SDA- Assignment 1

Jose Chacon (2699643)

Exercise 3.2

Description of the Sample

For the first exercise, we are given a txt file called sample32, which is a sample that contains 150 observations. Therefore, our first task is to investigate this sample. So, we have plotted some histograms of the sample with different numbers of breaks to better understand how the data is distributed (shape, skewness, and other statistics). It is evident from these plots that the data is right-side skewed. We can also see an almost bell-shape density with two tops. Two of these histograms can be found in the appendix.

Description of the Analysis

The goal of this analysis is to find an appropriate kernel density estimator based on the above-described sample. Therefore, we have to systematically find candidates for the best kernel density estimator. A kernel density function is based on the choice of two parameters: the kernel and the bandwidth. The bandwidth is usually the most important part because it decides how much detail will the density estimate have.

The first step of the process is to choose a kernel. The default kernel in the *density* function in R is Gaussian. From the theory, we know that even if the underlying distribution of the sample is not normal; we will obtain an accurate estimator. This is due to the fact that the estimator used is dependent only on the sample variance and sample size, and not in any particular characteristic of the underlying distribution. This yields a well-enough estimation for non-normal distributions. Therefore, we will begin the analysis using the Gaussian kernel.

However, it is very important to understand how the different kernels have an impact on the estimator. Therefore, we have repeated the whole KDE process with the following kernels: epanechnivok, triangular, and rectangular. Only the plots of the most relevant KDE estimators will be shown in the appendix.

The second step is to choose the bandwidth. First, we created an array with a sequence of bandwidths that we wanted to test. The sequence of values depended on which kernel we were testing (some kernels are more sensitive than others some we have to test with a range of smaller values). Furthermore, there are some built-in functionalities in the *density* function that allows us to calculate the bandwidths for different scenarios. The resulting KDE of these bandwidths are going to be presented below. We used the following commands:

1. *"nrd"*: is the default choice and it uses the rule-of-thumb or Silverman's rule (bw=0.202)
2. *"nrd0"*: similar to *"nrd"* but is less sensitive to outliers (bw=0.171)
3. *"ucv"*: unbiased cross-validation: minimizes MISE (bw=0.224)
4. *"bcv"*: biased cross-validation: biased correction added (bw=0.126)
5. *"sj"*: uses the Sheater-Jones method (bw=0.142)

Note: the bandwithd obtained from the *nrd* parameter is the same as the one obtained when using the *h_opt* function. The *h_opt* formula is based on the sample standard deviation of the normal distribution, we could use a different $\int (g'')^2(t)\,dt$ to test an optimal bandwidth for other distributions. To fully cover all aspects of the analysis, we also computed the optimal bandwidth of distributions mentioned below. We decided not to include these plots due to their lack of improvement from the ones already found.

(a) logistic distribution: with optimal bandwidth of 0.47, the density is over-smoothed

(b) double exponential distribution: with optimal bandwidth of 0.28, still in the range of over-smoothness

(c) exponential distribution: with optimal bandwidth of 0.09, too many details for an estimation.

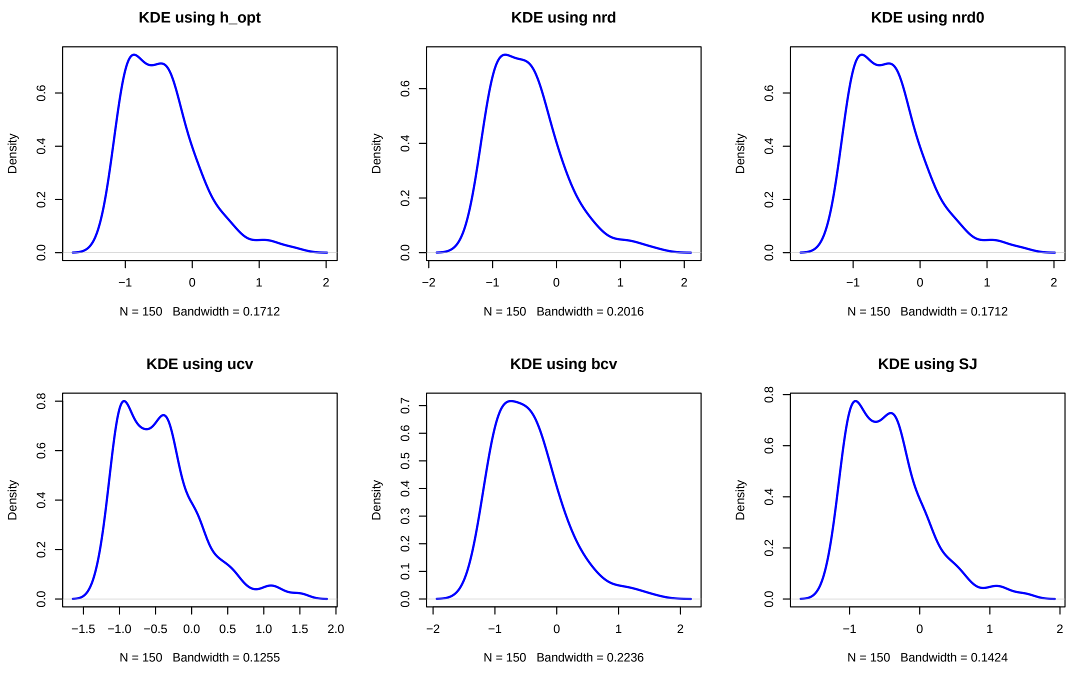

Figure 1: Kernel Density Estimator using different bandwidths

From the estimators above, we can see that the KDE using nrd (bw=0.201) and bcv (0.223) in the bandwidth parameter will cause the density to be over-smoothed, meaning that relevant information about the sample is lost. On the contrary, we see that parameters ucv (0.126) and sj (0.142) will show a much more detailed density, which in this specific case were too much. Finally, we see that the bandwidth obtained from h_opt (0.171) and nrd0 (0.172) maintains a balance between details and smoothness.

Conclusion

The appropriate choice for the kernel and the bandwidth depends on the characteristics of the sample data, and the analysis we would like to perform on it. In this case, having systematically tried several kernels with different values, we conclude that the best estimate (in our opinion, of course; this is subjective and depends on the usage of the estimate) is using the Gaussian kernel with a bandwidth of 0.17. We noticed that this would be very close to the bandwidth obtained with the ”nrd0” parameter; which means it is a variation of the rule of thumb with less sensitivity to outliers. Our choice is based on two important factors. First, we did not want the plot to be ”over-smoothed”, because we would have lost plenty of relevant information about the distribution. Second, we wanted to show a good balance between the details we show and the details of the sample. We believed we have achieved this by using the mentioned kernel and bandwidth.

