Computational Statistics Lab 4

Simge Cinar & Ronald Yamashita

2023-11-24

Question 1: Computations with Metropolis Hastings

Consider a random variable X with the following probability density function:

$$f(x) \propto x^5 e^{-x}, x > 0$$

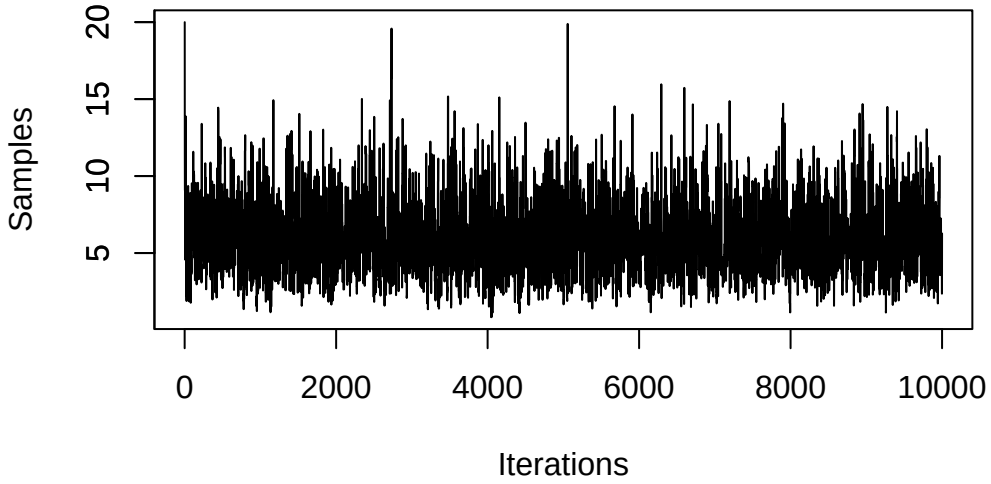
The distribution is known up to some constant of proportionality. If you are interested (NOT part of the Lab) this constant can be found by applying integration by parts multiple times and equals 120.

part a)

Question: Use the Metropolis–Hastings algorithm to generate 10,000 samples from this distribution by using a log–normal LN(Xt,1) proposal distribution; take some starting point. Plot the chain you obtained with iterations on the horizontal axis. What can you guess about the convergence of the chain? If there is a burn–in period, what can be the size of this period? What is the acceptance rate? Plot a histogram of the sample.

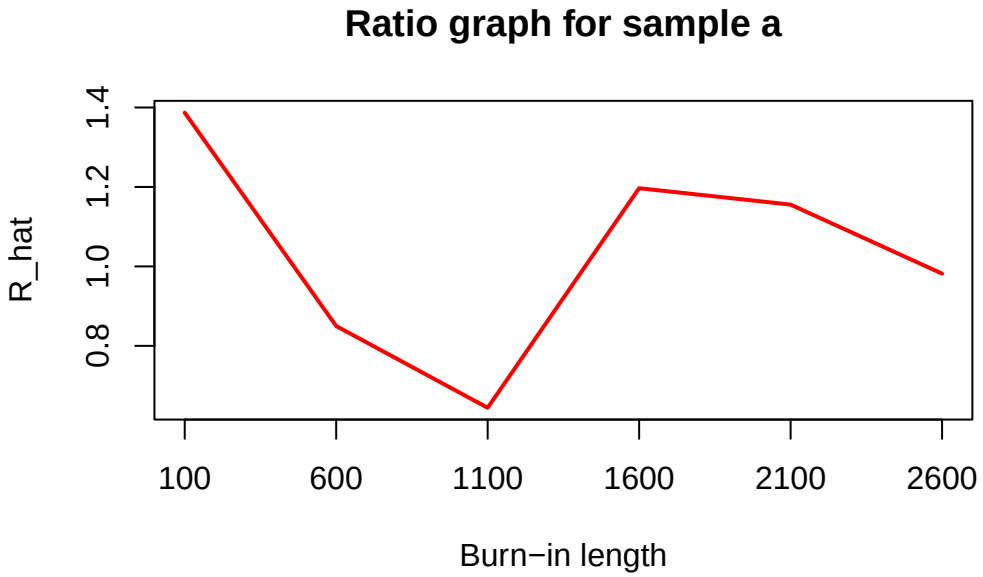
Answer:

Plot of the chain can be observed below. The chain quickly moved away its starting value, it is good mixing. Also the value density is between 5 and 10 which is reasonable because the expectation of gamma(6,1) is 6.

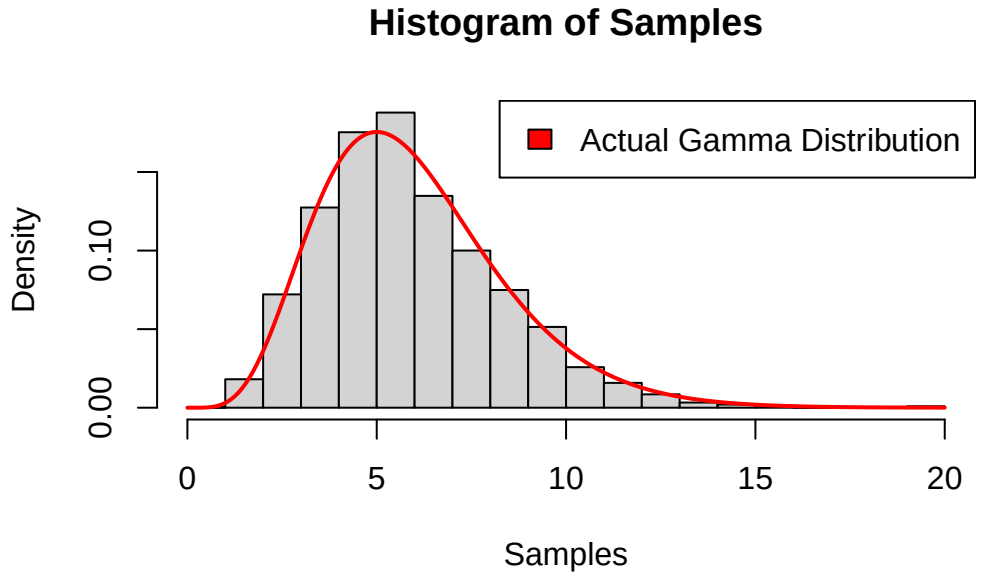


By looking at the chain plot, I would say we don't need a burn-in period because the chain moves quickly away from its starting value. To be ensure about it, the ratio calculation from the book (Givens and Hoeting 2013) in chapter 7.3 is implemented (page 221-222). According to the book, some authors suggest that if $\sqrt{\hat{R}} < 1.1$, the burn-in and chain length are sufficient.

The $\hat{R}$ is less than 1.1 when D is 600 hence we can choose the burn in length as 600 if we have to choose. It depends on the how we evaluate it and further investigations can be done but I don't think we need a burn-in periot. The histogram of the data fits the actual distribution quite good as well.



The histogram can be seen below and we can conclude that the generated sample looks reasonable, it fits the actual distribution.

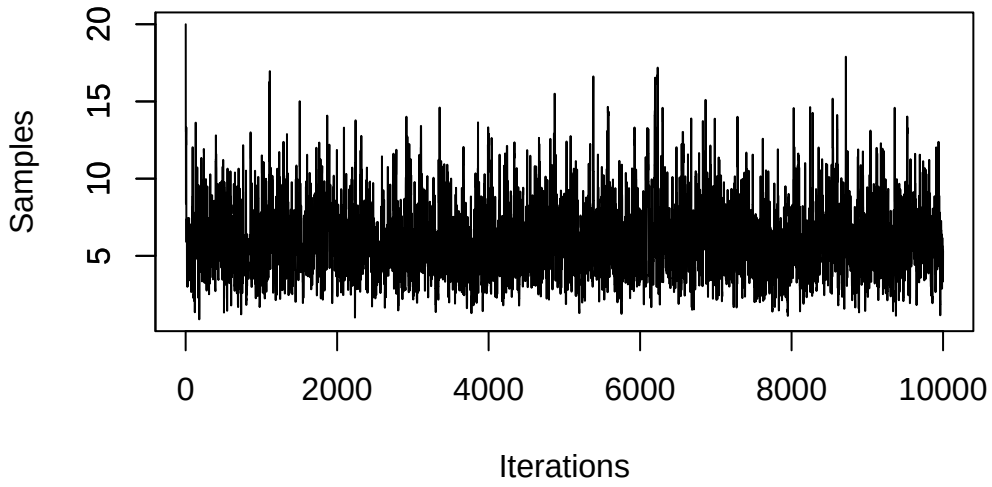



```
## Acceptance rate: 0.4476448
```

