hw8

陆靖磊 22300680221

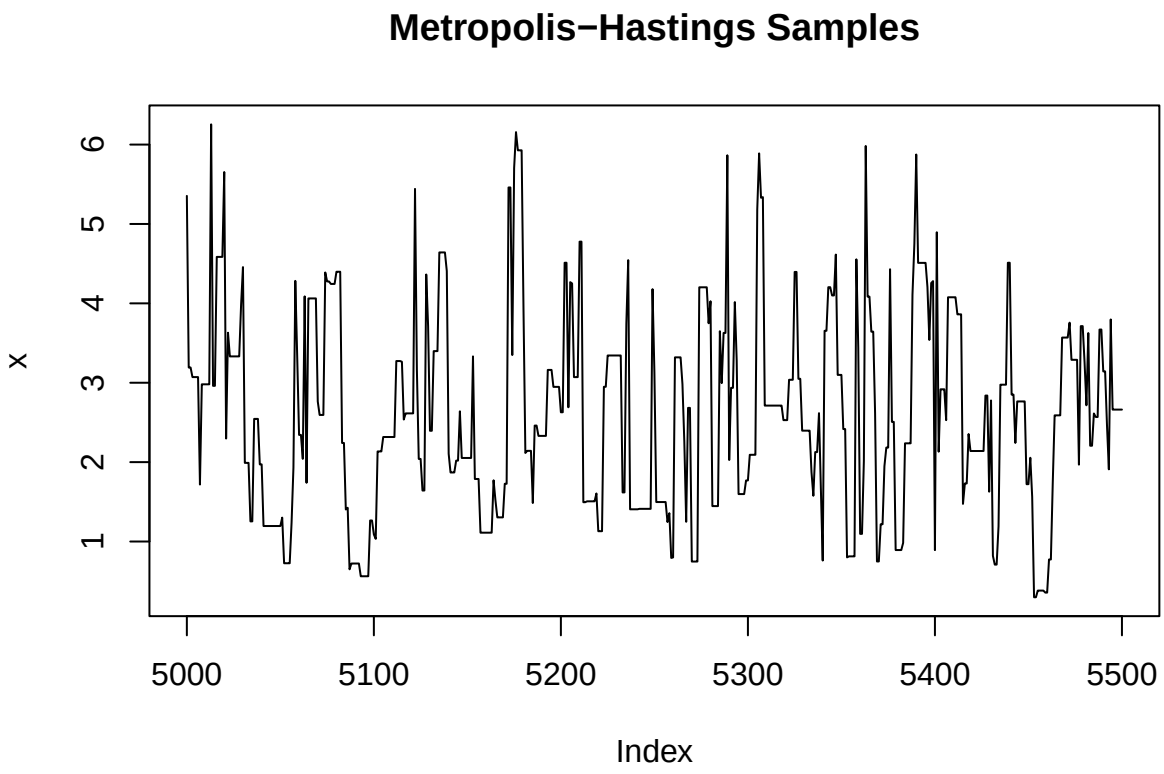
2024-12-04

11.1

```
num_samples <- 10000
sigma <- 2

samples <- numeric(num_samples)
samples[1] <- rchisq(1, df = 1)
rejection_count <- 0
uniform_samples <- runif(num_samples)

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

for (i in 2:num_samples) {
  current_sample <- samples[i - 1]
  proposed_sample <- rchisq(1, df = current_sample)

  numerator <- f(proposed_sample, sigma) * dchisq(current_sample, df = proposed_sample)
  denominator <- f(current_sample, sigma) * dchisq(proposed_sample, df = current_sample)
  acceptance_prob <- numerator / denominator

  if (uniform_samples[i] <= acceptance_prob) {
    samples[i] <- proposed_sample
  } else {
    samples[i] <- current_sample
    rejection_count <- rejection_count + 1
  }
}

print(rejection_count)
```

```
## [1] 5433
```

```
par(mfrow = c(1, 1))
index <- 5000:5500
subset_samples <- samples[index]
plot(index, subset_samples, type = "l", main = "Metropolis-Hastings Samples", ylab = "x", xlab = "Index
```



