Statistical Data Analysis
Assignment 3

Matt Kuodis & James Zoryk.
2640428 : 2663347
Group: 36
Due Date: 15.03.2022.

Question 3.2.

In this question we shall be investigating the data in 'sample32.txt' to find a suitable kernel density estimator of the sample. In order to do this we first want to better understand the characteristics of the data at hand. We begin by doing some basic data analysis, namely we compute a numerical summary of the data, see Table 1. Here we observe that the median and mean of the data are roughly

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-2.395	5.349	8.790	9.719	13.863	25.374

Table 1: Numerical summary of 'sample32.txt'.

the same with a difference of about 1, so the data is not heavily skewed.

We continue our analysis with a graphical method of plotting the histogram. Here we also consider the number and sizing of the amount of bins present in the histogram to ensure that no information is lost in the process. The results can be seen in Figure 1, noting that we have scaled the histograms so the total area is equal to 1.

As seen in the histogram plots (figure 1) the data looks like it could be from a normal distribution. The reason for this is due to the distinctive bell curve we see in the histogram and also the fact that it is not heavy skewed. Noting that in the third histogram, the one with more bins, we see that the peak density bar of the data is not in the centre of the histogram as what one may expect from a normal distribution. The result of this, is that it appears that there are two peaks on either side of the median.

With all this in mind, we shall now begin to pick a suitable kernel function for the density function of the data. We will first find a suitable range of bandwidths for the data. In doing so we shall use the Gaussian kernel since it is one of the smoothest kernels and hence can be use in most general cases, though we shall consider the affects of kernel choices later on.

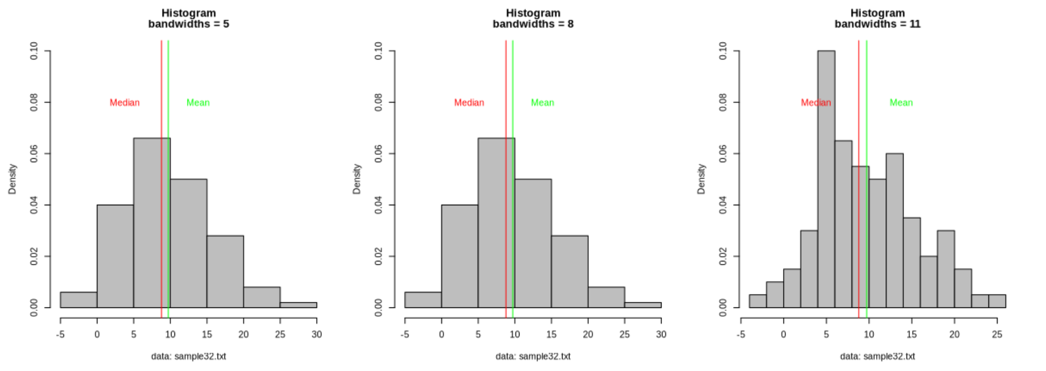

Figure 1: Histogram plots of a 'sample32.txt' with a range of bins sizes. here the mean (Green) and median (Red) lines are plotted.

To figure out which bandwidth is the most suitable for our given data, we plot the scaled histogram overlaid with the density function at various bandwidths. This is done in order to visually compare the density function to the characteristics of the histogram, the results can be seen in Figure 2.

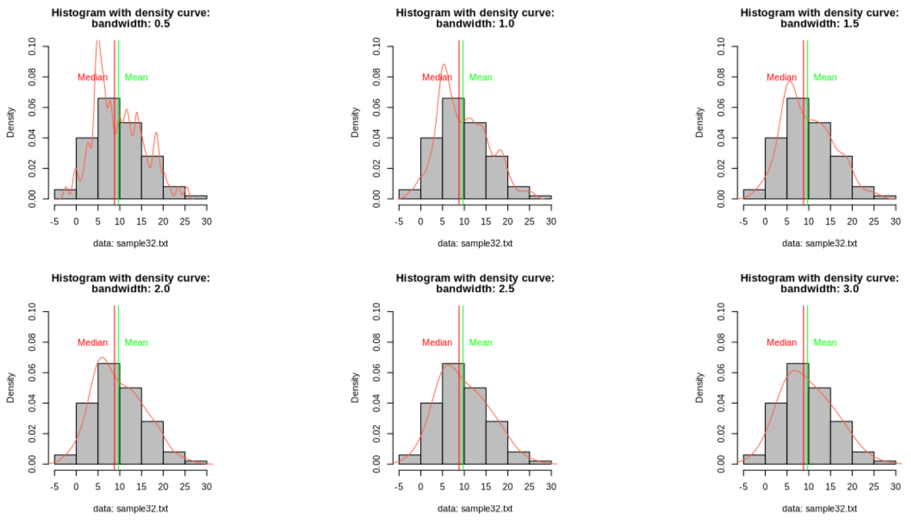

Figure 2: Histogram plots of a 'sample32.txt' with overlay density function at various bandwidths and ranges of bin sizes. The mean (Green) and median (Red) lines are plotted here.

