Nonparametric statistics: BE

P. Neuvial and P. Berthet

October 21, 2021

Contents

Instructions

Read the following text, execute the codes, then answer the 10 questions (11 is a bonus, rather open). You have to finish later and produce a pdf file containing your answers, comments, R-codes and output graphics.

Deadline : november 5, 2021. Please send your work by email at: philippe.berthet@math.univ-toulouse.fr

I. Introduction

Context

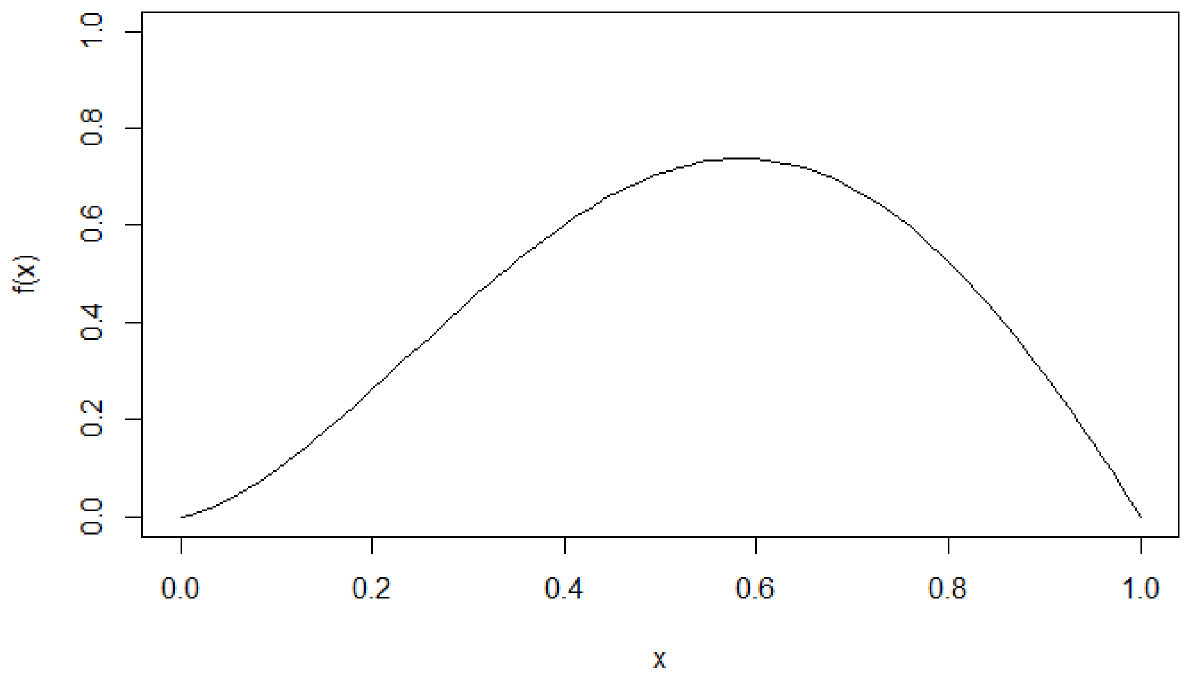
We focus on the nonparametric kernel regression estimation in a simple model signal f + noise ϵ,

$$Y_i = f(X_i) + \epsilon_i$$

with fixed design $X_1, ..., X_n$. We make use of the Mean (Integrated) Squared Error as a measure of risk.

More precisely, let consider the problem of estimating the following function:

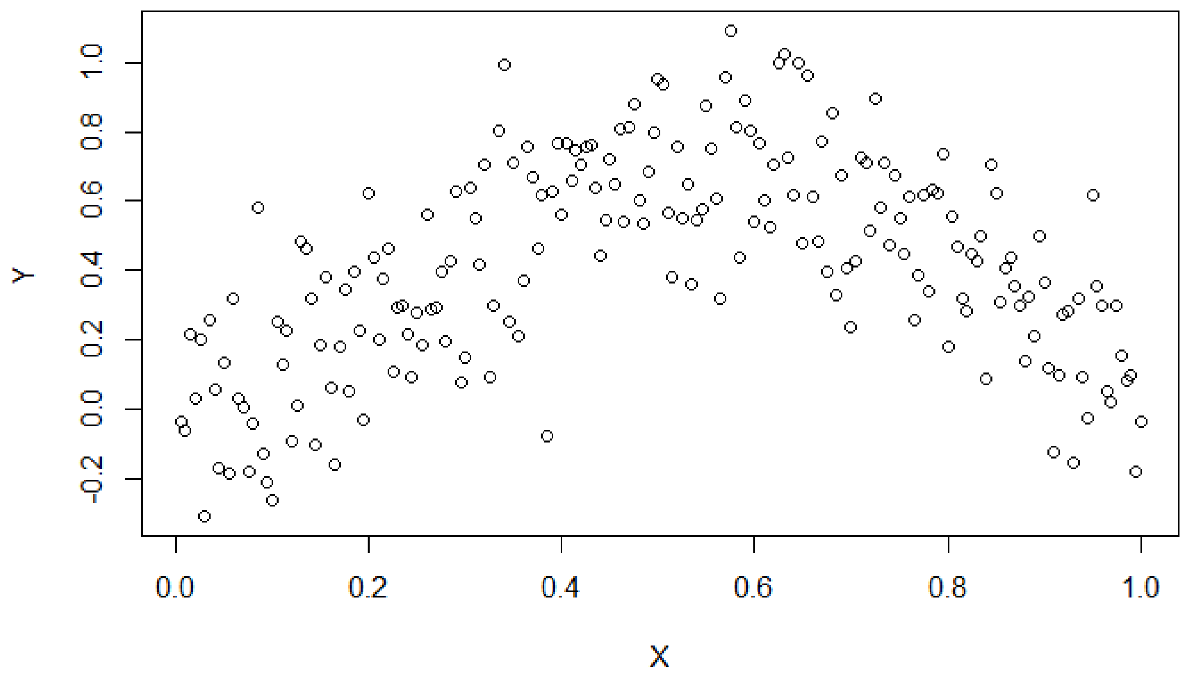
```r
f <- function(x) sqrt(x)*sin(pi*x)
curve(f, ylim = c(0, 1))
```



```r
n <- 200
X <- 1:n/n
```

