Lab 4 report

Simon Jorstedt & Siddhesh Sreedar

2023-11-28

Question 1

We are given a gamma distribution with density f, where

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

Question 1.a, 1.b, 1.c, 1.d

We will implement the Metropolis-Hastings algorithm as a function where the proposal distribution can be specified. We will then run the algorithm to generate 10^4 samples each for the Log-Normal $LN(X_t, 1)$, the chi-squared $\chi^2_{\lfloor X_t+1 \rfloor}$, and the exponential Exp(1) distributions. The result of this is reported in Figures 1, 2 and 3 respectively.

In Figure 1, we see that the used Lognormal distribution has poor convergence qualities. It appears to spend many iterations rejecting new samples. In Figure 2 we see a seemingly much better result. The algorithm appears to accept most new samples and the samples fall into a fairly constant range throughout the sampling. When comparing the histograms of Figure 1 and 2, it is very clear that the lognormal proposal distribution leads to a heavily oversampled lower half of the true distribution (roughly the interval $[4, 5]$. This is also evident from the included true density curves. In Figure 3 we see a result that is very similar to that of Figure 2. The samples quickly fall into a fairly constant range of values, and the histogram of samples is very similar to the one generated from using the Chisquared proposal distribution in Figure 2. For all three proposal distributions, we have removed the first 50 generated values as an assumed burn-in period. That period is seemingly not long enough for the Lognormal proposal distribution, and clearly fully sufficient for the Chisquared and exponential proposal distributions.

Below we see the acceptance rate (converted into a percentage) for the three proposal distributions. This percentage indicates what proportion of sampled x values (including the burn-in-period) that are accepted by the algorithm. As expected, the LogNormal has a very low average acceptance rate, and both the Chisquared and Exponential proposal distributions have much higher average acceptance rates.

```
## LogNormal

## Acceptance percentage: 1.67 %


## Chisquared

## Acceptance percentage: 61.32 %


## Exponential

## Acceptance percentage: 40.06 %
```

