# Project 2017

*Lucien Ledune*

*1 dÃ©cembre 2017*

## Introduction

In this project, we are going to discuss the estimation of the cumulative distribution and the density function of an exponentially distributed random variable. We will consider two different cases for the density estimator, 2nd order and 4th order kernels.

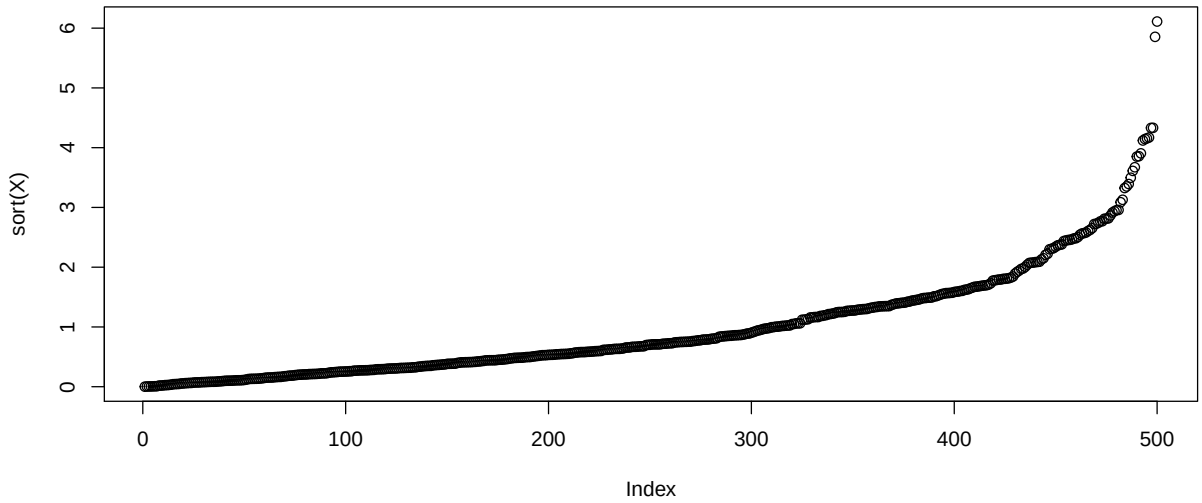
Then we will perform a Monte Carlo simulation on the IMSE of our estimators. Our goal here will be to investigate the rates of convergence of the three estimators.

## Preparation of data

In order to simulate a random variable, we will generate X points using rexp(). This will give us points following the exponential distribution.

We can plot the (ordered) estimated data here and make sure it follows an exponential distribution.

```
set.seed(856)

X = rexp(500)

plot(sort(X))
```



## Nonparametric estimation of the cumulative density function

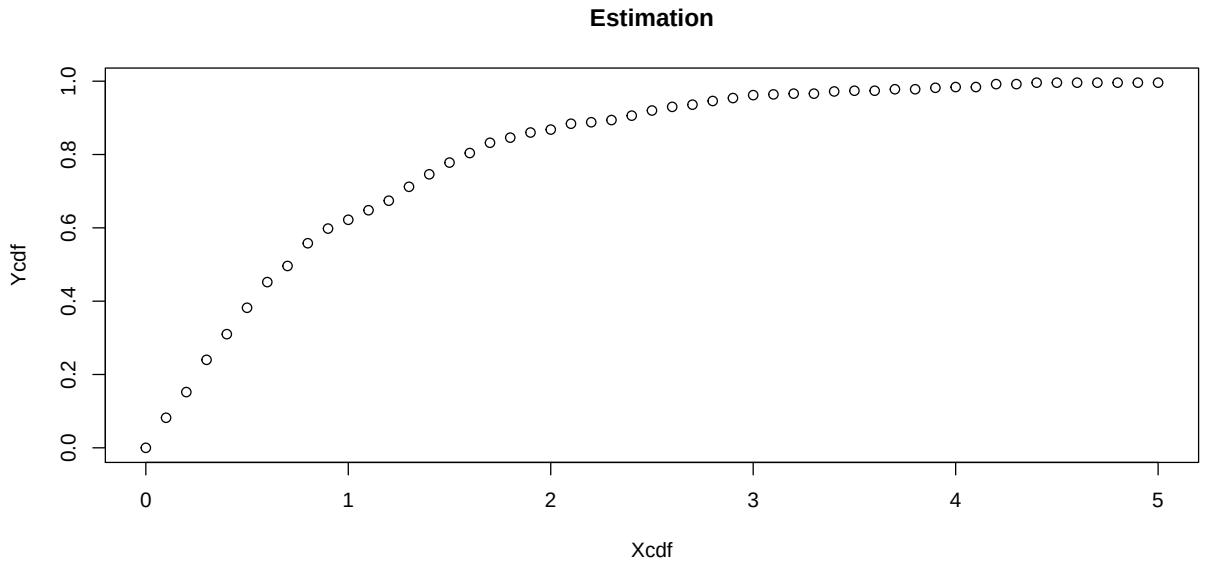
If we want to determine what the function's cumulative density function is, a simple way to do that is to use the given points to calculate the following distribution :

