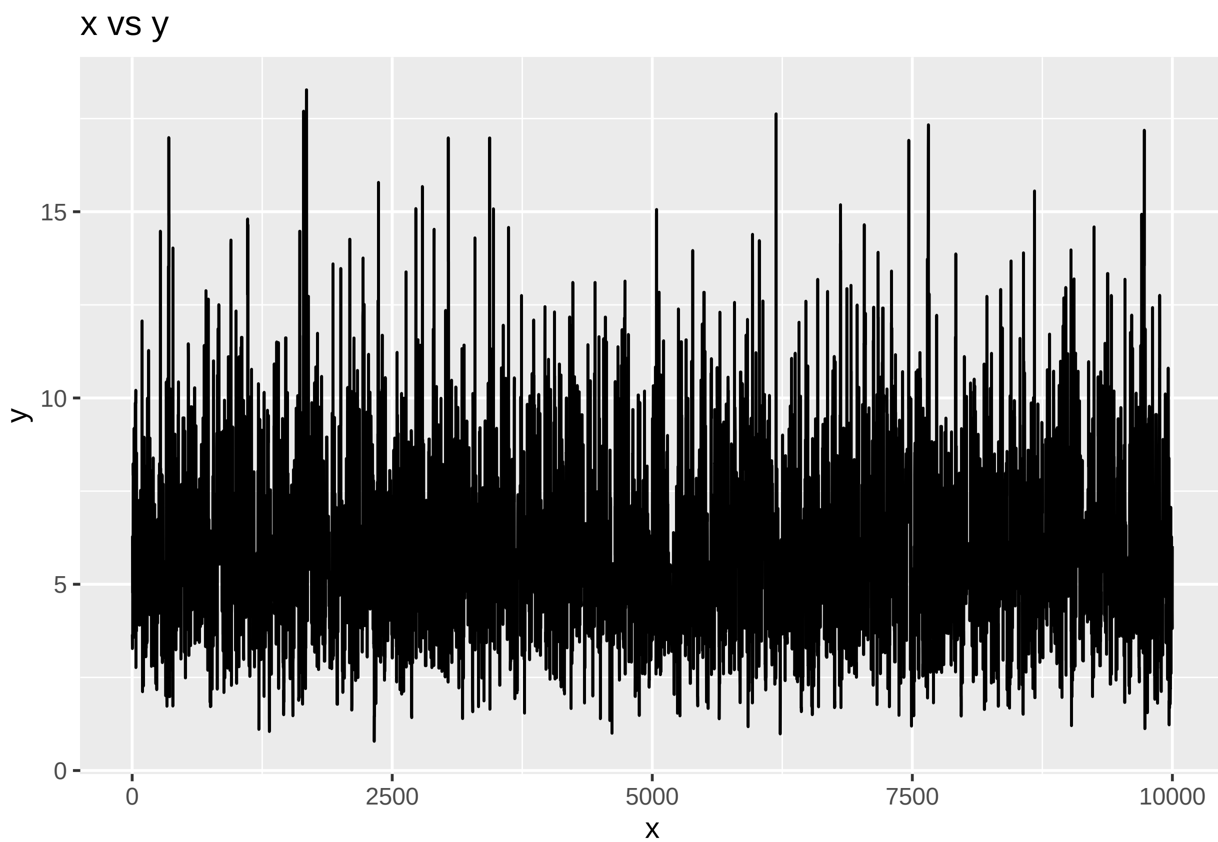
The hist plot as follows.



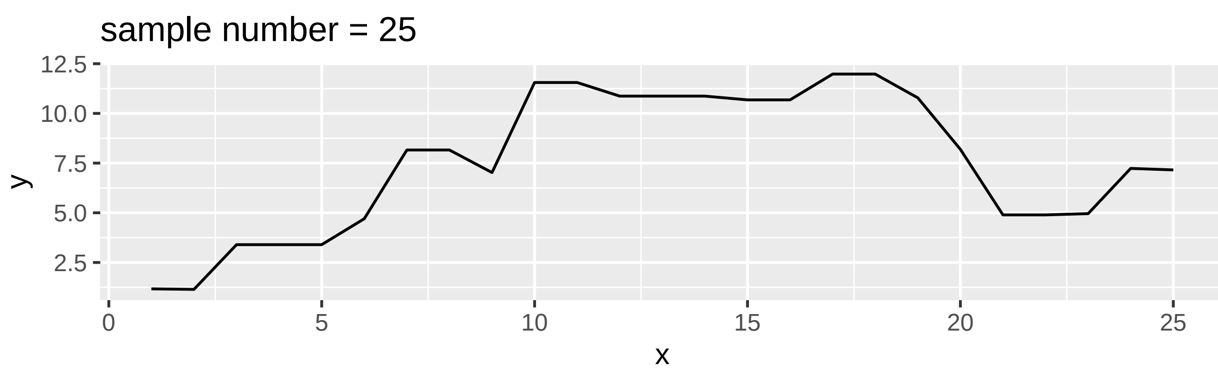
(1.b)

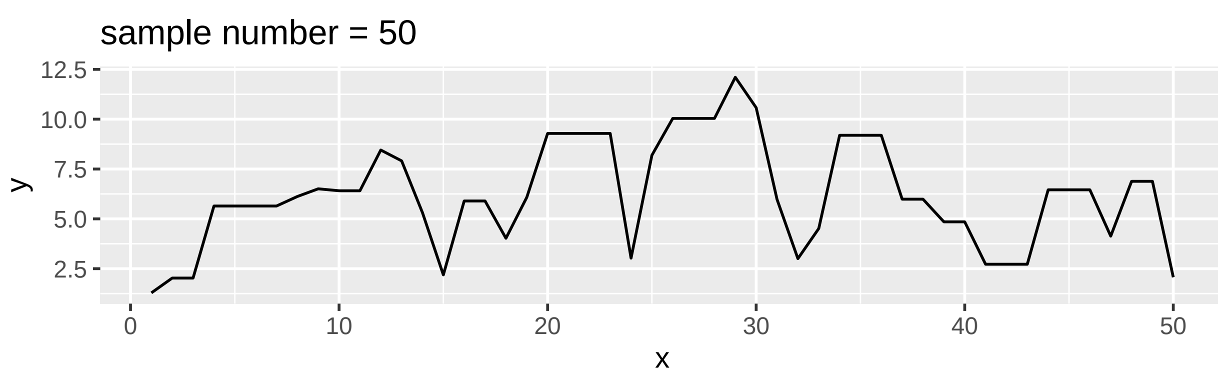
We use chi-square distribution $\chi^2(\lfloor x+1 \rfloor)$ as the proposal distribution.

