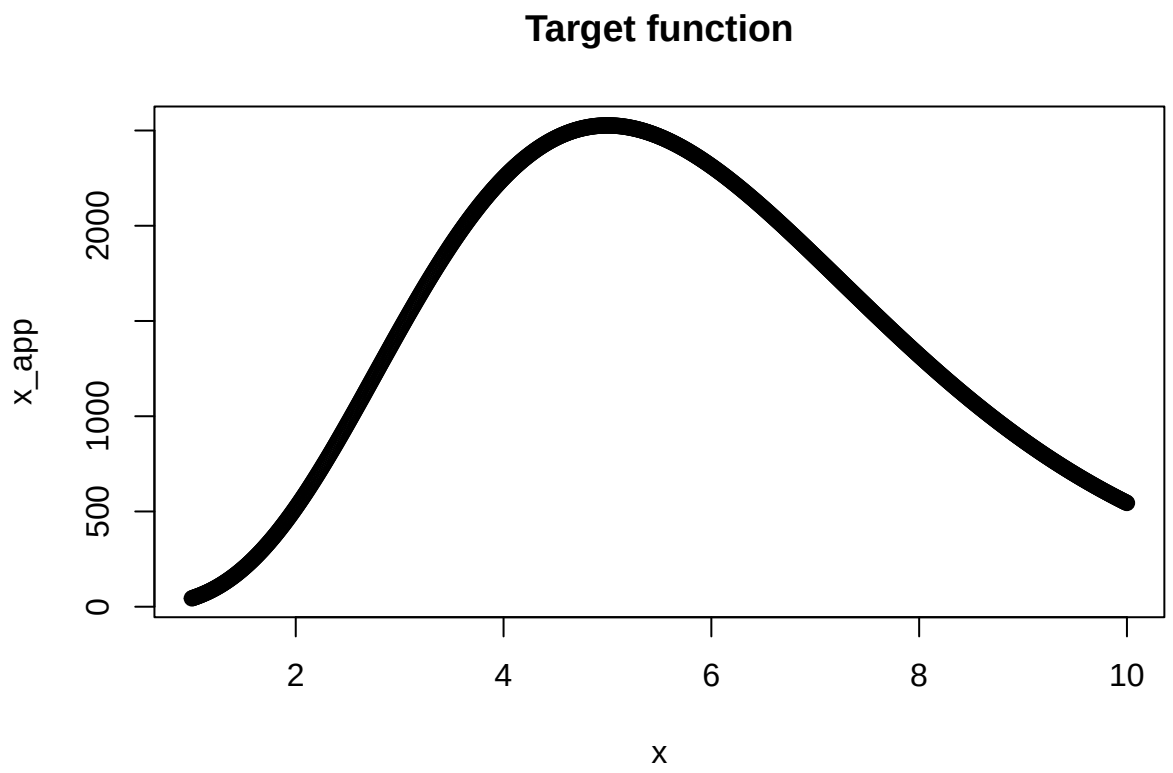
Question 1:

$$f(x) = 120x^5e^{-x}, x > 0$$

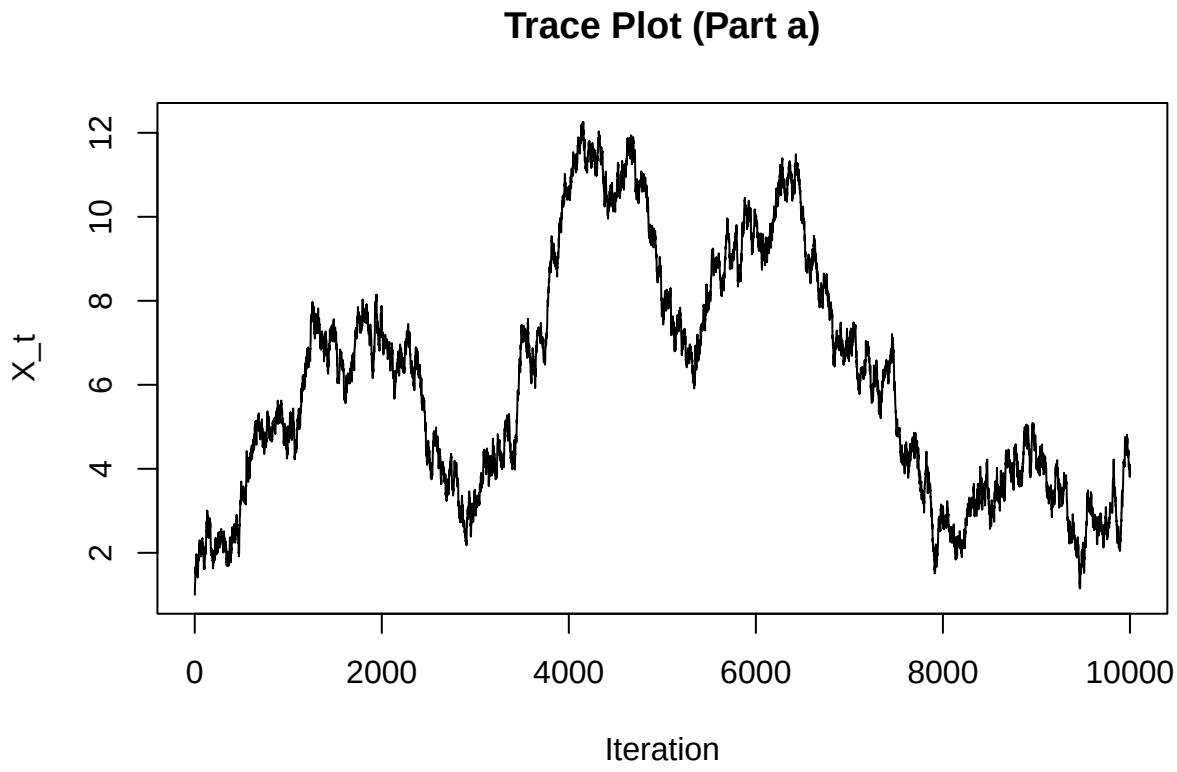
The target function has the following shape:

