# Choice of Tuning Parameter in Density Estimation

Shrayan Roy

**Abstract**

This paper introduces some intuitive methods for Binwidth/Bandwidth Selection of Histogram/Kernel Density Estimator.The Intuitions are inspired from Model fitting Techniques, Such as K Fold Cross Validation, AIC, BIC. These methods are then compared with some existing methods based on different Distance Metrics and Kullback Leibler Divergence using Monte-Carlo Simulation. In some cases the proposed methods are working better than some pre-existing methods. Stability of the optimum Binwidth/Bandwidth are also checked.
**Keywords:** Histogram,Density Estimation,AIC,Maximum Likelihood,Tuning Parameter

## 1 Introduction :

Density estimation is vital in statistics as it allows researchers to uncover the underlying distribution of data, enabling them to detect patterns, anomalies, and relationships between variables. By estimating the probability density function (PDF), statisticians gain valuable insights that inform decision-making in fields such as finance, medical research, environmental science, and machine learning. Density estimation techniques, both parametric and non-parametric, play a fundamental role in understanding data distributions and extracting meaningful information from the data.

- **Parametric Density Estimation** :

In this setup, we assume that the data has came from some density $f(.)$. Whose functional form is known, except some of the parameters. We estimate them based on the data and put back in the functional form. For example - We assume that data has came from $Normal(\mu, \sigma^2)$. Where, $\mu$ and $sigma$ are unknown. Thus, based on the data we need to estimate them. If $\hat{\mu}$ and $\hat{\sigma}$ are estimates respectively. Then, estimated density is $\frac{1}{\hat{\sigma}}\phi(\frac{x-\hat{\mu}}{\hat{\sigma}})$. Where, $\phi(.)$ is density of $Normal(0, 1)$ distribution.

Note that, this setup can be extended to PMF's also.

- **Non-Parametric Density Estimation** :

In this setup also, we assume that data has came from some density $f(.)$. But, $f(.)$ is completely unknown. This type of estimation problem is difficult. There are several ways to find an estimate in this case. For example - Histogram, Kernel Density Estimation.

## 2 Choice of Tuning Parameter :

Both Histogram and Kernel Density estimation rely upon some tuning-parameters. Which effects the performance of the density estimator. One of the such hyper parameter is bin-width/band-width ($h$). In case of Histogram, the bin-width represents the width of a bar of histogram and for Kernel Density estimation, it is not that simple. Depending upon choice $h$, the estimator will perform differently. Thus, choice of $h$ is very important. Here, we will suggest some intuitive methods to find optimal $h$ and will compare them with some existing methods. Now, the **Optimal** is itself very *subjective* thing. i.e. in which sense, we want optimality ? We will discuss all these one by one.

# 3 Histogram :

Histogram is the classical visualization tool.It is graphical representation of data points organized into user-specified ranges. Similar in appearance to a bar graph, the histogram condenses a data series into an easily interpreted visual by taking many data points and grouping them into logical ranges or bins.

For defining histogram as a density estimator in a adequate manner, we first fix some points on the real line, the points are of the form $\pm ih$, where $i \in Z^+ \cup \{0\}$. i.e. the points are $..., -(n-1)h, .... -2h, -h, 0, h, 2h, ...., (n-1)h, ....$ If denote the intervals $[(i-1)h, ih)$ where, $i \in Z$ by $B_i$. $h$ is bin-width of the histogram. Then, based on a sample of size n from a population. The histogram as a density estimator is defined as -

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} I(X_i \in B)$$

Where, $B$ is the interval such that $x \in B$. It is clear that $\hat{f}_h(x)$ is **not** an unbiased estimator of $f(x)$. Because the indicators $I(X_i \in B_i)$ are i.i.d random $Bern(p)$ random variables, with $p = P_r(X_1 \in B_i)$. Thus, $E_X[\hat{f}_h(x)] = \frac{p}{h}$. Similarly, $Var_X(\hat{f}_h(x)) = \frac{p(1-p)}{nh^2}$. Now, with the fixed points, the performance of the histogram as a density estimator is fully determined by the control parameter $h$. We are in search for an optimal $h$. The optimality condition can be to minimize some **Loss Function**. Using squared loss function, optimal $h$ has already been obtained and there are some existing methods as well. We will consider different loss function.

We have written R code for the histogram, which essentially gives point wise estimate of density.

```r
#fixed mess are -+ih; i = 0,1,2,....

Zero_Origin_Histogram <- function(x, data, h) {
  n <- length(data)
  i <- ceiling(x/h)
  estimated_desnity_at_x <- mean(data < i * h & data >= (i-1) * h) / h
  return(estimated_desnity_at_x)
}
```

`x` is the point at which we want to estimate the density and `h` is chosen binwidth.

## 3.1 Maximum Likelihood Based Approach for finding Optimal $h$ :

The intuition behind this method comes from Regression. We can think in the same line as we do in regression. In regression, we have data points $(x_i, y_i)_{i=1(1)n}$ and based on the data points we want to choose $\hat{f}$, which can well approximate $f$, i.e the true relation between $Y$ and $X$. So, based on the data points, using some method we obtain an estimate $\hat{f}$. We can compute $\hat{f}(x_1), \hat{f}(x_2), ...\hat{f}(x_n)$. If these are approximately equal to $y_1, y_2, ..., y_n$, then the training MSE i.e., $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$ is small. However that does not imply good fit. Thus, we want to know whether, $\hat{f}(x_0)$ is approximately equal to $y_0$, where $(x_0, y_0)$ is a previously unseen test observation not used to train the model. If this is small, then the selected model is good. But, unfortunately we do not know value of $y_0$. Thus, we make use of **Cross-Validation** to estimate $E[y_0 - \hat{f}(x_0)]^2$.

Let us try to apply this idea for choosing optimal $h$ in histogram. In our case we have only one parameter in model i.e., $h$. But there is a intrinsic difference. We don't have data on $f(x_i)$'s.Thus we will tackle the problem in a different way.

Suppose, we have observations $X_1, X_2, ...X_n$ from some unknown continuous population $f(.)$. We first divide the data into k different parts.Then, we will consider one such part as test data and the remaining data

training data. Suppose $X_{11}, X_{12}, ...., X_{1n_1}$ denote training data and $X_{21}, X_{22}...., X_{2n_2}$ denote test data. Then, based on the training data our histogram is

$$\hat{f}_h(x) = \frac{1}{n_1 h} \sum_{i=1}^{n_1} I(X_{1i} \in B)$$

Now, if we evaluate the estimated density based on the test data and calculate the estimated log likelihood. Then, it will be -

$$\hat{L}(h) = -n_2 log(n_1 h) + \sum_{j=1}^{n_2} log(\sum_{i=1}^{n_1} I(X_{1i} \in B_j))$$

Where, $B_j$ is the interval such that $X_{2j} \in B_j$. Then, we repeat this for other $k-1$ parts. In this way, we will get $\hat{L_1}(h), \hat{L_2}(h), ...., \hat{L_k}(h)$.Then, we will find $\sum_{m=1}^{k} \hat{L_m}(h)$. Then, our optimal $h$ would be -

$$h_{ML} = \underset{h>0}{argmax} \frac{1}{n} \sum_{m=1}^{k} \hat{L_m}(h)$$

Notice that, for some values of $X_{2j}$ we will have $\sum_{i=1}^{n_1} I(X_{1i} \in B_j) = 0$. Which will make $L(\hat{h})$ make $-\infty$. So, to avoid that problem we will replace those zero's with smallest nonzero value of the estimated density for test data. This adjustment is quite intuitive. Now, $\frac{1}{n} \sum_{m=1}^{k} \hat{L_m}(h)$ as a function of $h$ finite. But, still we will have some other problems to find the point of maximum, which we will see later.

Here, we have written an implementation of the above discussed method in R.

```
Optimal.histogram_ML <- function(data,k = 2,mplot = TRUE,...)
{
  assertthat::assert_that(k > 1)

  split.data <- split(1:length(data),sample(1:length(data),length(data),replace = F)%%k)
  r <- extendrange(range(data))

  Estimated.histogram.Likelihood <- function(h) {

    lik.vals <- lapply(split.data,FUN = function(test.data.index){

      training.data <- data[-test.data.index]
      test.data <- data[test.data.index]
      trbreaks <- h * seq(floor(r[1] / h), ceiling(r[2] / h))
      trhist <- hist(training.data, breaks = trbreaks, plot = FALSE)
      trhist$density[trhist$density == 0] <- min(trhist$density[trhist$density != 0])
      l <- trhist$density[ findInterval(test.data, vec = trbreaks, all.inside = TRUE) ]
      sum(log(l))

    })

    sum(unlist(lik.vals))/length(data)
  }

  n <- mean(unlist(lapply(split.data,FUN = length)))
  hhrange <- diff(r) / n^c(0.9, 0.4)
  hh <- seq(hhrange[1], hhrange[2], length.out = 1500)

  est.loglikelihood <-
    vapply(hh, FUN = Estimated.histogram.Likelihood,
```

```
            FUN.VALUE = 2)

  if(mplot){
    plot(hh, est.loglikelihood,type = 'l',col = 'red',xlab = 'h',
           ylab = 'Mean Estimated Log Likelihood',main = paste('k = ',k))

  }

  h.optimal <- hh[which.max(est.loglikelihood)]

  if(mplot){
    hist(data, breaks = h.optimal * seq(floor(r[1] / h.optimal), ceiling(r[2] / h.optimal)),
         col = 'yellow',main = paste('Histogram with Optimal h = ',round(h.optimal,3)),
         border = "red",freq = FALSE, ...)
    mtext("Using ML Approach",side = 3)
  }

  return(h.optimal)

}
```

Some important points related to above function -

- The function returns optimal value of $h$ based on the data and *by default* plots the histogram based on full data and the mean estimated log likelihood as a function of $h$. if you specify `mplot = FALSE`, then it will not give the above mentioned plots. Another important argument is `k`, *by default* `k = 2`. It determines the number of folds of cross-validation.
- If you notice carefully, we have not considered all possible values of $h$. Instead, we have made the values of $h$ dependent on **range of data** and **sample size**. (`hh` in the above code). We have divided the data into 1:1 ratio. Using this method, we have illustrated some examples below. We have plotted the true pdf also.

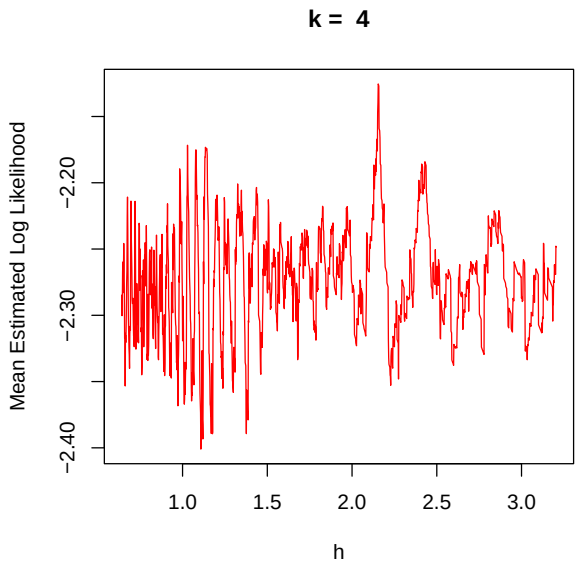
**Data is from Normal Distribution :**

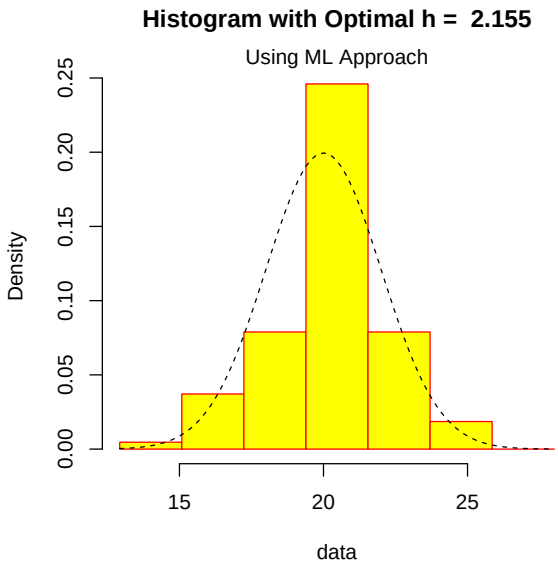
We have considered sample sizes ($n$) - 100,500,1000,10000

```
par(mfrow = c(1,2))

data <- rnorm(100,20,2)
noquote(paste("With n = 100, h = ",Optimal.histogram_ML(data,k = 4)))
```

```
## [1] With n = 100, h =  2.15453290063653
```

```
curve(dnorm(x,20,2),add = T,lty = 2)
```

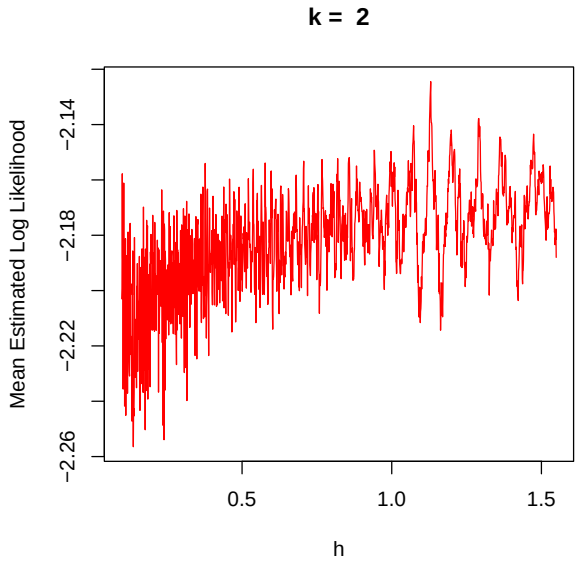


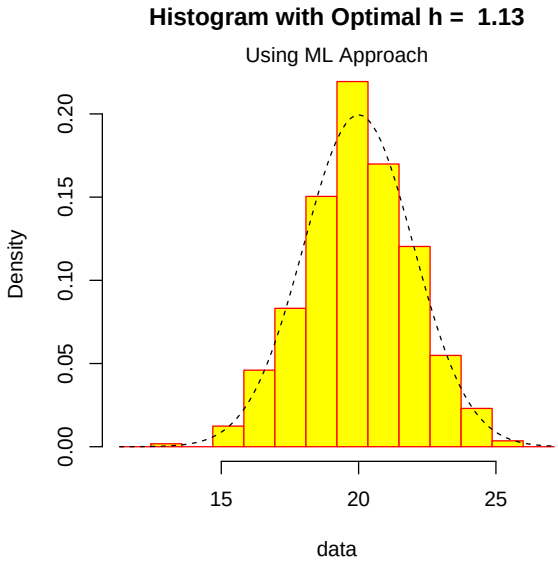




```
data <- rnorm(500,20,2)
noquote(paste("With n = 500, h = ",Optimal.histogram_ML(data,k = 2)))
```

```
## [1] With n = 500, h =  1.13013371168626
```

```
curve(dnorm(x,20,2),add = T,lty = 2)
```

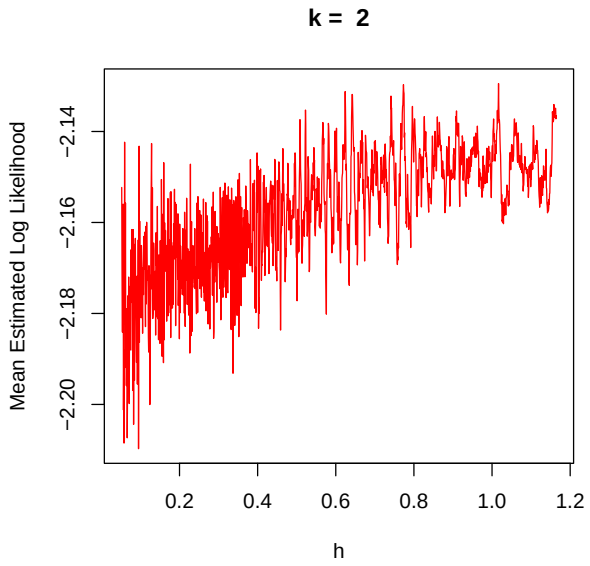


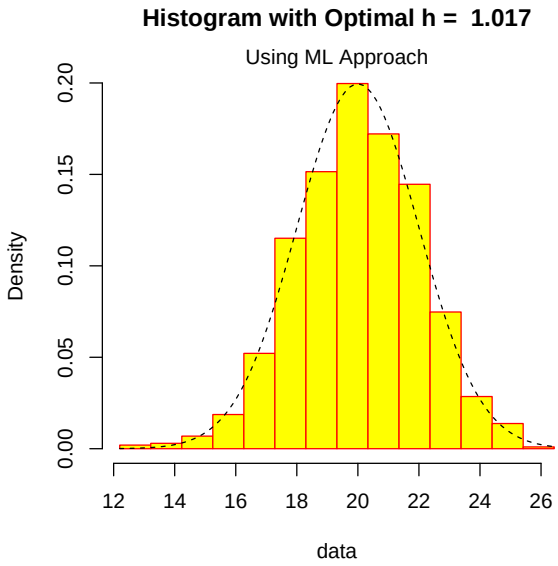




```
data <- rnorm(1000,20,2)
noquote(paste("With n = 1000, h = ",Optimal.histogram_ML(data,k = 2)))
```

```
## [1] With n = 1000, h =  1.01659943437213
```

```
curve(dnorm(x,20,2),add = T,lty = 2)
```

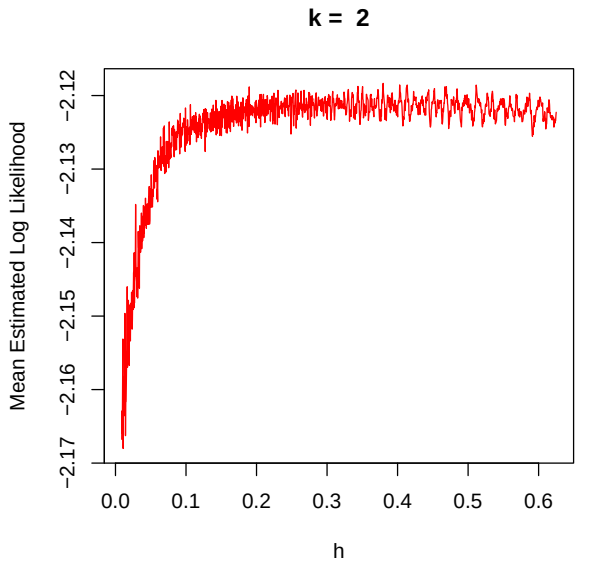


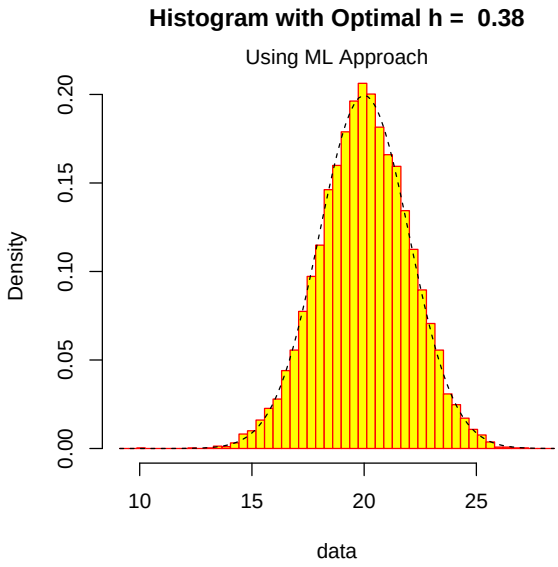




```
data <- rnorm(10000,20,2)
noquote(paste("With n = 10000, h = ",Optimal.histogram_ML(data,k = 2)))
```

```
## [1] With n = 10000, h =  0.379577075256714
```

```
curve(dnorm(x,20,2),add = T,lty = 2)
```




From the above plots, we make some important observations.

- As we **increase** $n$, the performance of the histogram as a density estimator becomes **better**.

- The estimated mean log likelihood as a function of $h$ is **not very stable**. i.e. it is not smooth. So, it is **difficult to find the point maximum** value of such a function. the reasons are the following -

  (i) To find the point maximum, we are just finding value of the function at some data dependent values of $h$.
  (ii) Since, the value of the function changes very rapidly within very small neighbourhood, no optimization technique can give global maximum point.

Thus, the optimum value of $h$ that we are getting may not be the actual optimum value, that we are interested to find.

- As, sample size $n$ increases, this behavior of the function changes and it's variance decreases.

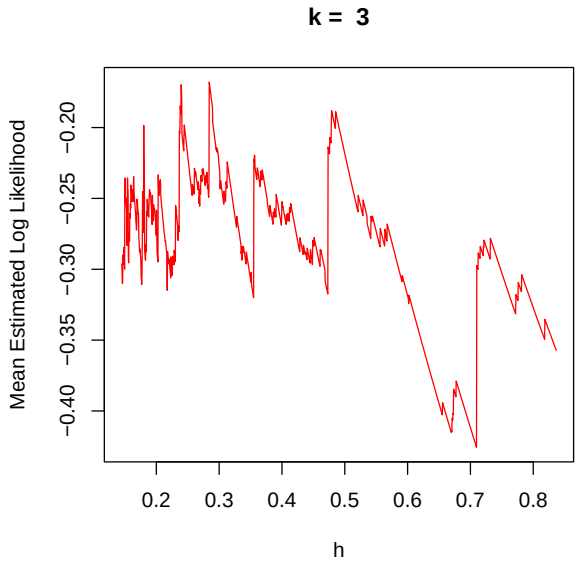
So, We have seen what happens if the underlying population is Normal Distribution. Let's move to other distributions.

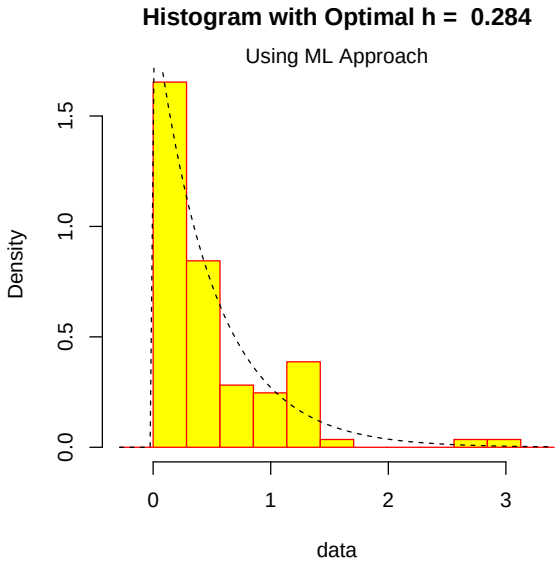
**Data is from Exponential Distribution :**

```
par(mfrow = c(1,2))

data <- rexp(100,2)
noquote(paste("With n = 100, h = ",Optimal.histogram_ML(data, k = 3)))
```

```
## [1] With n = 100, h =  0.284251823182283
```

```
curve(dexp(x,2),add = T,lty = 2)
```

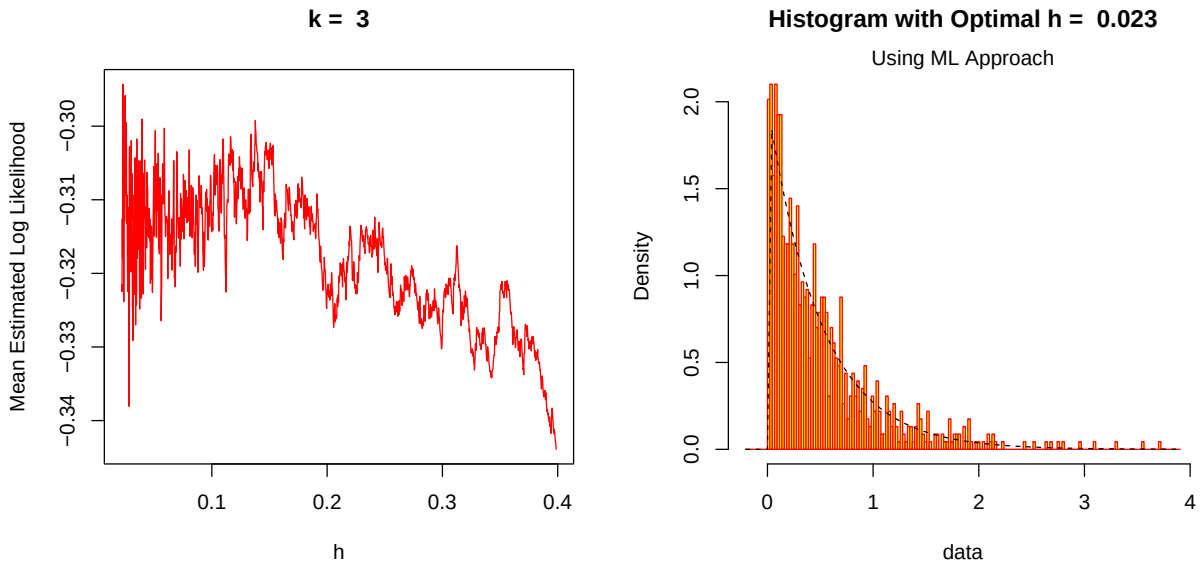



```
data <- rexp(1000,2)
noquote(paste("With n = 1000, h = ",Optimal.histogram_ML(data,k = 3)))
```

```
## [1] With n = 1000, h =  0.0228428591533196
```

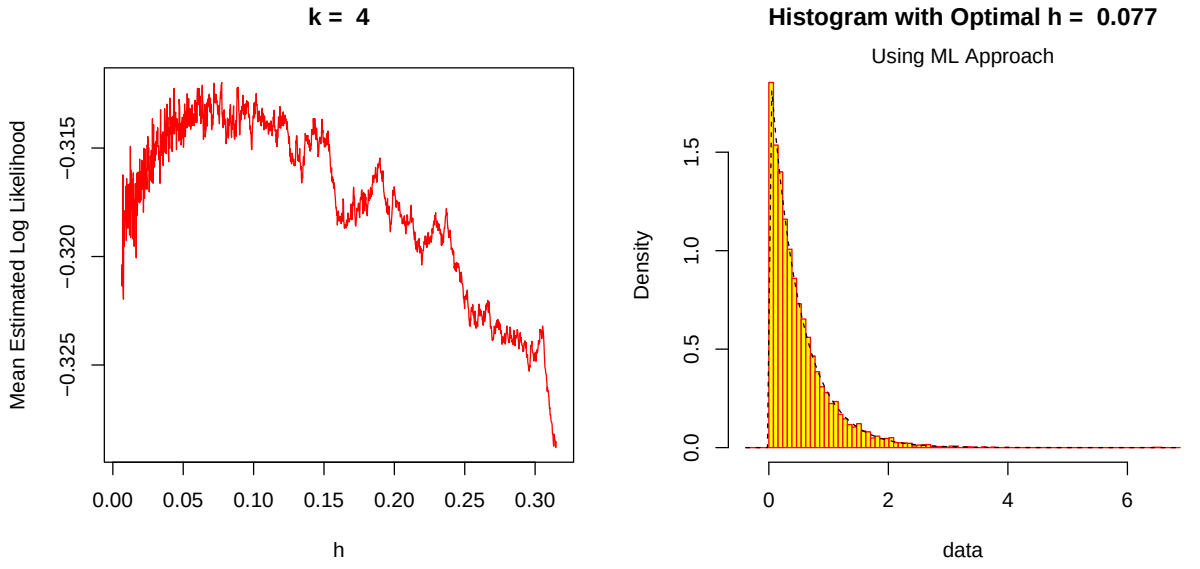
```
curve(dexp(x,2),add = T,lty = 2)
```




```
data <- rexp(10000,2)
noquote(paste("With n = 10000, h = ",Optimal.histogram_ML(data,k = 4)))
```

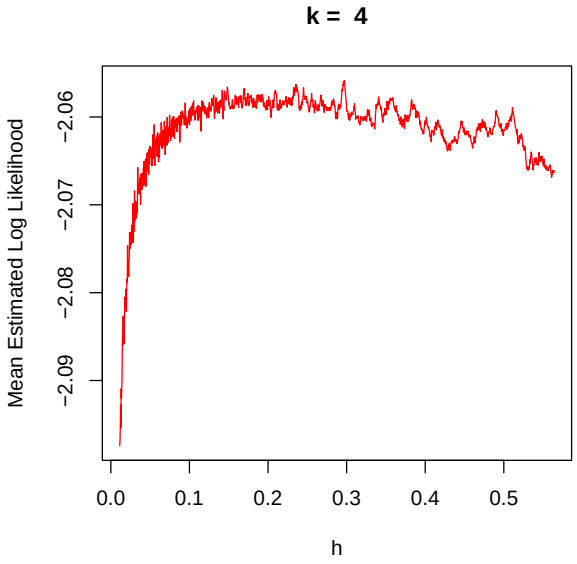
```
## [1] With n = 10000, h =  0.0773434683069144
```

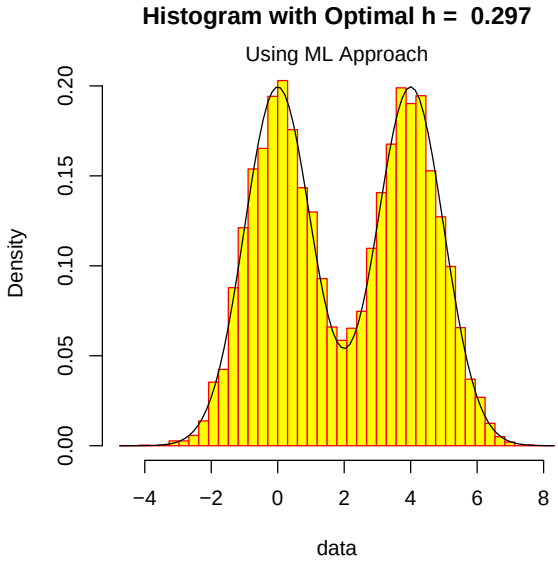
```
curve(dexp(x,2),add = T,lty = 2)
```



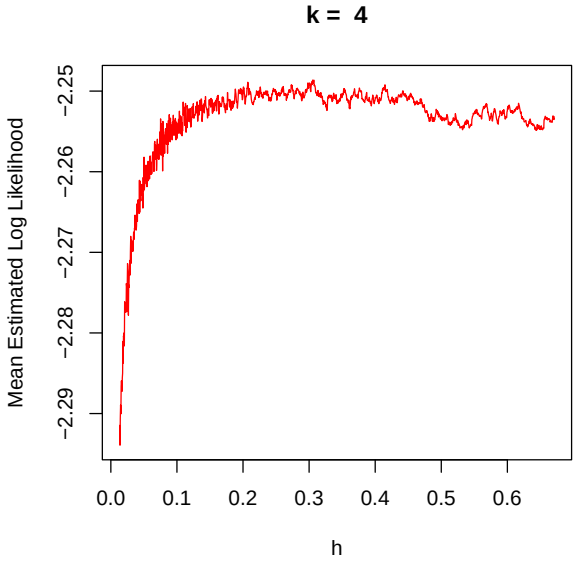

Here also we are getting similar kind of observations. Now, the problem of estimating density becomes **interesting** when true density is **not uni-modal**. We will consider some examples here. In each case we have $n = 10000$.

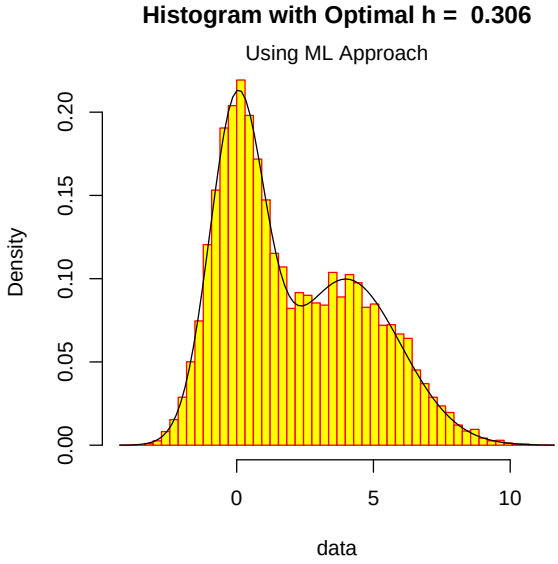
```
## [1] Optimal h =  0.297103470244056
```



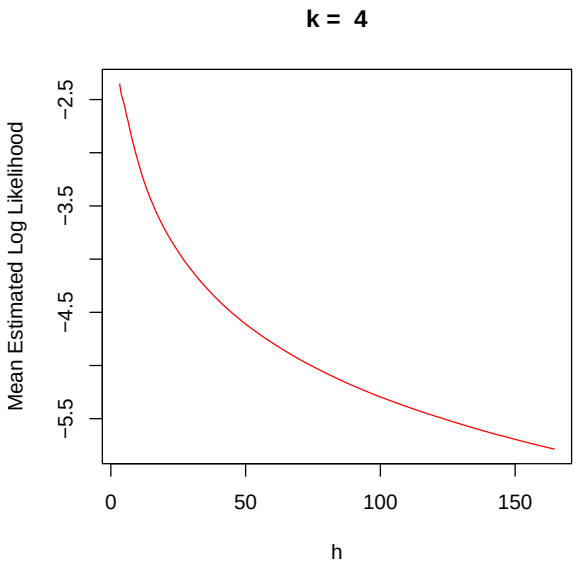



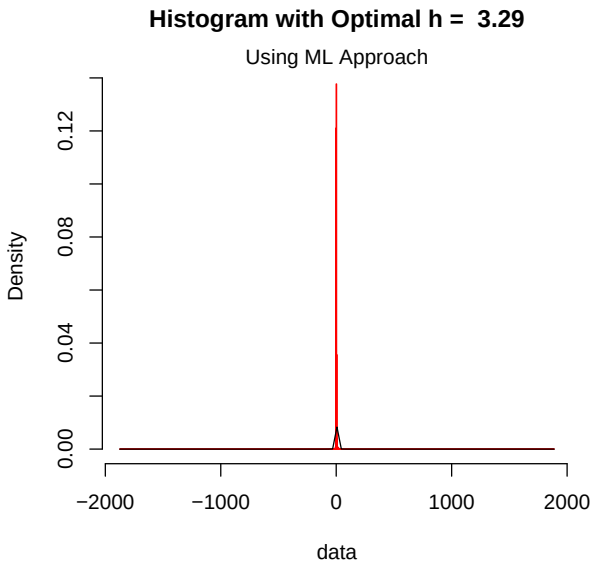

```
## [1] Optimal h =  0.305528245443658
```



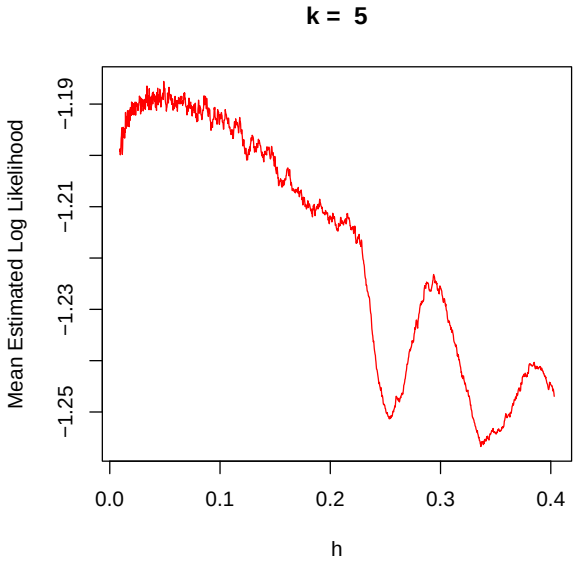



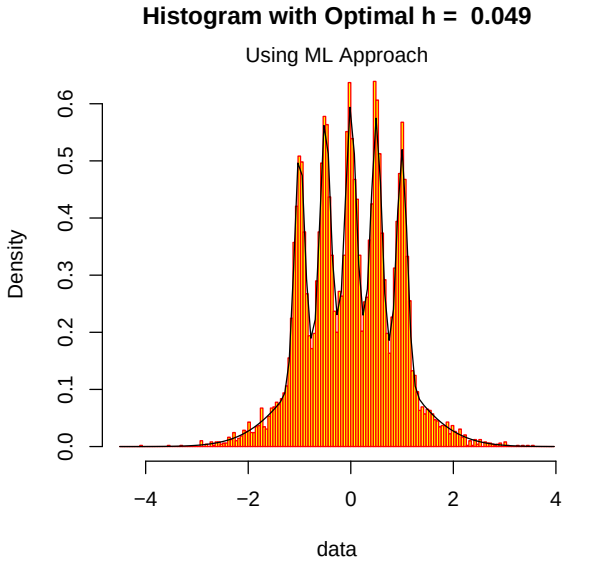

```
## [1] Optimal h =  3.28987906878708
```




```
## [1] Optimal h =  0.0489746710692917
```




So, in all cases it is performing well **visually**. The last distribution is **Claw Distribution**. you can check `??dclaw` in **R**.

## 3.2 AIC Based Bin-Width Selection Approach :

Suppose, we have a statistical model of some data. Let $k$ be the number of estimated parameters in the model. Let $\hat{L}$ be the maximized value of the likelihood function for the model. Then the AIC value of the model is given by -

$$AIC = 2k - 2ln(\hat{L})$$

Given a set of candidate models for the data, the preferred model is the one with the minimum AIC value. Thus, AIC rewards goodness of fit (as assessed by the likelihood function), but it also includes a penalty that is an increasing function of the number of estimated parameters.The penalty discourages over fitting, which is desired because increasing the number of parameters in the model almost always improves the goodness of the fit. We will try to use this concept for finding optimal $h$.

In the expression of $L(\hat{h})$, both $n_2 log(n_1 h)$ and $\sum_{i=1}^{n_1} I(X_{1i} \in B_j)$ also depends on $h$. As, $h \to 0$ both of them $\to -\infty$. But, $\sum_{i=1}^{n_1} I(X_{1i} \in B_j)$ becomes $-\infty$ more quickly. Thus, as a whole $L(\hat{h})$ will be $-\infty$. On the other hand, if we calculate the mean estimated log likelihood based on training data itself, then it will $\to \infty$ as $h \to 0$. So, if we find optimum $h$ by considering whole data as training data, we are likely to get a small value of $h$ as optimum value. So, the histogram will under-smooth the true density. Ideally, we should put some penalty. For a given data, small values of $h$ implies large number of bins. So, we can choose number of bins as choice of penalty.

If $L(\hat{h})$ is the likelihood obtained using histogram as a density estimator based on whole data. We will choose that $h$ for which $AIC = 2k - 2L(\hat{h})$ is minimum ($k$ is the number of bins for histogram based on whole data).

This is the R implementation of the above method. Notice that here, the same adjustments are made with zero density situations as we have done in case of Maximum Likelihood method.

```
Optimal.histogram_AIC <- function(data,mplot = TRUE,...){

  r <- extendrange(range(data))
  n <- length(data)

  h.AIC <- function(h) {

    tbreaks <- h * seq(floor(r[1] / h), ceiling(r[2] / h))
    thist <- hist(data, breaks = tbreaks, plot = FALSE)
    thist$density[thist$density == 0] <- min(thist$density[thist$density != 0])
    l <- thist$density[ findInterval(data, vec = tbreaks, all.inside = TRUE) ]

    return(2*length(thist$breaks) - 2*sum(log(l)))
  }

  hhrange <- diff(r) / n^c(0.9, 0.4)
  hh <- seq(hhrange[1], hhrange[2], length.out = 1500)

  AICVal <-
    vapply(hh, FUN = h.AIC,FUN.VALUE = 2)

  if(mplot){
    plot(hh,AICVal,type = 'l',col = 'red',xlab = 'h',
         ylab = 'AIC')

  }

  h.optimal <- hh[which.min(AICVal)]

  if(mplot){
    hist(data, breaks = h.optimal * seq(floor(r[1] / h.optimal), ceiling(r[2] / h.optimal)),
         col = 'cyan',main = paste('Histogram with Optimal h = ',round(h.optimal,3)),
         border = "red",freq = FALSE, ...)
    mtext("Using AIC",side = 3)
  }
```

```
  return(h.optimal)
}
```

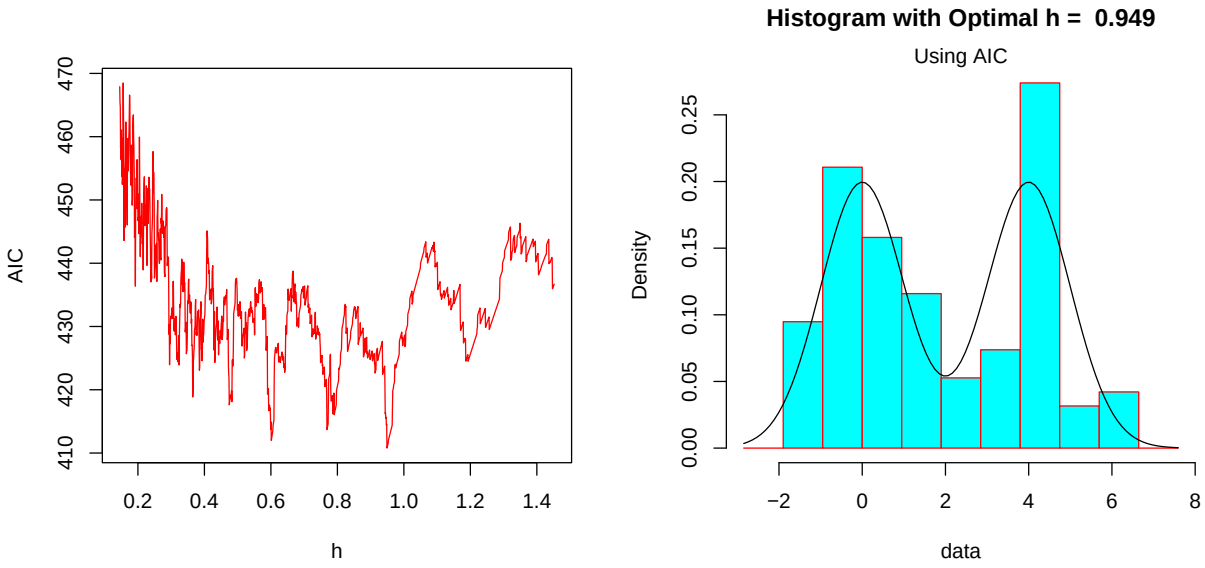
`Optimal.histogram_AIC` has same arguments as `Optimal.histogram_ML`, except `k`. It searches for optimal $h$ in data dependent values of $h$.

Here, We have used this method to find optimal binwidth for some examples.

```
par(mfrow = c(1,2))
pdf.d <- function(x){
  dnorm(x,0,1)*0.5 + dnorm(x,4,1)*0.5
} #Mixture of N(0,1) and N(4,1)

d <- rbinom(100,1,0.5)
data <- rnorm(100,0,1)*d + rnorm(100,4,1)*(1-d)
Optimal.histogram_AIC(data)
```

```
## [1] 0.9491225
```

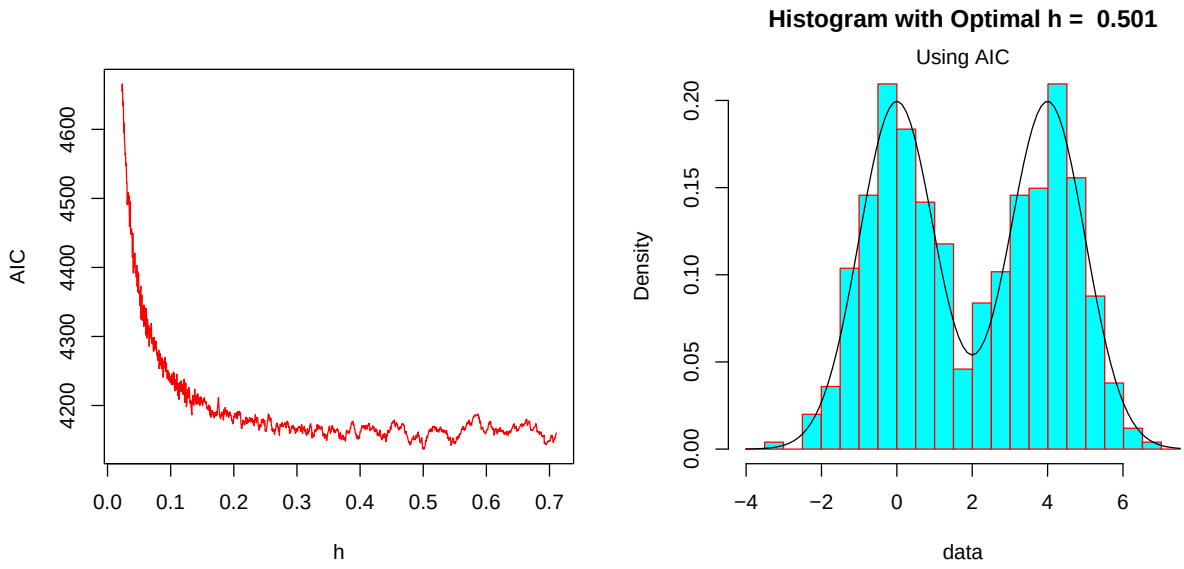
```
curve(pdf.d(x),add = T)
```




```
par(mfrow = c(1,2))
d <- rbinom(1000,1,0.5)
data <- rnorm(1000,0,1)*d + rnorm(1000,4,1)*(1-d)
Optimal.histogram_AIC(data)
```

```
## [1] 0.5012681
```

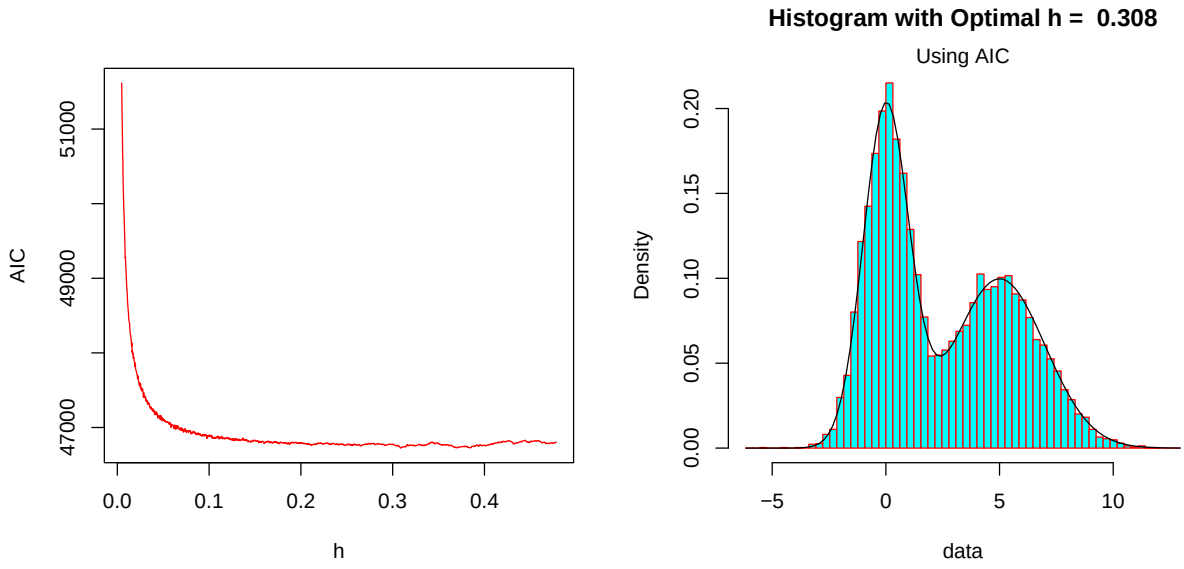
```
curve(pdf.d(x),add = T)
```




```
par(mfrow = c(1,2))
pdf.d <- function(x){
  dnorm(x,0,1)*0.5 + dnorm(x,5,2)*0.5
} #Mixture of N(0,1) and N(5,2)

d <- rbinom(10000,1,0.5)
data <- rnorm(10000,0,1)*d + rnorm(10000,5,2)*(1-d)
Optimal.histogram_AIC(data)
```

```
## [1] 0.3082725
```

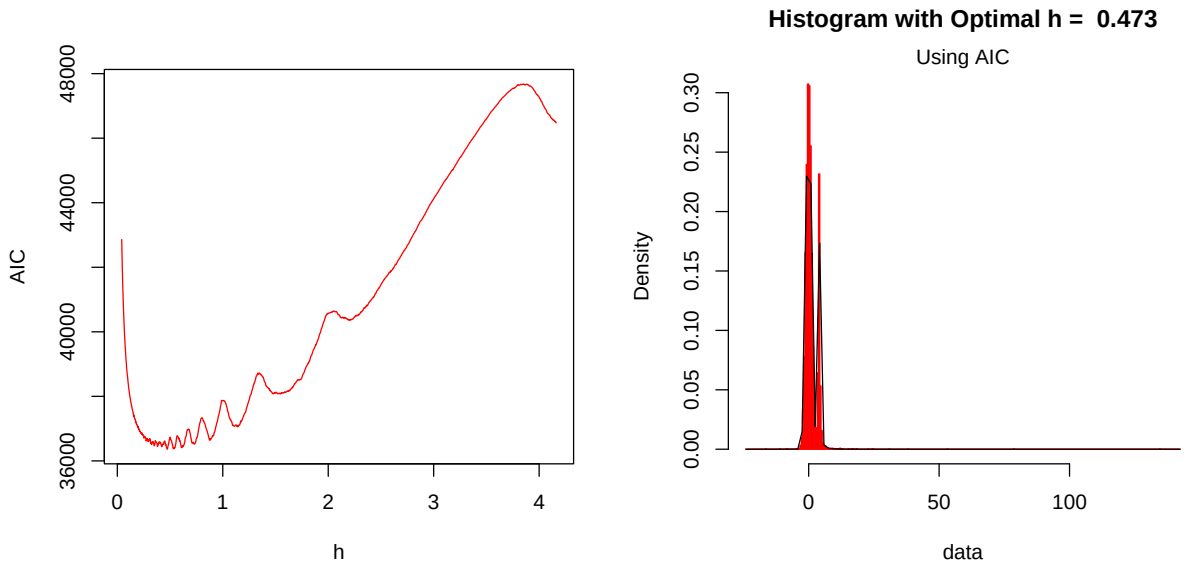
```
curve(pdf.d(x),add = T)
```



```
par(mfrow = c(1,2))
pdf.d <- function(x){
  dnorm(x,0,1)*0.8 + dcauchy(x,4,0.2)*0.2
}#Mixture of N(0,1) and Cauchy(4,0.2)

d <- rbinom(10000,1,0.8)
data <- rnorm(10000,0,1)*d + rcauchy(10000,4,0.2)*(1-d)
Optimal.histogram_AIC(data)
```

```
## [1] 0.4731738
```

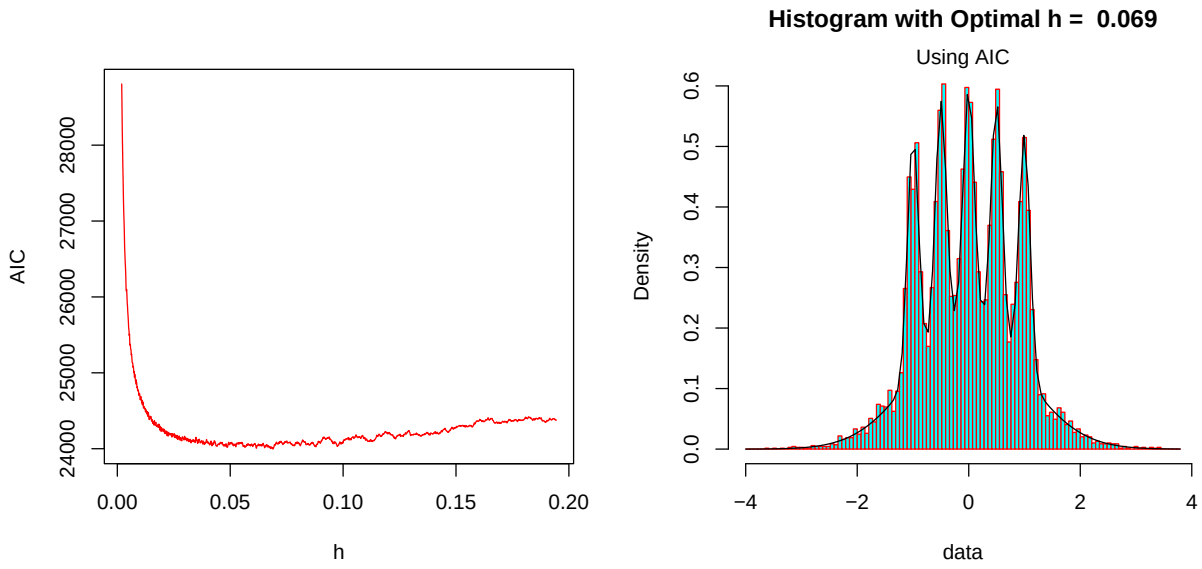
```
curve(pdf.d(x),add = T)
```




```
par(mfrow = c(1,2))
data <- rclaw(10000)  #Claw Distribution
Optimal.histogram_AIC(data)
```

```
## [1] 0.06895763
```

```
curve(dclaw(x),add = T)
```




Instead of generating data from mixture normal distribution in this way, you can use `nor1mix` for data generation.

