Jared Yu

Computational Statistics

Problem Set 11

NOTE: Code is in the Appendix at the end.

1. Problem 10.1 (a) and (b) **Use the F12 data that is posted on the course Blackboard site and not that posted by the authors. Please note that you will have to compute the random variable** X **since it is the log of the data as given.**

   For part (a), note the following:
   i. You do not need to use the UCV bandwidth, just the other three.
   ii. As part of your answer, plot each density estimate and a histogram of the data.
   iii. When calculating the Sheather-Jones method bandwith, use Silverman's Rule of thumb as your pilot h_0 and the Normal kernel for L to estimate $f''(x)$, then plug the result into Equation 10.24 to get the h for your final estimate of f. Do not use Equation 10.28. Also, you can use Monte Carlo Integration to estimate the roughness of $f''(x)$.

Below in Figures 1-3 are sets of histograms along with KDE estimates. Since histograms also have options for the number of m bins, a grid of 6 were created for each Figure. The grid contains histograms of $m = 10, 20, \cdots, 60$ bins. It seems that around the middle of 40 is appropriate, but the entire set was included just to see how well the KDE line would fit each histogram.

$Silverman$	$Sheather - Jones$	$Terrell$
0.2337528	0.1857723	0.2524386

The above table shows the corresponding bandwidths for each of the three different methods: Silverman's rule of thumb, the Sheather-Jones approach, and Terrell's maximal smoothing principle.

Both Silverman and Terrell bandwidths were quite simple to compute. The Sheather-Jones method was the most difficult. It was initially attempted to find a closed form solution for $R(\hat{f}'')$, however a certain solution was not found. Therefore, Monte Carlo Integration was utilized to try and estimate the value. Compared to the others, it's less certain whether the Sheather-Jones value is correct. By double-checking with the "locfit" package in R, it seems that the above Sheather-Jones bandwidth doesn't agree with the package value of 0.2022821. There is a similar pattern however where the Sheather-Jones bandwidth is smaller than both the Silverman and Terrell bandwidths. An important note is that Monte Carlo Integration was used so the above value derived from simulation is just an estimate.

Looking at the histograms themselves, it seems that the shape is overall unimodal, looking like the bell-curve from the Normal distribution. However, looking closer at the histograms with more bins, it's apparent that there are three miniature peaks in the middle. They're noticeable

enough to the point where it seems like they're features of the distribution of the data that are worth investigation.

The bandwidth from Terrell underfits this characteristic of the histogram and so it seems to do a poor job of noticing the three peaks in the density (Figure 3). Looking very closely at Figure 1, it seems that Silverman's bandwidth can barely do a better job of picking up on this feature of three peaks compared to Terrell's method. However, looking at Figure 2, it's clear that the Sheather-Jones bandwidth does pick up this characteristic and fits to it accordingly without simultaneously overfitting the histogram. In Figure 4 there's a single histogram ($m = 40$) with all three bandwidths applied together at the same time. It seems from this plot that for this data, the Silverman and Terrell bandwidths are virtually the same with only minor differences. However, the Sheather-Jones bandwidth makes a note of the three obvious peaks in the distribution.

The result from Terrell's bandwidth is expected, as in practice it tends to oversmooth. The idea behind it is that it's selected to discourage false modes being estimated. Therefore, it seems to have ignored the three peaks by design. Also, the result from Silverman makes sense too given the data's possible multimodal (three peaks) distribution. Silverman's method will also in practice oversmooth these characteristics due to the mathematical properties that Silverman uses to estimate $R(f'')$. The best bandwidth amongst the three seems to be the Sheather-Jones bandwidth. It seems to perform well on this dataset which is expected, since it's supposedly a reliable plug-in method.