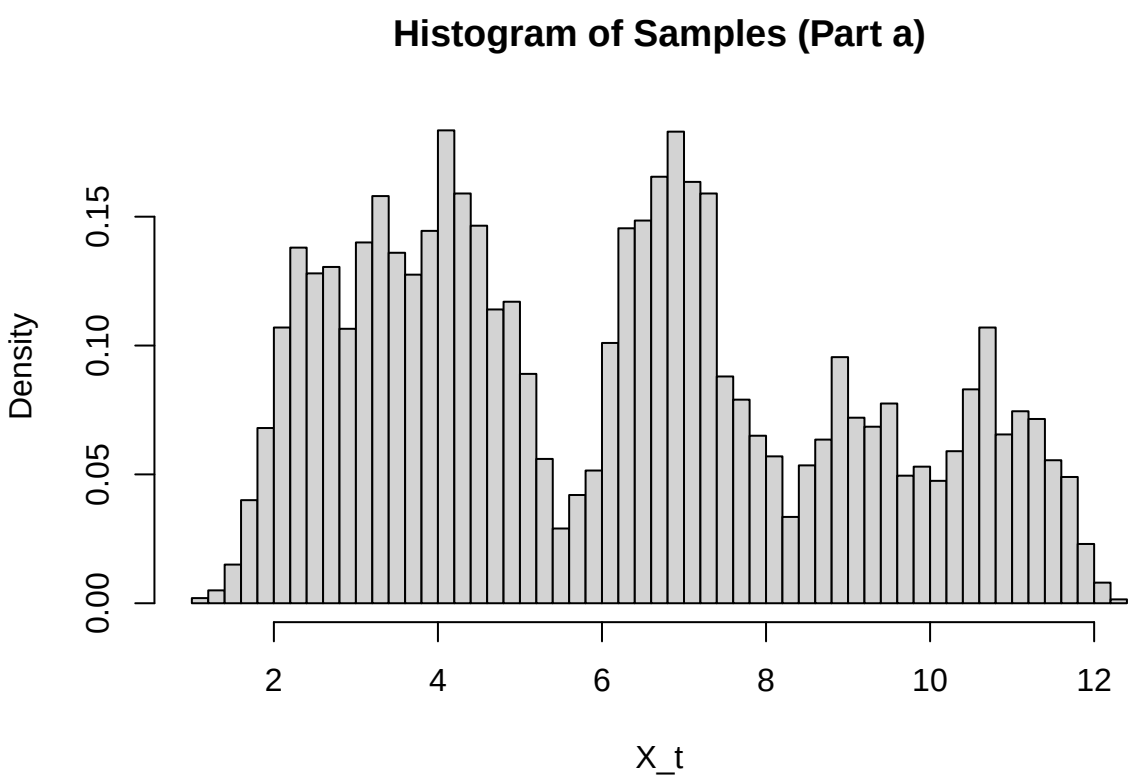



```r
set.seed(42)

metropolis_normal <- function(n_iter = 10000, start = 1, proposal_sd = 0.1) {
  x <- numeric(n_iter)
  x[1] <- start
  accept <- 0

  for (t in 2:n_iter) {
    proposal <- rnorm(1, mean = x[t - 1], sd = proposal_sd)

    # f undefined for x <= 0
    if (proposal > 0) {
      rate <- min(1, f(proposal) / f(x[t - 1]))
      if (runif(1) < rate) {
        x[t] <- proposal
        accept <- accept + 1
      } else {
```

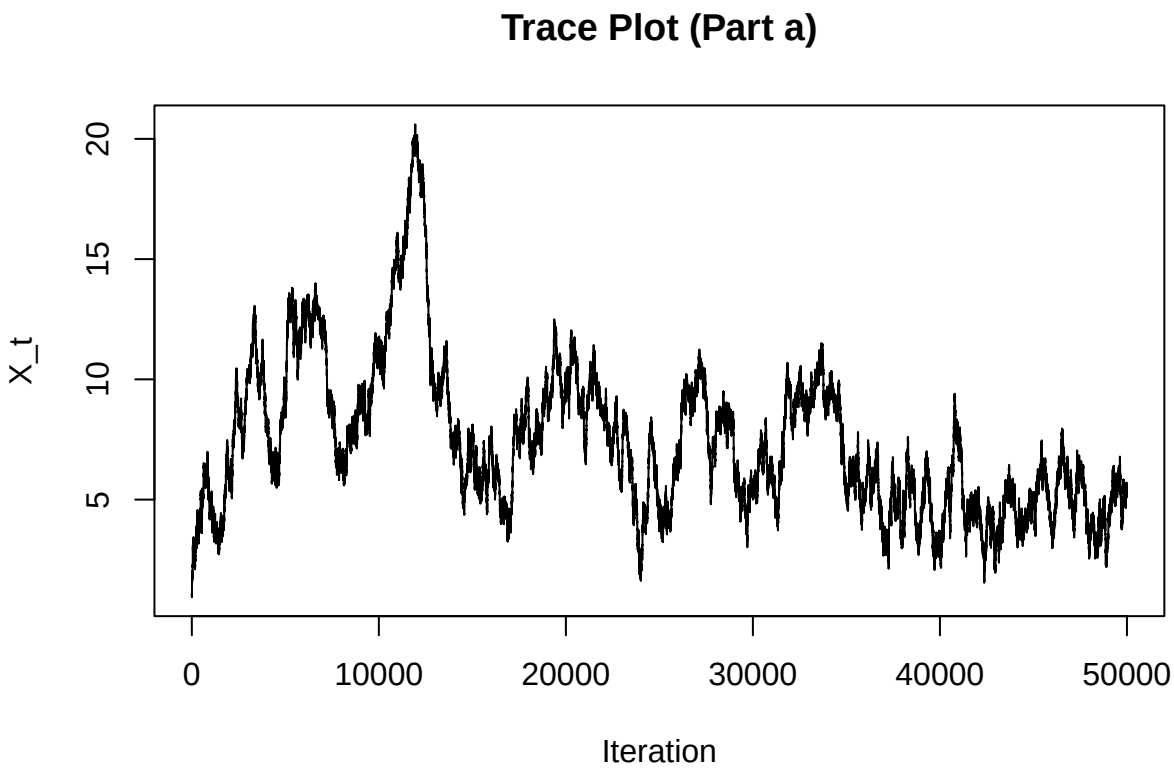
```
        x[t] <- x[t - 1]
      }
    } else {
      x[t] <- x[t - 1]
    }
  }

  return(list(chain = x, acceptance_rate = accept / n_iter))
}
```