## 3.3 Other Methods to Obtain Optimal $h$ for Histogram :

There are methods to to find optimal $h$. Sometimes, they give optimal number of bins also. We will discuss three such methods.

### 3.3.1 Struges Choice :

In this method, number of bins is obtained as -

$$k = \lceil log_2(n) + 1 \rceil$$

Where, $n$ is the number of observations.

### 3.3.2 Scott's Choice :

This method uses optimal $h$ as -

$$h = \frac{3.49\hat{\sigma}}{n^{\frac{1}{3}}}$$

Where,$\hat{\sigma}$ is the sample standard deviation. This is optimal for random samples of normally distributed data, in the sense that it minimizes the integrated mean squared error of the density estimate.

### 3.3.3 Freedman–Diaconis' Choice :

This method is based on interquartile range. Here, the optimal $h$ is given by -

$$h = \frac{2IQR(data)}{n^{\frac{1}{3}}}$$

It replaces $3.49\hat{\sigma}$ of Scott's rule with $2IQR(data)$, which is less sensitive than the standard deviation to outliers in data.

All these methods have implementation in **R**. You can check `?nclass.FD` , `?nclass.scott` and `?nclass.Sturges`. But, they returns number of bins.

# 4 Origin Of Histogram :

So far we have discussed about histogram in a setup, where the origin is zero. i.e, we have considered intervals of the form $[(i-1)h, ih)$ for histogram. But, we can also choose intervals of the form $[t_0 + (i-1)h, t_0 + ih)$. If we consider intervals of these forms. These the origin of the histogram is $t_0$. For appropriately chosen origin, we will get good estimate of the density function $f(x)$. Let's define histogram in these general setup. The points are of the form $t_0 \pm ih$, where $i \in Z^+ \cup \{0\}$. i.e. the points are $..., t_0 - (n-1)h, ....t_0 - 2h, t_0 - h, t_0, t_0 + h, t_0 + 2h, ...., t_0 + (n-1)h, ....$ If denote the intervals $[t_0 + (i-1)h, t_0 + ih)$ where, $i \in Z$ by $C_i$. $h$ is bin-width of the histogram. Then, based on a sample of size n from a population. The histogram as a density estimator is defined as -

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} I(X_i \in C_i)$$

The below is a **R** implementation of Origin Dependent Histogram. This will give us point estimate.

```
#fixed mesh are t0 -+ih; i = 0,1,2,....

Origin_Histogram <- function(x,data,t0,h){
  n <- length(data)
  i <- ceiling((x - t0)/h)
  estimated_desnity_at_x <- mean(data < t0 + i*h & data >= t0 + (i-1)*h)/h
  return(estimated_desnity_at_x)
}
```

`x` is the point at which we want to estimate the density, `t0` is the chosen origin and `h` is chosen binwidth.

Choice of Origin becomes important, when you know that your random variable under consideration takes value greater than $c$. Then, we can choose our origin as $c$. We have illustrate the importance of choice of origin in such a situation.

**Example** - Suppose, we have drawn a random sample of size 500 from Pareto distribution with location 1.2 and Shape 10. Then, we can get the estimate of density using histogram at different points. For example we want to estimate at x = 1.1 with h = 1

```
## True density value      With origin 0    With origin 1.2
##              0.000              0.992              0.000
```

So, choice of origin is very important.

The below are implementation of Maximum likelihood method and AIC method in **R** with choice of origin to user.

```
Optimal.Origin_histogram_ML <- function(data,t0,k = 2,...){

  assertthat::assert_that(k > 1)
```

```
  split.data <- split(1:length(data),sample(1:length(data),length(data),replace = F)%%k)
  r <- extendrange(range(data)) - t0

  Estimated.histogram.Likelihood <- function(h) {

    lik.vals <- lapply(split.data,FUN = function(test.data.index){

      training.data <- data[-test.data.index]
      test.data <- data[test.data.index]
      trbreaks <-  h * seq(floor(r[1] / h), ceiling(r[2] / h)) + t0
      trhist <- hist(training.data, breaks = trbreaks, plot = FALSE)
      trhist$density[trhist$density == 0] <- min(trhist$density[trhist$density != 0])
      l <- trhist$density[ findInterval(test.data, vec = trbreaks, all.inside = TRUE) ]
      sum(log(l))

    })

    sum(unlist(lik.vals))/length(data)
  }

  n <- mean(unlist(lapply(split.data,FUN = length)))
  hhrange <- diff(r) / n^c(0.9, 0.4)
  hh <- seq(hhrange[1], hhrange[2], length.out = 1500)

  est.loglikelihood <-
    vapply(hh, FUN = Estimated.histogram.Likelihood,
             FUN.VALUE = 2)


  h.optimal <- hh[which.max(est.loglikelihood)]

  if(mplot){
    plot(hh, est.loglikelihood,type = 'l',col = 'red',xlab = 'h',
           ylab = 'Mean Estimated Log Likelihood',main = paste('k = ',k))

  }

  h.optimal <- hh[which.max(est.loglikelihood)]

 if(mplot){
  hist(data, breaks = h.optimal * seq(floor(r[1] / h.optimal), ceiling(r[2] / h.optimal)) + t0,
    col = 'yellow',main = paste('Histogram with Optimal h = ',round(h.optimal,3)),border = "red",
    freq = FALSE, ...)
  mtext(paste("Using ML Approach and Origin",t0),side = 3)
 }

  return(h.optimal)
}
```

Except the origin `t_0`, `Optimal.Origin_histogram_ML` takes as same arguments as `Optimal.histogram_ML` and gives the same output.

Similarly, we can do for AIC based method.

```r
Optimal.Origin_histogram_AIC <- function(data,t0,...){

  r <- extendrange(range(data)) - t0
  n <- length(data)

  h.AIC <- function(h) {

    tbreaks <- h * seq(floor(r[1] / h), ceiling(r[2] / h)) + t0
    thist <- hist(data, breaks = tbreaks, plot = FALSE)
    thist$density[thist$density == 0] <- min(thist$density[thist$density != 0])
    l <- thist$density[ findInterval(data, vec = tbreaks, all.inside = TRUE) ]

    return(2*length(thist$breaks) - 2*sum(log(l)))
  }

  hhrange <- diff(r) / n^c(0.9, 0.4)
  hh <- seq(hhrange[1], hhrange[2], length.out = 1500)

  AICVal <-
    vapply(hh, FUN = h.AIC,FUN.VALUE = 2)

  if(mplot){
    plot(hh,AICVal,type = 'l',col = 'red',xlab = 'h',
         ylab = 'AIC')

  }

  h.optimal <- hh[which.min(AICVal)]

 if(mplot){
  hist(data, breaks = h.optimal * seq(floor(r[1] / h.optimal), ceiling(r[2] / h.optimal)) + t0,
  col = 'cyan',main = paste('Histogram with Optimal h = ',round(h.optimal,3)),
  border = "red",freq = FALSE, ...)
  mtext("Using AIC",side = 3)
 }

  return(h.optimal)
}
```

Except the origin `t_0`, `Optimal.Origin_histogram_AIC` takes same arguments as `Optimal.histogram_AIC` and gives the same output.

# 5 Average Shifted Histogram :

Depending upon what origin we choose, the performance of histogram differs. Thus, it would be better if the histogram is origin independent. Such a thing can be achieved by average shifted histogram. Averaged shifted histogram (ASH) is freed from the dependency of the origin and seems to correspond to a smaller bin-width than the histograms from which it was constructed. Even though the ASH can in some sense be viewed as having a smaller bin-width. But it is not the ordinary histogram with smaller bin-width. Let us define the ASH properly. Suppose, we choose $t_0 = 0$. Then, $B_i = [(i-1)h, ih)$ with $i \in Z$ defines the ordinary histogram with origin at 0. Let us generate $M-1$ new bin grids by shifting each $B_j$ by the amount $lh/M$ to the right. Then, for each shift we will get ordinary histogram with origin at $lh/M$ and

$B_{jl} = [(j-1+l/M)h, (j+l/M)h)$ for $l \in \{0, \ldots, M-1\}$.. So, $M$ histograms. Then we get $M$ different estimates for true density $f$ at each $x$.

$$\hat{f_{h,l}}(x) = \frac{1}{nh} \sum_{i=1}^{n} I(X_i \in B_{i,l})$$

Then, the ASH estimate of $f$ at $x$ is defined as -

$$\hat{f}_h(x) = \frac{1}{M} \sum_{i=1}^{n} \hat{f_{h,l}}(x)$$

As $M \to \infty$, the ASH is not dependent on the origin anymore and converts from a step function into a continuous function. This asymptotic behavior can be directly achieved by a different technique: Kernel density estimation.

This is the implementation in R of the above discussed method.

```
Avg_shifted.histogram <- function(x,data,h,M){

  ests_at_x <- NULL
  origin.ash <- (0:(M-1))*h/M

  origin.hist <- function(chosen.origin){
    j <- ceiling((x - chosen.origin)/h)
    mean(data < chosen.origin + j*h & data >= chosen.origin + (j-1)*h)/h
  }

  ests_at_x <- vapply(origin.ash, FUN = origin.hist, FUN.VALUE = 2)

  return(mean(ests_at_x))
}
```

But, this is tedious to work with. Because, of computational time and it will be more as we increase $M$ .

```
#we should always utilize the asymptotic behavior.
system.time(Avg_shifted.histogram(0,rnorm(500),0.5,50000))
```

```
##    user  system elapsed
##    0.14    0.00    0.47
```

Instead we will use a different approach. We will try to utilize the idea of Kernel density estimation here.

# 6 Kernel Density Estimation :

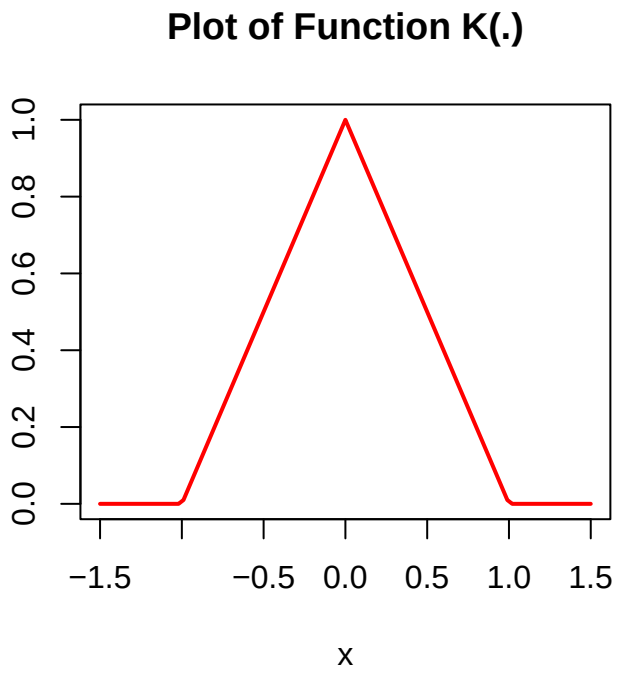
In Histogram , we put equal weights to the points $X_1, X_2, ...X_n$. Kernel Density estimation is a generalization of that. Here, we put non-uniform weights. Suppose, we have a random sample of size $n$, $X_1, X_2, ...X_n$. If we choose a function $K(.)$ as our weighting function (which has some nice properties), then kernel density estimator is defined as -

$$\hat{f}(x; h) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - X_i}{h})$$

Where, $K(.)$ is a function which is generally a density which is symmetric and uni modal at zero. Although we can take non-symmetric densities as choice of kernel.

Now, we will try to utilize the idea of Kernel density estimation for Average Shifted Histogram Case. We will choose Triangular Kernel. The kernel is given by -

$$K(x) = \begin{cases} (1 - |x|), & |x| \leq 1 \\ 0 & \text{o.w} \end{cases}$$




The above is the plot of the triangular kernel. You can see that it is triangular shaped within the range $[-1, 1]$. It can be shown that, Kernel density estimator obtained by chossing triangular kernel is same as Average Shifted Histogram as $M \to \infty$.Thus, by choosing Triangular kernel, we can develop the maximum likelihood method to find optimal $h$ in a similar method for Average Shifted Histogram.

## 6.1 Maximum Likelihood Based Approach for finding Optimal $h$ :

If we follow the same notations described in case of Maximum Likelihood method for Histogram. Then,

$$L\hat{(h)} = -n_2 log(n_1 h) + \sum_{j=1}^{n_2} log(\sum_{i=1}^{n_1} (1 - |\frac{X_{2j} - X_{1i}}{h}|)\ I(|\frac{X_{2j} - X_{1i}}{h}| \leq 1)$$

Then, for some values of $X_{2j}$ we will have $\sum_{i=1}^{n_1} (1 - |\frac{X_{2j} - X_{1i}}{h}|)\ I(|\frac{X_{2j} - X_{1i}}{h}| \leq 1) = 0$. Which will make $\frac{1}{n_2} L(\hat{h})$ make $-\infty$. So, to avoid that problem we will replace those zero's with smallest nonzero value of the estimated density for test data.That will make $\sum_{m=1}^{k} \hat{L_m}(h)$ as a function of $h$ finite.

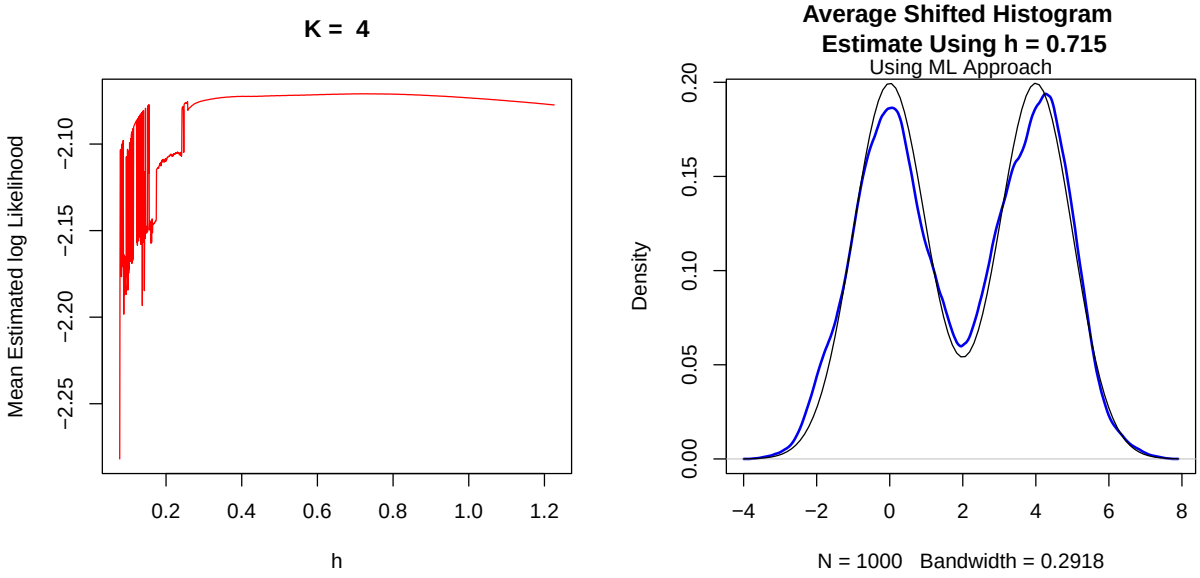
Below is the R implementation of the ML method in case of triangular kernel.

```r
Optimal_ASH_ML <- function(data,k = 2,mplot = T){

  assertthat::assert_that(k > 1)

  split.data <- split(1:length(data),sample(1:length(data),length(data),replace = F)%%k)
  r <- extendrange(range(data))

  Estimated_Likelihood <- function(h) {

    lik.vals <- lapply(split.data,FUN = function(test.data.index){

     training.data <- data[-test.data.index]
     test.data <- data[test.data.index]
     fitted.density <- density(training.data,kernel = "tri",bw = h*sqrt(1/6),
                from = min(training.data) - h,to = max(training.data) + h,
     n = max(2^(log2(ceiling(31*(max(training.data) - min(training.data) + 2*h)/(2*h) + 1))),
             2^10))
     approx.fitted.density <- approxfun(fitted.density$x,fitted.density$y)
     lik.val <- approx.fitted.density(test.data)
     lik.hh <- ifelse(is.na(lik.val) | (lik.val == 0),
                    min(lik.val[!(is.na(lik.val) | (lik.val == 0))]),lik.val)
     sum(log(lik.hh))

    })

    sum(unlist(lik.vals))/length(data)
  }

  n <- mean(unlist(lapply(split.data,FUN = length)))
  hhrange <- diff(r) / n^c(0.9, 0.4)
  hh <- seq(hhrange[1], hhrange[2], length.out = 1500)

  mean_est.loglikelihood <-
    vapply(hh, FUN = Estimated_Likelihood,
           FUN.VALUE = 2)

  if(mplot){
    plot(hh, mean_est.loglikelihood,type = 'l',col = 'red',xlab = 'h',
         ylab = 'Mean Estimated log Likelihood',main = paste("K = ",k))

  }

  h.optimal <- hh[which.max(mean_est.loglikelihood)]

  if(mplot){
    final.density <- density(data,kernel = "tri",bw = h.optimal*sqrt(1/6))
    plot(final.density,col = "blue2",lwd = 2,
         main = paste("Average Shifted Histogram \n Estimate Using h =",round(h.optimal,3)))
    mtext("Using ML Approach",side = 3)
  }

  return(h.optimal)
}
```

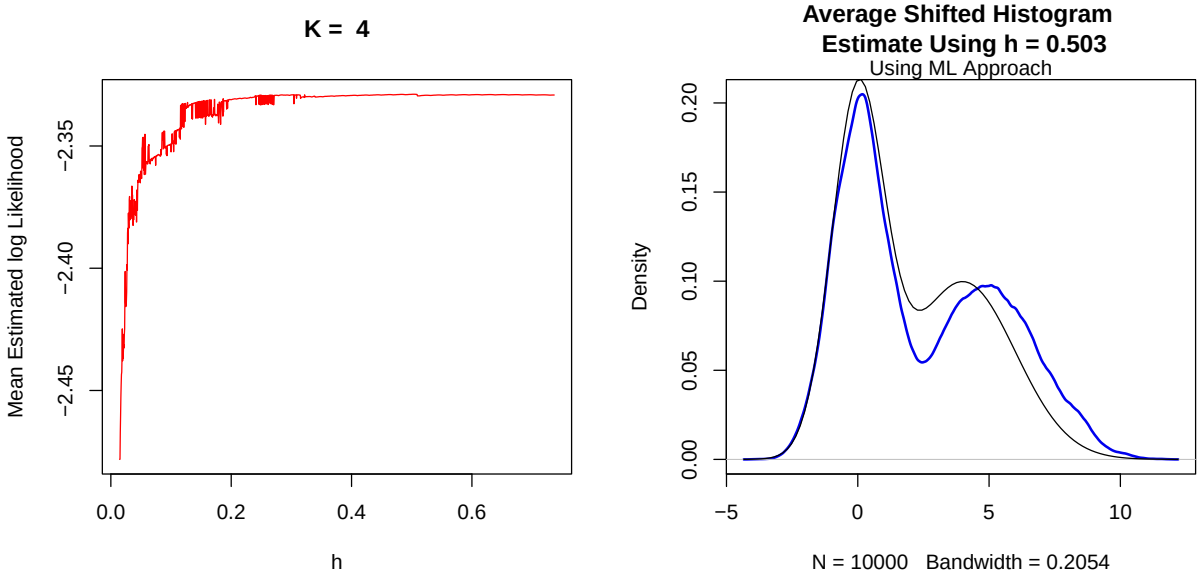
`Optimal_ASH_ML` also uses same data dependent values of $h$ and by default.

We have illustrated this using some examples.

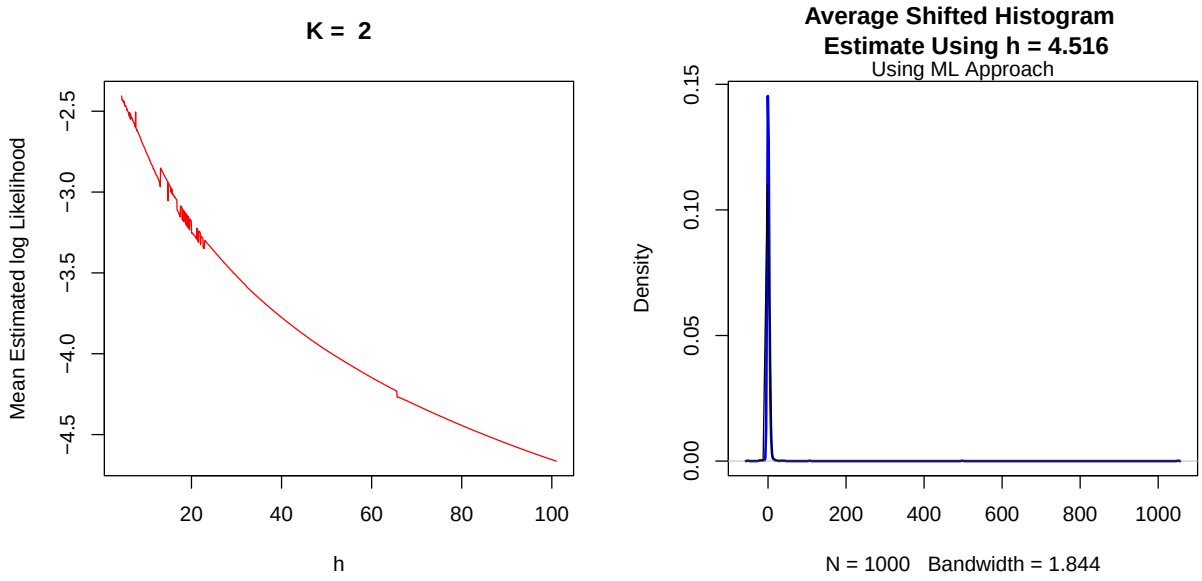
```
## [1] n =  1000, Optimal h =  0.714875350061073
```



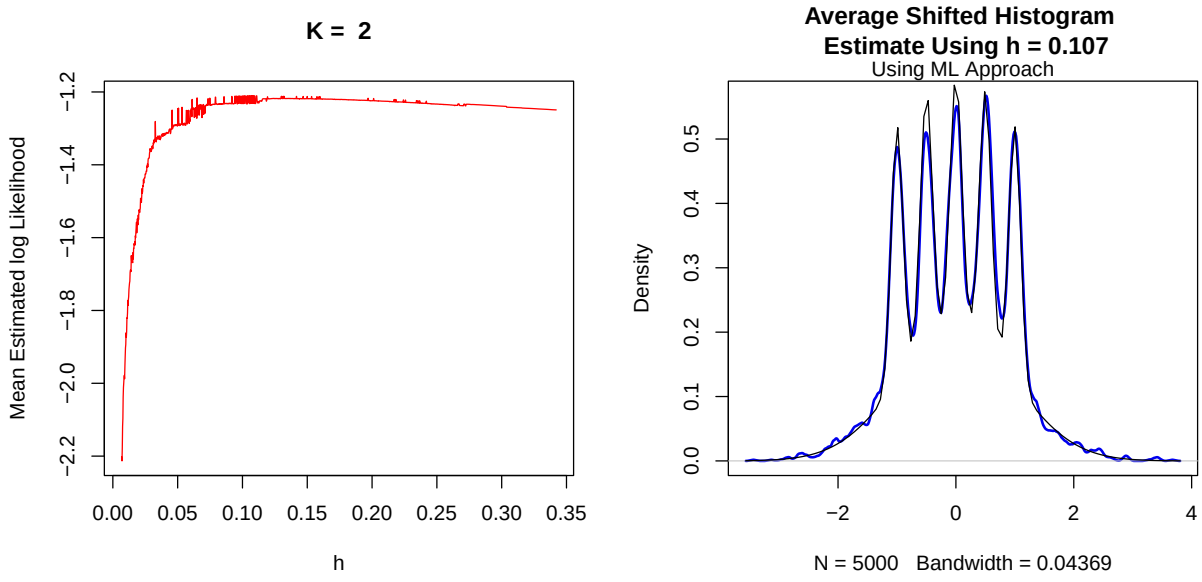

```
## [1] n = 10000, Optimal h =  0.503141014845552
```




```
## [1] n = 1000, Optimal h =  4.51576258146526
```




```
## [1] n = 5000, Optimal h =  0.107029023296336
```




From above plots, we can see that the ASH is giving very good density estimate with optimal $h$ obtained using Likelihood method. Also, notice that *mean estimated log-likelihood* as a function of $h$ is smooth now. The problem we were facing in case histogram is not here.

We can extend this ML based $h$ selection approach for other kernels also. Below is the **R** implementation of ML method for general kernel.

```
Optimal.KDE_ML <- function(data,k = 2,mplot = T,mkernel = "gaussian"){

  assertthat::assert_that(k > 1)
```

```
eta <- switch(mkernel,gaussian = 1,rectangular = 1/sqrt(3),
                      triangular = 1/sqrt(6),epanechnikov = 1/sqrt(5),
                      biweight = 1/sqrt(7),optcosine = sqrt(1-8/pi^2),
                      cosine = sqrt(1/3 - 2/pi^2))

m <- 1
m <- ifelse(mkernel == "gaussian",3.5,k)

split.data <- split(1:length(data),sample(1:length(data),length(data),replace = F)%%k)
r <- extendrange(range(data))

Estimated_Likelihood <- function(h){

  lik.vals <- lapply(split.data,FUN = function(test.data.index){

   training.data <- data[-test.data.index]
   test.data <- data[test.data.index]
   fitted.density <- density(training.data,kernel = mkernel,bw = h*eta,
                        from = min(training.data) - m*h,to = max(training.data) + m*h,
   n = max(2^(log2(ceiling(31*(max(training.data) - min(training.data) + 2*m*h)/(2*m*h) + 1))),
            2^10))
   approx.fitted.density <- approxfun(fitted.density$x,fitted.density$y)
   lik.val <- approx.fitted.density(test.data)
   lik.hh <- ifelse(is.na(lik.val) | (lik.val == 0),
                    min(lik.val[!(is.na(lik.val) | (lik.val == 0))]),lik.val)
   sum(log(lik.hh))

  })

  sum(unlist(lik.vals))/length(data)
}

n <- mean(unlist(lapply(split.data,FUN = length)))
hhrange <- diff(r) / n^c(0.9, 0.4)
hh <- seq(hhrange[1], hhrange[2], length.out = 1500)

mean_est.loglikelihood <-
  vapply(hh, FUN = Estimated_Likelihood,
         FUN.VALUE = 2)

if(mplot){
  plot(hh, mean_est.loglikelihood,type = 'l',col = 'red',xlab = 'h',
       ylab = 'Mean Estimated log Likelihood',main = paste("K = ",k))

}

h.optimal <- hh[which.max(mean_est.loglikelihood)]

if(mplot){
  final.density <- density(data,kernel = mkernel,bw = h.optimal*eta)
  plot(final.density,col = "aquamarine",lwd = 2,
       main = paste("Kernel Density Estimate,(",mkernel,"Kernel)\n Using h =",
                    round(h.optimal,3)))
```

```
    mtext("Using ML Approach",side = 3)
  }

  return(h.optimal)
}
```

`Optimal.KDE_ML` takes `data`,`k`,`kernel` as input. By **default** `mkernel = "gaussian"`. However, any of the following kernel can be used.

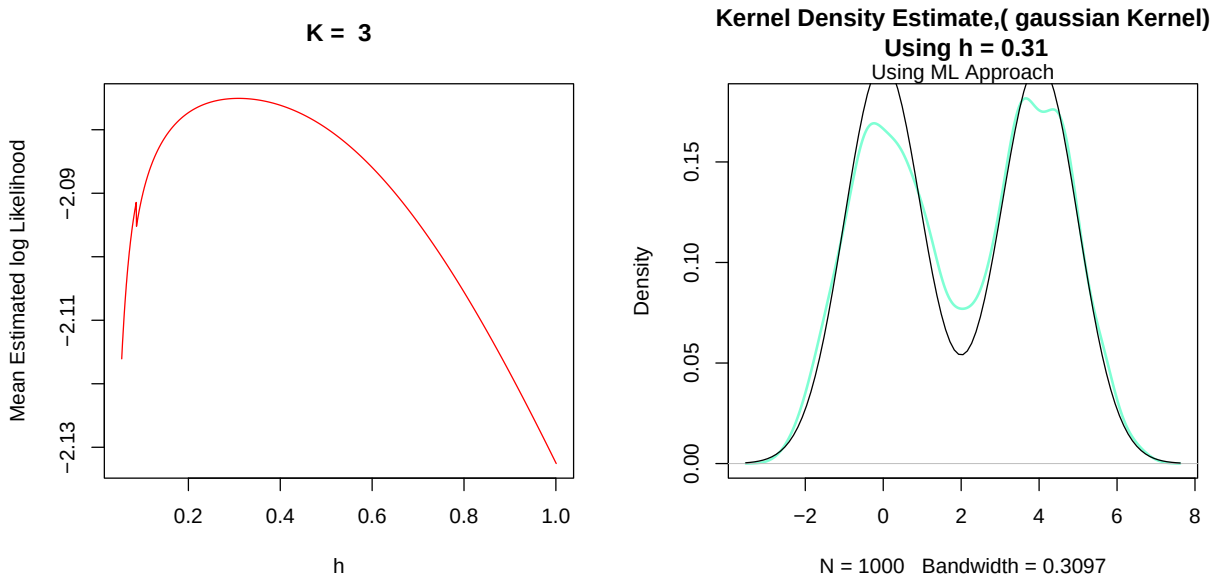
```
eval(formals(density.default)$kernel)
```

```
## [1] "gaussian"     "epanechnikov" "rectangular"  "triangular"   "biweight"
## [6] "cosine"       "optcosine"
```

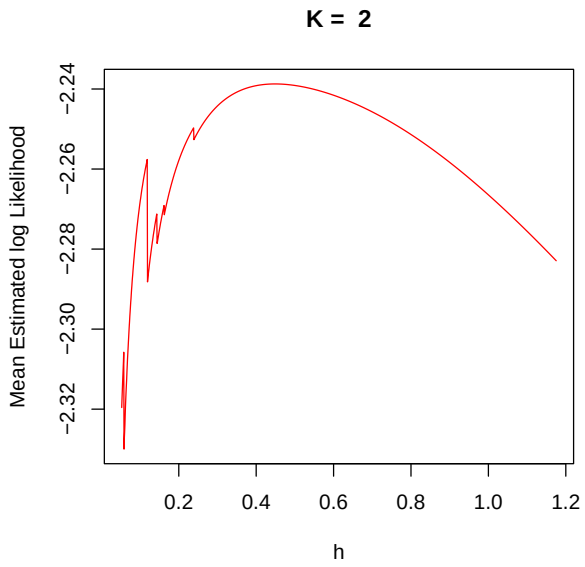
By **default** `mplot = TRUE`, which signifies whether to return the plots or not.

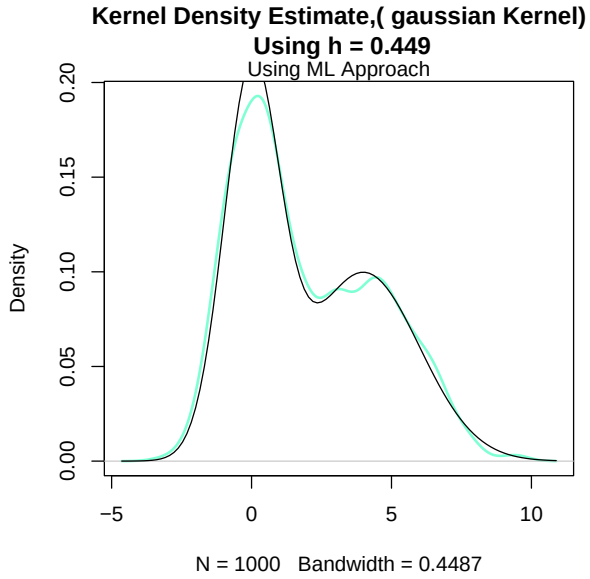
We have illustrated it using some examples. In each case $n = 1000$.

```
## [1] 0.3097485
```

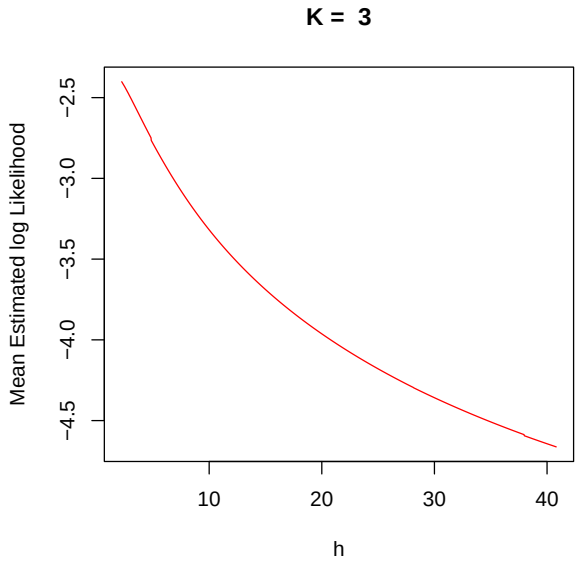


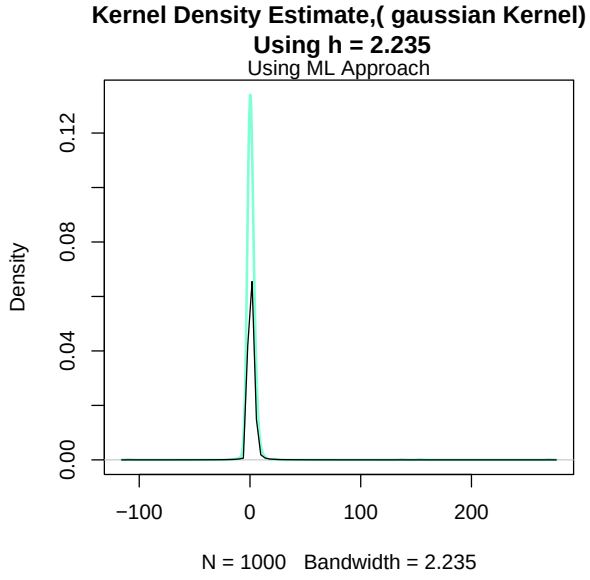

```
## [1] 0.4487482
```



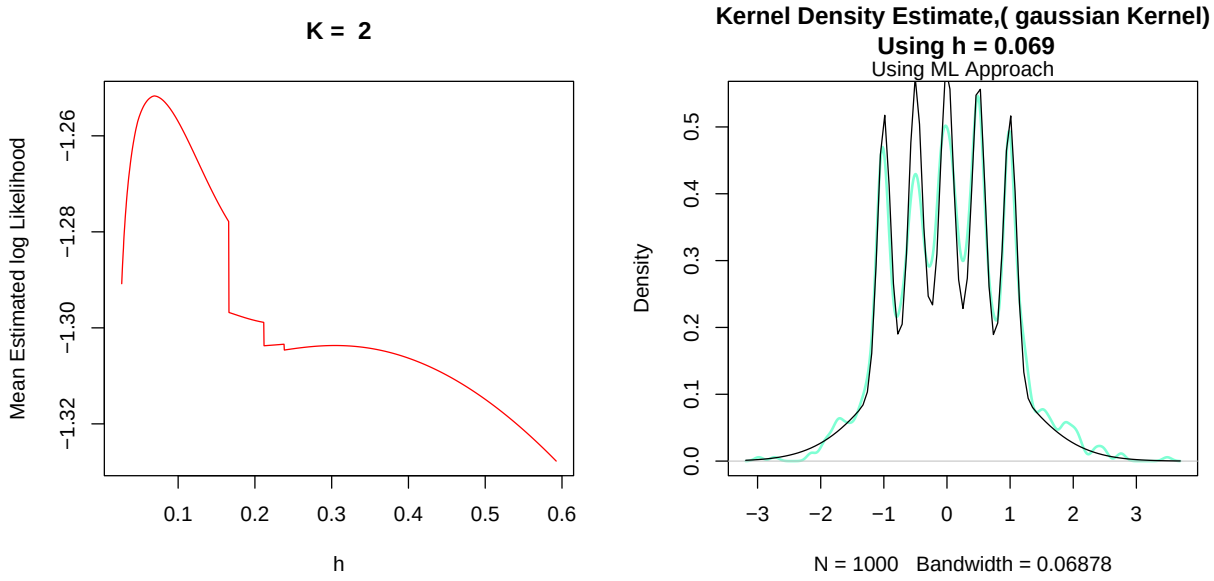




```
## [1] 2.235384
```




```
## [1] 0.06878425
```




Similarly, we can extend this method for other kernels. Here is the implementation of that in *R*.

## 6.2 AIC Based Bin-Width Selection Approach :

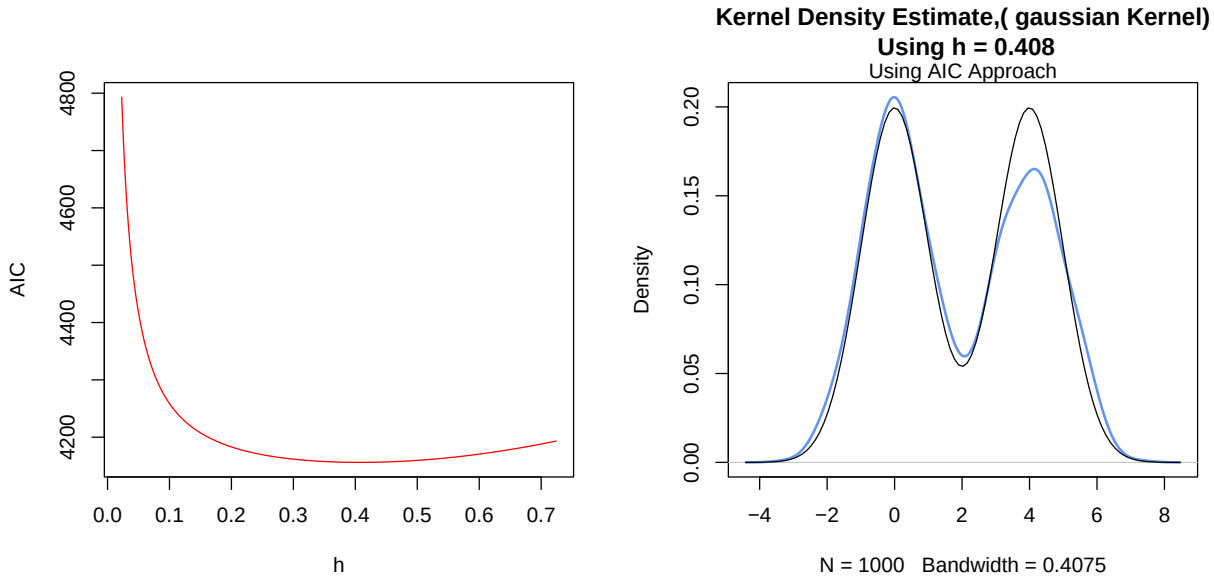
A similar type of penalty based approach can be developed here. Here, we don't have number of bins. But, we can consider $\frac{Range}{h}$ as penalty. Below is the R implementation of it.

```
Optimal.KDE_AIC <- function(data,mplot = TRUE,mkernel = "gaussian"){

  eta <- switch(mkernel,gaussian = 1,rectangular = 1/sqrt(3),
                        triangular = 1/sqrt(6),epanechnikov = 1/sqrt(5),
                        biweight = 1/sqrt(7),optcosine = sqrt(1-8/pi^2),
                        cosine = sqrt(1/3 - 2/pi^2))

  m <- 1
  m <- ifelse(mkernel == "gaussian",3.5,k)

  r <- extendrange(range(data))
  n <- length(data)

  h.AIC <- function(h) {

    fitted.density <- density(data,kernel = mkernel,bw = h*eta,
                              from = min(data) - m*h,to = max(data) + m*h,
    n = max(2^(log2(ceiling(31*(max(data) - min(data) + 2*m*h)/(2*m*h) + 1))),
            2^10))
    approx.fitted.density <- approxfun(fitted.density$x,fitted.density$y)
    lik.val <- approx.fitted.density(data)
    lik.hh <- ifelse(is.na(lik.val) | (lik.val == 0),
                     min(lik.val[!(is.na(lik.val) | (lik.val == 0))]),lik.val)

    return(- 2*sum(log(lik.hh)) + 2*(diff(range(data))/h))
  }
```

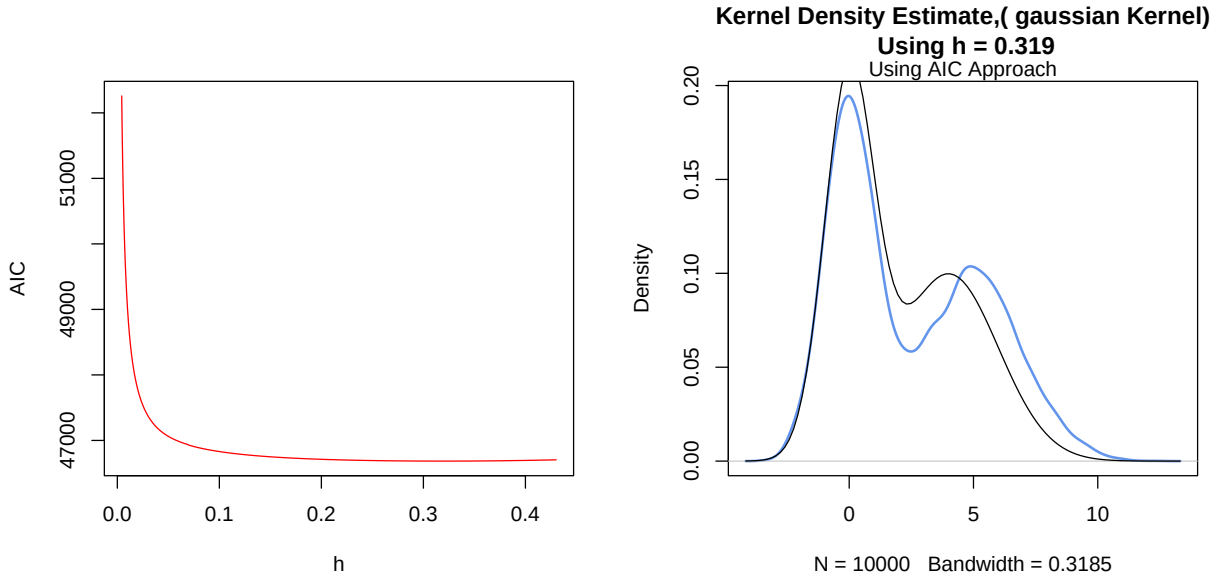
```
  hhrange <- diff(r) / n^c(0.9, 0.4)
  hh <- seq(hhrange[1], hhrange[2], length.out = 1500)

  AICVal <-
    vapply(hh, FUN = h.AIC,FUN.VALUE = 2)

  if(mplot){
    plot(hh,AICVal,type = 'l',col = 'red',xlab = 'h',
         ylab = 'AIC')

  }

  h.optimal <- hh[which.min(AICVal)]

  if(mplot){
    final.density <- density(data,kernel = mkernel,bw = h.optimal*eta)
    plot(final.density,col = "cornflowerblue",lwd = 2,
         main = paste("Kernel Density Estimate,(",mkernel,"Kernel)\n Using h =",
                      round(h.optimal,3)))
    mtext("Using AIC Approach",side = 3)
  }

  return(h.optimal)
}
```

Except $k$, `Optimal.KDE_ML` and `Optimal.KDE_AIC` has the same set of arguments and same outputs. We have illustrated some examples using it.