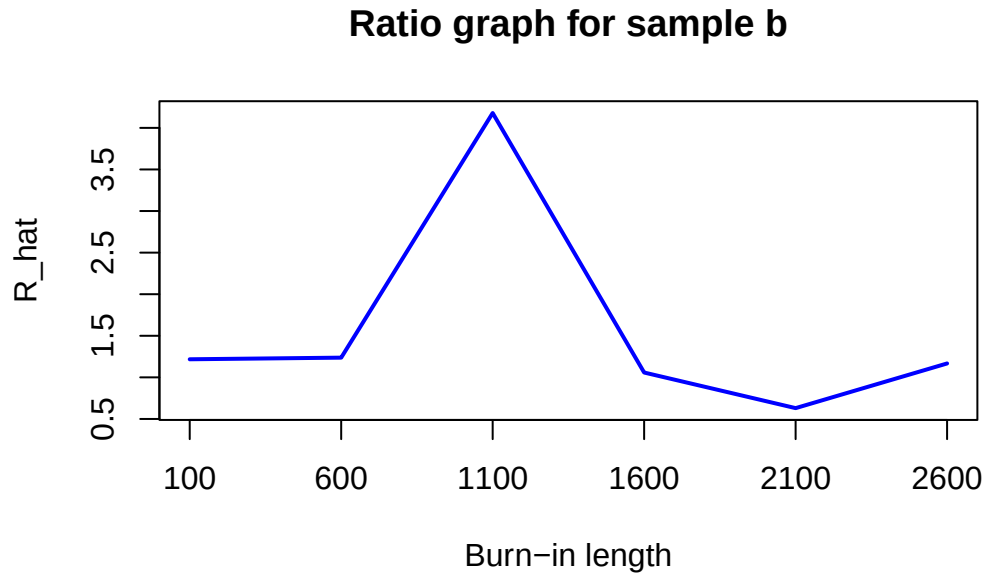
part b)

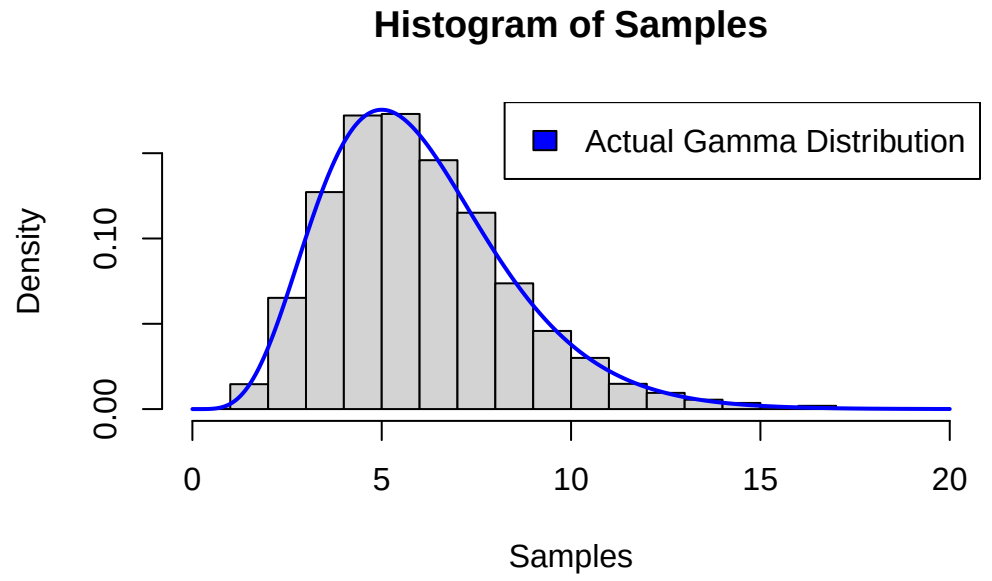
Question: Perform Part a by using the chi–square distribution $\chi^2(\lfloor Xt+1 \rfloor)$ as a proposal distribution, where $\lfloor x \rfloor$ is the floor function, meaning the integer part of x for positive x, i.e. $\lfloor 2.95 \rfloor = 2$

Answer:



The results we get is similar to part a, I think we don't need burn-in periot since the chain moves away quickly from the initial value but by looking at the plot below we could also suggest 100 as burn-in periot.



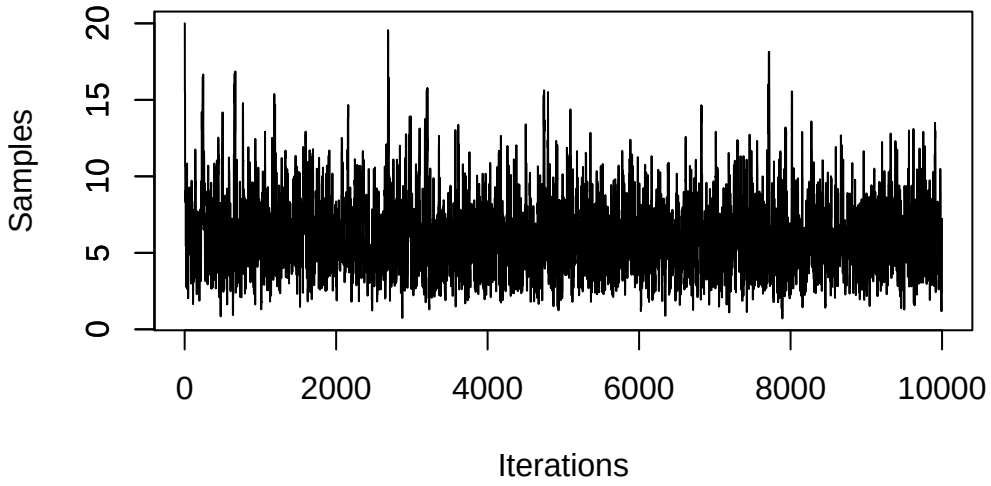


```
## Acceptance rate: 0.5929593
```

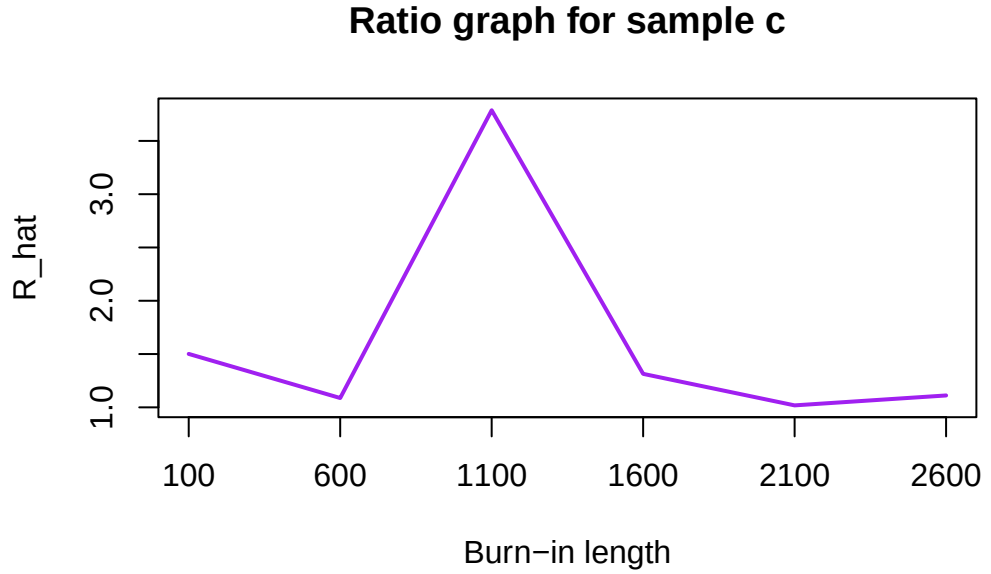
part c)

Question: Suggest another proposal distribution (can be a log normal or chi–square distribution with other parameters or another distribution) with the potential to generate a good sample. Perform part a with this distribution.

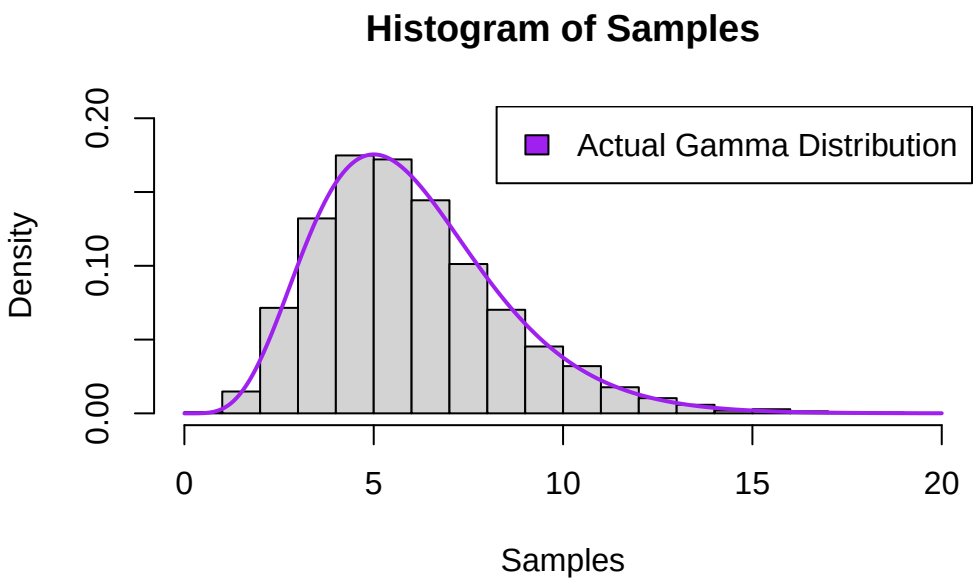
Answer: For this part, we used normal distribution with mean xt and variance 2. $N(Xt, 2)$. We used variance as 2 to capture the samples in the tails of the gamma distribution.



Again the burn-in length can be selected as 600 from the graph but I think we don't need a burn-in periot.