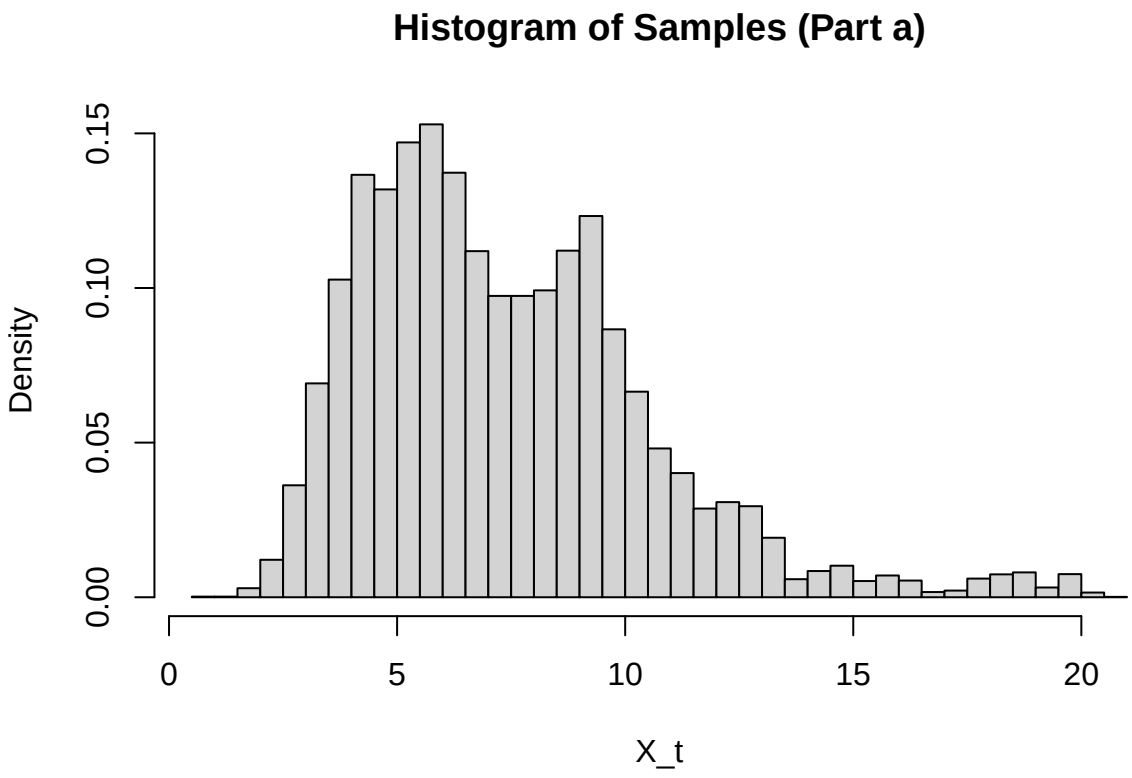
a. Use the Metropolis–Hastings algorithm to generate 10000 samples X_t, t = 1, . . . , 10000, from this distribution. Use a normal distribution with mean X_t and standard deviation 0.1 as proposal distribution when you generate X_{t+1}; take some starting point. Plot the chain you obtained with iterations 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.





- The generated sequence converges to the target distribution, if it is irreducible and aperiodic. Irreducibility is given by the fact that for the normal distribution every X_t has a non-zero probability to be reached from every other positive value. It is also aperiodic as the proposal distribution does not have a fixed cycle. Hence, we can say, that the sequence converges.
- According to the trace plot the chain seems too have a very long burn-in period, as it is not really fluctuating around a value. This would explain why the histogram of the samples has no similarity with the shape of the target distributions. Getting a bigger sequence with 50,000 samples instead confirms that the sequence converges. The burn-in period seems to be around 15000 iterations:

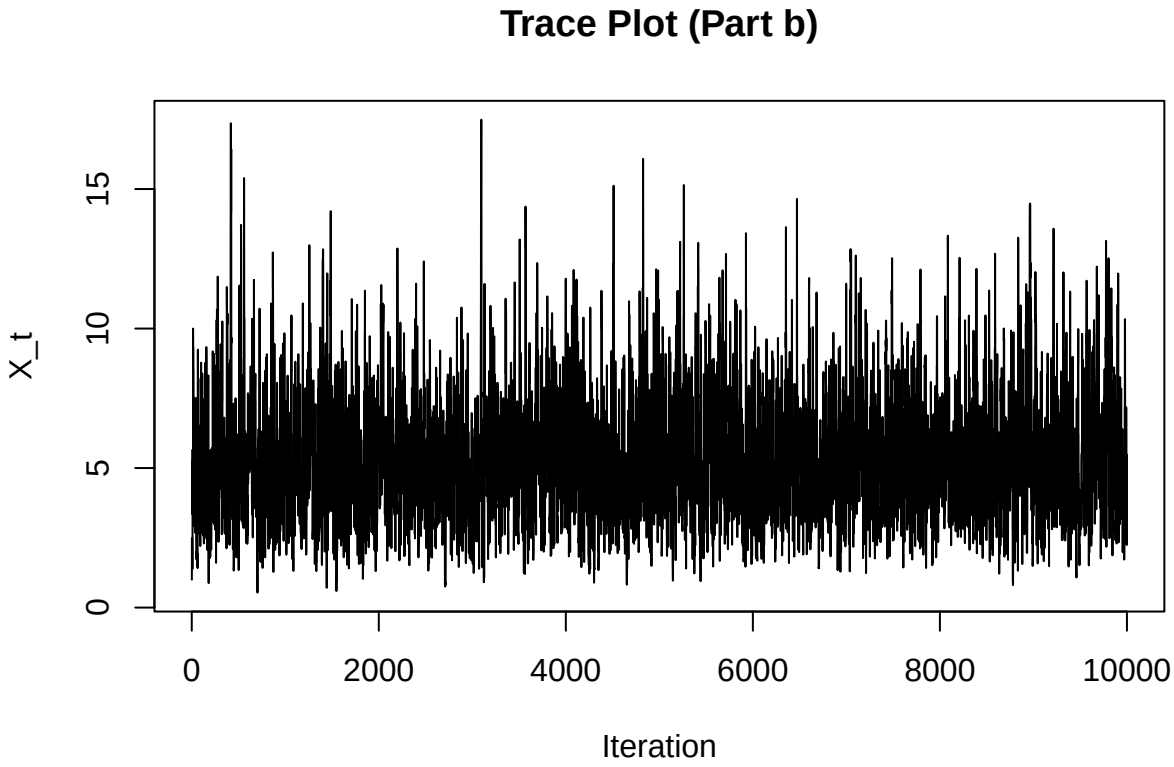

Trace Plot (Part a)
X_t
5
10
15
20
0
10000
20000
30000
40000
50000
Iteration

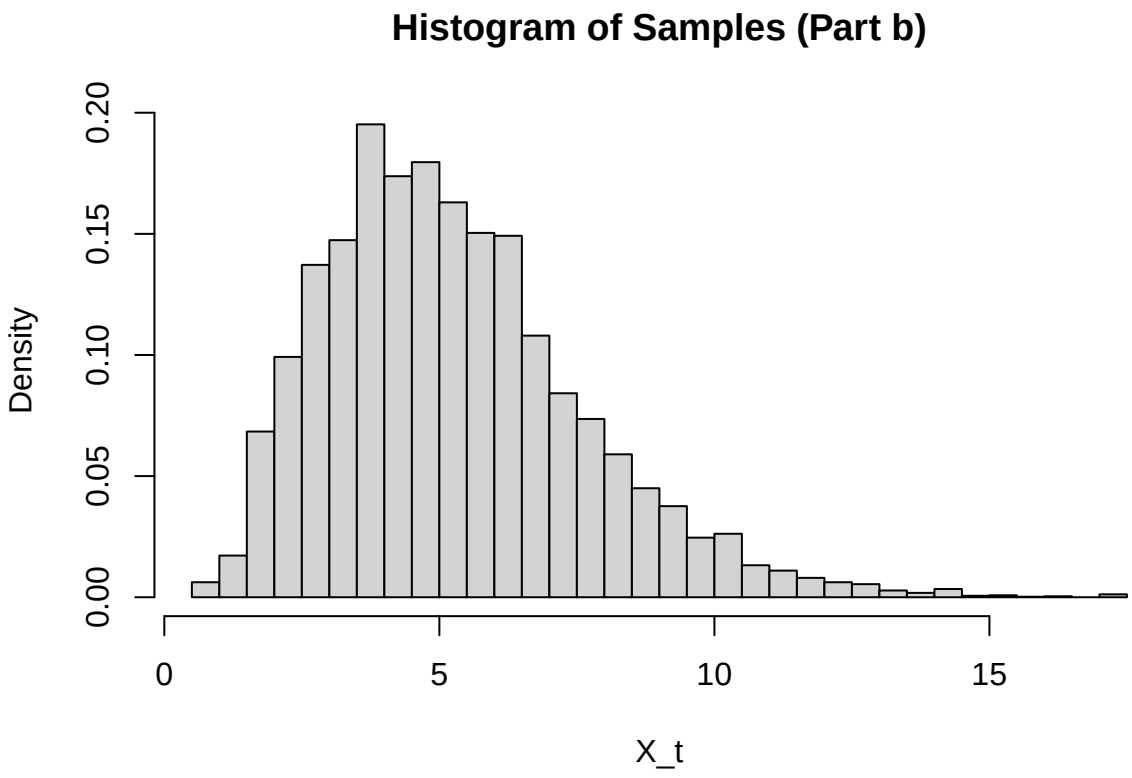


- The acceptance rate is very high: `0.9835`.

```
metropolis_chisq <- function(n_iter = 10000, start = 1) {
  x <- numeric(n_iter)
  x[1] <- start
  accept <- 0

  for (t in 2:n_iter) {
    proposal <- rchisq(1, floor(x[t - 1] + 1))
    rate <- min(1, f(proposal) / f(x[t - 1]))
    if (runif(1) < rate) {
      x[t] <- proposal
      accept <- accept + 1
    } else {
      x[t] <- x[t - 1]
    }
  }

  list(samples = x, acceptance_rate = accept / n_iter)
}
```

b. Perform part a. by using the chi-square distribution as a proposal distribution.





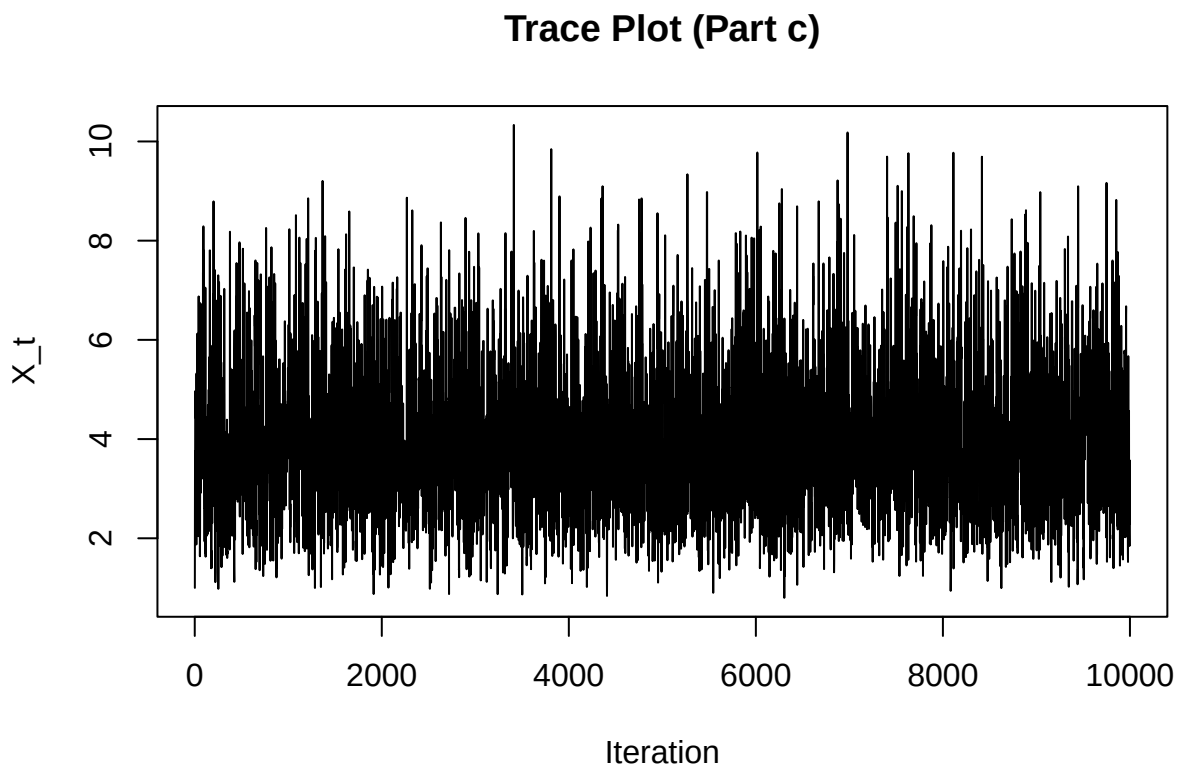
- According to the definition the sequence should converge again. The shape of the histogram confirms this assumption.
- For this proposal distribution the burn-in period is significantly shorter and can't really be derived from the trace plot.
- The acceptance rate is `0.6047`

c. Suggest another proposal distribution (can be a 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. For this sequence we choose a gamma distribution with $\alpha = 3$ and $\beta = 1$:

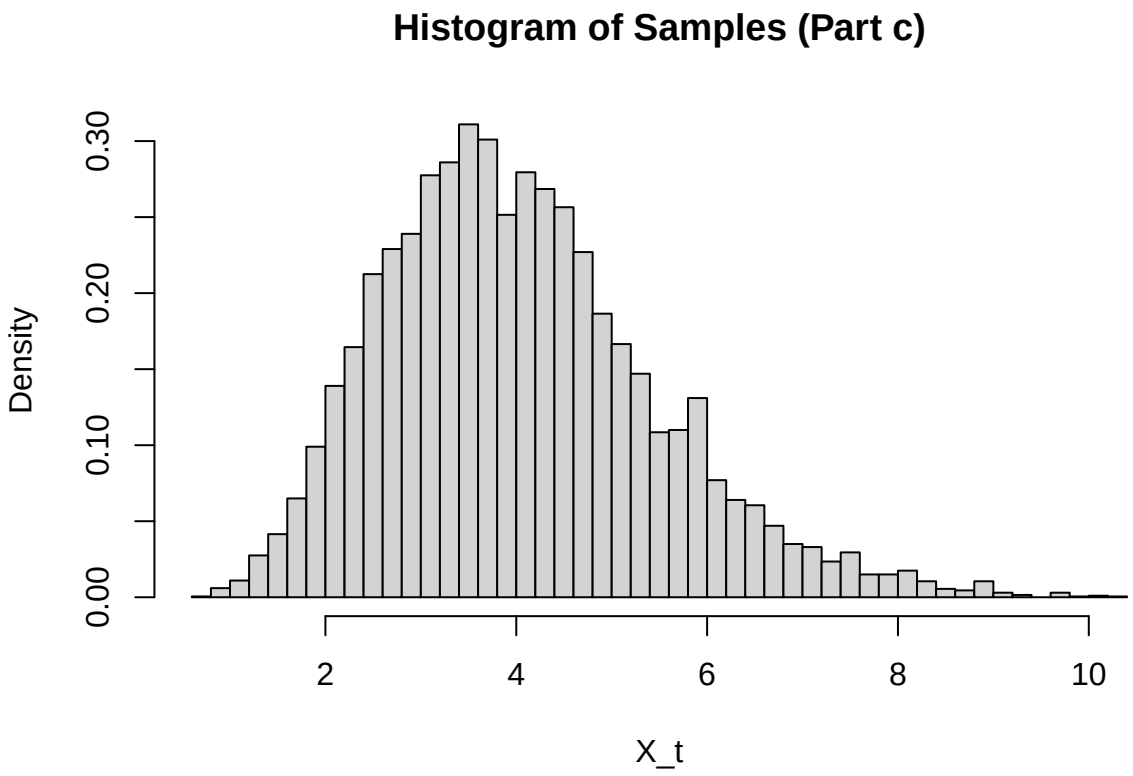
```
metropolis_gamma <- function(n_iter = 10000, start = 1, alpha = 3, beta = 1) {
  x <- numeric(n_iter)
  x[1] <- start
  accept <- 0

  for (t in 2:n_iter) {
    proposal <- rgamma(1, shape = alpha, rate = beta)
    rate <- min(1, f(proposal) / f(x[t - 1]))
    if (runif(1) < rate) {
      x[t] <- proposal
      accept <- accept + 1
    } else {
      x[t] <- x[t - 1]
```

```
    }
  }

  list(samples = x, acceptance_rate = accept / n_iter)
}
```

```
result_c <- metropolis_gamma()
samples_c <- result_c$samples
acceptance_rate_c <- result_c$acceptance_rate

plot(samples_c, type = "l", main = "Trace Plot (Part c)", xlab = "Iteration", ylab = "X_t")
```




```
hist(samples_c, probability = TRUE, breaks = 50, main = "Histogram of Samples (Part c)", xlab = "X_t")
```



- We expect the sequence to converge again using a gamma distribution as proposal distribution, according to the definition.
- The results are very similar to the ones from part b. The shape of the histogram follows the target distribution and the trace plot indicates a very short burn-in rate.
- With `0.5699` the acceptance rate is even a bit lower.

d. Compare the results of parts a, b and c and make conclusions. Sequences from the target distribution can be sampled with all three proposal distributions. Using the normal distribution requires significantly more iterations and a high burn-in period. It also has a very high acceptance rate, which makes it suboptimal as a proposal distribution.

The results for the other two proposal distributions are very similar. Though the histogram resulting from using the gamma distribution seems to fit even a bit better and it also has the lower acceptance rate that is closer to the ideal 0.44 for the unidimensional case.

Therefore, the gamma distribution would be our choice for the proposal distribution.

e. Estimate the expected value using samples from parts a, b and c.

```
## Estimated E(X) (Part a): 7.393247
```

```
## Estimated E(X) (Part b): 5.262541
```

```
## Estimated E(X) (Part c): 4.031483
```

f. The distribution generated is in fact a gamma distribution. Search in the literature/internet and define the actual value of the integral. Compare it with the one you obtained. Source for this task: https://en.wikipedia.org/wiki/Gamma_distribution

The general PDF for the gamma distribution is the following:

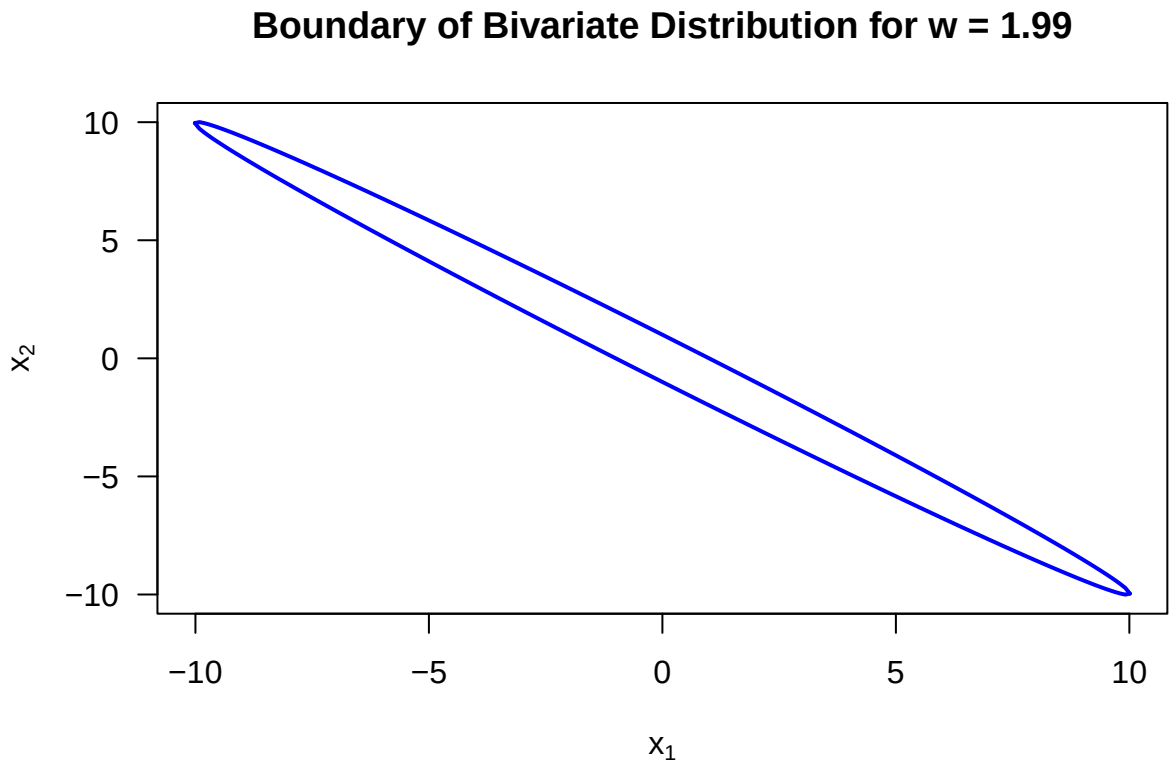
$$\frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$$

Matching this to the target function $f(x)$ we can derive that $\alpha = 5 + 1 = 6$ and $\beta = 1$. With the result $\Gamma(6) = (6-1)! = 120$ we get the required form.