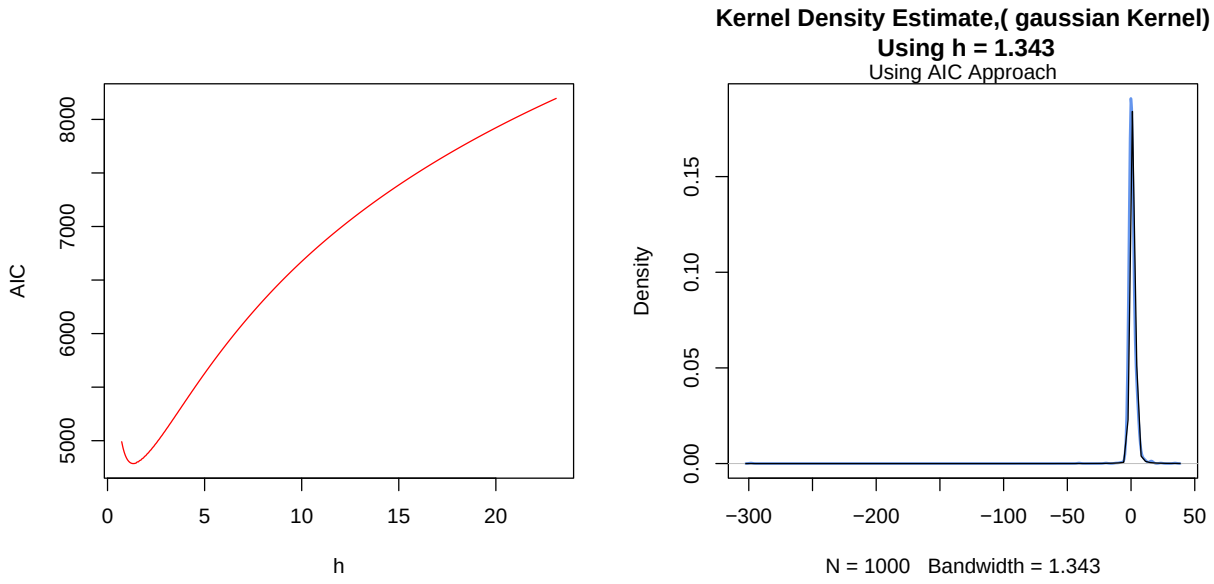
```
## [1] n =  1000, Optimal h =  0.407505821566271
```



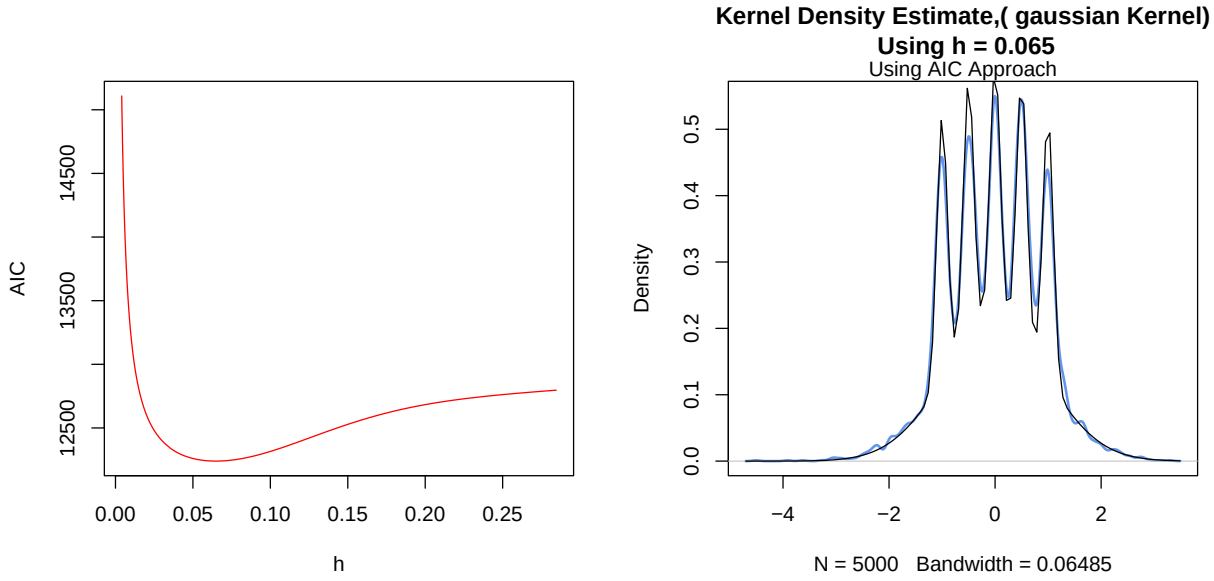

```
## [1] n = 10000, Optimal h =  0.318519347550103
```




```
## [1] n = 1000, Optimal h =  1.34277152941448
```




```
## [1] n = 5000, Optimal h =  0.0648535425163952
```

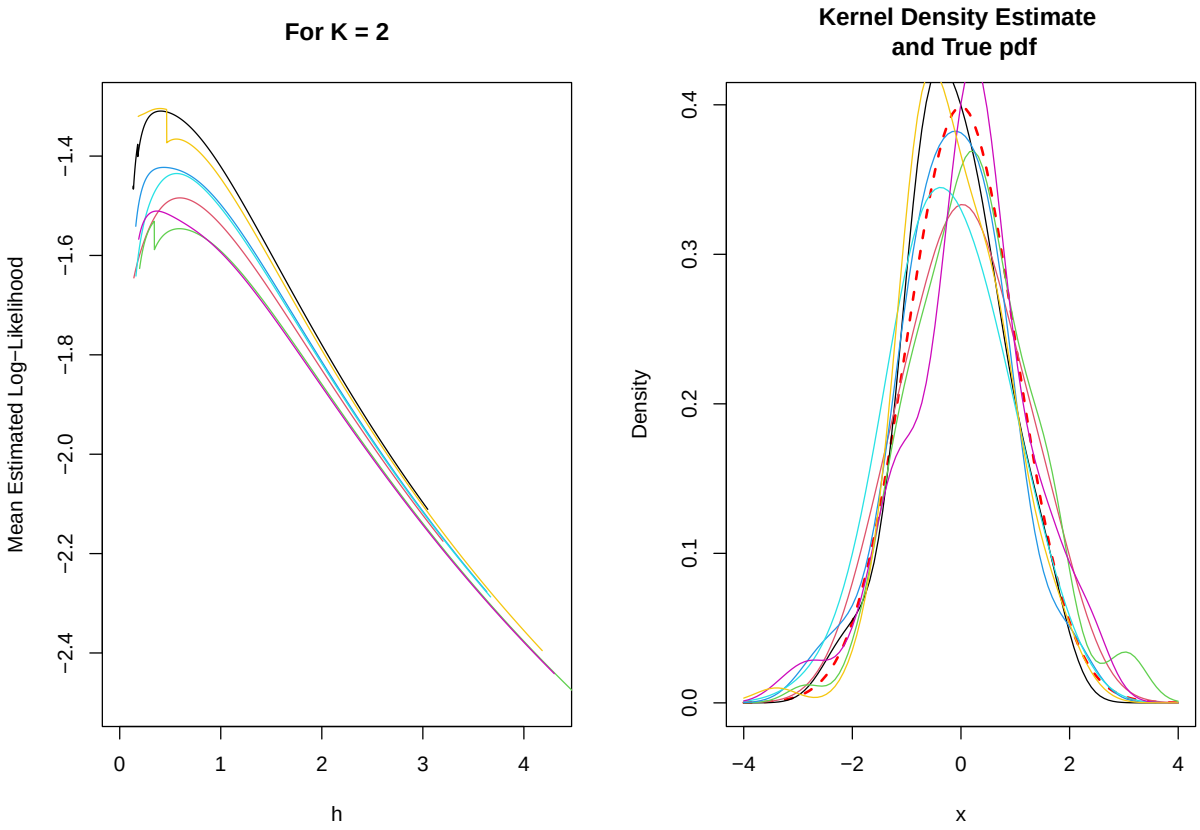



So, visually it is performing well.

## 6.3 Discussion :

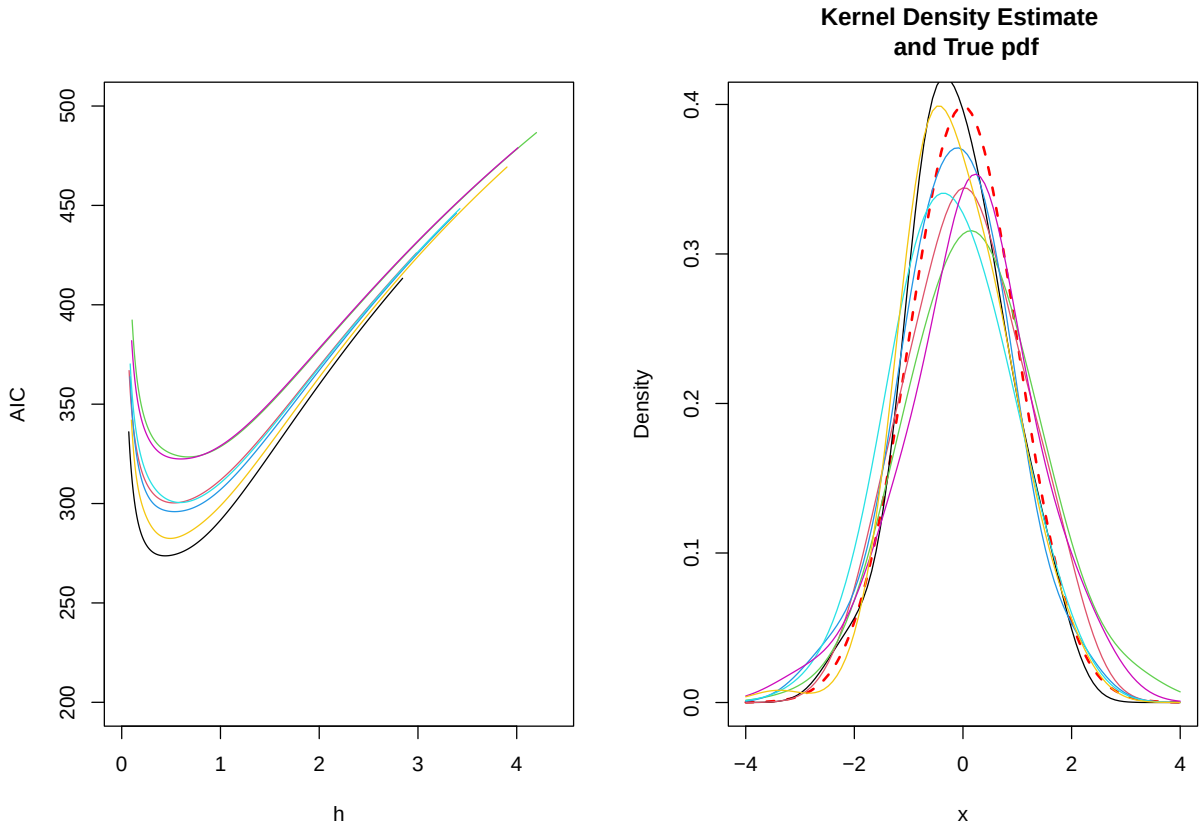
A very important thing is to study the nature of the functions that we are maximizing or minimizing in vase of ML method and AIC based method. The main reason is that function is random. In case of histogram, we have seen that such function changes it's value very rapidly in a small neighborhood. So, we will consider simulating data from from Normal distribution and we will plot the associated function (For ML, it is $\frac{1}{n}\sum_{m=1}^{k} \hat{L_m}(h)$ and For AIC, it is AIC) along with the estimated density.

### 6.3.1 ML Method :




From the above plot, we can see that the function $\frac{1}{n}\sum_{m=1}^{k} \hat{L_m}(h)$ is not very different for different data. But the estimates are not very stable.

### 6.3.2 AIC Based Method :




From the above plot, we can see that the function $AIC(h)$ is not very different for different data. Here the estimates are stable compared to Ml method.

## 6.4 Other Methods to find Optimal $h$ for Kernel Density Estimation :

There are several methods to find optimal band-width in case of kernel density estimation.

### 6.4.1 Scott's rule of thumb :

Scott suggested to take -

$$h = \frac{1.06\hat{\sigma}}{n^{\frac{1}{5}}}$$

Where, $\hat{\sigma}$ is sample standard deviation.This method requires the assumption that, the underlying population is Normal.

### 6.4.2 Silverman's rule of thumb :

Silverman suggested to take -

$$h = \frac{0.9\min(IQR, \hat{\sigma})}{n^{\frac{1}{5}}}$$

This method is more robust compared to Scott's method.

Thus, in simple cases (e.g., in the case of unimodal distribution), we can safely use the Scott's or Silverman's rule of thumb (the Silverman's rule is recommended because it's more robust). They work extremely fast and produce an excellent bandwidth value for the normal distribution and distributions close to normal.

However, if we are not sure about approximately or exact normality. We should use other methods such as -

i. Unbiased Cross Validation ([Rudemo1982], [Bowman1984])
ii. Biased Cross validation ([Scott1987])
iii. Sheather and Jones method ([Sheather1991])

**R** has implementation of all these methods.You can get more insight of other methods by `?bw.bcv`, `?bw.ucv` and `?bw.SJ`.

# 7 Comparison of Different Methods :

Till now, we are visually inspecting whether the estimated density is good or not. Basically, we were looking at $|f(\hat{x}) - f(x)|\ \forall x$. If this is small for $\forall x$, then our density estimator is good. So, we can look at $\int_{-\infty}^{\infty} |f(\hat{x}) - f(x)|\, dx$. But, it is difficult to handle mathematically. But, we can calculate it in **R** !

```
data <- wavethresh::rclaw(1000)


# Using ML method
h.optimal <- Optimal_ASH_ML(data,k=3,mplot = FALSE)
ML_optimal.density <-  density(data,kernel = "tri",bw = h.optimal*sqrt(1/6),
                  from = -5.5,to = 5.5,n = 2^15)
#estimated density
ML_est.density <- approxfun(ML_optimal.density$x,ML_optimal.density$y)
ML_Abs_Loss.func <- function(x){
  abs(ML_est.density(x) - dclaw(x))
}

#Using Silverman's method
SilverMan_optimal.density <- density(data,kernel = "tri",bw = "nrd0",
                                     from = -5.5,to = 5.5,n = 2^15)
#estimated density
SilverMan_est.density <- approxfun(SilverMan_optimal.density$x,
                                   SilverMan_optimal.density$y)
SilverMan_Abs_Loss.func <- function(x){
  abs(SilverMan_est.density(x) - dclaw(x))
}

#Using ML Method
integrate(f = ML_Abs_Loss.func,lower = -5,upper = 5,
          subdivisions = 200)
```

```
## 0.147477 with absolute error < 8.3e-05
```

```
#Using Silverman's Rule
integrate(f = SilverMan_Abs_Loss.func,lower = -5,upper = 5,
          subdivisions = 200)
```

```
## 0.3317322 with absolute error < 8.3e-06
```

We can see that, the error for Silverman's method is at least 1/2 the error for ML method.Similarly, we can try with other distributions and methods. However, we will also try some different methods for comparison. If we think carefully, our aim is to find distance between two distributions. One is the true density and other is the estimated density. For that, we can try **Kolmogorov-Smirnov Distance**, **Kullback-Leibler Distance**. The former is based on CDF and the latter is based on density.

## 7.1 Kolmogorov-Distance Statistic :

If $F$ and $G$ denote the distribution function of two distributions $P$ and $Q$ respectively. Then, Kolmogorov-Smirnov Distance is defined as -

$$KS(P,Q) = \underset{x \in R}{Sup} |F(x) - G(x)|$$

Small value of $KS(P,Q)$ indicates that the two distributions are very similar. In our context of density estimation, if $G$ denotes the estimated CDF derived from estimated density and $F$ denotes the true CDF. Then, small values of this measure would indicate good performance of the estimator. Notice that, the measure is symmetric in $P$ and $Q$.

## 7.2 Kullback-Leibler Distance :

If $f$ and $g$ denote the continuous density function of two distributions $P$ and $Q$ respectively. Then, Kullback-Leibler Distance is defined as -

$$KL(P,Q) = \int_S f(x) log(\frac{f(x)}{g(x)})\, dx$$

Where, $S = S_f \cap S_g$ is the common support of the two distributions. In our context of density estimation, if $f$ denotes the true density and $g$ denotes the estimated density. Then, small values of this measure would indicate good performance of the estimator. Here, the measure is not symmetric in $P$ and $Q$ as in case of Kolmogorov-Smirnov Distance.

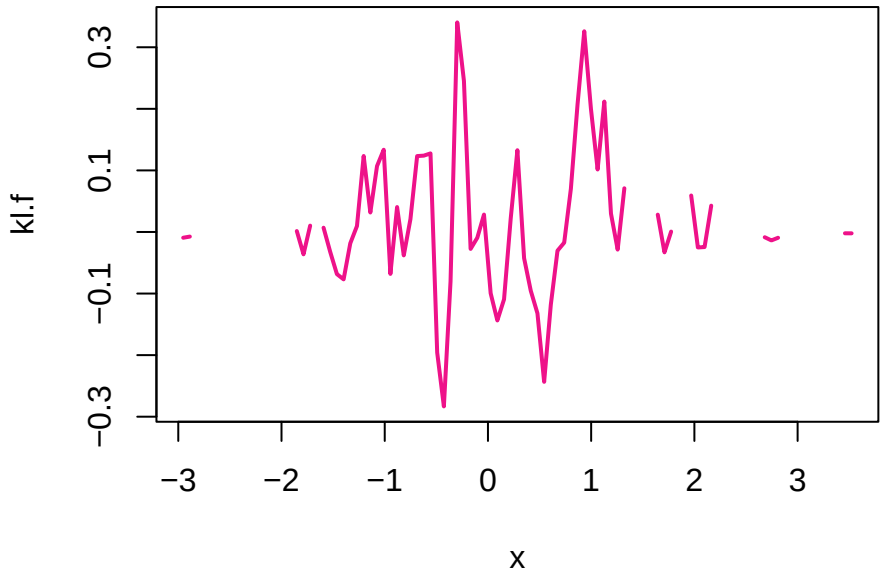
### 7.2.1 Computational Issues of Kullback-Leibler Distance :

If $\hat{f}$ denotes the estimated density and $f$ is the true density. Then, usually we choose $S = (X_{(1)}, X_{(n)})$. But, it is seen that based on the choice of kernel and the true pdf, the behavior of the integrand $f(x) log(\frac{f(x)}{\hat{f}(x)})$ is not good. Consider an example -

Suppose, $f$ is pdf of **Claw** Distribution and $\hat{f}$ is estimated density using Triangular kernel with $h = 0.1$ based on a sample of size $n = 100$.

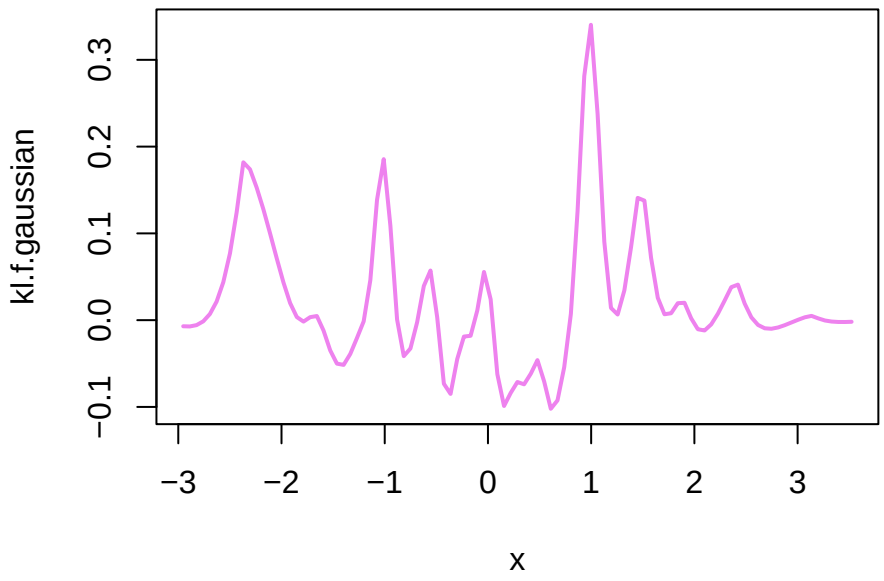
```
data =  wavethresh::rclaw(100)
h <- 0.1
est.density <- function(x){
  mean(EnvStats::dtri((x - data)/h,min=-1,max=1,mode = 0))/h
}
kl.f <- function(x){
  val = wavethresh::dclaw(x)*log(wavethresh::dclaw(x)/vapply(x,
```

```
                      FUN = est.density,FUN.VALUE = 2))
  return(val)
}

plot(kl.f,from = min(data),to = max(data),col = "deeppink2",lwd = 2)
```




The incomplete lines basically indicate that the integrand is $\infty$ at those points. So, integrating such a function is difficult. This behavior has been observed in other combination of choices of kernel and true density. The problem in our example is that, $\hat{f}$ becomes zero for some values of $x$, due to triangular kernel being zero outside of an interval($[-1, 1]$). But, if we choose kernel which is non zero for any finite value. Then, the problem can be resolved. For example - If we use Gaussian kernel for the last example.

```
est.density.gaussian <- function(x){
  mean(dnorm((x - data)/h))/h
}
kl.f.gaussian <- function(x){
  val = wavethresh::dclaw(x)*log(wavethresh::dclaw(x)/vapply(x,
                      FUN = est.density.gaussian,FUN.VALUE = 2))
  return(val)
}

plot(kl.f.gaussian,from = min(data),to = max(data),col = "violet",lwd = 2)
```




Now, the integrand is behaving well. Thus, we will base our analysis on **Gaussian** kernel only. **R** has `integrate` function which performs numerical integration. You can check `?integrate`. However, We will use definition of **Riemann Integral**. If we assume that, the integration is convergent. Then,

$$KL(f, \hat{f}) = \int_{[a,b]} f(x)\ log(\frac{f(x)}{\hat{f}(x)})dx \approx \Delta \sum_{i=1}^{k} f(t_i)\ log(\frac{f(t_i)}{\hat{f}(t_i)})$$

Where, $a = t_0 < t_1 < t_2 < .... < t_k = b$ and $\Delta = t_i - t_{i-1}\ \forall i$. $k$ needs to be sufficiently large for the approximation.(Right hand Riemann Sum) Choice of $a$ and $b$ can be $X_{(1)}$ and $X_{(n)}$ respectively. However, we choose $a$ and $b$, such that Integrand is well defined.

### 7.3 Negative Log-Likelihood Based Loss :

We can use log likelihood for comparison. The idea is very simple. We have data from an unknown density and we want to estimate it. To estimate the unknown density, we can use histogram or kernel density estimator. Using, different methods, we get different values of $h$. i.e. we get different types of histogram or kernel density estimator i.e. different density estimates. Now, we want to find, among these suggested densities(estimated densities), from which the data is more likely to come. So, we will see for which estimated density the likelihood is maximized or equivalently for which estimated density the estimated negative log-likelihood is minimum. If we use the same data to find the log-likelihood. Then intuitively there would be some bias. Instead, we will use new data from the same distribution. In practice, we cannot do it. But for comparison based on simulation, we can do that.

Suppose, we have $m$ different density estimates denoted by, $\hat{f}_1(.), \hat{f}_2(.), ...., \hat{f_m}(.)$. Based on new data, we calculate $\hat{L_1}, \hat{L_2}, ..., \hat{L_m}$. Then, based on the above discussed method we choose that $i$ for which $\hat{L_i}$ is maximum.

### 7.4 Integrated Absolute Loss :

We have already discussed this comparison criteria. The Integrated Absolute Loss is defined as -

$$IABSL(f, \hat{f}) = \int_S |f(x) - \hat{f}(x)| dx$$

Small values of this are good. Here also, we can use same type of approximation to find the integral.

### 7.5 Calculation of CDF from Estimated Density :

If we use kernel density estimation, then it is not very difficult to obtain CDF from estimated density. The reason is very beautiful. If we look at, kernel density estimator -

$$\hat{f}(x; h) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} K(\frac{x - X_i}{h})$$

Then, you can understand that it is mixture of kernels actually. If the associated kernel is pdf. Then, $\hat{f}(x; h)$ resembles to a mixture distribution with weights $\frac{1}{n}$ each. For example - If our choice of kernel is **Gaussian**, then $\hat{f}(x; h)$ is clearly pdf of Mixture Normal distribution with $\mu = \{X_1, X_2, ...X_n\}$ and Standard Deviation $h$. Thus, estimated CDF is -

$$\hat{F}(x; h) = \frac{1}{n} \sum_{i=1}^{n} M(\frac{x - X_i}{h})$$

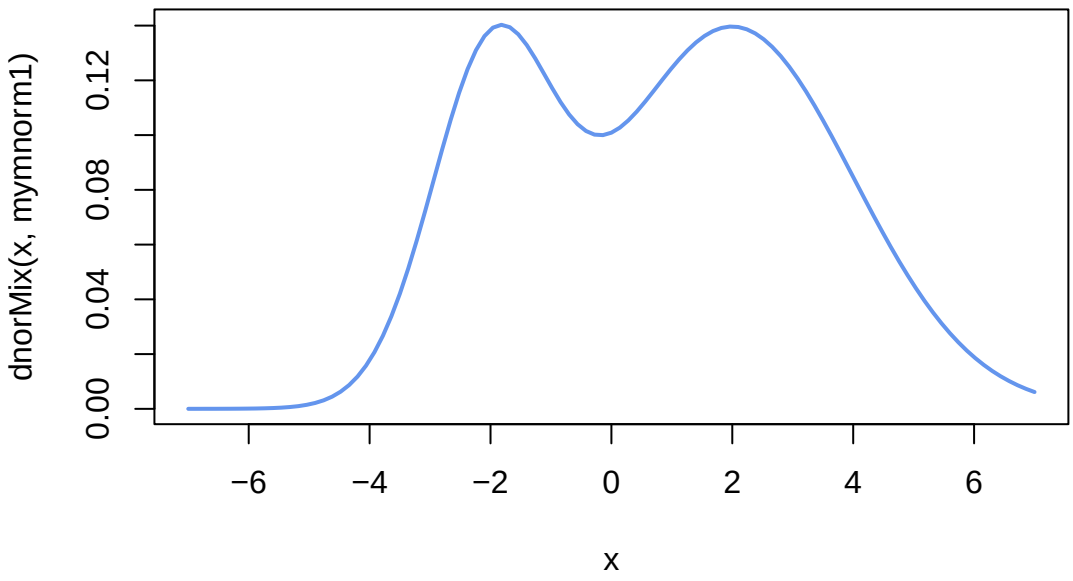
Where, $M(u) = \int_{-\infty}^{\infty} K(u)\ du$.

## 8 Simulation Study :

We have already discussed some methods to find bandwidth. We can adopt some methods for finding bandwidth using methods to find binwidth. **Why ?** Because, if we use triangular kernel. Then, bandwidth of triangular kernel is 2 times of binwidth of the corresponding histogram. Using all these methods, we will perform a simulation study to compare them in-terms of different measures. In each case Simulation Number is 2000.

### 8.1 Simulation - 1 :

We will consider the following Mixture Normal Distribution.



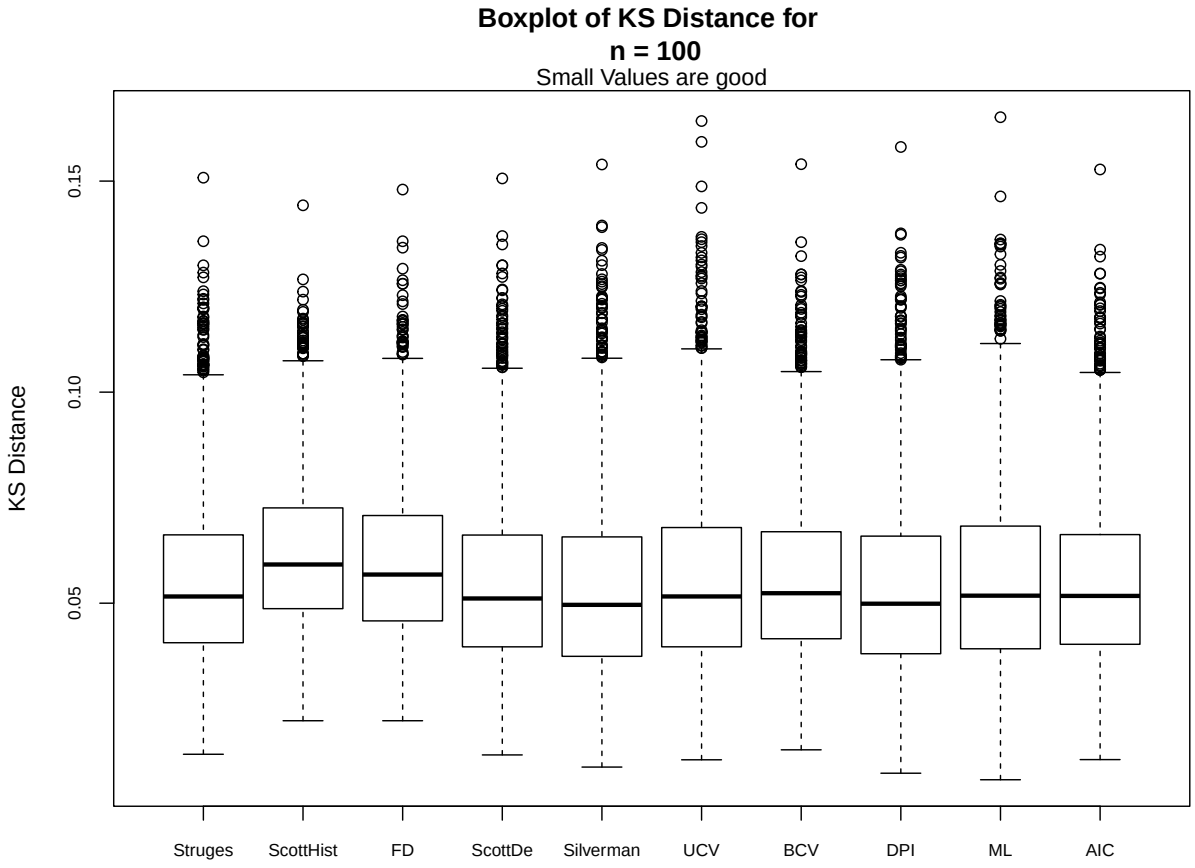

The pdf $f$ is given by -

$$f(x) = 0.3f_1(x) + 0.7f_2(x)$$

Where, $f_1(.)$ is the pdf of Normal(-2,1) and $f_2(.)$ is the pdf of Normal(2,2^2) distribution. We will consider different sample sizes. Such as n = 100,500,2000 and will compare different methods.

#### 8.1.1 For n = 100 :

##### 8.1.1.1 KS Distance :

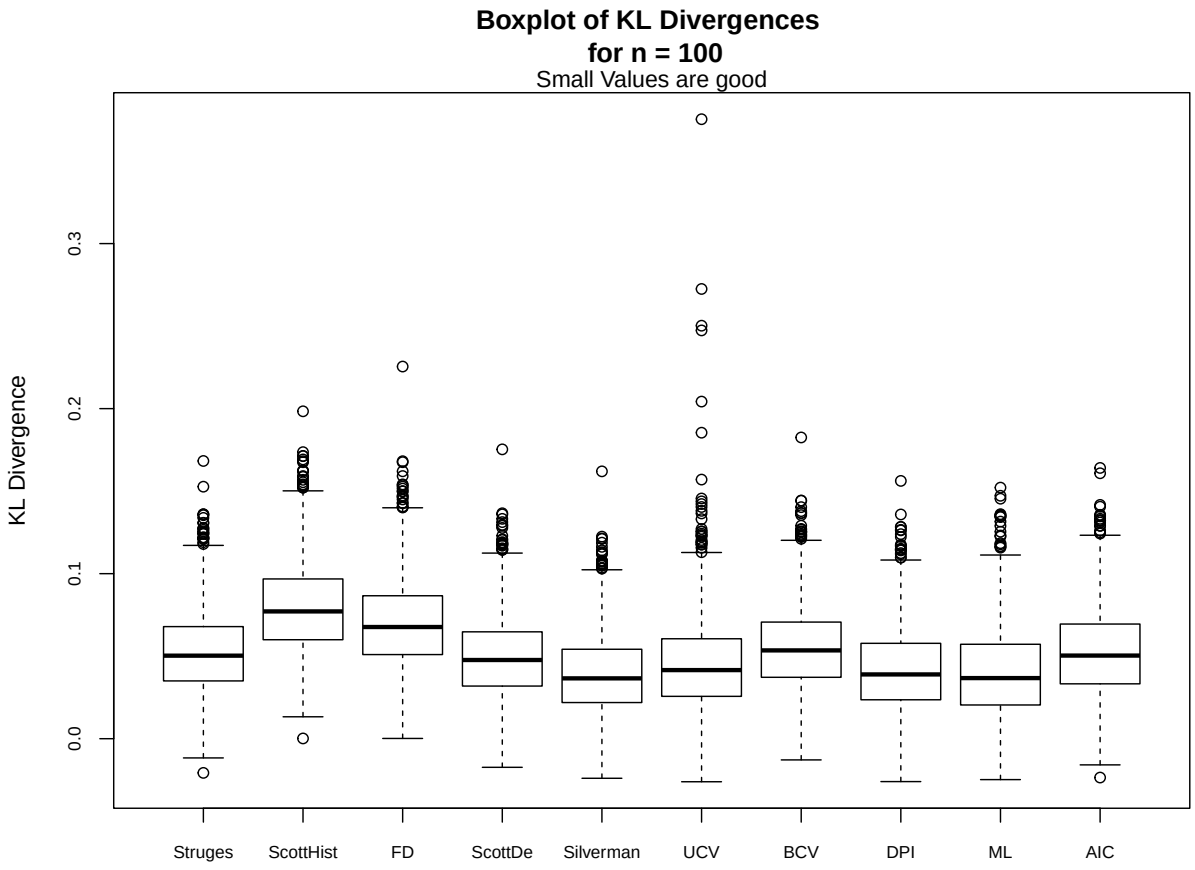



From the above Boxplot it is clear that, the last 6 methods(Silverman,UCV,BCV,DPI,ML and AIC) are performing very similarly. They are not very different. Also interestingly the *Kernel Density adjusted Struges Method* is performing quite similarly. Let's look at the Monte Carlo estimate of Expected KS Distances.

```
##  Silverman        DPI    ScottDe        AIC    Struges         ML        UCV
## 0.05354669 0.05404258 0.05476544 0.05500844 0.05509071 0.05564828 0.05583192
##        BCV         FD  ScottHist
## 0.05588728 0.05967351 0.06172304
```

From the Expected KS Distances also, We can see that the above mentioned methods are performing similarly and **Silverman** has least Monte carlo estimated Expected KS Distance.

#### 8.1.1.2 KL Divergence :

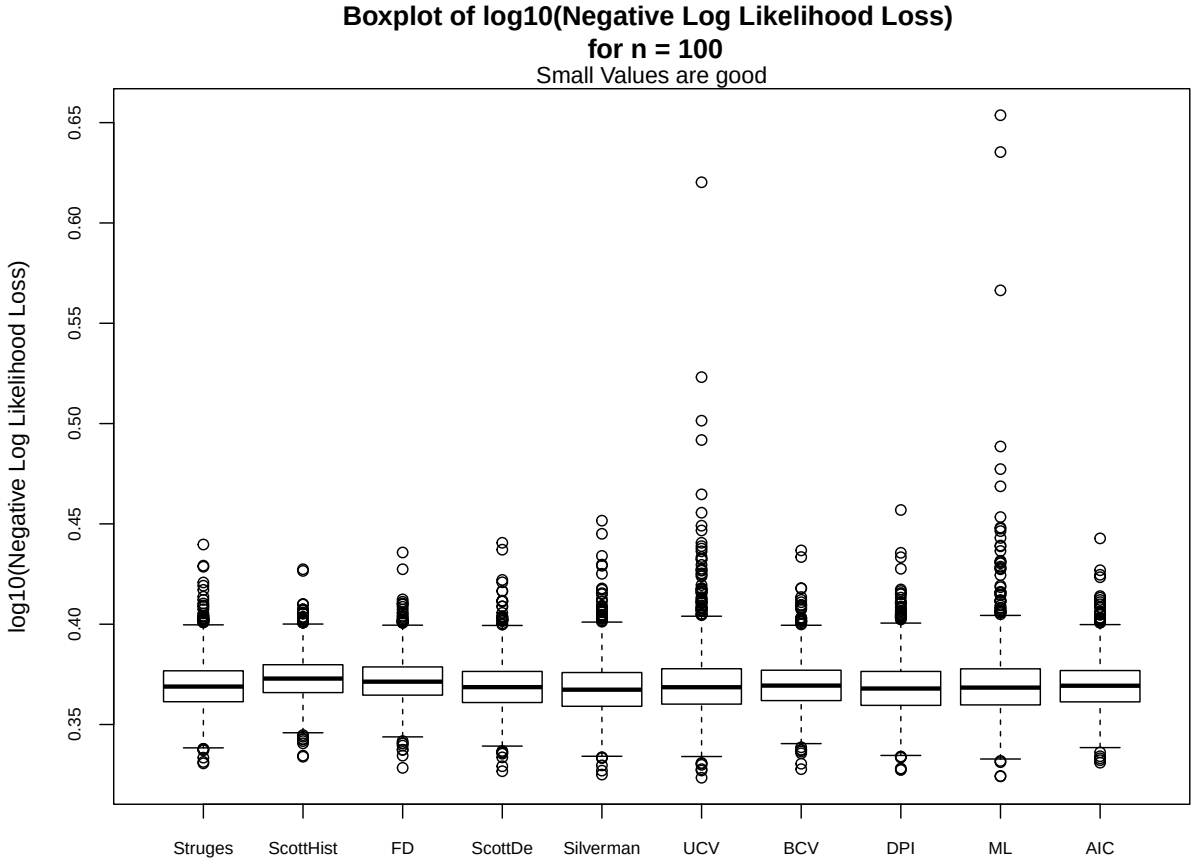



From the above Boxplot, we can see that Silverman,DPI and ML methods are performing similarly. For UCV method there are many large outliers also and these outliers are interesting and important. Now, let's look at the Monte carlo estimate of Expected KL Divergences.

```
##  Silverman         ML        DPI        UCV    ScottDe    Struges        AIC
## 0.03916323 0.04012291 0.04146220 0.04469539 0.04943443 0.05210051 0.05234620
##        BCV         FD  ScottHist
## 0.05502990 0.06970203 0.07918135
```

From the Expected KL Divergence also, We can see that the above mentioned methods are performing similarly and interestingly here also Silverman has least KL Divergence.

#### 8.1.1.3 Negative Log Likelihood Loss :

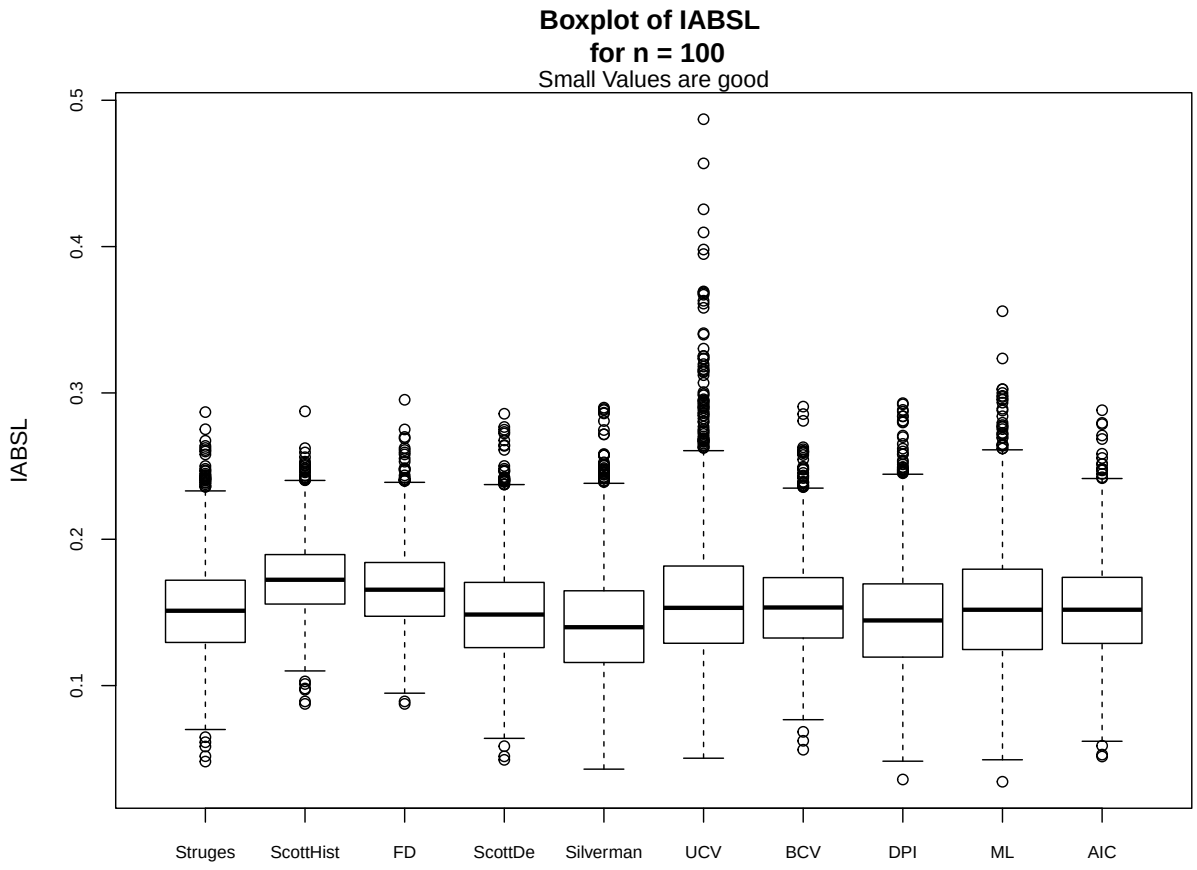



Here, we have drawn the Boxplot of log10(Negative Log Likelihood Loss). All methods are performing similarly.Even the *Kernel Density Adjusted* histogram methods also. But, **UCV** and **ML** method have many large outliers. Here, the meaning of Average of these Negative Log Likelihood Losses are not clear. But, let's see the Average values.

```
## Silverman       DPI   ScottDe   Struges       AIC       BCV       UCV        ML
##  2.335334  2.337244  2.340402  2.342246  2.342803  2.343951  2.346465  2.347325
##        FD ScottHist
##  2.354156  2.361071
```

Interestingly, here also Silverman method has least average Negative Log Likelihood Loss.

#### 8.1.1.4 Integrated Absolute Loss :



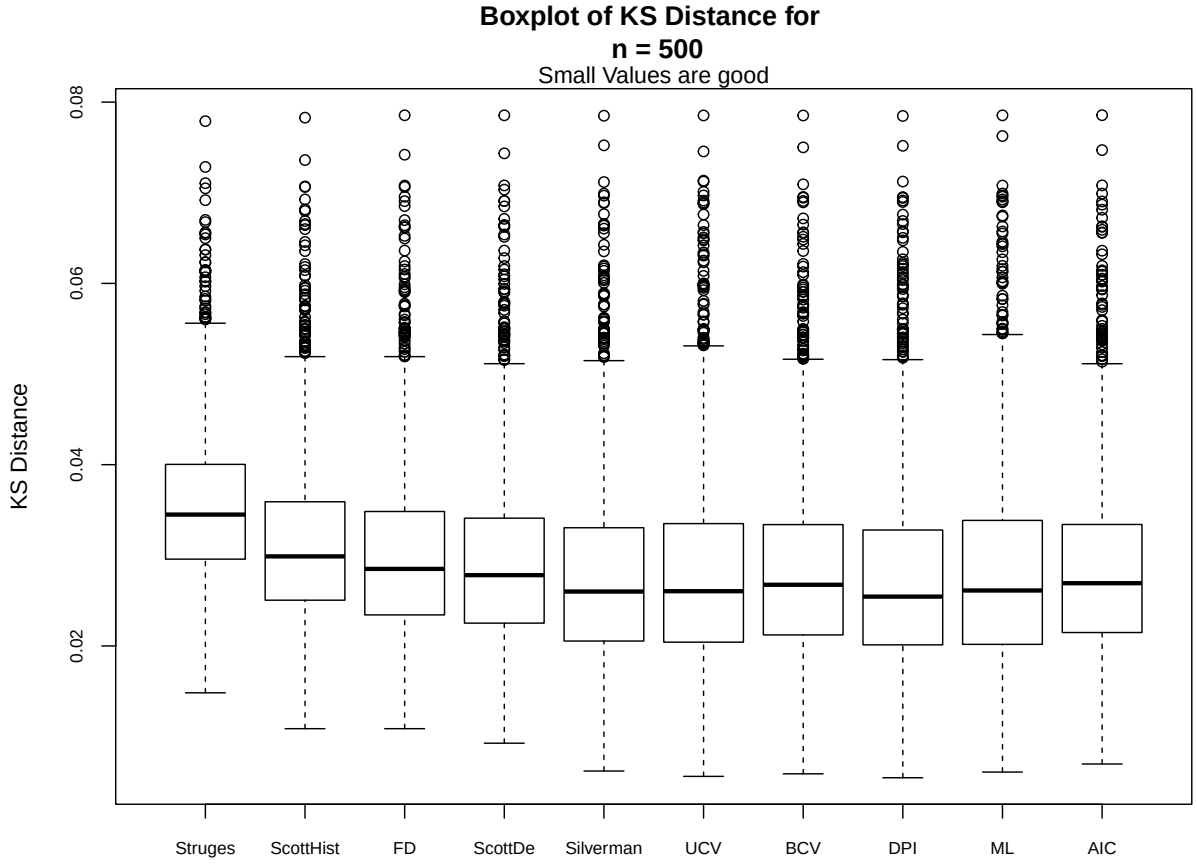

From the above Boxplot it is clear that, Silverman,DPI, Scott Density and *Kernel Density Adjusted* Struges method are performing very similarly and interestingly, UCV method has many large outliers, which are very interesting. Now, let's look at the Monte carlo estimate of Expected IABSL.

```
## Silverman       DPI   ScottDe   Struges       AIC        ML       BCV       UCV
## 0.1427680 0.1463579 0.1499022 0.1519670 0.1522281 0.1547057 0.1548651 0.1590318
##        FD ScottHist
## 0.1668133 0.1733614
```

So, the above mentioned methods are performing similarly and interestingly, here also **Silverman** has least Monte carlo estimated Expected IABSL. Let's see whether the above observations remain same incase of n = 500.

### 8.1.2 For n = 500 :

#### 8.1.2.1 KS Distance :

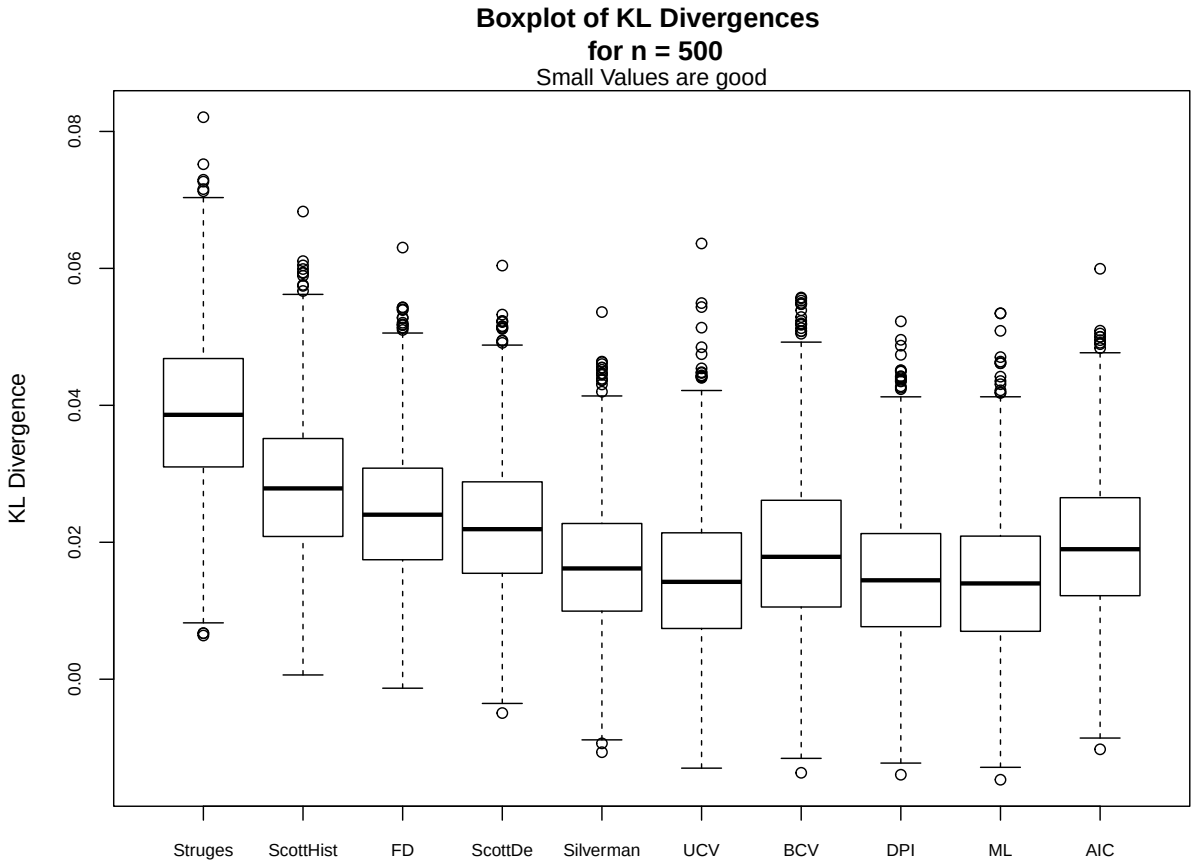



From the above Boxplot, it is clear that the last six methods are performing very similarly. The *Kernel Density Adjusted* Struges method is no longer performing similar to these six methods. Let's see the Monte carlo estimate of Expected KS Distances.

```
##         DPI  Silverman         ML        UCV        BCV        AIC    ScottDe
## 0.02739484 0.02763203 0.02786457 0.02791435 0.02818759 0.02833682 0.02924049
##         FD  ScottHist    Struges
## 0.03001576 0.03127023 0.03537972
```

From the Expected KS Distances, we are getting some interesting observations. The **ML** method is performing well than in case of n = 100 and it is performing very similar to the DPI (which has least Monte Carlo Estimated expected KS distance).