We observe that the bandwidths of $0.5, 1$ and 1.5 are too small since they are very sensitive to changes in the density of the data. This is seen by the amount of peaks and troughs present in these density function curves. Therefore one can rule out a bandwidth within this range. On the other hand we see that the bandwidths of 2.5 and 3 are too large since they remove too much of the information about the distribution of the data which we saw in Figure 1, resulting in a rather flat curve; especially on the right side. Hence we can remove these values from consideration. This then leaves us with the bandwidth equal to 2 as a suitable choice. Observing the plot we see that this is indeed the case, since the curve passes through the middle of the bars of the histogram while remaining smooth for the most part and only forming peaks in the relative places.

We expect that the optimum bandwidth is around 2, however we still need to prove that the Gaussian kernel is the best suited kernel for our data. We now plot the histogram of the data with the density function of different kernels with bandwidths between 1.75 to 2.25. Our motivation to do so is to visually inspect the fit of each type of kernel. The results of this can been seen in Figure 3 and 4.

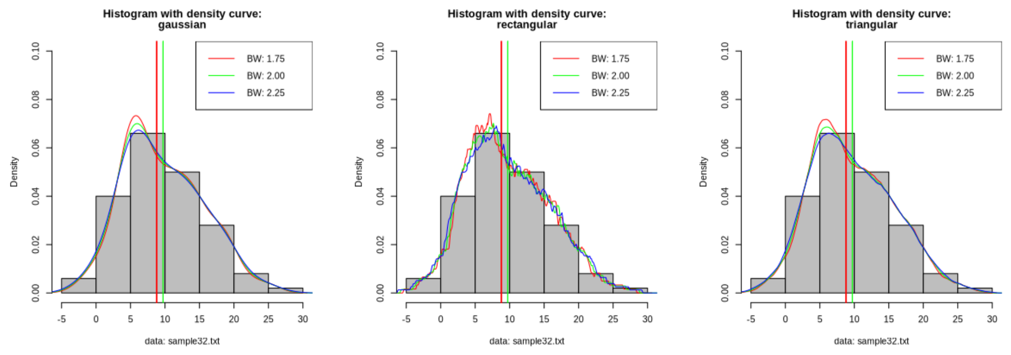

Figure 3: Histogram plots of a 'sample32.txt' with overlay density function at various bandwidths and ranges of bins sizes. The mean (Green) and median (Red) lines are plotted here.

Here, in Figure 3 it can be clearly seen that both the rectangular and triangular kernels yield a very jagged curve. Therefore we can rule these two kernel types out. While in Figure 4 we see that the Epanechnikov kernel induces more small peaks and troughs into the curve, which is not ideal. Moreover for both the biweight and cosine kernels we see that the curves are over damped at the smaller bandwidths, causing a more extreme change in the curve.

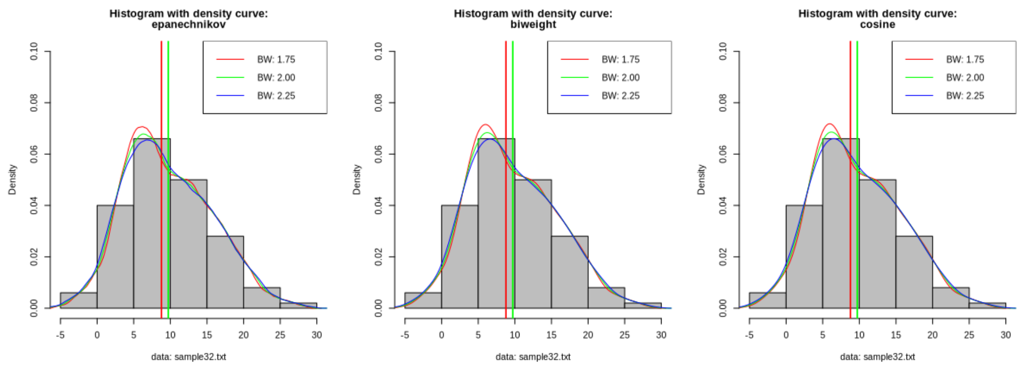

Figure 4: Histogram plots of a 'sample32.txt' with overlay density function at various bandwidths and ranges of bins sizes. The mean (Green) and median (Red) lines are plotted here.