- 0 if $x <= X_{(1)}$
- $k/n$ if $X_{(k)} <= x < X_{(k+1)}$ for $k = 1, ..., n-1$
- 1 if $x >= X_{(n)}$

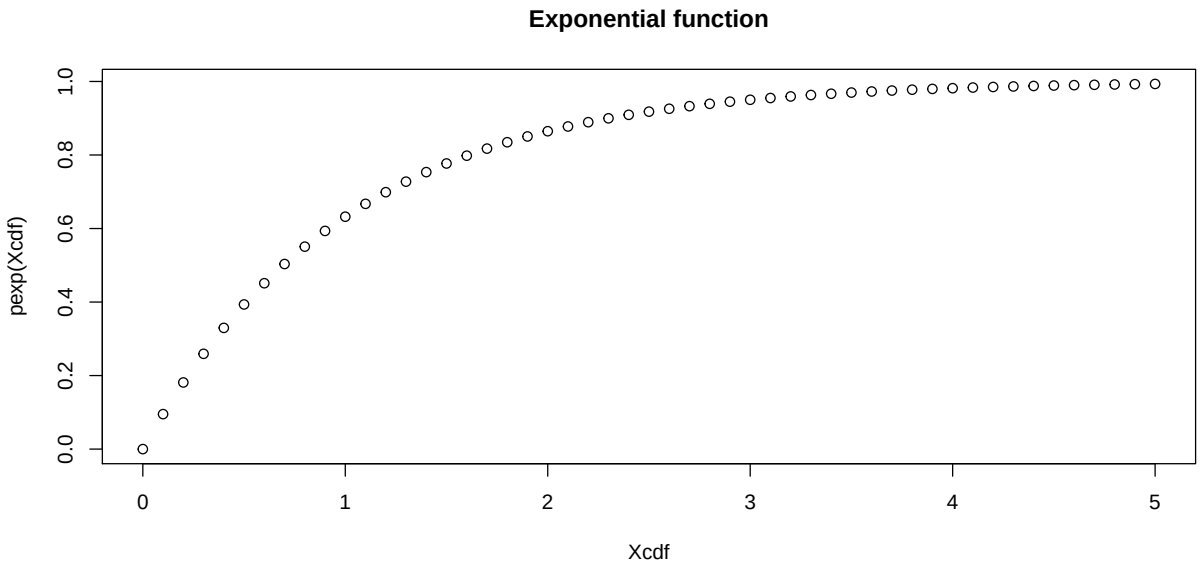
We are basically calculating the proportion of data under a point x.

Let's apply this to our function :

```
Ycdf = NULL
n = length(X)
Xcdf = seq(0,5,0.1)

for(i in 1:length(Xcdf)){
  Ycdf[i] = sum(X < Xcdf[i])/n
}

plot(Xcdf, Ycdf, main = "Estimation")
```




```
plot(Xcdf, pexp(Xcdf), main = "Exponential function")
```




We can easily observe that our 'non-parametric' estimation of the exponential cdf is close to reality. This method is very simple and intuitive, but as we can see here it is very good for estimating the cdf of an unknown function.

## Nonparametric estimation of density function

We are now going to discuss the density estimation of a function. The simple intuition behind density estimation is that we want to know how many % of the data falls between the interval [k, k+h], h being the wideness of our interval.

To get smoother result, we can use a kernel estimator, it is going to estimate the value of our point at the x points around it. We are basically giving the data points a "continuous" value, wich results in a smoother estimation.

$K = 1/nh \sum_{i=1}^{n}(x - X_i/h)$

Let's apply this to our data. First we build the Kernel function and the density estimation function. We are going to use Epanechnikov's second order kernel and gaussian kernel and compare them :

$K_{epach}(u) = 0,75 * (1 - u^2) * (|u| <= 1)$

(Gaussian kernel is simply the gaussian density formula).

We also need to find the optimal bandwith h, there is multiple methods for that but for now we are going to use the rule of thumb.