From the histogram it can be observed that the N(0,2) worked well as a proposal distribution. The generated samples fit the actual distribution.




```
## Acceptance rate: 0.5355536
```

part d)

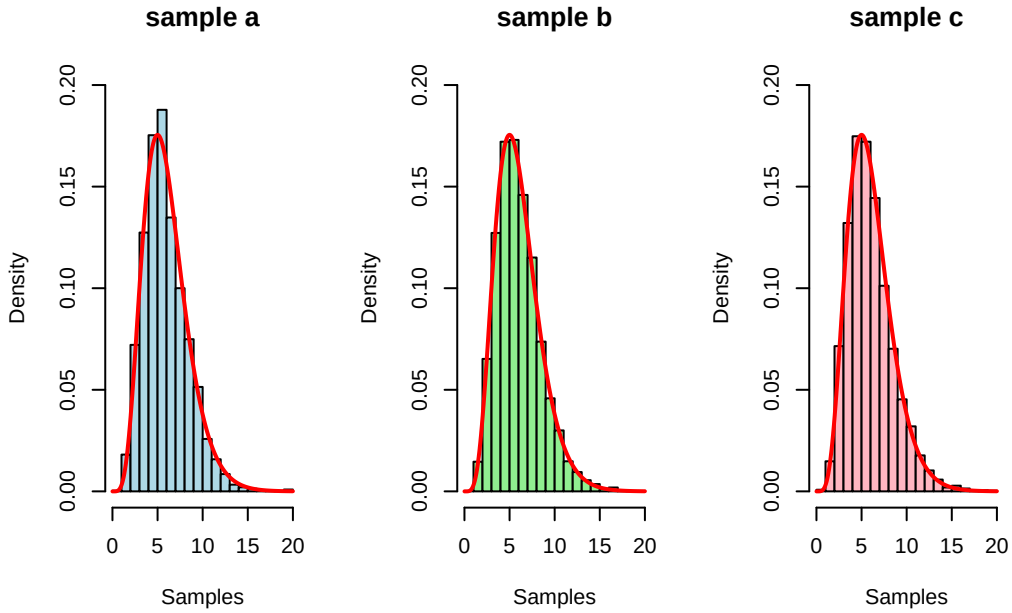
Question: Compare the results of Parts a, b, and c and make conclusions.

Answer: The acceptance table of part a,b & c can be seen below. The values are reasonable since they are not too high or low.

Table 1: Acceptance Rate Table

sample a	sample b	sample c
0.4476448	0.5929593	0.5355536

The histograms can be seen below, each proposal distribution worked well.



part e)

Question: Estimate

$$E[X] = \int_0^\infty x f(x) dx$$

using the samples from Parts a, b, and c.

Answer: The mean values from the generated samples is as follows:

Table 2: Mean & Variance Table

	Mean	Variance
sample a	5.906256	5.801632
sample b	6.021899	5.965698
sample c	5.973835	6.296328

The values are good since they are close to the actual mean which is 6. Also the variance is 6 and variance from the samples are close to 6 as well.

part f)

Question: The distribution generated is in fact a gamma distribution. Look in the literature and define the actual value of the integral. Compare it with the one you obtained.

Answer: The expectation of gamma distribution is as follows by (Wackerly, III, and Scheaffer 2002).

$$\mu = E[X] = \alpha\beta = 6 \cdot 1 = 6$$

In the target distribution, $\alpha = 6$ and $\beta = 1$ hence the value of the integral is 6.

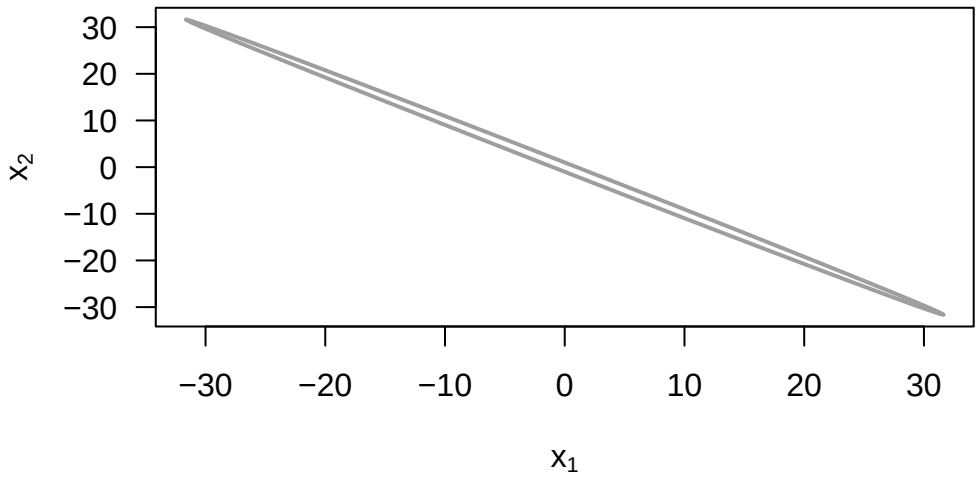
Question 2: Gibbs sampling

Let $X = (X_1, X_2)$ be a bivariate distribution with density $f(x_1, x_2) \propto 1\{x_1^2 + wx_1x_2 + x_2^2 < 1\}$ for some specific w with $|w| < 2$. X has a uniform distribution on some two-dimensional region. We consider here the case w = 1.999 (in Lecture 4, the case w = 1.8 was shown).

part a)

Question: Draw the boundaries of the region where X has a uniform distribution. You can use the code provided on the course homepage and adjust it.

Answer:



The boundaries of the region can be seen above.

part b)

Question: What is the conditional distribution of X_1 given X_2 and that of X_2 given X_1?

Answer:

$$\text{lower bound} = \frac{-b - \sqrt{\Delta}}{2a}$$

$$\text{upper bound} = \frac{-b - \sqrt{\Delta}}{2a}$$

$$\Delta = b^2 - 4ac$$

The following calculations are done using the quadratic equation above. Let's consider $x_2^2 + 1.999x_1x_2 + x_1^2 - 1 = 0$ Treat x_1 as a constant. $a = 1, b = 1.999x_1, c = x_1^2 - 1$

$$\Delta = (1.999x_1)^2 - 4 \cdot 1 \cdot (x_1^2 - 1)$$

$$\text{lower bound} = -0.9995x_1 - \sqrt{1 - 0.00099975x_1^2}$$

$$\text{upper bound} = -0.9995x_1 + \sqrt{1 - 0.00099975x_1^2}$$

For the bounds of x_1, x_1 is replaced with x_2 in the equations above.

The conditional distribution for x_2 given x_1 is a uniform distribution on the interval:
$(-0.9995x_1 - \sqrt{1 - 0.00099975x_1^2}, -0.9995x_1 + \sqrt{1 - 0.00099975x_1^2})$

The conditional distribution for x_1 given x_2 is a uniform distribution on the interval:
$(-0.9995x_2 - \sqrt{1 - 0.00099975x_2^2}, -0.9995x_2 + \sqrt{1 - 0.00099975x_2^2})$

part c)

Question: Write your own code for Gibbs sampling the distribution. Run it to generate n = 1000 random vectors and plot them into the picture from Part a. Determine $P(X_1 > 0)$ based on the sample and repeat this a few times (you need not to plot the repetitions). What should be the true result for this probability?

Answer:

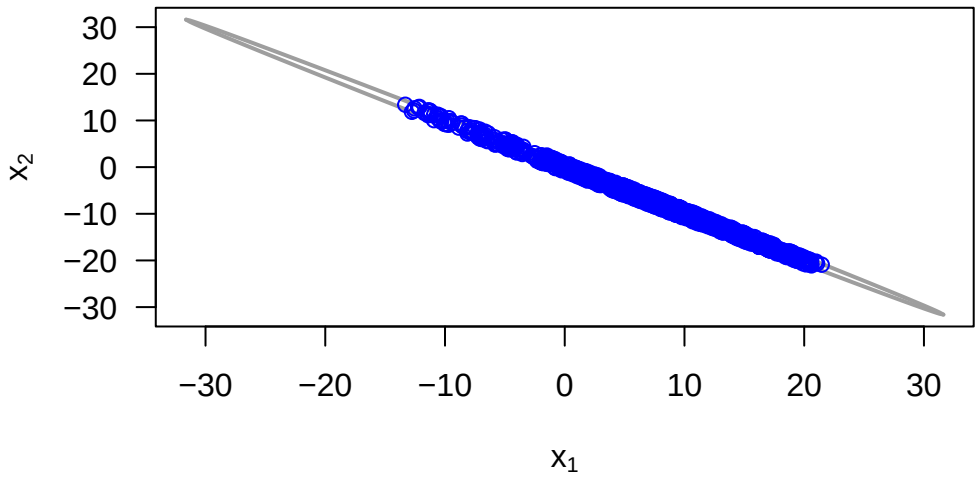


Table 3: P(X1 > 0) with each run

Probability
0.910
0.789
0.864
0.336
0.125
0.116
0.952
0.616
0.582
0.310

P(X1 > 0) fluctuates a lot. This table and the plot of the samples show that the algorithm didn't converge with 1000 iterations. True result of P(X1 > 0) is 0.5 by computing from the area.

```
## True result of P(X1 > 0) : 0.5
```

part d)

Question: Discuss, why the Gibbs sampling for this situation seems to be less successful for w = 1.999 compared to the case w = 1.8 from the lecture.