Therefore after inspecting these plots, the smoothest density kernel is the Gaussian kernels. Hence we shall choice the Gaussian kernel with a bandwidth of 2 to approximate our data sample.

Question 3.3.

For this question we shall consider the data from 'sample33.txt', in which we aim to find a suitable estimator for the density. Our first observation about the data is similar to that of a exponential distribution, moreover that the data peaks around the zero, while the data only contains positive values. This can be an issue when trying to select a suitable density estimator for it, as a density estimator may assign weight values to point that do not exist in our data. In order to address this issue, we will symmetrise the data, which will be out new data set we shall consider for the rest of the question.

To find a suitable bandwidth, we shall use the functions 'h_opt' and the RStudio built-in function 'density' to compute a suitable value. Computing these values using our data give 'h_opt' = 0.63 and 'density'= 0.38. Using these values for the bandwidth we can then plot with a range of kernels to see the effectiveness at matching the data. The plots can be seen in Figure 6.

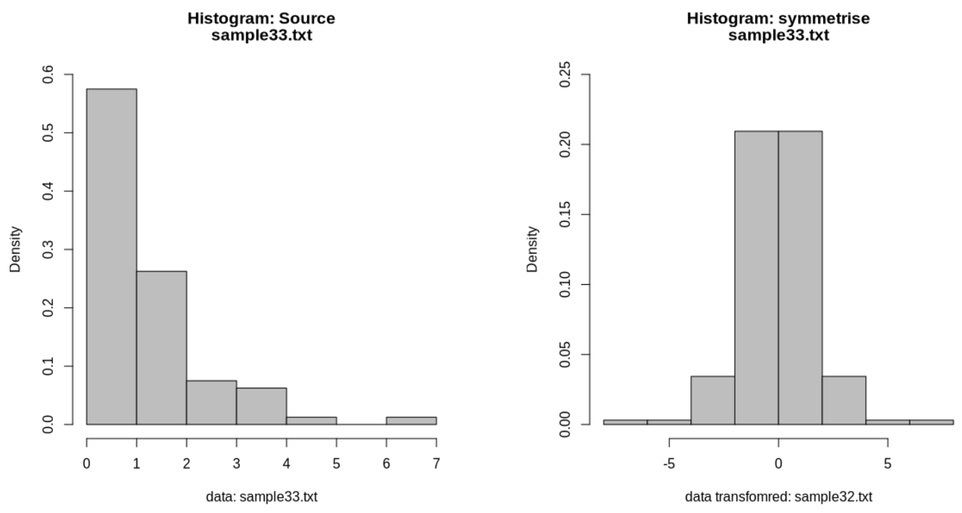

Figure 5: Histogram plot of data on the left and the symmetric on the right.

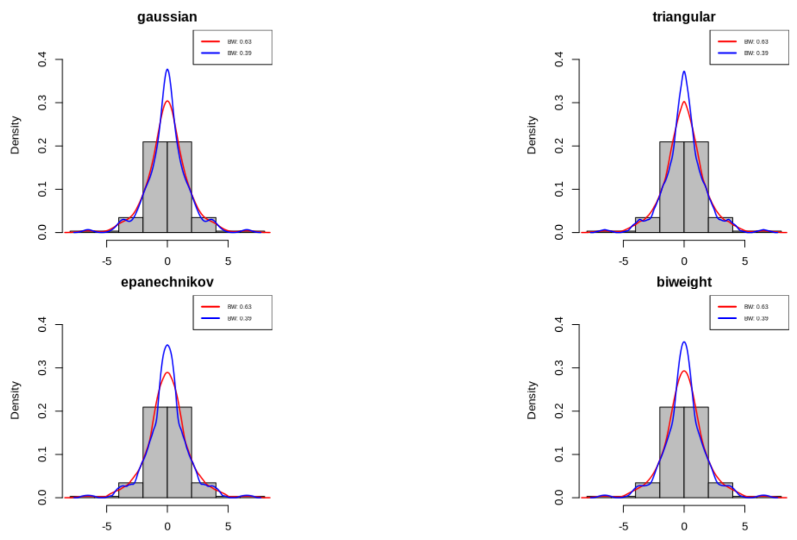

Figure 6: Histogram plot of symmetrise data with overlaying density curves at $bw = 0.63$(Red) and $bw = 0.39$ (Blue).

It becomes clear in Figure 6 that the bandwidth choice of 0.63 is too large since in the plots the curve is very smooth, hence we shall consider the bandwidth of 0.38. Moreover, for the choice of kernel, we observe that both 'triangular' and 'Epanechnikov' are very jagged for our chosen bandwidth and thus unsuitable. Then in the 'Gaussian' kernel we see that it is underdamped at various points on the curve, creating local peaks and troughs which is undesirable for our data. This leaves the 'Biweight', which we see fits the data well for our bandwidth. Therefore we choose bw= 0.38 and the Biweight kernel for the density estimation.