The number of observations is $n = 200$ and the fixed design is equispaced : $X_i = i/n, i = 1 \dots n$. We generate a sample by adding Gaussian errors $\epsilon_1, ..., \epsilon_n$ with variance $\sigma^2 = 0.04$ to the true function f.

```r
sigma <- 0.2
sigma2 <- sigma^2
eps <- rnorm(n, sd = sqrt(sigma2))
Y <- f(X) + eps
plot(X, Y)
```



Nadaraya-Watson estimator

The Nadaraya-Watson (NW) estimator of f at $x \in [0,1]$ associated to the kernel K is:

$$\hat{f}_n(x) = \frac{\sum_{i=1}^n Y_i K\left(\frac{X_i - x}{h}\right)}{\sum_{i=1}^n K\left(\frac{X_i - x}{h}\right)},$$

where $h > 0$ is called the bandwidth and $K : \mathbb{R} \to \mathbb{R}$ is a kernel. We focus here on the Gaussian kernel.

MSE and MISE

The Mean Squared Error (MSE) at a point is defined by:

$$MSE(x) = \mathbb{E}_f\left[\left(\hat{f}_n(x) - f(x)\right)^2\right],$$

and the integrated risk, the Mean Integrated Squared Error (MISE), is defined by:

$$MISE = \mathbb{E}_f\left[\int_0^1 \left(\hat{f}_n(x) - f(x)\right)^2 dx\right]$$

We consider a discretized version of the MISE:

$$MISE^D = \mathbb{E}_f\left[\frac{1}{n}\sum_{i=1}^n \left(\hat{f}_n(X_i) - f(X_i)\right)^2\right]$$

R functions for nonpametric regression

There are two main R functions for kernel nonparametric regression:

- ksmooth implements the Nadaraya-Watson kernel estimate with rectangular or Gaussian kernel.
- locpoly implements local polynomial estimators with Gaussian kernel.

We will only use the function locpoly. It is part of the package KernSmooth, so we load this package:

```
library("KernSmooth")
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

Let begin by looking at the help pages of these functions:

```
?locpoly
```

```
## starting httpd help server ... done
```

The three most important arguments for us are x, y, bandwidth, and degree.

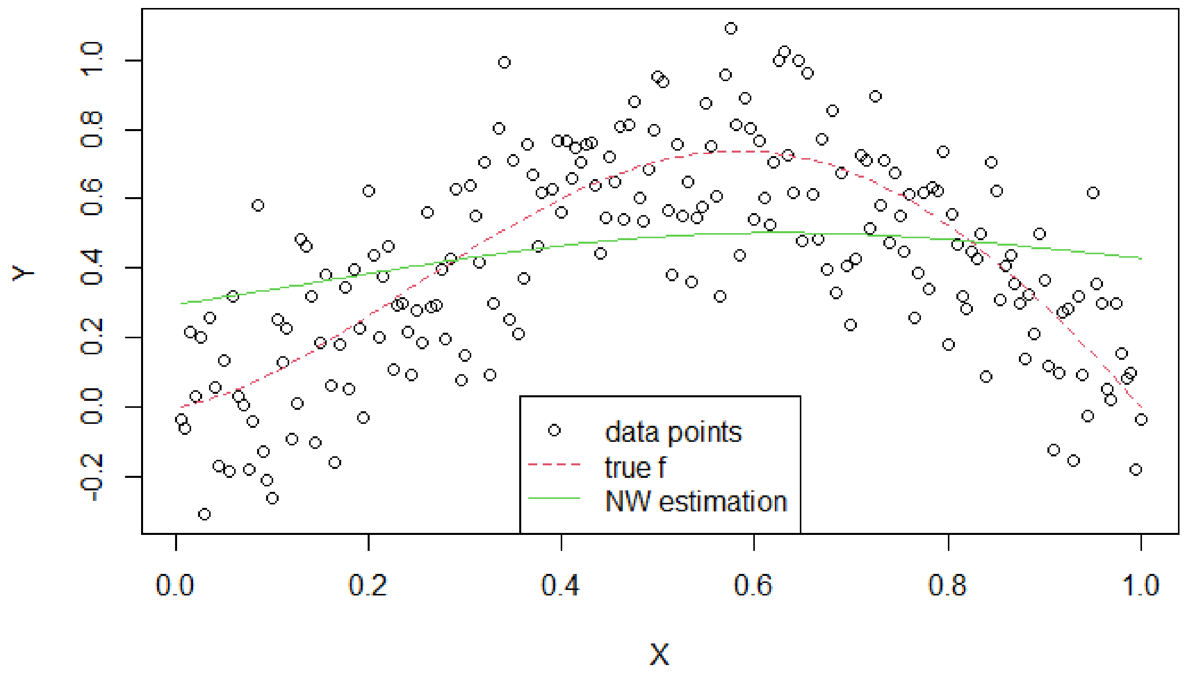
We will work with the following custom function:

```
NW <- function(x, y, bandwidth, gridsize = length(x), ...) {
  locpoly(x, y, degree = 0, bandwidth = bandwidth, gridsize = gridsize, ...)
}
```

II. Nadaraya-Watson estimator

We calculate the Nadaraya-Watson estimator and plot it along with the data points:

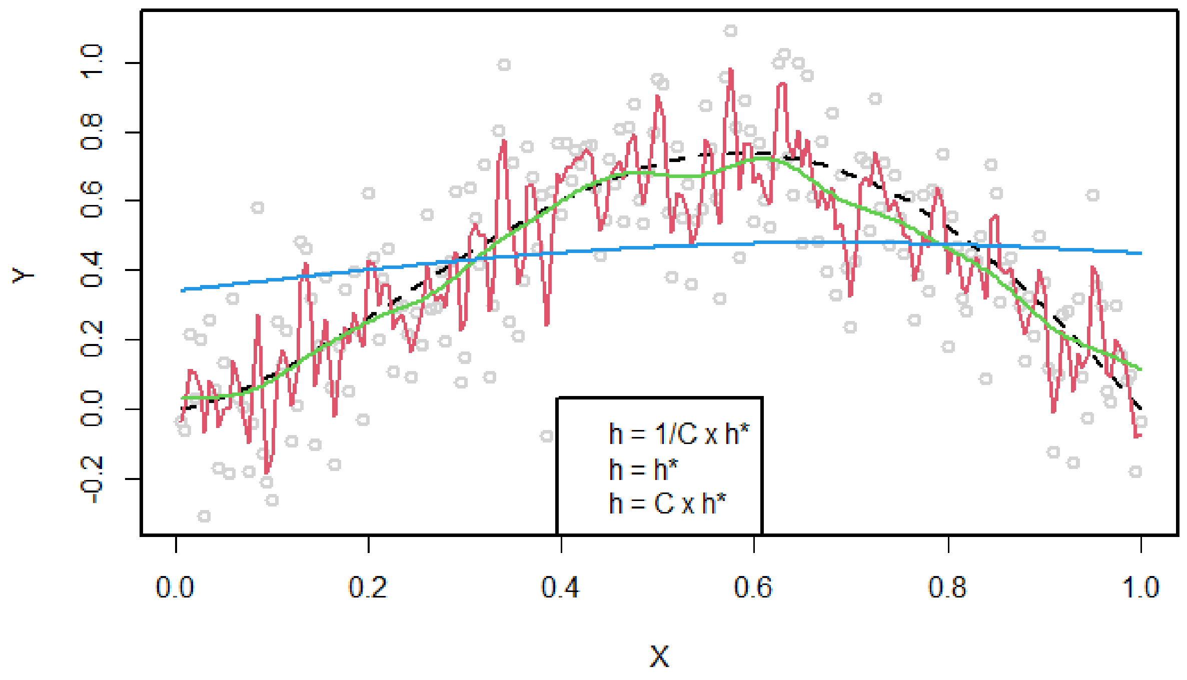
```
fit <- NW(X, Y, bandwidth = 0.3)
plot(X, Y)
curve(f, add = TRUE, col = 2, lty = 2)
lines(fit, col=3)
lgd <- c("data points", "true f", "NW estimation")
legend("bottom", lgd, col = 1:3, lty = c(NA, 2, 1), pch = c(1, NA, NA))
```



Question 1: influence of the bandwidth

Let us admit for a while that a "good choice" for the bandwidth in the NW estimator is $h^\star = 0.057$. In order to further asses the influence of the bandwidth on the quality of estimation, we compare this choice to two other choices $h = 1/C \times h^\star$ and $h = C \times h^\star$, for $C = 10$.

```r
par(lwd = 2)
plot(X, Y, col = "lightgray")
curve(f, add = TRUE, col = 1, lty = 2)
best <- 0.057
C <- 10
bwd <- c(1/C, 1, C)*best
for (kk in 1:length(bwd)) {
    fit <- NW(X, Y, bandwidth = bwd[kk])
    lines(fit, col = 1 + kk)
}
lgd <- c("h = 1/C x h*", "h = h*", "h = C x h*")
legend("bottom", lgd, col = 2:4)
```



1. Run the same code with smaller values for C.
2. Comment on the influence of the bandwidth on the quality of estimation.

III. Estimation of MSE and MISE

The goal of this section is to estimate the optimal bandwidth for the NW estimator, and justify the choice $h^\star = 0.057$ in the preceding section. We measure the quality of estimation by the discretized MISE:

$$MISE^D = \mathbb{E}_f \left[\frac{1}{n} \sum_{i=1}^{n} \left(\hat{f}_n(X_i) - f(X_i) \right)^2 \right]$$

If f was known, then we could estimate the discretized MISE by the following quantity:

$$\frac{1}{n} \sum_{i=1}^{n} \left(\hat{f}_n(X_i) - f(X_i) \right)^2$$

```r
MISE_NW <- function(bandwidth, x, y, f) {
    fit <- NW(x, y, bandwidth = bandwidth)
    mean((fit$y - f(fit$x))^2)
}
```

Question 2: theoretical bandwidth

To find the optimal bandwidth, we simply minimize `MISE_NW` as a function of h.

