University of Edinburgh, School of Mathematics
Statistical Research Skills
Assignment 3 - Simulation Report

Johnny Lee, s1687781

26th April 2022

1. Introduction

Density estimation is to form an estimate based on the original data of a probability density function which is popular in various fields including statistics and economics. In this report, we aim to investigate the performance of various density estimators and this report consists of two main parts. In the first part, we aim to perform one-shot experiment on kernel density estimator and compare against its other competitors such as orthogonal series estimator and penalised kernel density estimator. Hence, we will describe the preliminary experiment in the first part. In the second part, we will conduct a Monte Carlo simulation study for different sample sizes on the previously suggested methods. Thus, we will evaluate the integrated squared error (ISE) for different cases of sample sizes. Then we will end this report in the conclusion.

2. Preliminary Experiment

This section consists of the first part of the report where we conduct the preliminary experiments with 3 different density estimators. We will introduce the detailed methodology with its mathematical equation. With random data generation from univariate normal distribution and beta distribution, we will conduct one-shot experiment on these estimators.

2.1 Methodology

2.1.1 Kernel Denisty Estimator

Let $X_1, \ldots, X_n \overset{\text{iid}}{\sim} f$. The kernel density estimator of f is defined by (Silverman, 2018)[4] as

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

where $K_h(x) = \frac{K(x)}{h}$, K is a kernel and $h > 0$ is a parameter controlling smoothness of the estimate. Thus, we will perform an univariate density estimate with default kernel and bandwidth settings in this experiment.

2.1.2 Orthogonal Series Estimator

With reference to (Kreyszig, 1991)[3], Kreyszig introduced the orthogonal series estimator using normalised Hermite polynomials. These polynomials form a orthonormal sequence for the univariate case as below,

$$\hat{f}(x) = \frac{1}{(2^i i! \sqrt{x})^{\frac{1}{2}}} \exp(-x^2/2) H_i(x)$$

and

$$H_0(x) = 1, \quad H_i(x) = (-1)^i \exp(x^2) \frac{d^i}{dx^i} \exp(-x^2)$$

where $H_i(x)$ is called the *Hermite polynomial of order i* and orthogonal with respect to the probability density function. Furthermore, we can further simplify the process as Kreyszig showed its properties in a recurrence relation as below,

$$H_{i+1}(x) = 2xH_i(x) - H_i'(x), \quad H_i'(x) = 2iH_{i-1}(x)$$

Thus, we defined a new function to proceed to the iteration above to obtain the series of estimation.

```
# The implementation of the orthogonal series estimator, using Hermite series
OS_Hermite <- function(x, data){
  hermite <- function(x, m){
    if(m == 0){
      1
    }
    else if(m == 1){2*x}
    else{
      a <- 1
      b <- 2*x
      for(i in 2:m){
        c <- 2*x*b - 2*(i - 1)*a
        a <- b
        b <- c
      }
      c
    }
  }
  #computing normalised Hermite functions
  phi <- function(x, m){
    (hermite(x, m)/exp((x^2)/2))/sqrt((2^m)*factorial(m)*sqrt(pi))
  }
  n <- length(data)
  val <- 0
  for(m in 0:30){
    c <- sum(sapply(data, phi, m = m))/n
    val <- val + c*phi(x, m)
  }
  val
}
```

2.1.3 Penalised Kernel Density Estimator

(Kauermann et al, 2009)[2] introduces the penalised likelihood approach

$$\hat{f}(x) = \sum_{n=-m}^{m} c_n \phi_n(x),$$

where $\phi_n(x)$ is the basis densities. Then the weight c_n is parameterised as follow

$$c_n(y) = \frac{\exp(\beta_n)}{\sum_{n=-m}^{m} \exp(\beta_n)}$$

with $\beta_0 = 0$ and $\boldsymbol{\beta} = \beta_{-m}, \ldots, \beta_{-1}, \beta_1, \ldots \beta_m$ so that $\int f(x)dx = 1$. (Deng et al, 2011)[1] then further suggested the simplified kernel approach that looks similar to kernel density estimator's expression. The

following expression shows the penalised approach for nonparametric density estimation for univariate case,

$$\hat{f}(x) = \frac{1}{m}\sum_{n=1}^{m} K\left(\frac{x-\mu_n}{h}\right)$$

where μ_i is a hyperparameter known as knots being placed on an equally spaced locations on the domain of the dataset. (Gu, 2011)[5] introduced a package `gss` which uses a penalised likelihood approach for nonparametric density estimation. With the functions `ssden()` and `dssden()`, we can estimate the kernel density.