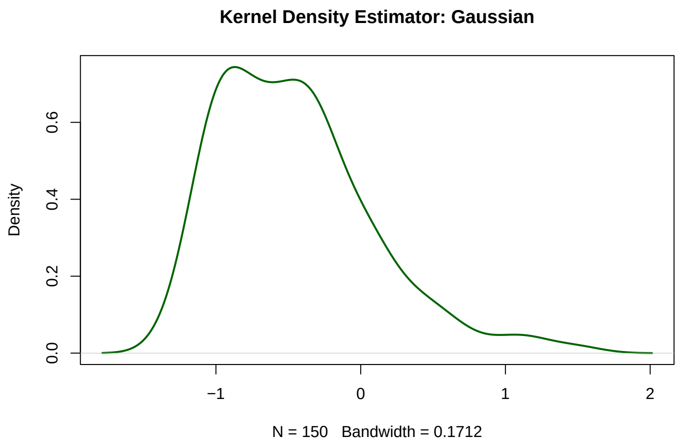

Figure 2: Kernel Density Estimator for the underlying distribution

Exercise 3.3

Problem Description

Just as in the previous exercise, we are looking for the best kernel density estimate based on a sample. The file named *sample33* contains a sample of 100 positive observations. However, because the sample contains only positive values, the kernel density estimator may not accurately represent the sample distribution. The KDE may include negative values, but this won't be in alignment with the positive values of the sample. To address this problem, we can perform some transformations to the data.

In this case, we are going to transform the sample using the logarithmic function: $\log(x)$. When applying a log-transformation we compress the range and make all values comparable. This will result in a better estimation of the density.

Analysis

We performed the following steps:

1. transform the sample: using the logarithm of each value of the sample, the data will be transformed into a new space with a less-skewed distribution
2. choose a kernel function: we mainly used the Gaussian because it yields a good enough estimate regardless of the underlying distribution. However, other kernels were also investigated. The KDEs plots calculated with help of the Epanechivok kernel are shown in the appendix.
3. choose a bandwidth: several bandwidths were used including nrd, nrd0, ucv, bcv, sj and more.
4. estimate the KDE: with help of the function *density* (or methods mentions below)
5. transform-back the density: into original scale using $f_y(t) = f_x(exp(t)) * exp(t)$
6. plot its density: on the original scale with help of the *plot* function in R

Implementation

We have found two ways to code this in R. First, we can achieve the solution by following the hint code (defining *yrange* and plotting the transformed density with the *lines* and *plot* functions). The second way to solve this exercise is using the package called *logKDE*. As its name suggests, it is a package used to find the KDE of a sample (with strictly positive values) with help of the log-transformation. It contains the functions *logdensity* and *logdensity_fft* , both methods compute the desired KDE in one code of line (the method *logdensity_fft* uses the fast Fourier technique).

As with the method *density*, we can choose in *logdensity* and *logdensity_fft* between a series of parameters to play with the KDE. We can change the kernel, the bandwidths, weights and much more. To illustrate the similarities of both approaches (hint code vs. package *lodKDE*, we have plotted the KDEs below.

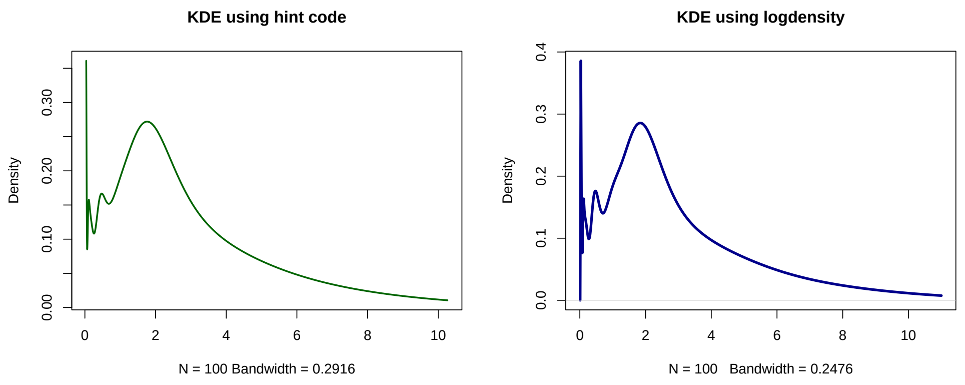

Figure 3: Kernel Density Estimator: using different implementation methods

Conclusion

After log-transforming the sample, we have concluded that the best KDE estimator for this sample with positive values is obtained by using the Gaussian kernel and applying the rule of thumb when selecting the optimal bandwidth (also obtainable with the parameter 'nrd'. Additionally, we would like to mention a second candidate with a slightly lower bandwidth. This density uses a bandwidth of 0.257 (while the nrd-bandwidth equals 0.2916). We believe that both estimates can encompass the sample and therefore have a good balance between the shown details and the smoothness of the plot. Both plots can be found in the figure below.