相比于图 11.1，更多的样本被拒绝，有效性降低

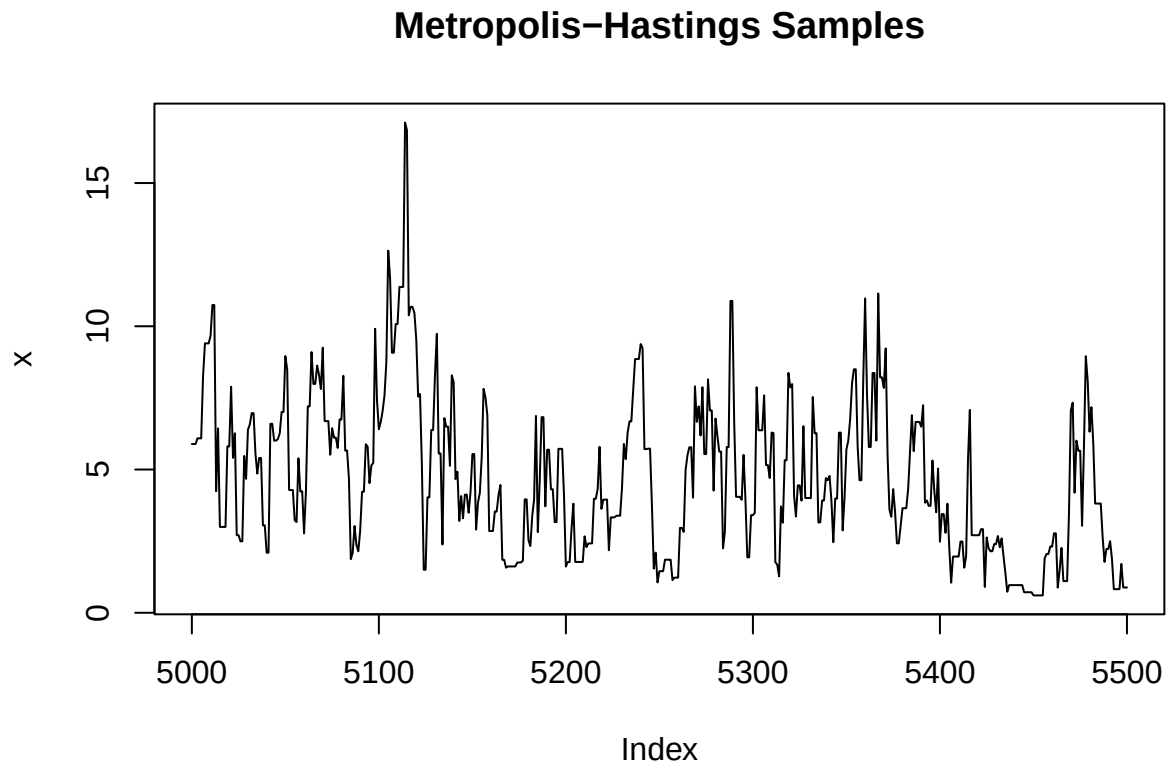
11.2

```
num_samples <- 10000
sigma <- 4
initial_value <- 1

samples <- numeric(num_samples)
samples[1] <- initial_value
rejection_count <- 0
uniform_samples <- runif(num_samples)

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

for (i in 2:num_samples) {
  current_sample <- samples[i - 1]
  proposed_sample <- rgamma(1, current_sample, 1)

  numerator <- target_distribution(proposed_sample, sigma) * dgamma(current_sample, proposed_sample, 1)
  denominator <- target_distribution(current_sample, sigma) * dgamma(proposed_sample, current_sample, 1
  acceptance_prob <- numerator / denominator
```

```
  if (uniform_samples[i] <= acceptance_prob) {
    samples[i] <- proposed_sample
  } else {
    samples[i] <- current_sample
    rejection_count <- rejection_count + 1
  }
}

print(rejection_count)
```

```
## [1] 3083
```

```
subset_index <- 5000:5500
subset_samples <- samples[subset_index]
plot(subset_index, subset_samples, type = "l", main = "Metropolis-Hastings Samples", ylab = "x", xlab =
```