2.2. Data Generating Process

```
# Generate data for the experiment
n <- 1000
set.seed(1)
# The sample one
sample1 <- rnorm(n, mean = 0, sd = 1)
# #The sample two
sample2 <- rbeta(n, shape1 = 2, shape2 = 4)
```

We randomly generated $X \sim N(0,1)$ with 1000 samples. These values are chosen on a random basis and we are generating the data in a random process with a fixed random seed. Thus, we define the two scenarios as follow,

- $X \sim N(0,1), \quad f(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2})$
- $X \sim Beta(2,4), \quad f(x) = \frac{\Gamma(6)x(1-x)^3}{\Gamma(2)\Gamma(4)}$

2.3. One-shot Experiments

Now we proceed to one-shot experiment on two randomly generated distribution in the previous section. First of all, one-shot experiment is a pre-experimental design which performs the same algorithm on a single simulated data set. With the outcome, we can design a more sophisticated experiment in the later part and choose suitable hyperparameters for efficient and accurate computation.

2.3.1. Scenario : Normal Distribution

```
set.seed(1)
# The density functions for the distributions of the simulated data
#density function for the normal distribution
density1 <- function(x){
  mu <- 0; sigma <- 1
  exp(-(x - mu)^2/(2*sigma^2))/(sqrt(2*pi)*sigma)
}
#computing density plot for each estimator
sample1.fit <- ssden( ~ sample1, domain = data.frame(sample1))
xx <- seq(min(sample1), max(sample1), len = n)
hist(sample1, freq = FALSE, main = "",
     xlab = "X", ylab = "Density", xlim = c(-3, 3), ylim = c(0, 0.5))
lines(density(sample1, from = min(sample1), to = max(sample1)),
      col = "green", lwd = 1.6)
lines(xx, density1(xx), type = "l", col = "black", lty = 2, lwd = 1.6)
lines(xx, sapply(xx, OS_Hermite, data = sample1), col = "red", lwd = 1.6)
lines(xx, dssden(sample1.fit, xx), type = "l", col = "orange", lwd = 1.6)
```

```
legend(x = "topright",
       legend = c("True Density",
                  "Kernel Density Estimator",
                  "Orthogonal Series Estimator",
                  "Penalised Kernal Density Estimator"),
       lty = c(2, 1, 1, 1),
       col = c("black", "green", "red", "orange"),
       lwd = 1.8, bty = "n")
```