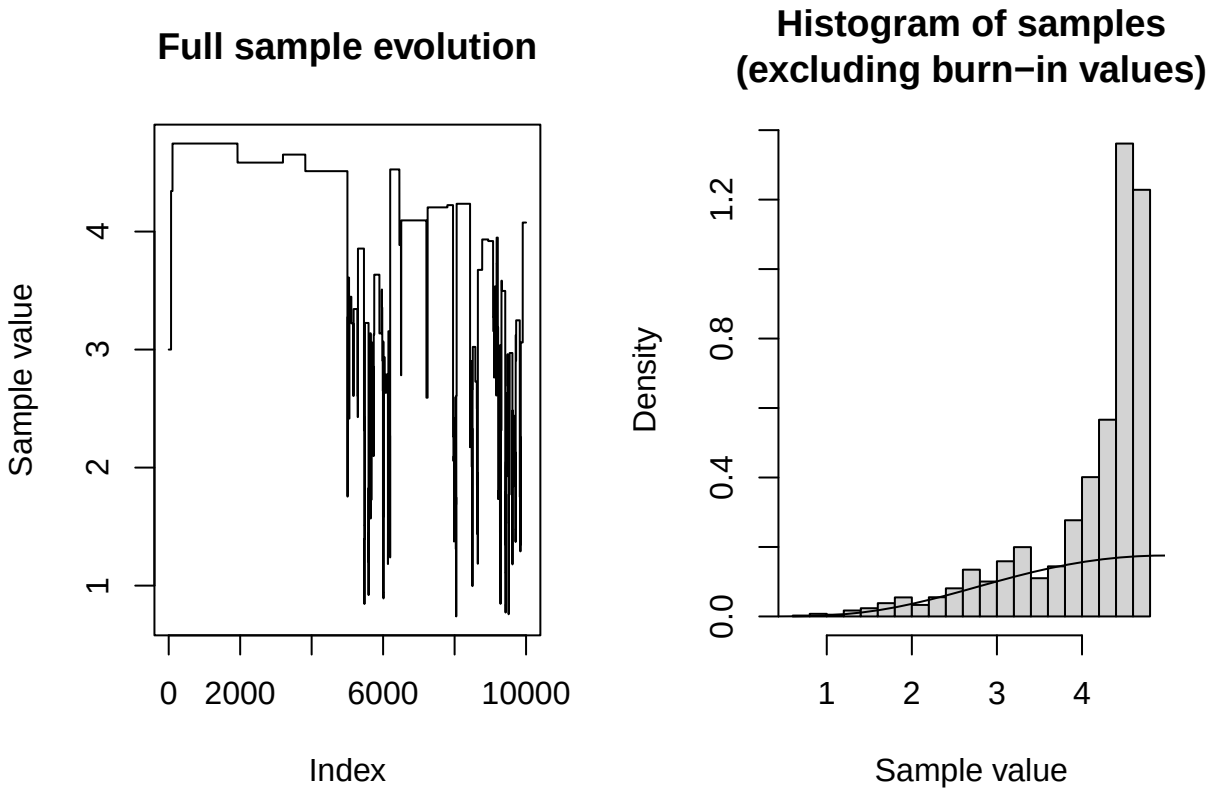


Figure 1: Lognormal porposal distribution

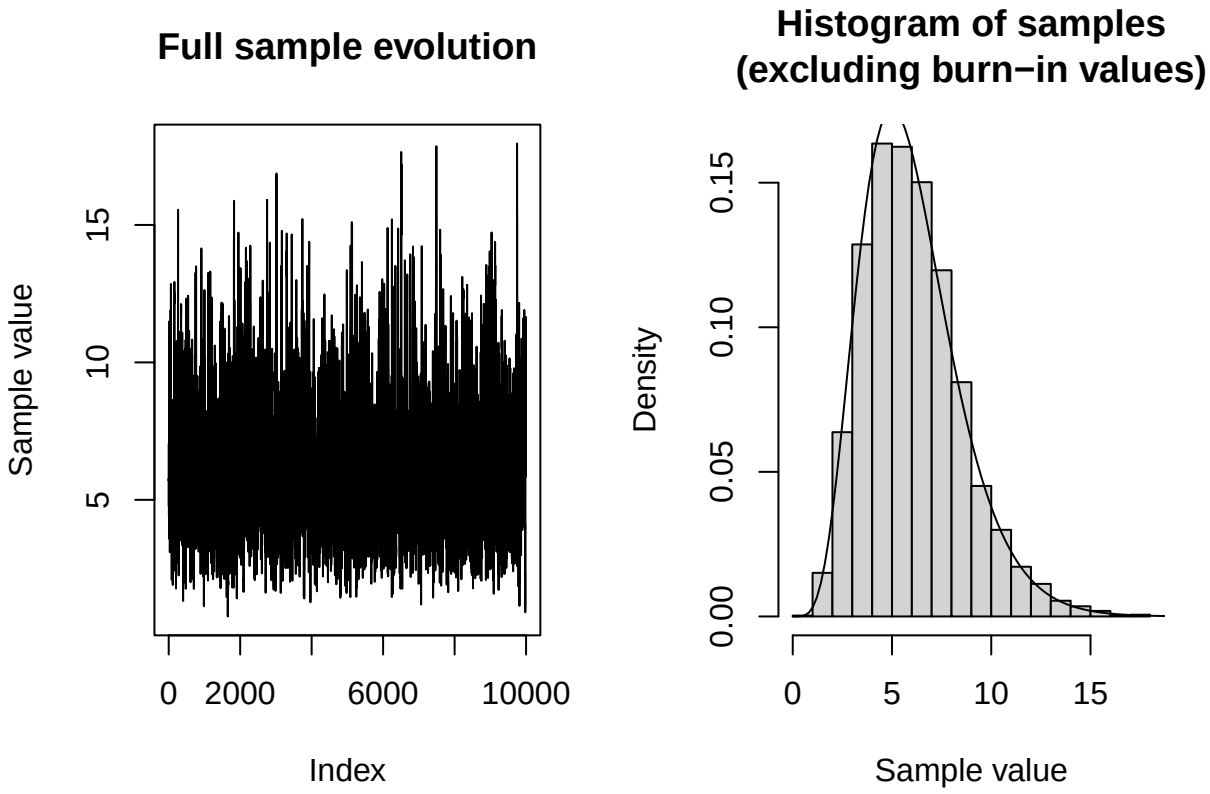


Figure 2: Chisquared proposal distribution

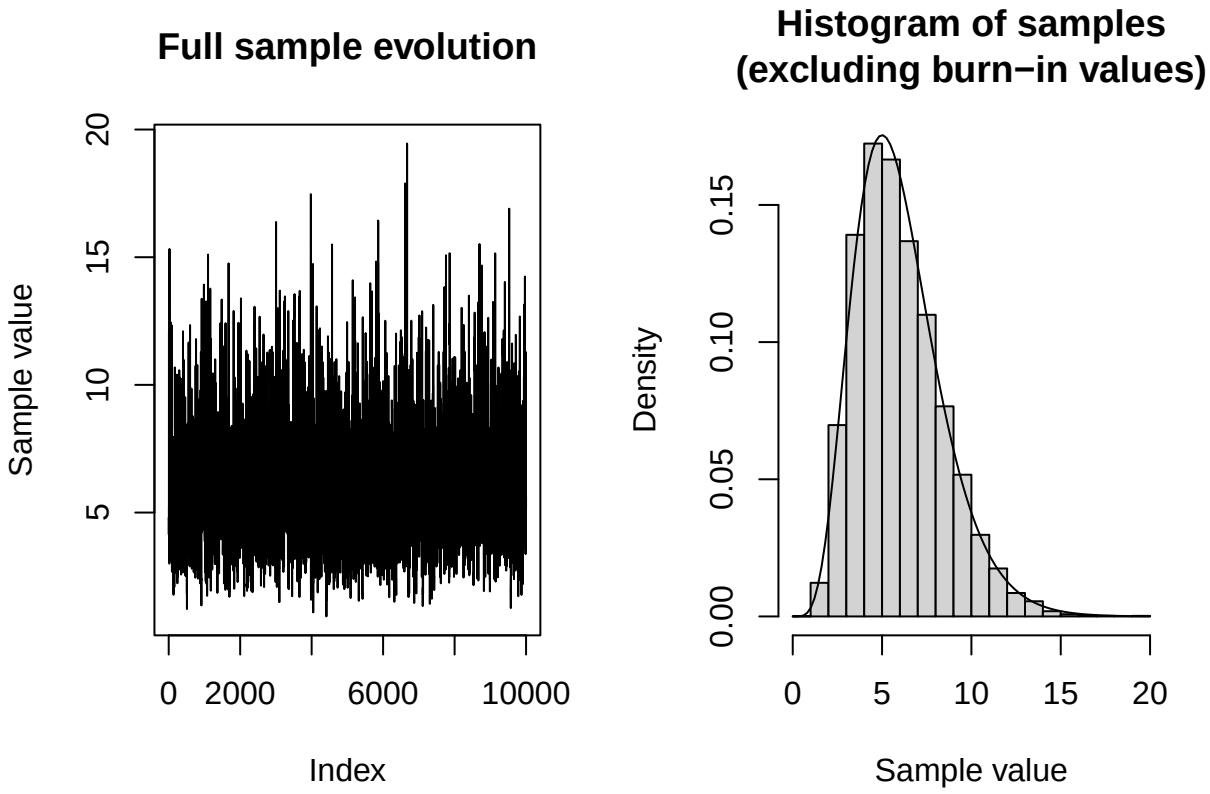


Figure 3: Exponential proposal distribution

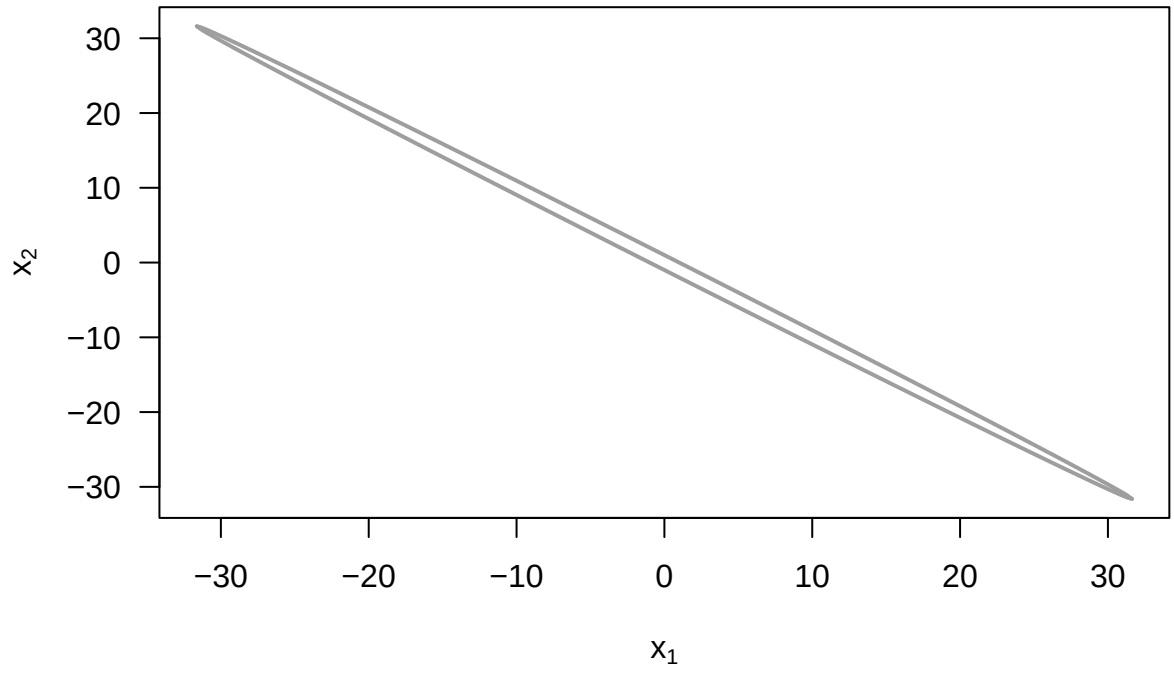
Question 1.e, 1.f

One can estimate the expected value from each of the samples by averaging over them, for the log normal proposal, this