11.3 (1)

```
num_samples <- 10000
sigma <- 3

samples <- numeric(num_samples)
samples[1] <- rnorm(1, 0, sigma)
rejection_count <- 0
uniform_samples <- runif(num_samples)
```

```r
# Metropolis-Hastings algorithm
for (i in 2:num_samples) {
  current_sample <- samples[i - 1]
  proposed_sample <- rnorm(1, current_sample, sigma)

  numerator <- 1 + current_sample^2
  denominator <- 1 + proposed_sample^2
  numerator <- numerator * dnorm(current_sample, proposed_sample, sigma)
  denominator <- denominator * dnorm(proposed_sample, current_sample, sigma)
  acceptance_prob <- numerator / denominator

  if (uniform_samples[i] <= acceptance_prob) {
    samples[i] <- proposed_sample
  } else {
    samples[i] <- current_sample
    rejection_count <- rejection_count + 1
  }
}

# Print the number of rejections
print(rejection_count)
```

```
## [1] 4472
```

(2)

```r
num_samples <- 10000
sigma <- 3
burn_in <- 1000

samples <- numeric(num_samples)
samples[1] <- rnorm(1, 0, sigma)
rejection_count <- 0
uniform_samples <- runif(num_samples)

# Metropolis-Hastings algorithm
for (i in 2:num_samples) {
  current_sample <- samples[i - 1]
  proposed_sample <- rnorm(1, current_sample, sigma)

  numerator <- 1 + current_sample^2
  denominator <- 1 + proposed_sample^2
  numerator <- numerator * dnorm(current_sample, proposed_sample, sigma)
  denominator <- denominator * dnorm(proposed_sample, current_sample, sigma)
  acceptance_prob <- numerator / denominator

  if (uniform_samples[i] <= acceptance_prob) {
    samples[i] <- proposed_sample
  } else {
    samples[i] <- current_sample
    rejection_count <- rejection_count + 1
  }
```

```
}

print(rejection_count)
```

```
## [1] 4863
```