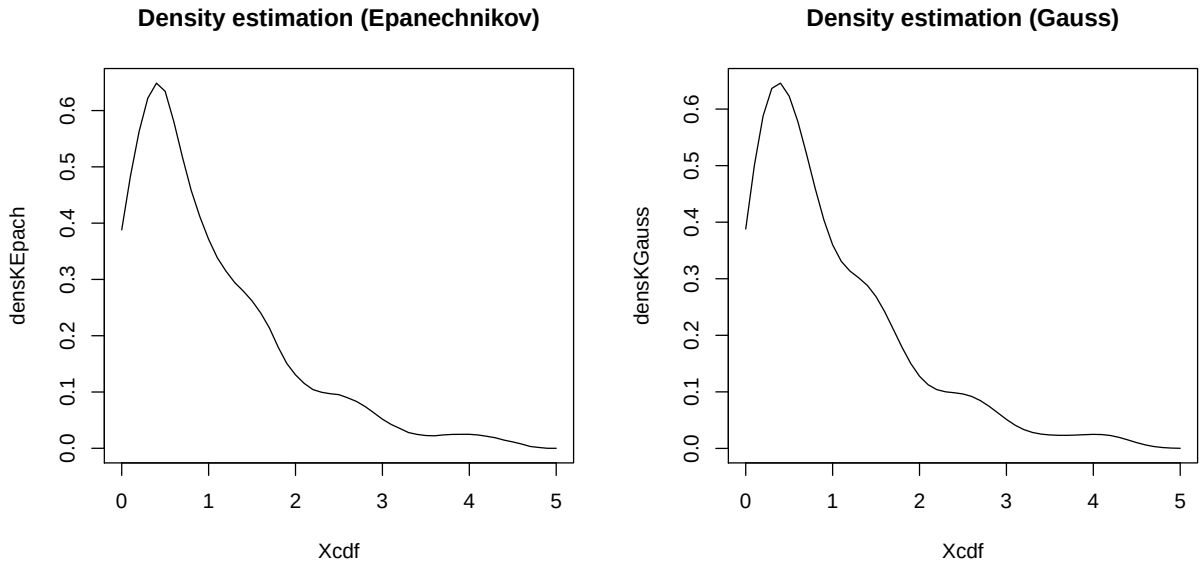
```
Kepach = function(u){ 0.75 * (1 - u^2) * (abs(u) <= 1) } #Epanechnikov kernel
Kgauss <- function(u) dnorm(u)  #Gaussian kernel


rot <- function(X, K) {#Optimal h
    RK <- integrate(function(u) K(u)^2, -Inf, Inf)$value ## integrated square kernel
    mu2 <- integrate(function(u) u^2 * K(u), -Inf, Inf)$value ## variance kernel
    R <- quantile(X, 0.75) - quantile(X, 0.25) ## interquartile range
    sig <- min(sd(X), R/1.349)
    return(((8 * sqrt(pi) * RK)/(3 * mu2^2))^0.2 * sig * length(X)^(-0.2))
```

```
}

hOptEpach = rot(X, Kepach)
hOptGauss = rot(X, Kgauss)

DensEst = function(x, X, h, K) mean(K((x - X)/h))/h #Density calculation
```

$hopt_{epach} = 0,55$ $hopt_{gauss} = 0,25$

Now we apply the density estimator function to some x points so we can plot it.

```
densKEpach = sapply(Xcdf, function(Xcdf) DensEst(Xcdf, X, hOptEpach, Kepach))
densKGauss = sapply(Xcdf, function(Xcdf) DensEst(Xcdf, X, hOptGauss, Kgauss))
par(mfrow = c(1,2))
plot(Xcdf, densKEpach, main = "Density estimation (Epanechnikov)", type = 'l')
plot(Xcdf, densKGauss, main = "Density estimation (Gauss)", type = 'l')
```




And we have a function quite similar to the real exponential value (obviously).

This estimation was done using Epanechnikov's Kernel, which is the most efficient (even if the difference of efficiency between commonly used kernels is very small).

## Higher order kernel : fourth order

We will now perform the same density estimation, but using a second order kernel. Those are often called 'bias reduction kernels'.

Fourth order Epanechnikov kernel :

$K = 15/8 * (1 - 7/3 * u^2) * K(u)$

```
Kepachfourth = function(u){ 15/8 * (1 - 7/3 * u^2) * Kepach(u) } #Epanechnikov kernel 4th order
h=0.95
```

```r
densK4 = sapply(Xcdf, function(Xcdf) DensEst(Xcdf, X, h, Kepachfourth))
plot(Xcdf, densK4, main = "Density estimation (Epanechnikov)", type = 'l')
```