ps11\_rmd

Jared Yu

April 19, 2020

### 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-equal-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))  
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')  
)  
  
# 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-equal-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,  
 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('$\\hat{f}(x)$'))