Université Catholique de Louvain
Institut de Statistique

LSTAT2150
Non-parametric statistics
Smoothing methods

Project number 7

December 7, 2020

LAMY Lionel
1294-1700

Contents

1 Introduction **2**

2 Quality assessment **2**
2.1 Bias . . . 2
2.2 Variance . . . 2
2.3 Mean squared error . . . 2
2.4 Mean integrated squared error . . . 2
2.5 Mean sum of squared error . . . 2

3 Regression estimators **3**
3.1 Nadayara-Watson (nonparametric) . . . 3
3.2 Polynomial (parametric) . . . 5

4 Investigation **6**

5 Conclusion **8**

A Appendix **9**
A.1 Figures . . . 9
A.2 Code . . . 10

1 Introduction

Let $\{(X_i, Y_i)\}_{i=1}^n$, $n = 100$, follow the regression model :

$$Y_i = m(X_i) + 0.5\,\varepsilon_i, \quad i = 1, \dots, n$$

$$\text{where } X_i \text{ i.i.d} \sim \text{Uniform}\,[0,1],\ \{\varepsilon_i\} \text{ i.i.d} \sim \mathcal{N}(0,1) \text{ and } m(x) = (\sin(2\,\pi\,x^3))^3$$

The objective of this project is to compare the performance between a parametric polynomial regression estimator and a nonparametric Nadayara-Watson regression estimator[1]. To do so, we will first try to determine the optimal polynomial order and bandwidth by varying over a grid of possible values and selecting those values that minimize the MISE (or the MSSE) with respect to the true function $m(x)$. Then, we will inquire about the bias and the variance. Finaly, we will investigate the MSE at locations we feel are relevant in order to conclude from the performance of our regressions.

2 Quality assessment

As announced in the introduction above, we will use estimators to judge and compare the quality of our models. As such, it seems important to us to define them beforehand.

2.1 Bias

The bias mesures the systematic deviation of the estimator from the true density. If the estimator tends to zero when the number of observations tends towards infinity, then it is considered asymptotically unbiased.

$$\text{Bias}\,[\widehat{m}(x)] = \text{E}[\widehat{m}(x)] - m(x)$$

2.2 Variance

The variance represents the mean of the squares of the deviations from the true mean and is used to quantify the dispersion of the estimates.

$$\text{Var}\,[\widehat{m}(x)] = \text{E}\,[(\widehat{m}(x) - \text{E}\,[\widehat{m}(x)])^2]$$

2.3 Mean squared error

The MSE is the average squared difference between the estimated values and the actual value which means that it characterizes the accuracy of an estimator.

$$\text{MSE}\,[\widehat{m}(x)] = \text{E}\,[(\widehat{m}(x) - m(x))^2]$$

2.4 Mean integrated squared error

We will mainly try to use this measurement to define the quality and accuracy of our models. It is the integration of the MSE on all possible values of x (in our case 0 to 1 as X_i follows an Uniform[0,1]).
It can also be computed as the mean of integrated squared errors (ISE).

$$\begin{aligned}\text{MISE}\,[\widehat{m}(\cdot)] &= \int \text{MSE}\,[\widehat{m}(x)]\,dx \\ &= \int \text{E}\,[(\widehat{m}(x) - m(x))^2]\,dx \ = \text{E}\int (\widehat{m}(x) - m(x))^2\,dx\end{aligned}$$

2.5 Mean sum of squared error

As an integral can be approximated by $\frac{1}{n}\sum_{i=1}^n$, we have a simpler equivalence of the MISE which saves us from having to solve an integral, which can sometimes be tricky.

$$\text{MSSE}[\widehat{m}(\cdot)] = \frac{1}{n}\sum_{i=1}^{n} \text{MSE}\,[\widehat{m}(x_i)]$$

[1] Called respectively 'LM' and 'NW' in this paper

3 Regression estimators

3.1 Nadayara-Watson (nonparametric)

3.1.1 Description

This estimator, published in 1964 by Nadaraya and Watson, proposes to consider the function m(x) as a locally weighted average of Y_i given observations X_i and try to estimate it using a kernel as a weighting function. The motivation behind it comes from :

$$m(x) = E[Y|X = x] = \int y \frac{f(x,y)}{f(x)} dy$$

Where the numerator $f(x, y)$ and denominator $f(x)$ can be estimated via a kernel density estimator :

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

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

Combining all in one, we therefore have the NW estimator, with h as the bandwidth of the kernel K:

$$\widehat{m}_{NW}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) Y_i}{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)}$$

The Kernel used is this work is the Gaussian, defined as $K_G = \exp\left(-\frac{1}{2}u^2\right)(\sqrt{2\pi})^{-1}$ or simply **dnorm**(u) in R. Notice that the equations above show that the bandwidth is what affects the estimator the most, so the choice of a Kernel isn't that much of a matter.