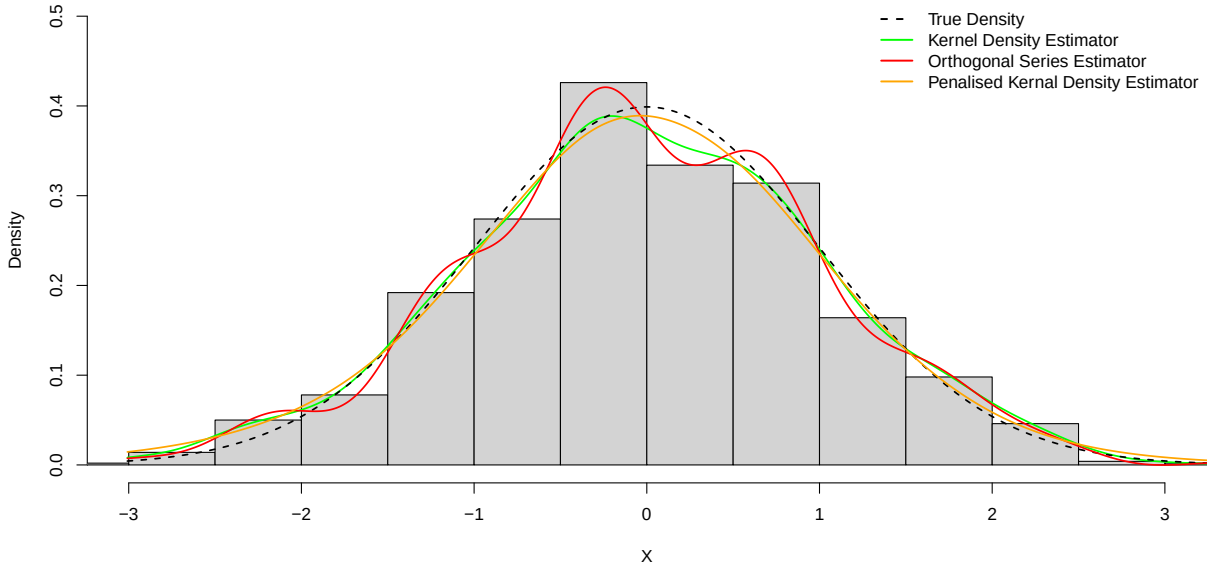

Figure 1: One-shot Experiment on Normal distribution

Let us look at Figure 1. The figure illustrates the true density with a histogram on the actual data set that we generated. Then we plotted 3 additional lines that represent each of the estimators. We can clearly observe that kernel density estimator and penalised kernel density estimator are having similar shapes to the true density. However, penalised kernel density estimator resembles the true density line the most compared to others. For the kernel density estimator, the peak is slightly on the right side of the distribution. On the other hand, orthogonal series estimator (red) is shown to have a two hump and being less smoother compared to the rest. Overall, we can conclude that penalised kernel density estimator is the best among others in the one-shot experiment on normal distribution.

2.3.2. Scenario 2 : Beta Distribution

Looking at Figure 2, we illustrated the true density with a histogram on the actual data set that we generated from. Then we plotted 3 additional lines that represent each of the estimators. By looking at the plot we can clearly observe all curves are smoothly plotted. The kernel density estimator and penalised kernel density estimator both resembles the true density line. On the other hand, orthogonal series estimator is shown to have the lowest peak with its peak centered on the right side and reasonably apart from the true density. Thus, for beta distribution we can conclude that kernel density estimator and penalised kernel density estimator work well. As we cannot distinguish the performance with beta distribution, we will move on to the Monte Carlo simulation to validate our experiment to discuss which estimators have the best performance.

```r
# The visualization based on sample two
# The density function for the distribution of sample two
density2 <- function(x){
  alpha <- 2; beta <- 4
  x^(alpha - 1)*(1 - x)^(beta - 1)/beta(alpha, beta)
}
sample2.fit <- ssden( ~ sample2, domain = data.frame(sample2))
xx <- seq(min(sample2), max(sample2), len = n)
hist(sample2, freq = FALSE, main="",
     xlab = "X", ylab = "Density", xlim = c(0, 0.8), ylim = c(0, 2.8))
lines(density(sample2, from = min(sample2), to = max(sample2)),
      col = "green", lwd = 1.6)
lines(xx, density2(xx), type = "l", col = "black", lty = 2, lwd = 1.6)
# axis(side = 1, at = seq(min(sample2), max(sample2), n))
lines(xx, sapply(xx, OS_Hermite, data = sample2), col = "red", lwd = 1.6)
lines(xx, dssden(sample2.fit, xx), type = "l", col = "orange", lwd = 1.6)
legend(x = "topright",
       legend = c("True Density",
                  "Kernel Density Estimator",
                  "Orthogonal Series Estimator",
                  "Penalised Kernal Density Estimator"),
       lty = c(2, 1, 1, 1),
       col = c("black", "green", "red", "orange"),
       lwd = 1.8, bty = "n")
```