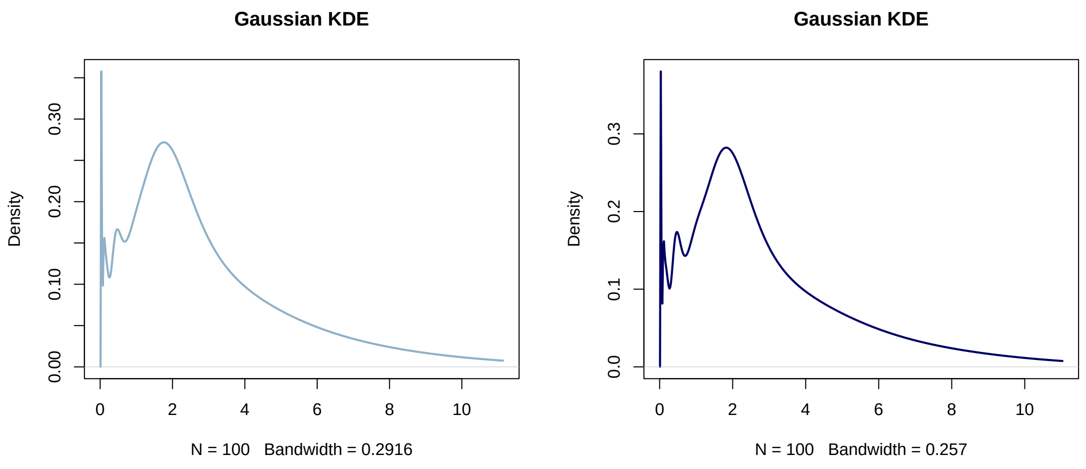

Figure 4: Optimal KDE after log-transformation

To ensure a proper representation of the sample by the kernel density estimator, we must transform back the log-transformed kernel density estimator to the original scale of the data. This transformation is possible with the formula mentioned on the step 5 of the analysis (using the exponential function). In the appendix, we provide plots of the density using the original sample without any transformation, the sample after an incorrect transformation, and the sample after a correct transformation. These plots allow us to compare the accuracy of the density estimates obtained with and without a proper transformation of the data.

Exercise 3.4

We are given a file called *list34.RData*, which contains a list with the following vectors: a sample with 120 observations, a "true" density x with 512 observations, and a "true" density y. We called the sample, *samples33* so we can perform some analysis on it. Note: it is not possible to know the true density; however we have accurately approached the "real" values by simulating them repeatedly.

Bandwidth via Silverman's Rule

First, we calculate the optimal bandwidth with the local function *h_opt*, which uses Silverman's rule to compute the bandwidth and is shown below. As explained before, this formula depends on the sample standard deviation and the sample size. For this specific sample, the optimal bandwidth using *h_opt* is 0.0078. The formula *h_opt* will be shown in the appendix as well as the plot using the computed bandwidth.

Bandwidth via Cross-validation

Second, we also estimated the density but this time using the cross-validation criterion on the interval [0.0005, 0.008]. So, we apply the given CV function to several different bandwidths on this interval; and we selected the minimum as our bandwidth. In this case, we obtained a bandwidth of 0.0036, which is significantly lower than before. This plot can be found in the appendix.

There are two ways to implement the cross-validation method. First, we can use the code given in the assignment instructions, which combined the *sapply* function with the given *CV* function to create a vector with bandwidths and then select the lower. Alternatively, we can use the below-shown code to compute the optimal bandwidth using cross-validation with biased and unbiased terms. Both the given method and this alternative (with unbiased terms) resulted in a bandwidth of approximately 0.0036.

```
bw.bcv(sample34) #0.0073
bw.ucv(sample34) #0.0037
```

Comparison of results

First, we plotted from the given samples, the ”true density of the data. We would like to mention that the three computed densities (KDE with H_opt, KDE with CV and the true density) are plotted separately in the appendix. However, the following figure will help us find the best estimate based on the true density .

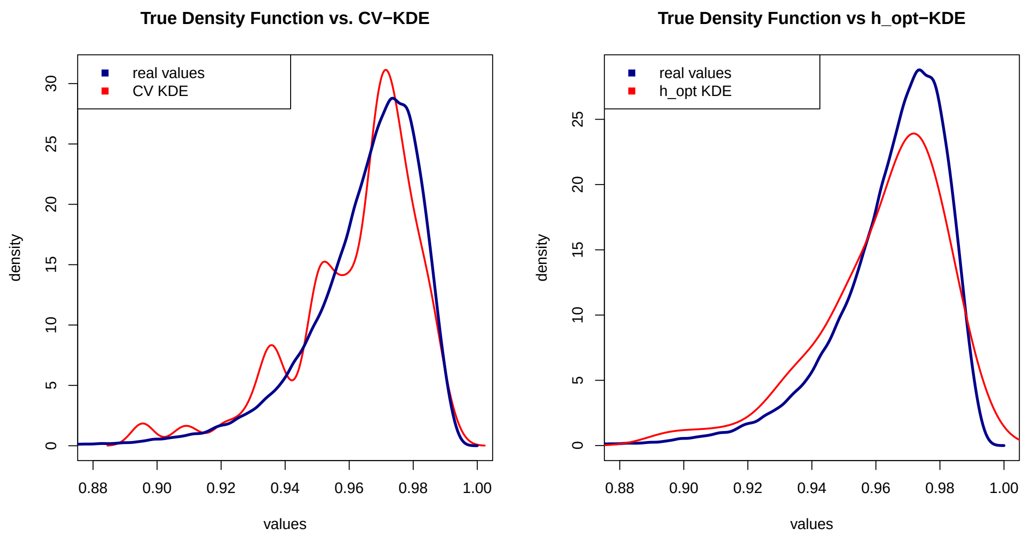

Figure 5: True Density of the Sample

From these plots, we can conclude that the estimate calculated with the *h_opt* bandwidth is better and preferable for our sample data. This bandwidth was calculated through Silverman’s rule (also known as the rule of thumb). It is preferable because of the similarities with the real distribution. We can clearly see that the estimated density with the cross-validation criteria is less than half the bandwidth *h_opt*. Consequently, the CV plot is much more detailed than needed (very inconsistent and wobbly). On the other hand, the *h_opt* plot has a similar smoothness as the true density. Therefore, the bandwidth computed by Silverman’s rule yielded a better KDE for this specific sample.

Because the *h_opt* formula is based on the sample standard deviation of the normal distribution, we could use a different $\int (g'')^2(t)\,dt$ to test an optimal bandwidth for other distributions. From the list given in the assignment, we have calculated the optimal bandwidth of the following distributions:

1. logistic distribution (with optimal bandwidth of 0.018)

2. exponential (with optimal bandwidth of 0.0037)

3. double exponential (with optimal bandwidth of 0.011)

Form these 3 plots, we can see that the best estimation would be the exponential. Even though it has a similar bandwidth than the CV-criterion method, which means that it contains more details than needed, we can see that the results are better compared to the other KDEs (using the optimal bandwidth of double exponential and logistic distribution). We see that the optimal bandwidths for the logarithmic and double exponential are too high, so they over-smooth the plot. This means that the balance between smoothness and the details of the density are in a better appropriate proportion when using the h_opt for the exponential distribution. However, we must mention that this KDEs is not an improvement from the KDE found with the optimal bandwidth for the normal distribution. Note: these plots can be found in the appendix.