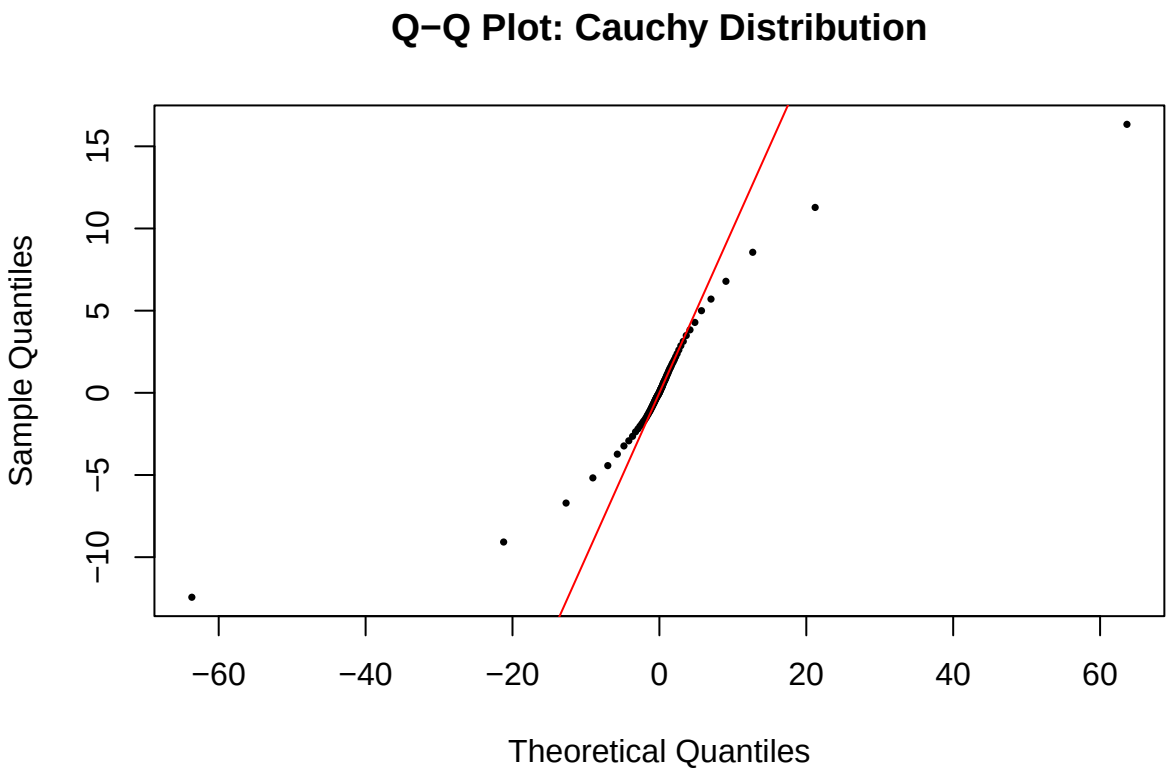
```
# Remove burn-in samples
post_burn_in_samples <- samples[(burn_in + 1):num_samples]

p_values <- seq(0.1, 0.9, 0.1)
sample_quantiles <- quantile(post_burn_in_samples, p_values)
cauchy_quantiles <- qcauchy(p_values)
quantile_comparison <- round(rbind(sample_quantiles, cauchy_quantiles), 3)
print(quantile_comparison)
```

```
##                      10%    20%    30%    40%   50%   60%   70%   80%   90%
## sample_quantiles -2.280 -1.164 -0.598 -0.232 0.029 0.341 0.763 1.434 2.995
## cauchy_quantiles -3.078 -1.376 -0.727 -0.325 0.000 0.325 0.727 1.376 3.078
```

```
p_values <- seq(0.95, 1, 0.01)
sample_quantiles <- quantile(post_burn_in_samples, p_values)
cauchy_quantiles <- qcauchy(p_values)
quantile_comparison <- round(rbind(sample_quantiles, cauchy_quantiles), 3)
print(quantile_comparison)
```

```
##                     95%   96%    97%    98%    99%   100%
## sample_quantiles 5.249 6.265  7.633  9.920 13.067 27.824
## cauchy_quantiles 6.314 7.916 10.579 15.895 31.821    Inf
```

```
#QQ Plot
p_points <- ppoints(100)
sample_quantiles <- quantile(post_burn_in_samples, p_points)
cauchy_quantiles <- qcauchy(p_points)
qqplot(cauchy_quantiles, sample_quantiles, cex = 0.5, pch = 16,
       main = "Q-Q Plot: Cauchy Distribution", xlab = "Theoretical Quantiles", ylab = "Sample Quantiles
abline(0, 1, col = "red")
```



11.6 (1)

```r
# Define the Metropolis-Hastings algorithm for Laplace distribution
rw.Laplace <- function(N, x0, sigma) {
  x <- numeric(N)
  x[1] <- x0
  u <- runif(N)
  k <- 0
  for (i in 2:N) {
    xt <- x[i - 1]
    y <- rnorm(1, xt, sigma)
    if (u[i] <= exp(abs(xt) - abs(y))) {
      x[i] <- y
    } else {
      x[i] <- x[i - 1]
      k <- k + 1
    }
  }
  return(list(x = x, k = k))
}
```

(2)

```r
N <- 5000
sigma <- c(0.5, 1, 2, 4)
x0 <- rnorm(1)
```

```r
rw1 <- rw.Laplace(N, x0, sigma[1])
rw2 <- rw.Laplace(N, x0, sigma[2])
rw3 <- rw.Laplace(N, x0, sigma[3])
rw4 <- rw.Laplace(N, x0, sigma[4])

print(c(rw1$k, rw2$k, rw3$k, rw4$k))
```