#### 8.1.2.2 KL Divergence :

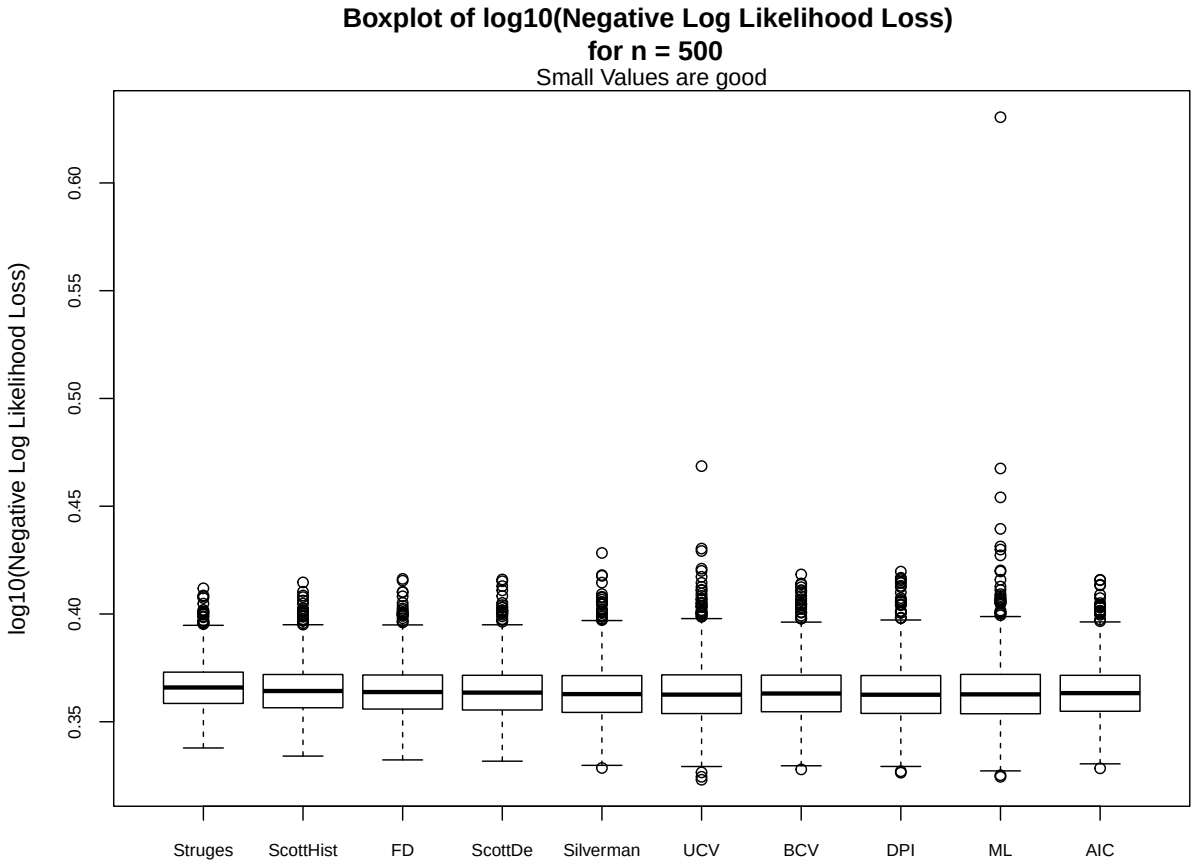



From the above Boxplot, we can see that UCV,ML,DPI and Scott Density are performing similarly. Which was true in case of n = 100 also. But now UCV has not very large outliers. Let's look at the Monte carlo estimate of Expected KL Divergences.

```
##         ML        UCV        DPI  Silverman        BCV        AIC    ScottDe
## 0.01438508 0.01465311 0.01491472 0.01664046 0.01862097 0.01945087 0.02239273
##         FD  ScottHist    Struges
## 0.02449577 0.02836035 0.03918129
```

Here, **ML** method has least Monte carlo estimated KL Divergence. Which is very interesting.

#### 8.1.2.3 Negative Log Likelihood Loss :



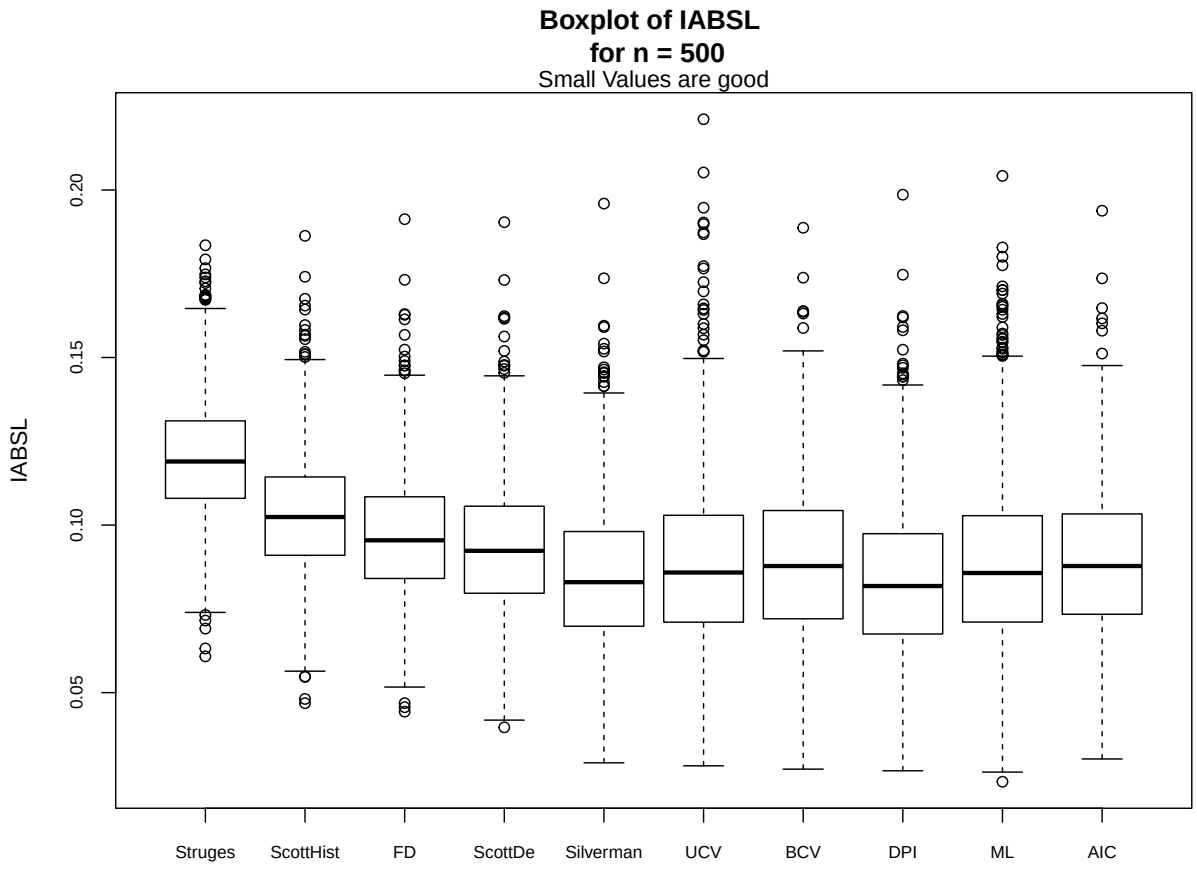

From the above Boxplot, it is clear that all methods are performing similarly and ML method has one large outliers.

The Average values of Negative Log Likelihood Loss are -

```
##       DPI Silverman       UCV       BCV       AIC        ML   ScottDe        FD
##  2.309947  2.310447  2.311288  2.311705  2.311948  2.312681  2.313461  2.314757
## ScottHist   Struges
##  2.317217  2.324784
```

The average values are very similar. Also, we should note that DPI method has least average Negative Log Likelihood Loss.

#### 8.1.2.4 Integrated Absolute Loss :



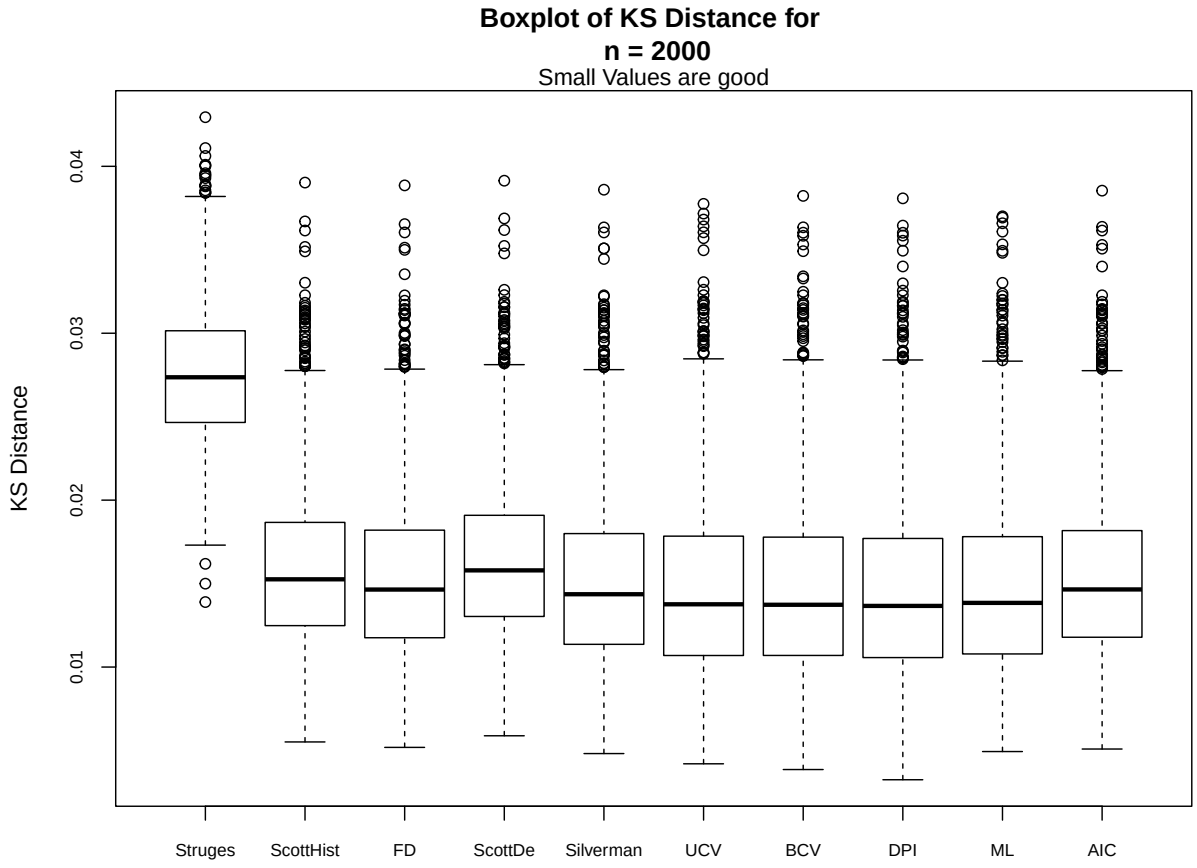

From the Boxplot, it is clear that Silverman,DPI and ML method are performing very similarly. Also, UCV has large many outliers (Which was in case of n = 100 also). Let's look at the Monte carlo estimate of Expected IABSL.

```
##         DPI  Silverman         ML        UCV        BCV        AIC    ScottDe
## 0.08351043 0.08450243 0.08825404 0.08832601 0.08854013 0.08887680 0.09327997
##         FD  ScottHist    Struges
## 0.09672551 0.10296285 0.11935871
```

We are getting very similar kind of observations. Except that *ML* method is performing better than *n = 100* case.

### 8.1.3 For n = 2000 :

#### 8.1.3.1 KS Distance :

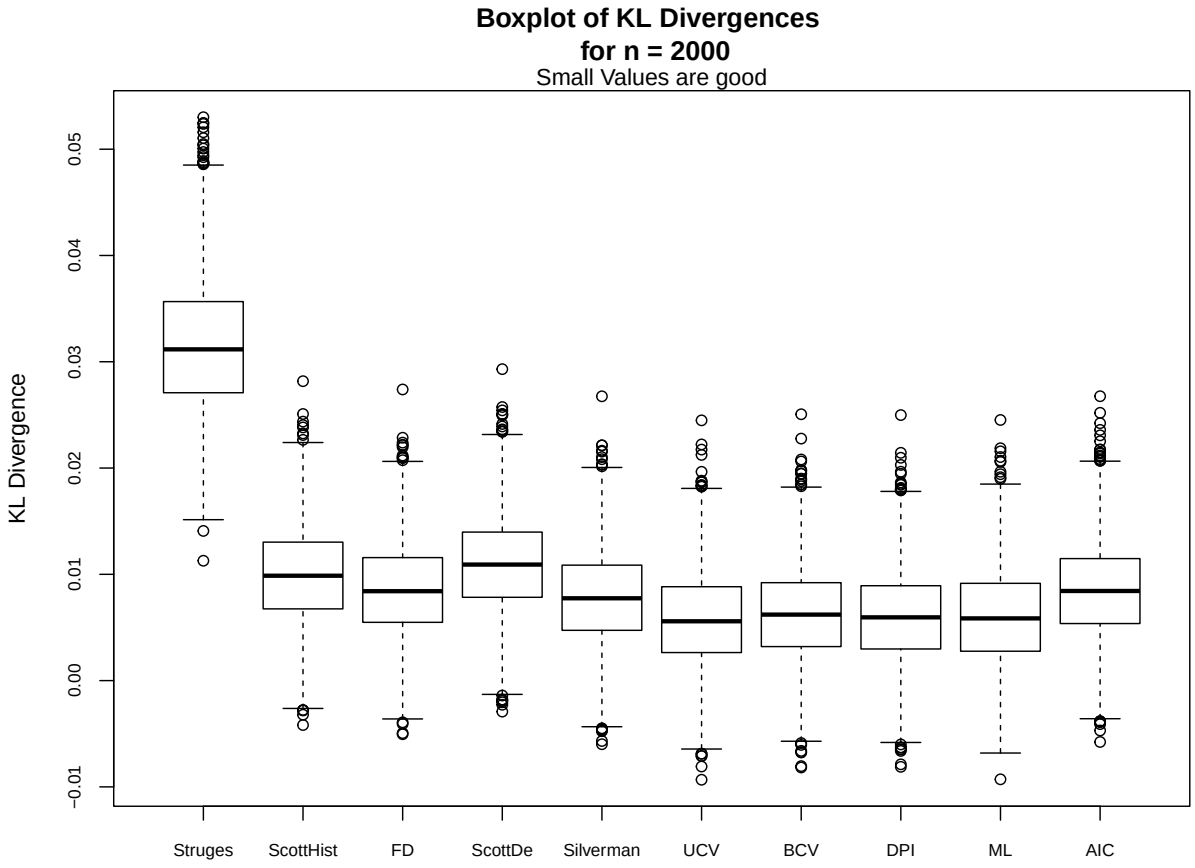



From the Boxplot, it is clear that except the *Kernel Density Adjusted* Struges method, all are performing very similarly. Let's look at the Monte carlo estimate of Expected KS Distances.

```
##        DPI        BCV        UCV         ML  Silverman        AIC         FD
## 0.01455181 0.01461718 0.01471075 0.01471151 0.01509021 0.01535802 0.01536807
##  ScottHist    ScottDe    Struges
## 0.01597786 0.01645306 0.02749818
```

We are getting similar type of observations like $\mathbf{n = 500}$ and the expected KS distance values are not very different.

#### 8.1.3.2 KL Divergence :

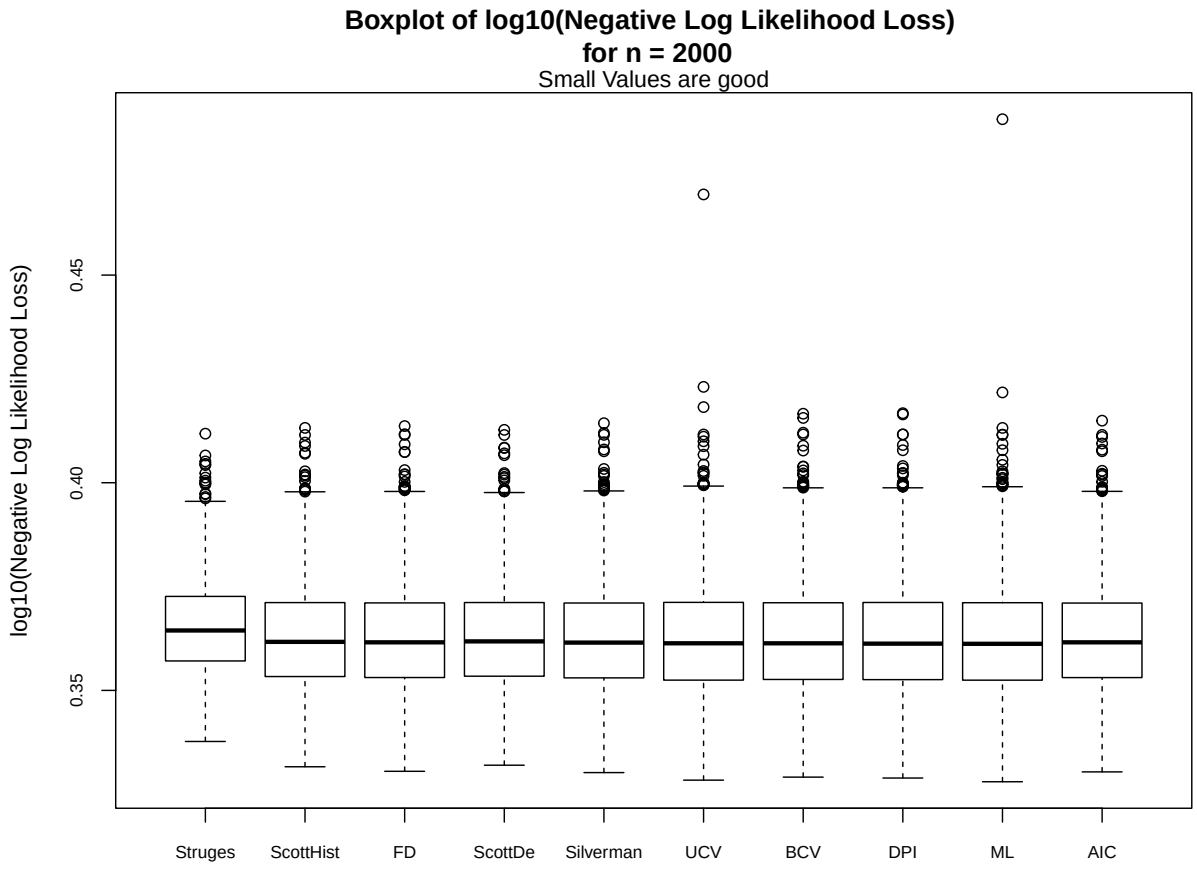



From the Boxplot, we are getting similar kind of observations as for KS Distance. Let's look at Monte carlo estimate of Expected KL Divergences.

```
##         UCV          ML         DPI         BCV   Silverman         AIC
## 0.005746545 0.005989210 0.006029304 0.006277462 0.007866614 0.008531124
##          FD   ScottHist     ScottDe     Struges
## 0.008545196 0.009949245 0.010968512 0.031556184
```

Here, also we are getting very similar kind of observations.

#### 8.1.3.3 Negative Log Likelihood Loss :



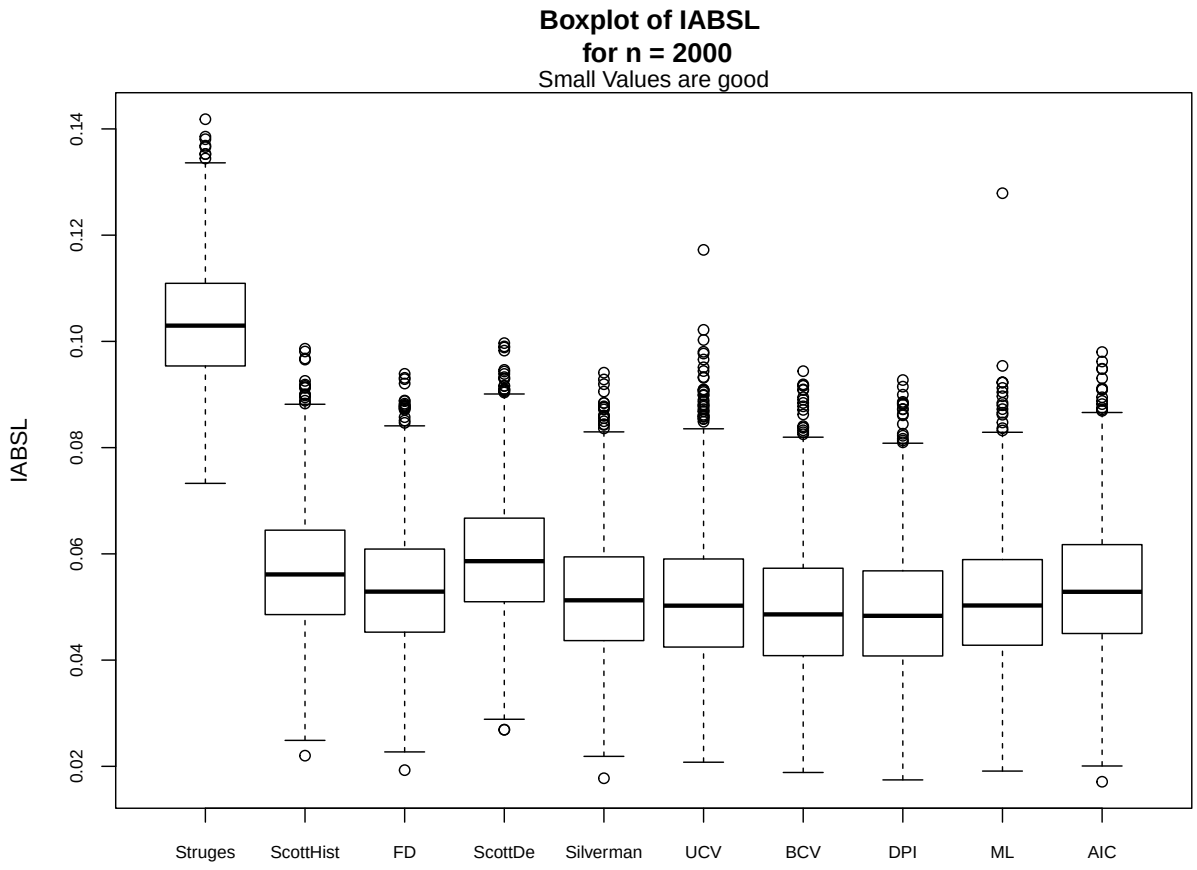

From the Boxplot, it is clear that all methods are performing similarly. As, we have observed in case of n = 100,500.

The Average values of Negative Log Likelihood Losses are -

```
##       DPI       BCV       UCV Silverman        ML        FD       AIC ScottHist
##  2.305008  2.305089  2.305533  2.305562  2.305740  2.305833  2.305848  2.306484
##   ScottDe   Struges
##  2.306995  2.320114
```

The average values are very much similar as we expected from Boxplot also.

#### 8.1.3.4 Integrated Absolute Loss :




From the Boxplot, it is clear that all methods are performing similarly except the *Kernel Density Adjusted* Struges Method and the Monte carlo estimate of Expected IABSL are -

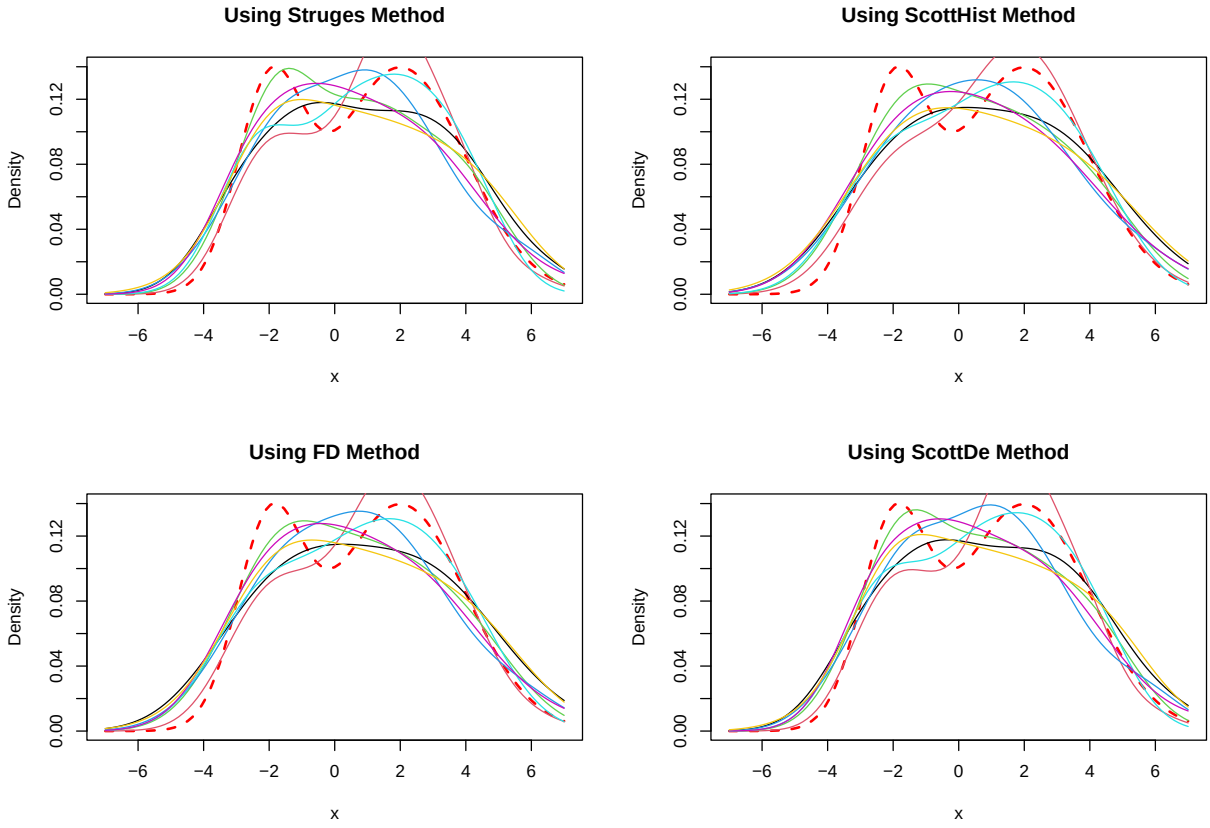
```
##        DPI        BCV         ML        UCV  Silverman         FD        AIC
## 0.04928623 0.04968857 0.05138084 0.05152009 0.05198458 0.05346006 0.05358931
##  ScottHist    ScottDe    Struges
## 0.05672905 0.05916232 0.10332408
```

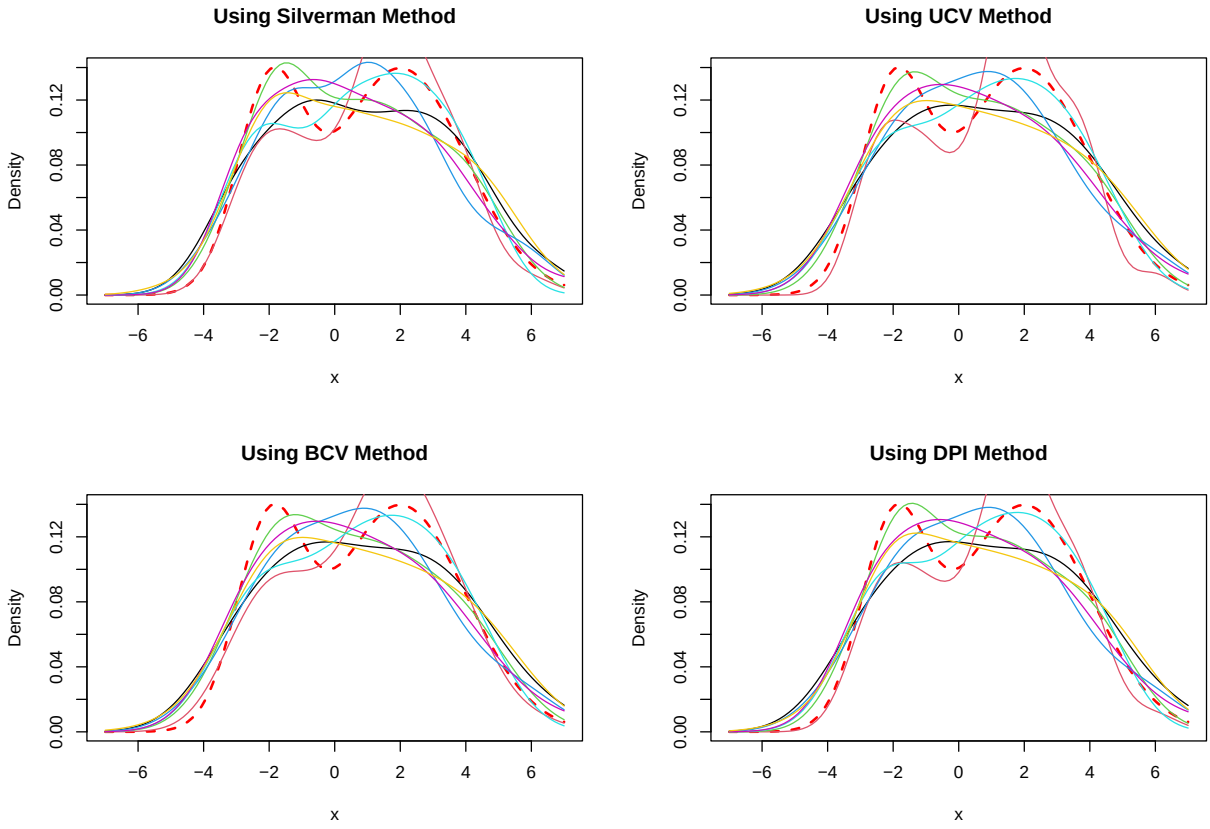
Here also we are getting very similar observations and the Monte carlo estimated expected IABSL values are not very different.

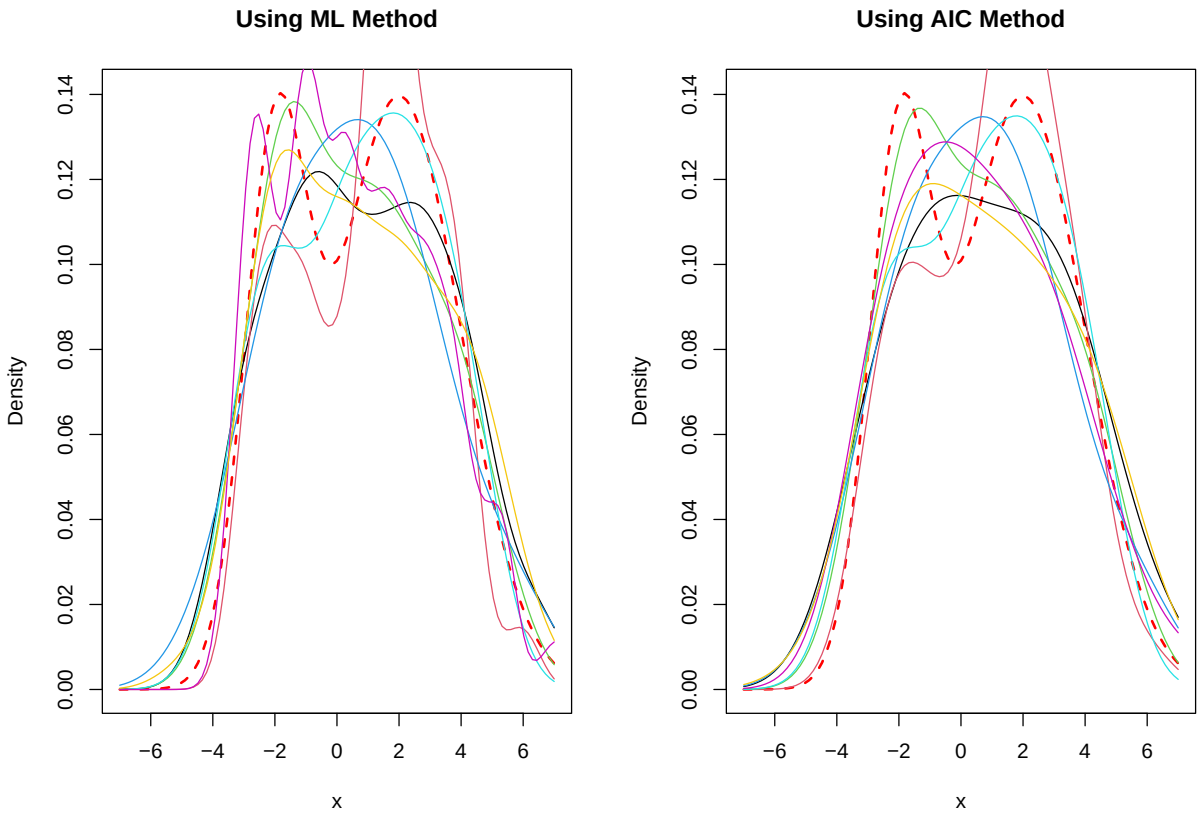
#### 8.1.4 Graphical Verification :

We should not entirely rely on Numbers only. Graphical verification is important. Here, we will draw samples from the above distribution (different instances) and using different methods, we will estimate the density in different instances and will compare them graphically.

**For n = 100**

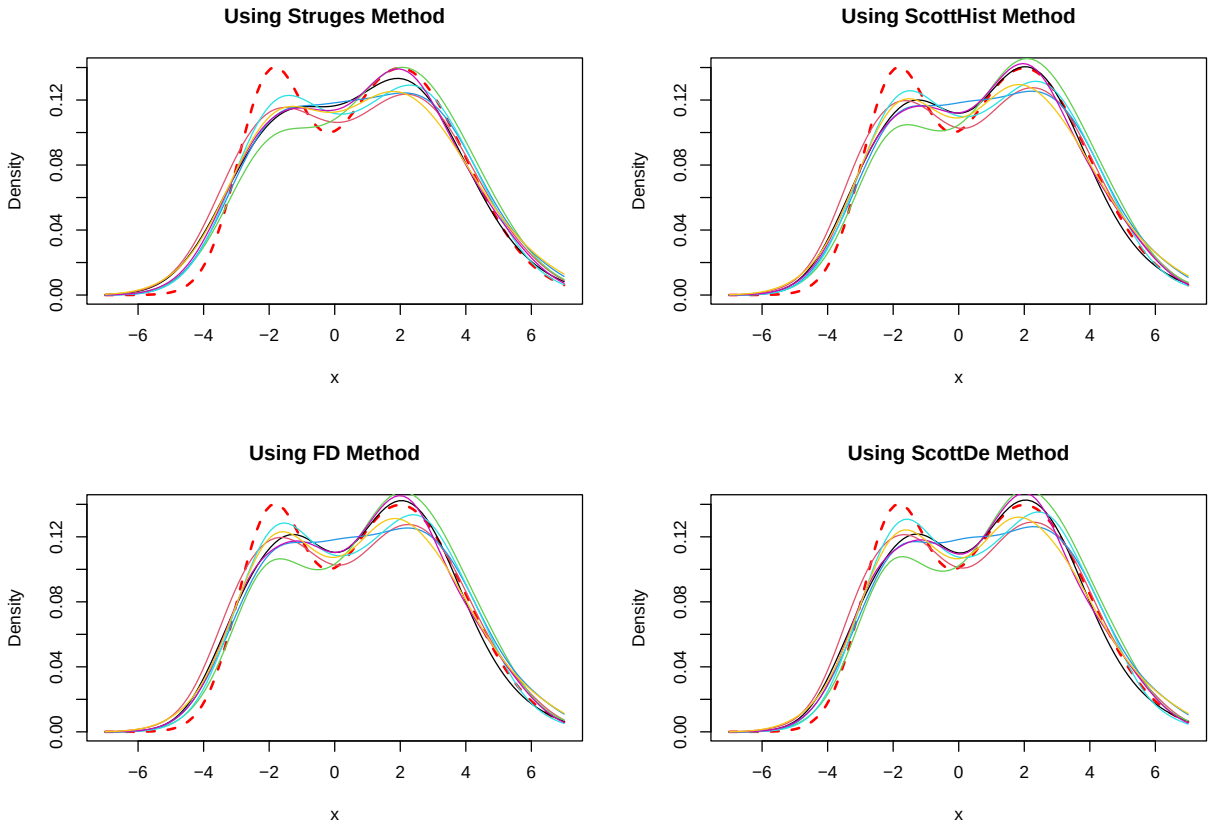

Using Struges Method
Using ScottHist Method
Using FD Method
Using ScottDe Method
Density
x
0.00
0.04
0.08
0.12
−6
−4
−2
0
2
4
6

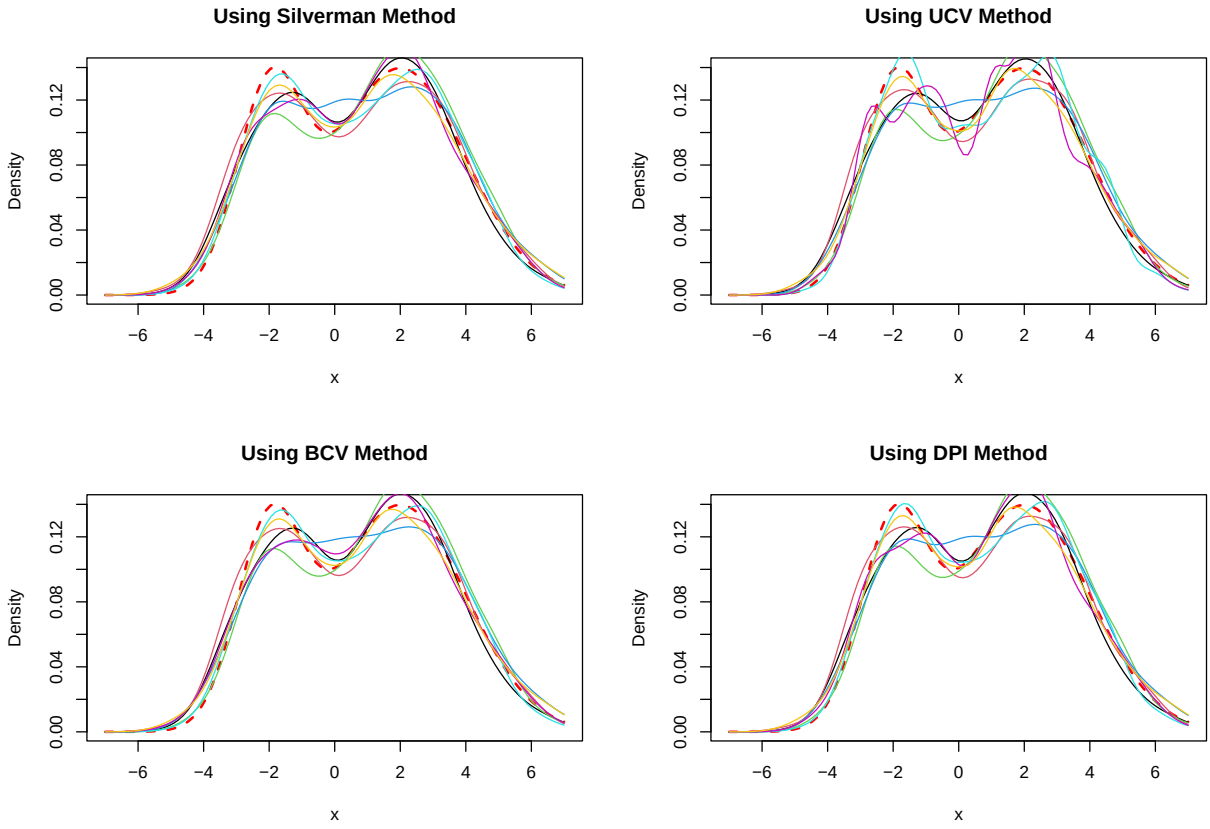

Using Silverman Method
Using UCV Method
Using BCV Method
Using DPI Method
Density
x
0.00
0.04
0.08
0.12
−6
−4
−2
0
2
4
6

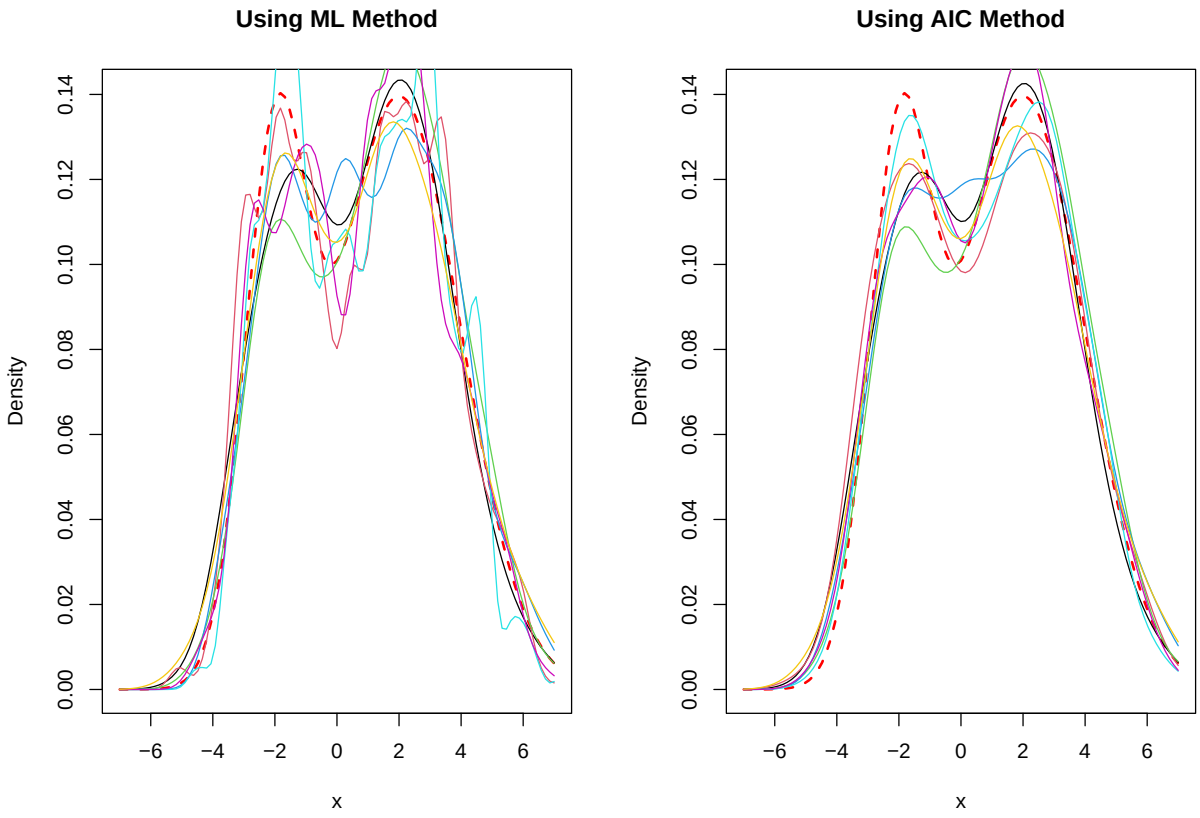



Here, none of the methods are performing that great visually. Histogram adjusted methods and Kernel Density methods are performing similarly. Most of the methods are estimating the bimodal distribution as unimodal distribution.

