Yichen Dong HW 11

Yichen Dong

November 19, 2018

## Problem 1A

Initial Setup

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

F12 = scan('F12.txt')  
F12 = as.data.frame(F12)  
F12 = F12 %>%  
 mutate(X = log(F12))  
n = length(F12$X)  
sd = sd(F12$X)  
F12 = F12%>%  
 arrange(X)

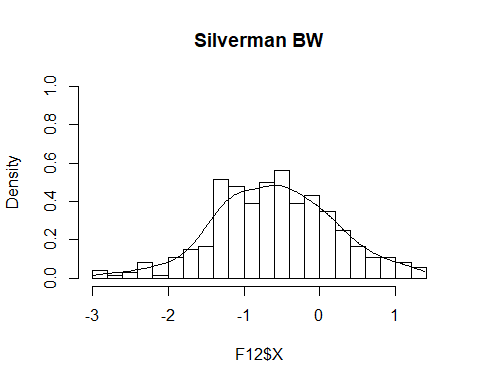
I sorted by X so that I can plot the density using lines later.

### Silverman\_bw

Silverman\_bw = (4/(3\*n))^(1/5)\*sd  
Silverman\_density = vector(mode = 'double',length = n)  
for(i in 1:length(F12$X)){  
 for(j in 1:length(F12$X)){  
 Silverman\_density[i] = max(Silverman\_density[i],0) + 1/(n\*Silverman\_bw)\*(exp(-(F12$X[i] - F12$X[j])^2/(2\*Silverman\_bw^2))/sqrt(2\*pi))  
 }  
}  
F12 = F12 %>%  
 cbind(Silverman\_density)  
  
Silverman\_bw

## [1] 0.2337528

hist(F12$X,freq = FALSE,ylim=c(0,1),breaks = 20,main='Silverman BW')  
lines(F12$X,F12$Silverman\_density)



### SJ BW

Note: I tried to find the SJ BW by hand, but I got really stuck on the integration for R(f’’). So I used the code in the book to generate a SJ BW so I could do the rest of the problem set.

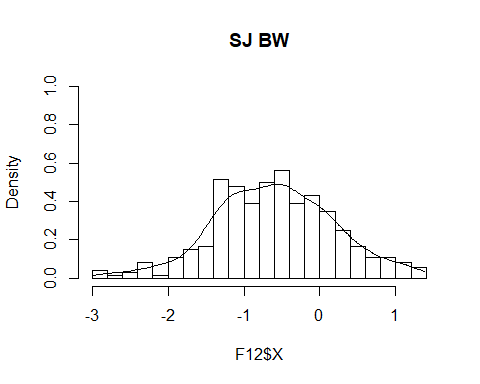
h\_0 = Silverman\_bw  
  
R\_f\_d2 = 0  
for (j in 1:length(F12$X)){  
 for (i in 1:length(F12$X)){  
 R\_f\_d2= -exp(-(F12$X[j] - F12$X[i])/h\_0)/(32\*n^2\*h\_0^9\*pi)  
 }  
}  
R\_k = 1/(2\*sqrt(pi))  
sd = sd(F12$X)  
SJ\_bw = (R\_k/(-n\*sd^4\*R\_f\_d2))^(1/5)  
R\_k/(h\_0^5\*n\*sd^4) #This is what R\_f\_d2 should be in order for it to match the method they used in the book

## [1] 1.686412

#I couldn't figure out how to get R(f''(x)), so I used the method they had in the book code  
SJ\_bw = bw.SJ(F12$X)  
SJ\_density = vector(mode = 'double',length = n)  
  
for(i in 1:length(F12$X)){  
 for(j in 1:length(F12$X)){  
 SJ\_density[i] = max(SJ\_density[i],0) + 1/(n\*SJ\_bw)\*(exp(-(F12$X[i] - F12$X[j])^2/(2\*SJ\_bw^2))/sqrt(2\*pi))  
 }  
}  
F12 = F12 %>%  
 cbind(SJ\_density)  
SJ\_bw