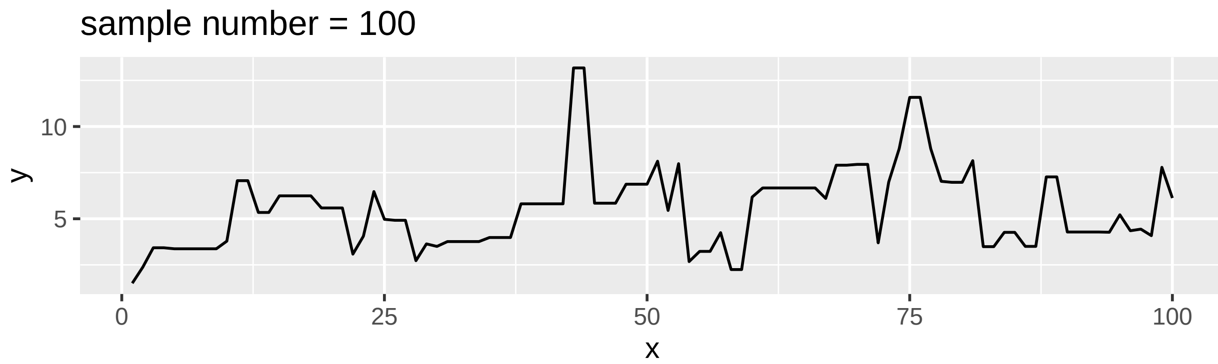
The following is generated plot. And according to the result, the convergence of the chain seems not converge to a fixed value. But fluctuates up and down around value 6. Mean is 6.123379.



If we set sample_number to 25,50 and 100 respectively , we will find the burn-in period around 8.

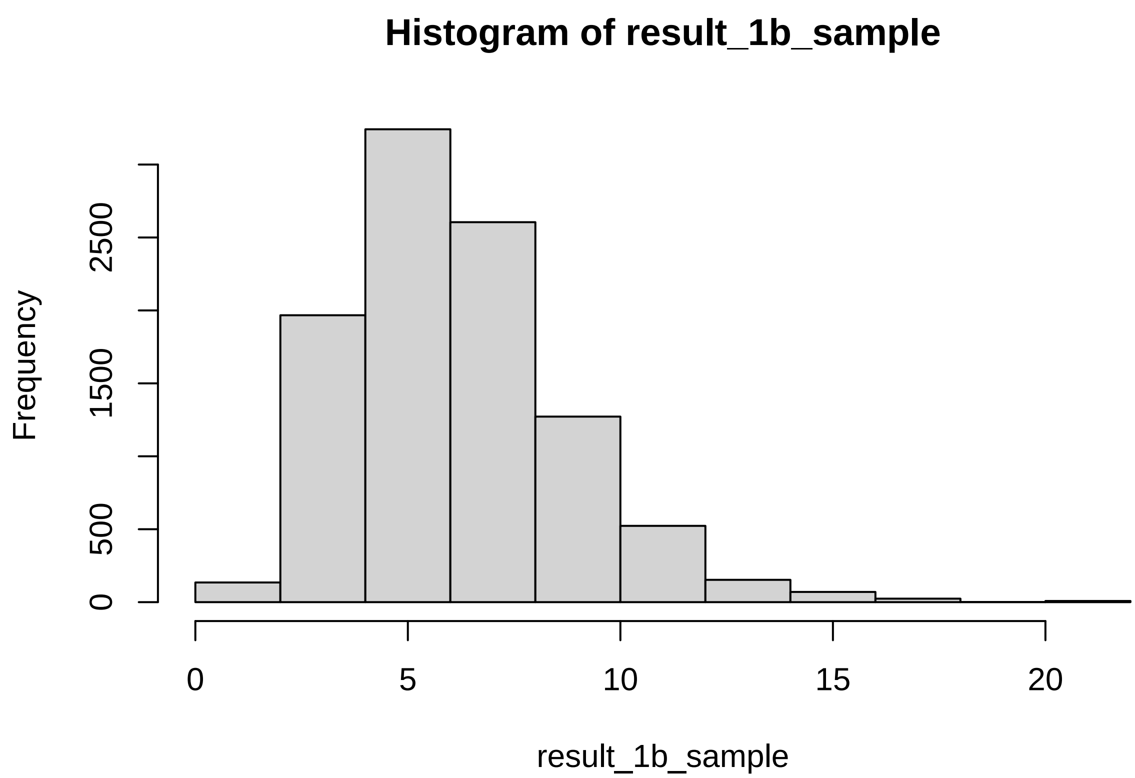






Acceptance rate of 10000 sample_number is 0.594.

