Honors Project

Grant Halver

April 24, 2019

Introduction

The Metropolis-Hastings Algorithm is an approximation of generating samples from a distribution $\pi(x)$. A move is accepted with probability $r(x,y) = min\{\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\}$. The transition probability is $p(x,y) = q(x,y)r(x,y)$. This may be used to approximate samples from any distribution, like a geometric distribution, but the method is more useful for continuous distributions like the Rayleigh Distribution.

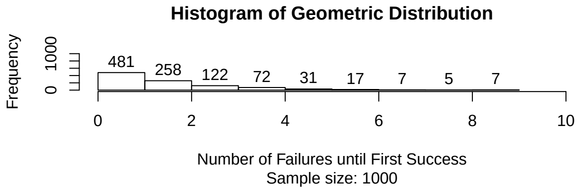
Geometric Distribution

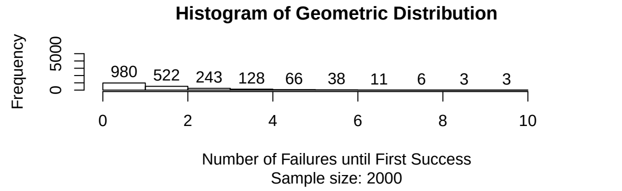
For the geometric distribution, we can check that the Metropolis-Hastings Algorithm produces a distribution that converges to the geometric distribution. A geometric distribution has $\pi(x) = \theta^x(1-\theta)$ for $x = 1, 2, 3, ...$ To generate the jumps, a symmetric random walk $q(x, x+1) = q(x, x-1) = 1/2$ will be used. By symmetry, $r(x,y) = min\{\pi(y)/\pi(x), 1\}$. When $x > 0$, then $\pi(x-1) > \pi(x)$ so $\pi(x-1)/\pi(x) > 1$ and $\pi(x+1)/\pi(x) = \theta$. Thus, $p(x, x-1) = 1/2$, $p(x+1, x) = \theta/2$, and $p(x,x) = (1-\theta)/2$. When x = 0, x cannot decrease any further due to the properties of the geometric distribution so $p(0,1) = \theta/2$ and $p(0,1) = 1 - \theta/2$.

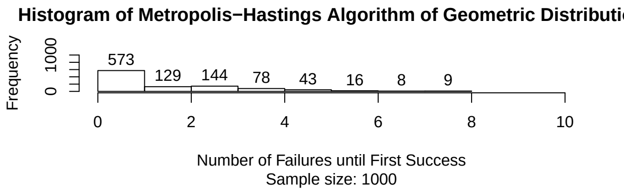
In this example, a geometric distribution with parameter $\theta = 0.5$ was used. Each datapoint on the graph is taken 1000 steps after the the last datapoint. From the graphs below, we can see that the Metropolis-Hasting Algorithm appears to converge to the geometric distribution when the sample size is about 10,000.

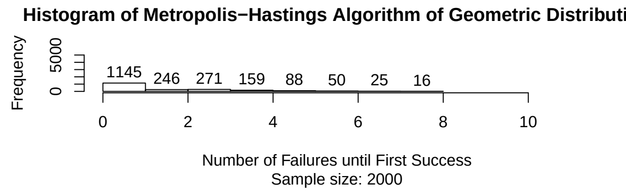
```
theta = 0.5
x = 0
p_Dec = 0.5
p_Same = theta/2
p_Inc = 1 - p_Dec - p_Same
result <- matrix(data = NA, nrow = 1, ncol = 1)
Size <- c(1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000,
    1000, 10000, 10000, 10000, 10000)

for (s in Size) {
    for (i in 1:s * 1000) {
        check = runif(1, 0, 100)

        # Case x = 0
        if (x == 0) {
            if (check < 100 * (p_Dec + p_Same)) {
                x = 0
            } else {
                x = 1
            }
        }

        # x is positive
        if (x != 0) {
            check = runif(1, 0, 100)
            if (check < 100 * p_Dec) {
                x = x - 1
            } else {
```