Question 3.4.

For this question we are given the data 'sample34.txt' that contains 90 observations. With this data set we are tasked to find two suitable kernel density estimates. The first of which, we shall compute the bandwidth by using the function 'h_opt' that is given in the file 'function_ch4', doing so gave us the bandwidth $bw_h = 0.68$.

Then the second method we used to compute the bandwidth is by Cross-validation (CV) criterion. The process to compute this is to first generate a vector with 199 values ranging between $[0, 1]$. With this we could then run a loop to compute the CV terms for four suitable kernels. In Figure 7, we plot the results of this loop, here the bandwidths are on the x-axis while the y-axis the (CV) criterion. We can then compute the *h_min* of the CV by taking the minimum value obtained in the CV criterion and its relative bandwidth for each of the kernels. The results can be seen in Table 2.

	Gaussian	Triangular	epanechnikov	Biweight
bw cv	0.355	0.330	0.330	0.335

Table 2: Numerical summary of 'sample32.txt'.

Now, we can plot the density functions with these kernels and our computed bandwidths to see which one will be suitable for matching our data.

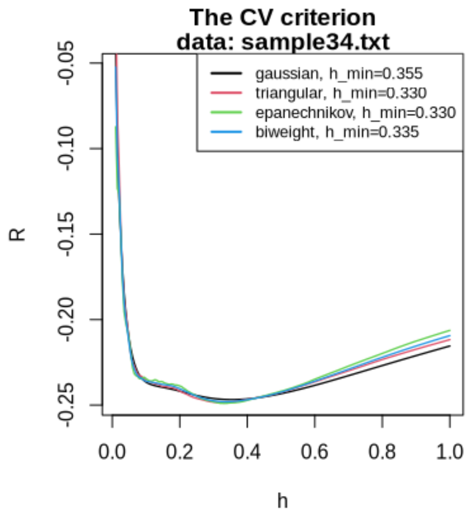

Figure 7: The Cross-Validation criterion on the y-axis as a function of the bandwidth using different kernels.

In Figure 8, we see that our computed *h_top* value is large since it is smooth with every kernel. However we see that 'Epanechnikov' looks like a suitable choice for a kernel at $bw_h = 0.68$ since it matches the data well and the peak of the curve aligns to the mean/median of the data with a smooth peak. Compared to our kernels the peak of the curve look quite sharp, which is undesirable. Then for the CV method, we observe that both the ’Triangle” and ‘Epanechnikov’ kernel are both very jagged, thus we shall rule them out. We see that in the ‘Gaussian’ kernel it underdampens on the curve, meaning that the curve oscillates causing troughs in the curve, and unlike in the ‘Biweight’ kernel, we see that it maintains a rather smooth flow of the curve at all times for the bandwidth $bw = 0.335$. Therefor we conclude that for our given data, it is best to use the ‘Biwieght’ kernel for the density estimator.

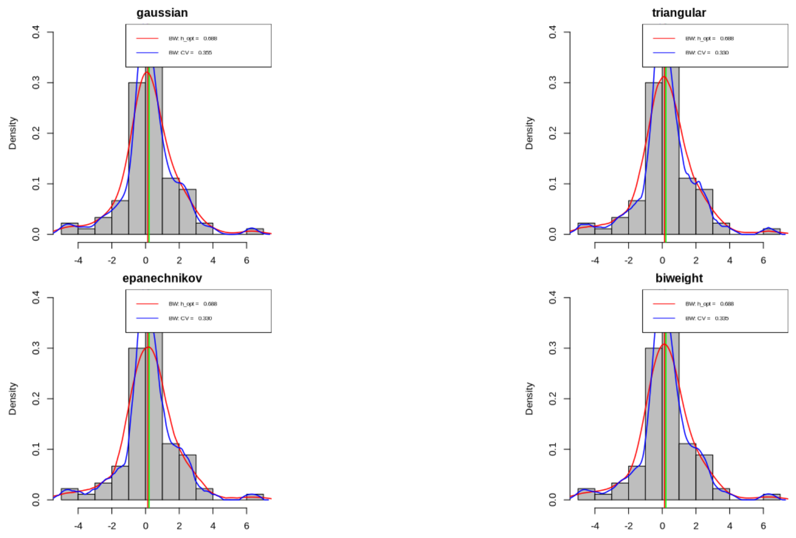

Figure 8: edit me

Now we want to compare our given data to that theoretical data, namely the double exponential distribution at $\beta = 0$ and $\sigma = 1$. We create our own random distribution function for this and plot the histogram and the density function, as seen in Figure 9. The true density function of the Laplace is more narrow compared to the sample density function, this can be seen with the large peak in the Laplace density function while with the sample the peak is below 0.4. This means that the data is more spread out than in the true data.