## [1] 0.208654

hist(F12$X,freq = FALSE,ylim=c(0,1),breaks = 20,main='SJ BW')  
lines(F12$X,F12$SJ\_density)

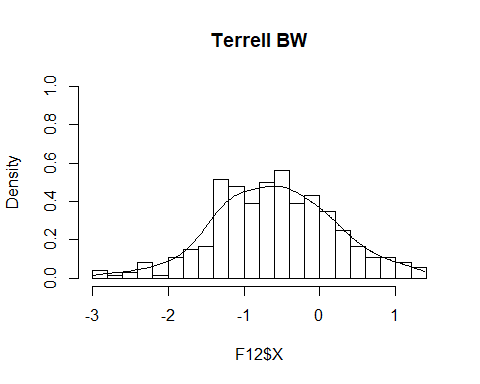


### Terrell BW

Terrell\_bw = 3\*(R\_k/(35\*n))^(1/5)\*sd  
Terrell\_density = vector(mode = 'double',length = n)  
for(i in 1:length(F12$X)){  
 for(j in 1:length(F12$X)){  
 Terrell\_density[i] = Terrell\_density[i] + 1/(n\*Terrell\_bw)\*(exp(-(F12$X[i] - F12$X[j])^2/(2\*Terrell\_bw^2))/sqrt(2\*pi))  
 }  
}  
F12 = F12 %>%  
 cbind(Terrell\_density)  
Terrell\_bw

## [1] 0.2524386

hist(F12$X,freq = FALSE,ylim=c(0,1),breaks = 20,main='Terrell BW')  
lines(F12$X,F12$Terrell\_density)

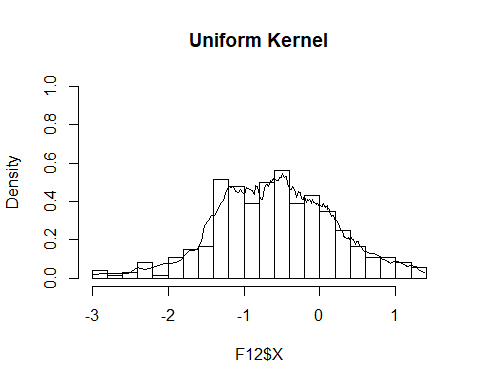


I feel like these are all so similar, which resulted in the density graph looking almost the same.

## Part B

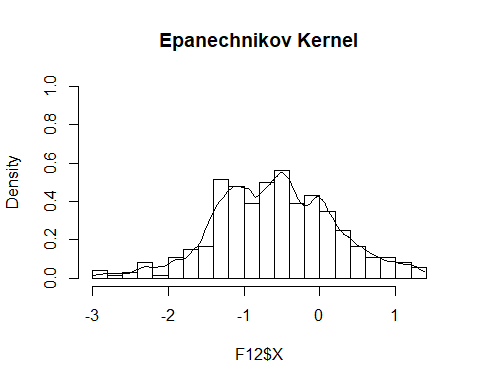
### Uniform

SJ\_density\_unif = vector(mode = 'double',length = n)  
for(i in 1:length(F12$X)){  
 for(j in 1:length(F12$X)){  
 SJ\_density\_unif[i] = SJ\_density\_unif[i] + 1/(n\*SJ\_bw)\*(1/2)\*(ifelse(abs((F12$X[i] - F12$X[j])/SJ\_bw)<1,1,0))  
 }  
}  
F12 = F12 %>%  
 cbind(SJ\_density\_unif)  
hist(F12$X,freq = FALSE,ylim=c(0,1),breaks = 20,main='Uniform Kernel')  
lines(F12$X,F12$SJ\_density\_unif)

 The uniform seems way too wiggly and seems to be overfitting the density on the underlying data.