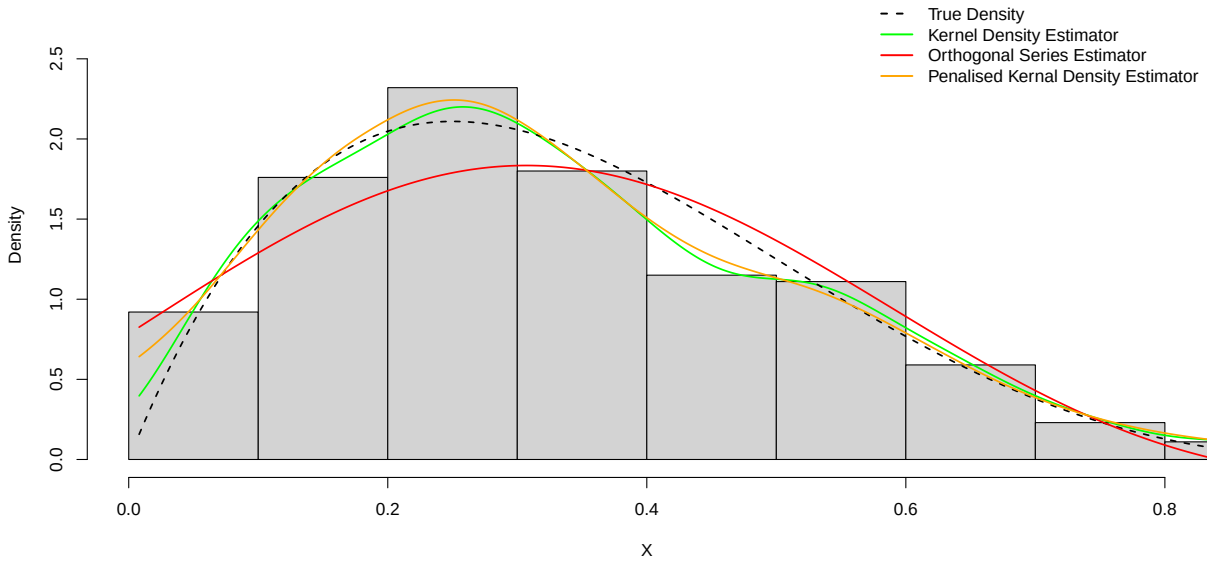

Figure 2: One-shot Experiment on Beta distribution

2.4. Strength & Weakness

2.4.1. Strength

Now lets discuss the strength of each estimator. All three estimators are known to produce a smooth density estimates where it is evident in the plot above. The kernel weight K is a unimodal probability density function that is symmetric, then the density estimate is guaranteed to be a density. Unlike the kernel density estimator methods, orthogonal series estimation does not keep the original data for evaluating the original data sets.

2.4.2 Weakness

Kernel density estimates, are influenced by the choice of kernels and bandwidth where a higher bandwidth result in shallow kernel which distant points can contribute. Thus, leading to biasness of the estimates near the boundaries. Both kernel density estimates and penalised kernel density estimates are nonparametric approaches which rely on the assumption that the function must be twice differentiable. Orthogonal series estimation takes longer time in convergence with larger sample size. It is also not useful when the distribution contains shape parameter as it cannot create the orthonormal base independently of the regressors.

3. Monte Carlo Simulation Study

Now we move onto the Monte Carlo Simulation Study, where we repeat the one-shot experiment R times, for different simulated dataset. Thus, we simulate data $X_1, \ldots, X_n \overset{\text{iid}}{\sim} f_\theta$, for a fixed $\theta \in \Theta$. Our goal in this section is to assess the performance of an estimator,

$$\hat{\theta}_n = \hat{\theta}_n(X_1, \ldots, X_n)$$

at recovering the true θ, over R simulated data sets. Hence, we repeated this process with $R = 100$ times on the randomly generated $X \sim N(0, 1)$ on sample sizes with 250, 500 and 1000. To that, we computed the integrated squared error,