Question 3.5. For the final exercise, we are tasked with analysing the data from ’t-sample.txt’, specifically the Median Absolute Deviation. We use the built in function to compute the MAD value of the sample, and use bootstrap methods of the empirical, and parametric kind, to find an approximation for this value, and to gain insight into the deviation of this value itself. To build up an intuitive preference for either of the methods, we plot the sample values generated by the bootstrap methods in histograms, together with a selection of samples from the true distribution, which is given.

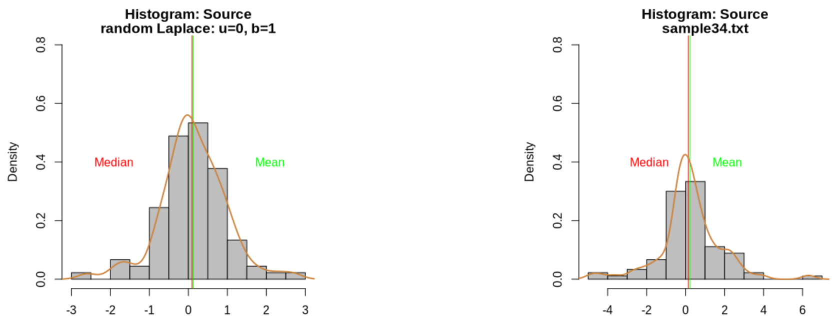

Figure 9: The true Laplace sample and density function (Left) compared to sample and density function (Right).

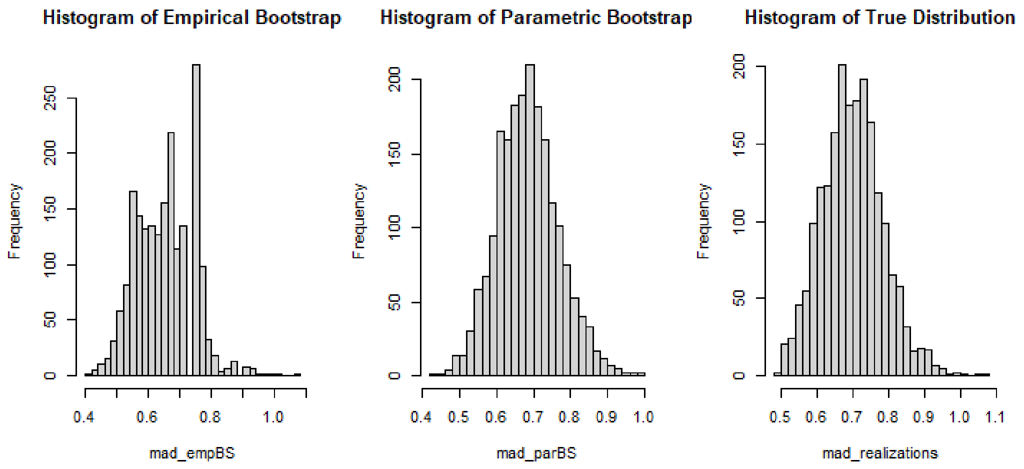

Figure 10: Sample MAD values gathered through empirical bootstrap, parametric bootstrap, and the true distribution.

We note that the parametric bootstrap does a much better job of approximating the true distribution of the data, than the empirical bootstrap. To support this, we look at the data summary of the various sample sets that are generated, and note that the parametric bootstrap sample fits much better than the empirical bootstrap sample, except for extreme values, which are more similar between the empirical and true distribution samples.

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.4085	0.5727	0.6464	0.6495	0.7143	1.0763

Table 3: Numerical summary of empirical bootstrap.

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.4375	0.6268	0.6816	0.6841	0.7354	0.9838

Table 4: Numerical summary of parametric bootstrap.

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.4880	0.6413	0.6966	0.6992	0.7523	1.0606

Table 5: Numerical summary of the true realization samples.

A brief note about the functions is in order. We set ”constant = 1” in the built in median absolute deviation function, since our data needs not be normalized such that the MAD is equal to the standard variance. This is so that we do not need to upscale the values we compute later on.

The RData file contains the important results, but we shall briefly outline them here. The MAD value we compute from the sample with the built in function is 0.65, which is best approximated by the mean of the empirical bootstrap sample. Though the data seems to come from the t-distribution with 10 degrees of freedom, and our approximate parametric sample seems to approximate it relatively well, the empirical bootstrap approximates the MAD value of the data the most accurately. This is most likely due to the fact that the empirical bootstrap is generated from samples of the actual data, which may come from a certain distribution, but may not represent that distribution the best.