Exercise 3.5

In this exercise, a sample from a t-distribution with an unknown k degree of freedom is given. We would like to estimate the statistic T = $\widetilde{F}(1)$ (the empirical cumulative distribution evaluated at 1), to ultimately investigate its distribution.

A couple of steps were performed in order to obtain the estimation of the T-statistic distribution. All steps will be mentioned; however, not all steps will be detailed explained.

(a) We computed the $\widetilde{F}(1)$ of the given sample, and its result is 0.78. The complete graph of the sample's empirical distribution function can be found in the appendix.

(b) Through the empirical bootstrap method, we have computed the estimates of 2000 t-statistics and saved them in a vector called $ecdf1\_empBS$.

(c) Through the parametric bootstrap method and using an estimator for the degrees of freedom, we have computed the estimates of 2000 t-statistics and saved them in a vector called $ecdf1\_parBS$.

(d) Furthermore, we have simulated 2000 independent samples of 50 observations from a t-distribution with 5 degrees of freedom. We have also computed the T-statistic of every simulation and saved it in a vector called $ecdf1\_realizations$. Then we plotted the histogram of this vector as well as the vectors mentioned in subsections b) and c). The histograms are shown in the figure below.

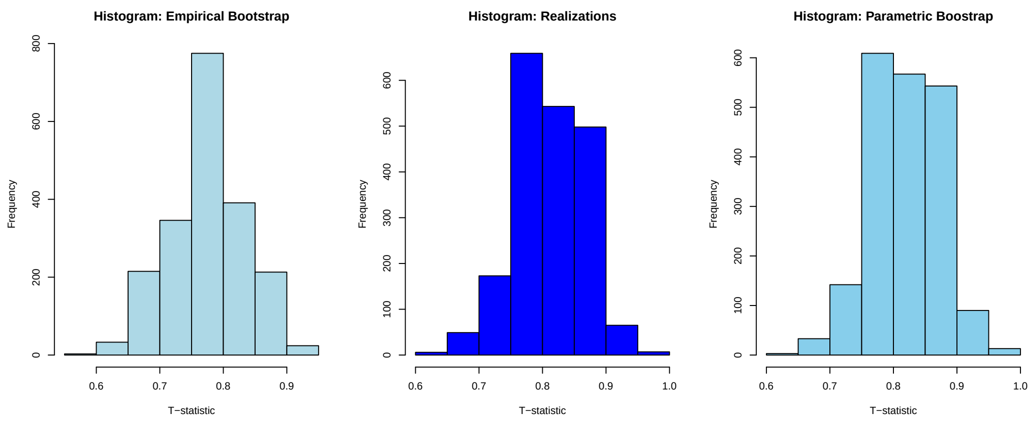

Figure 6: Histograms of bootstraps methods and the true distribution of t

From these histograms as well as for the average T-statistic, we have concluded that the parametric bootstrap method is preferable in this context. We see that the empirical bootstrap method yields a bell-shaped histogram, while both the real and the parametric bootstrap histograms are left skewed around the same values. The average T-statistic of the empirical bootstrap method sample is 0.779, while the average for the real and the parametric bootstrap method sample are 0.817 and 0.824, respectively.

The estimator $k = \frac{2s^2}{s^2-1}$ will estimate k to approximately 7 degrees of freedom, which is close enough to the real value 5. This is evidenced by the similitude in the histograms as well as in the average T-statistic.

(e) The last step involve the use of the empirical and parametric bootstrap samples to find estimates for the standard deviation of T. Finally, we compared it with the standard deviation from the true distribution. The results for the empirical, parametric, and real standard deviation for each corresponding sample are 0.059, 0.054, and 0.054, respectively. From these results, we see clearly that the parametric and real samples have virtually the same standard deviation.