yields 4.09. For the chi-squared proposal, this yields 6.08. For the exponential proposal, this yields 5.97. The Gamma distribution of interest can quickly be identified as having parameters $k = 6$, and $\theta = 1$, and thus expected value 6. As expected from studying Figures 2 and 3, the Chisquared and Exponential distributions achieve good estimates of the expected value.

Question 2

a)

```
# Question 2.a


n=1000

w  <- 1.999
xv <- seq(-1, 1, by=0.01) * 1/sqrt(1-w^2/4) # we divide by "sqrt(1-w^2/4)" to ensure the the value belo
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)
```



b) x2 given x1: $(-(1.999/2)*x_1 - \sqrt{1-(1-1.999^2)/4)*x_1^2}, -(1.999/2)*x_1 + \sqrt{1-(1-1.999^2/4)*x_1^2})$

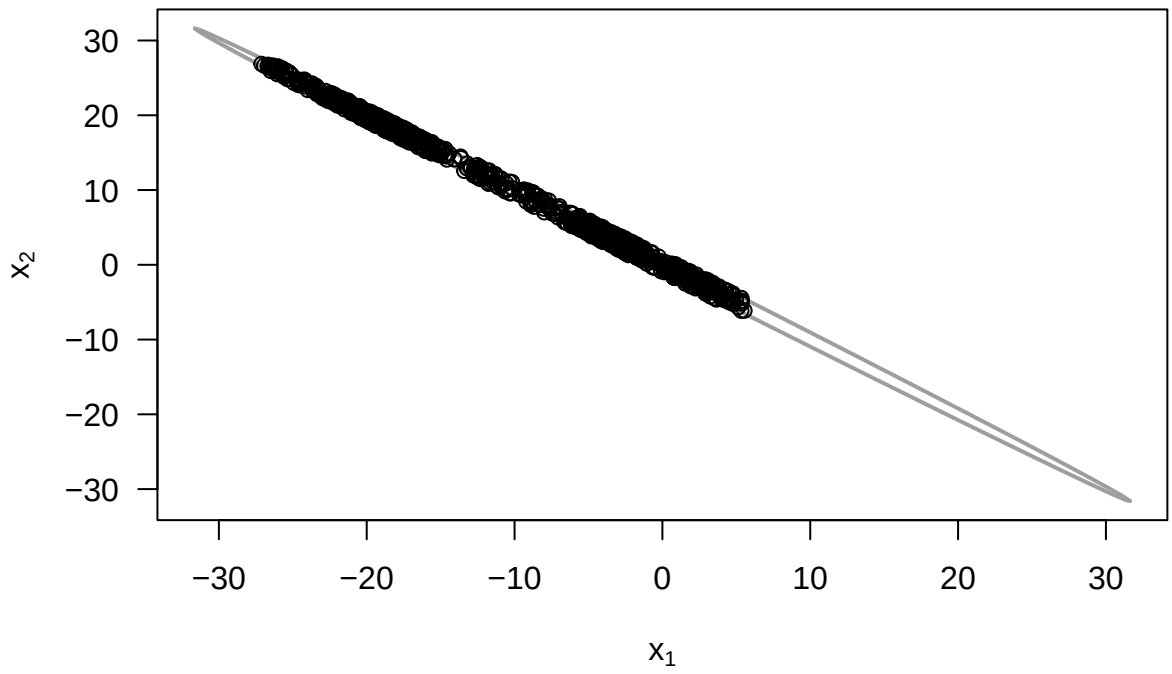
x1 given x2: $(-(1.999/2)*x_2 - \sqrt{1-(1-1.999^2/4)*x_2^2}, -(1.999/2)*x_2 + \sqrt{1-(1-1.999^2/4)*x_2^2})$