3.1.2 Optimal bandwidth selection

In order to find the best bandwidth, we code a R function that compute M regressions for each value of a sequence of possible bandwidths and returns the one where the average of the M minimize the MISE/MSSE. First, we tried bandwidths ranging from 0.01 to 0.50 in increments of 0.01 and with M=500. We obtained $h = 0.04$ as result. For the sake of accuracy, we started the experiment again with possible values ranging from 0.03 to 0.05 per 0.001 increase. We thus found that 0.036 seems to be the best bandwidth.[2]

On the graph on the left, we drew the estimate of a single NW regression, while on the right, we drew the mean of M=500 regressions. We notice that by bootstrapping, the regression fits slightly better to the true function but that the largest deviations still are at changes in convexity (second derivative).

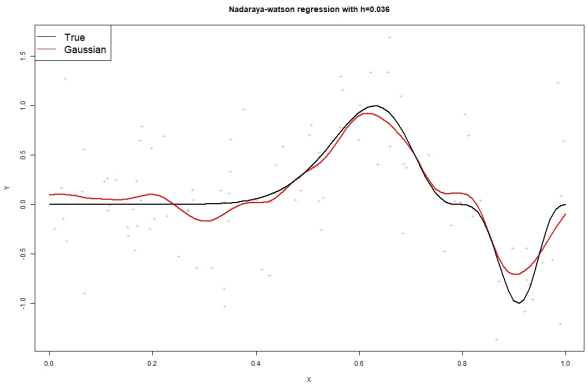

Figure 1: Fit of the NW estimator with the optimal h and n=100

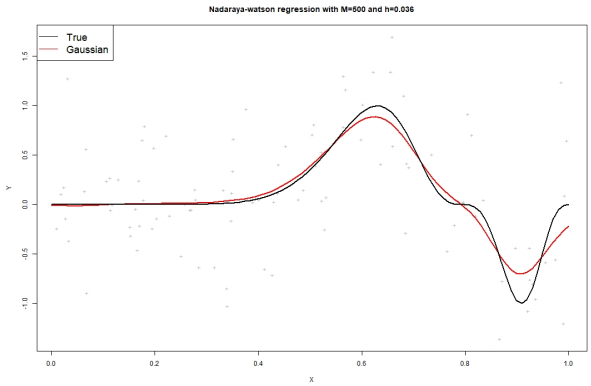

Figure 2: Fit of the NW estimator with the optimal h, M=500 and n=100

[2] An animated example of the bandwidth evolution is available here

We also wanted to know what would be the best h with a smaller sample size, and to verify in the process that the MSSE is a good approximation of the MISE. This appears to be confirmed in the table below. The left part represents the value of the quality estimators while the right part corresponds to the associated h values. We observe that as the sample size increases, the estimators and bandwidths appear to decrease.

n	Minimized		Bandwidth	
	MISE	MSSE	MISE	MSSE
25	0.6493	0.6411	0.145	0.145
50	0.1073	0.1122	0.127	0.124
100	0.0314	0.0317	0.036	0.036

Table 1: MISE/MSSE and the optimal bandwidth by n

3.1.3 Bias and variance

We will now focus on the bias and variance of the regression with the optimal bandwidth ($h = 0.036$) for n=100. For this, we have kept the M predictions made in the above section. Thus, we can easily obtain the bias by subtracting the true value from the mean of the regressions for each of the points. The same is done for the variance, where we regressed once again and subtracted the mean of the regressions, all squared. See equations in sections 2.1 and 2.2 for a better understanding.

Let's first analyze the bias. Graphically, we notice that the bias is very important on the right side, corresponding indeed to the worst-estimated points in the Figure 2. We were expecting this result because theoretically, we know that the bias of a Kernel estimator $\approx \frac{h^2}{2} m''(x) \int u^2 K(u)\, du$ and so is proportional to the second derivative of the density. Moreover, we are pleased to state that the bias only takes 0.2989 as maximum value, with a mean value of -0.0023 and median of 0.0098 and a absolute sum of 5.8867.

An additional plot with both regression and bias is available in appendix (Figure 14).