Appendix

Plots 3.2

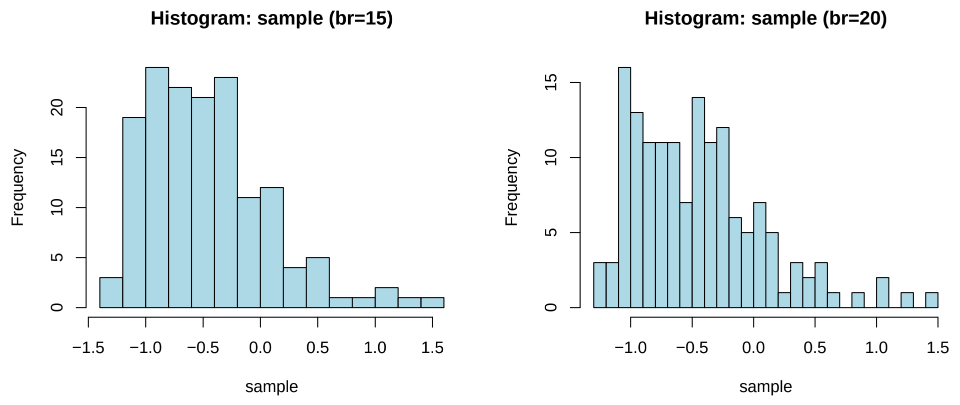

Figure 7: Supporting plots: histograms with different breaks

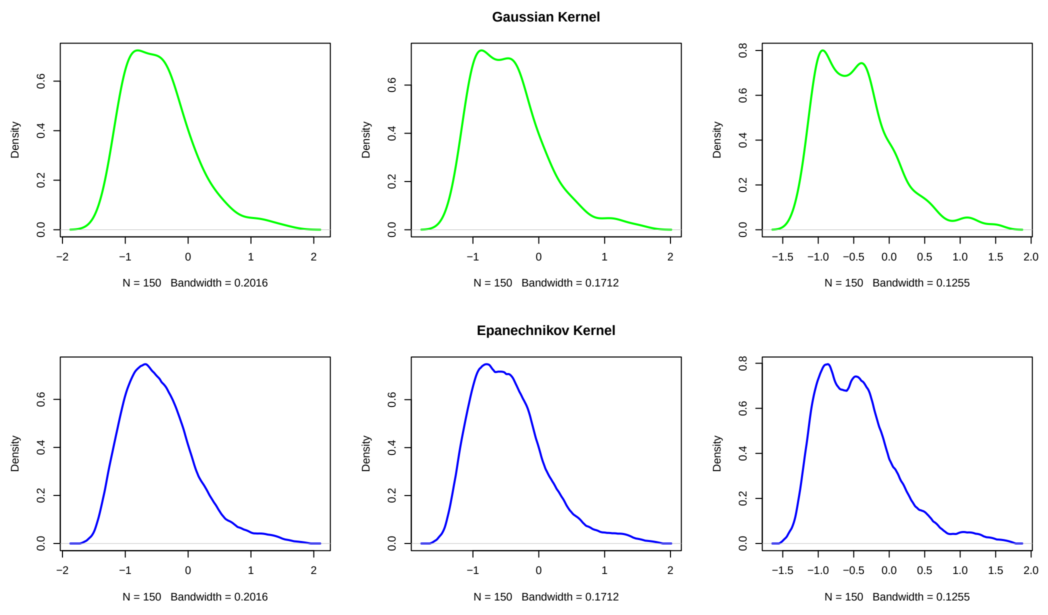

Figure 8: Kernel Density Estimations

Code 3.2

```
setwd("C:\\Users\\josec\\OneDrive\\Documentos\\VU\\Statisitcal Data Analysis\\Assignment 3")
source("functions_Ch4.txt")
sample32=scan("sample32.txt")

par(mfrow=c(1,2))
hist(sample32, breaks=15, main='Histogram: sample (br=15)', xlab='sample',
col='lightblue')
hist(sample32, breaks=20, main='Histogram: sample (br=20)', xlab='sample',
col='lightblue')
nh= h_opt(sample32)

h= c(0.15,0.2,0.3)
par(mfrow=c(1,3))
for(x in h){
  plot(density(sample32, kernel='gaussian', bw=x))
```

```
}
h= c(0.15,0.2,0.3)
par(mfrow=c(1,3))
for(x in h){
  plot(density(sample32, kernel='epanechnikov', bw=x))
}
h= c(0.15,0.2,0.3)
par(mfrow=c(1,3))
for(x in h){
  plot(density(sample32, kernel='triangular', bw=x))
}
h= c(0.15,0.2,0.3,0.5,0.6,0.7)
par(mfrow=c(2,3))
for(x in h){
  plot(density(sample32, kernel='rectangular', bw=x))
}
h= c(0.15,0.2,0.3,0.5,0.6,0.7)
par(mfrow=c(2,3))
for(x in h){
  plot(density(sample32, kernel='biweight', bw=x))
}
h= c(0.15,0.18,0.2)
par(mfrow=c(1,3))
for(x in h){
  plot(density(sample32, kernel='cosine', bw=x))
}

par(mfrow=c(2,3))
plot(density(sample32, bw=nh))
plot(density(sample32, bw="nrd"))#normal reference rule-of-thumb
plot(density(sample32, bw="nrd0")) ##rule of thumb with less sentitivity to outliers
plot(density(sample32, bw="ucv")) #unbiased cross validation (minimizes MISES)
plot(density(sample32, bw="bcv")) #biased cross validation
plot(density(sample32, bw="sj"))#sheater jones method

plot(density(sample32,kernel='gaussian', bw="nrd0"), lwd=2, col="darkgreen",
main='Kernel Density Estimator: Gaussian')

par(mfrow=c(2,3))
plot(density(sample32, bw="nrd"), lwd=2, col='green',main="")
plot(density(sample32, bw="nrd0"),lwd=2, col='green', main='Gaussian Kernel')
plot(density(sample32, bw="ucv"),lwd=2, col='green',main="")
plot(density(sample32, kernel='epanechnikov', bw="nrd"),lwd=2,col='blue',main="")
plot(density(sample32, kernel='epanechnikov', bw="nrd0"),lwd=2,col='blue',
main="Epanechnikov Kernel")
plot(density(sample32, kernel='epanechnikov', bw="ucv"),lwd=2,col='blue',main="")

plot(density(sample32, kernel='epanechnikov', bw="nrd"))
plot(density(sample32, kernel='epanechnikov', bw="nrd0"))
plot(density(sample32, kernel='epanechnikov', bw="ucv"))
plot(density(sample32, kernel='epanechnikov', bw="bcv"))
plot(density(sample32, kernel='epanechnikov', bw="sj"))

h_log <- function(x){
  c = pi^5 * (13/ (3^(7/2)*35) )
  return (c*sd(x)*length(x)^(-1/5))
```

```
}
h_dexp <- function(x){
  return (sqrt(2)*sd(x)*length(x)^(-1/5))
}
h_exp <- function(x){
  return (0.5*sd(x)*length(x)^(-1/5))
}

h1=h_log(sample32)
h2=h_dexp(sample32)
h3=h_exp(sample32)

par(mfrow=c(1,3))
plot(density(sample32, bw=h1),lwd=2, col='green',main="") #0.47 over smoothed
plot(density(sample32, bw=h2),lwd=2, col='green',main="") #0.28 over smoothed
plot(density(sample32, bw=h3),lwd=2, col='green',main="") #0.098 under smoothed
```