c)

```
# Question 2.c

one<-c()
two<-c()

gibbs<- function(x_one,x_two,n){

  one<-c()
  two<-c()

  i <- 2

  x1<-c(x_one)
  x2<-c(x_two)

  a<-list()
  b<-list()

  while (i < (n+1)){

    # x1 given x2
    a[i-1]<-list(c(-(1.999/2)*tail(x2,1)-sqrt(1-(1-1.999^2/4)*tail(x2,1)^2), -(1.999/2)*tail(x2,1)+sqrt
    x1[i]<-runif(1,min = a[[i-1]][1],max = a[[i-1]][2])

    #x2 given x1
    b[i-1]<-list(c(-(1.999/2)*tail(x1,1)-sqrt(1-(1-1.999^2/4)*tail(x1,1)^2), -(1.999/2)*tail(x1,1)+sqrt
    x2[i]<-runif(1,min = b[[i-1]][1],max = b[[i-1]][2])


    i<- i+1
  }
  one<<-x1
  two<<- x2
}
```

For starting values of 0 and 0, we plot the samples as well and determining P(X1 > 0):

```
# Question 2.c

gibbs(0,0,1000)
w  <- 1.999
xv <- seq(-1, 1, by=0.01) * 1/sqrt(1-w^2/4) # we divide by "sqrt(1-w^2/4)" to ensure the the value belo
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)
points(one,two)
```




```
sum(one>0)/n
```

```
## [1] 0.167
```

Determining P(X1 > 0) by repeating:

```
# Question 2.c

gibbs(0,0,1000)
sum(one>0)/n
```

```
## [1] 0.666
```

```
gibbs(0,0,1000)
sum(one>0)/n
```

```
## [1] 0.016
```

```
gibbs(0,0,1000)
sum(one>0)/n
```

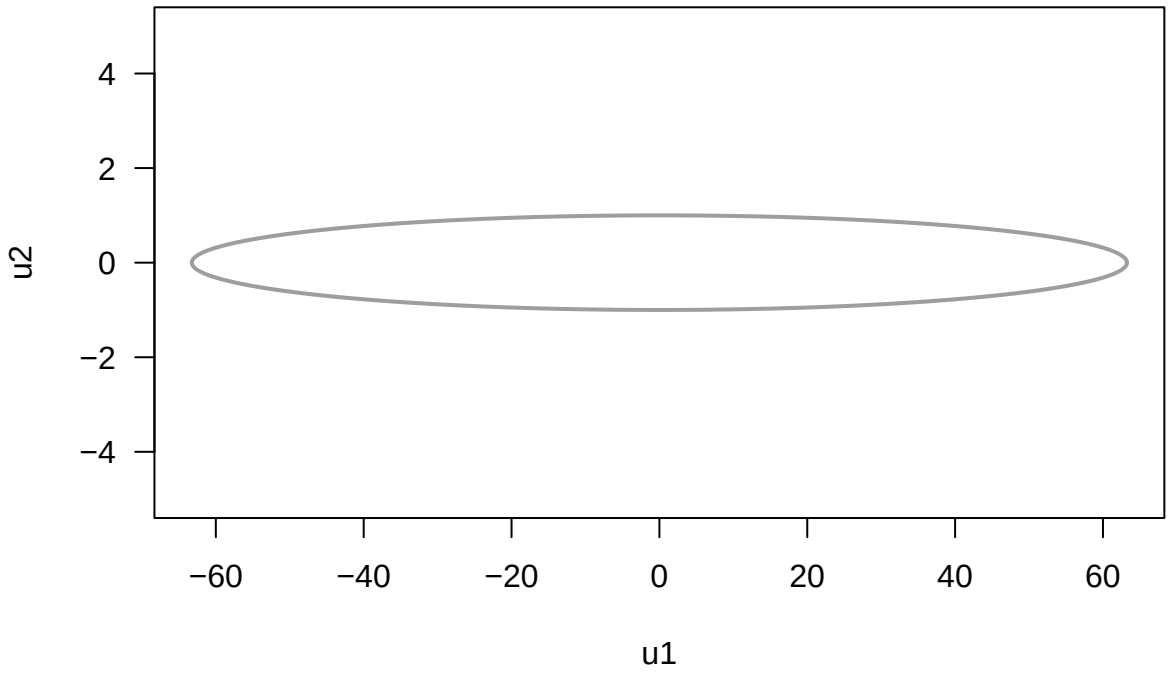
```
## [1] 0.989
```

The true probability would be the area to right of the ellipse starting from 0 divide by the total area of the ellipse. Since 0 is splits the ellipse equally, it should be around 0.5.

d) This is because, the samples given when w = 1.999 is more concentrated to the top left of the ellipse while when w = 1.8, the sampling values are more spread out along the ellipse. The axis limit is also larger (-30 to 30) compared to (-2 to 2). So it doesn't sample from the entire distribution but a portion.