Answer: The ellipse with w = 1.999 is more elongated compared to w = 1.8. This highly elongated ellipse result in slower convergence and it cannot converge when n = 1000. Consider the point $x_1 = -30$. In that case x_2 can take values in the interval [29.66842, 30.30158] and this interval is so narrow. When any of the x values move away from the center and take values on the corners, the interval for the other x becomes so narrow and leaving that corner requires many iterations hence Gibbs sampling cannot cover the all area.

part e)

Question: We might transform the variable X and generate $U = (U_1, U_2) = (X_1 - X_2, X_1 + X_2)$ instead. In this case, the density of the transformed variable $U = (U_1, U_2)$ is again a uniform distribution on a transformed region (no proof necessary for this claim). Determine the boundaries of the transformed region where U has a uniform distribution on. You can use that the transformation corresponds to $X_1 = (U_2 + U_1)/2$ and $X2 = (U_2 - U_1)/2$ and set this into the boundaries in terms of Xi. Plot the boundaries for (U_1, U_2). Generate n = 1000 random vectors with Gibbs sampling for U and plot them. Determine $P(X_1 > 0) = P((U_2 + U_1)/2 > 0)$. Compare the results with Part c.

Answer: Let's put $X_1 = \frac{(U_2+U_1)}{2}$ and $X_2 = \frac{(U_2-U_1)}{2}$. The new inequality is as follows:

$$\frac{(U_2+U_1)^2}{4} + w\frac{(U_2+U_1)(U_2-U_1)}{4} + \frac{(U_2-U_1)^2}{4} < 1$$

$$\frac{U_1^2(2-w) + U_2^2(2+w)}{4} < 1$$

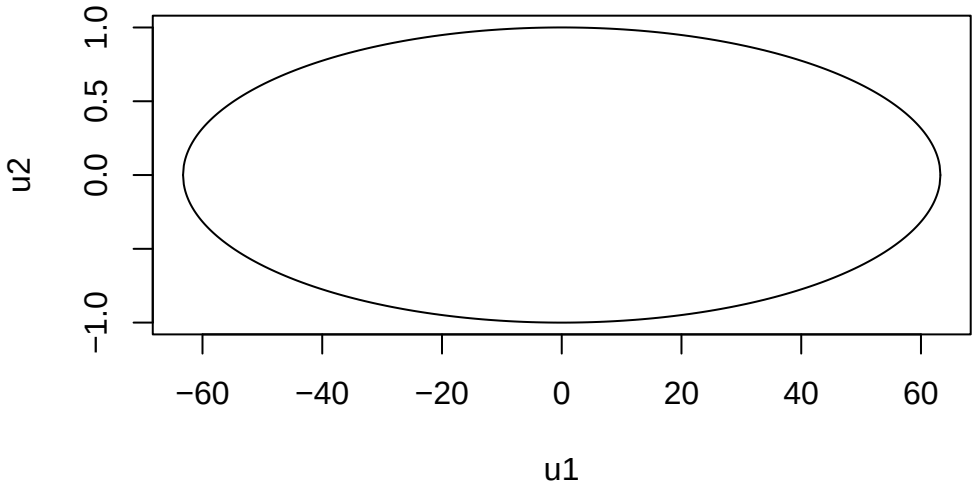
$$\frac{0.001 \cdot U_1^2 + 3.999 \cdot U_2^2}{4} < 1$$

$$-\sqrt{\frac{4 - 0.001 \cdot U_1^2}{3.999}} < U_2 < \sqrt{\frac{4 - 0.001 \cdot U_1^2}{3.999}}$$

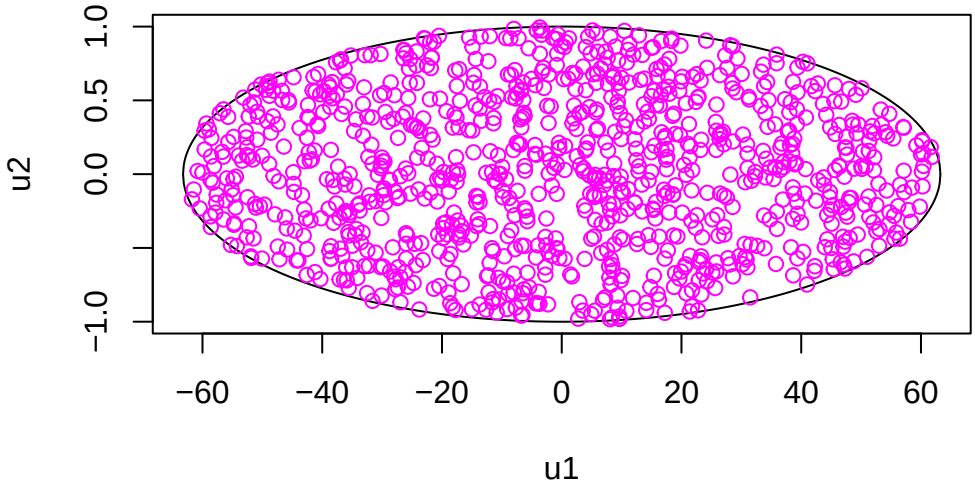
$$\sqrt{\frac{4 - 0.001 \cdot U_1^2}{3.999}} \geq 0$$

$$-\sqrt{\frac{4}{0.001}} \leq U_1 \leq \sqrt{\frac{4}{0.001}}$$

The boundaries of the transformed region can be seen below



The generated samples on the transformed region can be seen below. The gibbs sampling performed much better compared to part c because the new ellipse shape is less elongated and Gibbs sampling could cover the whole area with n = 1000.




```
## Result of P(X1 > 0) = P((U1 + U2)/2 > 0) = 0.497
```

The value of P(X1 > 0) = P((U1 + U2)/2 > 0) is close to the expected value 0.5.

References

Givens, Geof H., and Jennifer A. Hoeting. 2013. *Computational Statistics. John Wiley & Sons, Ltd.* https://doi.org/https://doi.org/10.1002/9781118555552.ch7.

Wackerly, Dennis D., William Mendenhall III, and Richard L. Scheaffer. 2002. *Mathematical Statistics with Applications*. Sixth edition. Duxbury Advanced Series.

Appendix

Question 1:

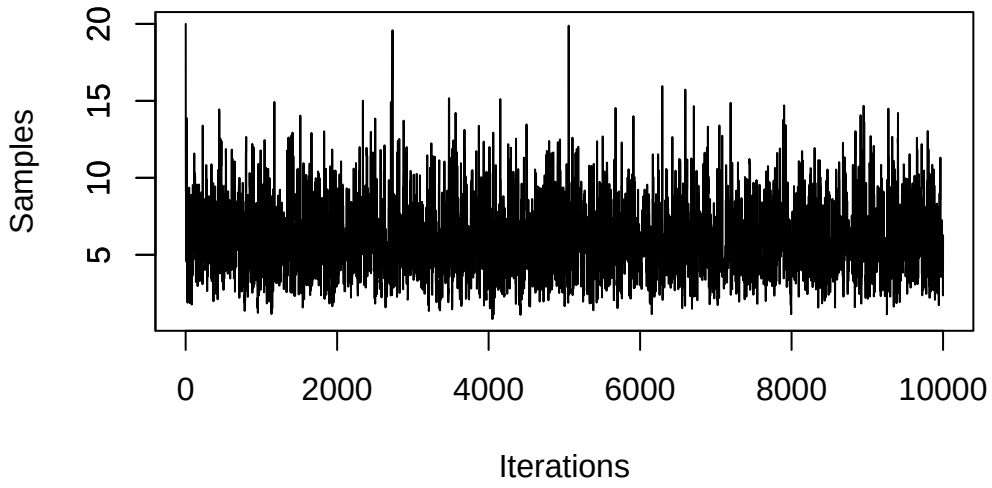
```
set.seed(123)

# Define the target function
target <- function(x){
  density <- (x^5)*(exp(-x))
  return(density)
}

# Function to compute gamma distribution PDF with alpha = 6 and beta = 1
gamma_pdf <- function(x, alpha = 6, beta = 1) {
  return(dgamma(x, shape = alpha, rate = beta))
}
x_gamma_values <- seq(0.001, 20, length.out = 1000)
y_gamma_values <- sapply(x_gamma_values, gamma_pdf)
```

part a)

```
function_a <- function(n, initial_val){
  # initialize
  accept_count_a <- 0
  samples_a <- numeric(n)
  samples_a[1] <- initial_val

  for(i in 1:(n-1)){
    xt <- samples_a[i]
    xt1 <- rlnorm(1, meanlog = log(xt), sdlog = 1) #sample candidate from proposal distribution

    f_xt <- target(xt)
    f_xt1 <- target(xt1)
    r_f <- f_xt1 / f_xt

    g_xt1_xt <- dlnorm(xt1, meanlog = log(xt), sdlog = 1)
    g_xt_xt1 <- dlnorm(xt, meanlog = log(xt1), sdlog = 1)
    r_g <- g_xt_xt1 / g_xt1_xt

    acceptance_ratio <- min(1, r_f*r_g)
    if (acceptance_ratio == 1){
      #accept
      samples_a[i+1] <- xt1
      accept_count_a <- accept_count_a + 1
    } else{
      U <- runif(1)
```