```r
opt <- optimize(MISE_NW, X, Y, f, interval = c(0, 1))
best <- round(opt$minimum, 3)
best
```

```
## [1] 0.057
```

1. Can the above be done in practice, i.e. given only the observations `X` and `Y`?
2. Estimate the above optimal bandwidth from another simulation run with $n = 2,000$. Are these results consistent with the theoretical upper bounds obtained during the course?
   [*Hint: recall the form of the optimal bandwidth for a kernel of order 1*].
3. Without doing further simulation runs, estimate the optimal bandwidth for $n = 20,000$ and $n = 2,000,000$.

Question 3: a first estimator of the MISE

We consider the following estimator of the discretized MISE:

$$\widehat{MISE}^{D,0} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{f}_n(X_i) - Y_i\right)^2$$

As a matter of fact, the unobserved errors $_i$ in $Y_i = f(x_i) + \epsilon_i$ can be estimated by

$$\hat{\epsilon_i} = Y_i - \hat{f}_n(x_i)$$

and assuming that they are independent, centered with same variance $\sigma^2 > 0$ (here we simulate Gaussians), a natural, converging estimator of $MISE$ is $\widehat{MISE}^{D,0} = \frac{1}{n}\sum_{i=1}^{n}\hat{\epsilon_i}^2$.

Since $MISE$ is at best of order $n^{2\beta/(2\beta+1)}$ for a regularity order β, the control of $\widehat{MISE}^{D,0}$ should be carefully studied. No time for theory today, let do simulations.

This estimator can be implemented as follows:

```r
MISE_NW_0 <- function(bandwidth, x, y) {
    fit <- NW(x, y, bandwidth = bandwidth)
    mean((fit$y - y)^2)
}
```

We define a grid of 100 values for the bandwidth `h` between 0 and 0.2.

```r
hs <- 1:100/500
```

1. Calculate the above estimator for all bandwidths, and plot the result as a function of `h`.
2. Comment the results and explain why this estimator cannot be used to estimate an optimal bandwidth.

IV. The bias/variance tradeoff

Question 4: an "oracle" estimator

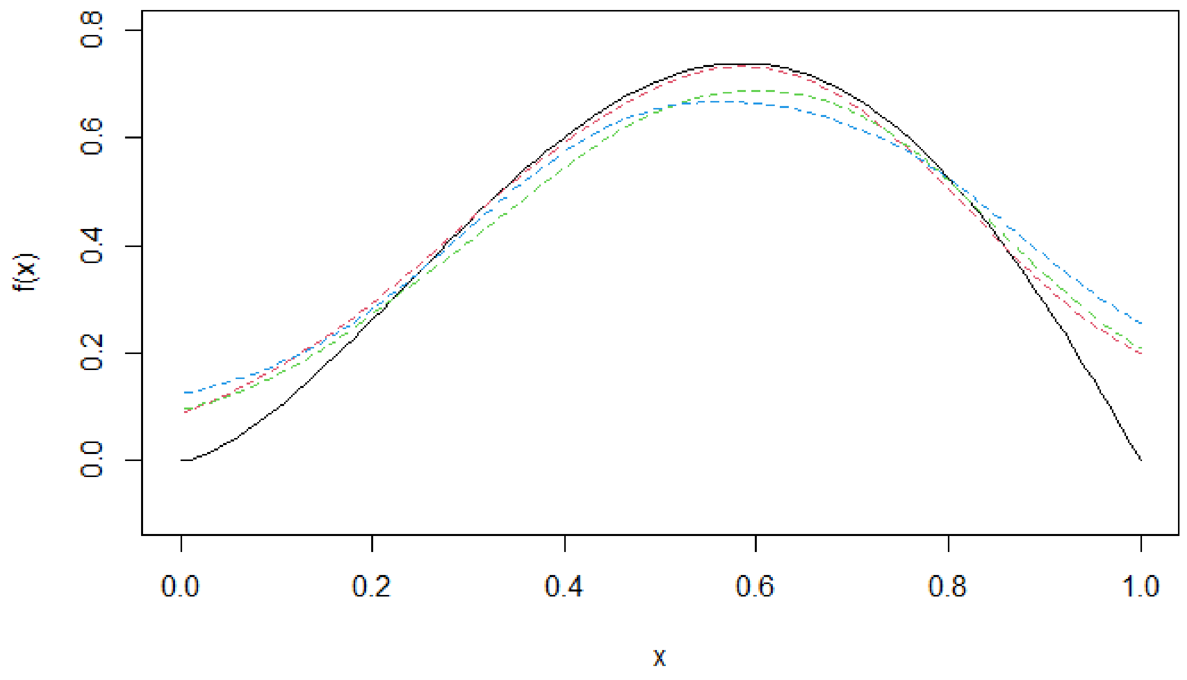
The estimator at Question 2 is based on the knowledge of f and a single realization (X, Y). To estimate the discretized MISE when the simulation model is known, we propose in this section to replace the expectation $\mathbb{E}_f$ in the definition of the discretized MISE by an empirical mean over simulation runs.

We start by defining a function that performs one simulation run and outputs the associated NW estimator:

```r
fHat <- function(bandwidth, n, sigma2, f) {
    eps <- rnorm(n, sd = sqrt(sigma2))
    X <- 1:n/n
    Y <- f(X) + eps
    fit <- NW(X, Y, bandwidth = bandwidth)
    fit$y
}
```

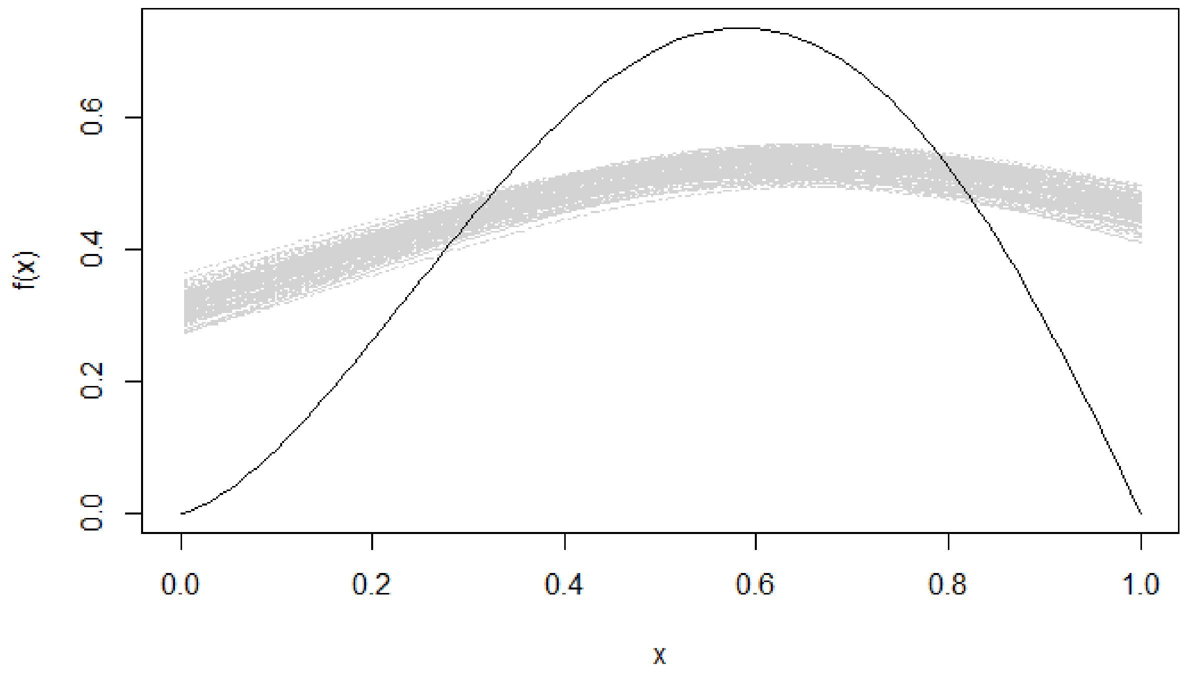
We display the result of three simulation runs:

```
curve(f, ylim = c(-0.1, 0.8))
lines(X, fHat(0.1,  n, sigma2, f), col = 2, lty = 2)
lines(X, fHat(0.1,  n, sigma2, f), col = 3, lty = 2)
lines(X, fHat(0.1,  n, sigma2, f), col = 4, lty = 2)
```



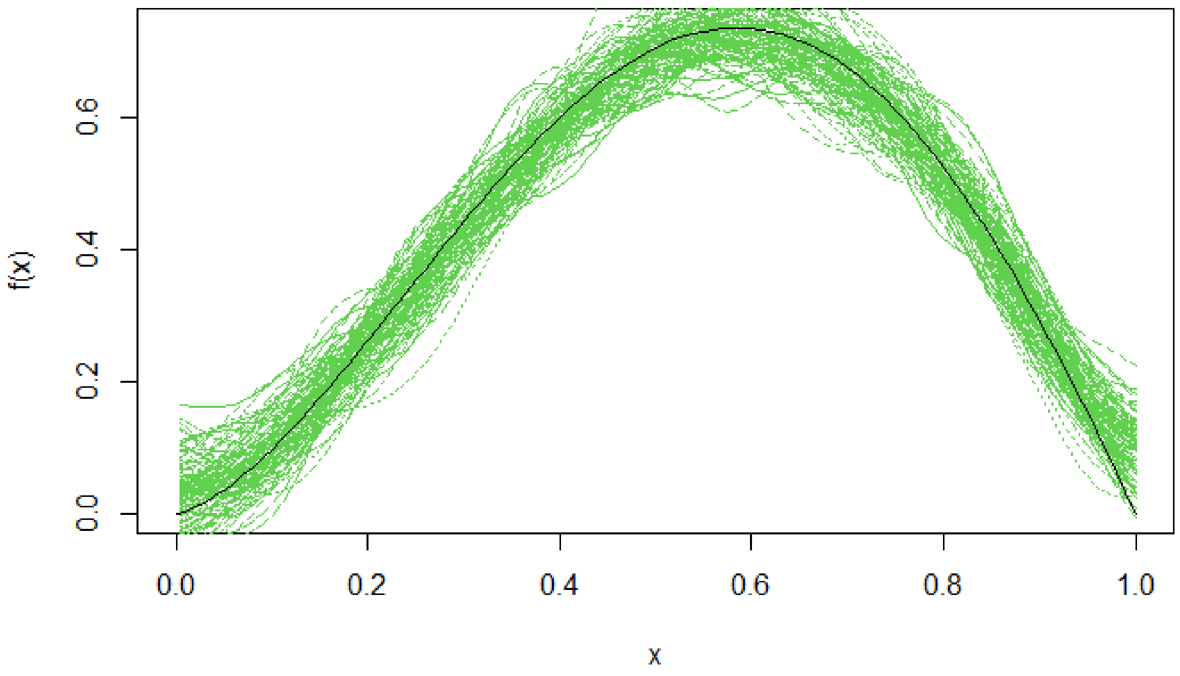
Following this idea, we generate and plot a matrix of realizations of $\hat{f}$:

```
bandwidth <- 0.3
fHatMat <- replicate(101, fHat(bandwidth, n, sigma2, f))
curve(f)
matlines(X, fHatMat, col = "lightgray")
curve(f, add = TRUE)
```



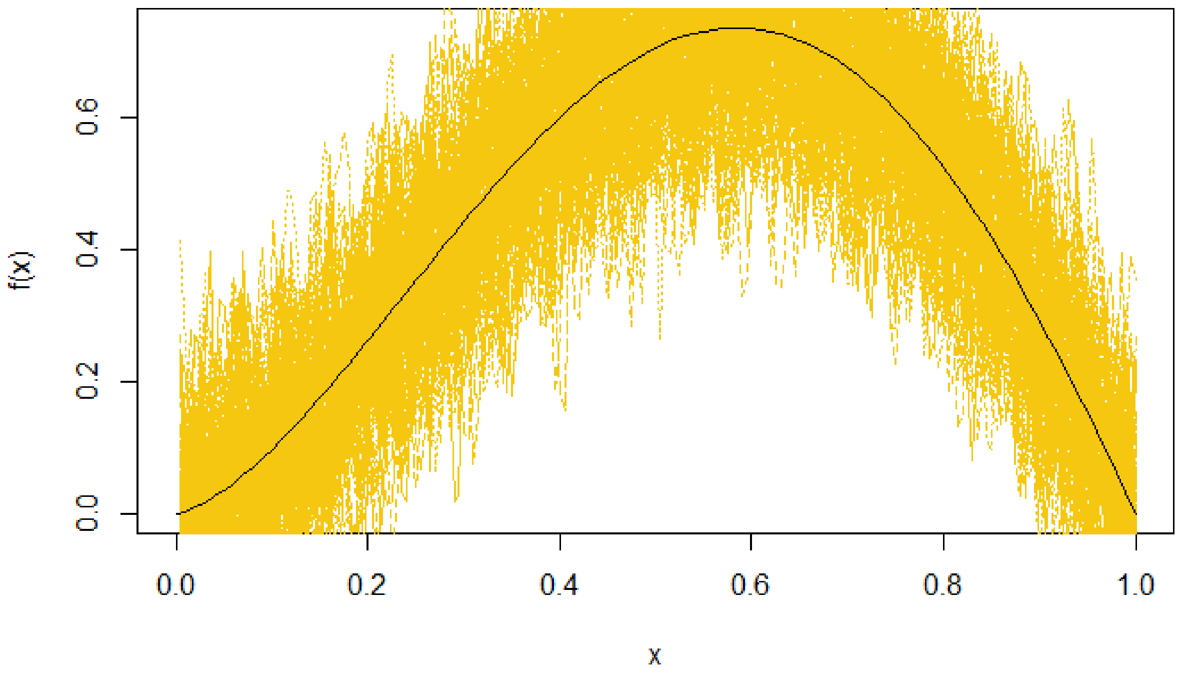
1. What represent the following distribution of green curves :

```r
bandwidth <- 0.04
fHatMat <- replicate(101, fHat(bandwidth, n, sigma2, f))
curve(f)
matlines(X, fHatMat, col = 3)
curve(f, add = TRUE)
```



2. Likewise, comment on the following distribution of yellow curves :

```
bandwidth <- 0.004
fHatMat <- replicate(101, fHat(bandwidth, n, sigma2, f))
curve(f)
matlines(X, fHatMat, col = 7)
curve(f, add = TRUE)
```



Question 5

1. Generate similar plots for other choices of bandwidths. Do these plots confirm your previous observations of the influence of the bandwidth?
2. We can form an estimate of the MSE as follows:

```
diffs <- fHatMat - f(X)            ## Note: this is a bit dangerous
diffs2 <- sweep(fHatMat, 1, f(X))  ## Safer but harder to understand
max(abs(diffs2-diffs))             ## sanity check
```

```
## [1] 0
```

```
str(rowMeans(diffs^2))
```

```
##  num [1:200] 0.0202 0.015 0.014 0.0141 0.0143 ...
```

We encapsulate the above in a function:

```
MSE <- function(bandwidth, n, sigma2, f, nRep) {
    fHatMat <- replicate(nRep, fHat(bandwidth, n, sigma2, f))
    diffs <- sweep(fHatMat, 1, f(X))
    mse <- rowMeans(diffs^2)
}
```

Recall that we have the following bias-variance decomposition:

$$MISE = \int_0^1 b^2(x)dx + \int_0^1 \sigma^2(x)dx,$$

where $b(x) = \mathbb{E}_f\left[\hat{f}_n(x)\right] - f(x)$ and $\sigma^2(x) = \mathbb{E}_f\left[(\hat{f}_n(x) - \mathbb{E}_f(\hat{f}_n(x)))^2\right]$.

What do the following quantities MSHh1,2,3 and Estim1,2,3 contain ?