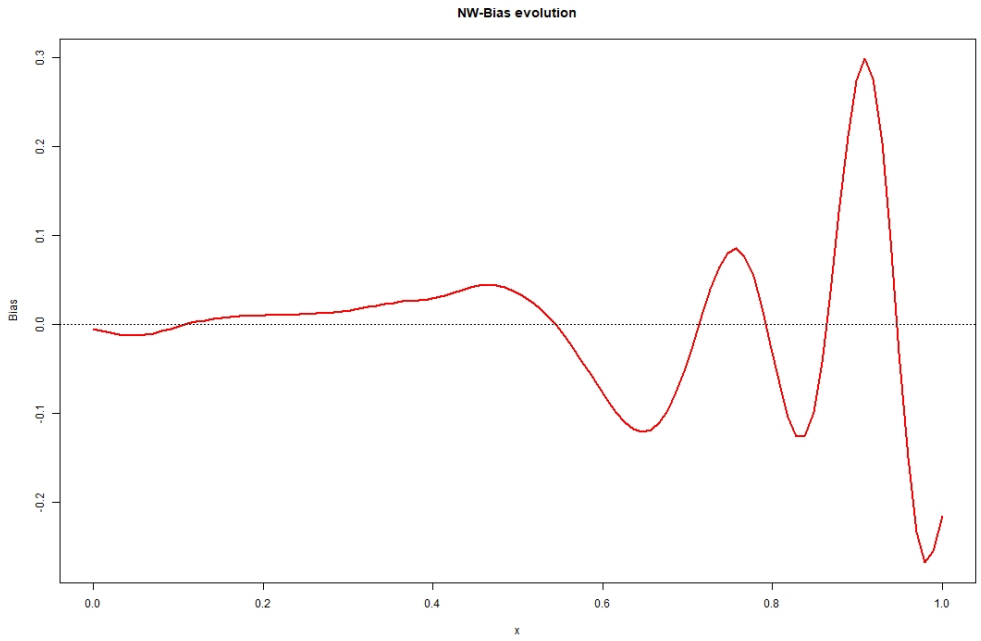

Figure 3: Bias evolution of the NW best regression

By generating a new regression on the same sample, we obtained a variance of 0.0038. In addition, by experimenting with different samples, we obtained a maximum variance of about 0.0450, which we believe to be a very good value.

3.2 Polynomial (parametric)

3.2.1 Optimal polynomial degree

We will use the following R command : **lm**(Y $\sim$ **poly**(X, degree)). The latter allows us to avoid using a long formula with multiple powers, thanks to the poly function that returns a matrix of orthogonal polynomials. Our goal right now is to find out what is the best degree to use to fit the real $m(x)$ function.

Unlike non-parametric regression, we were unable to find a way to calculate the MISE for linear regression and therefore sought to minimize the MSSE. Although this does not affect the results as we saw earlier, we wanted to make the point. So, we used the same strategy as for the Nadayara-Watson and generated M=500 polynomial regressions, for each degree ranging from 1 to 20 and retained the one with the smallest MSSE.

MSSE	Bias	Degree
0.0597	9.0993	7
0.0784	5.4018	19

Table 2: MSSE, bias and degree of polynomial regression

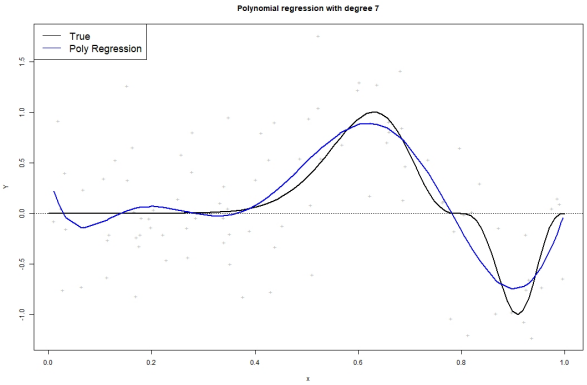

Figure 4: Fit with degree 7

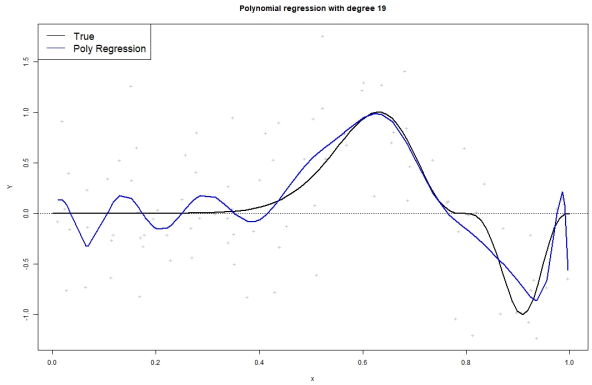

Figure 5: Fit with degree 19

However, by trying to minimize the bias rather than the MSSE, we noticed that the degree returned is higher and as you will see below in Figure 7, the mean prediction of M=500 simulations is a bit more accurate with degree 19 than degree 7 (Figure 6) despite a lower MSSE. This might be explained by the fact that with such a high degree, the regression is highly dependent on the sample (the distribution of X) and is therefore highly variable (overfit), leading to an increase of the MSSE. We observe this effect in figures 6 and 7, where the dotted lines are the minimum and maximum values taken by the M predictions (like a confidence interval). With the lower degree, we don't have the best results but each individual prediction is more stable and less far from the real function.