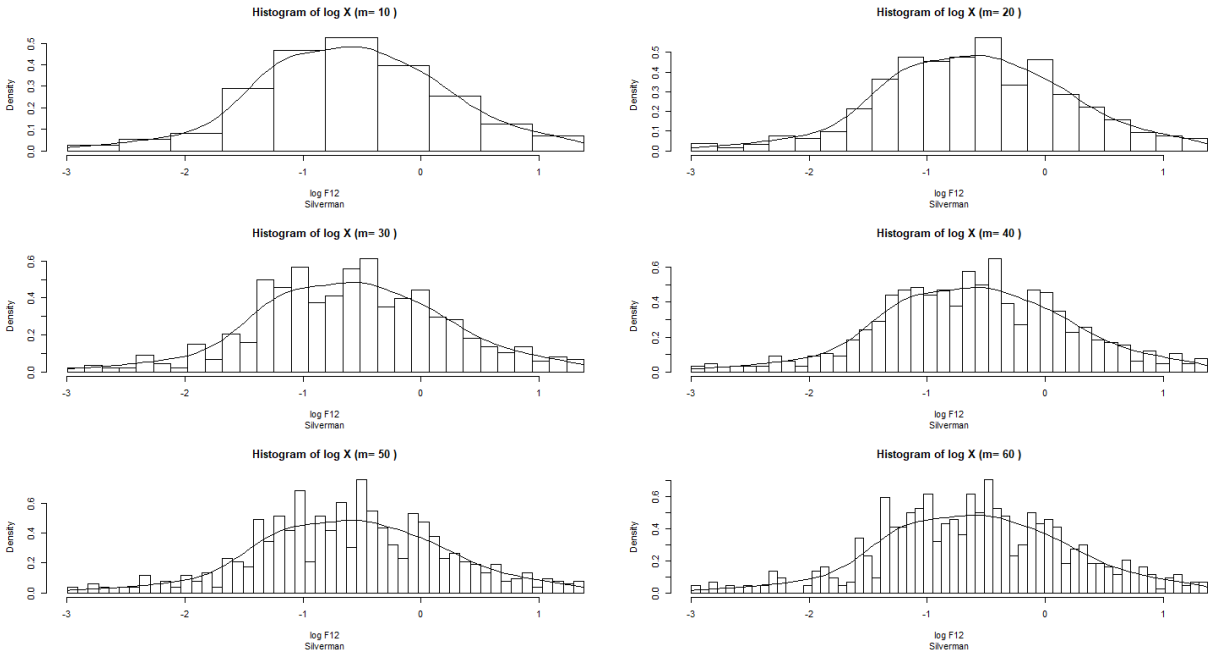

Figure 1 A grid of histograms using Silverman's rule of thumb.

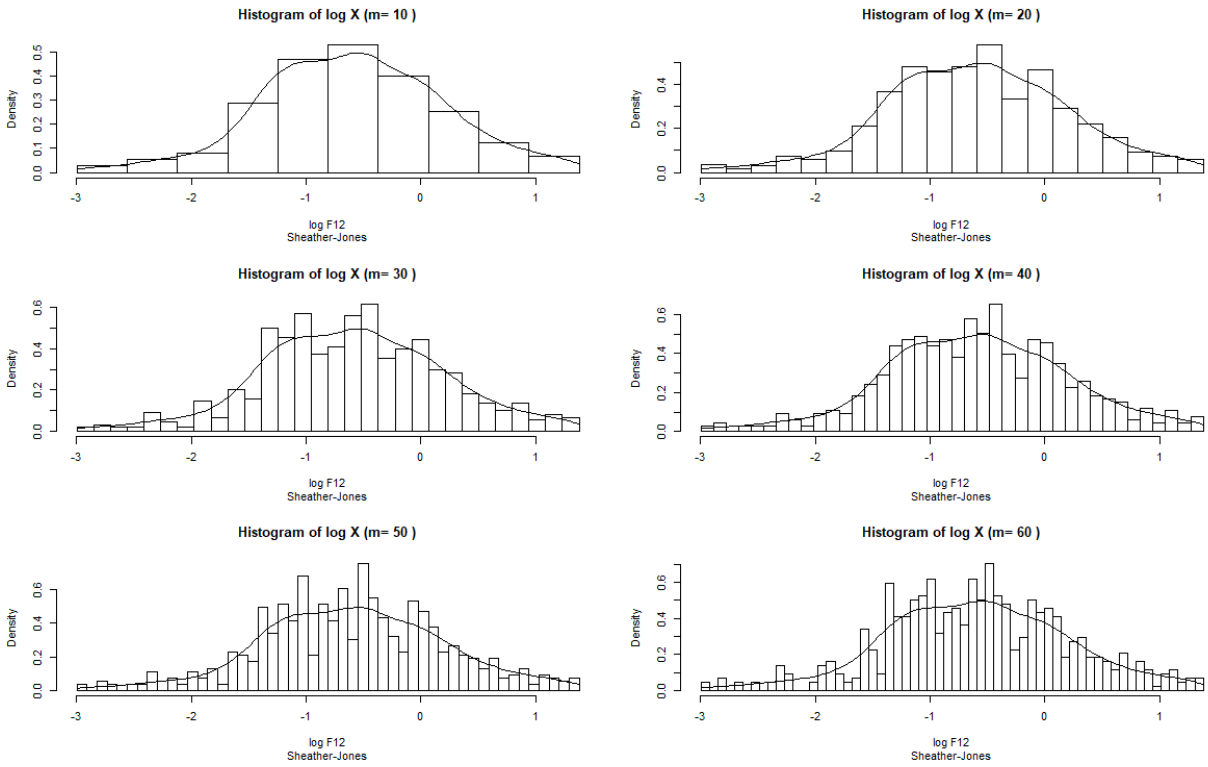

Figure 2 A grid of histograms using the Sheather-Jones approach.