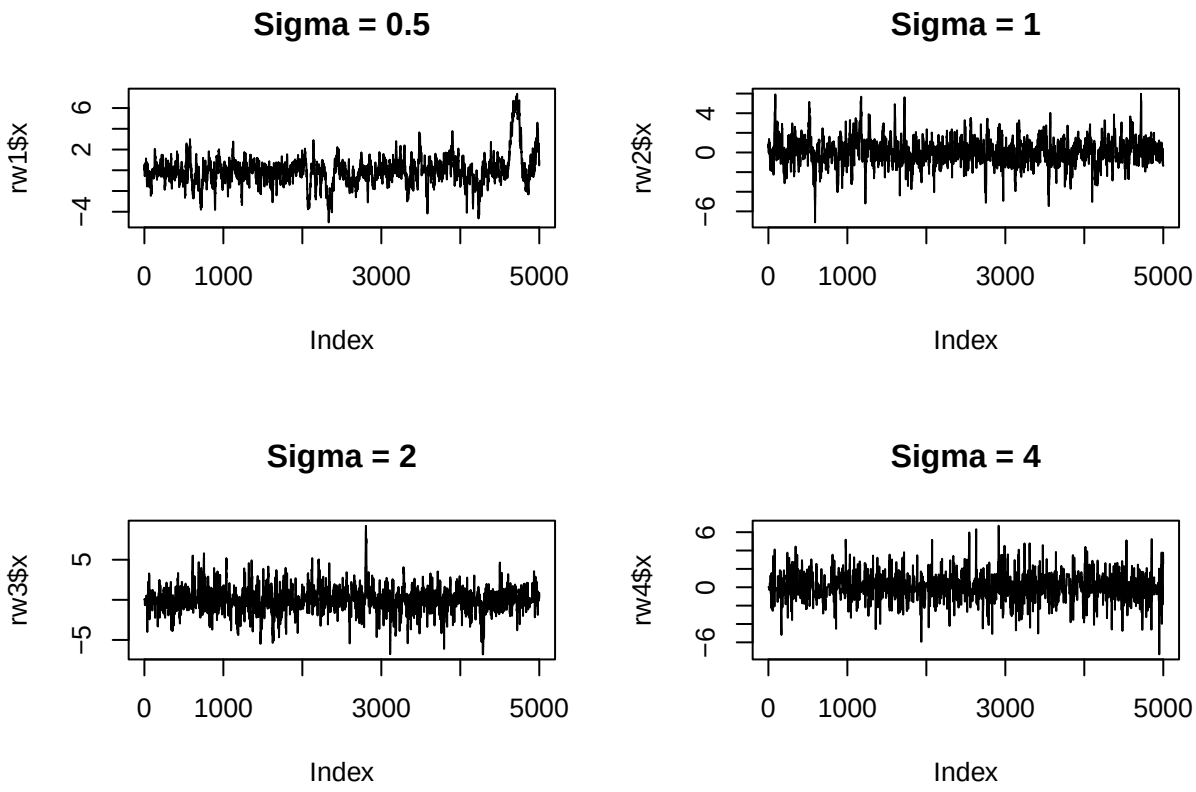
```
## [1]  834 1534 2383 3366
```

```r
cat("rejection rates ", (c(rw1$k, rw2$k, rw3$k, rw4$k)/N), "\n")
```

```
## rejection rates  0.1668 0.3068 0.4766 0.6732
```