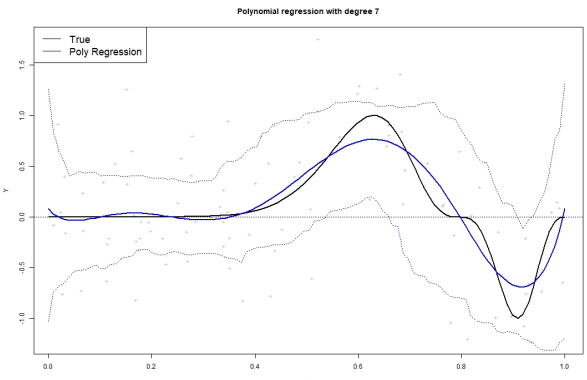

Figure 6: Mean fit with degree 7, M=500

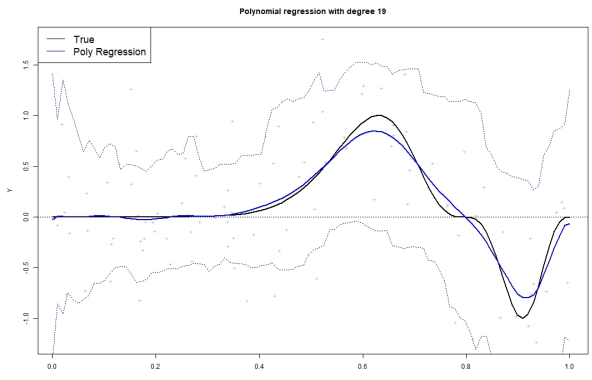

Figure 7: Mean fit with degree 19, M = 500

By precaution, and in order to follow the instructions given in the introduction, we have decided to keep $degree = 7$ for our next analyses.

3.2.2 Bias and variance

We observe the same limitations and weaknesses as with the Nadayara-Watson estimator yet with a slightly larger amplitude. The maximum bias is 0.3124, while the mean and the median are respectively 0.0004 and 0.0019. The sum, however is 9.1032, which corresponds to an increase of 155% compared to the bias obtained with NW which was 5.8867. The variance is 0.0184.

An additional plot with both regression and bias is also available in appendix (Figure 15)

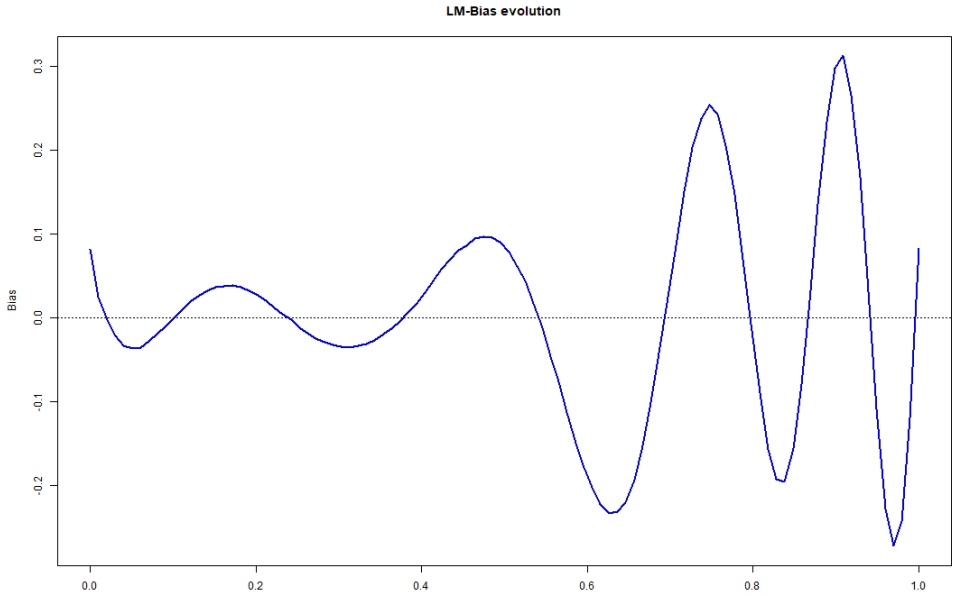

Figure 8: Bias evolution of the LM best regression

4 Investigation

In this section, we are going to investigate numerically with 5000 Monte-Carlo simulations the MSE of some points. The points that we find interesting to explore are [0, 0.2, 0.45, 0.63, 0.78, 0.91, 1] and are represented on the Figure 9 below by a red dashed line. We have chosen the two points at the extremities, the point between the maximum and the minimum, as well as these last two, and an additional point near the center.