The expected value of a gamma distribution is defined as $\frac{\alpha}{\beta}$. Therefore $E(X) = \frac{6}{1} = 6$. Contrary to expectations, the mean from part a is the closest.

Question 2

a) Task statement: Draw the boundaries of the region where X has a uniform distribution.



b) Task statement: What is the conditional distribution of X_1 given X_2 and that of X_2 given X_1? In order to derive the conditional distributions $X_1|X_2$ and $X_2|X_1$ we can rearrange the inequality as an equality to compute the boundaries of X_1 given X_2 and vice versa.

Since the joint distribution is symmetric we will show the proof for $X_1|X_2$ and infer that the same proof holds true for $X_2|X_1$.

The equality gives us:

$$x_1^2 + wx_1x_2 + x_2^2 = 1$$

We can rearrange this to a quadratic formula as shown below:

$$x_1^2 + wx_1x_2 + x_2^2 - 1 = 0$$

where $A = 1, B = wx_2$ and $C = x_2^2 - 1$.

Such a quadratic formula can be solved using the schema:

$$\frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

This equation yields two solutions which define the distribution as

$X_1|X_2 \sim Uniform(\frac{-wx_2 - \sqrt{wx_2^2 - 4(x_2^2-1)}}{2}, \frac{-wx_2 + \sqrt{wx_2^2 - 4(x_2^2-1)}}{2})$

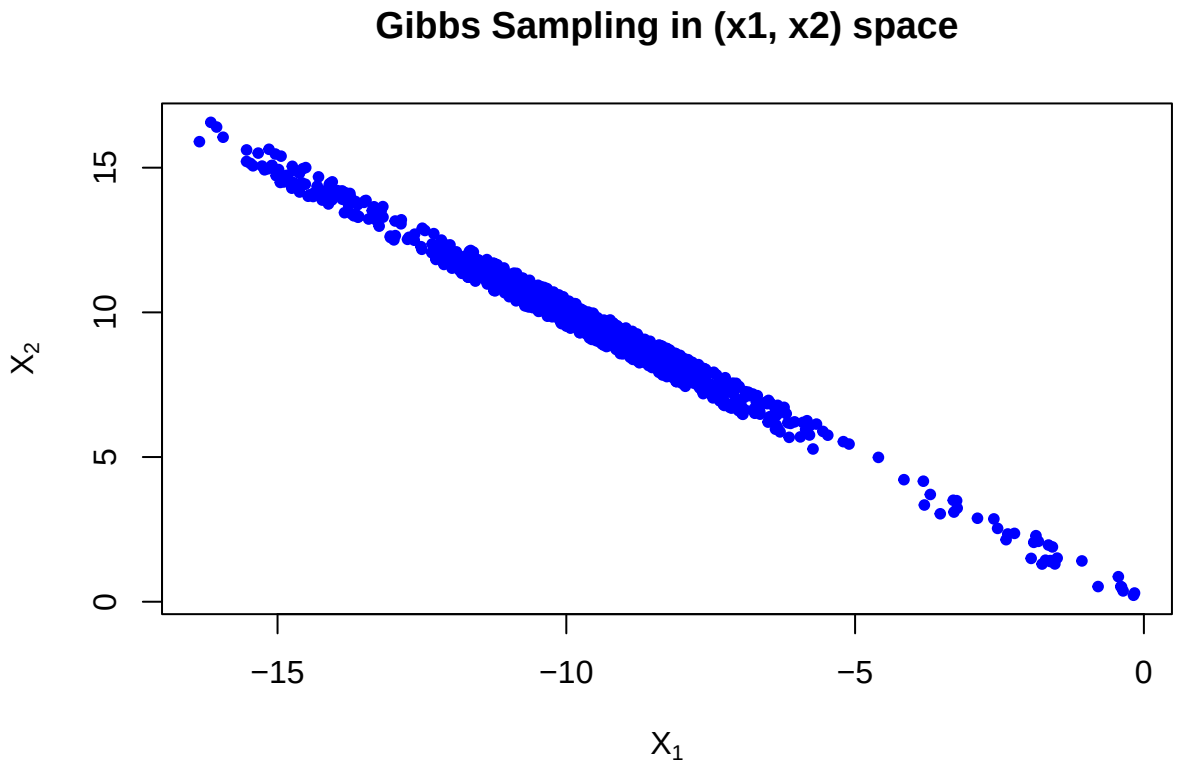
which in turn can be rewritten as

$$X_1|X_2 \sim Uniform(\frac{-wx_2 - \sqrt{x_2^2(w^2-4)+4}}{2}, \frac{-wx_2 + \sqrt{x_2^2(w^2-4)+4}}{2})$$

Thus

$$X_2|X_1 \sim Uniform(\frac{-wx_1 - \sqrt{x_1^2(w^2-4)+4}}{2}, \frac{-wx_1 + \sqrt{x_1^2(w^2-4)+4}}{2})$$

c) Task statement: 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. What should be the true result of this probability?



Since the distribution is symmetric around $X_1 = 0$, we expect the probability of $P(X_1 > 0)$ to be approximately 0.5. Based on repeated sampling, this assumption holds true.

d) Discuss, why the Gibbs sampling for this situation seems to be less successful for $w = 1.999$ compared to the case where $w = 1.8$ from the lecture. The range of possible values for w is $(-2; 2)$. The closer our w is to the boundaries of this region, the more narrow and stretched the ellipse becomes and for a positive w value X_1 and X_2 become almost perfectly negatively correlated, while for a negative w value X_1 and X_2 become almost perfectly negatively correlated.

This means, that given a value $x_1^{(0)}$ the subsequent value $x_2^{(0)}$ will fall in an increasingly smaller region, which in turn restricts the region that $x_1^{(1)}$ can fall in. This means that sampling at time step $t+1$ is correlated a lot closer to the samples at time step t than it would be for a smaller w value.

Hence, it takes more iterations for the Gibbs sampler to effectively "populate" the entire possible sample space.

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 $X_2 = (U_2 - U_1)/2$ and set this into the boundaries in terms of X_i. 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.

e)

We make the linear transformation $u_1 = x_1 - x_2$ and $u_2 = x_1 + x_2$.