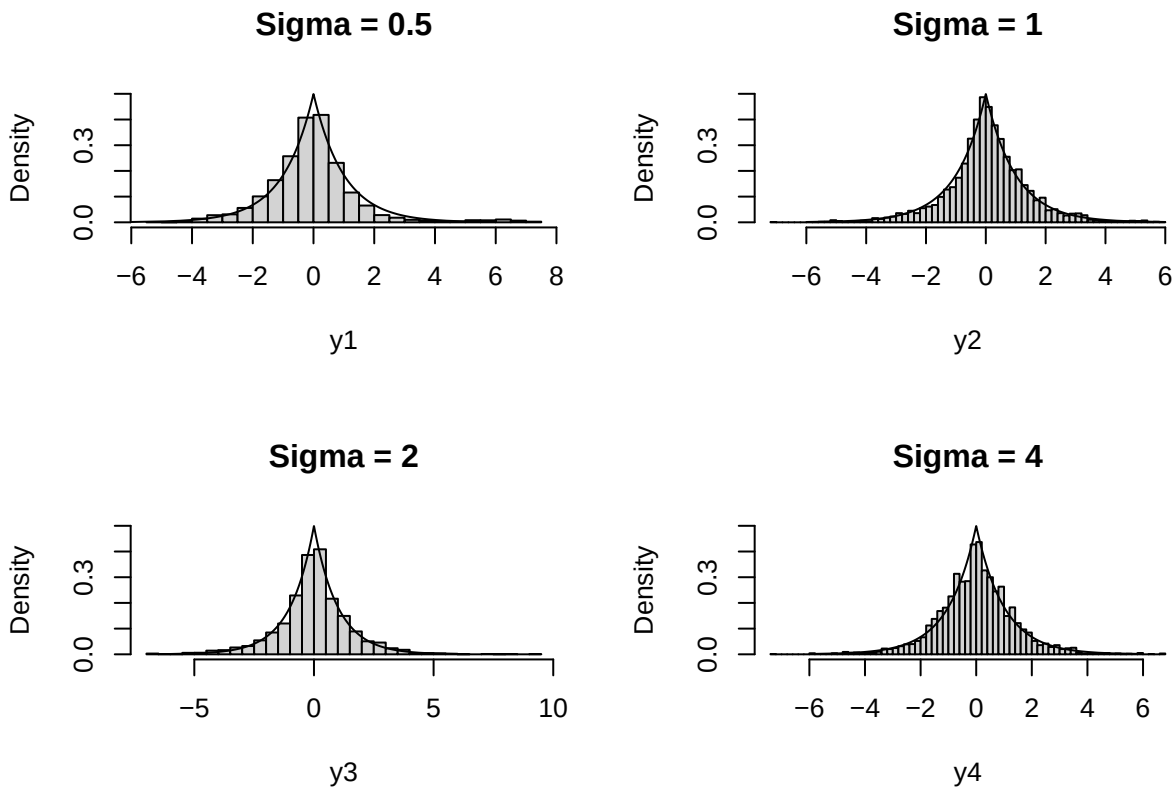
(3)

```r
# Remove burn-in samples
burn_in <- 100
y1 <- rw1$x[(burn_in + 1):N]
y2 <- rw2$x[(burn_in + 1):N]
y3 <- rw3$x[(burn_in + 1):N]
y4 <- rw4$x[(burn_in + 1):N]

par(mfrow = c(2, 2))
plot(rw1$x, type = "l", main = paste("Sigma =", sigma[1]))
plot(rw2$x, type = "l", main = paste("Sigma =", sigma[2]))
plot(rw3$x, type = "l", main = paste("Sigma =", sigma[3]))
plot(rw4$x, type = "l", main = paste("Sigma =", sigma[4]))
```