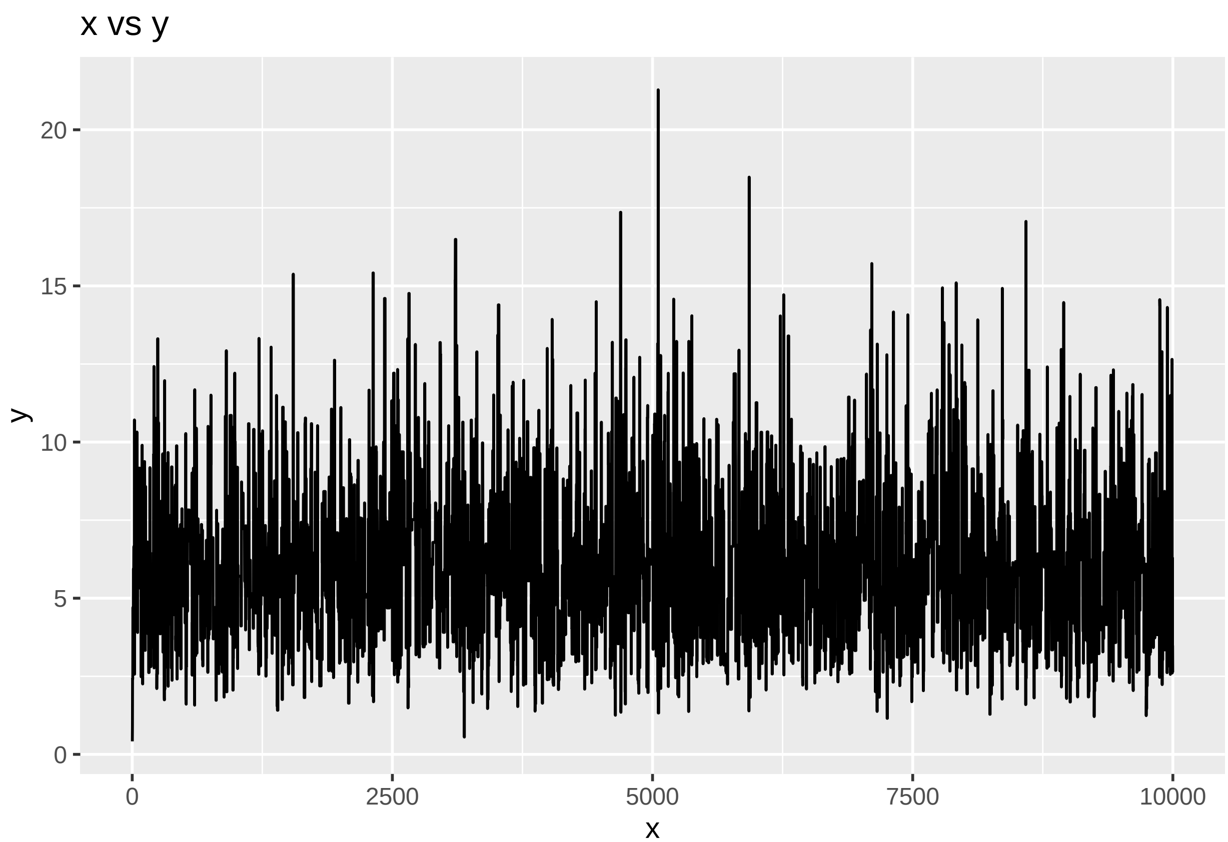
The hist plot as follows.



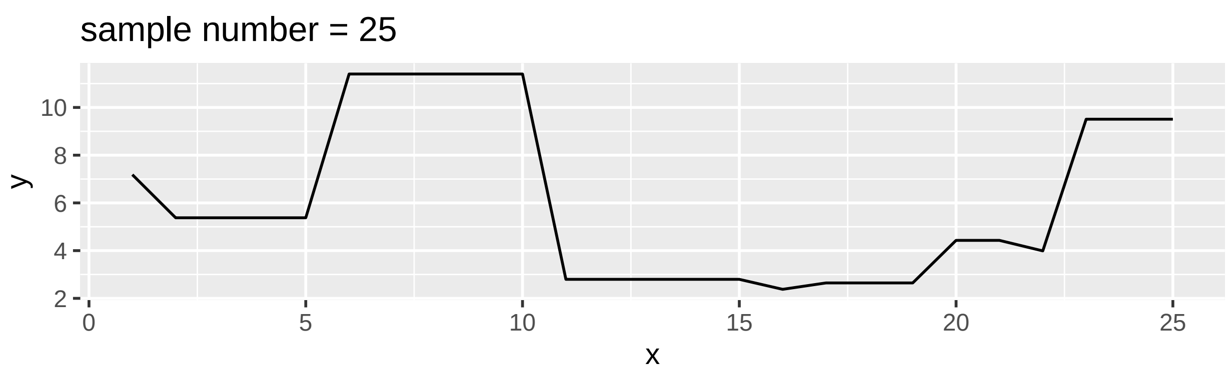
(1.c)

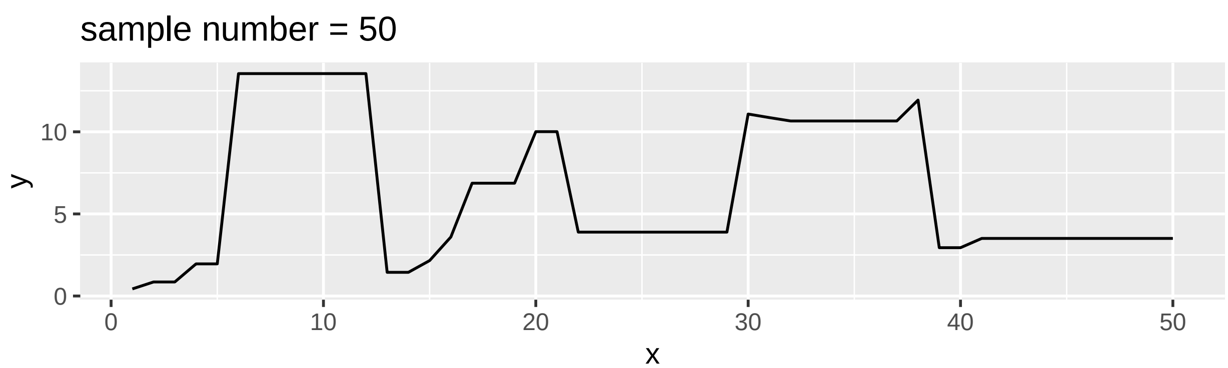
We use LN(X_{t},2) as the proposal distribution.