Solving for x_1 and x_2 this yields: $x_1 = \frac{u_1+u_2}{2}$ and $x_2 = \frac{u_2-u_1}{2}$

Plugging this back into the original distribution we get:

$\frac{(u_1+u_2)^2}{4} + w\frac{u_1+u_2}{2} * \frac{u_2-u_1}{2} + \frac{(u_2-u_1)^2}{4} = 1$

This can be rearranged to:

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

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

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

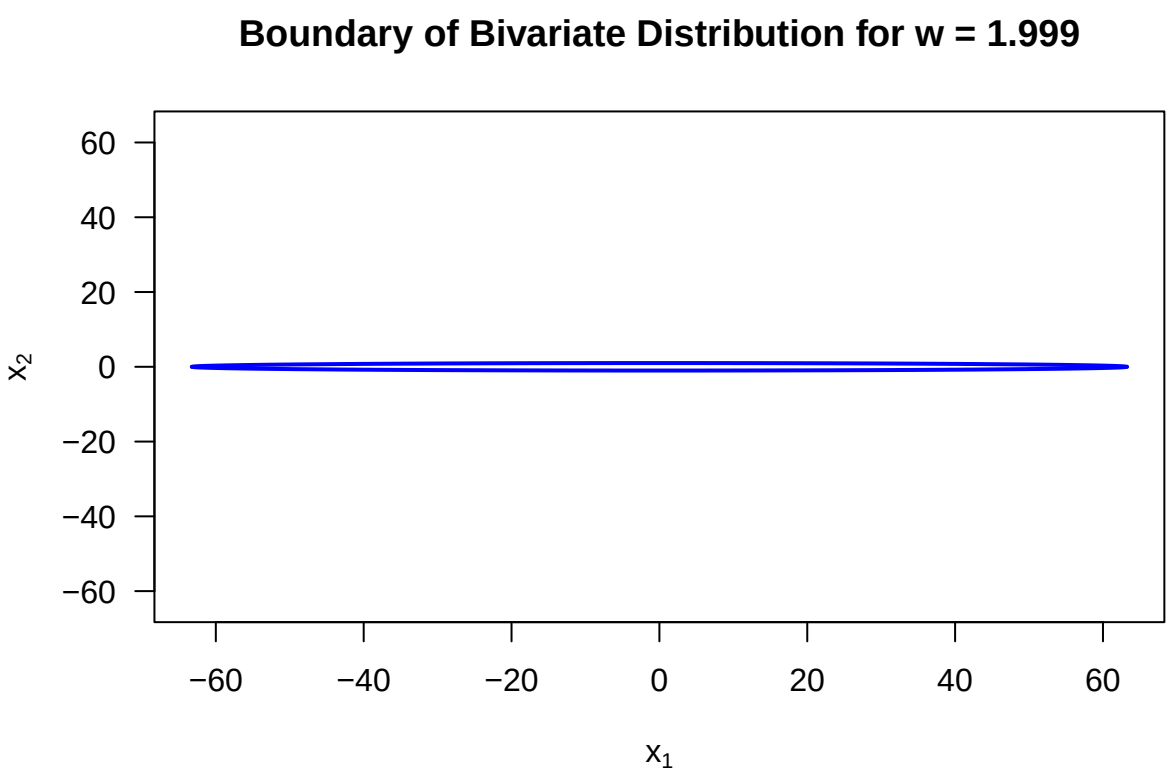
$\frac{u_1^2}{\frac{4}{2-w}} + \frac{u_2^2}{\frac{4}{2+w}} = 1$

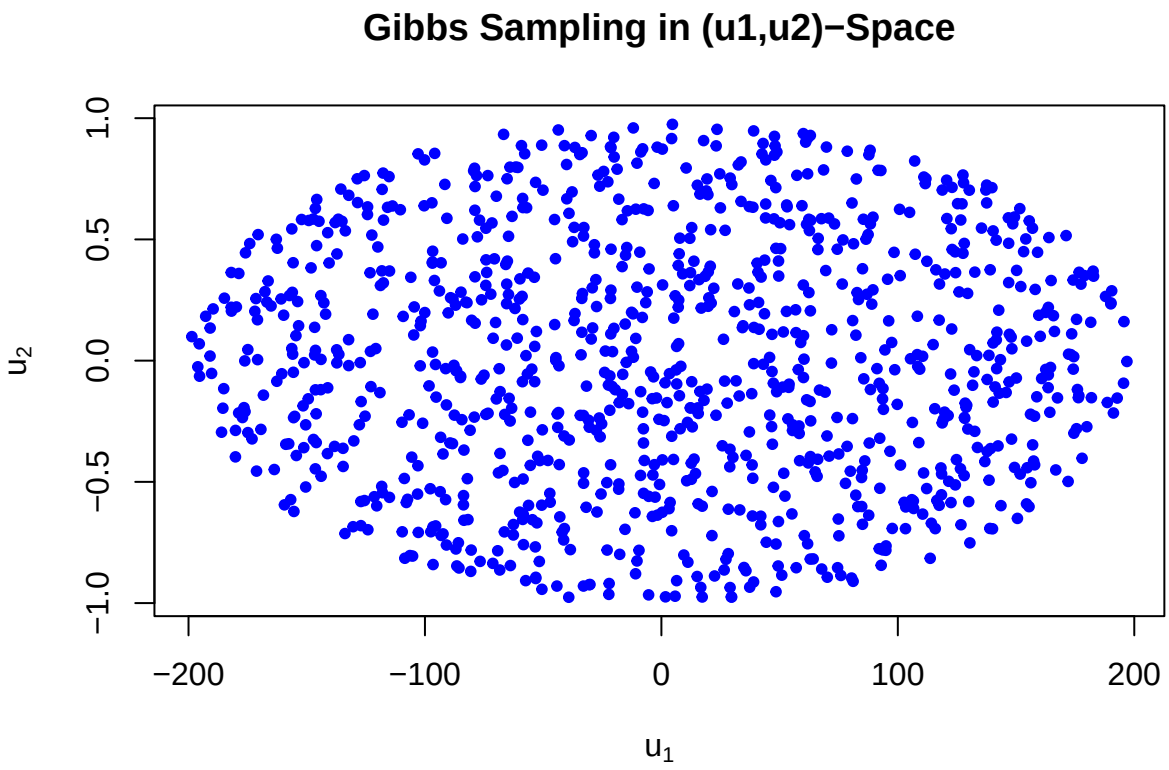
We substitute with $a^2 = \frac{4}{2-w}$ and $b^2 = \frac{4}{2+w}$

which gives $\frac{u_1^2}{a^2} + \frac{u_2^2}{b^2} = 1$

We can now solve this for u_2 and then iteratively calculate u_2 for a range of u_1s

$u_2 = \pm b\sqrt{1 - \frac{u_1^2}{a^2}}$


Boundary of Bivariate Distribution for w = 1.999
60
40
20
0
−20
−40
−60
x2
−60
−40
−20
0
20
40
60
x1



As one can deduct from the scatterplot, $P(\frac{U_1+U_2}{2} > 0) \approx 0.5$, this is in line with the finding that $P(X1 > 0) \approx 0.5$ as $\frac{U_1+U_2}{2}$ effectively is the same thing as X_1.