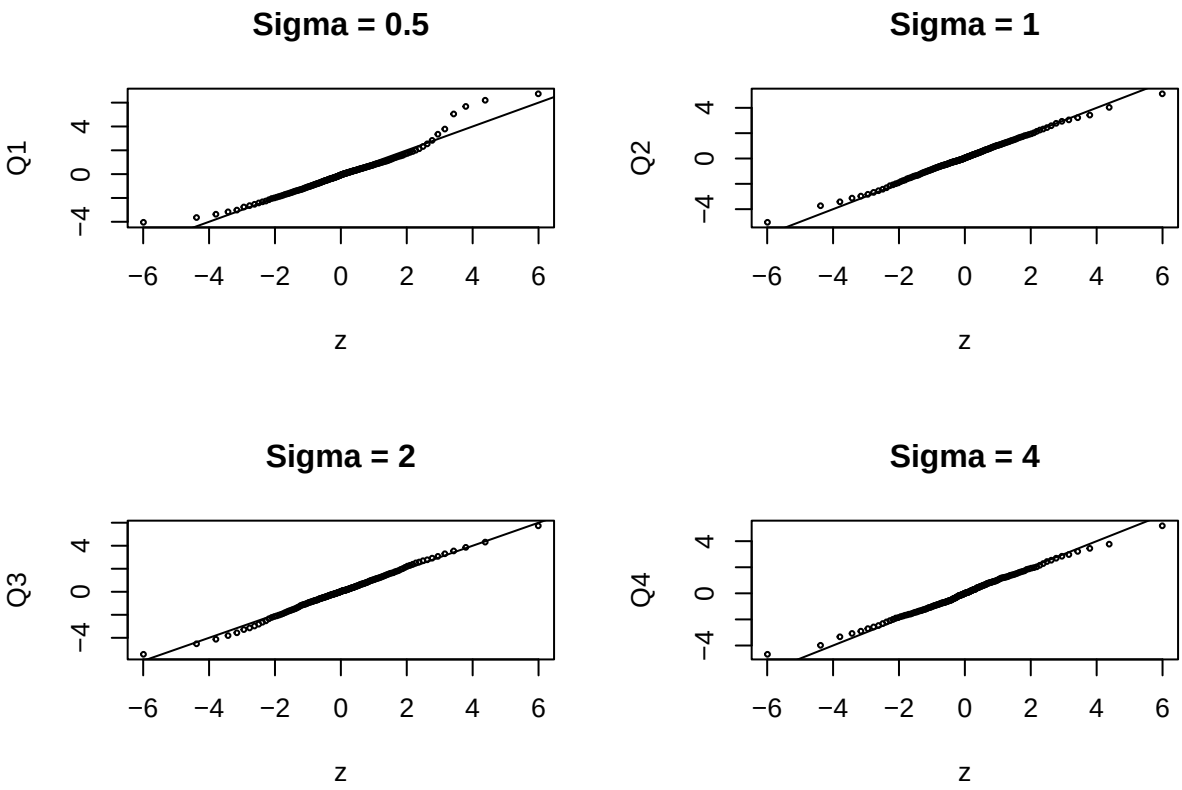
(4)

```r
par(mfrow = c(2, 2))
p <- ppoints(200)
y <- qexp(p, 1)
z <- c(-rev(y), y)
fx <- 0.5 * exp(-abs(z))

hist(y1, breaks = "Scott", freq = FALSE, ylim = c(0, 0.5), main = paste("Sigma =", sigma[1]))
lines(z, fx)

hist(y2, breaks = "Scott", freq = FALSE, ylim = c(0, 0.5), main = paste("Sigma =", sigma[2]))
lines(z, fx)

hist(y3, breaks = "Scott", freq = FALSE, ylim = c(0, 0.5), main = paste("Sigma =", sigma[3]))
lines(z, fx)

hist(y4, breaks = "Scott", freq = FALSE, ylim = c(0, 0.5), main = paste("Sigma =", sigma[4]))
lines(z, fx)
```



(5)

```
# Generate Q-Q plots and compare quantiles
par(mfrow = c(2, 2))
Q1 <- quantile(y1, p)
qqplot(z, Q1, cex = 0.4, main = paste("Sigma =", sigma[1]))
abline(0, 1)

Q2 <- quantile(y2, p)
qqplot(z, Q2, cex = 0.4, main = paste("Sigma =", sigma[2]))
abline(0, 1)

Q3 <- quantile(y3, p)
qqplot(z, Q3, cex = 0.4, main = paste("Sigma =", sigma[3]))
abline(0, 1)

Q4 <- quantile(y4, p)
qqplot(z, Q4, cex = 0.4, main = paste("Sigma =", sigma[4]))
abline(0, 1)
```


Sigma = 0.5
Q1
−4
0
4
−6
−4
−2
0
2
4
6
z
Sigma = 1
Q2
−4
0
4
−6
−4
−2
0
2
4
6
z
Sigma = 2
Q3
−4
0
4
−6
−4
−2
0
2
4
6
z
Sigma = 4
Q4
−4
0
4
−6
−4
−2
0
2
4
6
z