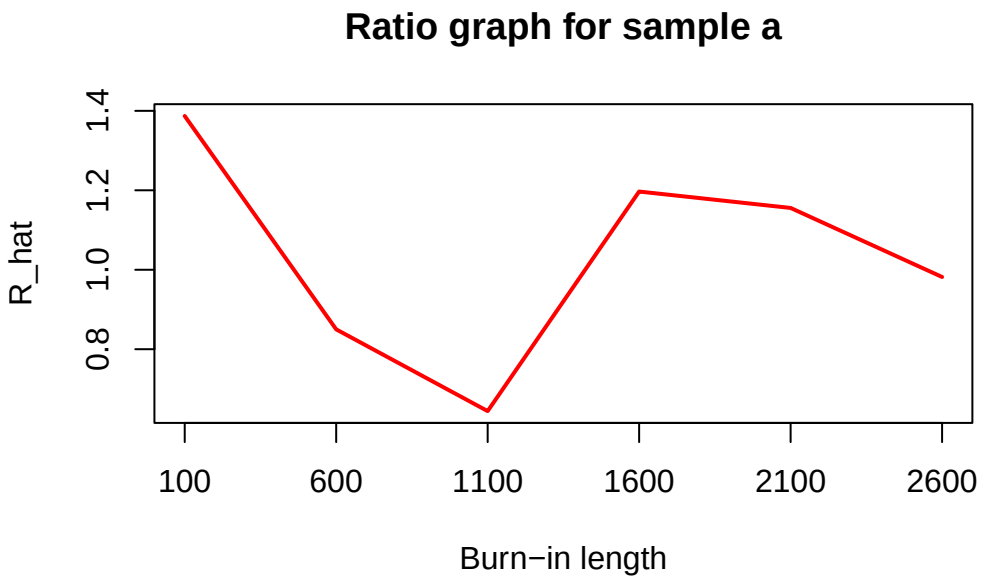
```
      if (U < acceptance_ratio){
        # accept
        samples_a[i+1] <- xt1
        accept_count_a <- accept_count_a + 1
      } else{
        # reject
        samples_a[i+1] <- xt
      }
    }
  }
  acc_rate <- accept_count_a/(n-1)
  result_list <- list(samples = samples_a, acceptance_rate = acc_rate)
  return(result_list)
}
summary_function_a <- function_a(n = 10000, initial_val = 20)
samples_a <- summary_function_a$samples
acceptance_rate_a <- summary_function_a$acceptance_rate
```

```
plot(samples_a, type = "l", xlab = "Iterations", ylab = "Samples")
axis(2, at = c(-5, 0, 5, 10, 15, 20, 25, 30, 35))
```

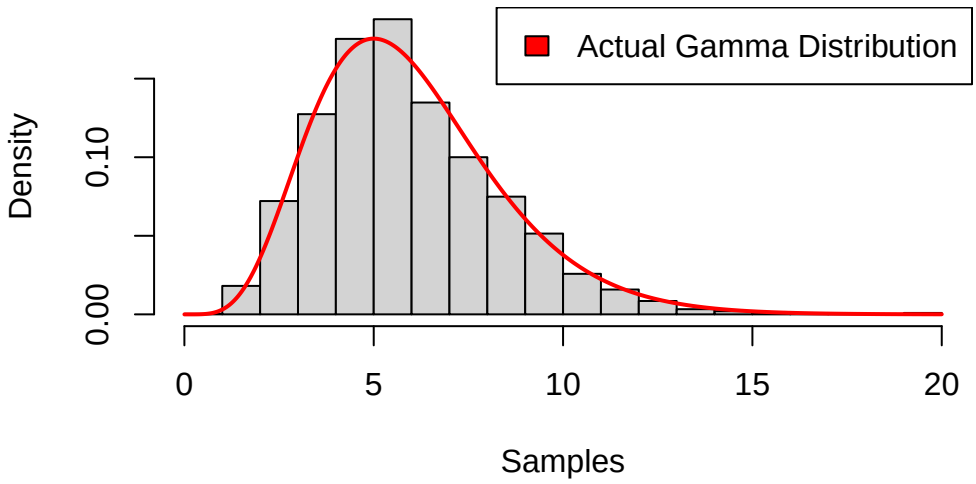



```
ratio_a <- function(D, n = 10000){
  L <- n-D
  J <- 5

  x_j_mean_list <- numeric(J)
  var_j_list <- numeric(J)
  for (i in 1:J){
    result_a <- function_a(n= 10000, initial_val=20)
    x_j <- result_a$samples
    x_j_mean <- (sum(x_j[(D+1):(D+L)]))/L
    x_j_mean_list[i] <- x_j_mean

    var_j_list[i] <- sum(x_j - x_j_mean)/(L-1)
  }

  x_mean <- mean(x_j_mean_list)
  B <- (L/(J-1))*sum((x_j_mean_list - x_mean)^2)
```

```
  W <- mean(var_j_list)

  R <- (((L-1)/L)*W + (1/L)*B) / W
  R_hat <- (((J+1)/J)*R) - ((L-1) / (J*L))

  return(sqrt(R_hat))
}

set.seed(48)
D <- seq(from = 100, to = 3000, by = 500)
ratio <- sapply(D, ratio_a)
plot(D, ratio, type = "l", main = "Ratio graph for sample a", xlab = "Burn-in length",
     ylab = "R_hat", col = "red", lwd = 2, xaxt = "n")
axis(1, at = D)
```




```
hist(samples_a, breaks = 25, main = "Histogram of Samples", xlab = "Samples", freq = FALSE)
lines(x_gamma_values, y_gamma_values, col = 'red', lwd = 2)
legend("topright", legend = c("Actual Gamma Distribution"), fill = c("red"), border = "black")
```




```
cat("Acceptance rate:", acceptance_rate_a)
```

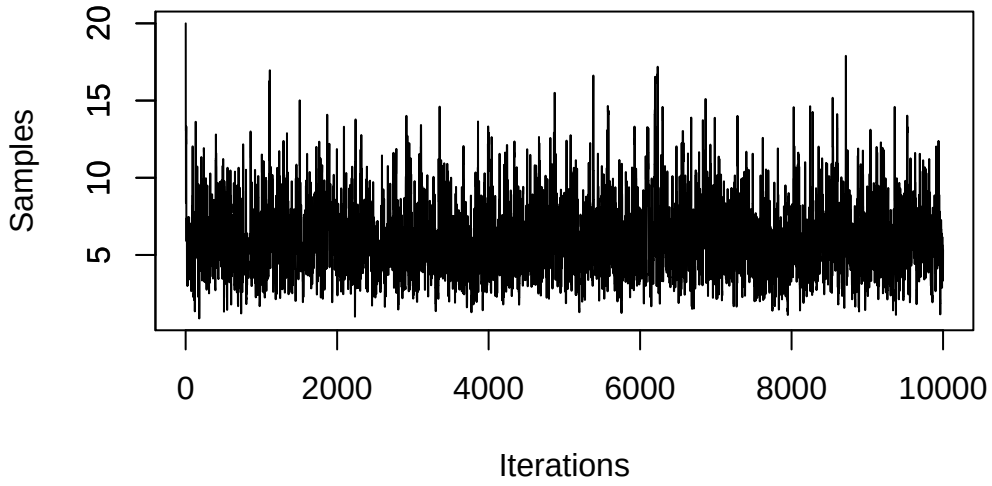
```
## Acceptance rate: 0.4476448
```

part b)

```
function_b <- function(n, initial_val){
  # initialize
  accept_count_b <- 0
  samples_b <- numeric(n)
  samples_b[1] <- initial_val

  for(i in 1:(n-1)){
    xt <- samples_b[i]
    xt1 <- rchisq(1, df = floor(xt)+1) #sample candidate from proposal distribution

    f_xt <- target(xt)
    f_xt1 <- target(xt1)
    r_f <- f_xt1 / f_xt

    g_xt1_xt <- dchisq(xt1, df = floor(xt)+1)
    g_xt_xt1 <- dchisq(xt, df = floor(xt1)+1)
    r_g <- g_xt_xt1 / g_xt1_xt

    acceptance_ratio <- min(1, r_f*r_g)
    if (acceptance_ratio == 1){
      #accept
      samples_b[i+1] <- xt1
      accept_count_b <- accept_count_b+1
    } else{
      U <- runif(1)
      if (U < acceptance_ratio){
        # accept
        samples_b[i+1] <- xt1
        accept_count_b <- accept_count_b+1
```