Appendix

```
f <- function(x){
  if (x > 0){
    return (120*x^5*exp(-x))
  }
  else{
    stop("Function undefined for x <= 0.")
  }
}
x <- seq(1,10,0.01)
x_app <- sapply(x,f)
plot(x, x_app, main = "Target function")

set.seed(42)

metropolis_normal <- function(n_iter = 10000, start = 1, proposal_sd = 0.1) {
  x <- numeric(n_iter)
  x[1] <- start
  accept <- 0
```

```r
  for (t in 2:n_iter) {
    proposal <- rnorm(1, mean = x[t - 1], sd = proposal_sd)

    # f undefined for x <= 0
    if (proposal > 0) {
      rate <- min(1, f(proposal) / f(x[t - 1]))
      if (runif(1) < rate) {
        x[t] <- proposal
        accept <- accept + 1
      } else {
        x[t] <- x[t - 1]
      }
    } else {
      x[t] <- x[t - 1]
    }
  }

  return(list(chain = x, acceptance_rate = accept / n_iter))
}

result_a <- metropolis_normal(n_iter=10000)
samples_a <- result_a$chain
acceptance_rate_a <- result_a$acceptance_rate
plot(samples_a, type = "l", main = "Trace Plot (Part a)", xlab = "Iteration", ylab = "X_t")
hist(samples_a, probability = TRUE, breaks = 50, main = "Histogram of Samples (Part a)", xlab = "X_t")

metropolis_chisq <- function(n_iter = 10000, start = 1) {
  x <- numeric(n_iter)
  x[1] <- start
  accept <- 0

  for (t in 2:n_iter) {
    proposal <- rchisq(1, floor(x[t - 1] + 1))
    rate <- min(1, f(proposal) / f(x[t - 1]))
    if (runif(1) < rate) {
      x[t] <- proposal
      accept <- accept + 1
    } else {
      x[t] <- x[t - 1]
    }
  }

  list(samples = x, acceptance_rate = accept / n_iter)
}

result_b <- metropolis_chisq()
samples_b <- result_b$samples
acceptance_rate_b <- result_b$acceptance_rate

plot(samples_b, type = "l", main = "Trace Plot (Part b)", xlab = "Iteration", ylab = "X_t")
hist(samples_b, probability = TRUE, breaks = 50, main = "Histogram of Samples (Part b)", xlab = "X_t")

metropolis_gamma <- function(n_iter = 10000, start = 1, alpha = 3, beta = 1) {
```

```r
  x <- numeric(n_iter)
  x[1] <- start
  accept <- 0

  for (t in 2:n_iter) {
    proposal <- rgamma(1, shape = alpha, rate = beta)
    rate <- min(1, f(proposal) / f(x[t - 1]))
    if (runif(1) < rate) {
      x[t] <- proposal
      accept <- accept + 1
    } else {
      x[t] <- x[t - 1]
    }
  }

  list(samples = x, acceptance_rate = accept / n_iter)
}

result_c <- metropolis_gamma()
samples_c <- result_c$samples
acceptance_rate_c <- result_c$acceptance_rate

plot(samples_c, type = "l", main = "Trace Plot (Part c)", xlab = "Iteration", ylab = "X_t")
hist(samples_c, probability = TRUE, breaks = 50, main = "Histogram of Samples (Part c)", xlab = "X_t")



estimate_expectation <- function(samples) {
  mean(samples)
}

cat("Estimated E(X) (Part a):", estimate_expectation(samples_a), "\n")
cat("Estimated E(X) (Part b):", estimate_expectation(samples_b), "\n")
cat("Estimated E(X) (Part c):", estimate_expectation(samples_c), "\n")

# Question 2:

# a)  Function to plot the boundary
plot_boundary <- function(w, col="blue") {
  # Range of x1 values
  xv <- seq(-1, 1, by=0.01) * 1/sqrt(1 - w^2 / 4)
  # Compute the corresponding x2 values
  lower_bound <- -(w*xv - sqrt(xv^2*(w^2-4)+4))/2
  upper_bound <- -(w*xv + sqrt(xv^2*(w^2-4)+4))/2


  plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1,
       main=paste("Boundary of Bivariate Distribution for w =", w))

  lines(xv, lower_bound, lwd=2, col=col)
  lines(xv, upper_bound, lwd=2, col=col)
}
```

```r
plot_boundary(1.9999)

# c) Gibbs sampling
# Helper function that samples one x-value given the other x-value

set.seed(1234) # reproducability
sample_dependent <- function(x_known, w) {
  lower_bound <- (-w * x_known - sqrt(x_known^2 * (w^2 - 4) + 4)) / 2
  upper_bound <- (-w * x_known + sqrt(x_known^2 * (w^2 - 4) + 4)) / 2
  runif(1, lower_bound, upper_bound) # random number between the bounds
}

# Gibbs Sampling function
gibbs_sampler <- function(n, w) {
  samples <- matrix(0, nrow=n, ncol=2)

  # Initial values (can be any valid point)
  x1 <- 0
  x2 <- 0

  for (i in 1:n) {
    x1 <- sample_dependent(x2, w)
    x2 <- sample_dependent(x1, w)
    samples[i, ] <- c(x1, x2)
  }

  return(samples)
}

n <- 1000
w <- 1.9999
samples <- gibbs_sampler(n, w)

# Plot the sampled points
plot(samples, col="blue", pch=20, xlab=expression(X[1]), ylab=expression(X[2]), main="Gibbs Sampling of

# e) Transformed boundary plotting
plot_boundary_transformed <- function(w, col="blue") {
  # Range of x1 values
  a_sq <- 4/(2-w)
  b <- sqrt(4/(2+w))
  a <- sqrt(a_sq)
  u1_seq <- seq(-a, a, length.out = 300)

  plot(u1_seq, u1_seq, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1,
       main=paste("Boundary of Bivariate Distribution for w =", w))


  lower_bound <- -b*sqrt(1-u1_seq^2/a_sq)
  upper_bound <- b*sqrt(1-u1_seq^2/a_sq)
  lines(u1_seq, lower_bound , lwd=2, col=col)
  lines(u1_seq, upper_bound, lwd=2, col=col)
}
```

```r
plot_boundary_transformed(w)

# Gibbs sampling from u
set.seed(1234) # For reproducability
sample_dependent <- function(x_known, w) {
  lower <- (-w * x_known - sqrt(1 - (1 - w^2/4) * x_known^2)) / 2
  upper <- (-w * x_known + sqrt(1 - (1 - w^2/4) * x_known^2)) / 2
  runif(1, lower, upper)
}

gibbs_sampler <- function(n, w) {
  samples <- matrix(0, nrow=n, ncol=2)

  # Initial values (can be any valid point)
  x1 <- 0
  x2 <- 0

  for (i in 1:n) {
    x1 <- sample_dependent(x2, w)
    x2 <- sample_dependent(x1, w)
    samples[i, ] <- c(x1, x2)
  }

  return(samples)
}

n <- 1000
w <- 1.9999
samples <- gibbs_sampler(n, w)

plot(samples, col="blue", pch=20, xlab=expression(X[1]), ylab=expression(X[2]), main="Gibbs Sampling in
```