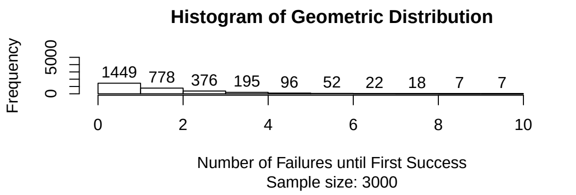
```
                if (check >= 100 * (p_Dec + p_Same)) {
                  x = x + 1
                }
            }
        }

        if (i%%1000 == 0) {
            if (is.element(NA, result)) {
                result = c(x)
            } else {
                result = array(c(result, x))
            }
        }
    }

    sample = length(result)
    geometric <- rgeom(sample, 0.5)

    if (max(table(geometric)) < max(table(result))) {
        maxY = max(table(result))
    } else {
        maxY = max(table(geometric))
    }

    mag = 10
    i = 1
    while (maxY%/%mag >= 10) {
        i = i + 1
        mag = 10^i
    }
    if (maxY%/%mag >= 4) {
        mag = mag * 10
    } else {
        mag = mag * 5
    }

    if (tail(sort(result), 1) < tail(sort(geometric), 1)) {
        maxX = tail(sort(geometric), 1)
    } else {
        maxX = tail(sort(result), 1)
    }

    par(mfrow = c(2, 1))
    hist(geometric, breaks = tail(sort(geometric), 1), main = "Histogram of Geometric Distribution",
        xlim = c(0, maxX + 1), ylim = c(0, mag), sub = paste("Sample size:",
            toString(sample)), labels = TRUE, xlab = "Number of Failures until First Success",
        right = F)
    hist(result, breaks = tail(sort(result), 1), main = "Histogram of Metropolis-Hastings Algorithm of
        xlim = c(0, maxX + 1), ylim = c(0, mag), sub = paste("Sample size:",
            toString(sample)), labels = TRUE, xlab = "Number of Failures until First Success",
        right = F)
}
```

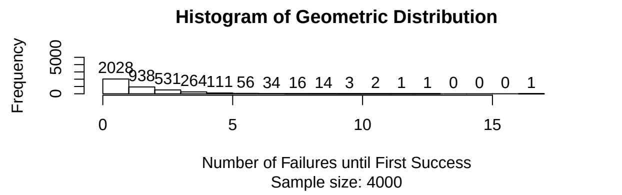

Histogram of Geometric Distribution
Frequency
0 1000
481 258 122 72 31 17 7 5 7
0 2 4 6 8 10
Number of Failures until First Success
Sample size: 1000

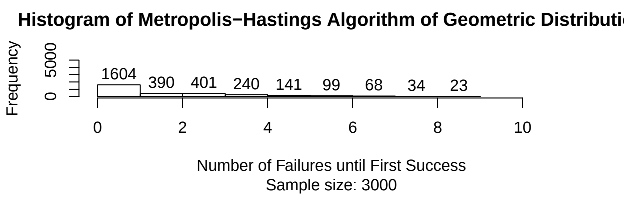

Histogram of Geometric Distribution
Frequency
0 5000
980 522 243 128 66 38 11 6 3 3
0 2 4 6 8 10
Number of Failures until First Success
Sample size: 2000

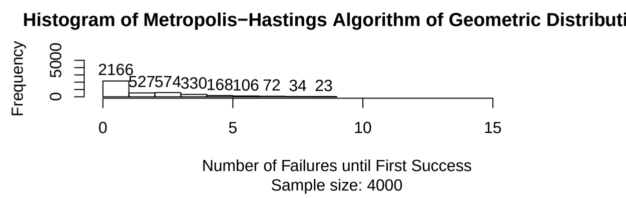

Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0 1000
573 129 144 78 43 16 8 9
0 2 4 6 8 10
Number of Failures until First Success
Sample size: 1000