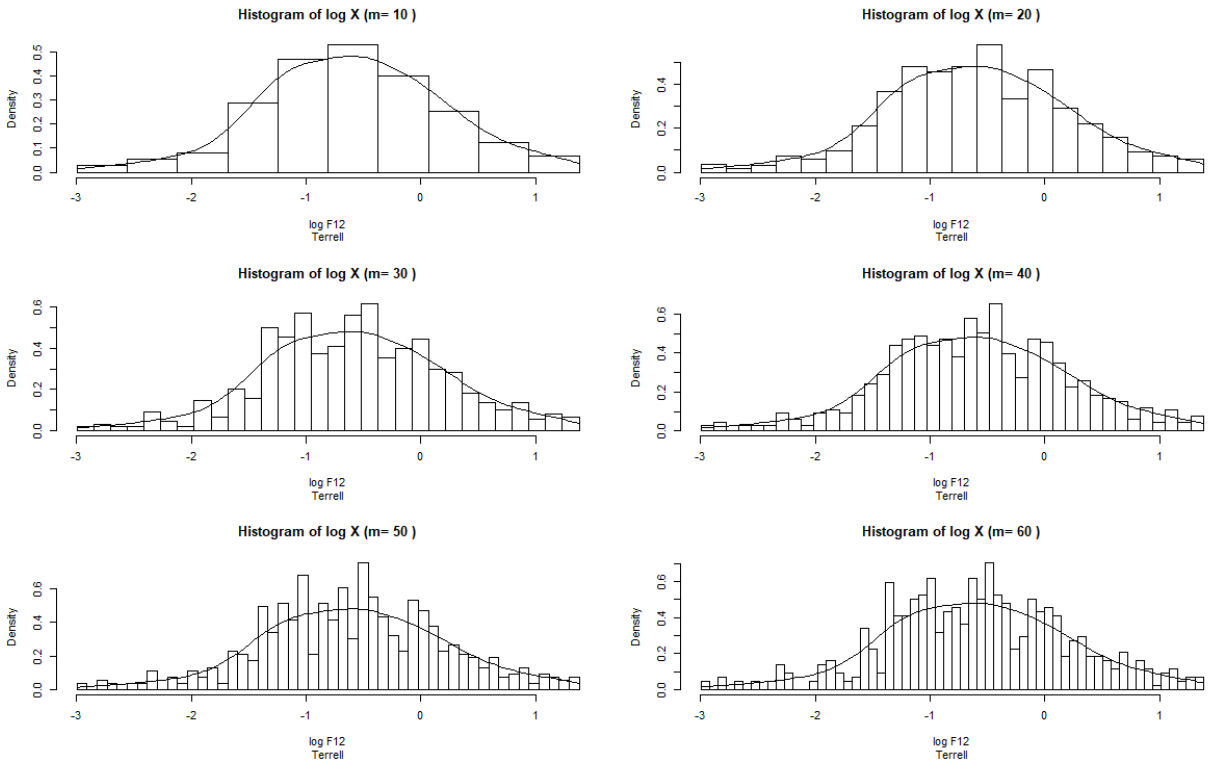

Figure 3 A grid of histograms using Terrell's maximal smoothing principle.