```
MSEh1 <- MSE(0.3, n, sigma2, f, 1001)
MSEh2 <- MSE(0.04, n, sigma2, f, 101)
MSEh3 <- MSE(0.003, n, sigma2, f, 101)
Estim1=sum(MSEh1)/n
Estim2=sum(MSEh2)/n
Estim3=sum(MSEh3)/n
Estim1
```

```
## [1] 0.03775501
```

```
Estim2
```

```
## [1] 0.001766445
```

```
Estim3
```

```
## [1] 0.02020174
```

Question 6

1. Can this estimator of MSE be used in practice?
2. Estimate $b^2 := \int_0^1 b^2(x)dx$, $\sigma^2 := \int_0^1 \sigma^2(x)dx$ and plot them as a function of h on the same figure along with $MISE$. Comment on this figure in terms of bias-variance tradeoff.

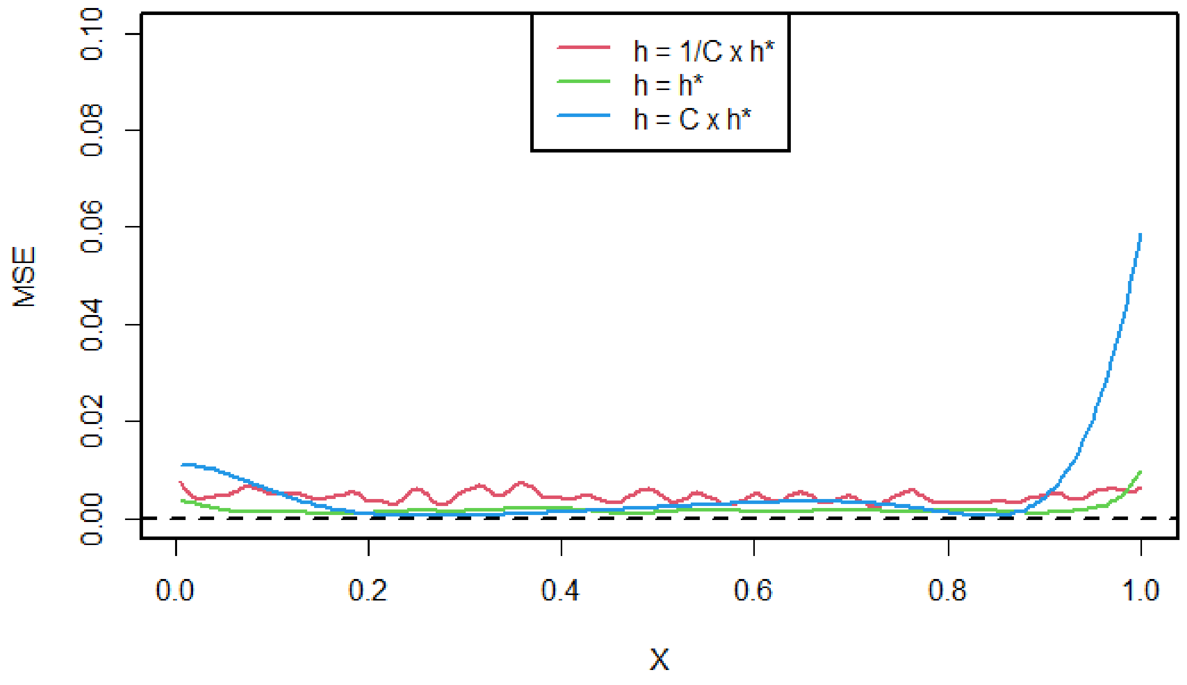
Remark again that since simulations can be replicated at will, these estimates of b^2 and σ^2 are sharp.

V. Boundary effects

We now focus at the pointwise risk (MSE). We can visualize the influence of the choice of h directly on MSE. Again, we compare the "optimal" bandwidth to two other choices $h = 1/C \times h^\star$ and $h = C \times h^\star$, for $C > 1$.

Execute then explain what represent the following plot. You may change C also.

```
par(lwd = 2)
C <- 3
plot(X, MSE(best/C, n, sigma2, f, 50), t = 'l', ylab = "MSE", col = 2, ylim = c(0, 0.1))
lines(X, MSE(best, n, sigma2, f, 50), col = 3)
lines(X, MSE(C*best, n, sigma2, f, 50), col = 4)
lgd <- c("h = 1/C x h*", "h = h*", "h = C x h*")
legend("top", lgd, col = 2:4, lty = 1)
abline(h=0, lty = 2)
```



Question 7

1. Is the estimated MSE for the optimal bandwidth uniformly low or do you suspect a systematic bias?
2. Recalling the shape of the true function f and recalling that we are studying the Nadaraya-Watson estimator, can you propose an explanation for what happens near the boundary of the interval [0,1]?
3. In particular, try to figure out what happens with the estimator when the bandwidth is rather large, and illustrate it.

VI. Local polynomials

Question 8

1. Using the above example of Nadaraya-Watson, create a function `fHat_LP` that generates a local polynomial regression estimator, and functions `MSE_LP` and `MISE_LP` that estimate the associated MSE and MISE, respectively.
2. Calculate the optimal bandwidth for local polynomial estimators of degree 0, 1, 2.
3. Plot the MSE associated to the optimal bandwidth, and compare the MSE at the boundary for local polynomial estimators of degree 0, 1, 2.

VII. Cross-validation

Recall that a linear nonparametric (NP) estimator is an estimator that may be written as

$$\hat{f}_{n,h}(x) = \sum_{i=1}^{n} Y_i W_{ni}(x,h)$$

It is linear in Y with weights $W_{ni}(x,h)$ depending on the sample and a smoothing parameter h. A nice property of linear NP estimators is that the leave-one-out cross-validation risk may be explicitly written as

$$CV(h) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{f}_{n,h}(X_i)}{1 - W_{ni}(X_i,h)}\right)^2$$

Local polynomial estimators are instances of linear NP estimators, with easily accessible $W_{ni}(x,h)$ through a least square minimization.

In the case of local polynomials, the matrix of weights $W = (W_{ni}(X_j))_{i,j}$ may be obtained using the following simple `R` function using 'locpoly'