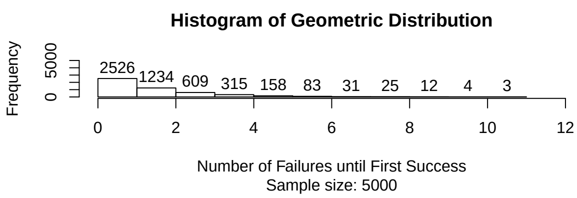

Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0 5000
1145 246 271 159 88 50 25 16
0 2 4 6 8 10
Number of Failures until First Success
Sample size: 2000

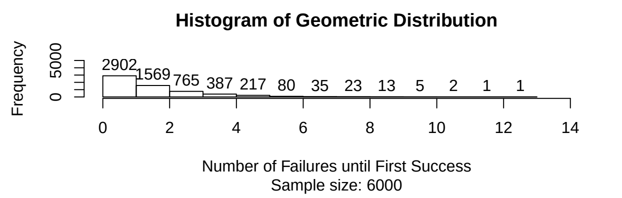

Histogram of Geometric Distribution
Frequency
0 5000
1449 778 376 195 96 52 22 18 7 7
0 2 4 6 8 10
Number of Failures until First Success
Sample size: 3000

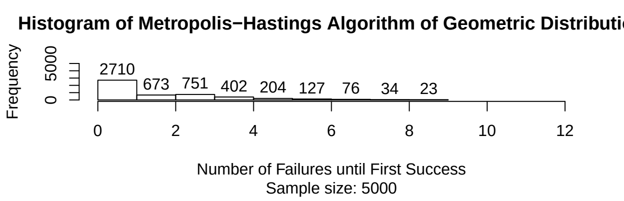

Histogram of Geometric Distribution
Frequency
0 5000
2028 938 531 264 111 56 34 16 14 3 2 1 1 0 0 0 1
0 5 10 15
Number of Failures until First Success
Sample size: 4000

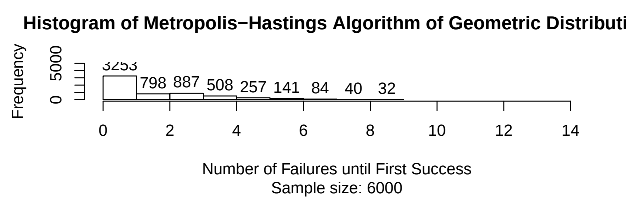

Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0 5000
1604 390 401 240 141 99 68 34 23
0 2 4 6 8 10
Number of Failures until First Success
Sample size: 3000