$$ISE = \int \{\hat{f}(x) - f(x)\}^2$$

to compare the performances of each density estimator on different sample sizes. In addition, we computed the mean integrated squared error (MISE), and illustrated the results in figures below. Note that for the reproduceability, we fixed a random seed in all of our experiments.

```
set.seed(1)
# Initialize some variables
mu <- 0; sigma <- 1
repeat.no <- 100
n <- c(250, 500, 1000)
df <- 5
ise <- list()
ise$second <- ise$first <- data.frame("kernel" = rep(0, repeat.no),
                                      "OS" = rep(0, repeat.no),
                                      "pkd" = rep(0, repeat.no))
ise$third <- ise$first

# The density function of the distribution of the sample
density_sam1 = function(x){
  exp(-(x - mu)^2/(2*sigma^2))/(sqrt(2*pi)*sigma)
}
#Monte Carlo Simulation with sample size n=250
for(i in 1:repeat.no){
  sample1 <- rnorm(n[1], mean = mu, sd = sigma)
```

```
  sample.fit <- ssden( ~ sample1, domain = data.frame(sample1))
  #computing ISE for the kernel density estimator
  ker_den <- density(sample1)
  interval <- ker_den$x[-1] - ker_den$x[-length(ker_den$x)]
  xx <- (ker_den$x[-1] + ker_den$x[-length(ker_den$x)])/2
  yy <- (ker_den$y[-1] + ker_den$y[-length(ker_den$y)])/2
  ise$first$kernel[i] <- sum((yy - density_sam1(xx))^2*interval)
  #computing ISE for the orthogonal series estimator
  ise$first$OS[i] <- integrate(function(x){
    (OS_Hermite(x, data = sample1) - density_sam1(x))^2},
    lower = min(sample1), upper = max(sample1))$value
  #computing ISE for the penalised kernel density estimator
  ise$first$pkd[i] <- integrate(function(sample1){
    (dssden(sample.fit, sample1) - dnorm(sample1, mean = 0, sd = 1))^2},
    lower = min(sample1), upper = max(sample1))$value
}

# The Monte Carlo Simulation with sample size n=500
for(i in 1:repeat.no){
  sample1 <- rnorm(n[2], mean = mu, sd = sigma)
  sample.fit <- ssden( ~ sample1, domain = data.frame(sample1))
  #computing ISE for the kernel density estimator
  ker_den <- density(sample1)
  interval <- ker_den$x[-1] - ker_den$x[-length(ker_den$x)]
  xx <- (ker_den$x[-1] + ker_den$x[-length(ker_den$x)])/2
  yy <- (ker_den$y[-1] + ker_den$y[-length(ker_den$y)])/2
  ise$second$kernel[i] <- sum((yy - density_sam1(xx))^2*interval)
  #computing ISE for the orthogonal series estimator
  ise$second$OS[i] = integrate(function(x){
    (OS_Hermite(x, data = sample1) - density_sam1(x))^2},
    lower = min(sample1), upper = max(sample1))$value
  #computing ISE for the penalised kernel density estimator
  ise$second$pkd[i] <- integrate(function(sample1){
    (dssden(sample.fit, sample1) - dnorm(sample1, mean = 0, sd = 1))^2},
    lower = min(sample1), upper = max(sample1))$value
}

#Monte Carlo Simulation with sample size n=1000
for(i in 1:repeat.no){
  sample1 <- rnorm(n[3], mean = mu, sd = sigma)
  sample.fit <- ssden( ~ sample1, domain = data.frame(sample1))
  # The calculation of the ISE for the kernel density estimator
  ker_den <- density(sample1)
  interval <- ker_den$x[-1] - ker_den$x[-length(ker_den$x)]
  xx <- (ker_den$x[-1] + ker_den$x[-length(ker_den$x)])/2
  yy <- (ker_den$y[-1] + ker_den$y[-length(ker_den$y)])/2
  ise$third$kernel[i] <- sum((yy - density_sam1(xx))^2*interval)
  # The calculation of the ISE for the penalised kernaldensity estimator
  ise$third$OS[i] <- integrate(function(x){
    (OS_Hermite(x, data = sample1) - density_sam1(x))^2},
    lower = min(sample1), upper = max(sample1))$value
  ise$third$pkd[i] <- integrate(function(sample1){
    (dssden(sample.fit, sample1) - dnorm(sample1, mean = 0, sd = 1))^2},
```

```
    lower = min(sample1), upper = max(sample1))$value
}
```