**For n = 500**

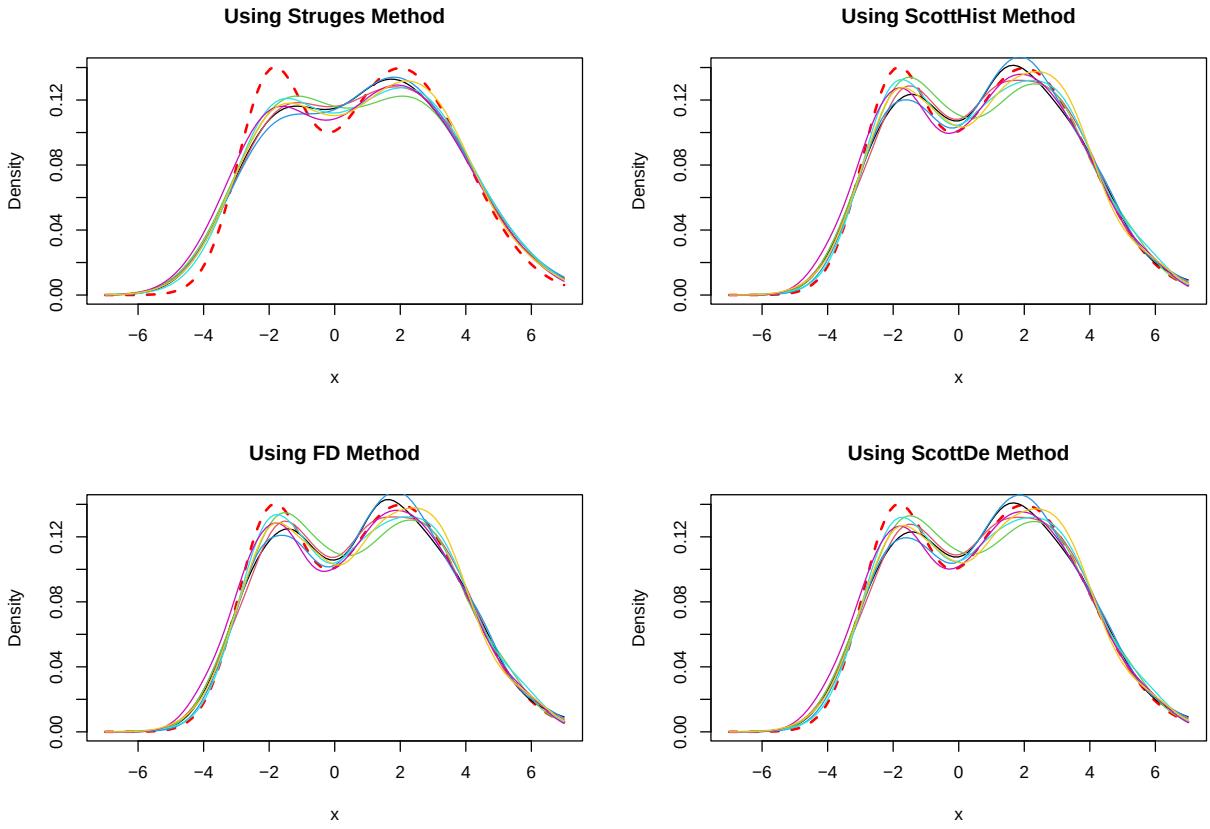

Using Struges Method
Using ScottHist Method
Using FD Method
Using ScottDe Method
Density
x
0.00
0.04
0.08
0.12
−6
−4
−2
0
2
4
6

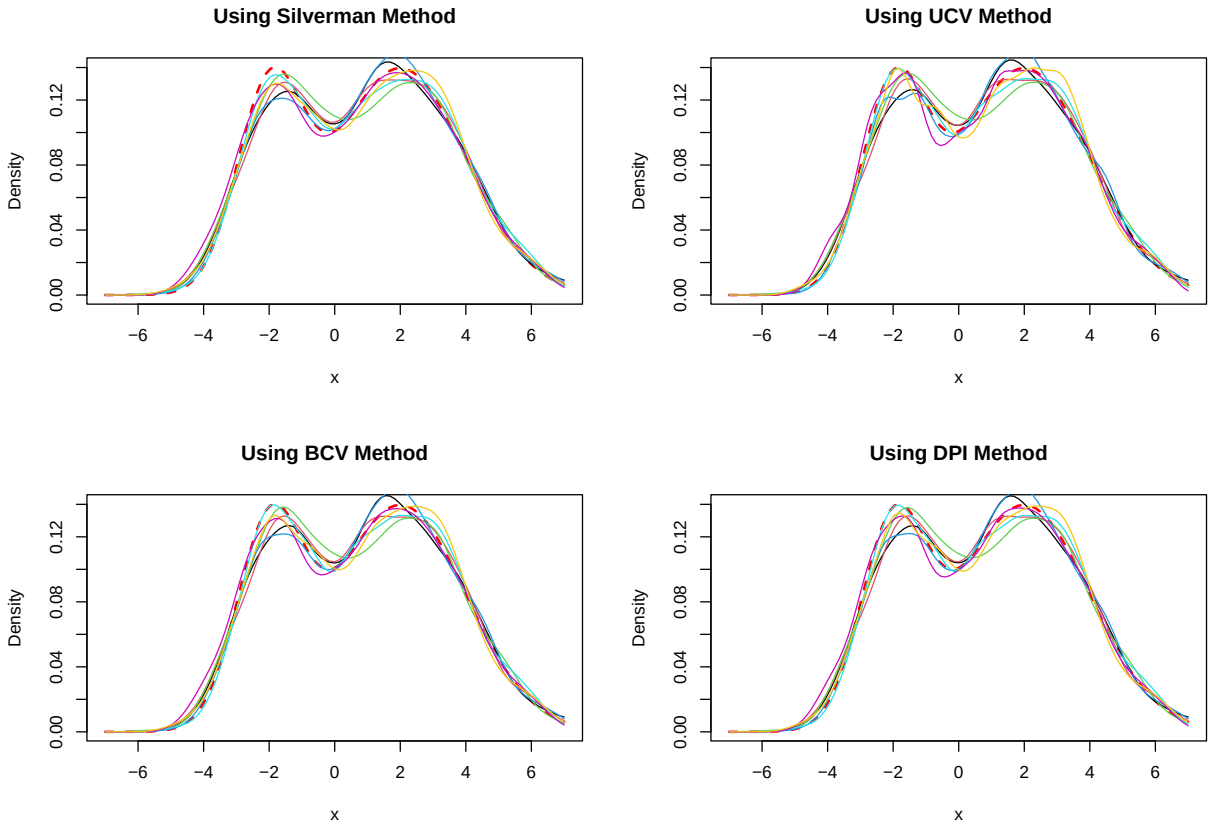

Using Silverman Method
Using UCV Method
Using BCV Method
Using DPI Method
Density
x
0.00
0.04
0.08
0.12
−6
−4
−2
0
2
4
6

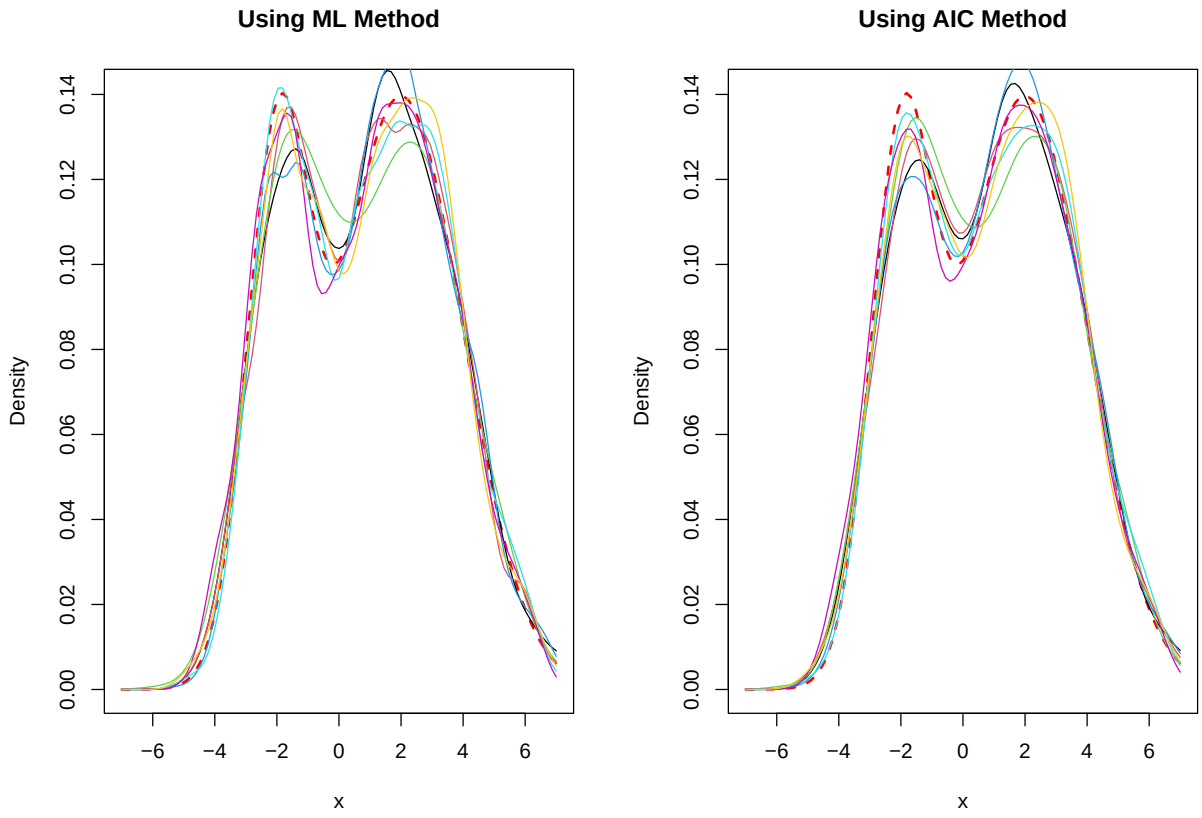



From the above plots, we can see that the performance of the methods have improved. There are some outlier instances for **UCV** and **ML**. **DPI,Silverman,BCV** and **AIC** are performing well.

**For n = 2000**


Using Struges Method
Using ScottHist Method
Using FD Method
Using ScottDe Method
Density
x
0.00
0.04
0.08
0.12
−6
−4
−2
0
2
4
6

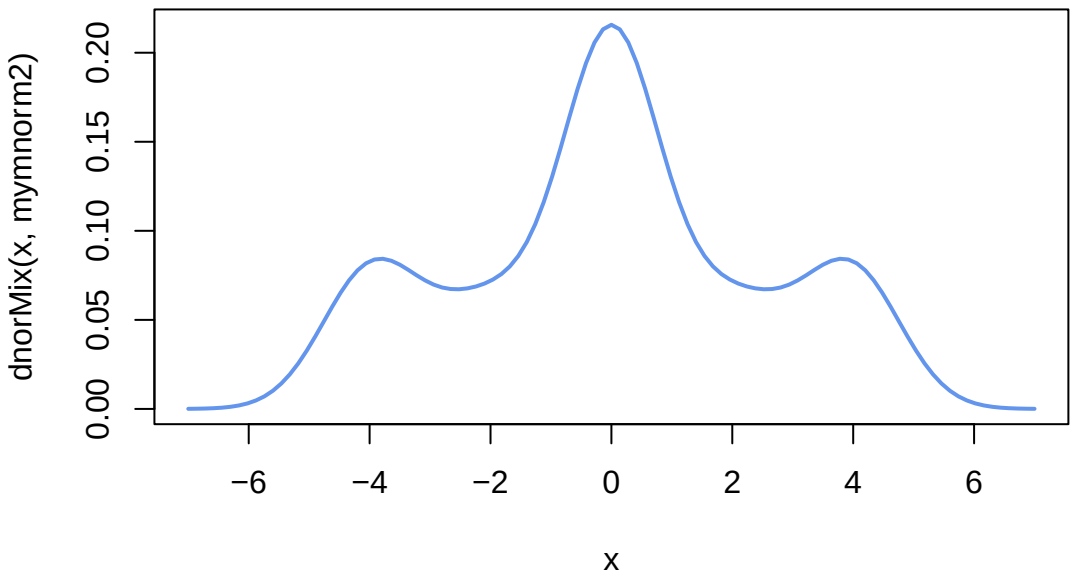

Using Silverman Method
Using UCV Method
Using BCV Method
Using DPI Method
Density
x
0.00
0.04
0.08
0.12
−6
−4
−2
0
2
4
6




Here all methods are performing well, except Struges method. We have seen these observations through numerical results also.

## 8.2 Simulation 2 :

In the last simulation we have considered a bimodal distribution. Now, we will consider a trimodal distribution.



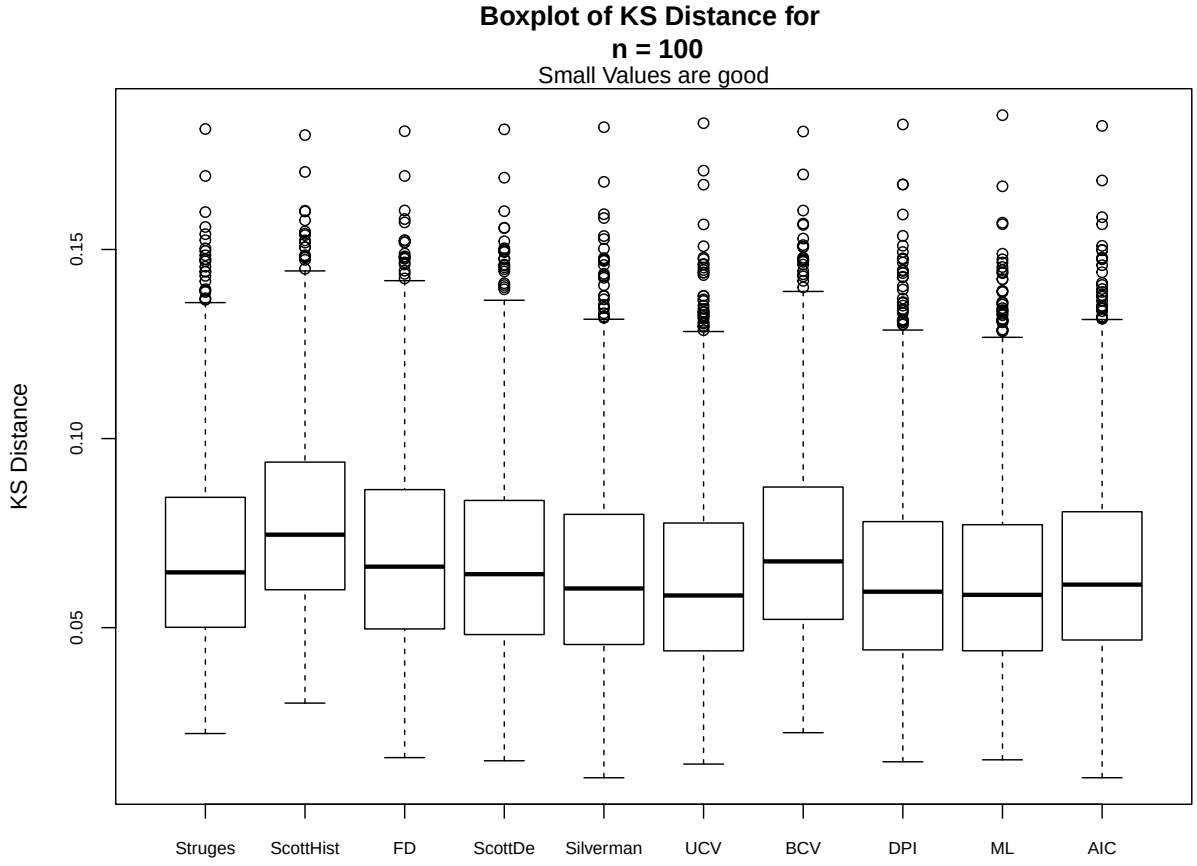

The pdf $f$ is given by -

$$f(x) = 0.15f_1(x) + 0.15f_2(x) + 0.4f_3(x) + 0.15f_4(x) + 0.15f_5(x)$$

Where, $f_1(.), f_2(.), f_3(.), f_4(.), f_5(.)$ are pdfs of $Normal(-4, 0.8^2)$,$Normal(-2, 1)$,$Normal(0, 0.8^2)$,$Normal(2, 1)$ and $Normal(4, 0.8^2)$ respectively. Intuitively the sd's 0.8 are reasons of the three peaks. For this distribution we will compare different methods for $n = 100, 500, 2000$

#### 8.2.1 For n = 100 :

##### 8.2.1.1 KS Distance :

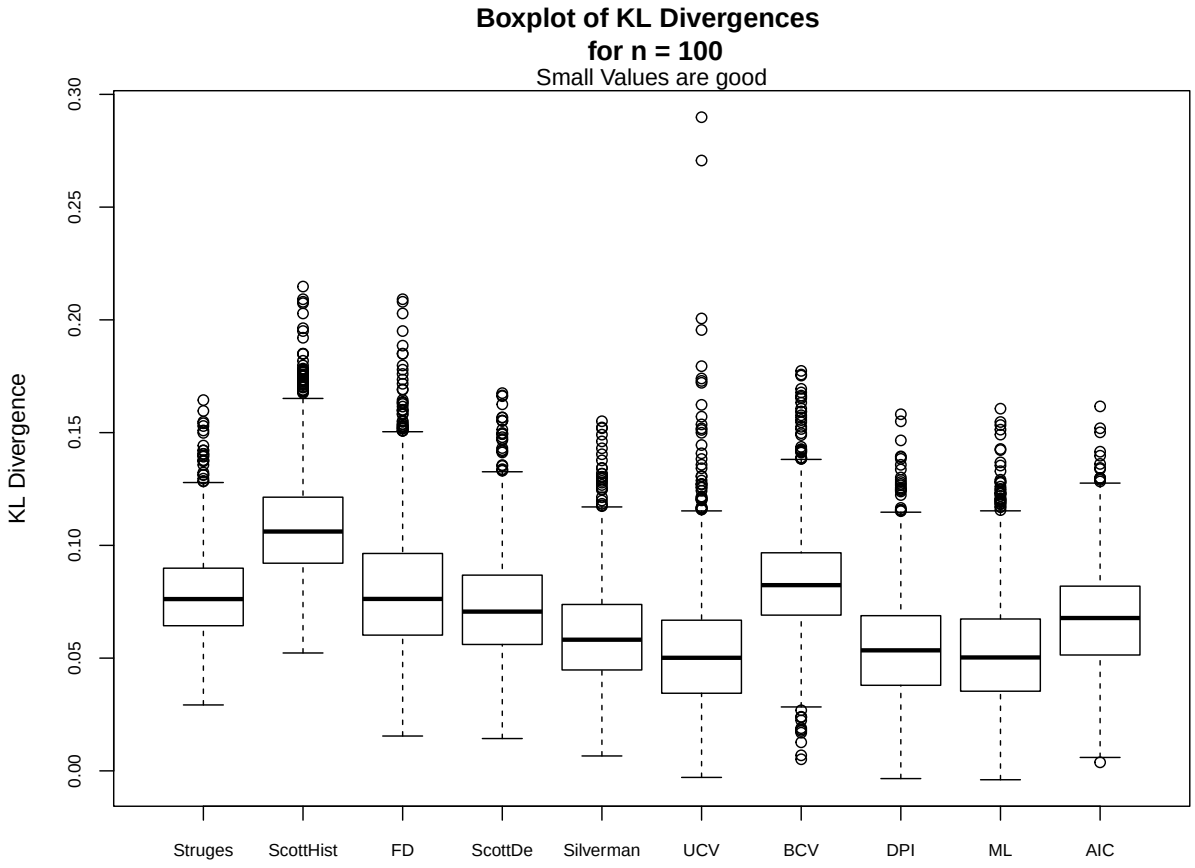



From the above Boxplot, we can see that Silverman,UCV,DPI,ML and AIC are performing similarly. Also, *Kernel Density Adjusted* Struges Method is giving similar type of results. Which is quite interesting. Now, let's look at the Monte carlo estimate of Expected KS Distances.

```
##         ML        UCV        DPI  Silverman        AIC    ScottDe    Struges
## 0.06250425 0.06287817 0.06323357 0.06449405 0.06565284 0.06774635 0.06876770
##         FD        BCV  ScottHist
## 0.06967950 0.07100406 0.07789594
```

The Expected KS Distance Values are depicting the same thing. Interestingly the **ML** method has least Expected KS Distance(Monte Carlo Estimated !).

#### 8.2.1.2 KL Divergence :

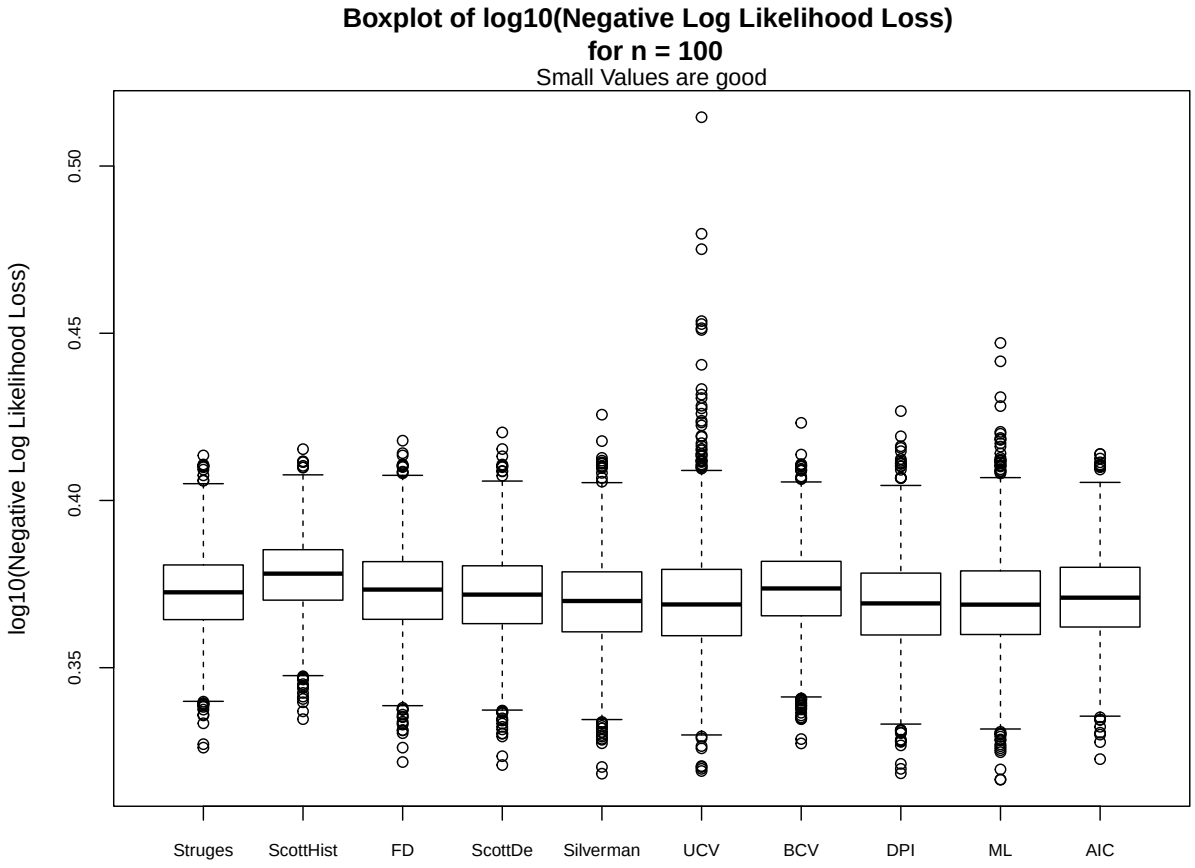



From the above Boxplot we can see that, DPI and ML are performing similarly. UCV method has very large outliers. On the other hand BCV method has some small outliers. Which is very interesting. Also, *Kernel Density Adjusted* Struges method and FD methods are also working good. Now, let's look at the Monte carlo estimate of Expected KL Divergences.

```
##         ML        UCV        DPI  Silverman        AIC    ScottDe    Struges
## 0.05278066 0.05293540 0.05495659 0.06024603 0.06745289 0.07267648 0.07794310
##         FD        BCV  ScottHist
## 0.08036131 0.08426093 0.10876692
```

The Expected KL Divergence values are also depicting the same thing. Also, the ML method has least Expected KL Divergence. Also, note that we are getting similar type of ordering as we get in case of **KS Distance**.

#### 8.2.1.3 Negative Log Likelihood Loss :



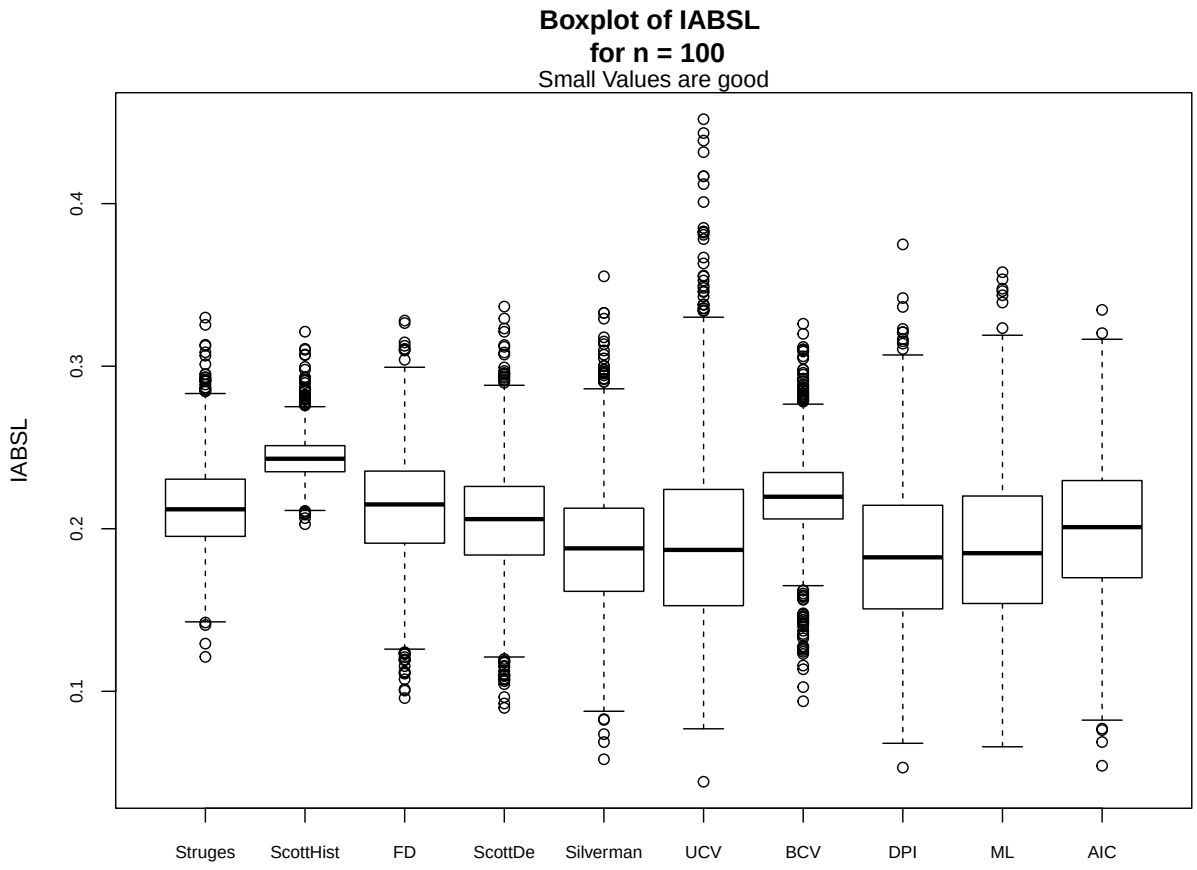

From the Boxplot, we can say that all methods are performing very similarly. Also, UCV has many outliers. We have seen these observations in case of **Simulation - 1** also. Note that this Boxplot of log10(Negative Log Likelihood Loss).

The Average values of Negative Log Likelihood Loss are -

```
##       DPI        ML Silverman       UCV       AIC   ScottDe   Struges        FD
##  2.340982  2.343060  2.344113  2.345682  2.349906  2.354027  2.358431  2.360646
##       BCV ScottHist
##  2.364261  2.386184
```

The Average values are also not very different, which is very expected.

#### 8.2.1.4 Integrated Absolute Loss :



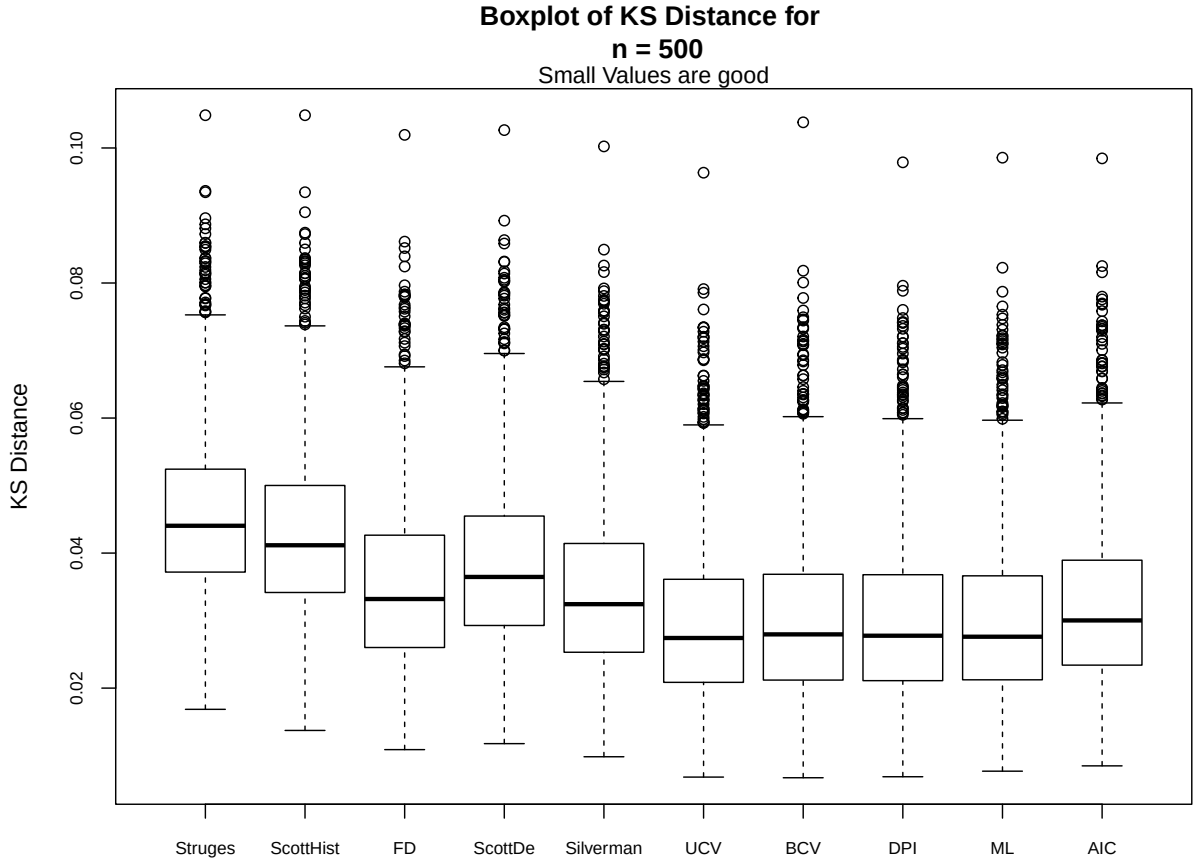

From the above Boxplot, it is clear that ML, Silverman and DPI are performing similarly. Silverman method has some small outliers. Now, let's look at the Monte carlo estimate of Expected IABSL.

```
##       DPI        ML Silverman       UCV       AIC   ScottDe        FD   Struges
## 0.1830541 0.1870189 0.1882873 0.1910438 0.1985398 0.2041839 0.2120718 0.2132995
##       BCV ScottHist
## 0.2209187 0.2437759
```

The Expected IABSL are depicting the same thing. DPI method has least expected IABSL (Monte Carlo Estimated !)

### 8.2.2 For n = 500 :

#### 8.2.2.1 KS Distance :

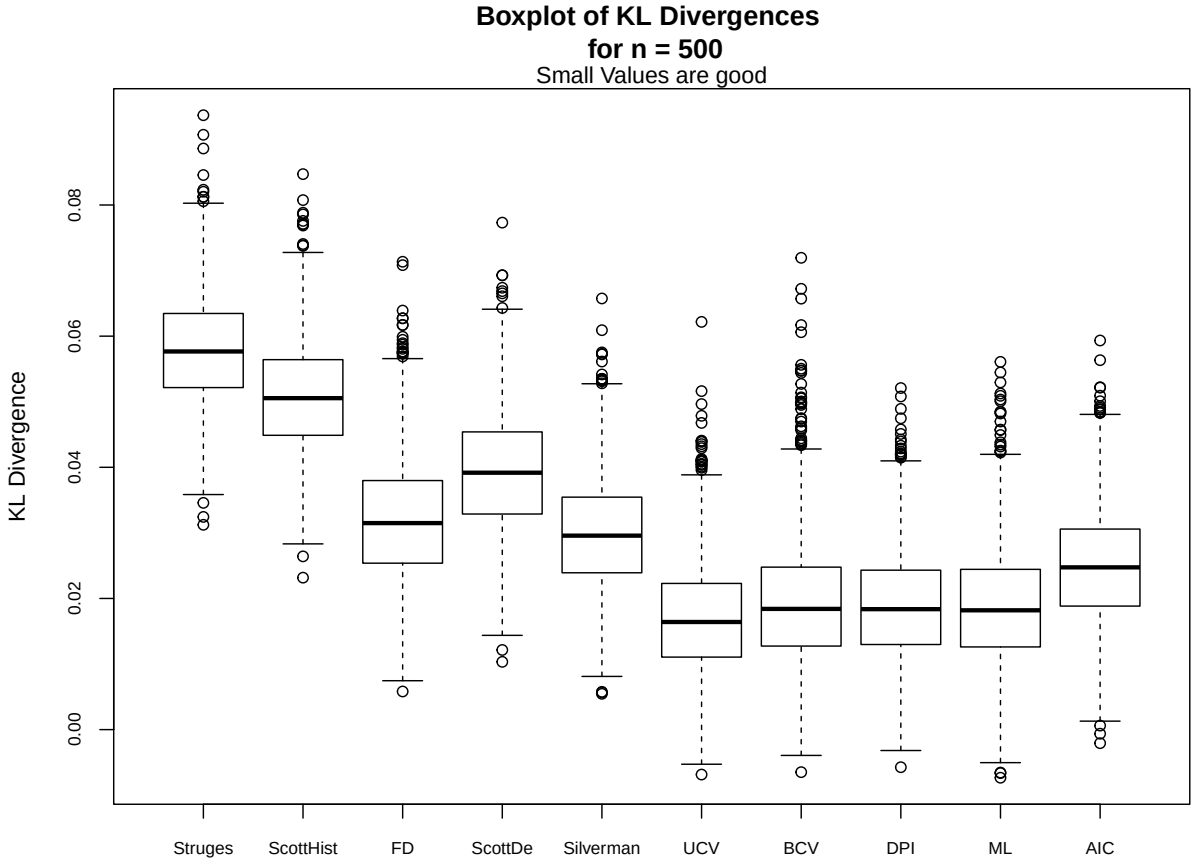



From the above Boxplot, we can see that last five methods are performing similarly. Interestingly, the *Kernel Density Adjusted* Struges method not performing well now. Let's see the Monte carlo estimate of Expected KS Distances.

```
##        UCV         ML        DPI        BCV        AIC  Silverman         FD
## 0.02952656 0.02985679 0.02986875 0.03005313 0.03196273 0.03432499 0.03516759
##    ScottDe  ScottHist    Struges
## 0.03822885 0.04279844 0.04556133
```

The Expected KS Distance values are depicting same thing. Notice that, **UCV** method has least estimated KS Distance in this case.

#### 8.2.2.2 KL Divergence :

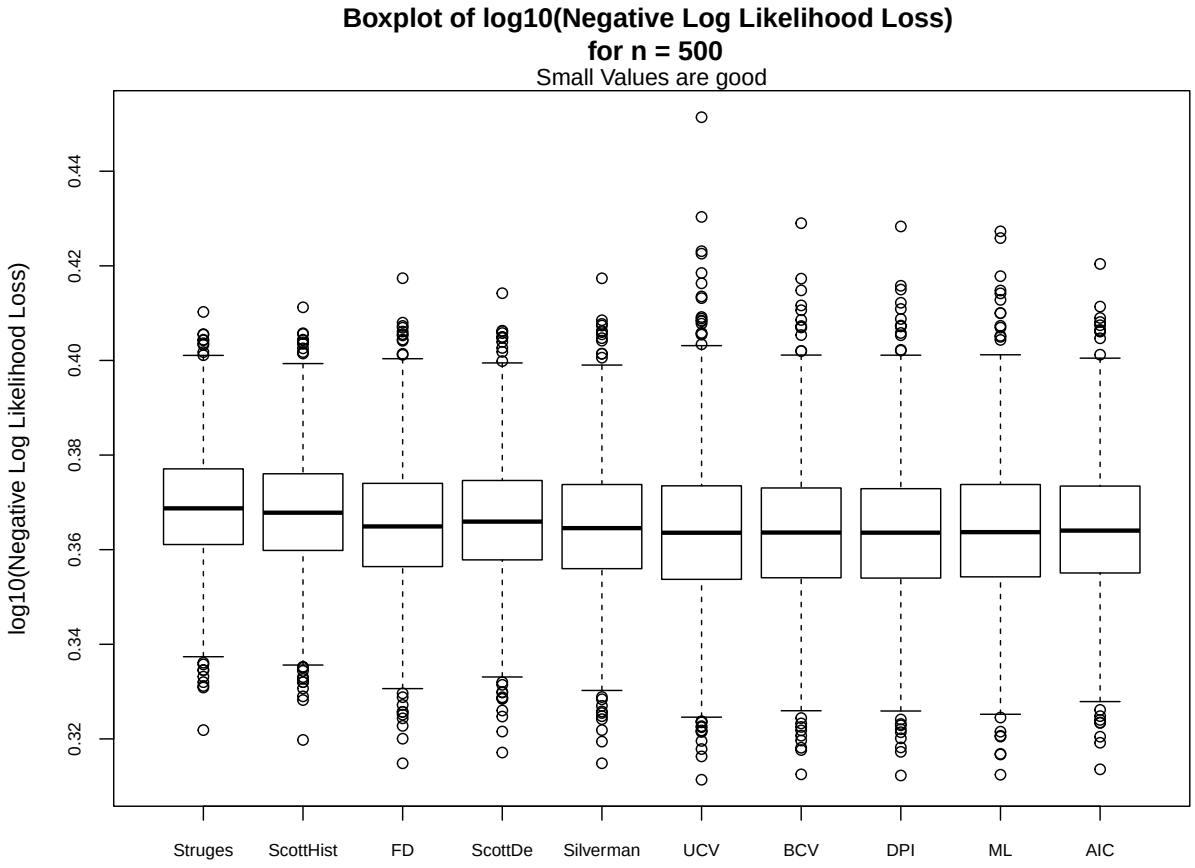



From the Boxplot, we can see that UCV,BCV,DPI and ML are performing similarly. AIC has some small as well as large outliers. Let's look at the Monte carlo estimate of Expected KL Divergences.

```
##        UCV         ML        DPI        BCV        AIC  Silverman         FD
## 0.01702644 0.01882366 0.01888531 0.01932031 0.02495131 0.02994977 0.03195645
##    ScottDe  ScottHist    Struges
## 0.03935066 0.05089337 0.05792035
```

The Expected KL Divergences are depicting the same thing. **UCV** has least estimated KL Distance. Interestingly, we are having similar observation as we have in case of KS Distance.

#### 8.2.2.3 Negative Log Likelihood Loss :



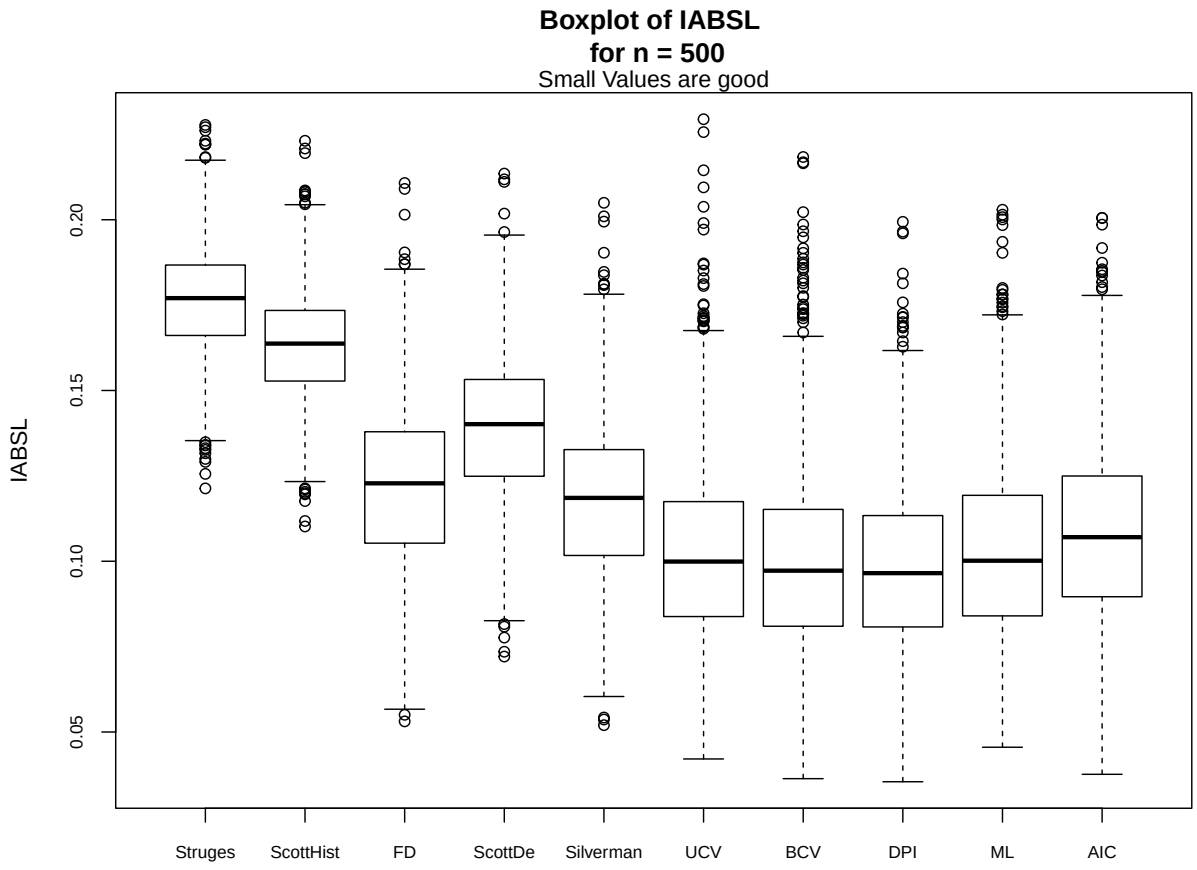

From the above Boxplot, all methods are performing similarly. UCV has some outliers. We have seen this observation in other cases also and this is really interesting.

The Average values of Negative Log-Likelihood are -

```
##       DPI       BCV       UCV        ML       AIC Silverman        FD   ScottDe
##  2.311911  2.312333  2.312716  2.313323  2.314937  2.318097  2.319526  2.325051
## ScottHist   Struges
##  2.334352  2.340304
```

The average values are also not very different.

#### 8.2.2.4 Integrated Absolute Loss :



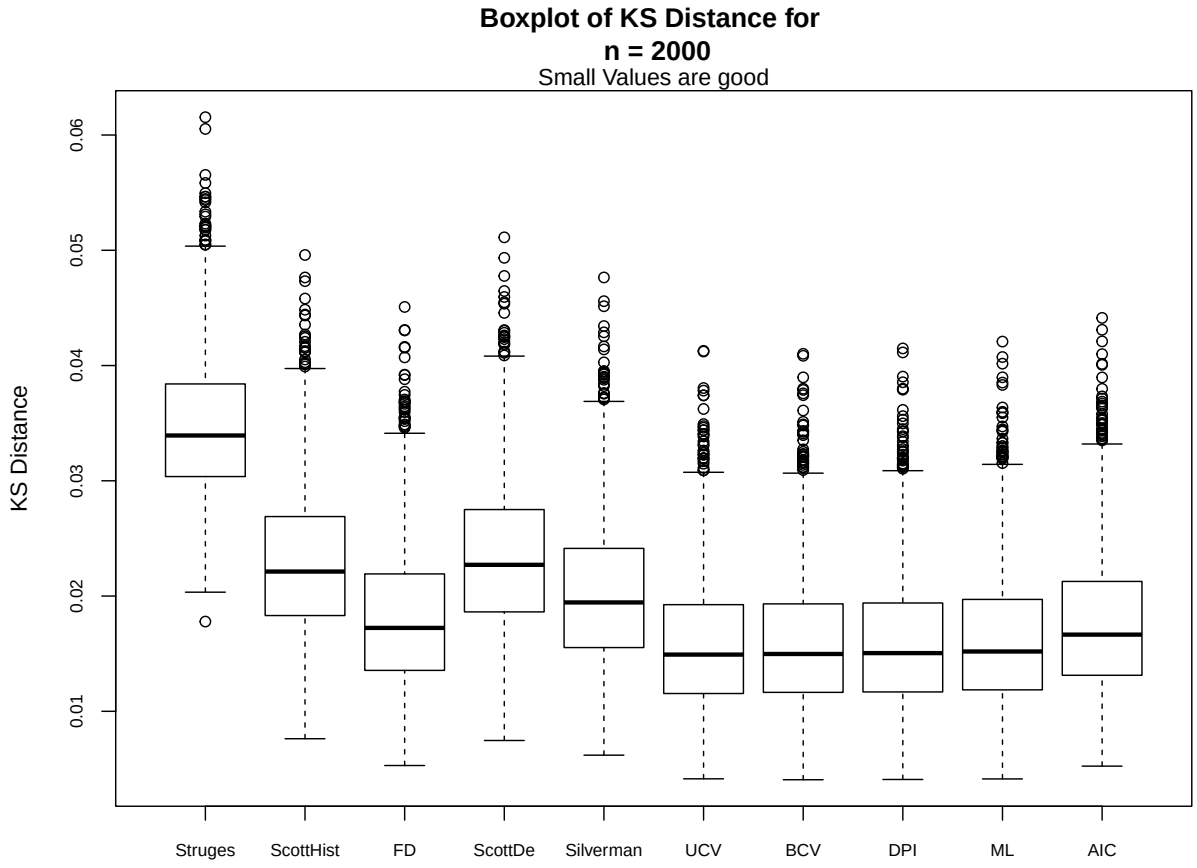

From the Boxplot, we can see that DPI and BCV are performing similarly and the *Kernel Density Adjusted* methods are performing quite bad. Let's see the Monte carlo estimate of Expected IABSL.

```
##        DPI        BCV        UCV         ML        AIC  Silverman         FD
## 0.09808858 0.09956038 0.10195085 0.10249757 0.10836978 0.11812679 0.12238323
##    ScottDe  ScottHist    Struges
## 0.13897888 0.16344963 0.17668936
```

The Expected IABSL are also depicting same thing. Also, DPI has the least expected IABSL.

### 8.2.3 For n = 2000 :

#### 8.2.3.1 KS Distance :

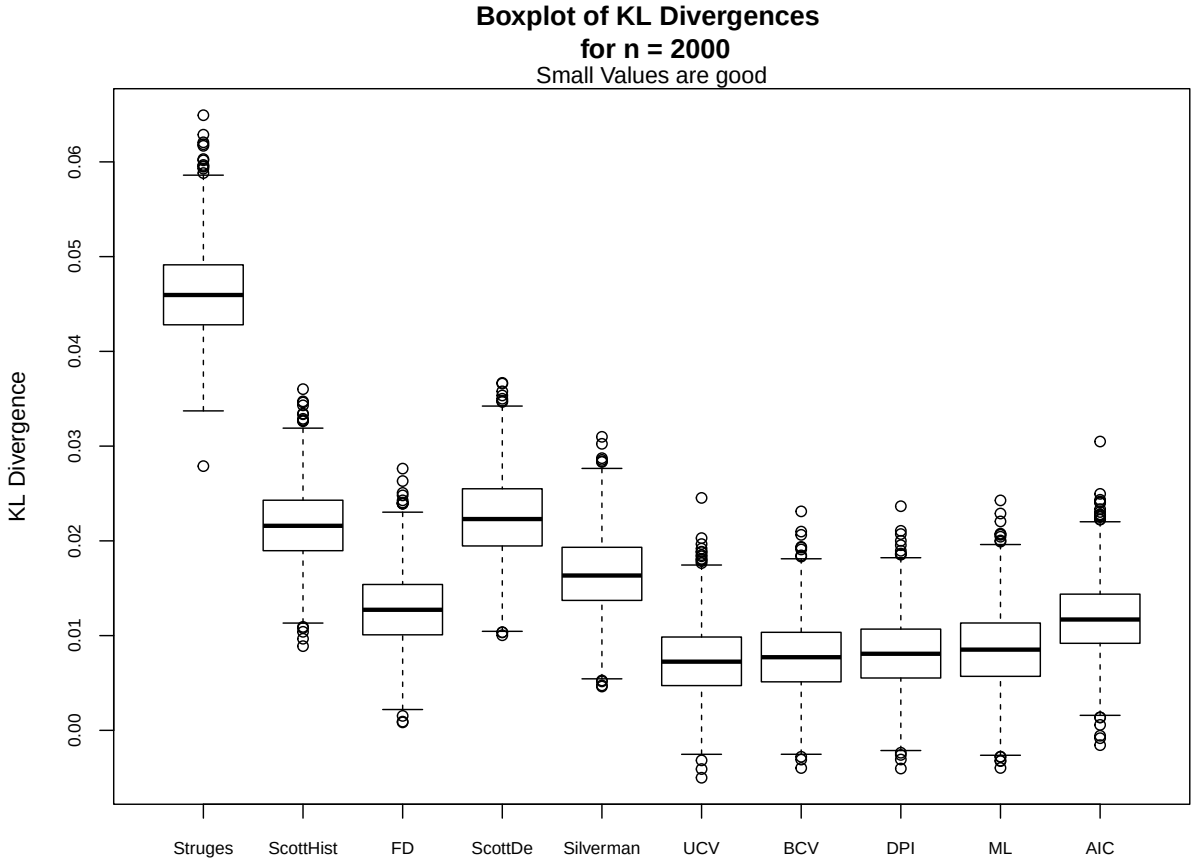



From the Boxplot, we can see that last five methods and *Kernel Density Adjusted* FD method are performing similarly. Let's look at the Monte carlo estimate of Expected KS Distances.

```
##        UCV        BCV        DPI         ML        AIC         FD  Silverman
## 0.01588321 0.01593487 0.01604073 0.01624200 0.01761462 0.01819103 0.02020716
##  ScottHist    ScottDe    Struges
## 0.02298491 0.02342928 0.03475277
```

The Expected KS Distance values are also depicting the same thing. Note that, **UCV** has least Expected KS Distance. (Monte Carlo Estimated !)