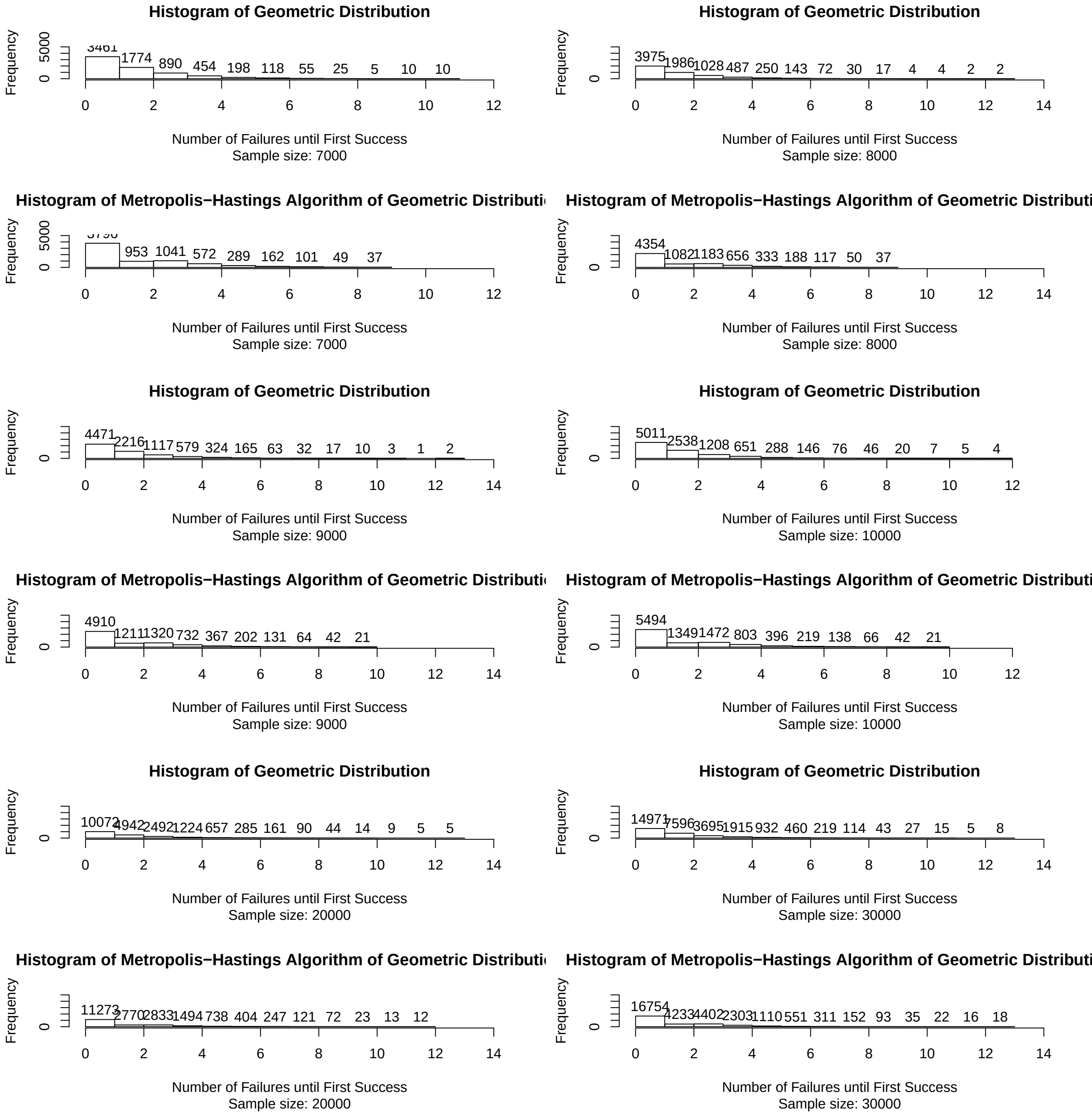

Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0 5000
2166 527 574 330 168 106 72 34 23
0 5 10 15
Number of Failures until First Success
Sample size: 4000

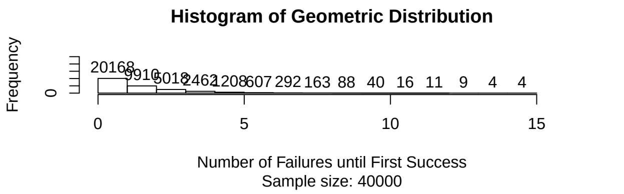

Histogram of Geometric Distribution
Frequency
0 5000
2526 1234 609 315 158 83 31 25 12 4 3
0 2 4 6 8 10 12
Number of Failures until First Success
Sample size: 5000

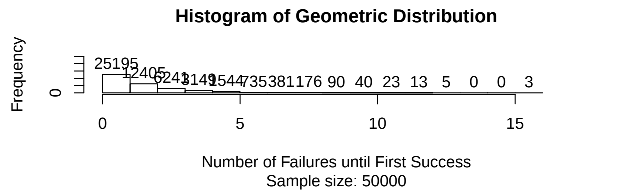

Histogram of Geometric Distribution
Frequency
0 5000
2902 1569 765 387 217 80 35 23 13 5 2 1 1
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 6000