Plots 3.3

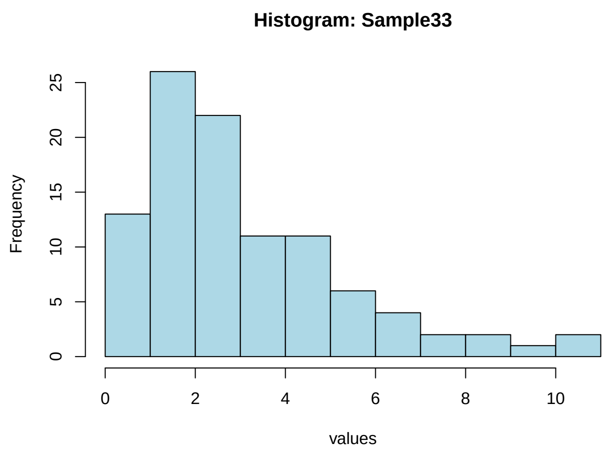

Figure 9: Histogram: Sample

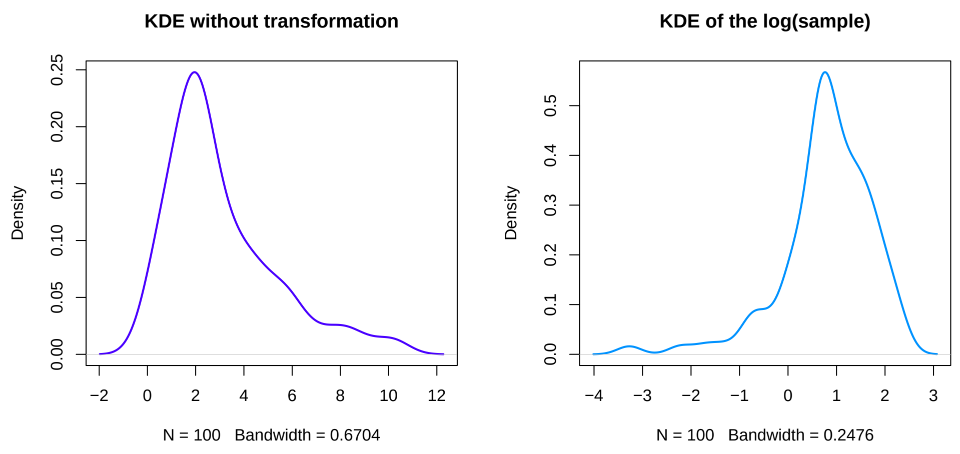

Figure 10: Wrong Estimation of the KDE

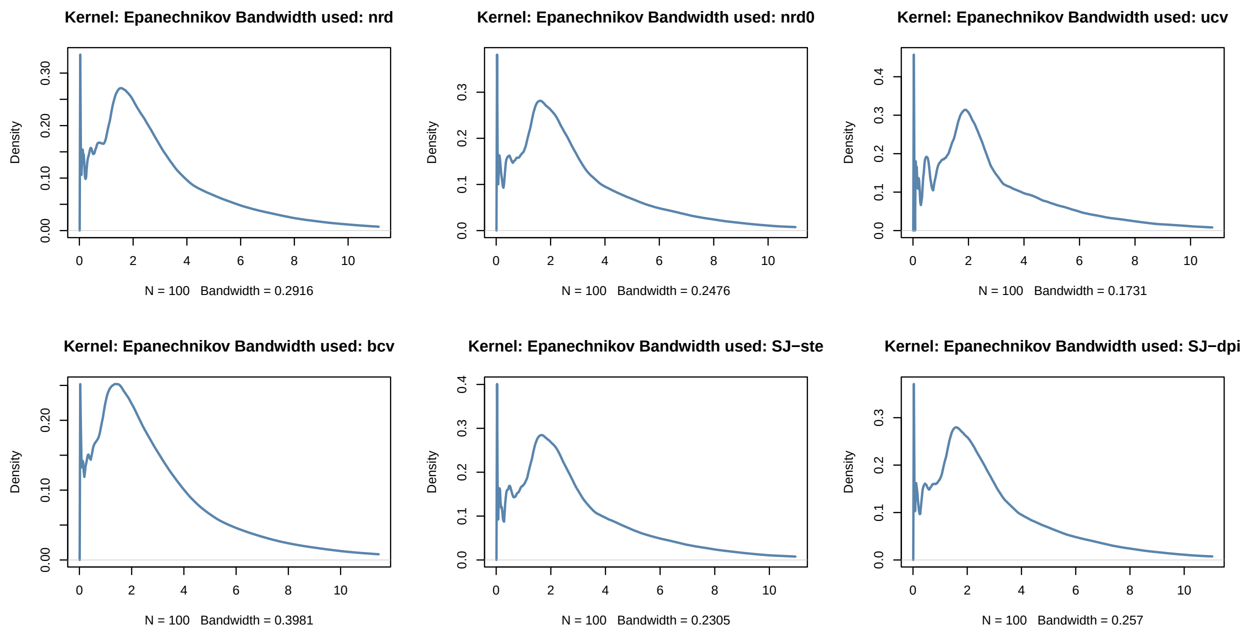

Figure 11: KDE using the Epanechnikov kernel

Code 3.3

```
install.packages("logKDE")
library(logKDE)
sample33 =scan("sample33.txt")

sample33 =scan("sample33.txt")
par(mfrow=c(1,1))
hist(sample33, main="Histogram: Sample33", xlab='values', col='lightblue')

y=log(sample33)
yrange = seq(min(y), max(y), length.out=512)

par(mfrow=c(1,2)) #plot of "wrong" KDE
plot(density(sample33), main="KDE without transformation", lwd=2, col=rainbow(7)[6])
plot(density(y), main='KDE of the log(sample)',lwd=2,col=rainbow(7)[5])

par(mfrow=c(1,1)) #plot of correct KDE
plot(exp(yrange), density(y, bw='nrd', from=min(yrange), to=max(yrange))$y/exp(yrange), type='l
     main='KDE using hint code',lwd=2, col='darkgreen', ylab='Density',
     xlab='N = 100 Bandwidth = 0.2916')

plot(logdensity(sample33), lwd=3,col='darkblue', main="KDE using logdensity")

plot(logdensity(sample33, bw = bw.logG(sample33)), lwd=3,col='pink')
plot(logdensity(sample33, bw = bw.logCV(sample33)), lwd=3,col='purple')

#optimal estimators
par(mfrow=c(1,2))
plot(logdensity(sample33, bw='nrd'), main='Gaussian KDE', lwd=2,
col="#8FB1C7")
plot(logdensity(sample33, bw='SJ-dpi'), main='Gaussian KDE', lwd=2,
col="#000066")

par(mfrow=c(2,3))
```

```
bw_list = list('nrd', 'nrd0', 'ucv','bcv','SJ-ste', 'SJ-dpi')
col_list = list("#FFFFFF","#8FB1C7" ,"#5A85AC" ,"#2E5E92", "#143C78", "#000066")

for (b in bw_list) {
  plot(logdensity(sample33, bw=b), lwd=2,
       main=sprintf("Kernel: Gaussian Bandwidth used: %s",b), col="#5A85AC")
}

kernel_list = list(
#logKDE.g = logdensity_fft(sample33, kernel = 'gaussian'),
logKDE.e = logdensity_fft(sample33, kernel = 'epanechnikov'),
logKDE.t = logdensity_fft(sample33, kernel = 'triangular'),
logKDE.u = logdensity_fft(sample33, kernel = 'uniform'),
logKDE.l = logdensity_fft(sample33, kernel = 'logistic'),
logKDE.lp = logdensity_fft(sample33, kernel = 'laplace'))


par(mfrow=c(2,3)) #plotting every graph separately
plot(logdensity(sample33), lwd=3 ,col='black')
i=1
for (estimate in kernel_list){
  #lines(estimate$x, estimate$y, col=rainbow(7)[i]) #in same plot
  plot(estimate, col=rainbow(7)[i])
  i=i+1
}
```