Referring to Figure 2, we plotted the boxplots of three different estimators with different sample sizes. `kernel` stands for the kernel density estimator, `OS` stands for the orthogonal series estimator and `pkd` stands for the penalised kernel density estimator. Scrutinising its general trend, we can observe that the `OS` has the largest quartile range and highest MISE values. We also observed that as the sample size, n increases, we can see that the ISE values decrease as when $n = 1000$ the ISE values are 10 times less than $n = 250$. Furthermore, penalised kernel density seems to show the shortest interquartile range for the ISE values and the lowest MISE among the others.

```
#computing the boxplot
par(mfrow=c(1,3))
boxplot(ise$first, main = "n = 250")
boxplot(ise$second, main = "n = 500")
boxplot(ise$third, main = "n = 1000")
```

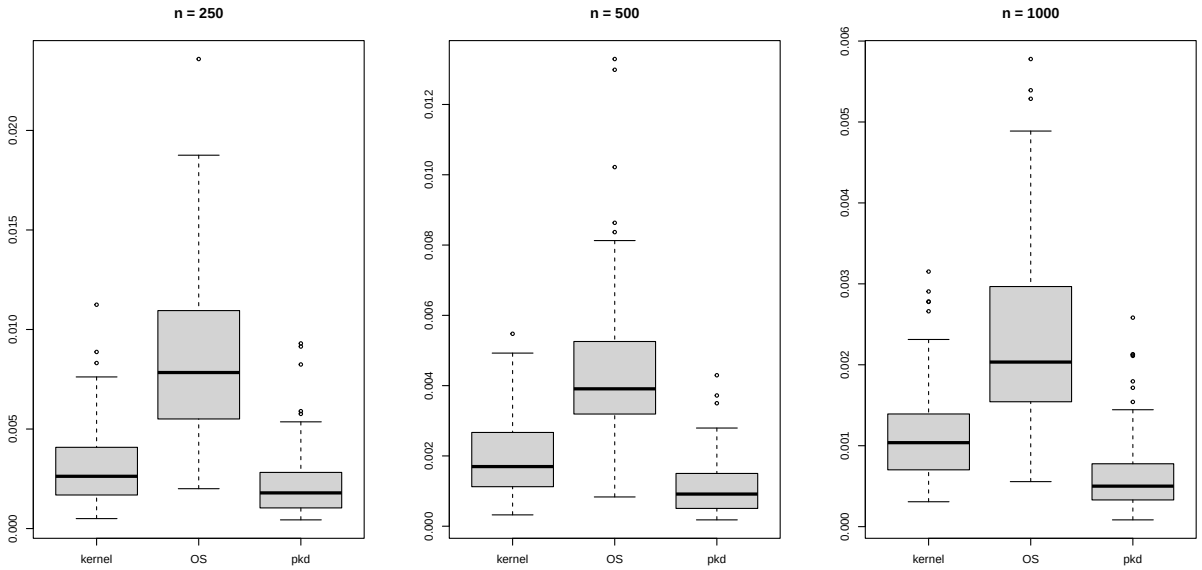

Figure 3: Boxplot of Different Estimators with Different Sample Sizes

Figure 3 further illustrates the MISE values of the three estimators where it solidifies that all estimators have lower MISE values with larger sample size and penalised kernel density estimator having the lowest MISE values. Therefore, if an accurate estimation of densities is required, a large sample size is a better choice.

```
#computing the MISE for each density estimator
ise.mean <- data.frame(kernel = c(mean(ise$first$kernel),
                                  mean(ise$second$kernel),
                                  mean(ise$third$kernel)),
                       OS = c(mean(ise$first$OS),
                              mean(ise$second$OS),
                              mean(ise$third$OS)),
                       pkd = c(mean(ise$first$pkd),
                               mean(ise$second$pkd),
                               mean(ise$third$pkd)),
                       row.names = c("first", "second", "third"))
```

```
plot(ise.mean$kernel, type = "b", col = "green", ylim = c(0,0.009),
     xaxt = "n", xlab = "sample size")
axis(1, at = 1:3, labels = c(250, 500, 1000))
lines(ise.mean$OS, type = "b", col = "red")
lines(ise.mean$pkd, type = "b", col = "orange")

legend(x = "topright",
       legend = c("Kernel Density Estimator",
                  "Orthogonal Series Estimator",
                  "Penalised Kernal Density Estimator"),
       lty = c(1, 1, 1),
       col = c("green", "red", "orange"),
       lwd = 1.8, bty = "n")
```

