Carey Expected Return R Project Test

Exercise 1

a)

We first write a `R` function that determines the value of the parameters μ and σ^2 from the expectation $\mathbb{E}\{X\}$ and the variance $\mathbb{V}\{X\}$ of a univariate lognormal random variable $X \sim LogN(\mu, \sigma^2)$:

```
find_params <- function(exp, var) {
  sig_sq <- log(var / exp^2 + 1)
  mu <- log(exp) - sig_sq / 2
  return(list("mu" = mu, "sig_sq" = sig_sq))
}
```

Note: We managed to derive these formulas by rearranging the expressions for the mean and variance of a log-normal r.v.

b)

Next, we use the function created in a) to determine the parameters μ and σ^2 such that $\mathbb{E}\{X\} = 3$ and $\mathbb{V}\{X\} = 5$:

```
params <- find_params(3, 5)
ln_mu <- params$mu; ln_sig_sq <- params$sig_sq
ln_mu; ln_sig_sq
```

```
## [1] 0.8776959
```

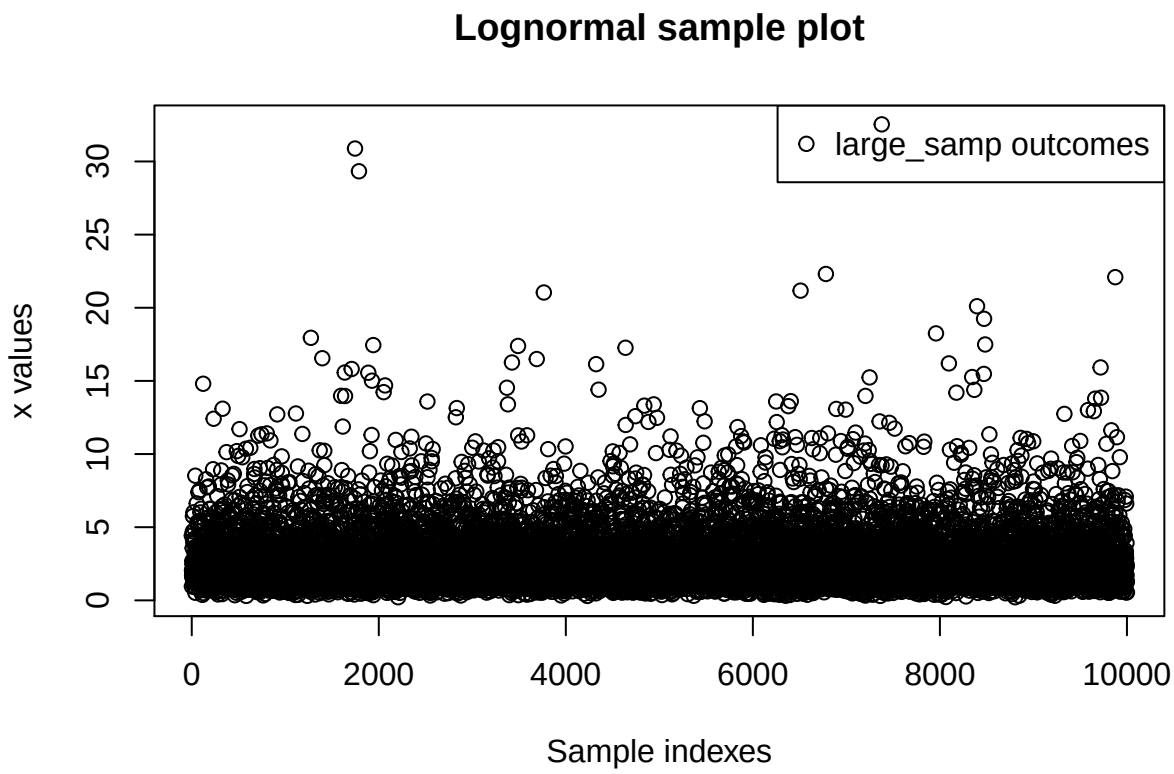
```
## [1] 0.4418328
```

Next, we generate a large sample (of size $n = 10,000$) from the distribution of X with such parameters:

```
n <- 10000
ln_sig <- sqrt(ln_sig_sq)
large_samp <- rlnorm(n, ln_mu, ln_sig)
```