Plots 3.4

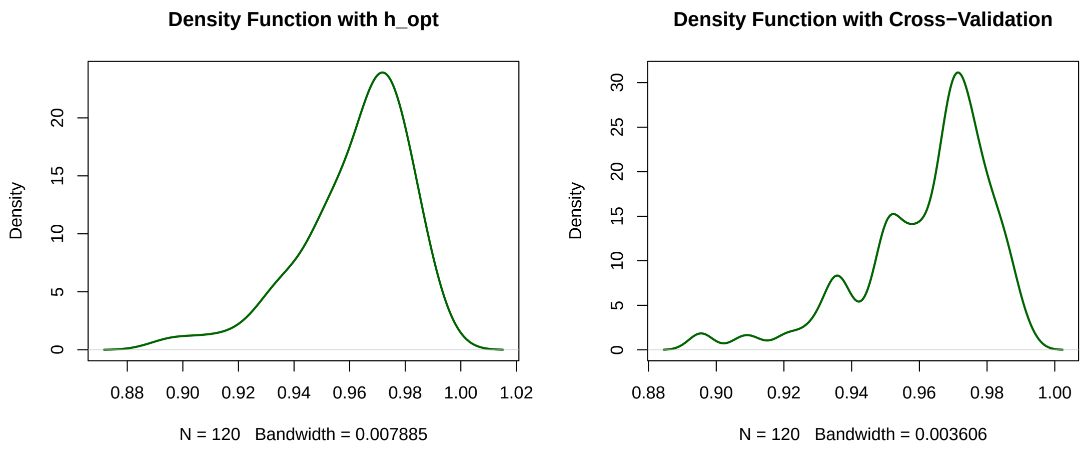

Figure 12: KDE: optimal bandwidth and cross-validation

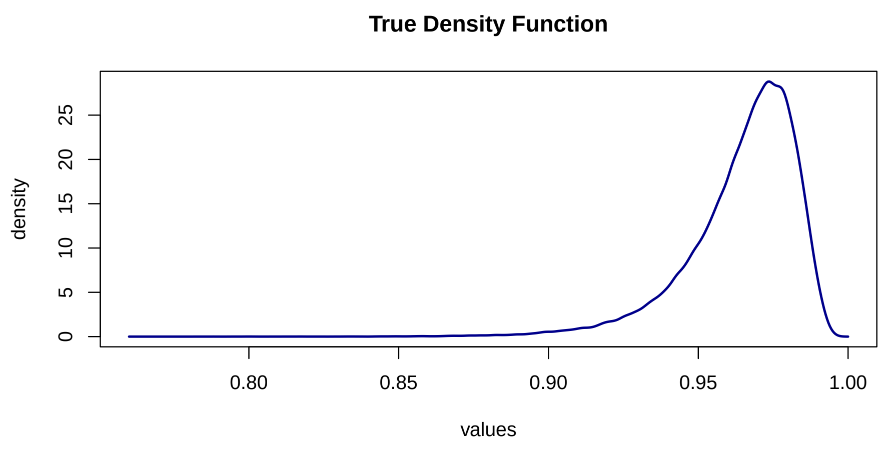

Figure 13: True Density of the Sample

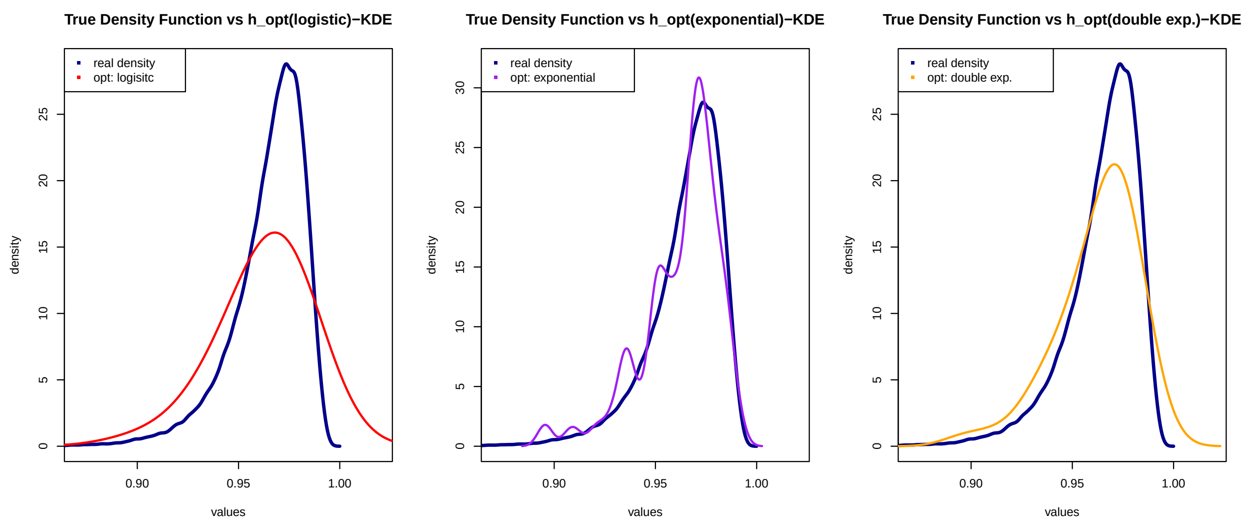

Figure 14: Optimal bandwidth for logistic, exponential and double exponential distributions in contrast with true density