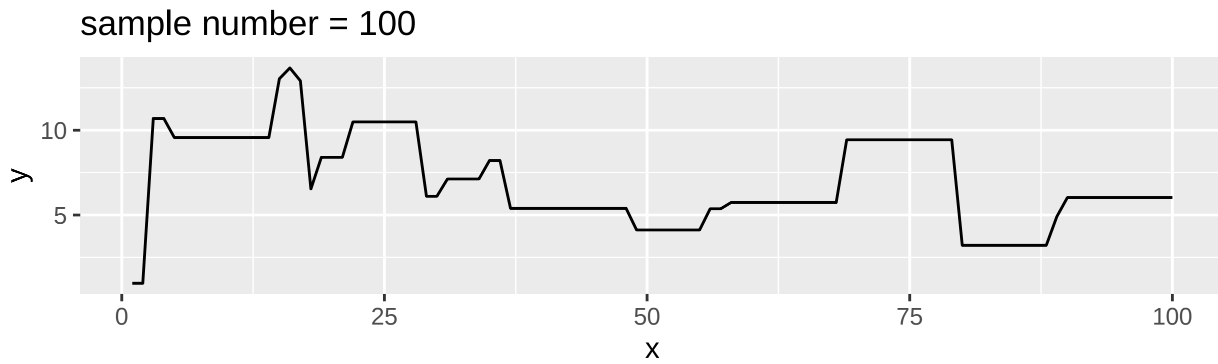
The following is generated plot. And according to the result, the convergence of the chain seems not converge to a fixed value. But fluctuates up and down around value 6. And mean is 6.049095.



If we set sample_number to 25,50 and 100 respectively , we will find the burn-in period around 5.

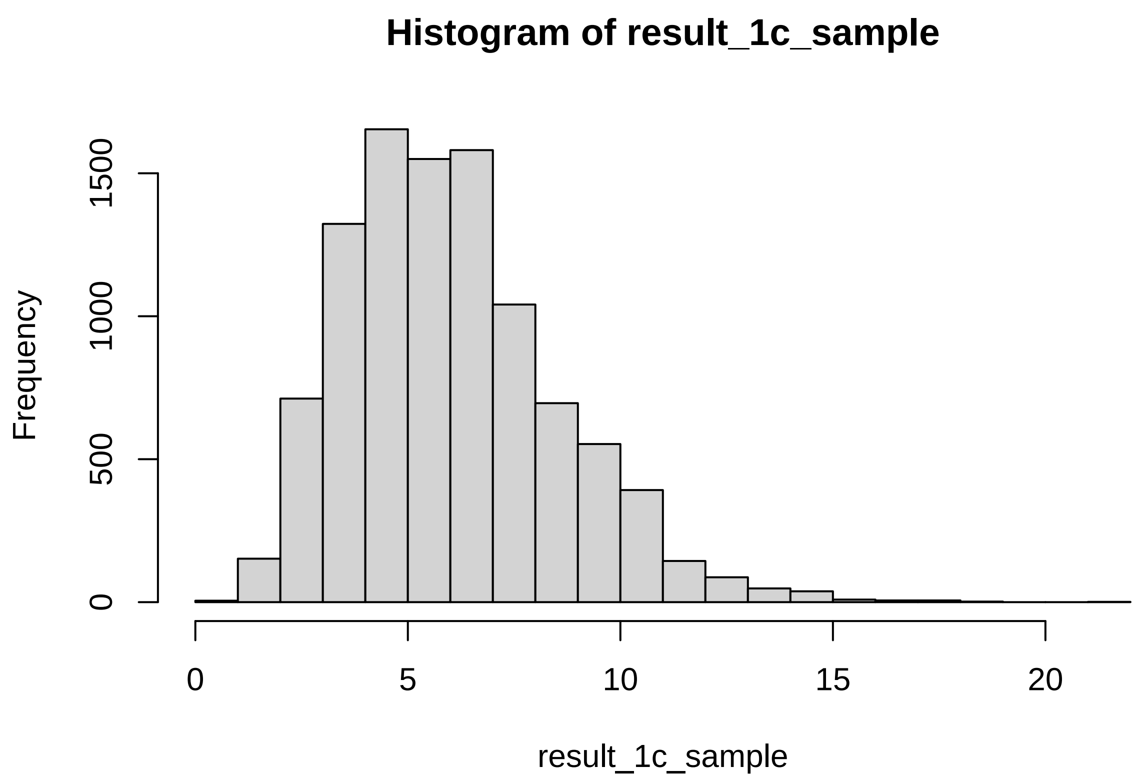






Acceptance rate of 10000 sample number is 0.2447.

The hist plot as follows.



(1.d)

According to the result showed above, we have the table as follows.

	Mean	Acceptance rate	Burnin
1.a	6.009954	0.445	10
1.b	6.123379	0.594	8
1.c	6.049095	0.2447	5

All three methods have the same mean, and acceptance rate and burnin period will change according to the parameters given.

(1.e)

As described in 1.d, we can think this is the mean of 3 data set, which is 6.009954, 6.123379 and 6.049095 respectively.