### Epanechnikov

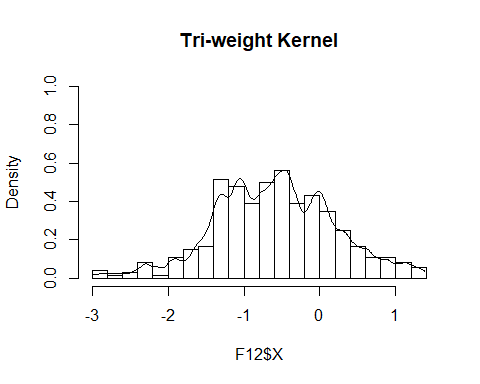
SJ\_density\_Ep = vector(mode = 'double',length = n)  
for(i in 1:length(F12$X)){  
 for(j in 1:length(F12$X)){  
 SJ\_density\_Ep[i] = SJ\_density\_Ep[i] + 1/(n\*SJ\_bw)\*(3/4\*(1-((F12$X[i]-F12$X[j])/SJ\_bw)^2))\*(ifelse(abs((F12$X[i] - F12$X[j])/SJ\_bw)<1,1,0))  
 }  
}  
F12 = F12 %>%  
 cbind(SJ\_density\_Ep)  
hist(F12$X,freq = FALSE,ylim=c(0,1),breaks = 20,main='Epanechnikov Kernel')  
lines(F12$X,F12$SJ\_density\_Ep)



The density in this case seems smooth and doesn’t wiggle too much, while still capturing some of the variations in the distribution.

### Tri-weight

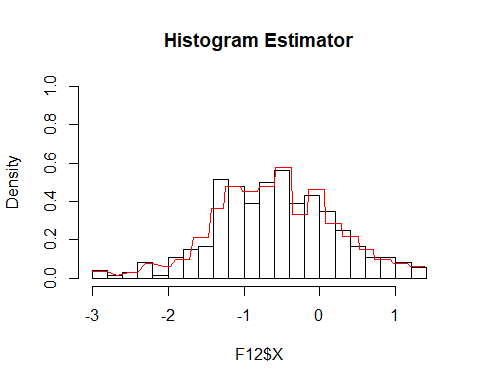
SJ\_density\_Tri = vector(mode = 'double',length = n)  
for(i in 1:length(F12$X)){  
 for(j in 1:length(F12$X)){  
 SJ\_density\_Tri[i] = SJ\_density\_Tri[i] + 1/(n\*SJ\_bw)\*(35/32\*(1-((F12$X[i]-F12$X[j])/SJ\_bw)^2)^3)\*(ifelse(abs((F12$X[i] - F12$X[j])/SJ\_bw)<1,1,0))  
 }  
}  
F12 = F12 %>%  
 cbind(SJ\_density\_Tri)  
hist(F12$X,freq = FALSE,ylim=c(0,1),breaks = 20,main='Tri-weight Kernel')  
lines(F12$X,F12$SJ\_density\_Tri)



This looks similar to the Epanechnikov Kernel, except it tracks the movement of the distribution more closely. It might be slightly overfitting, or we could say the Epanechnikov is slightly underfitting.

### Histogram

h\_bins = seq(from = min(F12$X),to = max(F12$X),by=(max(F12$X)-min(F12$X))/20)  
v\_k = h\_bins[2] - h\_bins[1]  
n\_k = NULL  
hist\_estimator = vector(mode = 'double',length = n)  
for (i in 1:(length(h\_bins)-1)){  
 n\_k[i] = sum(ifelse((F12$X >= h\_bins[i] & F12$X <= h\_bins[i+1]),1,0))  
}  
  
for(i in 1:length(F12$X)){  
 for(j in 1:length(n\_k)){  
 hist\_estimator[i] = hist\_estimator[i] + n\_k[j]/(n\*v\_k)\*ifelse((F12$X[i] >= h\_bins[j] & F12$X[i] <= h\_bins[j+1]),1,0)  
 }  
}  
F12 = F12 %>%  
 cbind(hist\_estimator)  
hist(F12$X,freq = FALSE,ylim=c(0,1),breaks = 20,main='Histogram Estimator')  
lines(F12$X,F12$hist\_estimator,col='red')



Not too sure what I expected to be honest. This basically looks like a histogram, with slightly different densities.