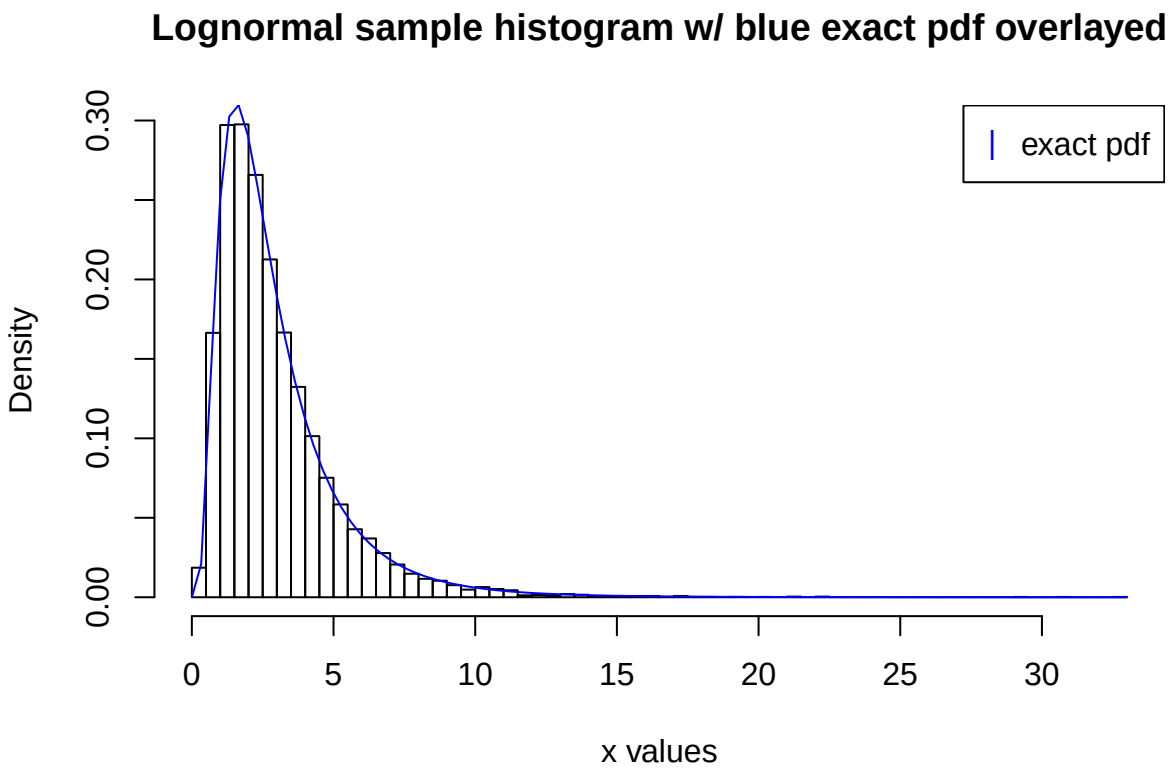
After that, we plot the sample as dots:

```
plot(large_samp, main = "Lognormal sample plot", xlab = "Sample indexes",
     ylab = "x values")
legend("topright", legend = "large_samp outcomes", pch = 1)
```



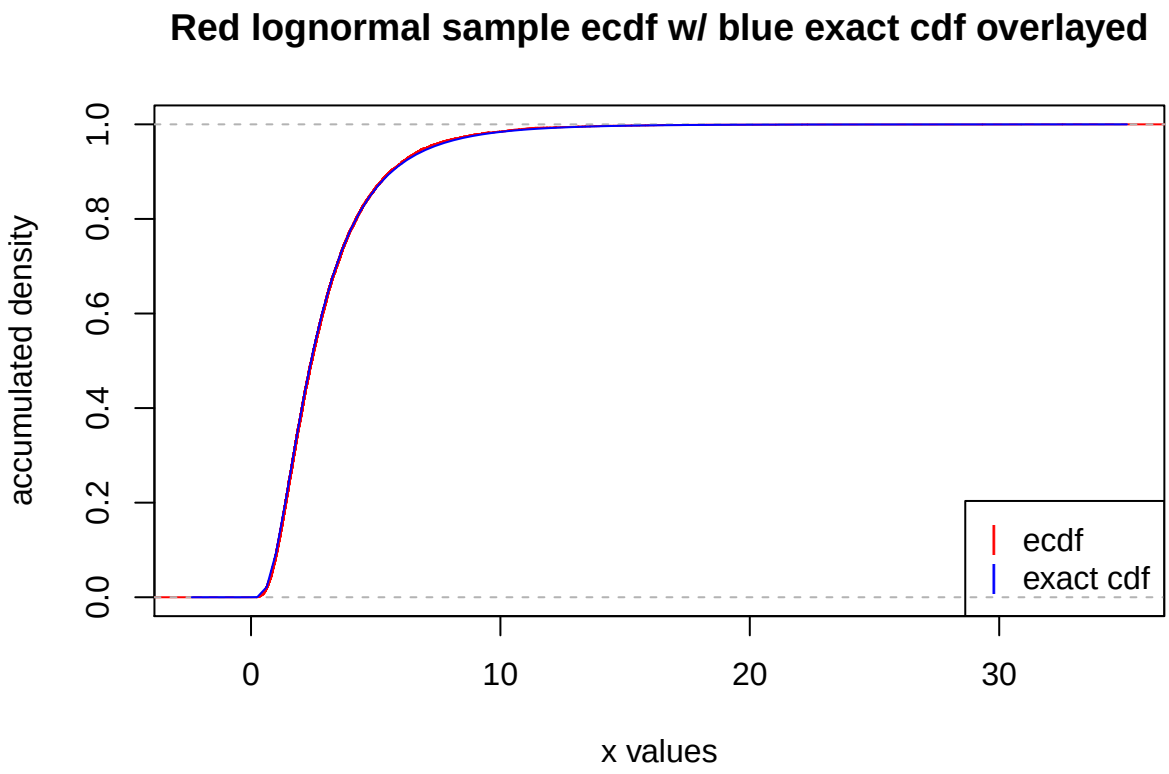
Next, we plot the histogram and overlay the the exact pdf for comparison:

```
hist(large_samp, 50, FALSE,
     main = "Lognormal sample histogram w/ blue exact pdf overlayed",
     xlab = "x values")
curve(dlnorm(x, meanlog = ln_mu, sdlog = ln_sig), add = TRUE, col = "blue")
legend("topright", legend = "exact pdf", col = "blue", pch = "|")
```