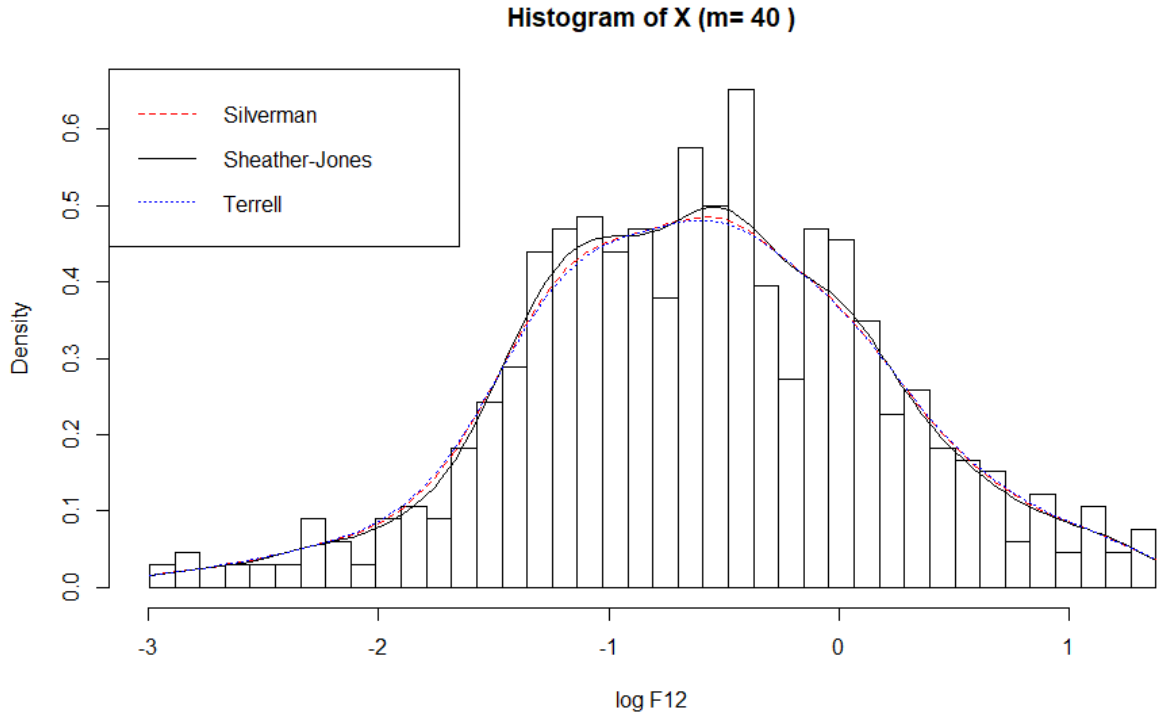

Figure 4 Combined histogram (m=40) that includes all three bandwidths.

For part (b):

i. As part of your answer, plot each density estimate.
ii. Use the Silverman's Rule of thumb for the bandwidth developed in part (a).
iii. Note that you already found the estimate using the normal kernel in part (a).
iv. Note that for all the kernels except the normal kernel, $K(z)$ as defined in Table 10.1 must be multiplied by $I_{\{|z|<1\}}$.
v. In addition to finding the kernel estimates, find a histogram estimator using the Silverman's Rule of Thumb for the bandwidth.

Below in Figures 5-7 are similar grids of histograms as seen before in part (a). The Figures below plot the KDE using the above Sheather-Jones bandwidth for the following kernels: Uniform, Epanechnikov, and Triweight respectively.

In Figure 5 with the Uniform kernel, there's obviously some severe overfitting being seen in the KDE curve. In each of the different histograms, it's apparent that the kernel is accounting for many nuances in the data that are unnecessary. The KDE curve is quite jagged, and it seems to pick up on the three peaks in addition to many other minor features that are likely not as important.

In Figure 6 with the Epanechnikov kernel, the kernel does a good job of noticing the three separate peaks in the distribution quite well. It also doesn't seem to overfit the data too severely to the point where the features look exaggerated. This kernel helps to point out that the three peaks are not the same, with the one on the right being noticeably smaller than the other two.

In Figure 7 with the Triweight kernel, there seems to be a similar plot as with the Epanechnikov kernel. However, there seems to be a bit of extra undersmoothing to the point where it seems to become too jagged. For example, it picks up on the three peaks, but it also makes note of possibly a fourth by breaking the left peak into two. This is also possibly an important feature, but it's not certain. Perhaps by noticing this extra difference, there is some significance to this characteristic in the true density.

The differences seen in the above three kernels make sense given that they utilize an indicator function. The normal kernel on the other hand will stretch along the entire real number line. These kernels however are cut shorter and so they have the potential to do less of a job of oversmoothing the overall KDE curve. This seems to be an important distinction given the dataset. The data seems to have an apparent unimodal distribution that becomes multimodal as more bins are added to the histogram. The normal kernel seems to do a poorer job of noticing this feature compare to the other kernels.