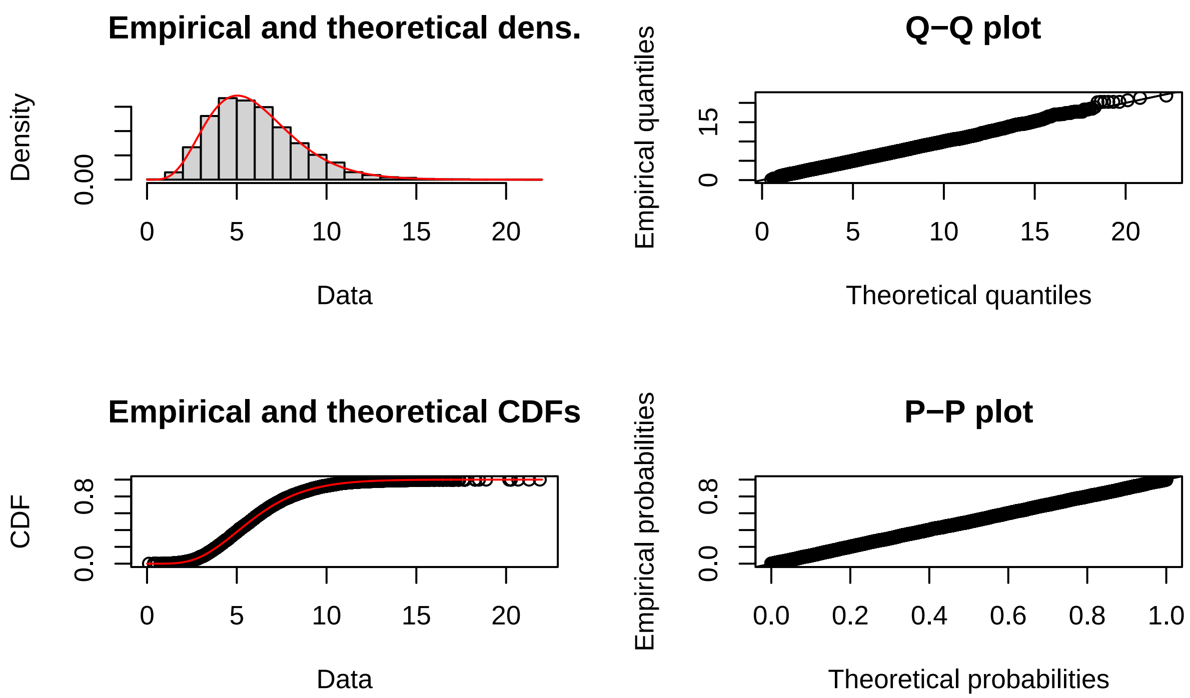
(1.f)

Using the following code, we fit the gamma distribution and we get the gamma distribution parameters as: shape=6.0551684 and rate = 0.9990455.

```
########################### [ 1.f ] ##################################################
all_samples <- c(result_1a_sample, result_1b_sample, result_1c_sample)
fit.gamma <- fitdist(all_samples, distr = "gamma", method = "mle")
summary(fit.gamma)
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters :
##       estimate  Std. Error
## shape 5.915793 0.047003003
## rate  0.977479 0.008105587
## Loglikelihood:  -68159.48   AIC:  136323   BIC:  136339.6
## Correlation matrix:
##          shape     rate
## shape 1.000000 0.958153
## rate  0.958153 1.000000
```

```
plot(fit.gamma)
```



Question 2: Gibbs sampling (Solved by Satya Sai Naga Jaya Koushik Pilla)

Answer:

(2.b)

Since w = 1.999, and $x_1^2 + 1.999 * x_1 * x_2 + x_2^2 < 1$

Using the quadratic formula, we apply this on

$x_1^2 + 1.999 * x_1 * x_2 + x_2^2 - 1 = 0$

we have rhe conditional distribution for x_2 given x_1 is a uniform distribution on the interval:

$(-0.9995 * x_1 - \sqrt{1 - 0.00099975 * x_1^2}, -0.9995 * x_1 + \sqrt{1 - 0.00099975 * x_1^2})$

The conditional distribution for x_1 given x_2 is a uniform distribution on the interval

$(-0.9995 * x_2 - \sqrt{1 - 0.00099975 * x_2^2}, -0.9995 * x_2 + \sqrt{1 - 0.00099975 * x_2^2})$

(2.a)(2.c)

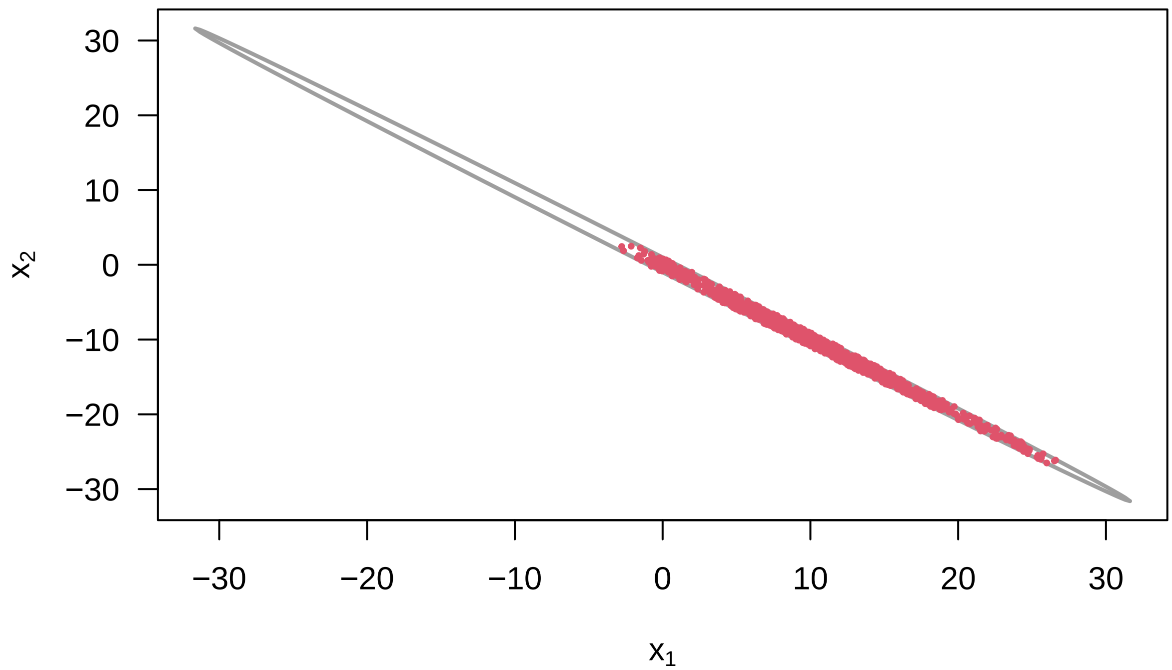
We draw the boundaries using the code provided on the course home page.