Finally, we plot the empirical cdf and overlay the exact cdf for comparison:

```
plot(ecdf(large_samp),
     main = "Red lognormal sample ecdf w/ blue exact cdf overlayed",
     xlab = "x values", ylab = "accumulated density", col = "red")
curve(plnorm(x, meanlog = ln_mu, sdlog = ln_sig), add = TRUE, col = "blue")
legend("bottomright", legend = c("ecdf", "exact cdf"), col = c("red", "blue"),
       pch = "|")
```



Exercise 2

a)

First, we load in a library we will require for our simulation in b) and then proceed to write the function:

```
# Load MASS library for use of mvrnorm function
library(MASS)

fit_locdisp_mlfp <- function(eps_mat, p_vec, nu, thresh) {
  # Extract length of p_vec (also width of eps_mat) and num_rows
  t_bar <- length(p_vec)
  num_rows <- nrow(eps_mat)

  # Introduce helper functions for constructing m_vec, s_mat
  construct_m_vec <- function(vec) {
    # simple matrix multiplication
    m_vec_ <- eps_mat %*% vec
    return(m_vec_)
  }
  construct_s_mat <- function(vec, m_vec_) {
    # Replace each 2*1 col of eps_mat w/ 2*2 prod
    mat_prod <- matrix(apply(eps_mat, 2, function(eps) (eps - m_vec_)
                             %*% t(eps - m_vec_)), 2)
    # Multiply each 2*2 square w/in matrix by corresponding vec val
```

```
    weighted_mat_prod <- sweep(mat_prod, 2, rep(vec, each = 2), '*')
    # Sum up 2*2 squares w/in matrix
    s_mat_ <- matrix(c(rowSums(weighted_mat_prod[, seq(1, 2*t_bar, 2)]),
                       rowSums(weighted_mat_prod[, seq(2, 2*t_bar, 2)])), 2)
  }
  # Learned sweep method from
  # https://intellipaat.com/community/24687/multiply-rows-of-matrix-by-vector

  # 0. Initialize
  m_vec <- construct_m_vec(p_vec)
  s_mat <- construct_s_mat(p_vec, m_vec)
  if (nu > 2) {
    s_mat <- ((nu - 2) / nu) * s_mat
  }

  # Construct norm values to be updated after each run to check for convergence
  rel_euc_norm <- thresh * 2
  rel_fro_norm <- thresh * 2

  while ((rel_euc_norm > thresh) || (rel_fro_norm > thresh)) {
    # 1. Update weights and FP
    w_vec <- (nu + num_rows) /
      (nu + apply(eps_mat, 2, function(eps) t(eps - m_vec) %*% t(s_mat) %*%
                    (eps - m_vec)))
    q_vec <- p_vec * w_vec / sum(p_vec * w_vec)

    # 2. Update output
    m_vec_old <- m_vec; m_vec_old
    m_vec <- construct_m_vec(q_vec)
    s_mat_old <- s_mat; s_mat_old
    s_mat <- construct_s_mat(q_vec, m_vec)

    # Update norm values to check for convergence; return to 1) if no
    # convergence
    rel_euc_norm <- (norm(m_vec - m_vec_old)) / norm(m_vec_old); rel_euc_norm
    rel_fro_norm <- (det(s_mat - s_mat_old)) / det(s_mat_old); rel_fro_norm
  }
  # 3. If convergence, output (mu_vec, sig_sq_vec)
  return(list("m_vec" = m_vec, "s_mat" = s_mat))
}
```

b)

Next, we test the function:

```
n <- 1000
mu <- c(0, 0)
Sigma <- matrix(c(1, 0, 0, 1), 2)

eps_mat <- t(mvrnorm(n, mu, Sigma))
p_vec <- rep(1 / n, n)
nu <- 100
thresh <- 10^(-9)
```

```
fit_locdisp_mlfp(eps_mat, p_vec, nu, thresh)
```

```
## $m_vec
##             [,1]
## [1,] -0.01651437
## [2,] -0.04649793
##
## $s_mat
##             [,1]        [,2]
## [1,] 0.990638985 0.004369264
## [2,] 0.004369264 0.921856770
```

As these maximum likelihood estimates very closely resemble the vector of means and covariance matrix parameters we passed into the mvnorm function to perform our 1000 simulations from the standard bivariate normal distribution, this test provides evidence of the functionality of our fit_locdisp_mlfp function.