#### 8.2.3.2 KL Divergence :

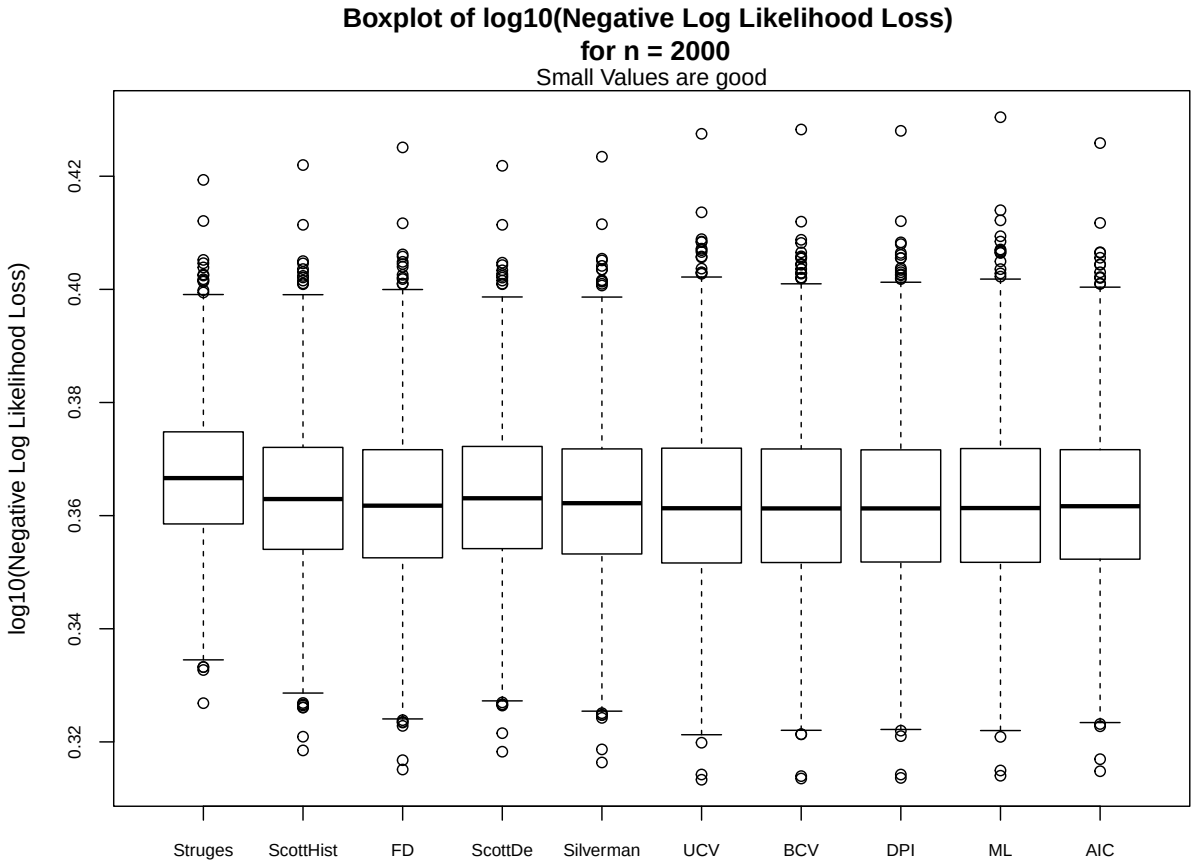



From the above Boxplot, it is clear that UCV,BCV,DPI and ML are performing similarly. The *Kernel Density Adjusted* FD is performing quite well. While Struges Method is performing very bad. Let's see the Monte carlo estimate of Expected KL Divergences.

```
##         UCV         BCV         DPI          ML         AIC          FD
## 0.007319081 0.007806641 0.008161923 0.008566354 0.011839937 0.012800559
##   Silverman   ScottHist     ScottDe     Struges
## 0.016532449 0.021683792 0.022479902 0.046103619
```

The Expected KL Divergence also depicting same thing and **UCV** has least Expected KL Divergence.

#### 8.2.3.3 Negative Log Likelihood Loss :

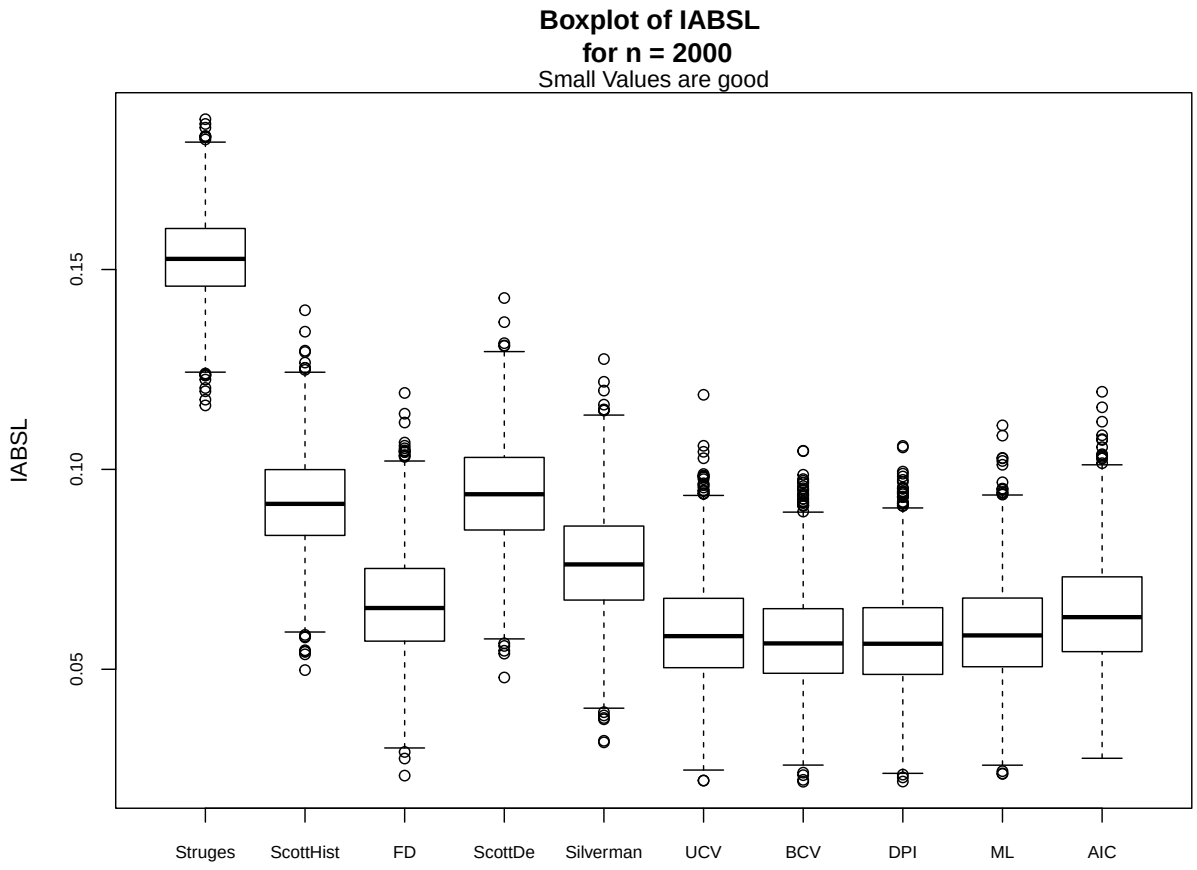



From the above Boxplot it is clear that, all methods are performing similarly.

The Average values of Negative Log Likelihood Loss are -

```
##       DPI       BCV       UCV        ML       AIC        FD Silverman ScottHist
##  2.302159  2.302161  2.302416  2.302584  2.303282  2.303639  2.305582  2.308783
##   ScottDe   Struges
##  2.309315  2.327728
```

#### 8.2.3.4 Integrated Absolute Loss :




From the above Boxplot, we can see that last five methods and *Kernel Density Adjusted* FD method are performing quite similarly. Let's see the Monte Carlo Estimate of Expected IABSL.

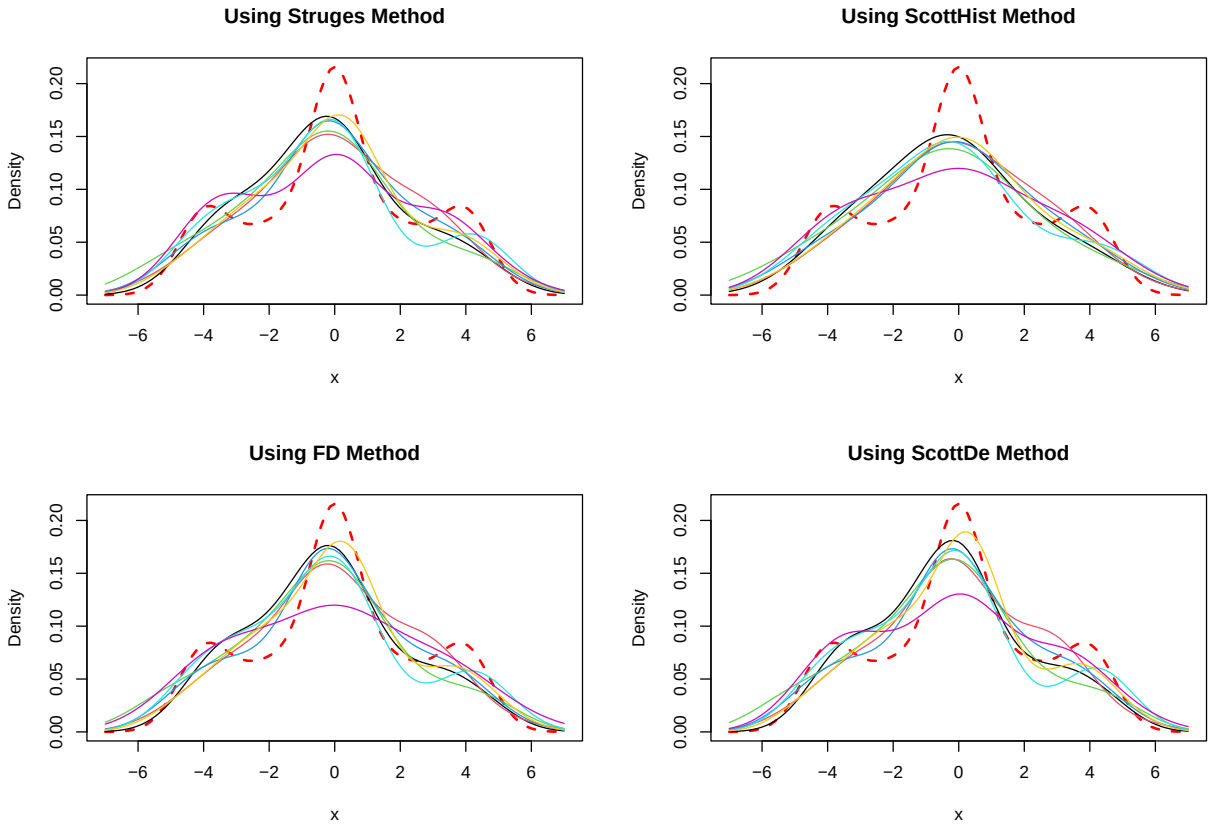
```
##        BCV        DPI         ML        UCV        AIC         FD  Silverman
## 0.05739064 0.05739459 0.05946271 0.05946911 0.06398986 0.06609509 0.07656732
##  ScottHist    ScottDe    Struges
## 0.09171546 0.09390641 0.15279947
```

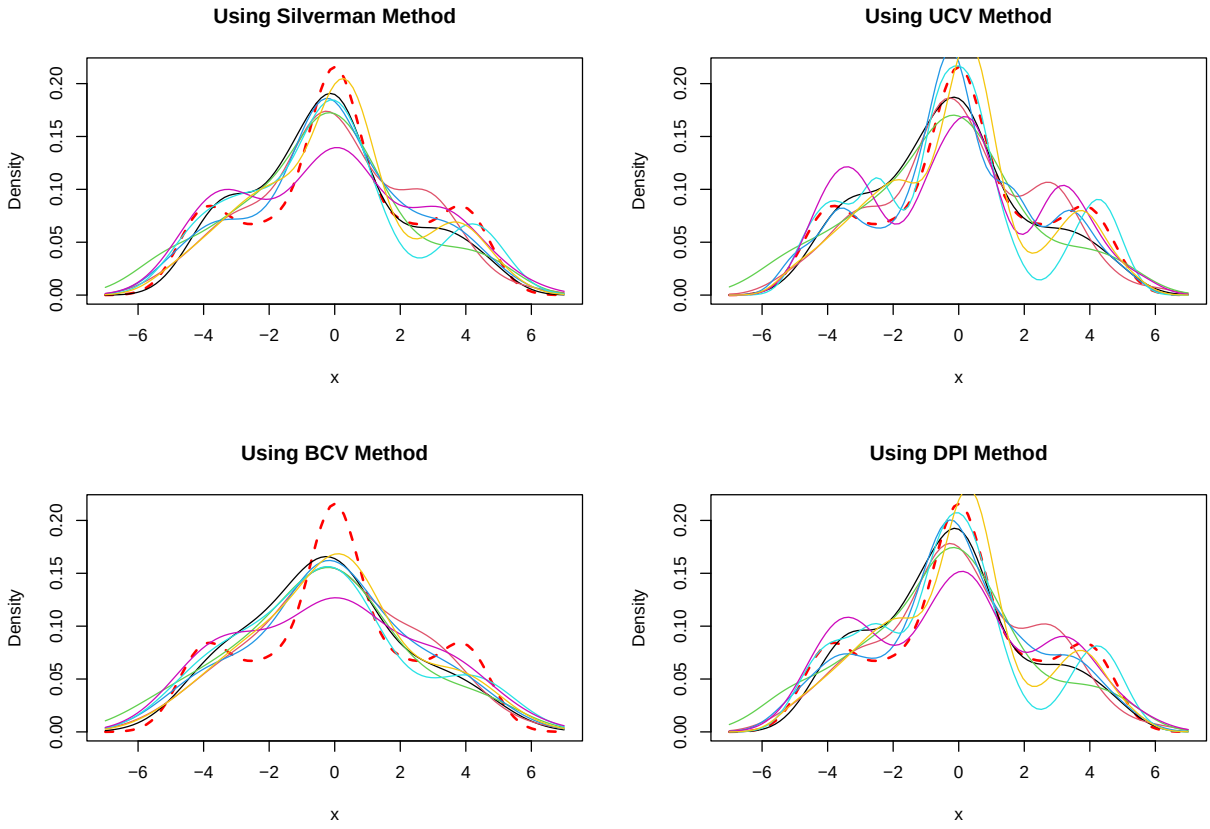
The Expected IABSL values are also depicting the same thing and **BCV** has least expected IABSL.

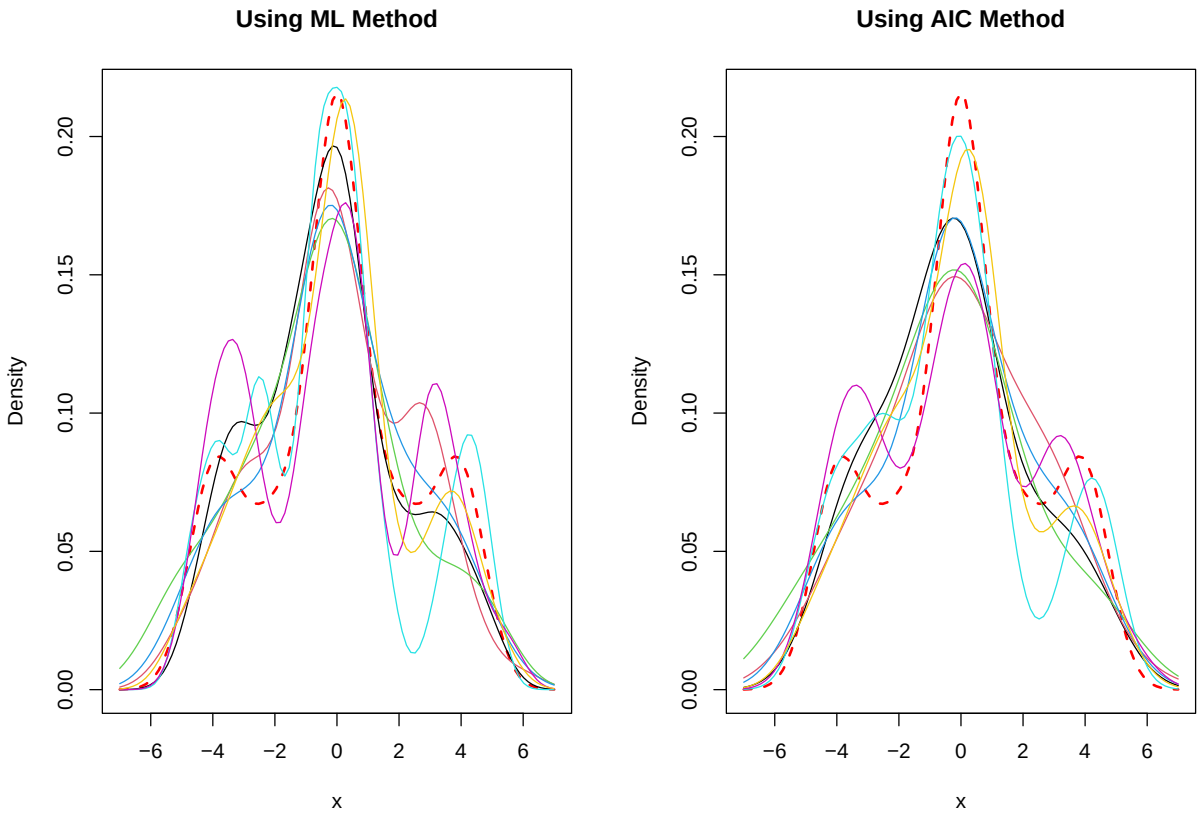
### 8.2.4 Graphical Verification :

As we have seen graphical verification is important. So, we have done similar thing in this case also.

**For n = 100**

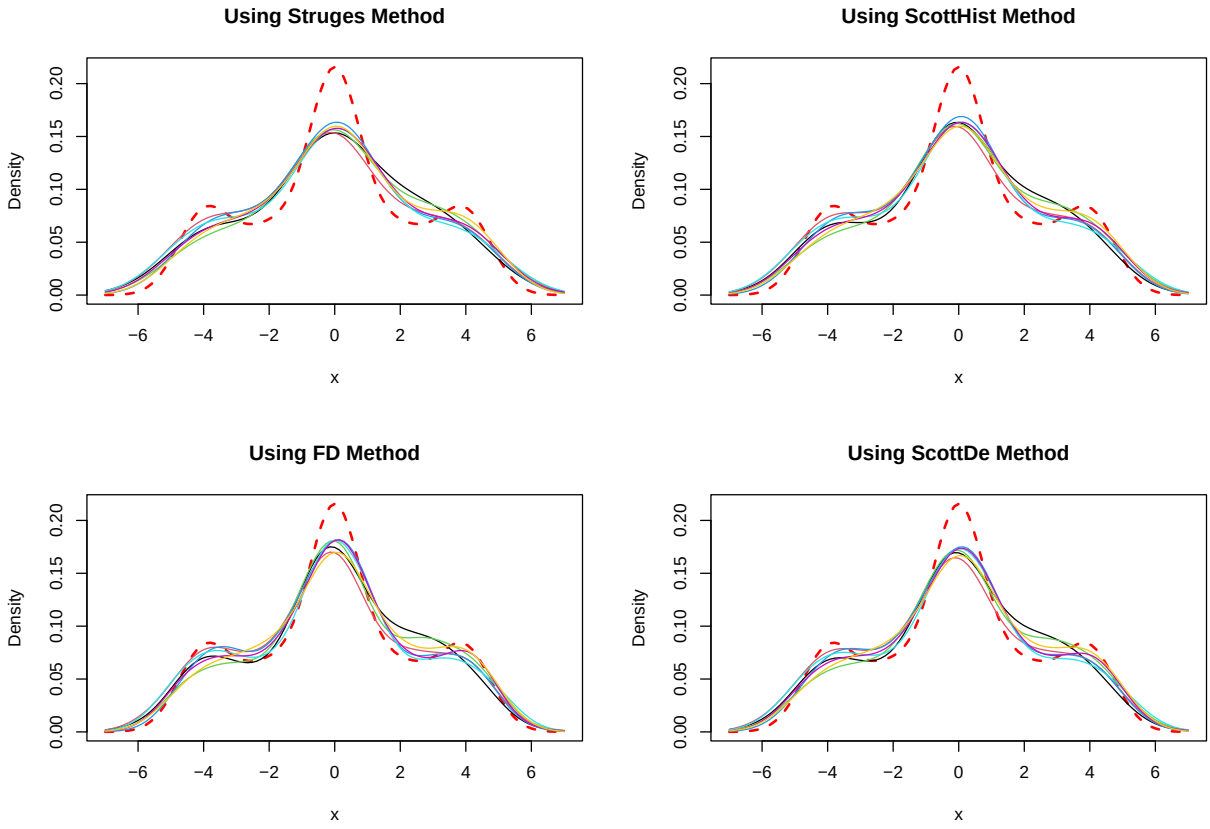

Using Struges Method
Using ScottHist Method
Using FD Method
Using ScottDe Method
Density
x
0.00
0.05
0.10
0.15
0.20
−6
−4
−2
0
2
4
6

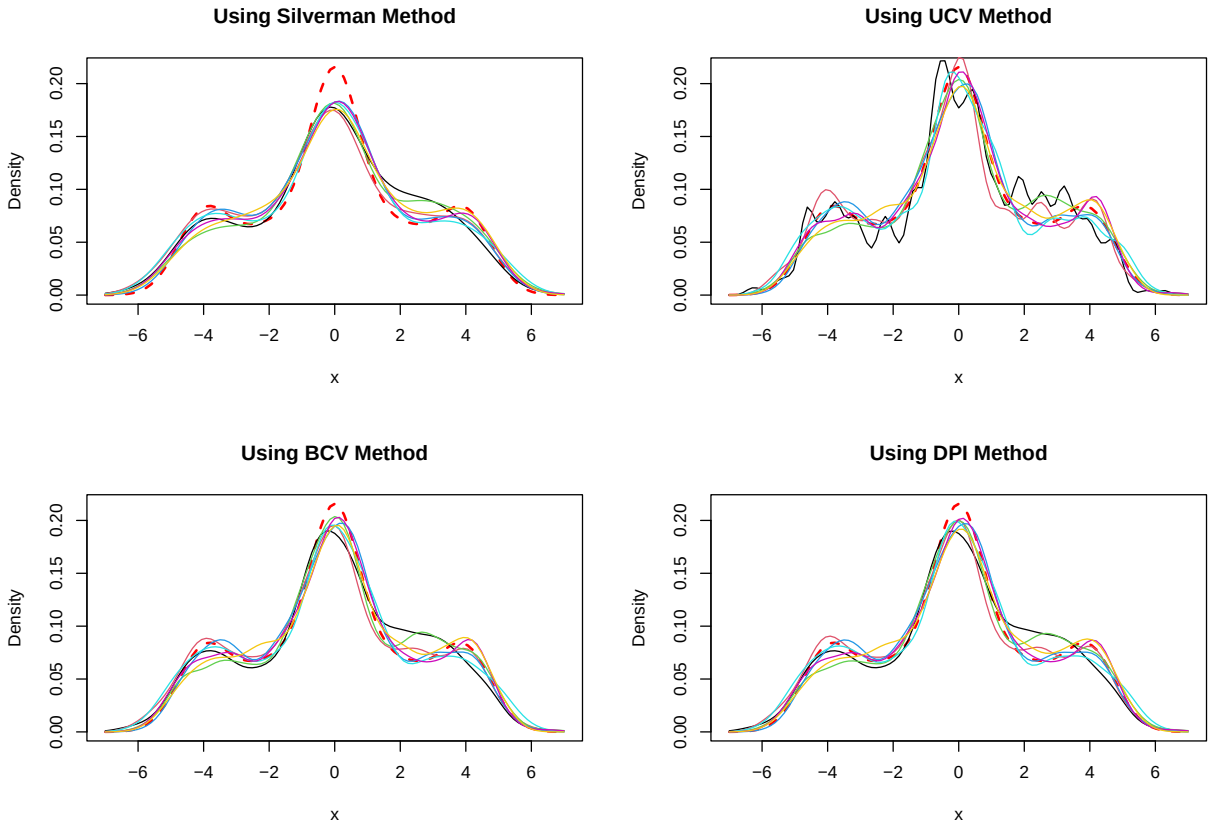

Using Silverman Method
Using UCV Method
Using BCV Method
Using DPI Method
Density
x
0.00
0.05
0.10
0.15
0.20
−6
−4
−2
0
2
4
6

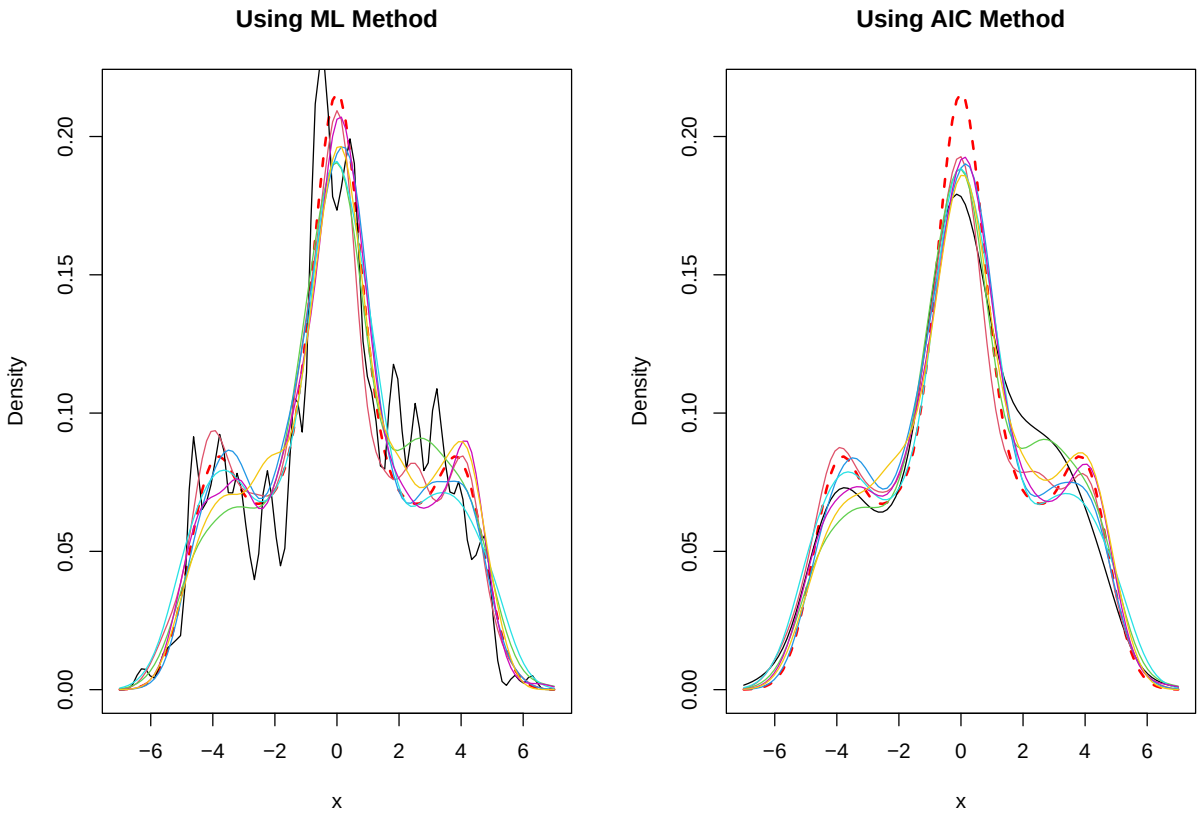



The numerical observations from simulation result are matching with the above plots. Interestingly most of the methods are not able to estimate the maximum height of the density. While **UCV,Silverman,ML,DPI** are able to catch that to some extent. **ScottHist** is even unable to catch the all modes of the distribution.

**For n = 500**

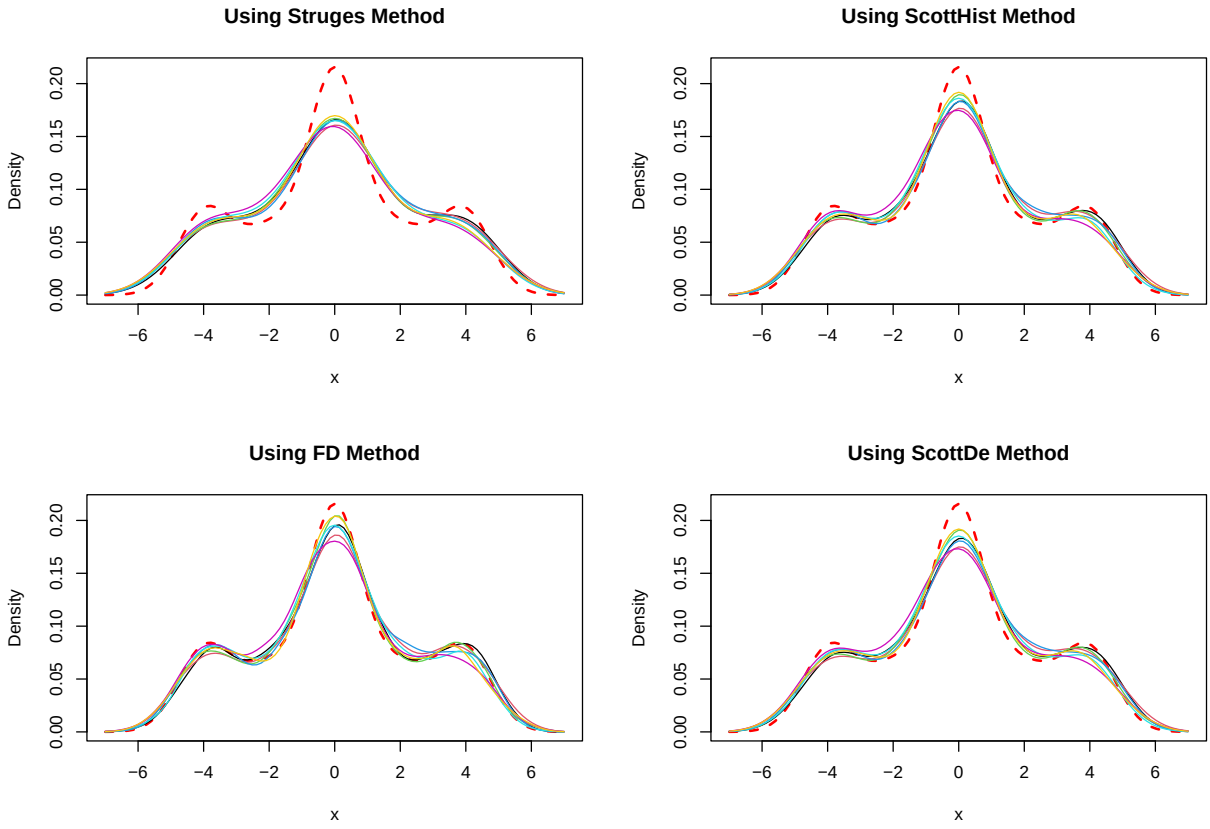

Using Struges Method
Using ScottHist Method
Using FD Method
Using ScottDe Method
Density
x
0.00
0.05
0.10
0.15
0.20
−6
−4
−2
0
2
4
6

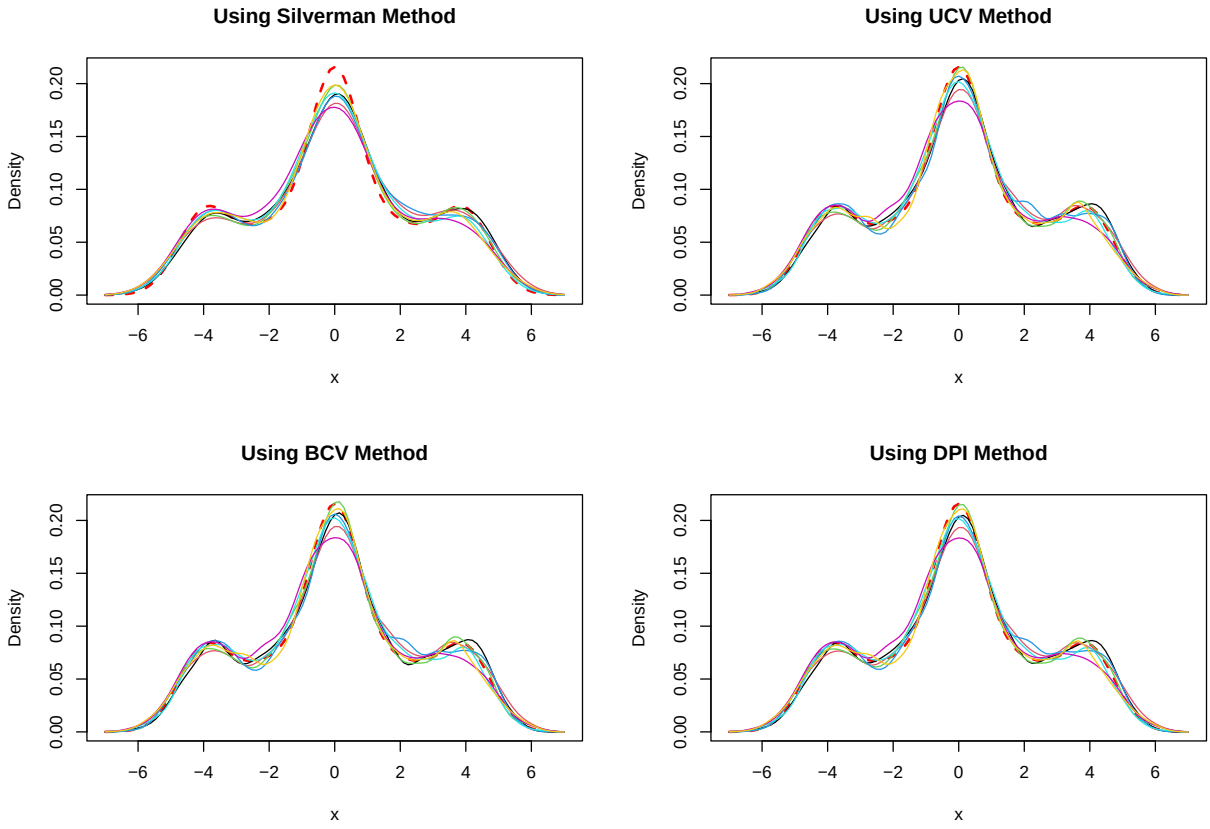

Using Silverman Method
Using UCV Method
Using BCV Method
Using DPI Method
Density
x
0.00
0.05
0.10
0.15
0.20
−6
−4
−2
0
2
4
6

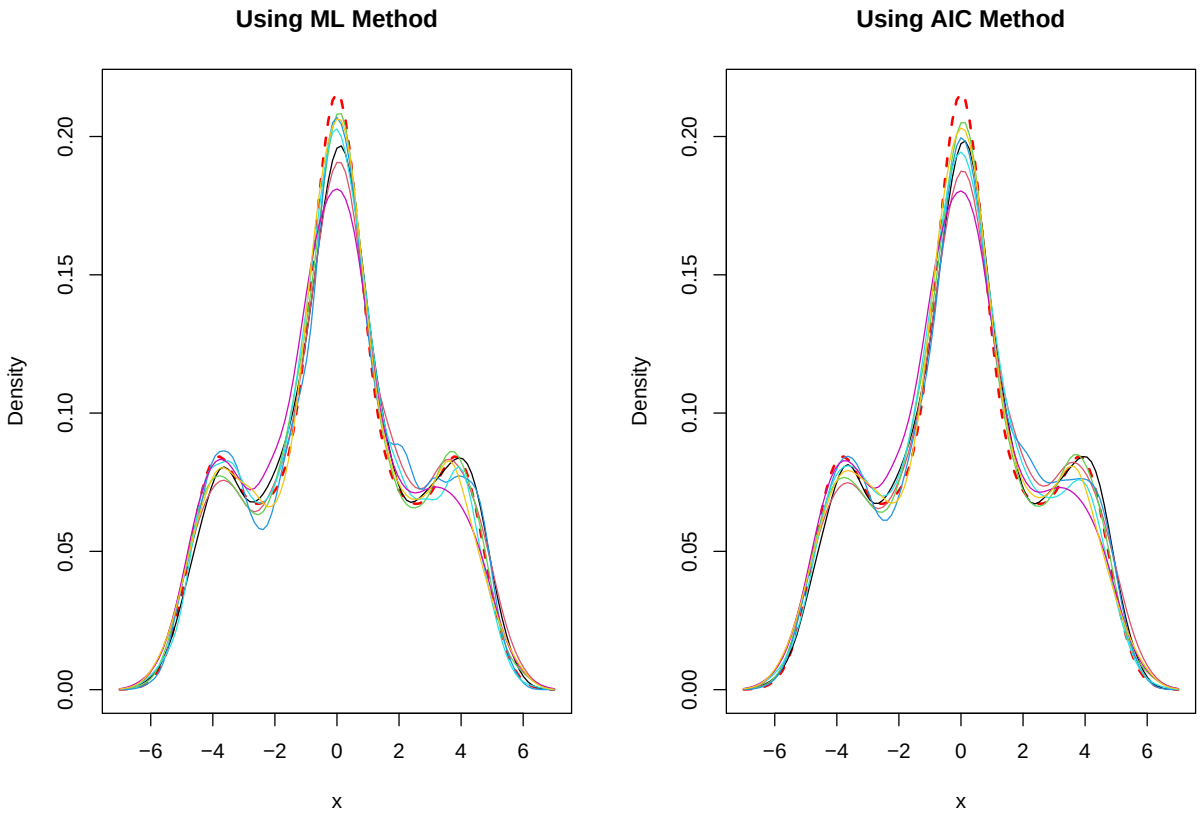



Interestingly, **ML** and **UCV** are sometimes working bad. These are the outlier instances. But they are important. **DPI** and **BCV** are performing better. (Although they can have some outlier instances).

**For n = 2000**


Using Struges Method
Using ScottHist Method
Using FD Method
Using ScottDe Method
Density
x
0.00
0.05
0.10
0.15
0.20
−6
−4
−2
0
2
4
6

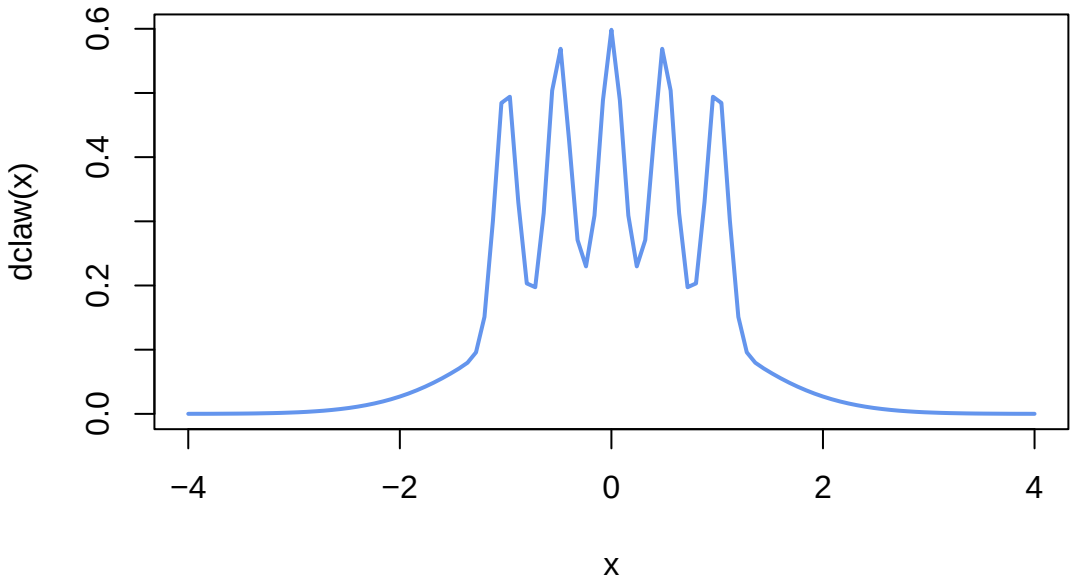

Using Silverman Method
Using UCV Method
Using BCV Method
Using DPI Method
Density
x
0.00
0.05
0.10
0.15
0.20
−6
−4
−2
0
2
4
6




Here all methods are performing well, except some of the adjusted histogram methods. We have seen these observations through numerical results also.

## 8.3 Simulation - 3 :

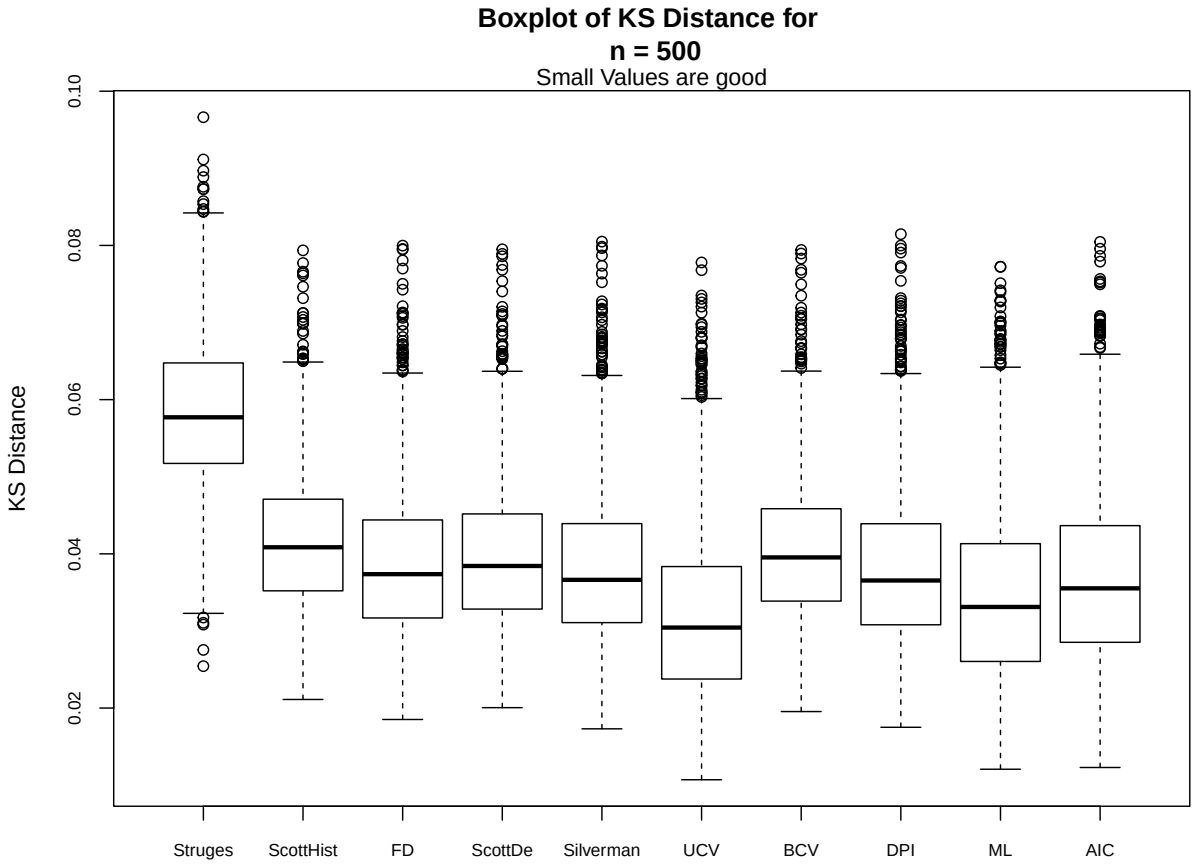
Now we consider claw distribution, which is basically a distribution with four modes. you can see `?rclaw` for more details.




For this distribution we will carry forward the same analysis for $n = 500, 2000$

### 8.3.1 For n = 500 :

#### 8.3.1.1 KS Distance :

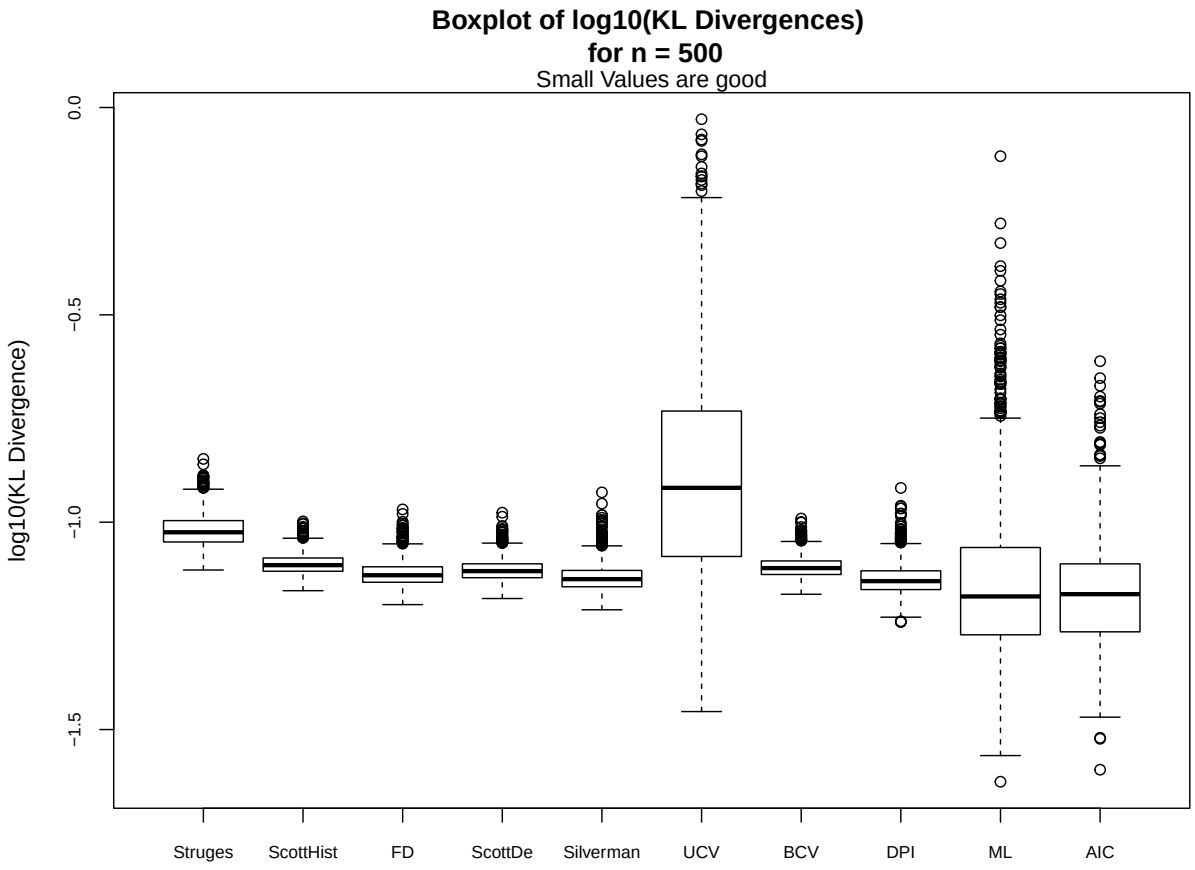



From the above Boxplot, we can see that UCV,ML and AIC are performing quite similarly. Let's see the Monte carlo estimate of Expected KS Distances.

```
##        UCV         ML        AIC        DPI  Silverman         FD    ScottDe
## 0.03208984 0.03452087 0.03680027 0.03822838 0.03831434 0.03883796 0.03955843
##        BCV  ScottHist    Struges
## 0.04047380 0.04160770 0.05811908
```

The Expected KS Distance values are also depicting the same thing. Interestingly, AIC is performing quite well than the previous Simulations. Note that **UCV** has least Expected KS Distance.