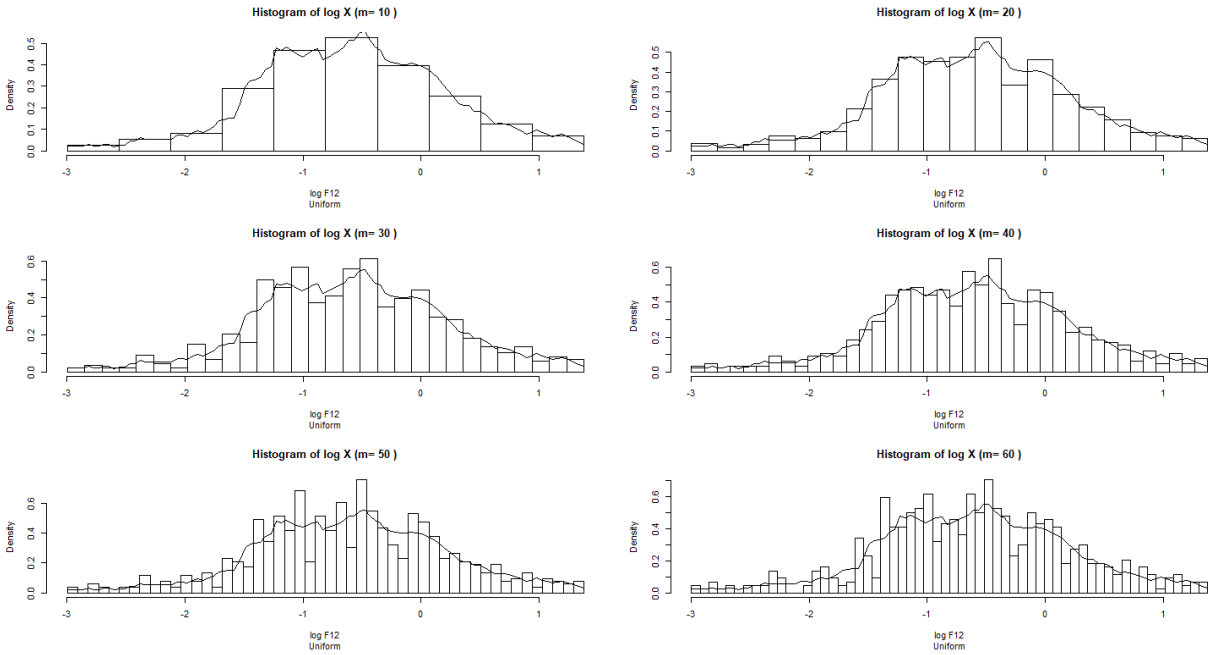

Figure 5

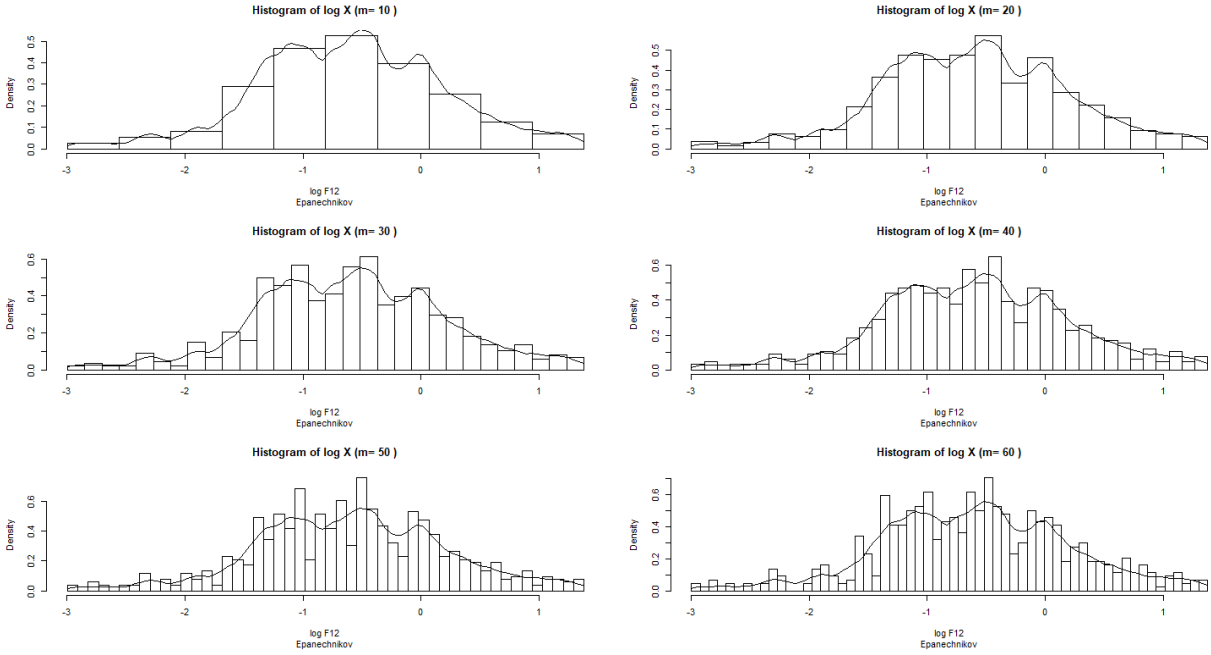

Figure 6

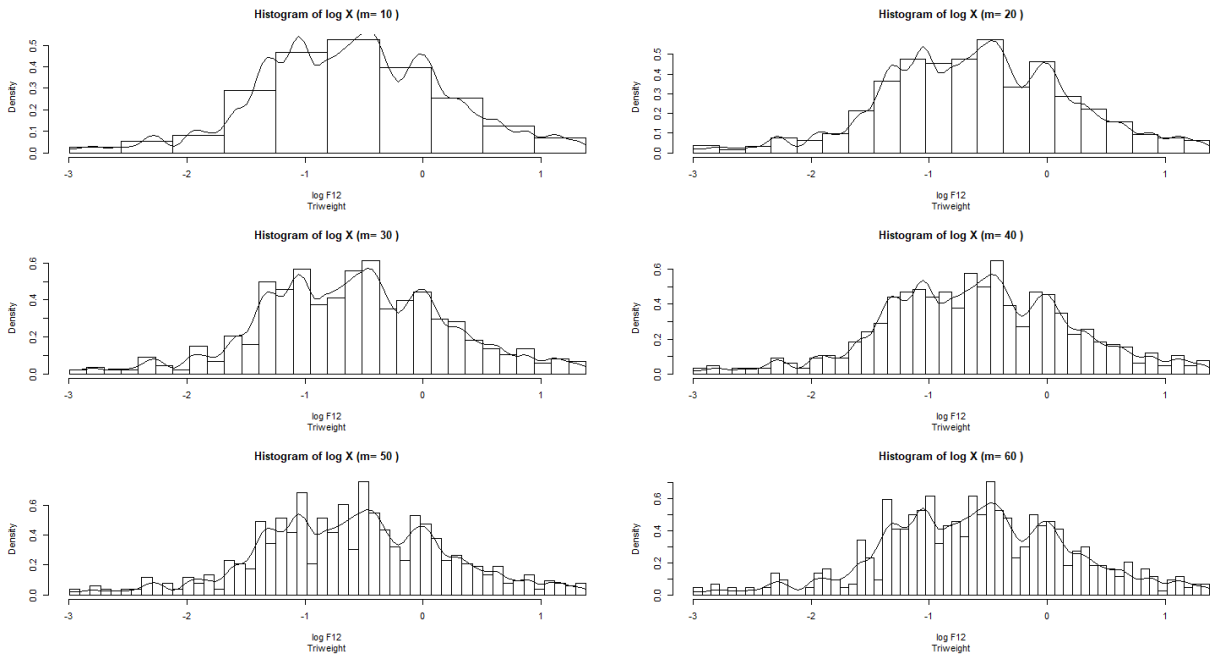

Figure 7

Below is a histogram estimator utilizing the Silverman's rule of thumb for a bandwidth.