Another bit of supporting data comes from the Shapiro-Wilk test on the ’t-sample’ data, which returns a large p-value, further supporting the empirical bootstrap method as the appropriate one to examine this data set.

R code for exercise 3.2

```
####################################################
# SDA HW3
# Exercise: 2
####################################################
# Pull data from disk
setwd("~/Maths/SDA/Assignment_3")
#read file
data <- scan("sample32.txt")

# get data summary
summary(data)

# comparing hist bw
par(mfrow=c(1,3)) # set 2x3 plot display
par(pty="s")      # set aspect ration of plot
list_breaks =  seq(5, 11, 3)  # create a vector of suitable breaks
# loop for plotting histograms
for(i in 1:length(list_breaks)){
hist(data,breaks = list_breaks[i],
     prob = TRUE, col = "grey",
     ylim = c(0, 1/10),
     xlab= "data: sample32.txt",
     main = c("Histogram",sprintf("bandwidths = %.1i", list_breaks[i])
         ))
     abline(v = median(data), col = "red",lwd = 1) # median line
     abline(v = mean(data), col = "green",lwd = 1) # mean line
     text(x = median(data)+5, y= 0.08,"Mean" , col="green") # text
     text(x = median(data)-5, y= 0.08,"Median" , col="red")


}

# Lists
list_kernel = c("gaussian", "rectangular", "triangular","epanechnikov"
    ,"biweight","cosine","cosine")
list_colours = c("red", "green", "blue","pink","tomato", "peru","
    paleturquoise")

# bandwidth selection
list_band = seq(0.5, 3,0.5)
par(mfrow=c(2,3)) # set 2x3 plot display
par(pty="s")      # set aspect ration of plot
# loop for plotting histogram overlayed with density plots
for( i in 1:length(list_band) ){
  hist(data, prob = TRUE, col = "grey",
       ylim = c(0, 1/10),
       main = c("Histogram with density curve:",
                 sprintf("bandwidth: %0.1f",list_band[i])),
       xlab= "data: sample32.txt",
  )

  lines(density(data, bw=list_band[i]),
        col="tomato",
         lwd= 1.0)
   abline(v = median(data), col = "red",lwd = 1) # median line
   abline(v = mean(data), col = "green",lwd = 1) # mean line
   text(x = median(data)+5, y= 0.08,"Mean" , col="green")
   text(x = median(data)-5, y= 0.08,"Median" , col="red")


}


# bandwidth plus kernel
par(mfrow=c(1,3))
list_band = seq(7/4, 9/4, 1/4)
par(pty="s")
for( i in 1:6 ){
   hist(data, prob = TRUE, col = "grey",
        ylim = c(0, 1/10),
        main = c("Histogram with density curve:", list_kernel[i] ),
        xlab= "data: sample32.txt"

   )
   for(j in 1:length(list_band)){
     lines(density(data,
                   kernel = list_kernel[i],
                   bw=list_band[j]),
          col=list_colours[j],
          lwd=1
          )
     abline(v = median(data), col = "red",lwd = 1) # median line
     abline(v = mean(data), col = "green",lwd = 1) # mean line
#     text(x = median(data)+5, y= 0.08,"Mean" , col="green")
#     text(x = median(data)-5, y= 0.08,"Median" , col="red")
   }
     legend("topright",
            c(sprintf("BW: %0.2f", list_band[1:3])),
            col = c(list_colours[1:3]),
            lty = 1)

}
```

code/SDA_hw3_ex2.R

R code for exercise 3.3

```
#####################################################
# SDA HW3
# Exercise: 3
#####################################################
# Pull data from disk
setwd("~/Maths/SDA/Assignment_3")
#read file
data_source <- scan("sample33.txt")
source("functions_Ch4.txt")
source("functions_Ch5.txt")
data <-c(data_source, -data_source )
```

```
# comparing hist bw
par(mfrow=c(1,2)) # set 2x3 plot display
par(pty="s")      # set aspect ration of plot
# over lay
hist(data_source, prob = TRUE, col = "grey",
     ylim = c(0, 3/5),
     main = c("Histogram: Source", "sample33.txt"),
     xlab= "data: sample33.txt",)
hist(data, prob = TRUE, col = "grey",
     ylim = c(0, 0.25),
     main = c("Histogram: symmetrise", "sample33.txt"),
     xlab= "data transfomred: sample32.txt",)


# compute h_opt band width
  data_density <- density(data)
  bw_opt = h_opt(data)
  list_bw = c(bw_opt, unlist(data_density["bw"], use.names = FALSE))


  # Lists
  list_kernel = c("gaussian", "triangular","epanechnikov","biweight","
      cosine")
  list_colours = c("red","blue", "green", "pink","tomato", "peru","
      paleturquoise")

    # bandwidth plus kernel
  par(mfrow=c(2,2))
  par(pty="s")
  par(mar = c(2, 0.1, 2, 0.1))    # reduce white space around plots
  for( i in 1:4 ){
    hist(data, prob = TRUE, col = "grey",
         ylim = c(0, 0.45),
         main = c(list_kernel[i] ),
         xlab= "data transfomred: sample32.txt"

    )
    for(j in 1:2){
      lines(density(data,
                    kernel = list_kernel[i],
                    bw=list_bw[j]),
            col=list_colours[j],
            lwd=1.5
      )
    }

    legend("topright",
           legend= c(sprintf("BW: %0.2f", list_bw[1:2])),
           col = c(list_colours[1:2]),
           lty=1:1, lwd=2)

  }
```