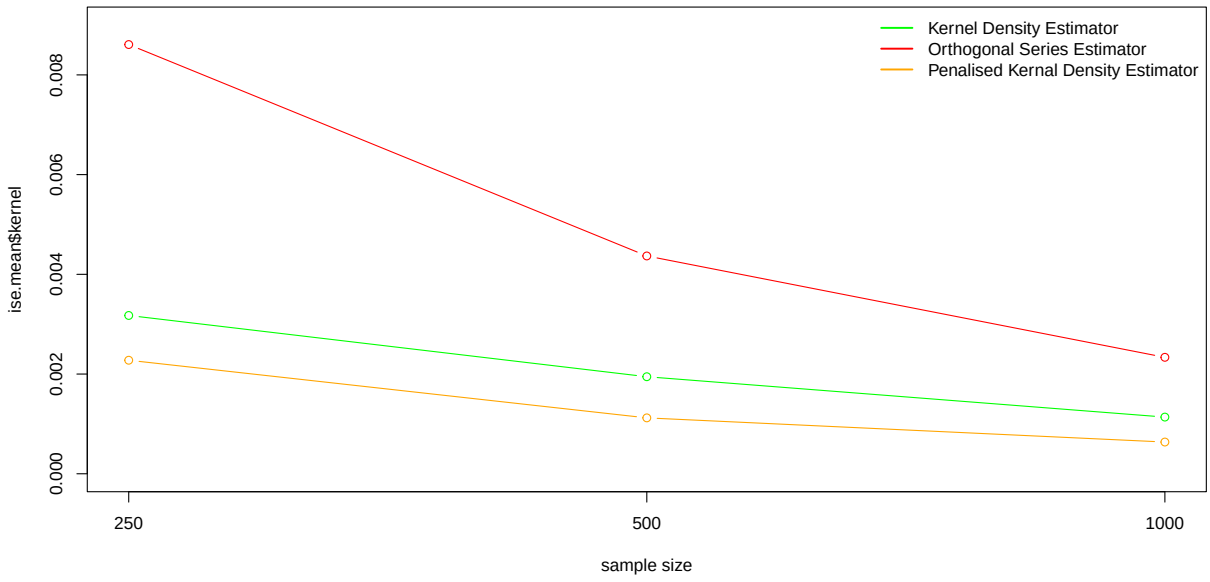

Figure 4: Mean Integrated Squared Error with Different Sample Sizes

4. Conclusion

In conclusion, we conducted experiments on three different density estimators. From one-shot experiment to Monte Carlo simulation study, penalised kernel density estimator showed the best performance. Among kernel density estimator and orthogonal series estimator, penalised kernel density estimator had the lowest MISE and showed the best fit to the true values. In the future, we aim to perform a comparison in finding the optimal bandwidth within the kernel density estimators to acquire the best estimates.

Reference

1. Deng, H. and Wickham, H., 2011. Density estimation in R. Electronic publication.
2. Kauermann, G. and Schellhase, C., 2019. Density Estimation with a Penalized Mixture Approach.

3. Kreyszig, E., 1991. Introductory functional analysis with applications (Vol. 17). John Wiley & Sons.
4. Silverman, B.W., 2018. Density estimation for statistics and data analysis. Routledge.
5. Gu, C., 2011. Smoothing spline ANOVA models: R package gss. Journal of Statistical Software, 58, pp.1-25.