#### 8.3.1.2 KL Divergence :

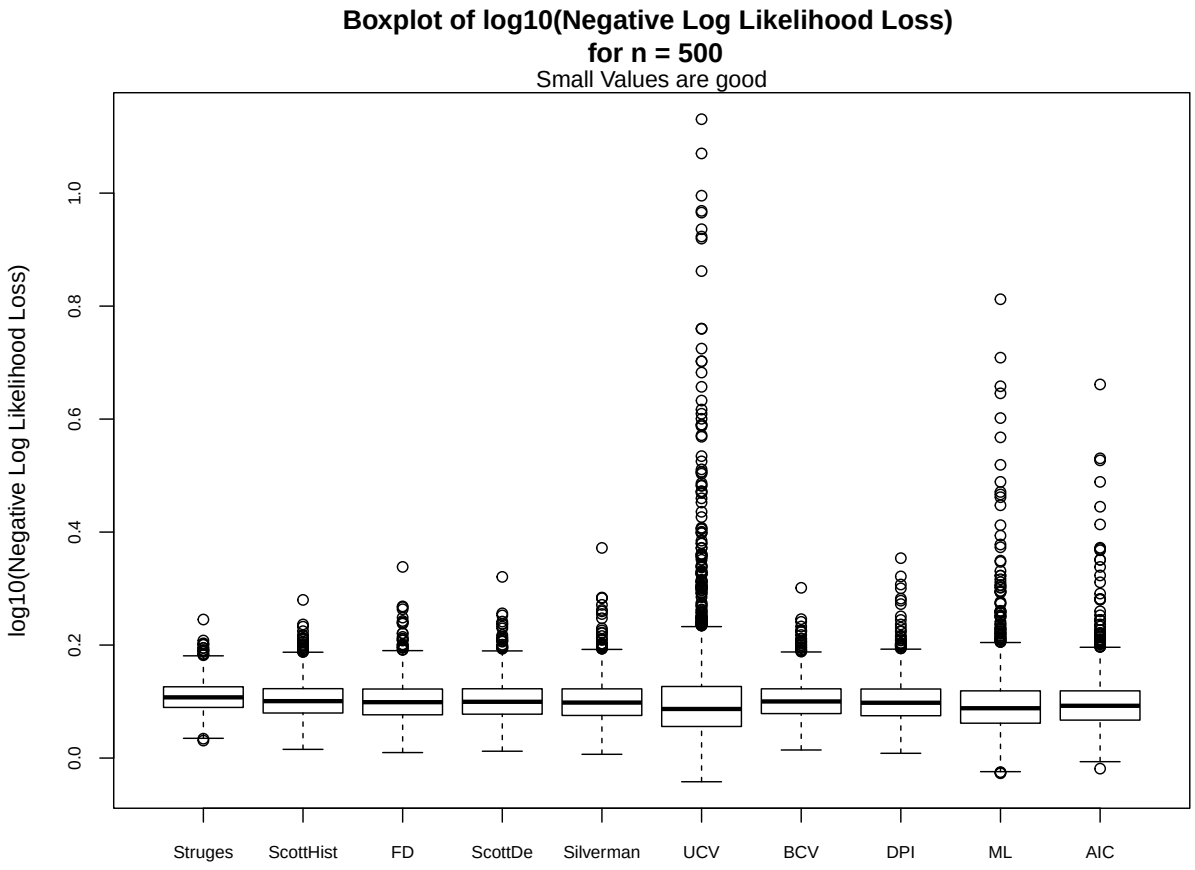



We have drawn Boxplot for log10(KL Divergence). From the Boxplot, we can see that UCV and ML have many large outliers. Let's see the Monte carlo estimate of Expected KL Divergences.

```
##        AIC        DPI  Silverman         FD    ScottDe        BCV         ML
## 0.06898976 0.07305072 0.07379350 0.07526735 0.07674310 0.07799492 0.07918187
##  ScottHist    Struges        UCV
## 0.07935795 0.09598027 0.15173763
```

Interestingly, AIC method has least Expected KL Divergence.

#### 8.3.1.3 Negative Log Likelihood Loss :



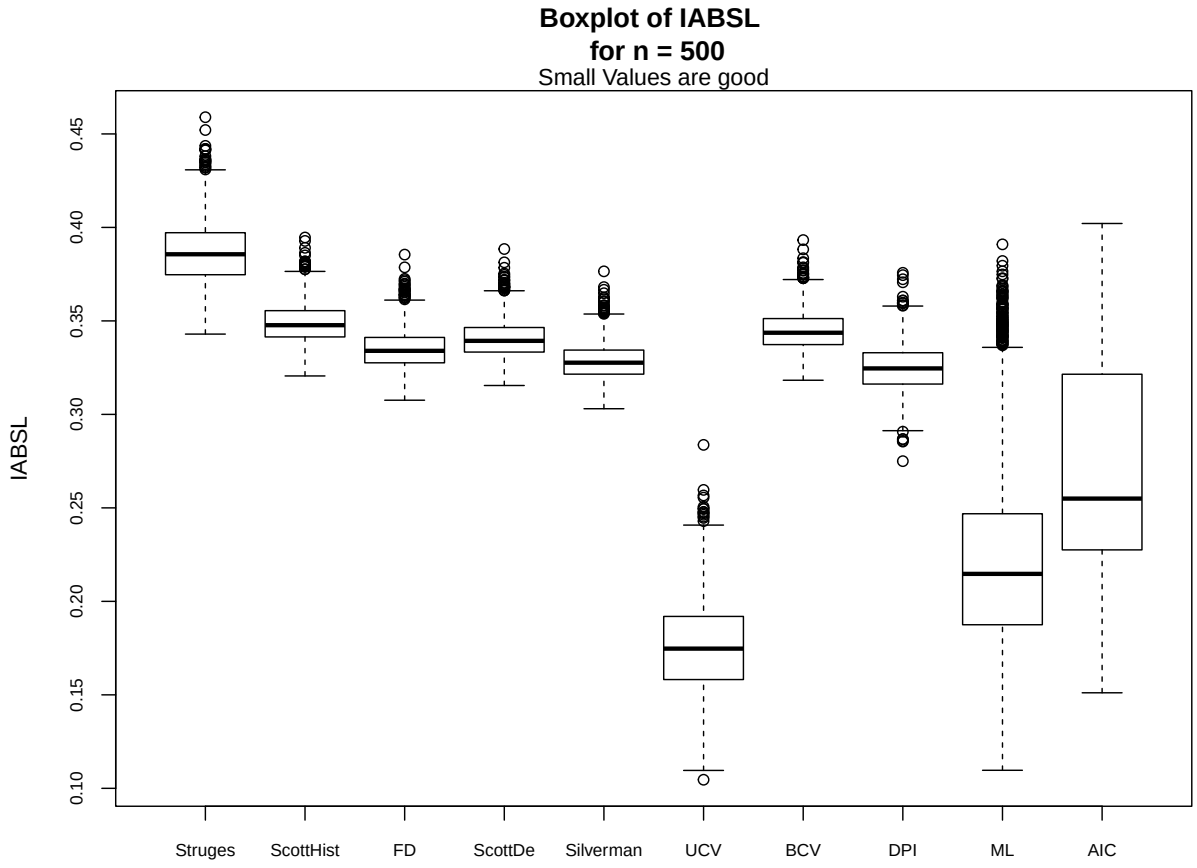

Here also, we have drawn Boxplot of log10(Negative Log Likelihood Loss). From the Boxplot, we can see that UCV and ML have many large outliers. Others methods are performing similarly.

The Average values of Negative Log Likelihood Losses are -

```
##       AIC       DPI        ML Silverman        FD   ScottDe       BCV ScottHist
##  1.259315  1.264456  1.264526  1.265149  1.266594  1.268043  1.269285  1.270598
##   Struges       UCV
##  1.287288  1.344143
```

The Average values are not very different.

#### 8.3.1.4 Integrated Absolute Loss :



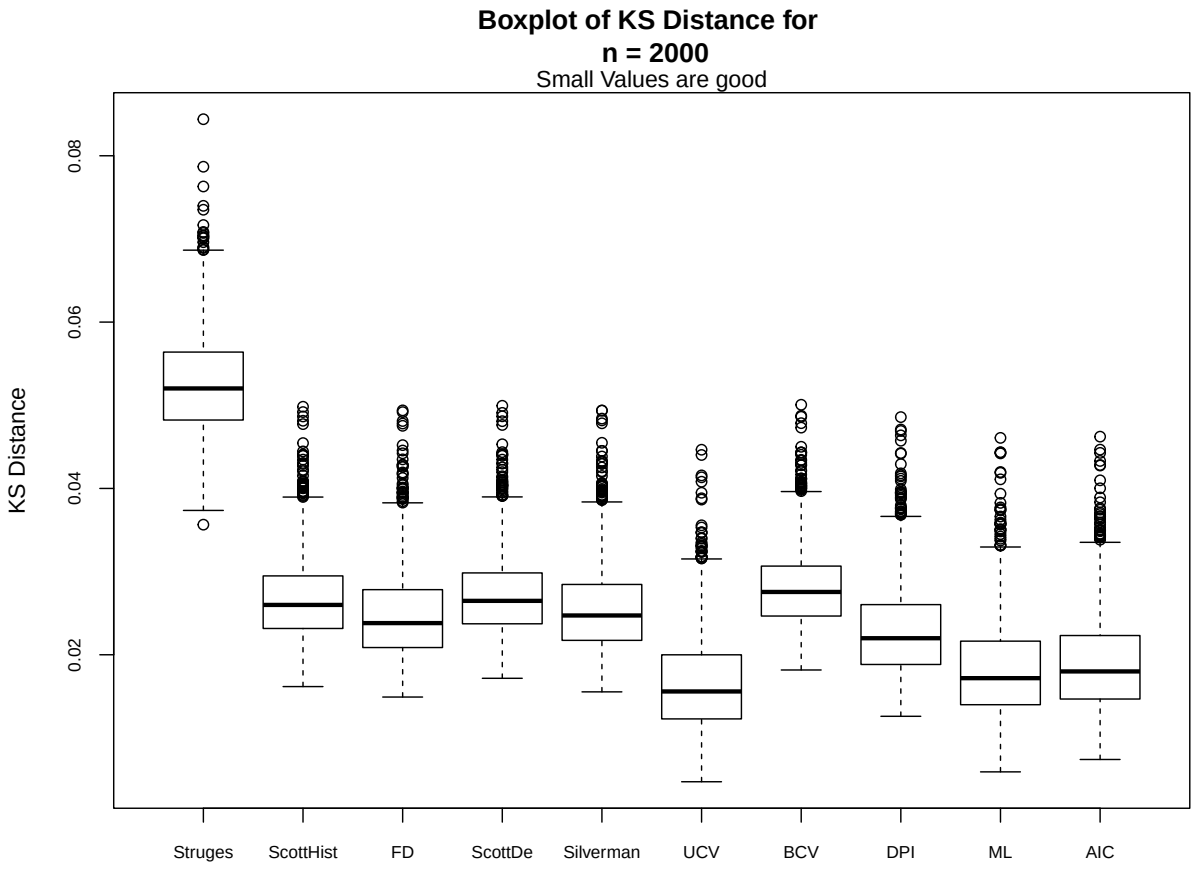

From the above Boxplot, we can see that UCV has small values of IABSL. It is very interesting observation. Let's see the Monte carlo estimate of Expected IABSL.

```
##       UCV        ML       AIC       DPI Silverman        FD   ScottDe       BCV
## 0.1759128 0.2226958 0.2680325 0.3248739 0.3284910 0.3349173 0.3402962 0.3445891
## ScottHist   Struges
## 0.3487830 0.3867129
```

We can see that **UCV** has least expected IABSL.

### 8.3.2 For n = 2000 :

#### 8.3.2.1 KS Distance :

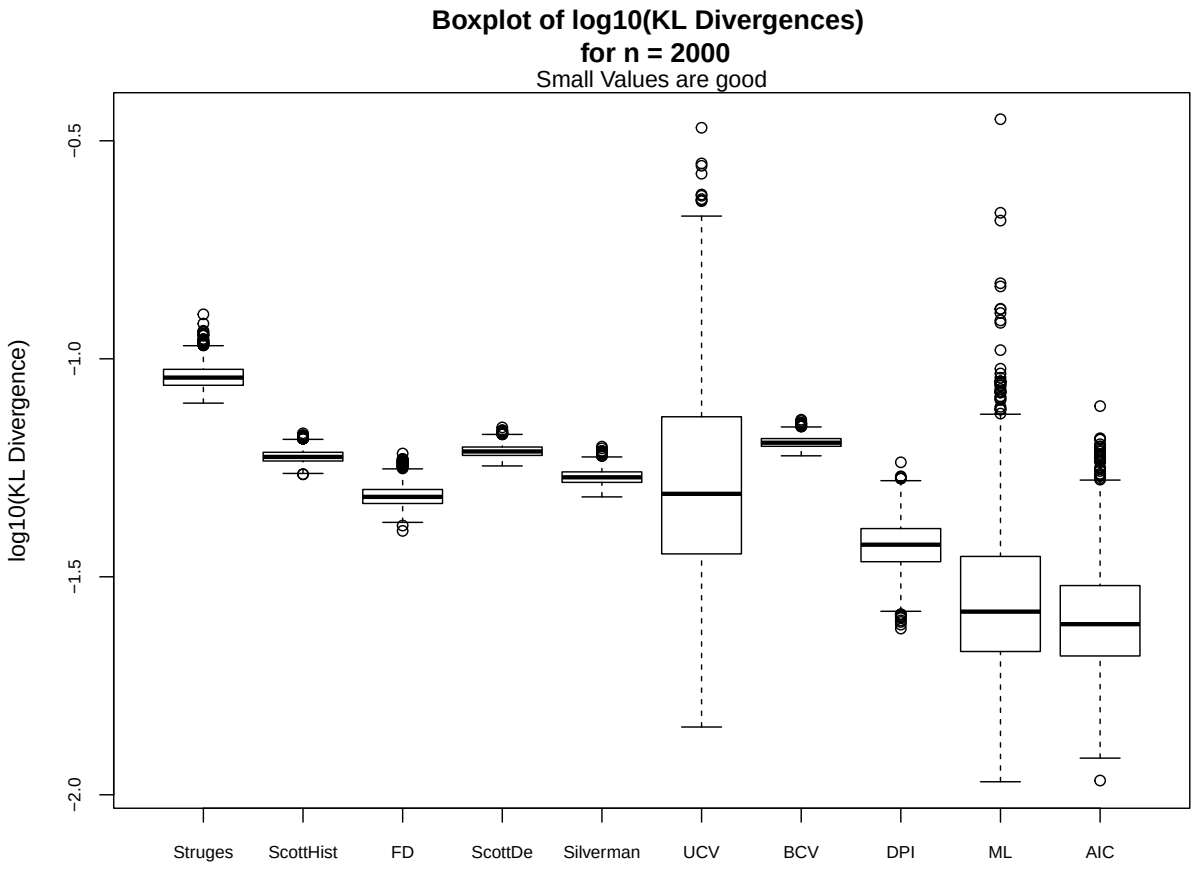



From the Boxplot, we are getting similar kind of observation as in n = 500 case. Let's see Monte carlo estimate of Expected KS Distances.

```
##        UCV         ML        AIC        DPI         FD  Silverman  ScottHist
## 0.01655030 0.01820956 0.01898058 0.02288617 0.02478246 0.02560228 0.02674576
##    ScottDe        BCV    Struges
## 0.02716534 0.02806259 0.05248491
```

The expected KS Distance values are also depicting the same thing and **UCV** has expected KS Distance.

#### 8.3.2.2 KL Divergence :

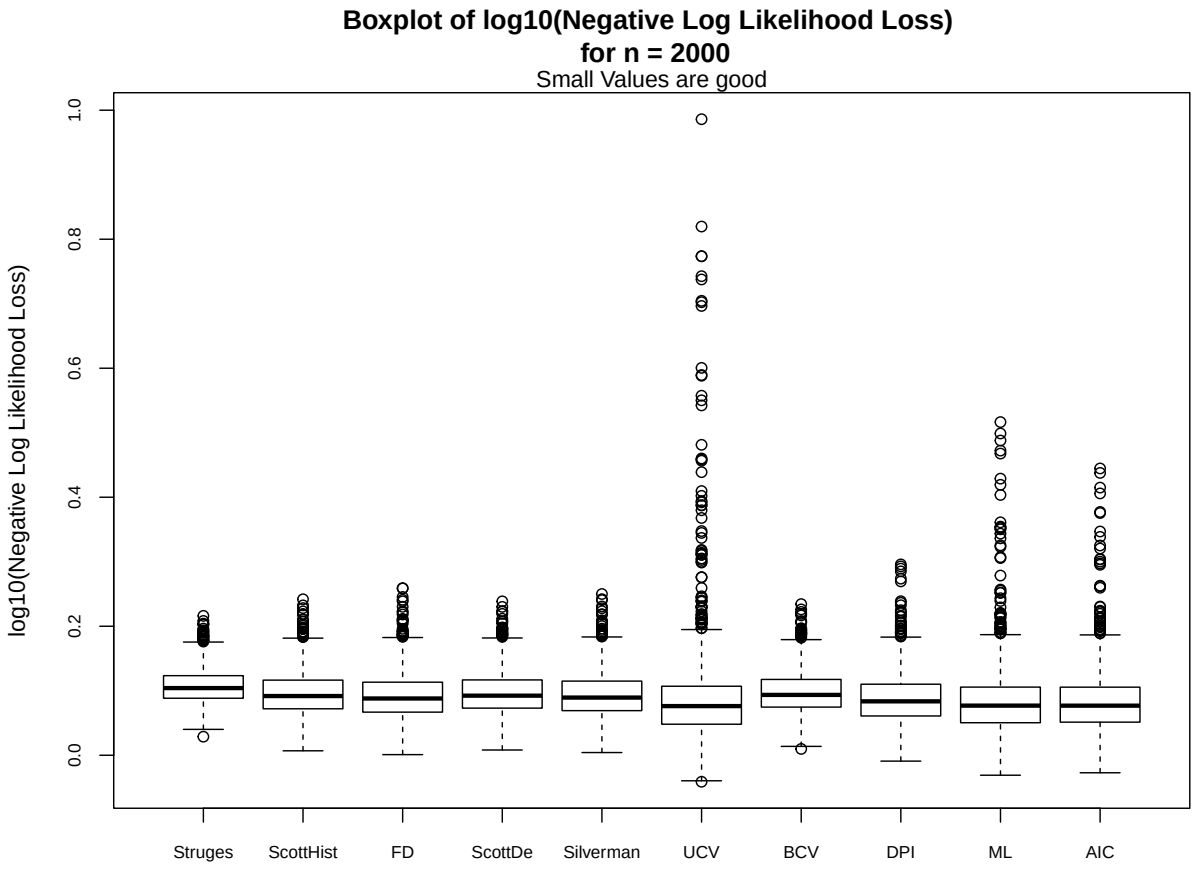



From the Boxplot also, we are getting similar kind of observation as in n = 500 case. Now, let's see the Monte carlo estimate of Expected KL Divergences.

```
##        AIC         ML        DPI         FD  Silverman        UCV  ScottHist
## 0.02652967 0.03084694 0.03763938 0.04846448 0.05365320 0.05914331 0.05972017
##    ScottDe        BCV    Struges
## 0.06146873 0.06436800 0.09124718
```

The expected KL Divergence values are also depicting the same thing and here also AIC has least Expected KL Distance.

#### 8.3.2.3 Negative Log Likelihood Loss :



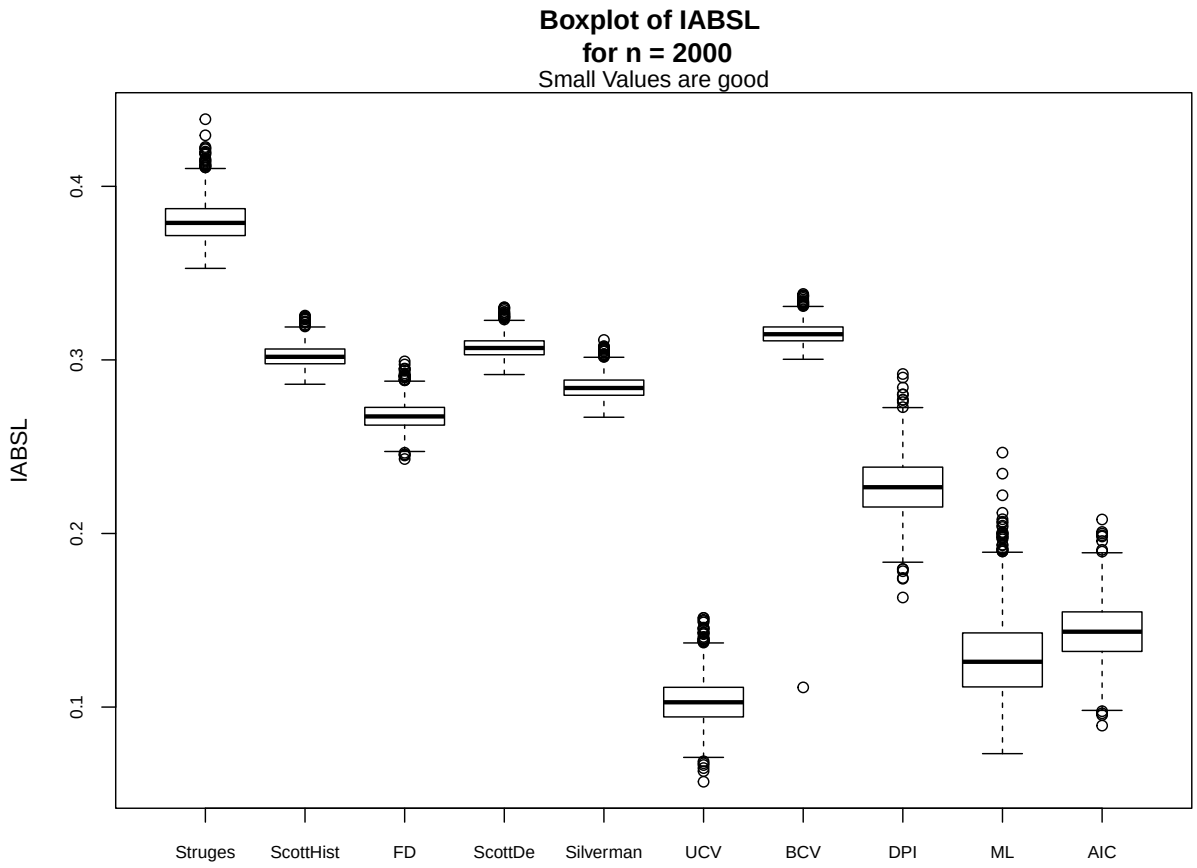

From the above Boxplot we can see that UCV has many large outliers. Other methods are performing similarly.

The Average values of Negative Log-Likelihood Losses are -

```
##       AIC        ML       DPI        FD Silverman       UCV ScottHist   ScottDe
##  1.215184  1.218534  1.226945  1.237959  1.243249  1.249289  1.249393  1.251156
##       BCV   Struges
##  1.254032  1.281204
```

The Average Values are not very different !

#### 8.3.2.4 Integrated Absolute Loss :




From the above Boxplot, we can see that UCV is performing better than others. Although it has some small and large outliers. Here, we are getting similar kind of observations as in case of n = 500 case. Let's see the Monte carlo estimate of Expected IABSL.

```
##       UCV        ML       AIC       DPI        FD Silverman ScottHist   ScottDe
## 0.1031319 0.1286591 0.1434905 0.2267613 0.2677746 0.2842706 0.3022353 0.3072160
##       BCV   Struges
## 0.3151770 0.3799546
```

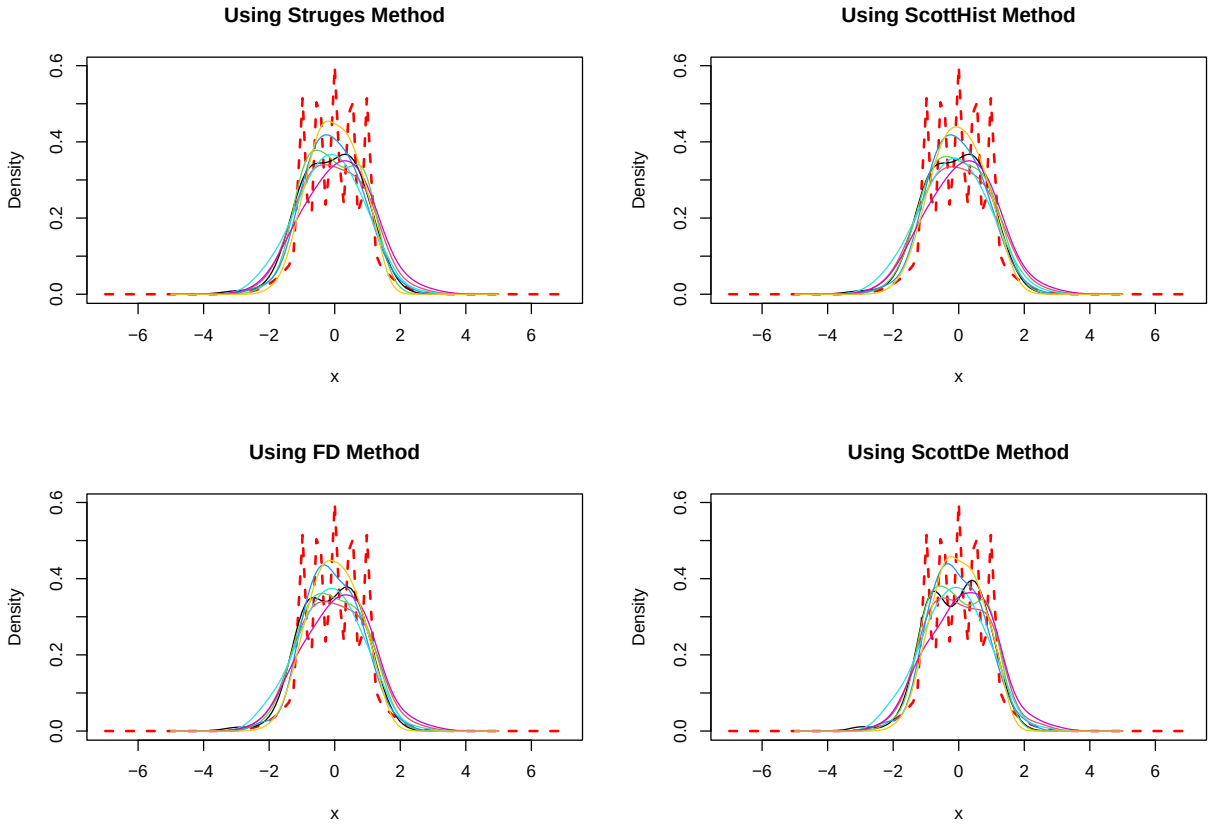
Here also **UCV** has least expected IABSL.

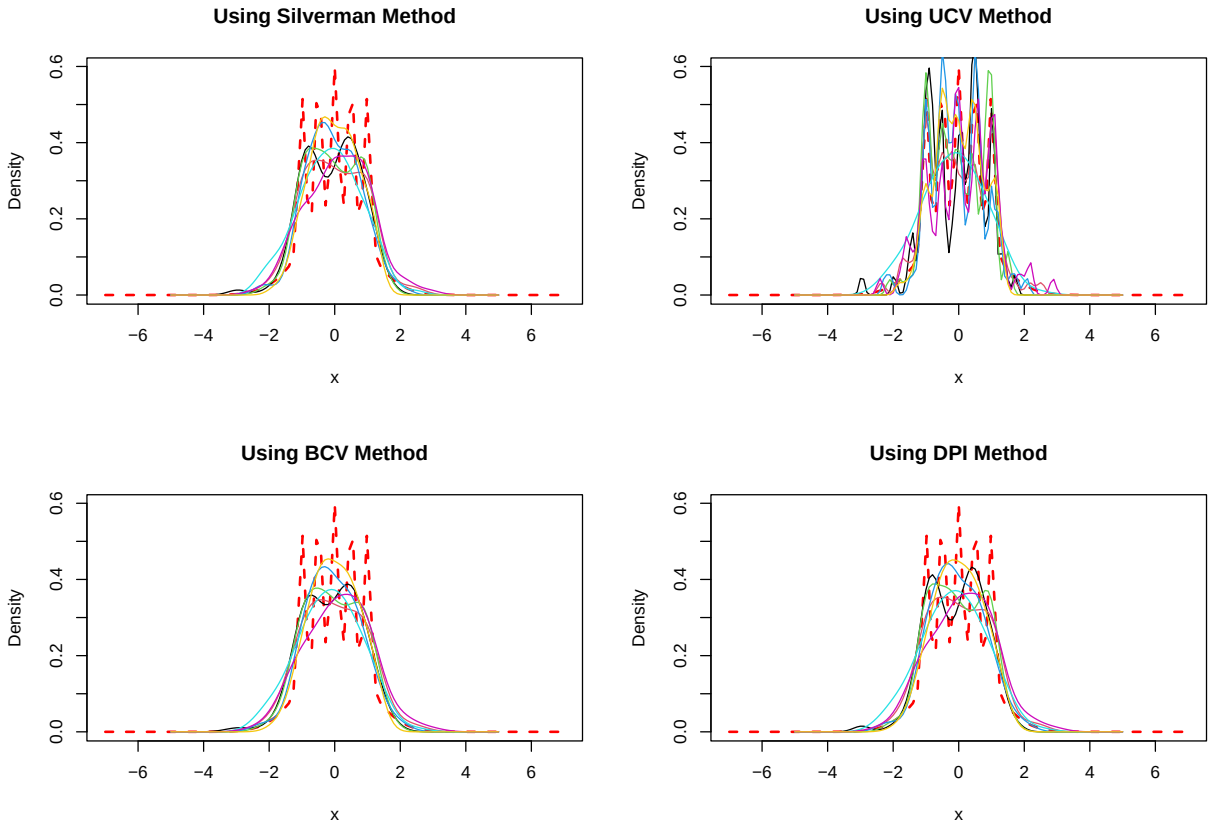
### 8.3.3 Graphical Verification :

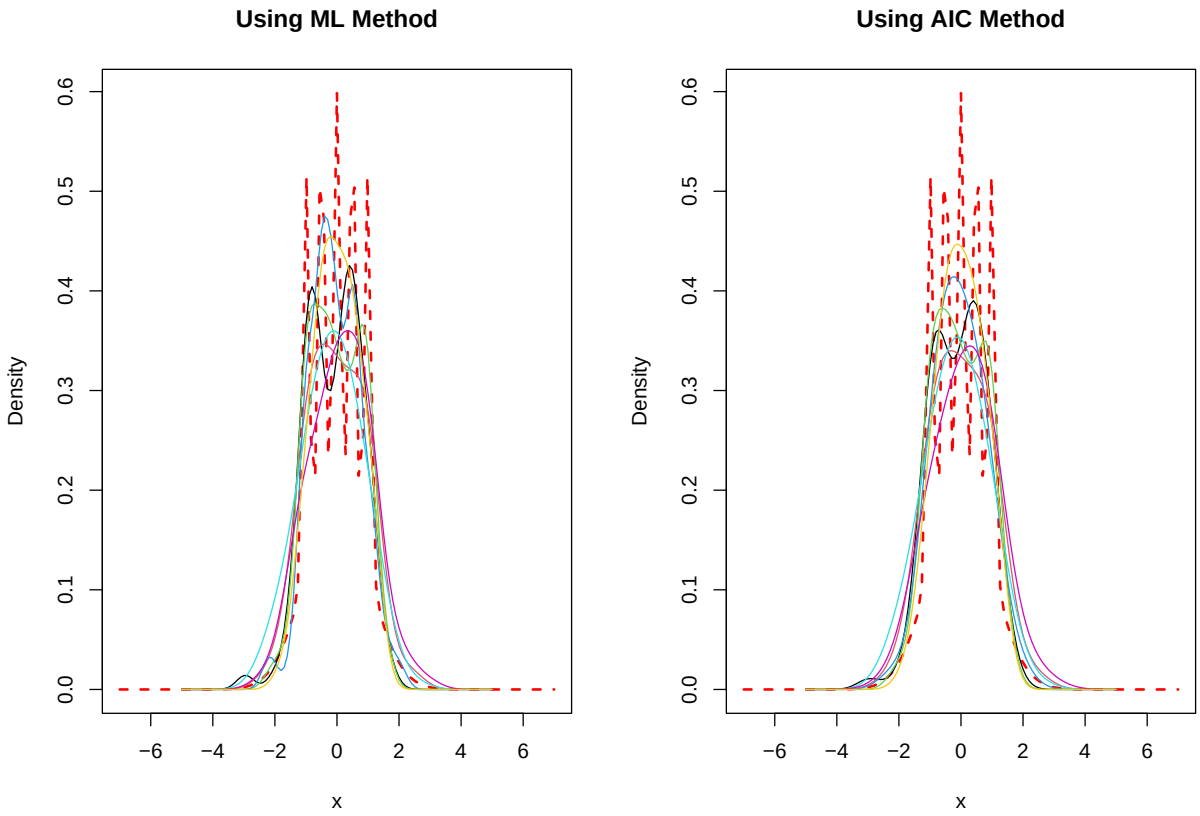
As we have seen graphical verification is important. So, we have done similar thing in this case also.

**For n = 100**

We have not done the simulation for n = 100. But it is interesting to see for n = 100 also.

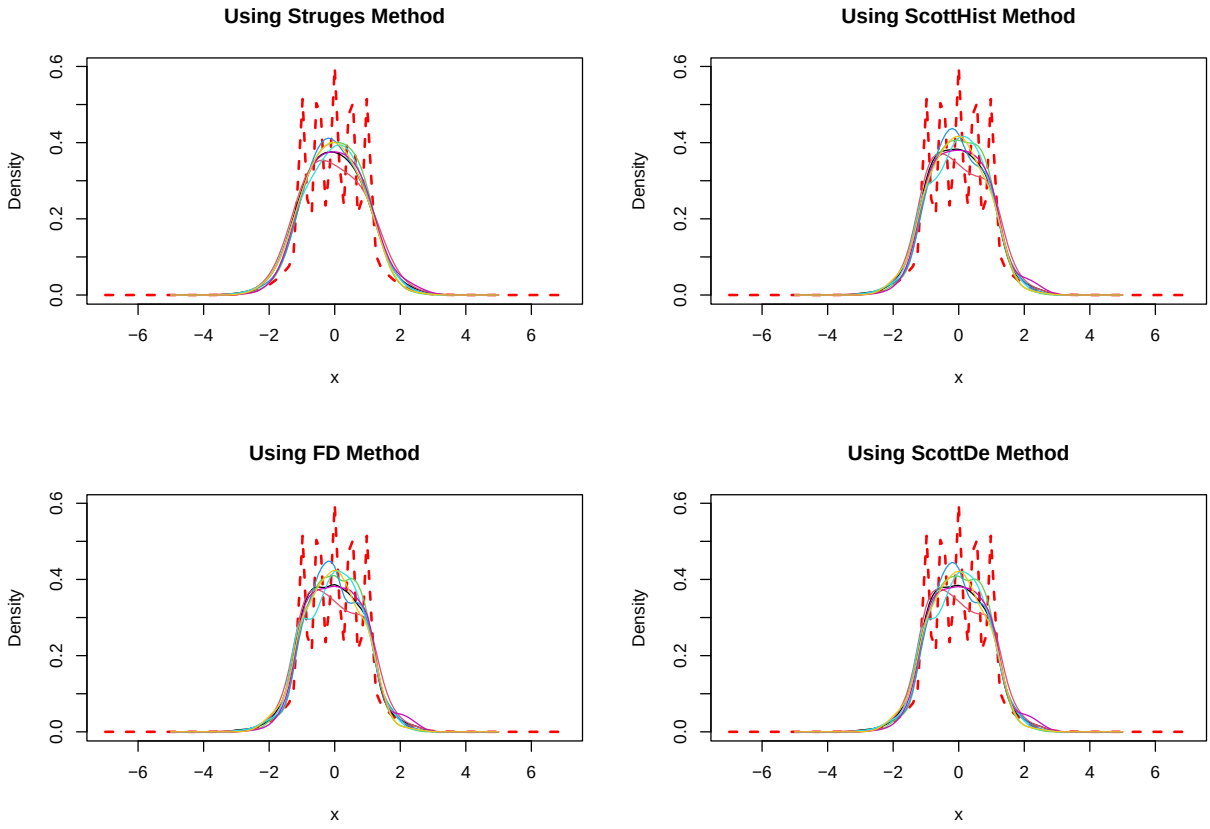

Using Struges Method
Using ScottHist Method
Using FD Method
Using ScottDe Method
Density
x
0.0
0.2
0.4
0.6
−6
−4
−2
0
2
4
6

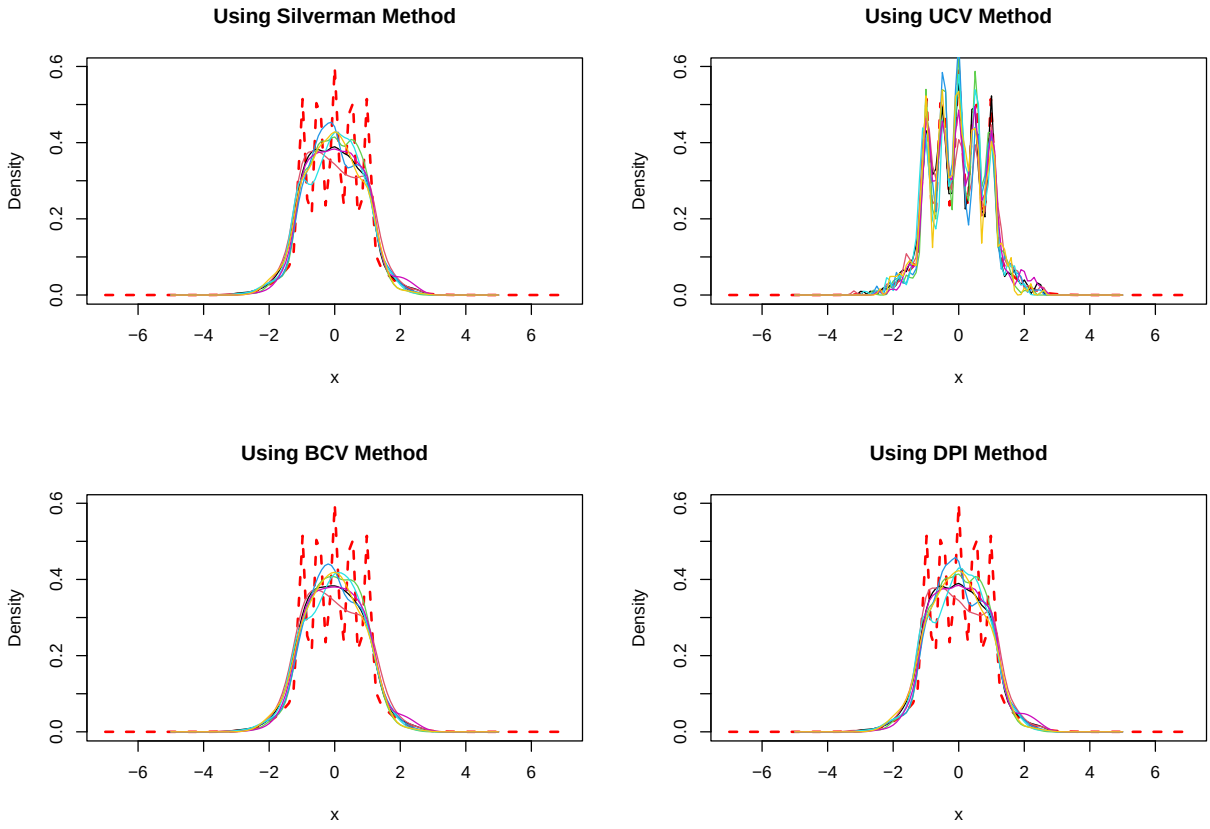

Using Silverman Method
Using UCV Method
Using BCV Method
Using DPI Method
Density
x
0.0
0.2
0.4
0.6
−6
−4
−2
0
2
4
6

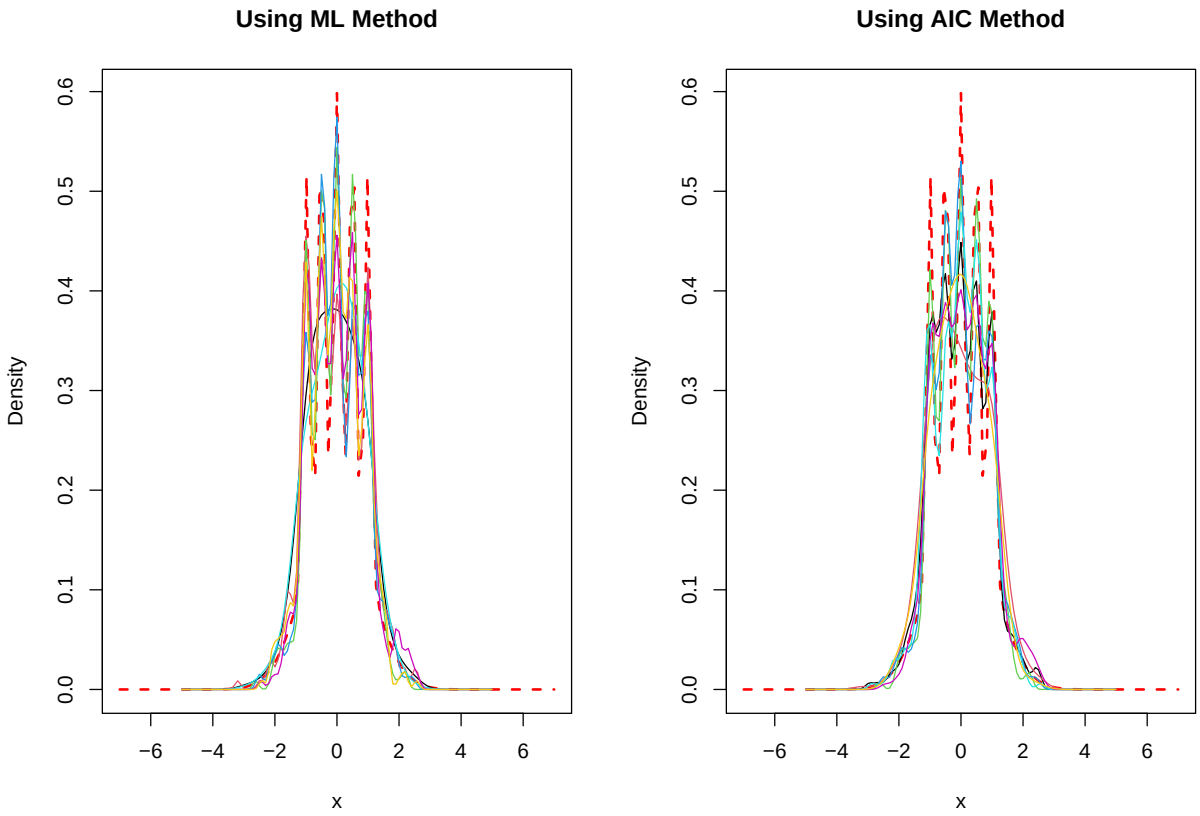



All methods are performing very bad. But, **UCV** is able to capture the modes of the distribution.

**For n = 500**

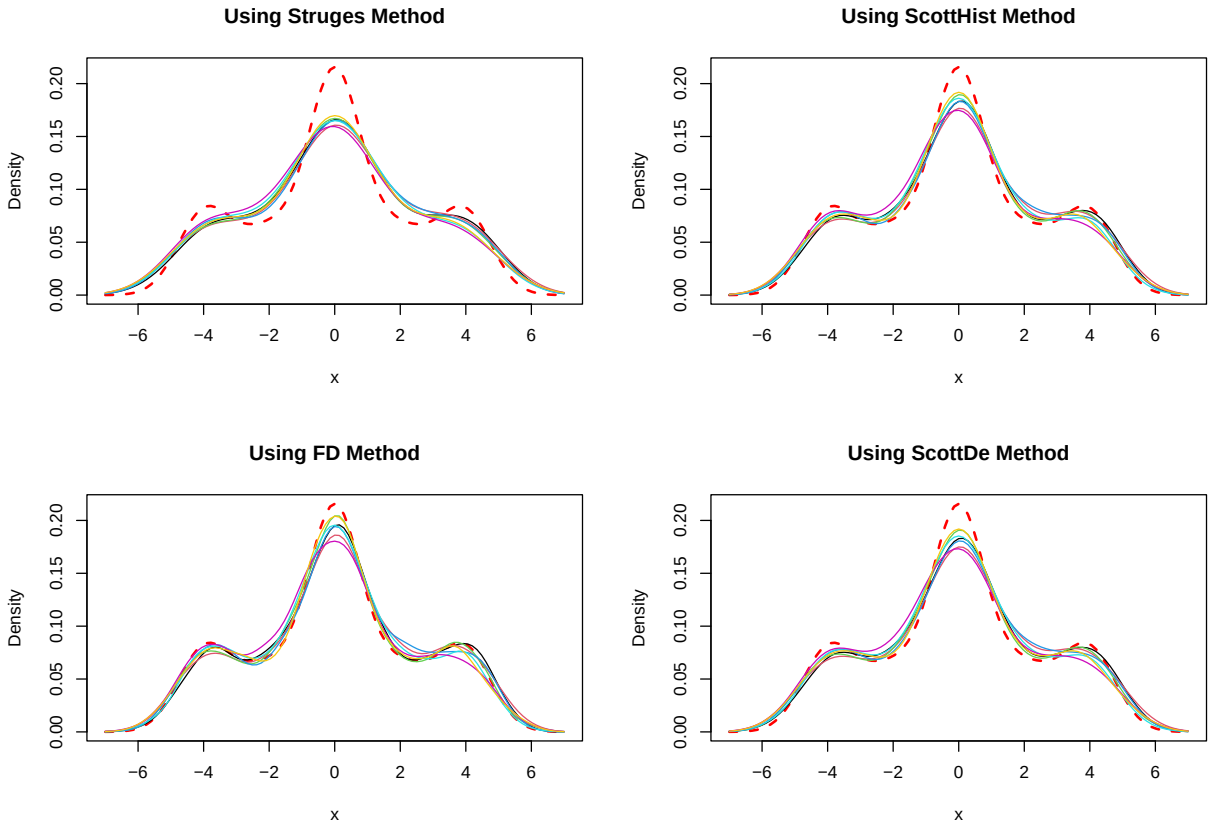

Using Struges Method
Using ScottHist Method
Using FD Method
Using ScottDe Method
Density
x
0.0
0.2
0.4
0.6
−6
−4
−2
0
2
4
6

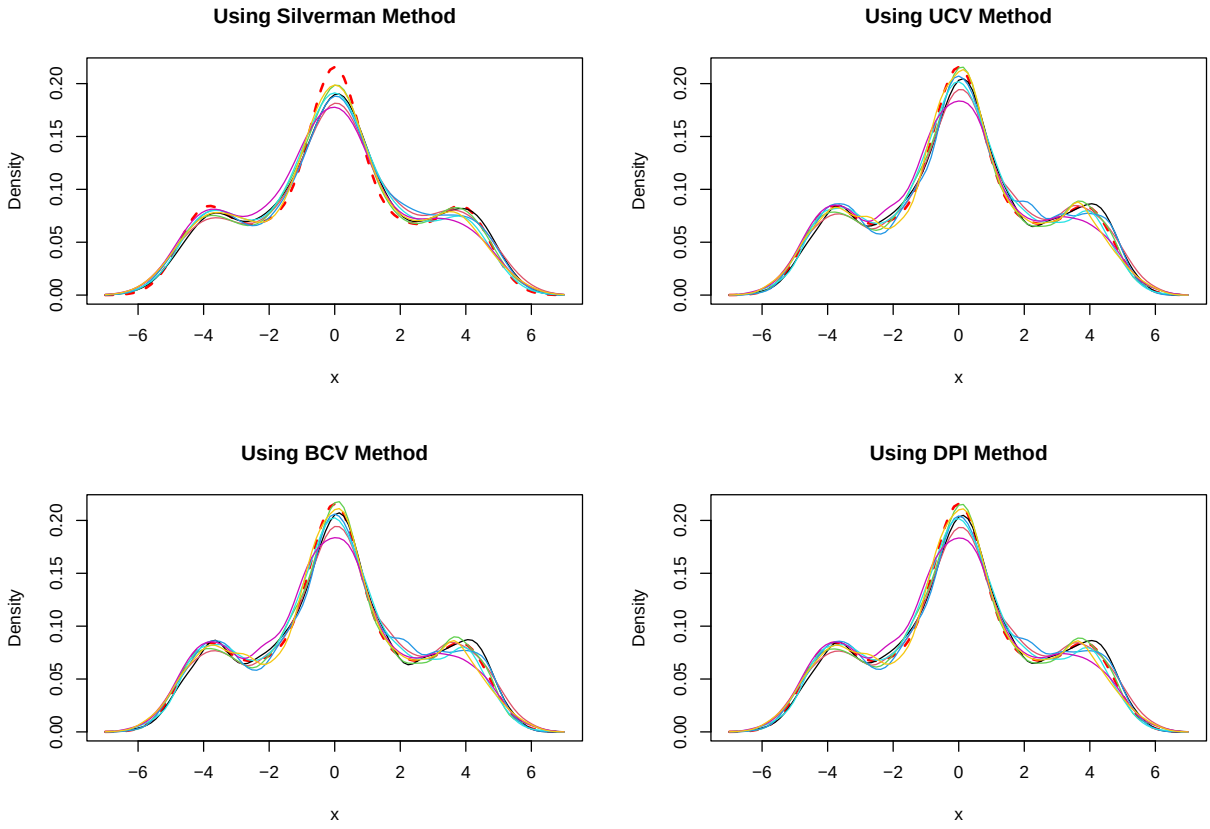

Using Silverman Method
Using UCV Method
Using BCV Method
Using DPI Method
Density
x
0.0
0.2
0.4
0.6
−6
−4
−2
0
2
4
6

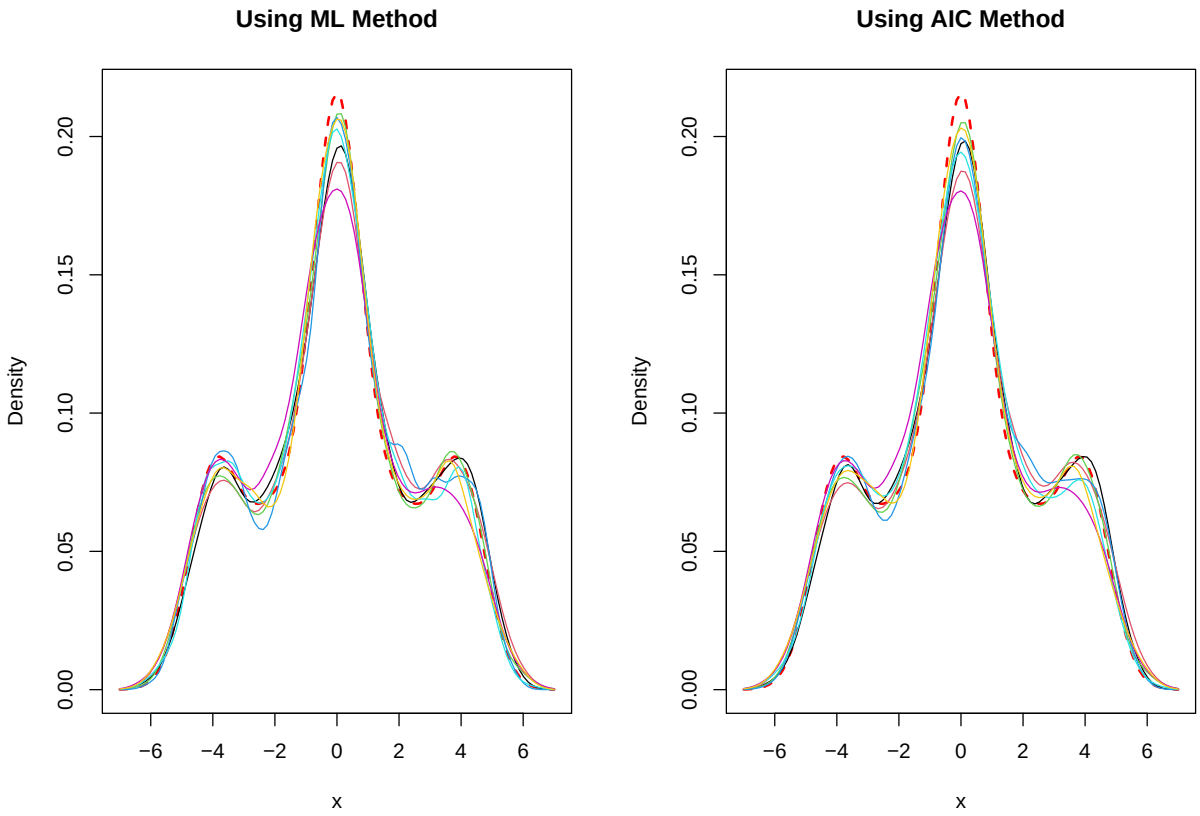



**ML,UCV** and **AIC** are performing much better than others. This observation is consistent with our observation from simulation.