code/SDA_hw3_ex3.R

R code for exercise 3.4

```
####################################################
# SDA HW3
# Exercise: 4
####################################################

# Pull data from disk
setwd("~/Maths/SDA/Assignment_3")
#read file
data<- scan("sample34.txt")
source("functions_Ch4.txt")
source("functions_Ch5.txt")

# Hist
par(mfrow=c(1,1))
par(pty="s")
par(mar = c(5, 0.1, 2, 0.1))    # reduce white space around plots
hist(data, prob = TRUE, col = "grey",
     ylim = c(0, 0.4),
     main = c("Histogram: Source", "sample34.txt"),
     xlab= "data: sample34.txt",)
abline(v = median(data), col = "red",lwd = 1) # median line
abline(v = mean(data), col = "green",lwd = 1) # mean line
    text(x = median(data)+2, y= 0.4,"Mean" , col="green")
    text(x = median(data)-2, y= 0.4,"Median" , col="red")

# compute bw
data_density <- density(data)
bw_opt = h_opt(data)

#CV
h_vec = seq(0.01, 1, 1/200)
# list
list_kernel = c("gaussian", "triangular","epanechnikov","biweight","
    cosine")


# create space
h_gauss=0
h_trianular=0
h_epanechikov=0
h_biweight=0
cv_gauss <- c()
cv_trianular <- c()
cv_epanechikov <- c()
cv_biweight<- c()
# list cont
list_cv = list(cv_gauss, cv_trianular, cv_epanechikov, cv_biweight)
list_hmin = list(h_gauss, h_trianular, h_epanechikov, h_biweight)
# loop to compute cv of 4 kernels
```

```
for(i in 1:length(list_cv)){
  list_cv[[i]] <- sapply(h_vec, CV, sample=data, kernel=list_kernel[i
      ])
  list_hmin[[i]] <- h_vec[which(list_cv[[i]] == min(list_cv[[i]]))]
}

#plot cv criterion
par(mfrow=c(1,1))
par(pty="s")
plot(h_vec, list_cv[[i]], type="n",
      main = c("The CV criterion","data: sample34.txt"),
      xlab= "h",
      ylab ="R" )
for (i in 1:length(list_cv)){
  lines(h_vec, list_cv[[i]], col=i , lwd=1.3)
}
legend("topright",
        legend= c(sprintf("%s, h_min=%0.3f", list_kernel[1:length(list_
            cv)],list_hmin[1:length(list_cv)])),
        col = seq(1,4,1),
        lty=1:1, lwd=2, cex = 0.8,
)



# Lists

list_colours = c("red","blue","pink", "green","tomato", "peru","
    paleturquoise")
list_bw = c(bw_opt,h_min)
list_name_bw = c("h_opt = ", "CV = ")


# bandwidth plus kernel
par(mfrow=c(2,2))
par(pty="s")
par(mar = c(2, 0.1, 2, 0.1))    # reduce white space around plots
for( i in 1:4 ){
  hist(data, prob = TRUE, col = "grey",
        ylim = c(0, 0.4),
        main = c(list_kernel[i] ),
        xlab= "data transfomred: sample34.txt"

  )
  abline(v = median(data), col = "red",lwd = 1.5) # median line
  abline(v = mean(data), col = "green",lwd = 1.5) # mean line
  lines(density(data,
                    kernel = list_kernel[i],
                    bw=bw_opt),
            col=list_colours[1],
            lwd=1.5
  )
  lines(density(data,
                  kernel = list_kernel[i],
                bw=unlist(list_hmin[i], use.names = FALSE)),
         col=list_colours[2],
         lwd=1.5
   )
   legend("topright",
          legend= c(sprintf("BW: %s   %0.3f", list_name_bw[1:2],c(bw_opt
              , unlist(list_hmin[i], use.names = FALSE)) )),
          col = c(list_colours[1:length(list_bw)]),
          lty=1:1, lwd=1, cex = 0.6,
   )


}



# compare
# make our own random gen function for double exp / laplace
rlaplace = function(n,mu,sigma){
  #create list of random values
  U = runif(n,0,1)
  #This will give negative value half of the time
  sign = ifelse(rbinom(n,1,.5)>.5,1,-1)
  y = mu + sign*sigma/sqrt(2)*log(1-U)
  y
}

par(mfrow=c(1,2))
par(pty="s")
par(mar = c(2, 0.1, 2, 0.1))    # reduce white space around plots

rx = rlaplace(90,0,1)
hist(rx, prob = TRUE, col = "grey",
     ylim = c(0, 0.6),
     main = c("Histogram: Source", "random Laplace: u=0, b=1"),
     xlab= "data: sample34.txt",)


abline(v = median(rx), col = "red",lwd = 1) # median line
abline(v = mean(rx), col = "green",lwd = 1) # mean line
text(x = median(rx)+2, y= 0.4,"Mean" , col="green")
text(x = median(rx)-2, y= 0.4,"Median" , col="red")
lines(density(rx), col="peru", lwd=2.0)


hist(data, prob = TRUE, col = "grey",
     ylim = c(0, 0.6),
     main = c("Histogram: Source", "sample34.txt"),
     xlab= "data: sample34.txt",)
abline(v = median(data), col = "red",lwd = 1) # median line
abline(v = mean(data), col = "green",lwd = 1) # mean line
text(x = median(data)+2, y= 0.4,"Mean" , col="green")
text(x = median(data)-2, y= 0.4,"Median" , col="red")
lines(density(data), col="peru", lwd=2.0)
```