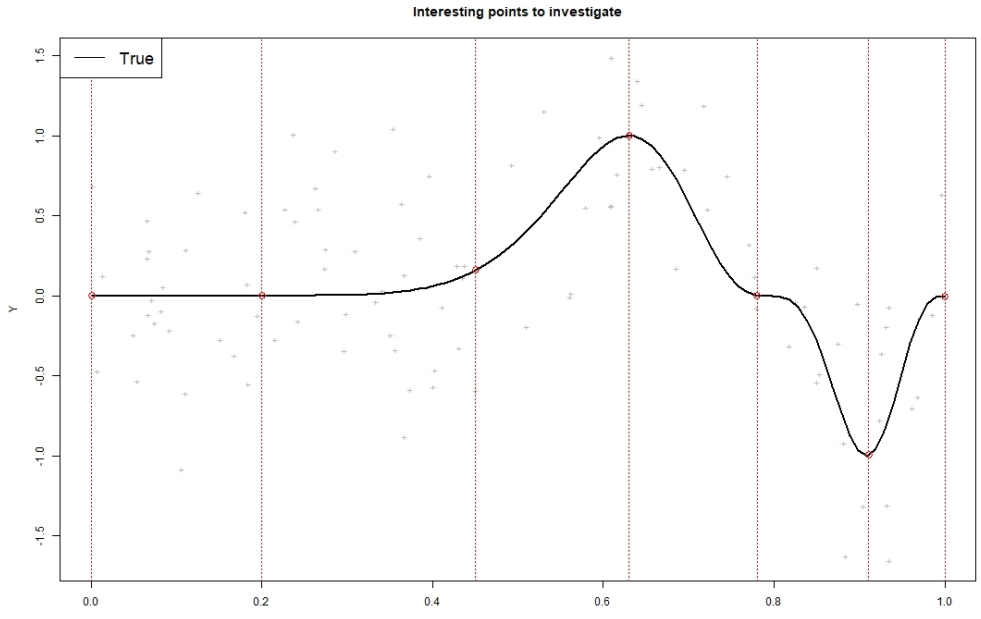

Figure 9: Interesting points marked by a red line and a circle

By calculating the MSE for each point of interest for n = 25, 50, 100 and 1000 we notice several things: i) the mean MSE of all points decreased drastically between n=25 and n=1000 for both polynomial and Nadaraya-Watson regression. ii) the two points on the right are always with the highest mean squared error. iii) the Nadaraya-Watson estimator seems to have systematically less error, with the exception of point 0.2.

	Nadaraya-Watson					**Polynomial**				
Points	n=25	n=50	n=100	n=1000	Mean	n=25	n=50	n=100	n=1000	Mean
0	0.1441	0.0889	0.0424	0.0040	0.0699	0.1941	0.1643	0.1191	0.0270	0.1261
0.2	0.0909	0.0440	0.0212	0.0020	0.0395	0.0713	0.0338	0.0179	0.0032	0.0316
0.45	0.0992	0.0450	0.0220	0.0026	0.0422	0.1261	0.0769	0.0513	0.0100	0.0662
0.63	0.1120	0.0572	0.0308	0.0111	0.0528	0.2339	0.1483	0.0819	0.0327	0.1392
0.78	0.0996	0.0458	0.0235	0.0034	0.0431	0.3828	0.2292	0.1406	0.0224	0.1938
0.91	0.2459	0.1562	0.1146	0.0864	0.1508	0.7049	0.2811	0.1394	0.0641	0.2974
1	0.3630	0.1951	0.1033	0.0454	0.1767	0.4177	0.2481	0.1435	0.0931	0.2256
Mean	0.1650	0.0903	0.0511	0.0221	0.0821	0.3130	0.1688	0.0991	0.0362	0.1543

Table 3: MSE evolution and comparison at specifics points

Graphically, we see more easily the very fast decrease in error at the beginning and we also notice that an asymptotic trend is emerging. The mean squared error progressively decreases. To remedy this and get even closer to zero, the optimal bandwidth and degree values would have to be recalculated with a sample size larger than n=100.

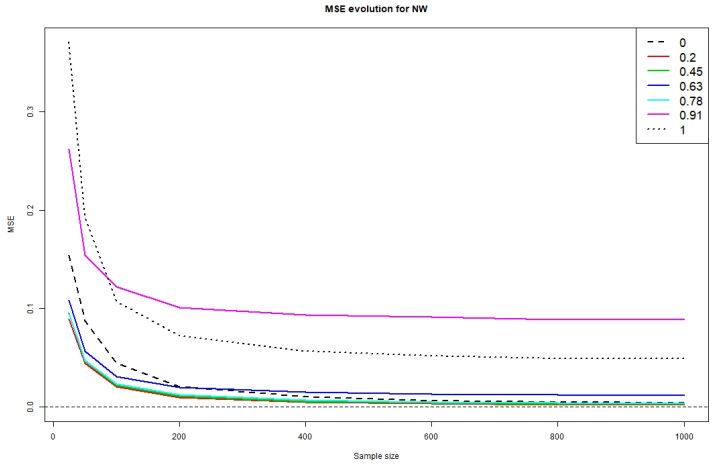

Figure 10: NW-MSE evolution

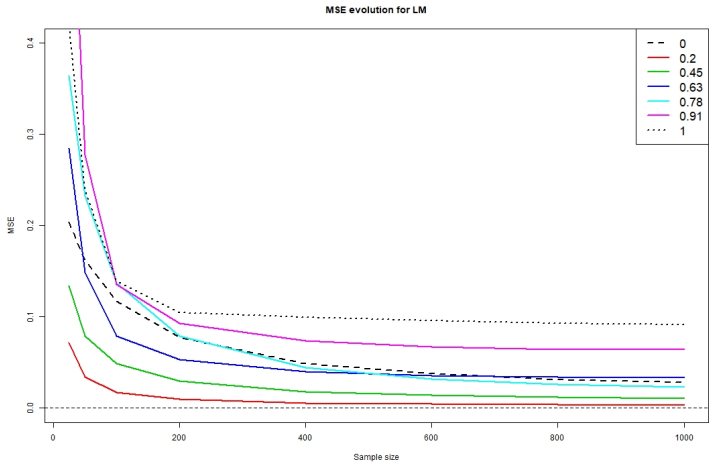

Figure 11: LM-MSE evolution

5 Conclusion

To conclude this work and this comparison, we will end with two final graphs with the two estimators. In our case the **Nadayara-Watson estimator did slightly better to estimate the true function** $m(x)$. Nevertheless, we believe that since the true function $m(x)$ is relatively simple, with a higher n and a higher degree, the parametric regression could be better. In addition, a perhaps more interesting approach to explore would be the use of spline, via a GAM model for example. However, we consider ourselves satisfied and leave this possibility open for future work.