e)

```
# Question 2.e

u1<-seq(-63.245,63.245,by=0.01)
uv <- u1
plot(uv, uv, type="n",xlab= "u1", ylab="u2", las=1,ylim = c(-5,5))
# ellipse
lines(uv, sqrt((4-0.001*uv^2)/3.999), lwd=2, col=8)
lines(uv, -sqrt((4-0.001*uv^2)/3.999), lwd=2, col=8)
```

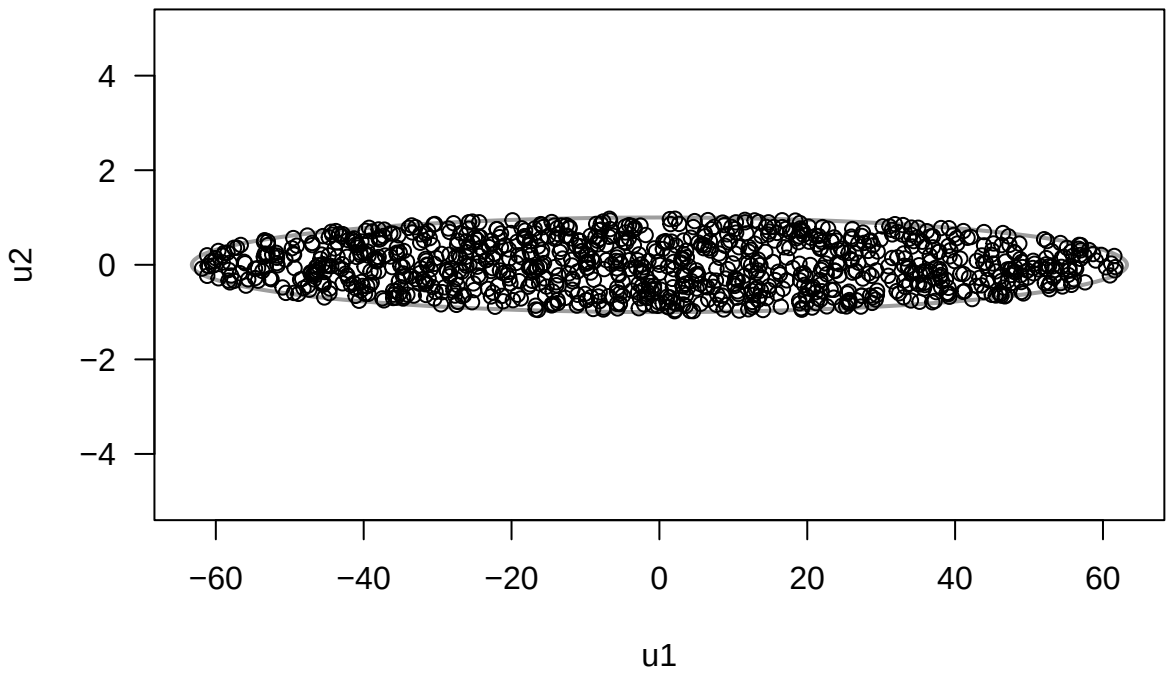



```
# Question 2.e

one_u<-c()
two_u<-c()

gibbs_2<- function(u_one,u_two,n){
```

```
  one_u<-c()
  two_u<-c()

  i <- 2

  u1<-c(u_one)
  u2<-c(u_two)

  a<-list()
  b<-list()

  while ( i < (n+1)){

    # u1 given u2
    a[i-1]<-list(c(-sqrt((4-3.999*tail(u2,1)^2)/0.001),sqrt((4-3.999*tail(u2,1)^2)/0.001)))
    u1[i]<-runif(1,min = a[[i-1]][1],max = a[[i-1]][2])

    #u2 given u1
    b[i-1]<-list(c(-sqrt((4-0.001*tail(u1,1)^2)/3.999),sqrt((4-0.001*tail(u1,1)^2)/3.999)))
    u2[i]<-runif(1,min = b[[i-1]][1],max = b[[i-1]][2])


    i<- i+1
  }
  one_u<<- u1
  two_u<<- u2
}
```

For starting values of 0 and 0, we plot the samples as well and determining P((U1 + U1)/2 > 0):

```
# Question 2.e

gibbs_2(0,0,1000)
u1<-seq(-63.245,63.245,by=0.01)
uv <- u1
plot(uv, uv, type="n",xlab= "u1", ylab="u2", las=1,ylim = c(-5,5))
# ellipse
lines(uv, sqrt((4-0.001*uv^2)/3.999), lwd=2, col=8)
lines(uv, -sqrt((4-0.001*uv^2)/3.999), lwd=2, col=8)
points(one_u,two_u)
```




```
sum((one_u+two_u)/2>0)/1000
```

```
## [1] 0.482
```

determining P((U1 + U1)/2 > 0) by repeating:

```
# Question 2.e
gibbs_2(0,0,1000)
sum((one_u+two_u)/2>0)/1000
```

```
## [1] 0.521
```

```
gibbs_2(0,0,1000)
sum((one_u+two_u)/2>0)/1000
```

```
## [1] 0.494
```

```
gibbs_2(0,0,1000)
sum((one_u+two_u)/2>0)/1000
```

```
## [1] 0.48
```

Upon comparing the probabilities, we can see that the probabilities are generally more consistent around 0.5 as compared to part c).