**For n = 2000**


Using Struges Method
Using ScottHist Method
Using FD Method
Using ScottDe Method
Density
x
0.00
0.05
0.10
0.15
0.20
−6
−4
−2
0
2
4
6


Using Silverman Method
Using UCV Method
Using BCV Method
Using DPI Method
Density
x
0.00
0.05
0.10
0.15
0.20
−6
−4
−2
0
2
4
6



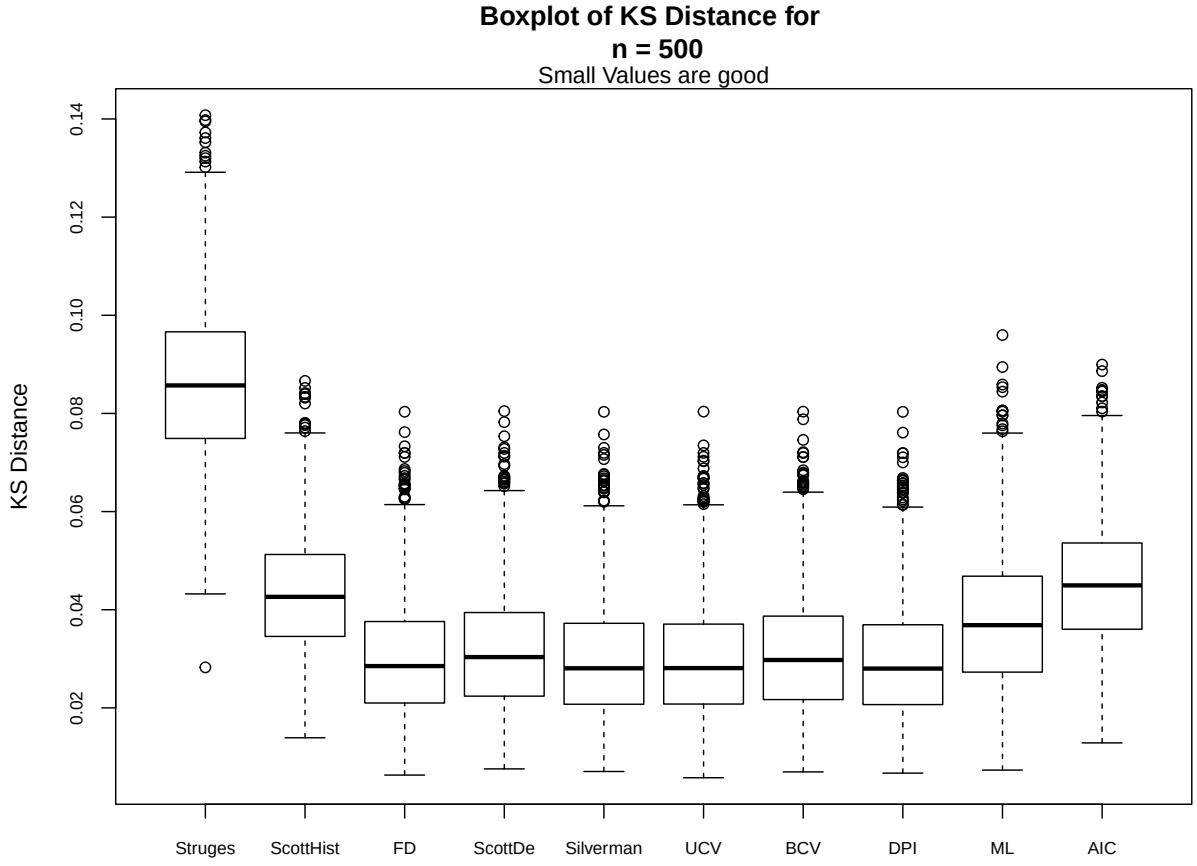

Here also, **ML,UCV** and **AIC** are performing much better than others.

## 8.4 Simulation - 4 :

Next, we have considered Laplace Distribution. For this case, we have considered only for $n = 500$

### 8.4.1 For n = 500 :

#### 8.4.1.1 KS Distance :

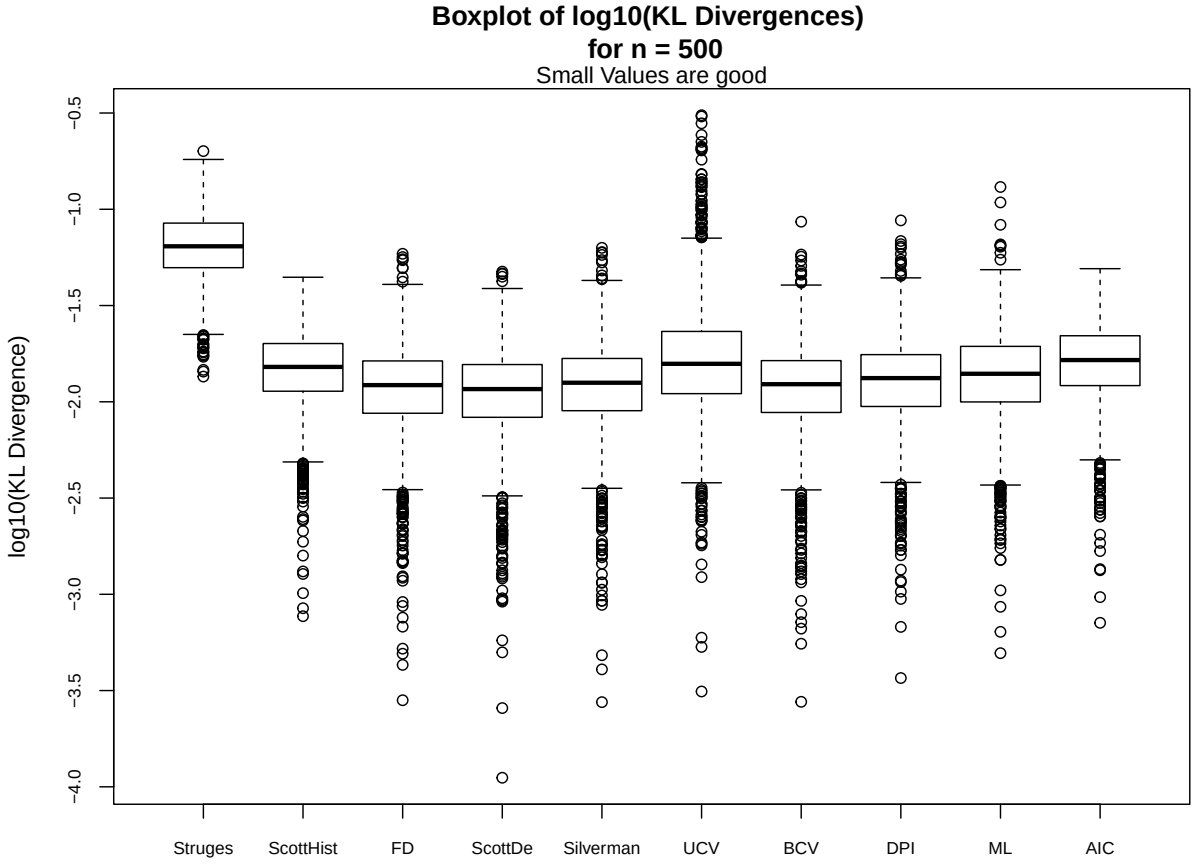



From the above Boxplot, we can see that Scott Density Method, Silverman,UCV,BCV and DPI are performing similarly. Also, *Kernel Density Adjusted* FD method is also performing similarly. Let's look at the Monte carlo estimate of Expected KS Distances.

```
##        DPI  Silverman        UCV         FD        BCV    ScottDe         ML
## 0.02957565 0.02980637 0.02981359 0.03019898 0.03102515 0.03160924 0.03790864
##  ScottHist        AIC    Struges
## 0.04344914 0.04532522 0.08640269
```

The Expected KS Distance values also depicting the same thing. Also, DPI has least Expected KS Distance.

#### 8.4.1.2 KL Divergence :

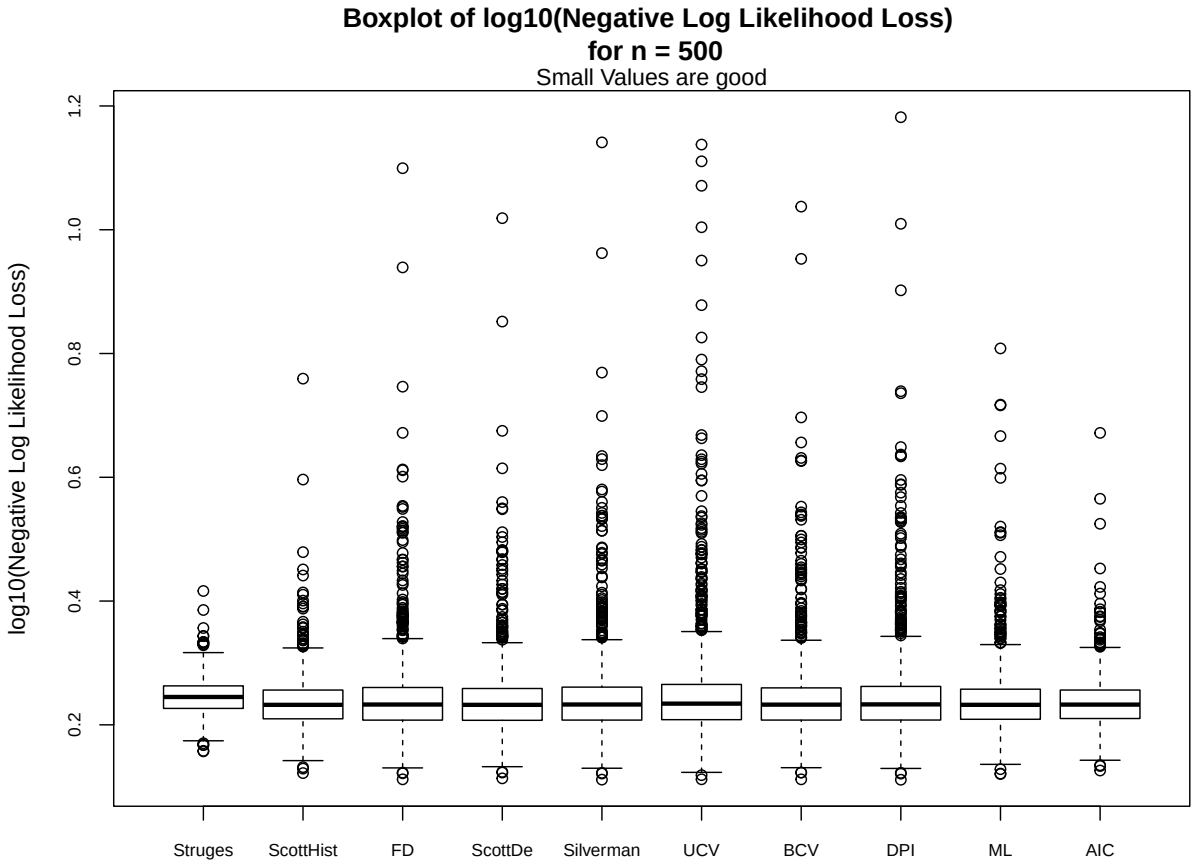



We have drawn Boxplot of log10(KL Divergence). From the Boxplot, we can see that except Struges method, all methods are performing similarly. UCV method has many small as well as large outliers. Let's see the Monte carlo estimate of Expected KL Divergences.

```
##    ScottDe         FD        BCV  Silverman        DPI         ML  ScottHist
## 0.01236189 0.01305311 0.01320699 0.01347434 0.01441718 0.01540887 0.01602994
##        AIC        UCV    Struges
## 0.01744023 0.02147915 0.06909623
```

The Expected KL Divergence values are not very different.

#### 8.4.1.3 Negative Log Likelihood Loss :

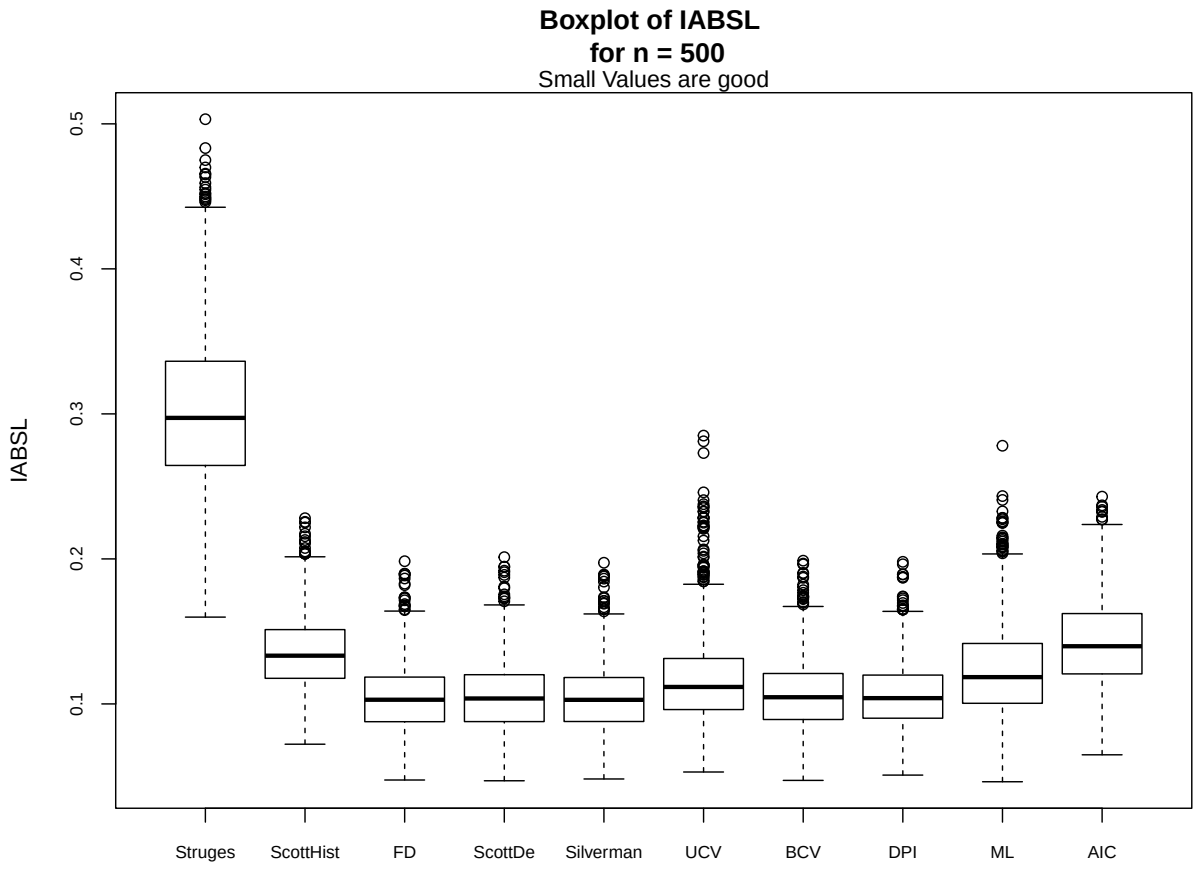



We have drawn Boxplot of log10(Negative Log Likelihood Loss). From the Boxplot, we can see that all are performing similarly. Although UCV has large outliers.

The Average values of Negative Log-Likelihood Losses are -

```
## ScottHist       AIC   ScottDe       BCV        FD Silverman   Struges       DPI
##  1.722965  1.722976  1.743020  1.752633  1.755034  1.760415  1.762180  1.771109
##       UCV        ML
##       Inf       Inf
```

#### 8.4.1.4 Integrated Absolute Loss :



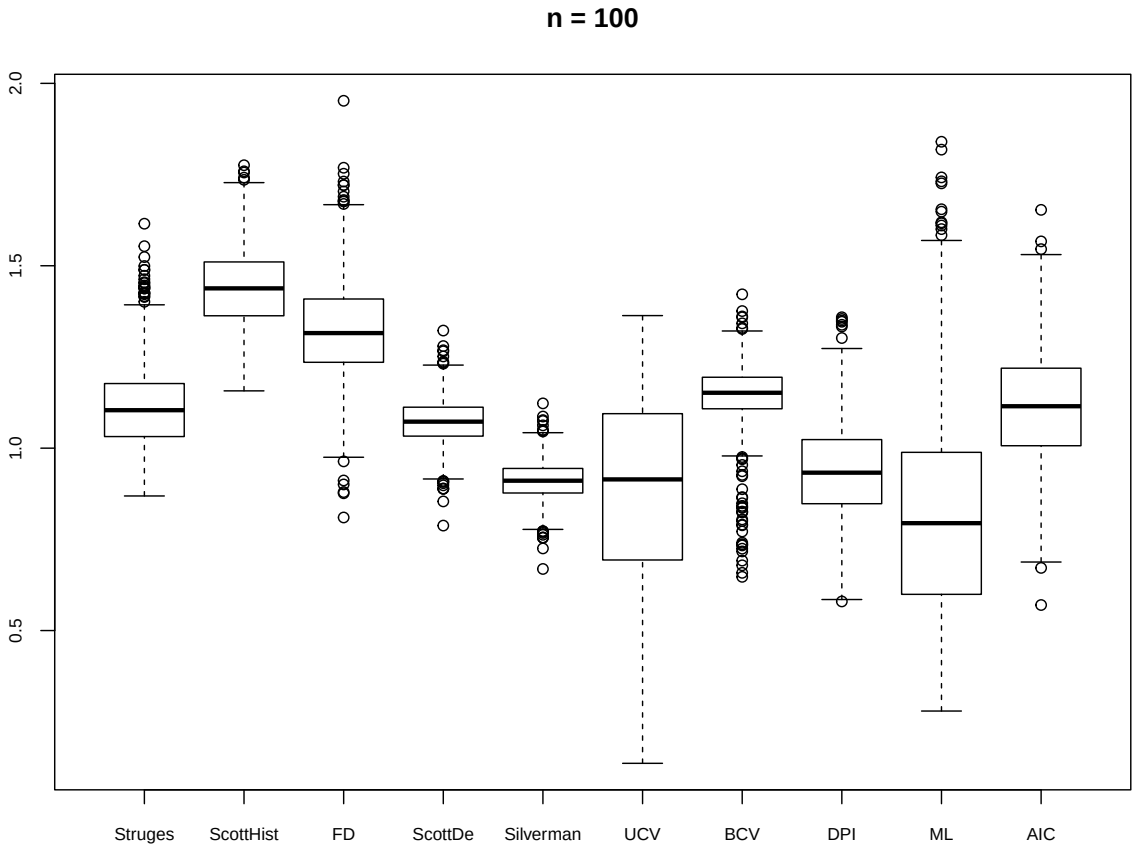

From the above Boxplot, we can see that FD,Scott Density method, Silverman,UCV,BCV,DPI are performing quite similarly. Let's see the Monte carlo estimate of Expected IABSL.

```
##        FD Silverman   ScottDe       DPI       BCV       UCV        ML ScottHist
## 0.1042598 0.1043958 0.1053532 0.1060035 0.1063459 0.1160602 0.1228695 0.1350079
##       AIC   Struges
## 0.1418350 0.3023604
```

Expected IABSL values are also depicting the same thing. Interestingly, *Kernel Density Adjusted* **FD** has least Expected IABSL.

# 9 Boxplot of $h$ values :

## 9.1 Bimodal Mixture Normal Distribution :




From the above Boxplot, we can see that different methods are producing very different set of values of $h$. These set of values of $h$ has very central tendency.

```
##   Struges ScottHist        FD   ScottDe Silverman       UCV       BCV       DPI
## 1.1107145 1.4397096 1.3215905 1.0732837 0.9112786 0.8738495 1.1481244 0.9385246
##        ML       AIC
## 0.8085395 1.1129418
```

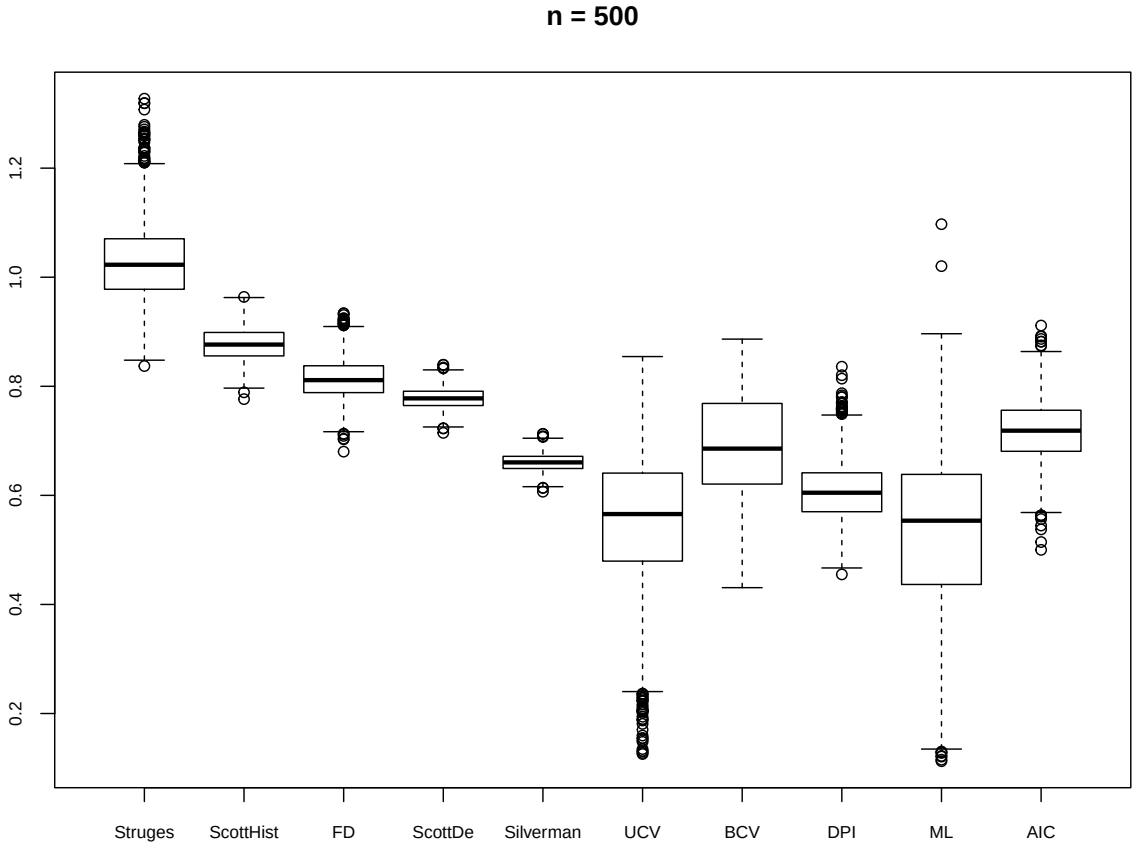
These are the Expected $h$ values obtained using different methods. They may be near to optimum $h$ obtained using some criteria. For example - $h$ that minimizes $E[IABSL]$ i.e. $L_1$ criteria or $E[IMSE]$ i.e. $L_2$ criteria (Obviously these $h$ values will depend on true pdf $f$ which is unknown to us). We should also see the stability of such $h$ values. i.e. even if the expected value of $h$ is near the optimum $h$ obtained using some method. It may happen that distribution of $h$ values are dispersed. Let's see the S.D of the distribution of $h$ obtained using different methods.

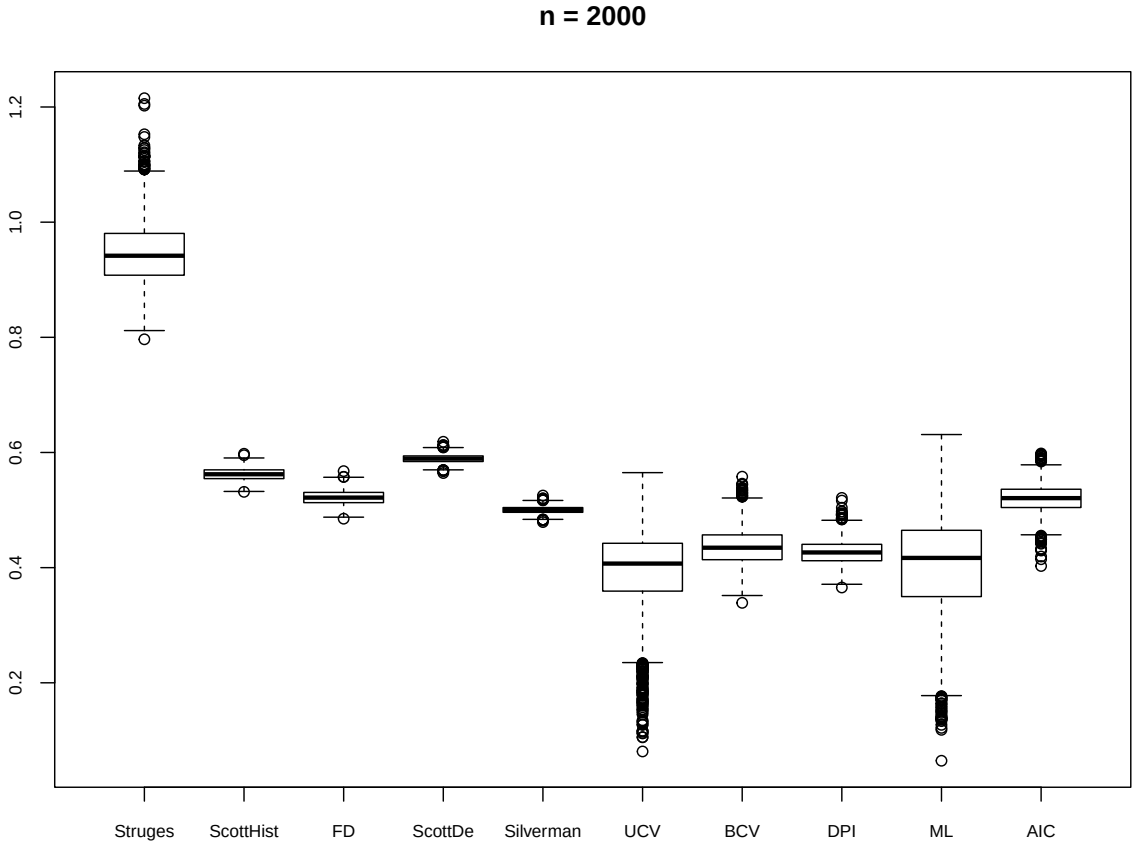
```
##    Struges  ScottHist         FD    ScottDe  Silverman        UCV        BCV
```

```
## 0.10638347 0.10483392 0.13047425 0.06034258 0.05123427 0.25669363 0.07680463
##        DPI         ML        AIC
## 0.12737036 0.27692411 0.15199636
```

As expected ML and UCV methods have large variability. This is a problematic thing. Because for large to small values of $h$ we will get very different types density estimate.

We have drawn the Boxplot $h$ values for n = 500,2000 also.



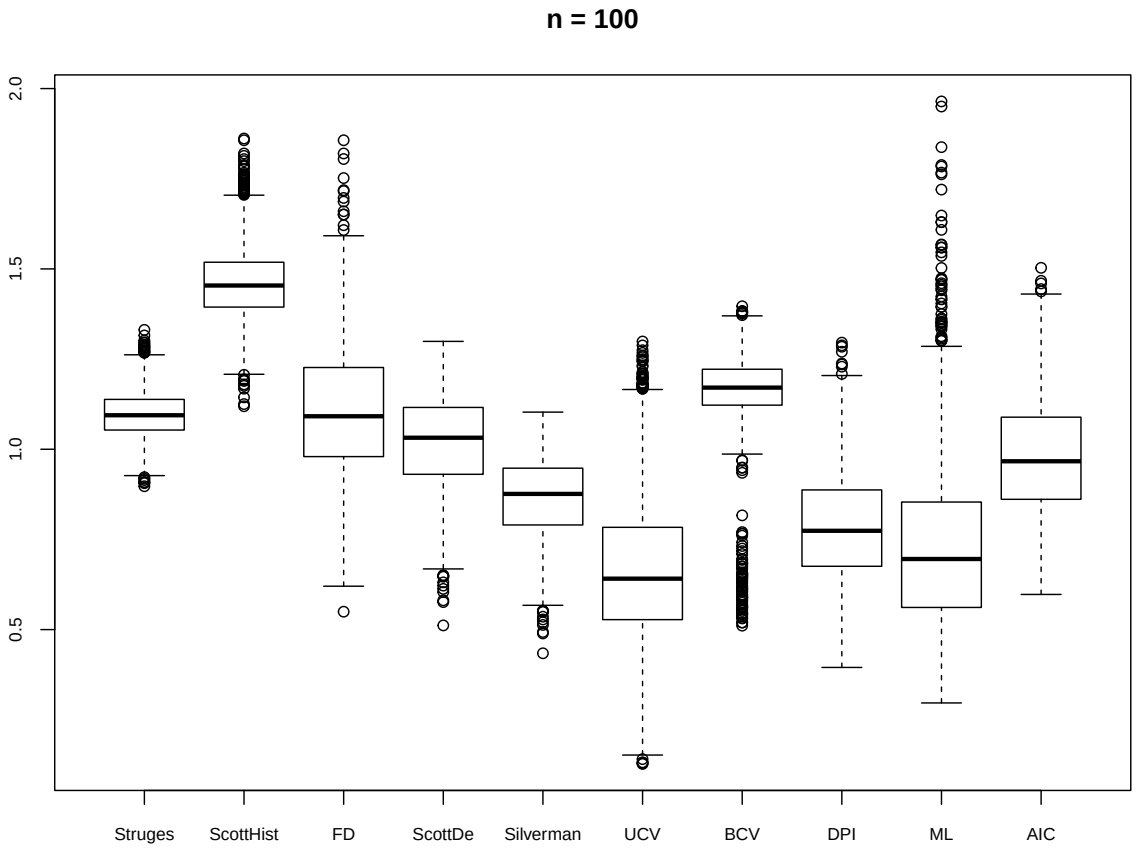



From the above Boxplots, we can see that as sample size increase the variability of $h$ values decreases. Our ML method may produce good results (in terms of different measures), but the $h$ values are dispersed.

For n = 2000, the expected $h$ values are -

```
##   Struges ScottHist        FD   ScottDe Silverman       UCV       BCV       DPI
## 0.9481003 0.5624203 0.5217610 0.5891845 0.5002510 0.3928583 0.4366482 0.4271110
##        ML       AIC
## 0.4018848 0.5197861
```

## 9.2 Trimodal Mixture Normal Distribution :

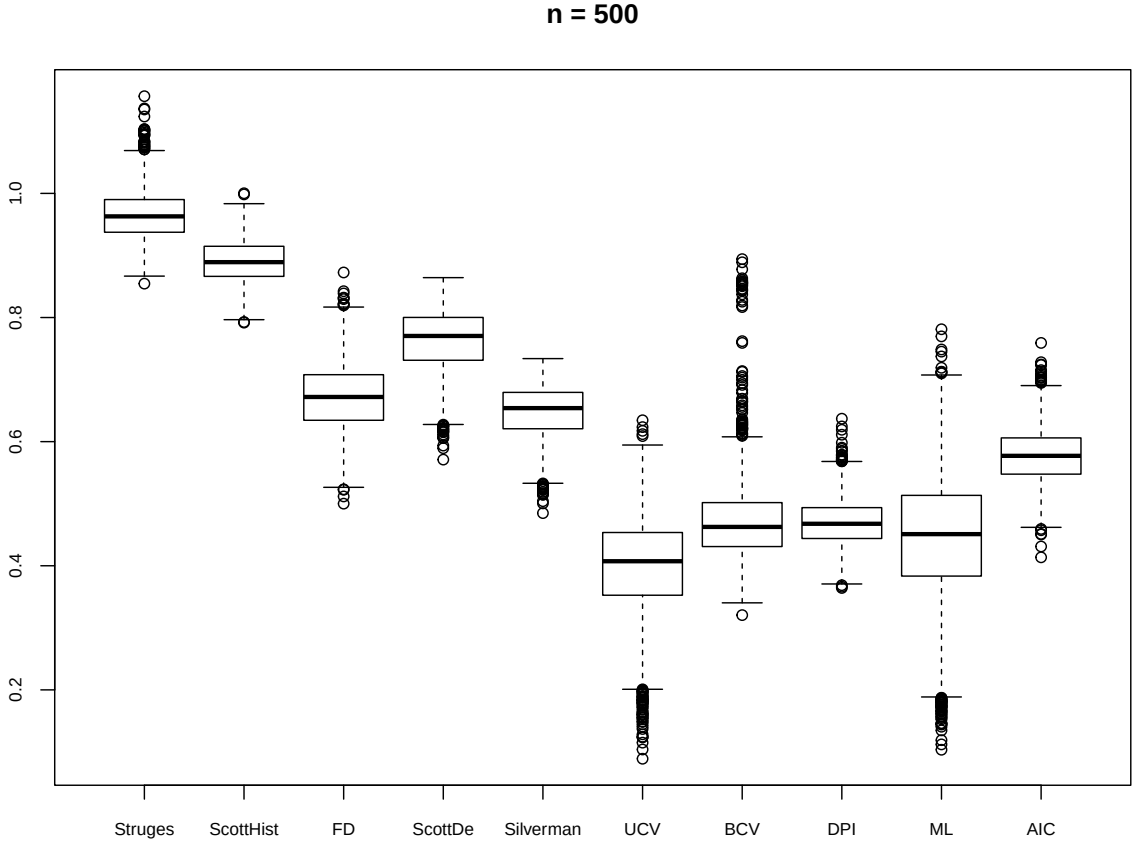


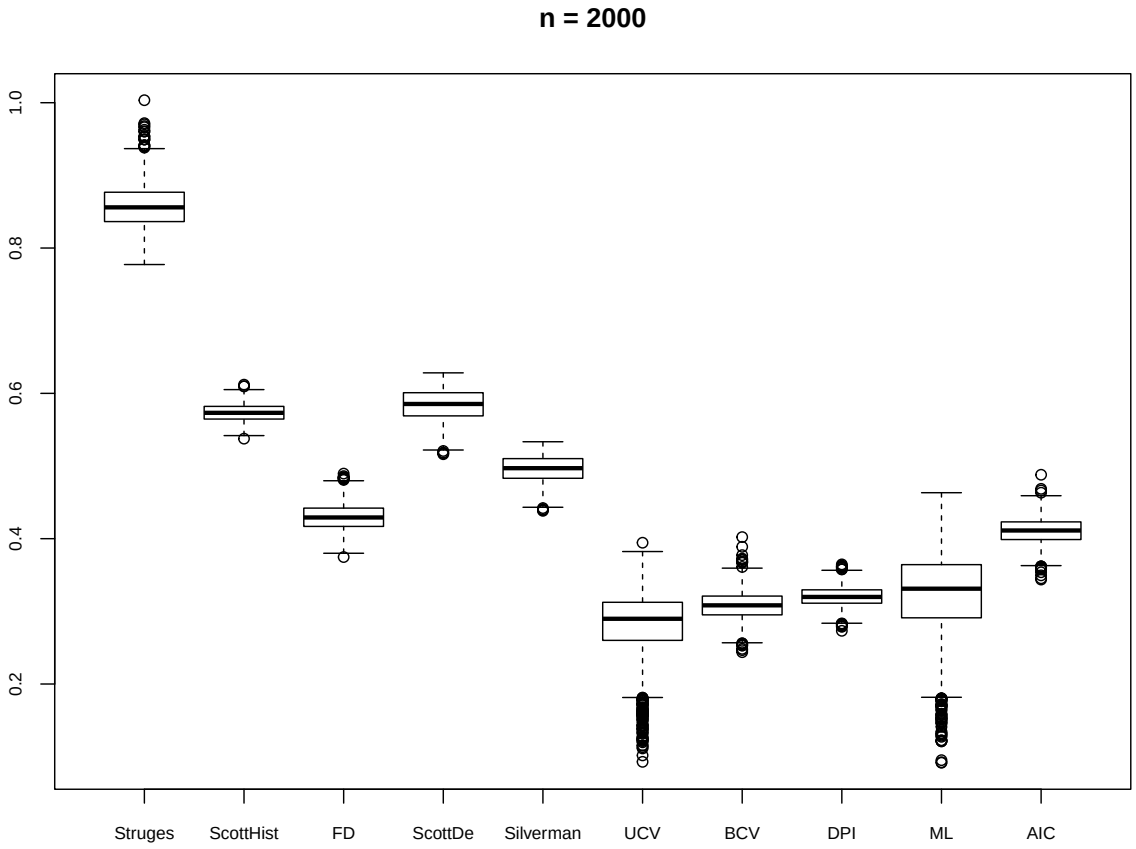

In this case also, the $h$ values obtained using different methods are very different. Let's look at the expected $h$ values.

```
##   Struges ScottHist        FD   ScottDe Silverman       UCV       BCV       DPI
## 1.0975469 1.4596370 1.1078315 1.0184086 0.8646865 0.6632560 1.1563211 0.7866098
##        ML       AIC
## 0.7257442 0.9793768
```

As expected the values are quite different and they have different types of variability also.

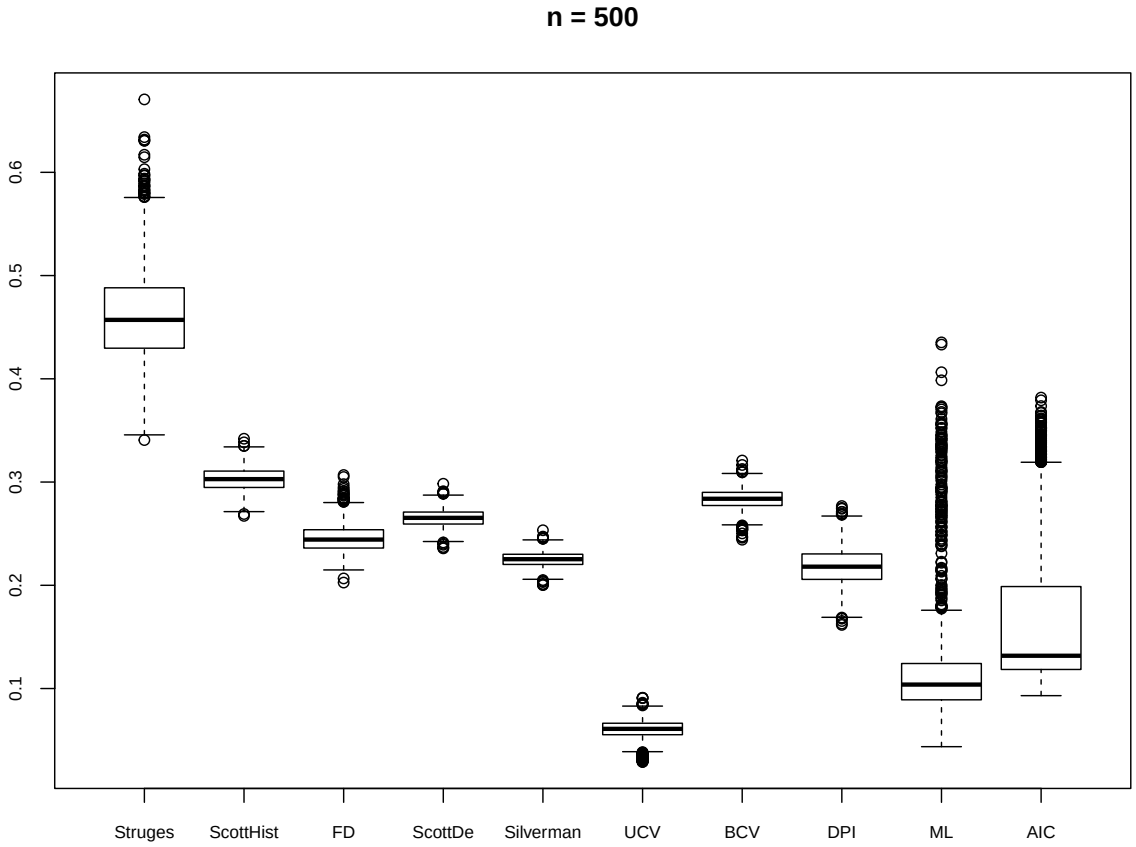
We have drawn the Boxplot for n = 500,2000 also.


n = 500
1.0
0.8
0.6
0.4
0.2
Struges
ScottHist
FD
ScottDe
Silverman
UCV
BCV
DPI
ML
AIC




Here we are getting similar type of observations. Our ML method may produce good results (in terms of different measures), but the $h$ values are dispersed.

## 9.3 Four Modal Claw Distribution :



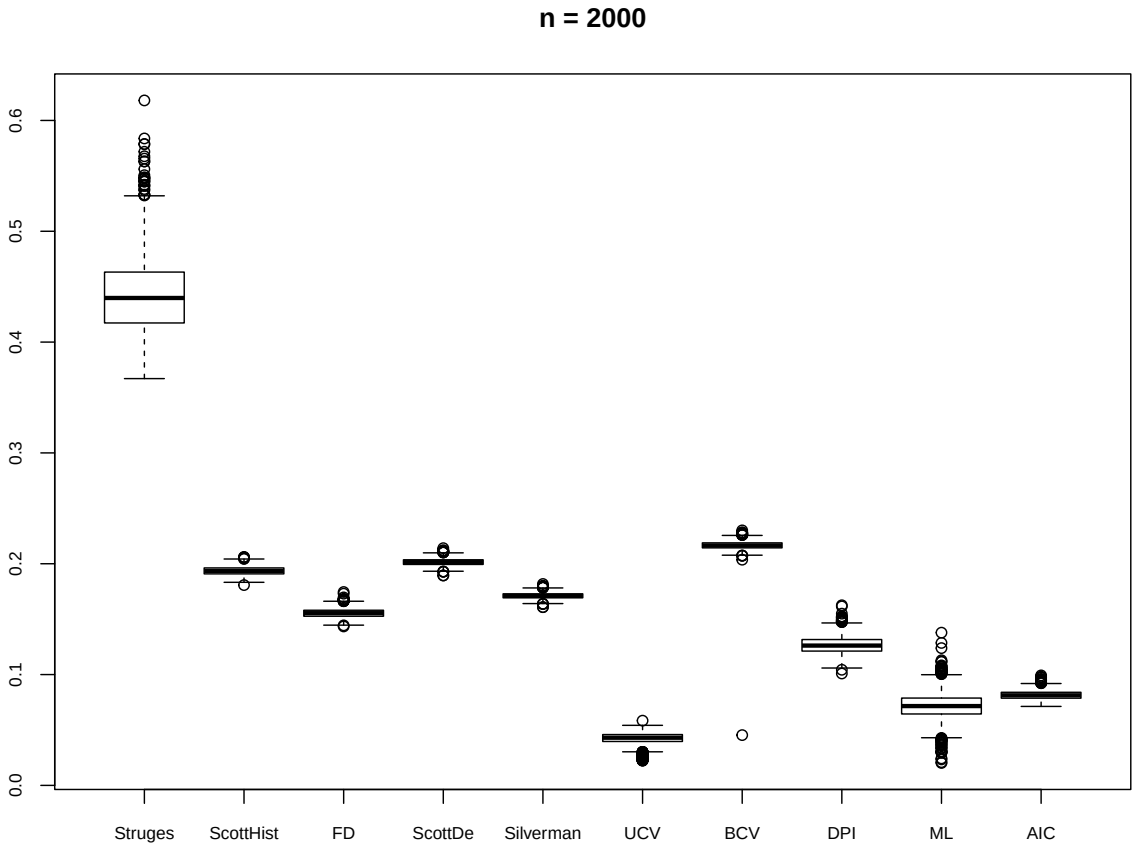

From the above Boxplot, we can see that **ML** method has very outlier $h$ values. While for other methods $h$ values are not very dispersed ! Let's see the expected $h$ values.

```
##   Struges ScottHist        FD   ScottDe Silverman       UCV       BCV       DPI
## 1.0975469 1.4596370 1.1078315 1.0184086 0.8646865 0.6632560 1.1563211 0.7866098
##        ML       AIC
## 0.7257442 0.9793768
```

And the corresponding S.D's are given by -

```
##   Struges ScottHist        FD   ScottDe Silverman       UCV       BCV       DPI
## 0.0644054 0.1037517 0.1825634 0.1284410 0.1090537 0.2161888 0.1240452 0.1490568
##        ML       AIC
## 0.2408232 0.1594684
```

We have drawn the Boxplot of $h$ values for n = 2000.




Again we are getting similar type of observations.

# 10 Conclusion :

From the above discussion, we can see that the proposed methods are performing well in general and it is performing better than existing methods in some cases. In most of the cases, graphical verification indicates the method with least IABSL. The density estimation becomes difficult when the distribution has several local modes. For such a distribution if we have less number of observation, then it becomes very difficult. Almost all methods fail in that case. Also, it is very difficult to estimate density for extreme outlier distributions. The different methods of comparison we have used, it seems that KL Divergence and IABSL are more appropriate criteria. Lastly, we must agree that **there is no method that works best for every situation**.

# 11 References :


- Bernard Silverman : Density Estimation for Statistics and Data Analysis
- bandwidth: Bandwidth Selectors for Kernel Density Estimation in R.

- nclass: Compute the Number of Classes for a Histogram in R.
- density: Kernel Density Estimation in R.
- hist: Histograms in R.
- wavethresh: Wavelets Statistics and Transforms in R.
- nor1mix: Normal aka Gaussian (1-d) Mixture Models in R.