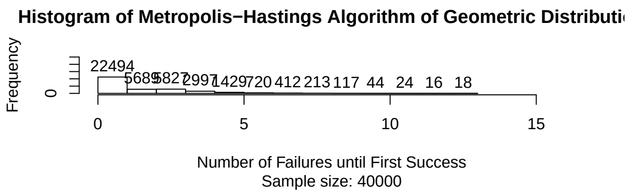

Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0 5000
2710 673 751 402 204 127 76 34 23
0 2 4 6 8 10 12
Number of Failures until First Success
Sample size: 5000

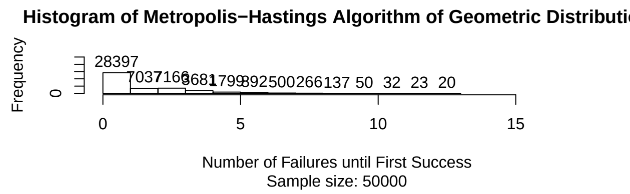

Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0 5000
3253 798 887 508 257 141 84 40 32
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 6000


Histogram of Geometric Distribution
Frequency
0 5000
1774 890 454 198 118 55 25 5 10 10
0 2 4 6 8 10 12
Number of Failures until First Success
Sample size: 7000
Histogram of Geometric Distribution
Frequency
0
3975 1986 1028 487 250 143 72 30 17 4 4 2 2
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 8000
Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0 5000
953 1041 572 289 162 101 49 37
0 2 4 6 8 10 12
Number of Failures until First Success
Sample size: 7000
Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0
4354 1082 1183 656 333 188 117 50 37
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 8000
Histogram of Geometric Distribution
Frequency
0
4471 2216 1117 579 324 165 63 32 17 10 3 1 2
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 9000
Histogram of Geometric Distribution
Frequency
0
5011 2538 1208 651 288 146 76 46 20 7 5 4
0 2 4 6 8 10 12
Number of Failures until First Success
Sample size: 10000
Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0
4910 1211 1320 732 367 202 131 64 42 21
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 9000
Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0
5494 1349 1472 803 396 219 138 66 42 21
0 2 4 6 8 10 12
Number of Failures until First Success
Sample size: 10000
Histogram of Geometric Distribution
Frequency
0
10072 4942 2492 1224 657 285 161 90 44 14 9 5 5
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 20000
Histogram of Geometric Distribution
Frequency
0
14971 7596 3695 1915 932 460 219 114 43 27 15 5 8
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 30000
Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0
11273 2770 2833 1494 738 404 247 121 72 23 13 12
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 20000
Histogram of Metropolis−Hastings Algorithm of Geometric Distributi
Frequency
0
16754 4233 4402 2303 1110 551 311 152 93 35 22 16 18
0 2 4 6 8 10 12 14
Number of Failures until First Success
Sample size: 30000