Appendix

```r
# Question 1.a, 1.b, 1.c

# Define a function proportional to the target density
f_propo <- function(x){if(all(x >= 0)){(1/120)*x^5 * exp(-x)} else{NA}}

# Create proposal density functions
lognorm_proposal <- function(x=NA, param, sample=F){
  if(sample == F){dlnorm(x, meanlog = param, sdlog = 1)}
  else{rlnorm(1, meanlog = param)}
}

chi_proposal <- function(x=NA, param, sample=F){
  if(sample == F){dchisq(x, df = floor(param+1))}
  else{rchisq(1, df=floor(param+1))}
}

new_proposal <- function(x=NA, param, sample=F){
  if(sample == F){dexp(x, rate=1/param)}
  else{rexp(1, rate=1/param)}
}


# Metropolis_Hasting function Instructions
#
# x_start is a 1-dim numeric indicating the starting value of the algorithm.
#
# prop_dens is a function which takes an argument `x`, an argument `param` specifying the model,
# and an optional argument `sample` which can be used to indicate that prop_dens
# should return one sampled number from the specified distribution.
# Otherwise prop_dens should return the distribution density at x.
#
# n is a numeric, indicating number of observations to take (including burn-in).
Metro_Hasting <- function(x_start, prop_dens, n=1) {
  # Setup
  x_t <- x_start
  x_values <- c()
  n_accepted <- 0

  # Generate samples until there are n of them
  repeat{
    # Generate new proposal
    x_new <- prop_dens(param=x_t, sample = T)
    #x_new <- rlnorm(1, meanlog = x_t, sdlog = 1)

    # Calculate probability value a
    # as product of Bayesian proposal ratio
    # and proposal density ratio (g(x_t | x_new) / g(x_new | x_t))
    Bay_pos_rat <- (f_propo(x_new) / f_propo(x_t))
    prop_ratio <- prop_dens(x=x_t, x_new) / prop_dens(x=x_new, x_t)
    #prop_ratio <- dlnorm(x_t, meanlog = x_new) / dlnorm(x_new, meanlog = x_t)
    a =  Bay_pos_rat * prop_ratio
```

```r
    # Evaluate proposal value
    # Accept the new value if u < a (even if a >= 1)
    # Reject the new value with probability 1-a (if a < 1)
    u <- runif(1)
    if(u < a){x_t <- x_new; n_accepted <- n_accepted + 1}

    # Store the new value (even if x_t has not changed)
    x_values <- c(x_values, x_t)

    # Check break condition
    if(length(x_values) >= 10^4){break}
  }

  cat("Acceptance percentage:", 100*n_accepted/n, "%\n")
  return(x_values)
}

# Question 1.d

# Generate Figure 1: Lognormal(x_t, 1) as proposal
## Generate x values
cat("LogNormal\n")
set.seed(43849664)
x_values_1 <- Metro_Hasting(3, prop_dens = lognorm_proposal, n = 10^4)

## Plot the chain, histogram and combine them
par(mfrow=c(1,2))
plot(x_values_1, type = "l", main="Full sample evolution", ylab = "Sample value")
hist(x_values_1[-c(1:51)],
     main="Histogram of samples\n(excluding burn-in values)",
     xlab = "Sample value",
     freq = F)
points(seq(0, 20, length=100), f_propo(seq(0, 20, length=100)), type="l")

# Question 1.d

# Generate Figure 2: Chisquared(floor(x_t+1)) as proposal
## Generate x values
cat("Chisquared\n")
set.seed(9427452)
x_values_2 <- Metro_Hasting(3, prop_dens = chi_proposal, n = 10^4)

## Plot the chain, histogram and combine them
par(mfrow=c(1,2))
plot(x_values_2, type = "l", main="Full sample evolution", ylab = "Sample value")