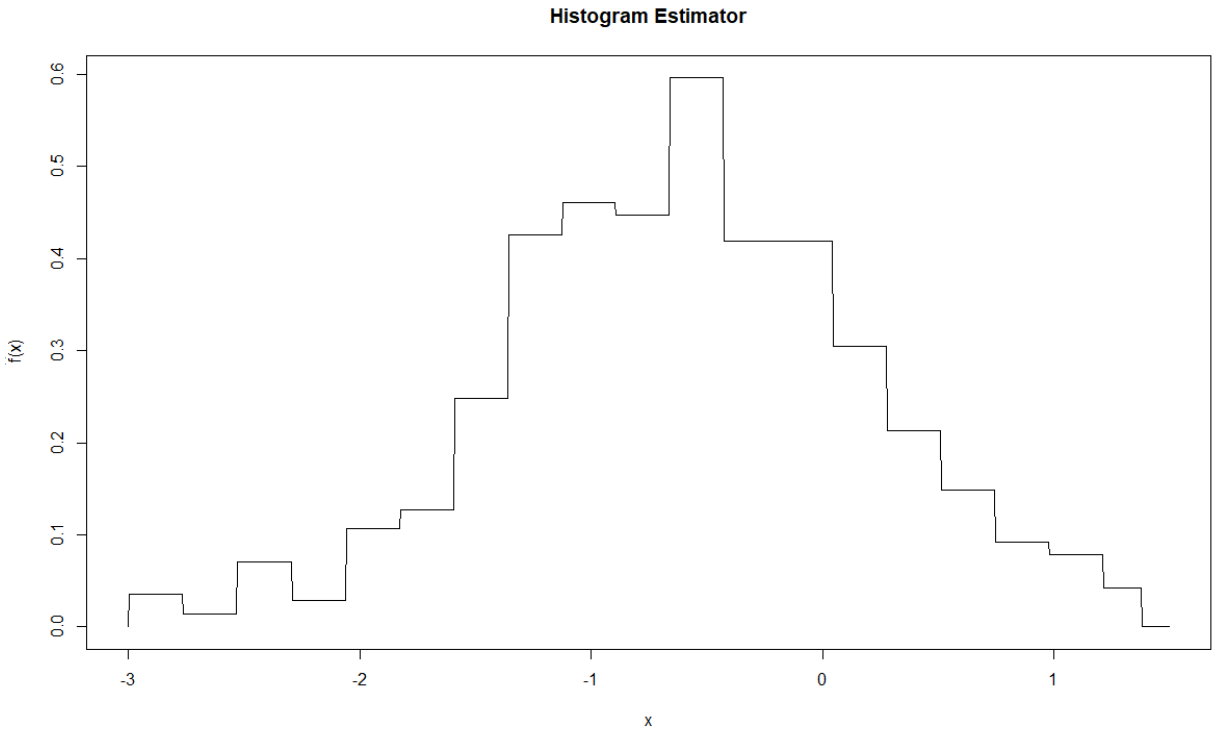

Figure 8

Appendix

```
### 10.1
f12 <- read.csv('F12.txt', header = FALSE) # load data
f12 <- log(f12[[1]]) # take log

# custom histogram function
histogram <- function(df = f12, round_digits = 3, m = 10, bandwidth_type = 'Terrell') {
  D <- c( # histogram range
    -round(abs(range(df)[1]), digits = round_digits), # lower range
    round(range(df)[2], digits = round_digits) # upper range
  )
  # Reference: https://stackoverflow.com/questions/22051345/breaking-a-mathematical-range-in-e
qual-parts
  bin_width <- (D[2] - D[1]) / m # width of each bin

  # create (m + 1) break points for the m intervals
  break_pts <- matrix(NA, nrow = m + 1)
  break_pts[1] <- D[1]; break_pts[m + 1] <- D[2]
  for (i in 1:(m - 1)) {
    break_pts[i + 1] <- D[1] + i * bin_width
  }

  hist(x = df, breaks = break_pts, freq = FALSE, # plot histogram
       main = paste('Histogram of log X', '(m=', m,')'),
       xlab = 'log F12', sub = bandwidth_type)
}

# normal pdf
normal <- function(z) { dnorm(z) }

# f-hat function for KDE + vectorized
f_hat <- function(x, X_i, h, K) {
  n <- length(X_i)
  (1 / h) * mean(K((x - X_i) / h))
}
f_hat_vec <- Vectorize(f_hat, vectorize.args = 'x')

# plot KDE
kde <- function(var1 = f12, h = silverman_h, K = normal,
                histogram_round = 3, histogram_m = 10, bw_name = 'Silverman') {
  # n <- length(var1)
  D <- c( # histogram range
    -round(abs(range(var1)[1]), digits = histogram_round), # lower range
    round(range(var1)[2], digits = histogram_round) # upper range
  )
  x_pts <- seq(D[1], D[2], length.out = 1e2) # sequence of x values for KDE
  f_hat_pts <- f_hat_vec(x = x_pts, X_i = var1, h = h, K = K) # KDE
  histogram(df = var1, round_digits = histogram_round,
            m = histogram_m, bandwidth_type = bw_name) # plot
  lines(x_pts, f_hat_vec(x = x_pts, X_i = var1, h = h, K = K)) # add KDE
}

# part (a)
# Silverman's rule of thumb
n <- length(f12)
sigma_hat <- sd(f12)
silverman_h <- ((4 / (3 * n))^(1 / 5)) * sigma_hat

m_vec <- seq(10, 60, length.out = 6)
par(mfrow = c(3,2))
```

```r
sapply(m_vec, function(x)
  kde(var1 = f12, h = silverman_h, K = normal, histogram_round = 3,
      histogram_m = x, bw_name = 'Silverman')
)

# Sheather-Jones approach
# Monte Carlo Integration
# g function
g <- function(y, X = f12, h_0 = silverman_h, A = 1e5) {
  n <- length(X)
  # A <- sum(
  #   sapply(X, function(z) {
  #     ( exp(-0.5 * ((y - z) / h_0)^2) *
  #       (1 - ((y - z) / h_0))^2 )^2
  #   })
  # )

  outer_matrix <- outer(X = X, Y = X, FUN = function(u, v) {
    (exp(-0.5 * ((y - u) / h_0)^2) *