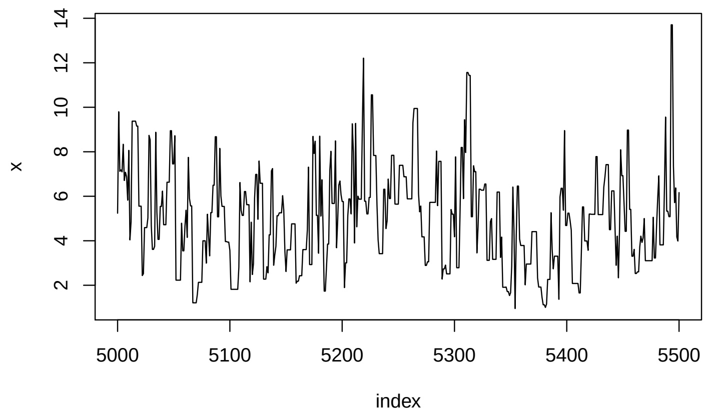




Rayleigh Distribution

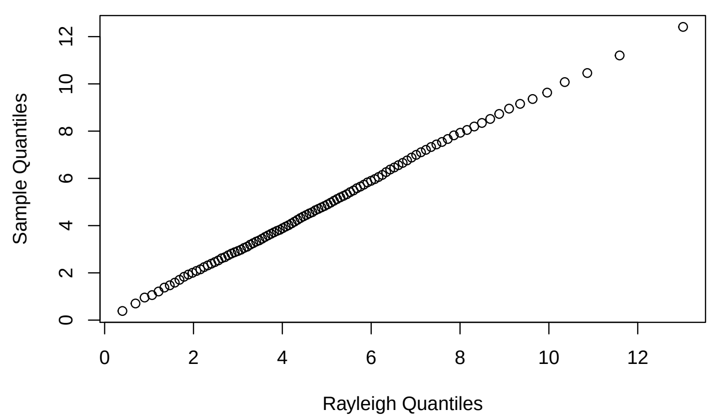
However, the main purpose of the Metropolis-Hastings algorithm is to approximate continuous distributions so we shall also make plots for a continuous distribution, the Rayleigh Distribution.

```
f <- function(x, sigma) {
    if (any(x < 0))
        return(0)
    stopifnot(sigma > 0)
    return((x/sigma^2) * exp(-x^2/(2 * sigma^2)))
}

m <- 10000
sigma <- 4
x <- numeric(m)
x[1] <- rchisq(1, df = 1)
k <- 0
u <- runif(m)

for (i in 2:m) {
    xt <- x[i - 1]
    y <- rchisq(1, df = xt)
    num <- f(y, sigma) * dchisq(xt, df = y)
    den <- f(xt, sigma) * dchisq(y, df = xt)
    if (u[i] <= num/den)
        x[i] <- y else {
        x[i] <- xt
        k <- k + 1  #y is rejected
    }
}

index <- 5000:5500
y1 <- x[index]
plot(index, y1, type = "l", main = "", ylab = "x")
```



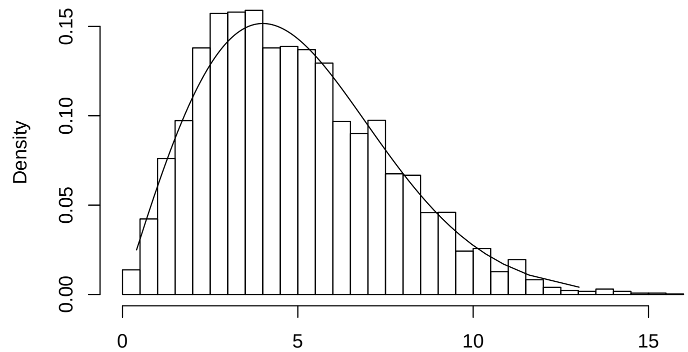
This plot gives the time index of the Metropolis-Hastings algorithm along the longitudinal axis and the x value along the latitudinal axis. Notice that when the next move is rejected, there are horizontal lines where the x value does not change.

```
b <- 2001  #discard the burnin sample
y <- x[b:m]
a <- ppoints(100)
QR <- sigma * sqrt(-2 * log(1 - a))  #quantiles of Rayleigh
Q <- quantile(x, a)

qqplot(QR, Q, main = "", xlab = "Rayleigh Quantiles", ylab = "Sample Quantiles")
```



Notice that the data appear to follow a linear trend, so the data generate using the Metropolis-Hastings Algorithm appear to follow the Rayleigh distribution.

```
hist(y, breaks = "scott", main = "", xlab = "", freq = FALSE)
lines(QR, f(QR, 4))
```



On this histogram we can see that the data are distributed approximately the same as the Rayleigh distribution. Hence, the Metropolis-Hastings algorithm generated data that follow the Rayleigh distribution.

Conclusion

The Metropolis-Hastings algorithm generates samples that follow the desired distribution. This can be used for both discrete and continuous distributions.