And plot the random vectors to the graph as follows.code in appendix.

We also print out the probability that x1 > 0.

As the output value increase to 1 , so we can not decide the true result for this probability.

```
## The probability that x1 > 0 is 0
## The probability that x1 > 0 is 0
## The probability that x1 > 0 is 0
## The probability that x1 > 0 is 0.25
## The probability that x1 > 0 is 0.2
## The probability that x1 > 0 is 0.1666667
## The probability that x1 > 0 is 0.1428571
## The probability that x1 > 0 is 0.125
## The probability that x1 > 0 is 0.2222222
## The probability that x1 > 0 is 0.3
## The probability that x1 > 0 is 0.3636364
## The probability that x1 > 0 is 0.4166667
## The probability that x1 > 0 is 0.4615385
## The probability that x1 > 0 is 0.5
## The probability that x1 > 0 is 0.5333333
## The probability that x1 > 0 is 0.5625
## The probability that x1 > 0 is 0.5882353
## The probability that x1 > 0 is 0.6111111
## The probability that x1 > 0 is 0.6315789
```



(2.d)

As we can see from the 2 plots, the ellipse for w = 1.999 is more flat than the ellipse for w = 1.8.

(2.e)

We need to transform the variable X and convert to $U = (U1, U2) = (X_1 - X2, X1 + X2)$

And since U is still a uniform distribution, we can use the same method as in 2c

We have $U1 = X_2 - X_2, U2 = X_1 + X_2$

so we have $X_1 = (U1 + U2)/2, X_2 = (U2 - U1)/2$

We substitute this into the original function, and after some simplification, we have

$(2 + w) * u_2^2 + (2 - w) * u_1^2 - 4 = 0$

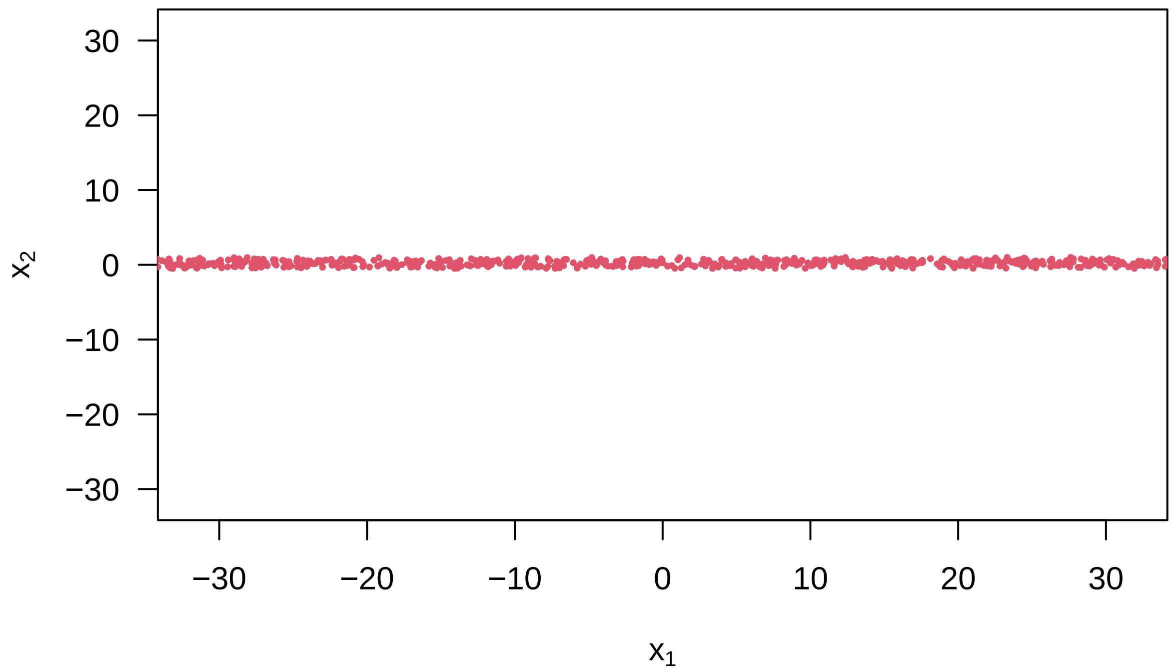
Calculate the boundary of the ellipse, we have

$(-\sqrt{(4 - (2 - w) * u_1^2)/(2 + w)}, \sqrt{(4 - (2 - w) * u_1^2)/(2 + w)})$

and

$(-\sqrt{(4 - (2 + w) * u_2^2)/(2 - w)}, \sqrt{(4 - (2 + w) * u_2^2)/(2 - w)})$

The probability that x1 > 0 is 0.506.