code/SDA_hw3_ex4.R

R code for exercise 3.5

```
##########################################
# SDA HW3
# Exercise: 5
##########################################
# Pull data from disk
setwd("~/Desktop/P4_and_P5_2022/SDA_Assignments/SDA3")
#read file
data <- scan("t-sample.txt")
source("functions_Ch4.txt")
source("functions_Ch5.txt")

# get data summary
summary(data)


#set the seed and establish the amount of samples
set.seed(20220302+36)
B = 2000
sample_size=100
#compute the MAD value for the t-sample data, using the built in
    function
mad_sample = mad(data, constant=1)

#compute empirical bootstrap
mad_empBS=numeric(B)

for (i in 1:B) {

  xstar = sample(data, replace=TRUE)
  xbar = median(data)

  deviations = numeric(sample_size)
  for (j in 1:sample_size) {
    deviations[j] = abs(xstar[j] - xbar)
  }

  mad_empBS[i] = median(deviations)

}


#compute the parametric bootstrap
mad_parBS=numeric(B)
#estimate for the degrees of freedom
s = var(data)
k = (2*s)/(s-1)

for (i in 1:B) {
  xstar2 = rt(sample_size, k)
  xbar2 = median(xstar2)

  deviations2 = numeric(sample_size)
  for (j in 1:sample_size) {
    deviations2[j] = abs(xstar2[j] - xbar2)
  }

  mad_parBS[i] = median(deviations2)

}

#sample from the true distribution
mad_realizations = numeric(B)
truek = 10

for (i in 1:B) {
  xstar3 = rt(sample_size, truek)
  xbar3 = median(xstar3)

  deviations3 = numeric(sample_size)
  for (j in 1:sample_size) {
      deviations3[j] = abs(xstar3[j] - xbar3)
  }

  mad_realizations[i] = median(deviations3)

}

#plot histograms of the empirical, parametric, and true computations
    for MAD
par(mfcol=c(1,3))
hist(mad_empBS, main="Histogram of Empirical Bootstrap", breaks=25)
hist(mad_parBS, main="Histogram of Parametric Bootstrap", breaks=20)
hist(mad_realizations, main="Histogram of True Distribution", breaks
    =25)

summary(mad_empBS)
summary(mad_parBS)
summary(mad_realizations)

var_empBS = var(mad_empBS)
var_parBS = var(mad_parBS)
var_realizations = var(mad_realizations)

#save the RData file
mylist=list(studno =c(2663347 , 2640428), mad_sample = mad_sample,
            mad_empBS = mad_empBS, mad_parBS = mad_parBS,
            mad_realizations = mad_realizations,
            var_empBS = var_empBS, var_parBS = var_parBS,
            var_realizations = var_realizations)
save(mylist, file ="~/Desktop/P4_and_P5_2022/SDA_Assignments/SDA3/
    myfile3_36.RData")
```

code/SDA_hw3_ex5.R