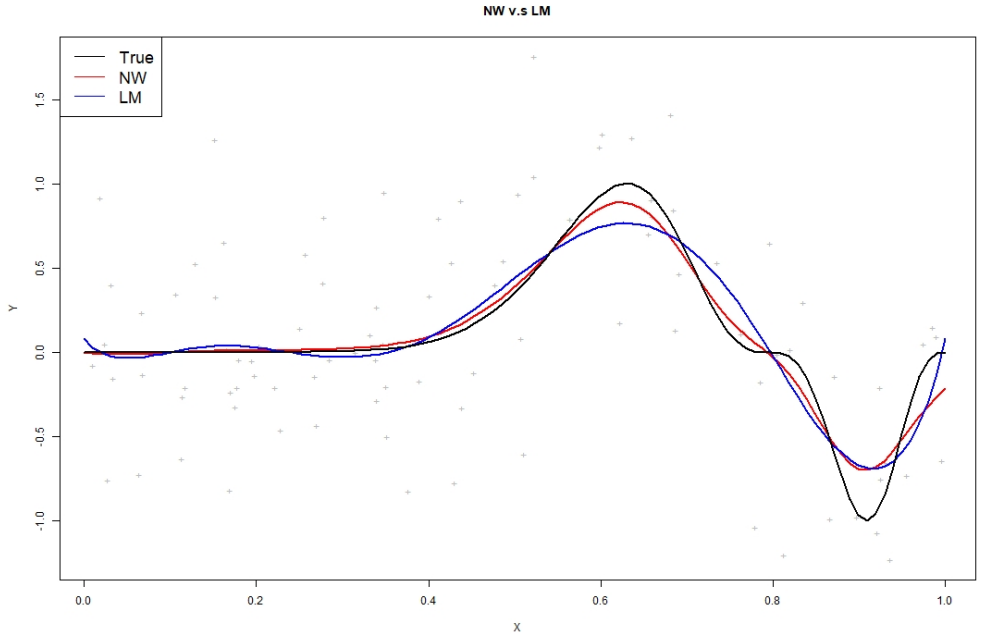

Figure 12: Comparison of the two estimators

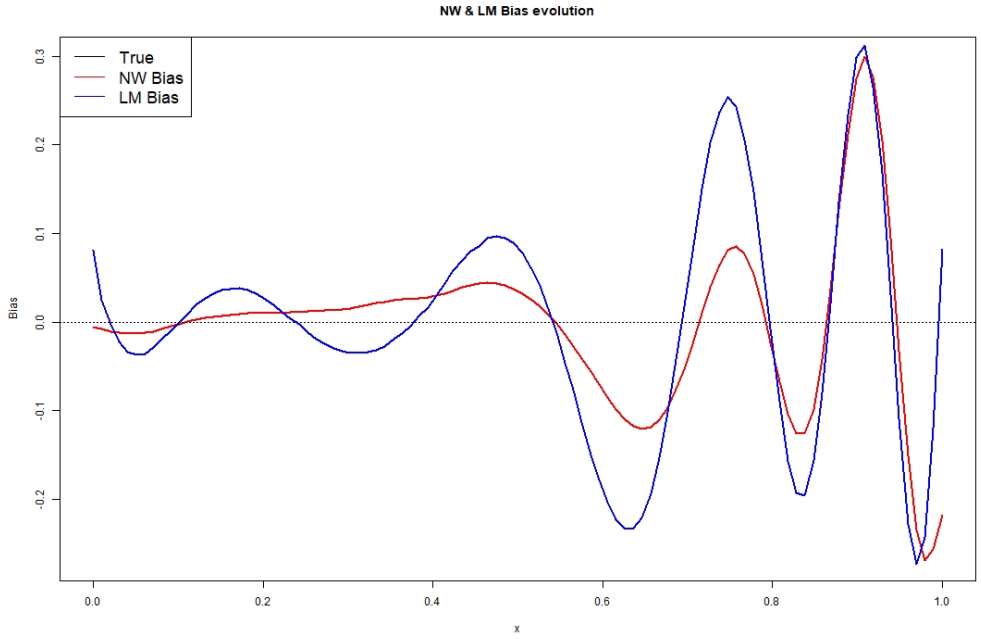

Figure 13: NW and LM bias together

A Appendix

> **Info:** All the content (including full-sized figures) and the code can be found on github via https://github.com/lamylio/LSTAT2150-Project