30
20
10
0
−10
−20
−30
x_2
−30
−20
−10
0
10
20
30
x_1

Appendix: Code for this report

```
######################### Init code for question 1 ############################
rm(list = ls())
library(ggplot2)
library(fitdistrplus)
set.seed(12345)
######################### [ 1.a1] ##############################################
fx <- function(x) {
  return(ifelse(x <= 0, 0, x^5 * exp(-x)))
}

metropolis_hastings_log_normal_1a <- function(n) {
  samples <- rep(0, n)

  # sample from a log-normal distribution
  xt <- rlnorm(1, 0, 1)
  samples[1] <- xt

  # set some initial values
  count <- 1
  accepted_count <- 0

  while (count <= n) {
    # sample a candidate from log normal distribution
    x_star <- rlnorm(1, log(xt), 1)

    # calc a ratio
    mh_ratio <- (fx(x_star) * dlnorm(xt, log(x_star), 1)) /
      (fx(xt) * dlnorm(x_star, log(xt), 1))


    # accept according to the ratio
    if (mh_ratio > 1) {
      xt_plus <- x_star
      accepted_count <- accepted_count + 1
    } else {
      u <- runif(1)
      if (u < mh_ratio) {
        xt_plus <- x_star
        accepted_count <- accepted_count + 1
      } else {
        xt_plus <- xt
      }
    }

    samples[count] <- xt_plus
    count <- count + 1
    xt <- xt_plus
  }
  return(list(random_variable = samples, acceptance_rate = accepted_count / n))
}

sample_number <- 10000
```

```r
result_1a <- metropolis_hastings_log_normal_1a(sample_number)

data_set_1a <- data.frame(x = 1:sample_number, y = result_1a$random_variable)

result_1a_sample <- result_1a$random_variable

ggplot(data = data_set_1a, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("") +
  xlab("x") +
  ylab("y")

########################## [ 1.a2] #############################################

sample_number <- 25
result_1a2 <- metropolis_hastings_log_normal_1a(sample_number)

data_set_1a2 <- data.frame(x = 1:sample_number, y = result_1a2$random_variable)

ggplot(data = data_set_1a2, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("sample number = 25") +
  xlab("x") +
  ylab("y")

########################## [ 1.a3] #############################################
sample_number <- 50
result_1a3 <- metropolis_hastings_log_normal_1a(sample_number)

data_set_1a3 <- data.frame(x = 1:sample_number, y = result_1a3$random_variable)

ggplot(data = data_set_1a3, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("sample number = 50") +
  xlab("x") +
  ylab("y")
########################## [ 1.a4] #############################################
sample_number <- 100
result_1a4 <- metropolis_hastings_log_normal_1a(sample_number)

data_set_1a4 <- data.frame(x = 1:sample_number, y = result_1a4$random_variable)

ggplot(data = data_set_1a4, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("sample number = 100") +
  xlab("x") +
  ylab("y")
########################## [ 1.a5] #############################################
# The acceptance rate is 0.445
result_1a_acceptance_rate <- result_1a$acceptance_rate
mean_1a <- mean(result_1a4$random_variable)
########################## [ 1.a6] #############################################
hist(result_1a_sample)
```

```r
########################## [ 1.b1 ] ############################################

metropolis_hastings_chi_square_1b <- function(n) {
  samples <- rep(0, n)

  # sample from a chi-square distribution
  xt <- rchisq(n = 1, df = 1)
  samples[1] <- xt

  # set some initial values
  count <- 1
  accepted_count <- 0

  while (count <= n) {
    # sample a candidate from log normal distribution
    x_star <- rchisq(n = 1, df = floor(xt + 1))

    # calc a ratio

    mh_ratio <- (fx(x_star) * dchisq(xt, df = floor(x_star + 1)) /
                 (fx(xt) * dchisq(x_star, floor(xt + 1))))

    # accept according to the ratio
    if (mh_ratio > 1) {
      xt_plus <- x_star
      accepted_count <- accepted_count + 1
    } else {
      u <- runif(1)
      if (u < mh_ratio) {
        xt_plus <- x_star
        accepted_count <- accepted_count + 1
      } else {
        xt_plus <- xt
      }
    }

    samples[count] <- xt_plus
    count <- count + 1
    xt <- xt_plus
  }
  return(list(random_variable = samples, acceptance_rate = accepted_count / n))
}

sample_number <- 10000
result_1b <- metropolis_hastings_chi_square_1b(sample_number)

data_set_1b <- data.frame(x = 1:sample_number, y = result_1a$random_variable)

result_1b_sample <- result_1b$random_variable

ggplot(data = data_set_1b, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("x vs y") +
```

```r
  xlab("x") +
  ylab("y")
########################### [ 1.b2] ###############################################

sample_number <- 25
result_1b2 <- metropolis_hastings_chi_square_1b(sample_number)

data_set_1b2 <- data.frame(x = 1:sample_number, y = result_1b2$random_variable)

ggplot(data = data_set_1b2, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("sample number = 25") +
  xlab("x") +
  ylab("y")

########################### [ 1.b3] ###############################################
sample_number <- 50
result_1b3 <- metropolis_hastings_chi_square_1b(sample_number)

data_set_1b3 <- data.frame(x = 1:sample_number, y = result_1b3$random_variable)

ggplot(data = data_set_1b3, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("sample number = 50") +
  xlab("x") +
  ylab("y")
########################### [ 1.b4] ###############################################
sample_number <- 100
result_1b4 <- metropolis_hastings_chi_square_1b(sample_number)

data_set_1b4 <- data.frame(x = 1:sample_number, y = result_1b4$random_variable)

ggplot(data = data_set_1b4, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("sample number = 100") +
  xlab("x") +
  ylab("y")
########################### [ 1.a5] ###############################################
result_1b_acceptance_rate <- result_1b$acceptance_rate
########################### [ 1.a6] ###############################################
hist(result_1b_sample)
########################### [ 1.c1 ] ###############################################
metropolis_hastings_log_normal_1c <- function(n) {