Code 3.4

```
h_opt <- function(x){
  (4*pi)^(-1/10)*(3/8*pi^(-1/2))^(-1/5) * sd(x) * length(x)^(-1/5)
}

sample34 = list34$sample34 #n=120
true.x = list34$true.density.x #n=512 with x-values of true density
true.y = list34$true.density.y #n=512 with y-values

h34 = h_opt(sample34) #0.00788

h_vec = seq(0.0005,0.008,length.out = 100)
cv_crit = sapply(h_vec, CV, sample=sample34, kernel="gaussian")
h_min =h_vec[which(cv_crit == min(cv_crit))] #0.0036

bw.nrd(sample34) #0.0071
bw.bcv(sample34) #0.0073
bw.ucv(sample34) #0.0037

plot(true.x, true.y, type='l', main='True Density Function',lwd=2, col='darkblue',
ylab='density', xlab='values')

par(mfrow=c(1,2))
plot(density(sample34, kernel='gaussian', bw=h34), main='Density Function with
h_opt',lwd=2, col='palegreen')
plot(density(sample34, bw=h_min), main='Density Function with
Cross-Validation',lwd=2, col='darkgreen')

h_log <- function(x){
  c = pi^5 * (13/ (3^(7/2)*35) )
  return (c*sd(x)*length(x)^(-1/5))
}
h_dexp <- function(x){
  return (sqrt(2)*sd(x)*length(x)^(-1/5))
}
h_exp <- function(x){
  return (0.5*sd(x)*length(x)^(-1/5))
}

h_log = h_log(sample34) #0.018
h_dexp = h_dexp(sample34) #0.011
h_exp = h_exp(sample34) #0.0037

cv_density = density(sample34, bw=h_min)
hopt_density = density(sample34, bw =h_opt(sample34))
xlim = c(0.88, 1)

par(mfrow=c(1,2))
plot(cv_density$x, cv_density$y, type='l', main='True Density Function vs. CV-KDE',lwd=2, col='
lines(true.x, true.y, col='darkblue', lwd=3)
legend('topleft', legend=c('real values', 'CV KDE'), col=c('darkblue','red'), pch=c(15,15))

plot(true.x, true.y, type='l', main='True Density Function vs h_opt-KDE',lwd=3, col='darkblue',
lines(hopt_density$x, hopt_density$y, col='red', lwd=2)
legend('topleft', legend=c('real values', 'h_opt KDE'), col=c('darkblue','red'), pch=c(15,15))

par(mfrow=c(1,3))
xlim2 = c(0.87,1.02)
plot(true.x, true.y, type='l', main='True Density Function vs h_opt(logistic)-KDE',
lwd=3, col='darkblue', ylab='density', xlab='values', xlim=xlim2)
lines(density(sample34, bw=h_log), lwd=2, col='red')
legend("topleft", legend=c('real density', 'opt: logisitc'), col=c('darkblue', 'red'),
pch=c(15,15), pt.cex=0.6)

plot(true.x, true.y, type='l', main='True Density Function vs h_opt(exponential)-KDE',
lwd=3, col='darkblue', ylab='density', xlab='values', xlim=xlim2, ylim=c(0,32))
lines(density(sample34, bw=h_exp), lwd=2, col='purple')
legend("topleft", legend=c('real density', 'opt: exponential'), col=c('darkblue',
'purple'), pch=c(15,15), pt.cex=0.6)

plot(true.x, true.y, type='l', main='True Density Function vs h_opt(double exp.)-KDE',
lwd=3, col='darkblue', ylab='density', xlab='values', xlim=xlim2)
lines(density(sample34, bw=h_dexp), lwd=2, col='orange')
legend("topleft", legend=c('real density', 'opt: double exp.'), col=c('darkblue',
'orange'), pch=c(15,15), pt.cex=0.6)
```

Plots 3.5

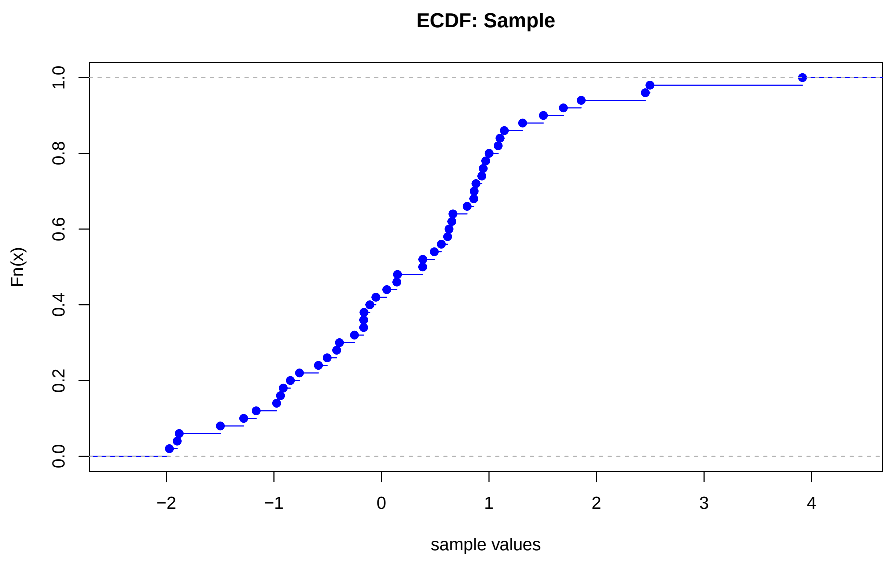

Figure 15: ECDF of the sample data

Code 3.5

```
t.sample= scan('t-sample.txt')

ecdft = ecdf(t.sample)
ecdf1_sample = ecdft(1) #0.780
plot(ecdft)

set.seed(20230303+34)
Tstatistic = function(x){
  return (ecdf(x)(1))
}

ecdf1_empBS = bootstrap(t.sample, Tstatistic, B=2000)
hist(ecdf1_empBS)
avg_emp = mean(ecdf1_empBS) #0.779

set.seed(20230303+34)
s2 = var(t.sample) #1.395
k = 2*s2/(s2-1) #7.064

ecdf1_parBS = replicate(2000,Tstatistic(rt(length(t.sample), df=k)))
hist(ecdf1_parBS)
avg_par = mean(ecdf1_parBS) #0.824

set.seed(20230303+34)
ecdf1_realizations = replicate(2000, Tstatistic(rt(50, df=5)))
avg_realizations = mean(ecdf1_realizations) #0.817

par(mfrow=c(1,3))
hist(ecdf1_empBS,main='Histogram: Empirical Bootstrap', col="lightblue",
xlab="T-statistic")
hist(ecdf1_realizations, main='Histogram: Realizations', col="blue",
xlab="T-statistic")
hist(ecdf1_parBS,main='Histogram: Parametric Boostrap', col="skyblue",
xlab="T-statistic")

sd_empBS = sd(ecdf1_empBS) #0.059
sd_parBS = sd(ecdf1_parBS) #0.054
sd_realizations = sd(ecdf1_realizations) #0.054
```