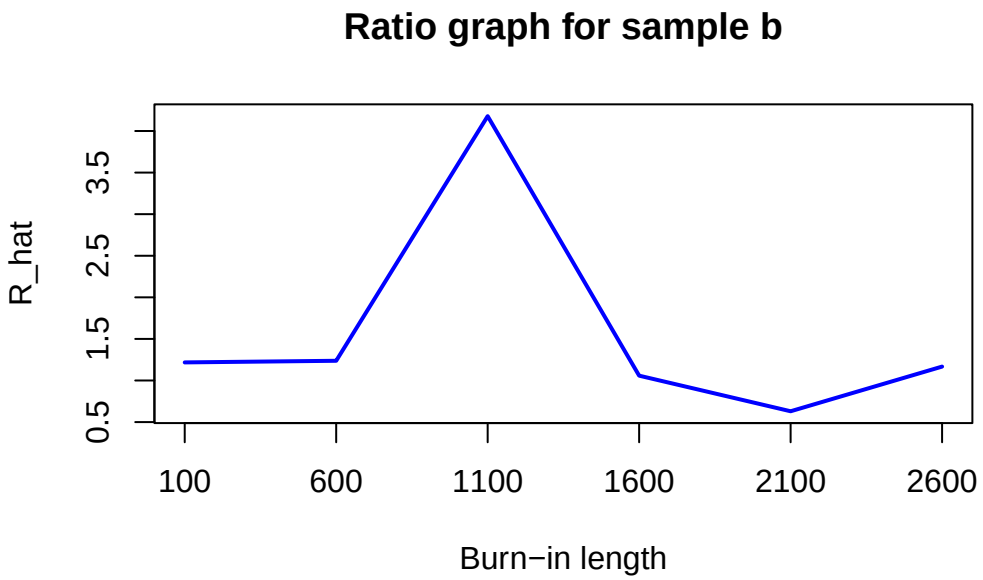
```
      } else{
        # reject
        samples_b[i+1] <- xt
      }
    }
  }
  acc_rate <- accept_count_b/(n-1)
  result_list <- list(samples = samples_b, acceptance_rate = acc_rate)
  return(result_list)
}
summary_function_b <- function_b(n = 10000, initial_val = 20)
samples_b <- summary_function_b$samples
acceptance_rate_b <- summary_function_b$acceptance_rate
```

```
plot(samples_b, type = "l", xlab = "Iterations", ylab = "Samples")
axis(2, at = c(-5, 0, 5, 10, 15, 20, 25, 30, 35))
```

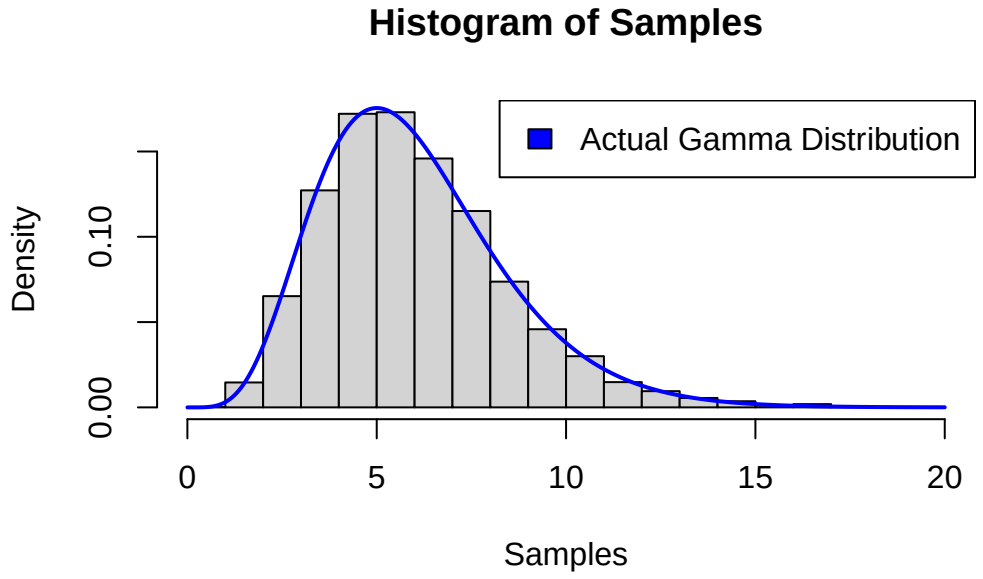



```
ratio_b <- function(D, n = 10000){
  L <- n-D
  J <- 5

  x_j_mean_list <- numeric(J)
  var_j_list <- numeric(J)
  for (i in 1:J){
    result_b <- function_b(n= 10000, initial_val=20)
    x_j <- result_b$samples
    x_j_mean <- (sum(x_j[(D+1):(D+L)]))/L
    x_j_mean_list[i] <- x_j_mean

    var_j_list[i] <- sum(x_j - x_j_mean)/(L-1)
  }

  x_mean <- mean(x_j_mean_list)
  B <- (L/(J-1))*sum((x_j_mean_list - x_mean)^2)
  W <- mean(var_j_list)

  R <- (((L-1)/L)*W + (1/L)*B) / W
  R_hat <- (((J+1)/J)*R) - ((L-1) / (J*L))
```

```
  return(sqrt(R_hat))
}
set.seed(24)
D <- seq(from = 100, to = 3000, by = 500)
ratio <- sapply(D, ratio_b)
plot(D, ratio, type="l", main = "Ratio graph for sample b", xlab = "Burn-in length",
     ylab = "R_hat", col = "blue", lwd = 2, xaxt = "n")
axis(1, at = D)
```




```
hist(samples_b, breaks = 25, main = "Histogram of Samples", xlab = "Samples", freq = FALSE)
lines(x_gamma_values, y_gamma_values, col = 'blue', lwd = 2)
legend("topright", legend = c("Actual Gamma Distribution"), fill = c("blue"), border = "black")
```



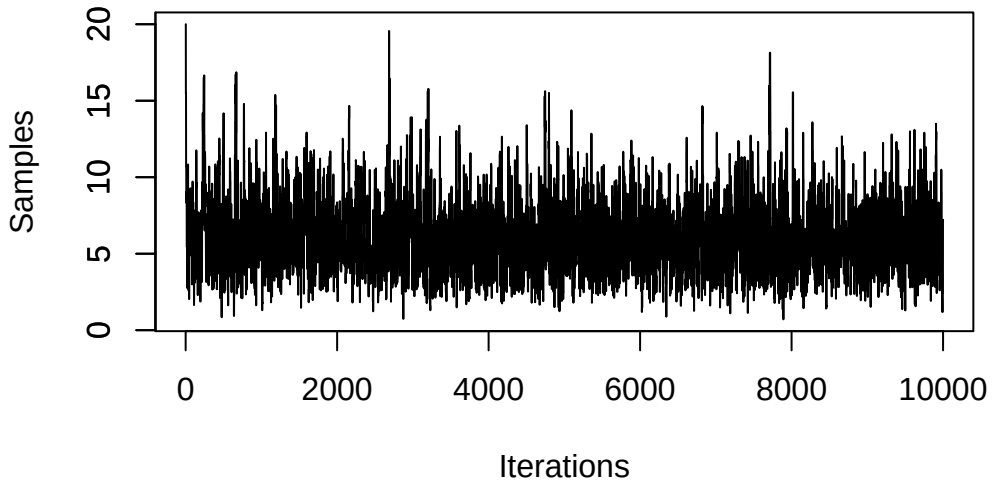

```
cat("Acceptance rate:", acceptance_rate_b)
```

```
## Acceptance rate: 0.5929593
```

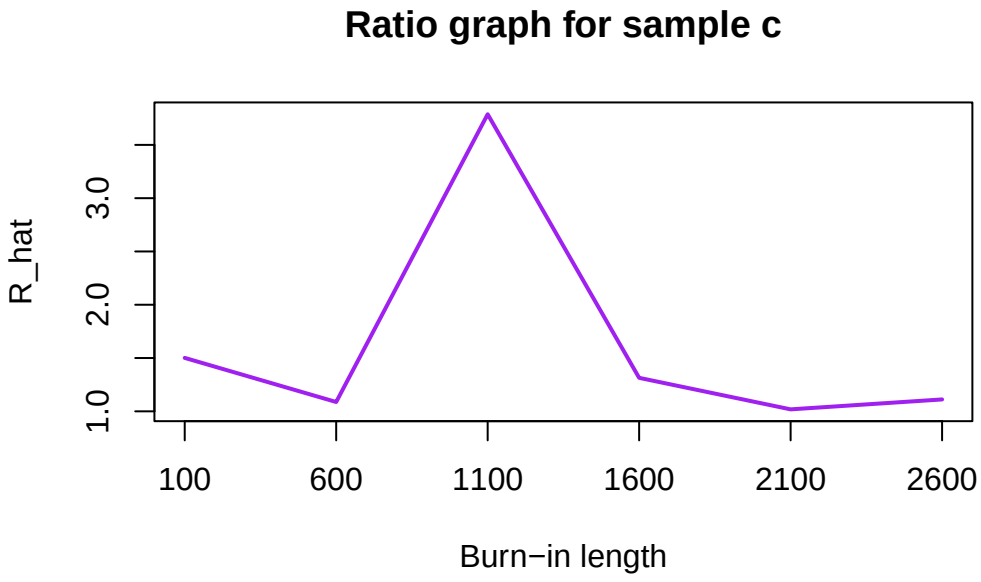
part c)

```
function_c <- function(n, initial_val){
  # initialize
  accept_count_c <- 0
  samples_c <- numeric(n)
  samples_c[1] <- initial_val

  for(i in 1:(n-1)){
    xt <- samples_c[i]
    xt1 <- rnorm(1, mean = xt, sd = 4) #sample candidate from proposal distribution

    f_xt <- target(xt)
    f_xt1 <- target(xt1)
    r_f <- f_xt1 / f_xt

    acceptance_ratio <- min(1, r_f)
    if (acceptance_ratio == 1){
      #accept
      samples_c[i+1] <- xt1
      accept_count_c <- accept_count_c+1
    } else{
      U <- runif(1)
      if (U < acceptance_ratio){
        # accept
        samples_c[i+1] <- xt1
        accept_count_c <- accept_count_c+1
      } else{
        # reject
        samples_c[i+1] <- xt
      }
    }
  }
  acc_rate <- accept_count_c/(n-1)
  result_list <- list(samples = samples_c, acceptance_rate = acc_rate)
  return(result_list)
}
summary_function_c <- function_c(n = 10000, initial_val = 20)
samples_c <- summary_function_c$samples
acceptance_rate_c <- summary_function_c$acceptance_rate
```