```r
getW <- function(X, h) {
    n <- length(X)
    W <- matrix(NA, n, n)
    for (ii in 1:n){
        Yi <- rep(0, n)
        Yi[ii] <- 1
        fit <- locpoly(X, Yi, bandwidth = h, gridsize = n)
        W[ii, ] <- fit$y
    }
    W
}
```

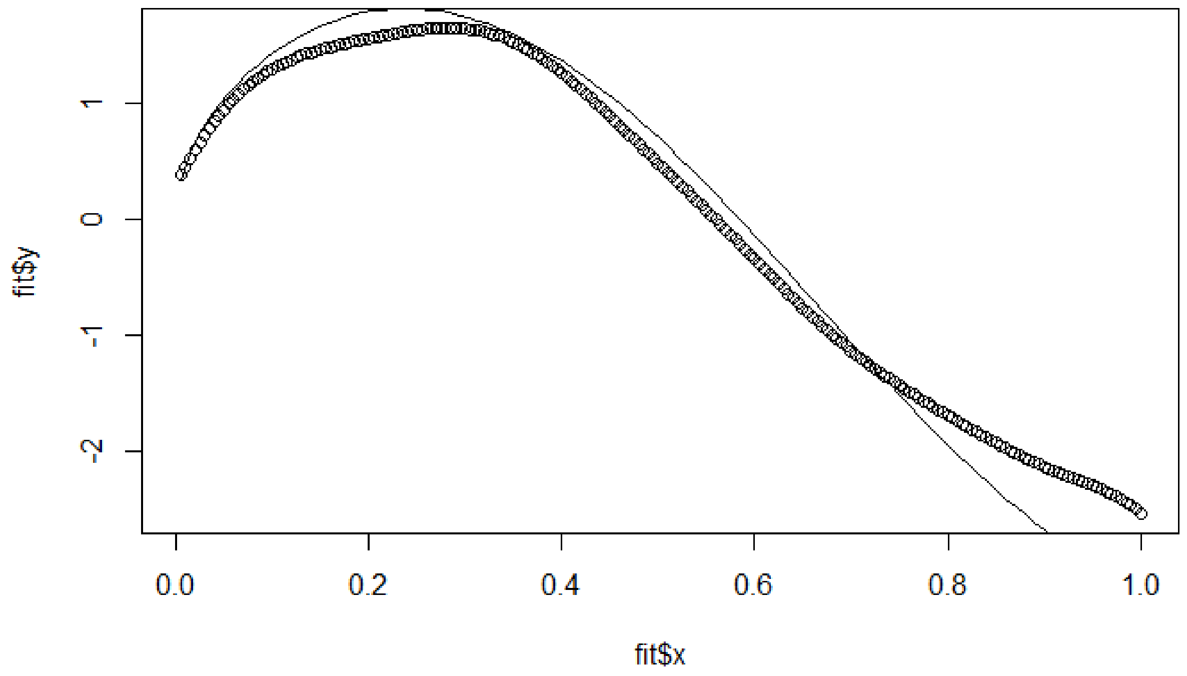
Question 9

1. Explain why the above code does indeed calculate W.
2. Write a function to estimate the cross-validation risk for a local polynomial estimator.
3. Compare the cross-validation bandwidth to the optimal bandwidth in terms of associated MSE.
4. Which of the two should be used in practice?

VII. Derivative

The function `locpoly` can also calculate derivatives of the regression function:

```r
fit <- locpoly(X, Y, drv = 1, bandwidth = 0.1, gridsize=length(X))
plot(fit$x, fit$y)
ff <- function(x) sin(pi*x)/sqrt(x)/2 + pi*sqrt(x)*cos(pi*x)
curve(ff, add = TRUE)
```



Question 10

1. What is pictured in the above estimation plot?
2. Evaluate the MSE and MISE performance of the local polynomial estimator of the derivative of f.

VII. Bonus

Question 11

1. Change the power of x in $f(x)$ (freely) into $f_B(x)$ then again simulate the model with $n = 300$ observations to find by simulations the best bandwidth $h_{1,n}$ for the Gaussian kernel NW estimator $f_{1,n}(x)$ of $f_B(x)$.
2. Compare $h_{1,n}$ with the best bandwidth $h_{2,n}$ computed by simulations for the Gaussian kernel local polynomial estimator $f_{2,n}$ of order 2.
3. What is the less risky estimator ?
4. Compare with the performance of a projection estimator on the Fourier basis as detailed in the lecture note. You should estimate the MISE by simulation again, and let vary the number of Fourier coefficients to estimate in order to retain the best compromise (not many coefficients since $n = 300$ is small).

Appendix

Why fixing the grid size in `locpoly`?

By default the `locpoly` function returns a vector of values for $\hat{f}(x)$ for 401 equally-spaced values of x in $[0, 1]$. Here, we want $\hat{f}(X_i)$ for $1 \leq i \leq n$, where the X_i are `n` equally-spaced values in $[0, 1]$:

```
fit <- locpoly(X, Y, bandwidth = 1)
length(fit$y)
length(fit$y) == length(X)
```

Our custom function `locpoly2` forces the option `gridsize` n `locpoly` to match the size of the input design:

```
fit <- locpoly2(X, Y, bandwidth = 1)
length(fit$y)
length(fit$y) == length(X)
```

```
max(abs(X - fit$x))
```

```
## [1] 1.110223e-16
```