  samples <- rep(0, n)

  # sample from a log-normal distribution
  xt <- rlnorm(1, 0, 2)
  samples[1] <- xt

  # set some initial values
  count <- 1
  accepted_count <- 0
```

```r
  while (count <= n) {
    # sample a candidate from log normal distribution
    x_star <- rlnorm(1, log(xt), 2)

    # calc a ratio
    mh_ratio <- (fx(x_star) * dlnorm(xt, log(x_star), 2)) /
                (fx(xt) * dlnorm(x_star, log(xt), 2))

    # accept according to the ratio
    if (mh_ratio > 1) {
      xt_plus <- x_star
      accepted_count <- accepted_count + 1
    } else {
      u <- runif(1)
      if (u < mh_ratio) {
        xt_plus <- x_star
        accepted_count <- accepted_count + 1
      } else {
        xt_plus <- xt
      }
    }

    samples[count] <- xt_plus
    count <- count + 1
    xt <- xt_plus
  }
  return(list(random_variable = samples, acceptance_rate = accepted_count / n))
}

sample_number <- 10000
result_1c <- metropolis_hastings_log_normal_1c(sample_number)

data_set_1c <- data.frame(x = 1:sample_number, y = result_1c$random_variable)

result_1c_sample <- result_1c$random_variable

ggplot(data = data_set_1c, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("x vs y") +
  xlab("x") +
  ylab("y")
########################## [ 1.c2 ] #############################################
sample_number <- 25
result_1c2 <- metropolis_hastings_log_normal_1c(sample_number)

data_set_1c2 <- data.frame(x = 1:sample_number, y = result_1c2$random_variable)

ggplot(data = data_set_1c2, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("sample number = 25") +
  xlab("x") +
  ylab("y")
########################## [ 1.c3 ] #############################################
```

```r
sample_number <- 50
result_1c3 <- metropolis_hastings_log_normal_1c(sample_number)

data_set_1c3 <- data.frame(x = 1:sample_number, y = result_1c3$random_variable)

ggplot(data = data_set_1c3, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("sample number = 50") +
  xlab("x") +
  ylab("y")
########################## [ 1.c4 ] #############################################
sample_number <- 100
result_1c4 <- metropolis_hastings_log_normal_1c(sample_number)

data_set_1c4 <- data.frame(x = 1:sample_number, y = result_1c4$random_variable)

ggplot(data = data_set_1c4, mapping = aes(x = x, y = y)) +
  geom_line() +
  ggtitle("sample number = 100") +
  xlab("x") +
  ylab("y")
########################## [ 1.c5 ] #############################################
#
result_1c_acceptance_rate <- result_1c$acceptance_rate
########################## [ 1.c6 ] #############################################
hist(result_1c_sample)
########################## [ 1.f ] #############################################
all_samples <- c(result_1a_sample, result_1b_sample, result_1c_sample)
fit.gamma <- fitdist(all_samples, distr = "gamma", method = "mle")
summary(fit.gamma)
plot(fit.gamma)
########################## Init code for question 1 ############################
rm(list = ls())
########################## [ 2.c ] #############################################

w  <- 1.999

# Draw the boundary of the ellipse
# a range of x1-values, where the term below the root is non-negative
xv <- seq(-1, 1, by=0.01) * 1/sqrt(1-w^2/4)
plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1)

# ellipse
lines(xv, -(w/2)*xv-sqrt(1-(1-w^2/4)*xv^2), lwd=2, col=8)
lines(xv, -(w/2)*xv+sqrt(1-(1-w^2/4)*xv^2), lwd=2, col=8)

gibbs_sampling_2c <- function(n, x0) {
    samples <- matrix(nrow = n, ncol = 2)
    x1 <- x0[1]
    x2 <- x0[2]

    count <- 1
    while (count <= n) {
```

```r
        # Sample x1 from the conditional distribution for x1 given x2
        x1 <- runif(1, -0.9995 * x2 - sqrt(1-0.00099975 * x2^2), -0.9995 * x2 + sqrt(1-0.00099975 * x2^

        # Sample x2 from the conditional distribution for x2 given x1
        x2 <- runif(1, -0.9995 * x1 - sqrt(1-0.00099975 * x1^2), -0.9995 * x1 + sqrt(1-0.00099975 * x1^

        samples[count,] <- c(x1, x2)

        # Calculate p(x1 > 0)
        prob_value <- sum(samples[1:count, 1] > 0) /
              length(samples[1:count, 1])

        if (count < 20){
          cat("The probability that x1 > 0 is", prob_value, "\n")
        }

        count <- count + 1
    }

    return(samples)
}

x0 <- c(0, 0)
n <- 1000
gibbs_sampling_result_2c <- gibbs_sampling_2c(n, x0)

# plot the sample point on the eclipse
points(gibbs_sampling_result_2c[,1], gibbs_sampling_result_2c[,2], col=2, pch=20, cex=0.5)
########################## [ 2.e ] #############################################
w  <- 1.999

# Draw the boundary of the ellipse
# a range of x1-values, where the term below the root is non-negative

xv <- seq(-1, 1, by=0.01) * 1/sqrt(1-w^2/4)
plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1)

gibbs_sampling_2e <- function(n, u0) {
    samples <- matrix(nrow = n, ncol = 2)
    u1 <- u0[1]
    u2 <- u0[2]

    count <- 1
    while (count <= n) {
        # Sample x1 from the conditional distribution for x1 given x2
        u1_new <- runif(1,
                        -1 * sqrt((4 - 3.999 * u2^2) / 0.001),
                        sqrt((4 - 3.999 * u2^2) / 0.001)
                        )

        # Sample x2 from the conditional distribution for x2 given x1
        u2_new <- runif(1,
                        -1 * sqrt((4 - 0.001 * u1^2)) / 3.999,
```

```r
                        sqrt((4 - 0.001 * u1^2) / 3.999)
                        )

        samples[count,] <- c(u1_new, u2_new)

        u1 <- u1_new
        u2 <- u2_new

        count <- count + 1
    }

    return(samples)
}

u0 <- c(0, 0)
n <- 1000
gibbs_sampling_result_2e <- gibbs_sampling_2e(n, u0)

# Calculate p(x1 > 0) = p((u1 + u2) / 2 > 0)
prob_value <- sum((gibbs_sampling_result_2e[1:n, 1] + gibbs_sampling_result_2e[1:n, 2])/2 > 0) /
              length(gibbs_sampling_result_2e[1:n, 1])

#cat("The probability that x1 > 0 is", prob_value, "\n")

# plot the sample point on the eclipse
points(gibbs_sampling_result_2e[,1], gibbs_sampling_result_2e[,2], col=2, pch=20, cex=0.5)
```