hist(x_values_2[-c(1:51)],
     main="Histogram of samples\n(excluding burn-in values)",
     xlab = "Sample value",
     freq=F)
points(seq(0, 20, length=100), f_propo(seq(0, 20, length=100)), type="l")

# Question 1.d
```

```r
# Generate Figure 3: Normal(x_t, sd=1) as proposal
## Generate x values
cat("Exponential\n")
set.seed(971614)
x_values_3 <- Metro_Hasting(5, prop_dens = new_proposal, n = 10^4)

## Plot the chain, histogram and combine them
par(mfrow=c(1,2))
plot(x_values_3, type = "l", main="Full sample evolution", ylab = "Sample value")
hist(x_values_3[-c(1:51)],
     main="Histogram of samples\n(excluding burn-in values)",
     xlab = "Sample value",
     freq=F)
points(seq(0, 20, length=100), f_propo(seq(0, 20, length=100)), type="l")

# Question 2.a


n=1000

w  <- 1.999
xv <- seq(-1, 1, by=0.01) * 1/sqrt(1-w^2/4) # we divide by "sqrt(1-w^2/4)" to ensure the the value belo
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)

# Question 2.c

one<-c()
two<-c()

gibbs<- function(x_one,x_two,n){

  one<-c()
  two<-c()

  i <- 2

  x1<-c(x_one)
  x2<-c(x_two)

  a<-list()
  b<-list()

  while (i < (n+1)){

    # x1 given x2
    a[i-1]<-list(c(-(1.999/2)*tail(x2,1)-sqrt(1-(1-1.999^2/4)*tail(x2,1)^2), -(1.999/2)*tail(x2,1)+sqrt
    x1[i]<-runif(1,min = a[[i-1]][1],max = a[[i-1]][2])

    #x2 given x1
    b[i-1]<-list(c(-(1.999/2)*tail(x1,1)-sqrt(1-(1-1.999^2/4)*tail(x1,1)^2), -(1.999/2)*tail(x1,1)+sqrt
```

```
    x2[i]<-runif(1,min = b[[i-1]][1],max = b[[i-1]][2])


    i<- i+1
  }
  one<<-x1
  two<<- x2
}

# Question 2.c

gibbs(0,0,1000)
w  <- 1.999
xv <- seq(-1, 1, by=0.01) * 1/sqrt(1-w^2/4) # we divide by "sqrt(1-w^2/4)" to ensure the the value belo
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)
points(one,two)
sum(one>0)/n

# Question 2.c

gibbs(0,0,1000)
sum(one>0)/n


gibbs(0,0,1000)
sum(one>0)/n

gibbs(0,0,1000)
sum(one>0)/n

# Question 2.e

u1<-seq(-63.245,63.245,by=0.01)
uv <- u1
plot(uv, uv, type="n",xlab= "u1", ylab="u2", las=1,ylim = c(-5,5))
# ellipse
lines(uv, sqrt((4-0.001*uv^2)/3.999), lwd=2, col=8)
lines(uv, -sqrt((4-0.001*uv^2)/3.999), lwd=2, col=8)

# Question 2.e

one_u<-c()
two_u<-c()

gibbs_2<- function(u_one,u_two,n){

  one_u<-c()
  two_u<-c()

  i <- 2
```

```
  u1<-c(u_one)
  u2<-c(u_two)

  a<-list()
  b<-list()

  while ( i < (n+1)){

    # u1 given u2
    a[i-1]<-list(c(-sqrt((4-3.999*tail(u2,1)^2)/0.001),sqrt((4-3.999*tail(u2,1)^2)/0.001)))
    u1[i]<-runif(1,min = a[[i-1]][1],max = a[[i-1]][2])

    #u2 given u1
    b[i-1]<-list(c(-sqrt((4-0.001*tail(u1,1)^2)/3.999),sqrt((4-0.001*tail(u1,1)^2)/3.999)))
    u2[i]<-runif(1,min = b[[i-1]][1],max = b[[i-1]][2])


    i<- i+1
  }
  one_u<<- u1
  two_u<<- u2
}

# Question 2.e

gibbs_2(0,0,1000)
u1<-seq(-63.245,63.245,by=0.01)
uv <- u1
plot(uv, uv, type="n",xlab= "u1", ylab="u2", las=1,ylim = c(-5,5))
# ellipse
lines(uv, sqrt((4-0.001*uv^2)/3.999), lwd=2, col=8)
lines(uv, -sqrt((4-0.001*uv^2)/3.999), lwd=2, col=8)
points(one_u,two_u)
sum((one_u+two_u)/2>0)/1000

# Question 2.e
gibbs_2(0,0,1000)
sum((one_u+two_u)/2>0)/1000


gibbs_2(0,0,1000)
sum((one_u+two_u)/2>0)/1000

gibbs_2(0,0,1000)
sum((one_u+two_u)/2>0)/1000
```