A.1 Figures

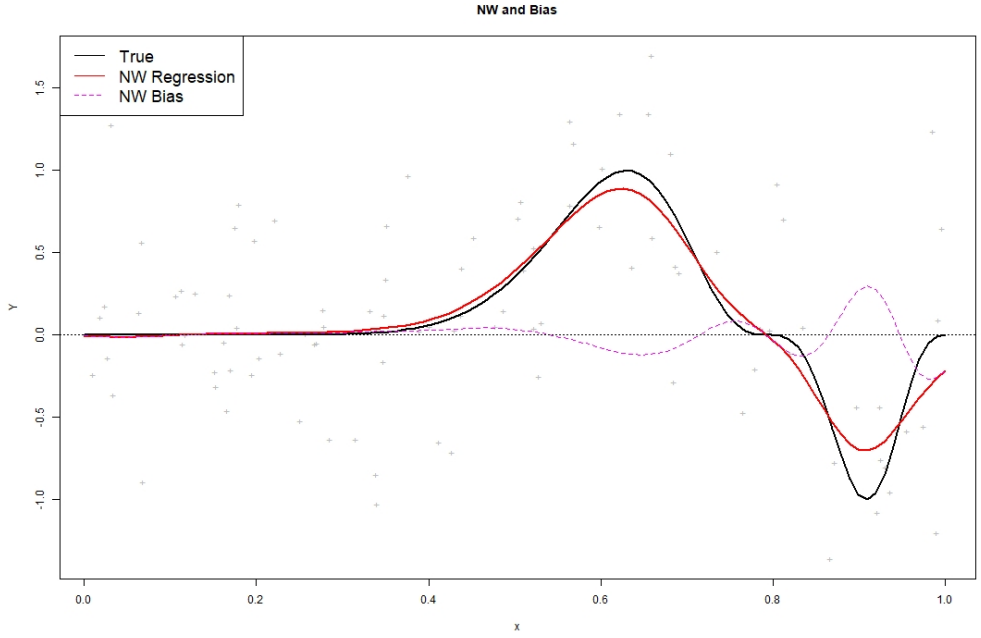

Figure 14: Bias with the NW best regression in same time

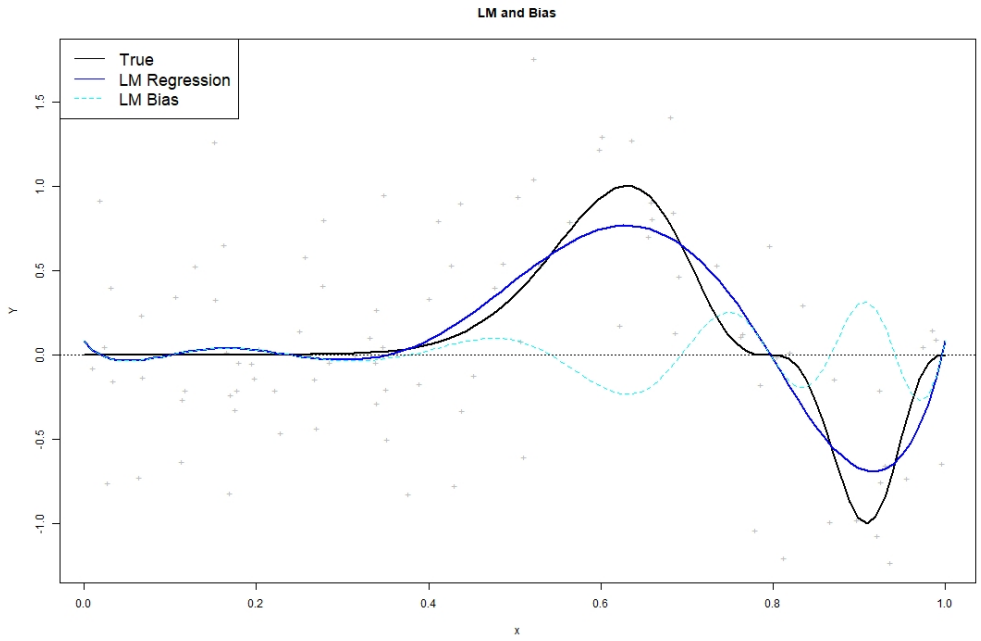

Figure 15: Bias with the LM best regression in same time

A.2 Code

A.2.1 Main

```r
# ============================================
# By: LAMY Lionel
# Noma: 1294-17-00
# https://github.com/lamylio/LSTAT2150-Project
# ============================================


# =====
# Please see Setup
# =====

set.seed(122020)
source("./sources/setup.r")
source("./sources/drawing.r")
source("./sources/minimization.r")
source("./sources/investigate.R")

# Global X and Y
CP.X = sort(X(n))
CP.Y = Y(CP.X)
CP.x = seq(0, 1, length = n)

# Kernel
Knorm = function(u) dnorm(u) #Gaussian kernel

# NW Estimator
NW.regEst <- function(x, X, Y, h, K) sum(Y * K((x - X)/h))/sum(K((x - X)/h))



# =====
# Please see Minimize
# =====

# ---
# Warning: might take a while
# Use ni=100, M=100 by default but better results with higher values
# ---

# NW.minimizeMISE(h.test = seq(0.03, 0.05, 0.001), M=500)

NW.optimal = NW.minimizeMSSE(h.test = seq(0.03, 0.05, 0.001), M=500)
LM.optimal = LM.minimizeMSSE(M=500)

# Optimal bandwidth
NW.h = NW.optimal$h[1] # 0.036
# Optimal degree
LM.poly = LM.optimal$poly[1] # 7

# Models (for futur use)
NW.optimal.model = sapply(seq(0,1,length=nrow(NW.optimal)), function(x) NW.regEst(x, CP.X, CP
LM.optimal.model = lm(CP.Y ~ poly(CP.X, LM.poly))$fitted


# As X.optimal already contains the bias and the MSSE
# we only need to compute the variance, via MReg

NW.optimal$variance = round(
  mean((NW.optimal.model - NW.optimal$MReg)^2)
,4)
```

```r

LM.optimal$variance = round(
  mean((LM.optimal.model - LM.optimal$MReg)^2)
,4)

# head(NW.optimal, 1)
# head(LM.optimal, 1)



# =====
# Please see Investigate
# =====

CP.to_check = c(0,0.2,0.45,0.63,0.78,0.91,1)
CP.investigation.samples = c(25,50,100,200,400,600,800,1000) # smoother lines

# Draw which points
plot(CP.X, CP.Y, pch="+", col = "grey", xlab = "x", ylab="Y")
lines(CP.x, m(CP.x), col = 1, lwd=2)
abline(v=CP.to_check, col=2, lty=3)
title("Interesting points to investigate")
legend("topleft",
    legend = c("True"),