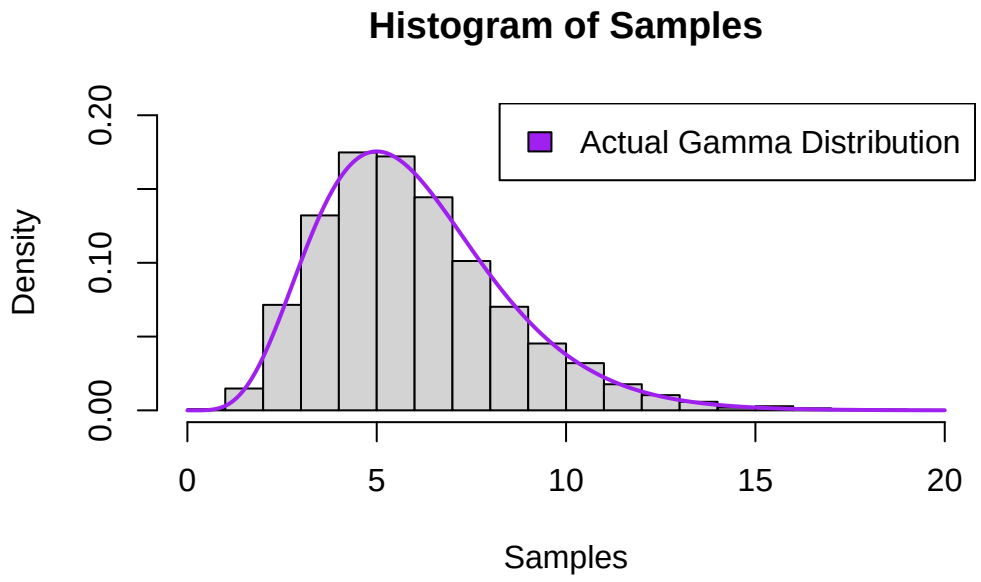
```
plot(samples_c, type = "l", xlab = "Iterations", ylab = "Samples")
axis(2, at = c(-5, 0, 5, 10, 15, 20, 25, 30, 35))
```




```r
ratio_c <- function(D, n = 10000){
  L <- n-D
  J <- 5

  x_j_mean_list <- numeric(J)
  var_j_list <- numeric(J)
  for (i in 1:J){
    result_c <- function_c(n= 10000, initial_val=20)
    x_j <- result_c$samples
    x_j_mean <- (sum(x_j[(D+1):(D+L)]))/L
    x_j_mean_list[i] <- x_j_mean

    var_j_list[i] <- sum(x_j - x_j_mean)/(L-1)
  }

  x_mean <- mean(x_j_mean_list)
  B <- (L/(J-1))*sum((x_j_mean_list - x_mean)^2)
  W <- mean(var_j_list)

  R <- (((L-1)/L)*W + (1/L)*B) / W
  R_hat <- (((J+1)/J)*R) - ((L-1) / (J*L))

  return(sqrt(R_hat))
}

set.seed(12)
D <- seq(from = 100, to = 3000, by = 500)
ratio <- sapply(D, ratio_c)
plot(D, ratio, type="l", main = "Ratio graph for sample c", xlab = "Burn-in length",
     ylab = "R_hat", col = "purple", lwd = 2, xaxt = "n")
axis(1, at = D)
```




```
hist(samples_c, breaks = 25, main = "Histogram of Samples",
     xlab = "Samples", freq = FALSE, ylim = c(0, 0.2))
lines(x_gamma_values, y_gamma_values, col = 'purple', lwd = 2)
legend("topright", legend = c("Actual Gamma Distribution"), fill = c("purple"), border = "black")
```




```
cat("Acceptance rate:", acceptance_rate_c)
```

```
## Acceptance rate: 0.5355536
```

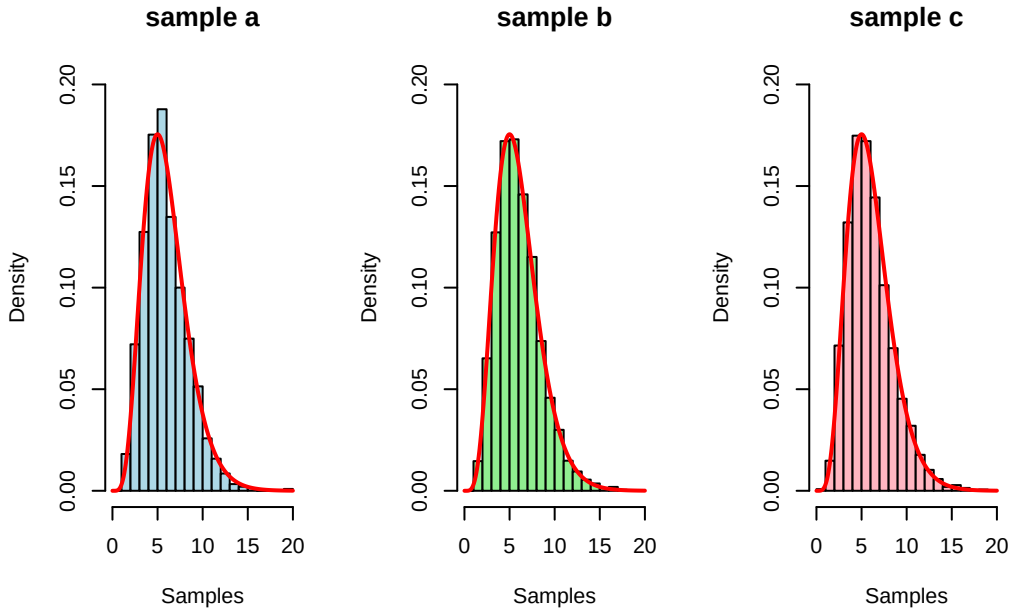
part d)

```
library(knitr)
acc_rate <- c(acceptance_rate_a, acceptance_rate_b, acceptance_rate_c)
data <- data.frame("Sample a" = acc_rate[1], "Sample b" = acc_rate[2], "Sample c" = acc_rate[3])
colnames(data) <- c("sample a", "sample b", "sample c")
kable(data, caption = "Acceptance Rate Table")
```

Table 4: Acceptance Rate Table

sample a	sample b	sample c
0.4476448	0.5929593	0.5355536

```
par(mfrow = c(1, 3))
hist(samples_a, breaks = 25, main = "sample a", xlab = "Samples", freq = FALSE,
     col = "lightblue", ylim = c(0,0.2))
lines(x_gamma_values, y_gamma_values, col = 'red', lwd = 2)

hist(samples_b, breaks = 25, main = "sample b", xlab = "Samples", freq = FALSE,
     col = "lightgreen", ylim = c(0,0.2))
lines(x_gamma_values, y_gamma_values, col = 'red', lwd = 2)

hist(samples_c, breaks = 25, main = "sample c", xlab = "Samples", freq = FALSE,
     col = "lightpink", ylim = c(0,0.2))
lines(x_gamma_values, y_gamma_values, col = 'red', lwd = 2)
```



part e)

```
library(knitr)
means <- c(mean(samples_a), mean(samples_b), mean(samples_c))
variances <- c(var(samples_a), var(samples_b), var(samples_c))

# Create a data frame
data <- data.frame(Mean = means, Variance = variances)
rownames(data) <- c("sample a", "sample b", "sample c")
kable(data, caption = "Mean & Variance Table")
```

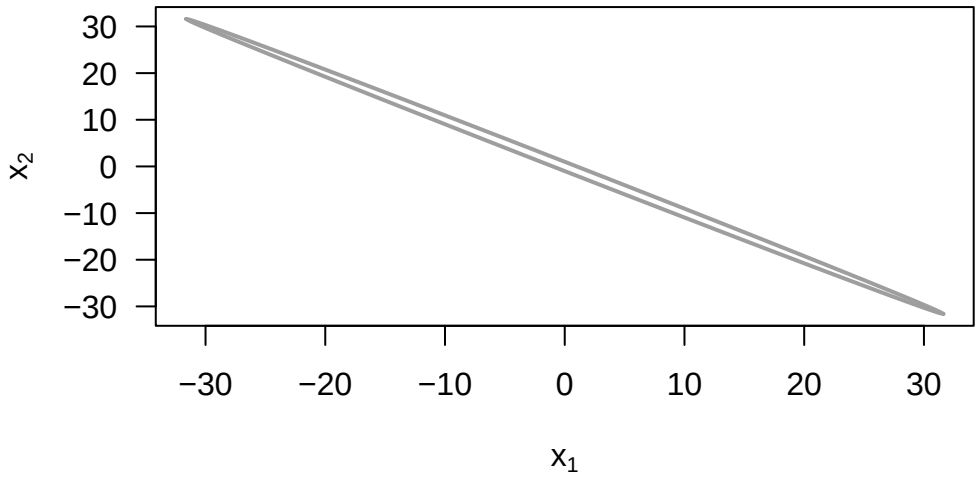
Table 5: Mean & Variance Table

	Mean	Variance
sample a	5.906256	5.801632
sample b	6.021899	5.965698
sample c	5.973835	6.296328