       (1 - ((y - u) / h_0))^2) *
      (exp(-0.5 * ((y - v) / h_0)^2) *
         (1 - ((y - v) / h_0))^2)
  })
  # print(paste('A == B?', A == sum(diag(B_part))))
  L_val <- sum(outer_matrix)


  g_output <- 2 * A * (1 / (n * (h_0^3)))^2 * L_val
  return(g_output)
}
g_vec <- Vectorize(g)

test_values <- seq(-10,10,length.out = 21)
test_values[which(g_vec(test_values) > 1e-3)]
set.seed(0)
u_sample <- runif(1e5, -2,  2)
g_eval <- g_vec(x = u_sample)
R_f <- mean(g_eval)

R_K <- 1 / (2 * sqrt(pi))
h_hat <- (R_K / (n * R_f))^(1 / 5)
sheather_jones_h <- h_hat

par(mfrow = c(3,2))
sapply(m_vec, function(x)
  kde(var1 = f12, h = sheather_jones_h, K = normal, histogram_round = 3,
      histogram_m = x, bw_name = 'Sheather-Jones')
)

locfit::sjpi(x = f12, a = silverman_h)

# Terrell's maximal smoothing principal
R_K <- 1 / (2 * sqrt(pi))
terrell_h <- 3 * ((R_K / (35 * n))^(1 / 5)) * sigma_hat

par(mfrow = c(3,2))
sapply(m_vec, function(x)
  kde(var1 = f12, h = terrell_h, K = normal, histogram_round = 3,
      histogram_m = x, bw_name = 'Terrell')
)
```

```r
# Combined histogram + kde estimates
histogram_combined <- function(df = f12, round_digits = 3, m = 40) {
  D <- c( # histogram range
    -round(abs(range(df)[1]), digits = round_digits), # lower range
    round(range(df)[2], digits = round_digits) # upper range
  )
  # Reference: https://stackoverflow.com/questions/22051345/breaking-a-mathematical-range-in-e
qual-parts
  bin_width <- (D[2] - D[1]) / m # width of each bin

  # create (m + 1) break points for the m intervals
  break_pts <- matrix(NA, nrow = m + 1)
  break_pts[1] <- D[1]; break_pts[m + 1] <- D[2]
  for (i in 1:(m - 1)) {
    break_pts[i + 1] <- D[1] + i * bin_width
  }

  hist(x = df, breaks = break_pts, freq = FALSE, # plot histogram
       main = paste('Histogram of X', '(m=', m,')'),
       xlab = 'log F12')
}

kde_combined <- function(var1 = f12, h_s = silverman_h,
                         h_sj = sheather_jones_h, h_t = terrell_h,
                         K = normal, histogram_round = 3, histogram_m = 40) {
  D <- c( # histogram range
    -round(abs(range(var1)[1]), digits = histogram_round), # lower range
    round(range(var1)[2], digits = histogram_round) # upper range
  )
  x_pts <- seq(D[1], D[2], length.out = 1e2) # sequence of x values for KDE
  histogram_combined(df = var1, round_digits = histogram_round, m = histogram_m) # plot

  # Silverman
  lines(x_pts, f_hat_vec(x = x_pts, X_i = var1, h = h_s, K = K), col = 'red', lty = 2)
  # Sheather-Jones
  lines(x_pts, f_hat_vec(x = x_pts, X_i = var1, h = h_sj, K = K), col = 'black', lty = 1)