Question 2

part a)

```
w  <- 1.999
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)

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)
```

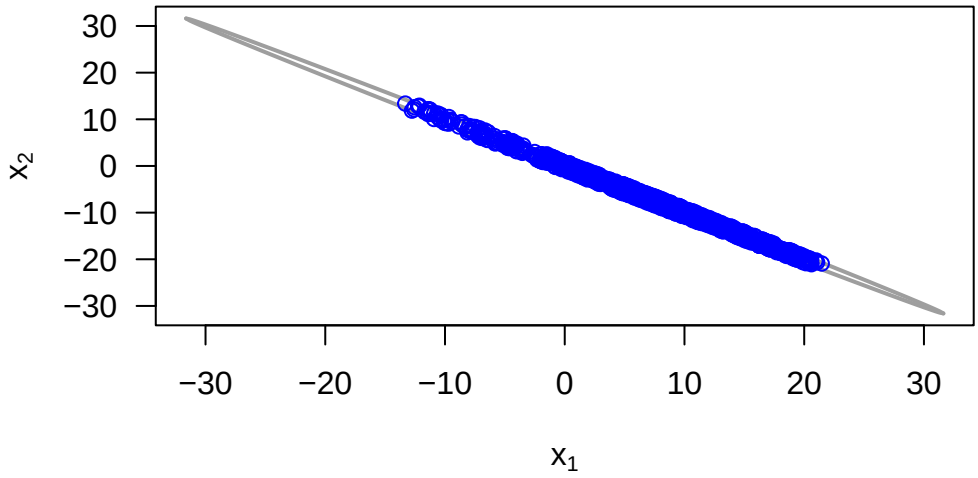


part c)

```
sample_X1_given_X2 <- function(x2) {
  lower_bound <- -0.9995*x2 - sqrt(1-0.00099975*(x2^2))
  upper_bound <- -0.9995*x2 + sqrt(1-0.00099975*(x2^2))
  return(runif(1, lower_bound, upper_bound))
}

sample_X2_given_X1 <- function(x1) {
  lower_bound <- -0.9995*x1 - sqrt(1-0.00099975*(x1^2))
  upper_bound <- -0.9995*x1 + sqrt(1-0.00099975*(x1^2))
  return(runif(1, lower_bound, upper_bound))
}

gibbs_sampling <- function(n){
  samples <- matrix(NA, nrow = n, ncol = 2)
  # initial values
  x1 <- 0
  x2 <- 0
```

```
  for (i in 1:n) {
    # Sample x1 given x2
    x1 <- sample_X1_given_X2(x2)
    samples[i, ] <- c(x1, x2)

    # Sample x2 given x1
    x2 <- sample_X2_given_X1(x1)
    samples[i, ] <- c(x1, x2)
  }
  return(samples)
}
samples <- gibbs_sampling(1000)
```

```
# Ellipse
w  <- 1.999
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)
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)

points(samples, col = "blue")
```




```
# calculate P(X1>0) with repetition
prob <- numeric(10)
for (i in 1:10){
  samples <- gibbs_sampling(1000)
  num_X1_gt_0 <- sum(samples[, 1] > 0)
  total_samples <- nrow(samples)
  P_X1_gt_0 <- num_X1_gt_0 / total_samples
  prob[i] <- P_X1_gt_0
}

library(knitr)
data <- data.frame(Probability = prob)
kable(data, caption = "P(X1 > 0) with each run")
```

Table 6: P(X1 > 0) with each run

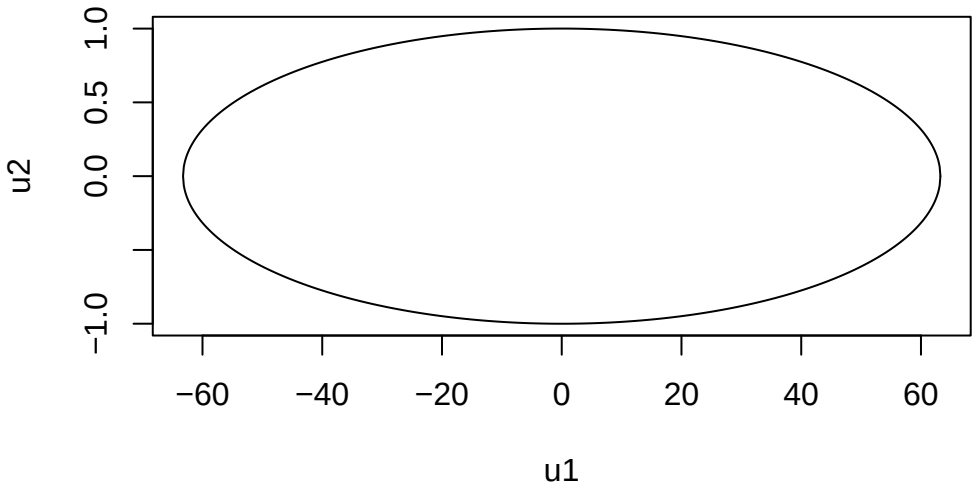
Probability
0.910
0.789
0.864
0.336
0.125
0.116
0.952
0.616
0.582
0.310

```r
# Calculation is done using the area
w <- 1.999
total_area <- pi * 1 * sqrt(1 - (1 - w^2 / 4))
area_X1_gt_0 <- 0.5 * pi * 1 * sqrt(1 - (1 - w^2 / 4))
P_X1_gt_0_area <- area_X1_gt_0 / total_area
cat("True result of P(X1 > 0) :", P_X1_gt_0_area, "\n")
```

```
## True result of P(X1 > 0) : 0.5
```

part e)

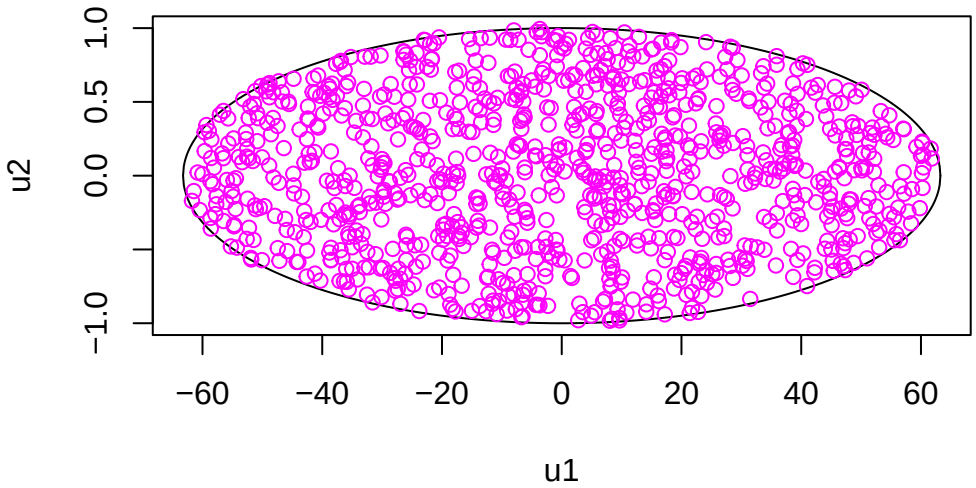
```r
u1 <- seq(-sqrt(4/0.001), sqrt(4/0.001), length.out = 1000)
upper <- sqrt((4-0.001*u1^2)/3.999)
lower <- -sqrt((4-0.001*u1^2)/3.999)

plot(u1, upper, type = "l", col = "black", xlab = "u1", ylab = "u2", ylim=c(-1,1))
lines(u1, lower, col = "black")
```




```r
sample_U1_given_U2 <- function(u2) {
  lower_bound <- -sqrt((4-3.999*u2^2)/0.001)
  upper_bound <- sqrt((4-3.999*u2^2)/0.001)
  return(runif(1, lower_bound, upper_bound))
}
```

```r
sample_U2_given_U1 <- function(u1) {
  lower_bound <- -sqrt((4-0.001*u1^2)/3.999)
  upper_bound <- sqrt((4-0.001*u1^2)/3.999)
  return(runif(1, lower_bound, upper_bound))
}

n <- 1000
samples2 <- matrix(NA, nrow = n, ncol = 2)

u1 <- 0
u2 <- 0

for (i in 1:n) {
  # Sample u1 given u2
  u1 <- sample_U1_given_U2(u2)
  samples2[i, ] <- c(u1, u2)

  # Sample u2 given u1
  u2 <- sample_U2_given_U1(u1)
  samples2[i, ] <- c(u1, u2)
}
```

```r
u1 <- seq(-sqrt(4/0.001), sqrt(4/0.001), length.out = 1000)
upper <- sqrt((4-0.001*u1^2)/3.999)
lower <- -sqrt((4-0.001*u1^2)/3.999)

plot(u1, upper, type = "l", col = "black", xlab = "u1", ylab = "u2", ylim=c(-1,1))
lines(u1, lower, col = "black")
points(samples2, col = "magenta")
```




```r
result <- sum(((samples2[,1] + samples2[,2])/2)>0) / nrow(samples2)
cat("Result of P(X1 > 0) = P((U1 + U2)/2 > 0) =", result, "\n")
```

```
## Result of P(X1 > 0) = P((U1 + U2)/2 > 0) = 0.497
```