  # Terrell
  lines(x_pts, f_hat_vec(x = x_pts, X_i = var1, h = h_t, K = K), col = 'blue', lty = 3)
  legend("topleft", legend = c('Silverman', 'Sheather-Jones', 'Terrell'),
         col = c('red', 'black', 'blue'), lty = c(2,1,3))
}

# part (b)
# Uniform
uniform <- function(z) {
  ifelse(abs(z) < 1, 1 / 2, 0)
}

# Epanechnikov
epanechnikov <- function(z) {
  ifelse(abs(z) < 1, (3 / 4) * (1 - z^2), 0)
}

# Triweight
triweight <- function(z) {
  ifelse(abs(z) < 1, (35 / 32) * ((1 - z^2)^3), 0)
}

par(mfrow = c(3,2))
sapply(m_vec, function(x)
  kde(var1 = f12, h = sheather_jones_h, K = uniform, histogram_round = 3,
```

```r
      histogram_m = x, bw_name = 'Uniform')
)
sapply(m_vec, function(x)
  kde(var1 = f12, h = sheather_jones_h, K = epanechnikov, histogram_round = 3,
      histogram_m = x, bw_name = 'Epanechnikov')
)
sapply(m_vec, function(x)
  kde(var1 = f12, h = sheather_jones_h, K = triweight, histogram_round = 3,
      histogram_m = x, bw_name = 'Triweight')
)

# histogram estimator
hist_est <- function(x, var1 = f12, h = silverman_h) {
  var1_range <- range(var1) # range of data

  # Check if x outside range of f-hat
  if ((x > var1_range[2]) | (x < var1_range[1])) {
    return(0)
  }

  D <- var1_range[2] - var1_range[1] # length D of support
  # v_k <- D / m # volume of bin (length of interval)
  v_k <- h # volume of bin (length of interval)
  n <- length(var1)
  m <- ceiling(D / v_k)

  break_pts <- matrix(NA, nrow = m + 1) # Find break points
  break_pts[1] <- var1_range[1]; break_pts[m + 1] <- var1_range[2]
  for (i in 1:(m - 1)) {
    break_pts[i + 1] <- var1_range[1] + i * v_k # possible bias towards right side
  } # create (m + 1) intervals
  break_pts <- as.vector(break_pts)

  m_bins <- matrix(0, nrow = m)
  for (i in var1) { # calculate the number of obs. in each bin
    lower_interval <- tail(which(i > break_pts), 1)
    m_bins[lower_interval] <- m_bins[lower_interval] + 1
    if (i == min(var1)) { # when i == min value
      m_bins[1] <- m_bins[1] + 1
    }
  }

  p_k <- m_bins / n # proportion per bin
  kth_bin <- tail(which(x >= break_pts), 1)
  f_hat <- p_k / v_k

  if ((kth_bin >= 1) & (kth_bin <= m)) {
    return(f_hat[kth_bin]) # return f-hat
  } else if (kth_bin == (m + 1)) {
    return(f_hat[m]) # edge case for last bin, return m'th bin
  } else {
    return(0)
  }
}
hist_est_vec <- Vectorize(hist_est, vectorize.args = c('x'))
xs <- seq(-3, 1.5, length.out = 1e3)
plot(xs, hist_est_vec(x = xs, var1 = f12, h = silverman_h),
     type = 'l', main = 'Histogram Estimator',
     xlab = latex2exp::TeX('$x$'), ylab = latex2exp::TeX('$\\hat{f}(x)$'))
```