    col = c(1),
    lty = c(1),
    cex = 1.5
)

# Compute all the MSE
NW.investigation.results = CP.investigation.toframe(NW.investigation)
LM.investigation.results = CP.investigation.toframe(LM.investigation)



# =====
# Please see Draw
# =====

# ---
# Draw the evolution of MSE
CP.drawMSE(NW.investigation.results, "MSE evolution for NW", save=T)
CP.drawMSE(LM.investigation.results, "MSE evolution for LM", save=T)

# ---
# Plot both LM and NW best regression
CP.drawComparison(NW.optimal$MReg, LM.optimal$MReg, 100, save=T)

# ===

# ---
# Plot the bias alone
jpeg("./plots/bias/NW-Alone.jpg", quality = 100, width = 1080, height = 720)
plot(CP.x, NW.optimal$Bias, col=2, lwd=2, lty=1,type="l", xlab = "x", ylab="Bias", main="NW-B
abline(h=0, col=1, lty=3)
dev.off()

jpeg("./plots/bias/LM-Alone.jpg", quality = 100, width = 1080, height = 720)
plot(CP.x, LM.optimal$Bias, col=4, lwd=2, lty=1, type="l", xlab = "x", ylab="Bias", main="LM-
abline(h=0, col=1, lty=3)
dev.off()

# Both in the same time
```

```
jpeg("./plots/bias/NW-LM-Alone.jpg", quality = 100, width = 1080, height = 720)
plot(CP.x, NW.optimal$Bias, col=2, lwd=2, lty=1,type="l", xlab = "x", ylab="Bias", main="NW &
lines(CP.x, LM.optimal$Bias, col=4, lwd=2, lty=1)
abline(h=0, col=1, lty=3)
legend("topleft",
   legend = c("True", "NW Bias", "LM Bias"),
   col = c(1, 2, 4),
   lty = c(1, 1, 1),
   cex = 1.5
)
dev.off()

# ---
# Plot bias and regression
jpeg("./plots/bias/NW-Both.jpg", quality = 100, width = 1080, height = 720)
plot(CP.X, CP.Y, pch="+", col = "grey", xlab = "x", ylab="Y", main="NW and Bias")
lines(CP.x, m(CP.x), col = 1, lwd=2)
lines(CP.x, NW.optimal$MReg, col=2, lty=1, lwd=2)
lines(CP.x, NW.optimal$Bias, col=6, lty=2)
abline(h=0, col=1, lty=3)
legend("topleft",
   legend = c("True", "NW Regression", "NW Bias"),
   col = c(1, 2, 6),
   lty = c(1, 1, 2),
   cex = 1.5
)
dev.off()

jpeg("./plots/bias/LM-Both.jpg", quality = 100, width = 1080, height = 720)
plot(CP.X, CP.Y, pch="+", col = "grey", xlab = "x", ylab="Y", main="LM and Bias")
lines(CP.x, m(CP.x), col = 1, lwd=2)
lines(CP.x, LM.optimal$MReg, col=4, lty=1, lwd=2)
lines(CP.x, LM.optimal$Bias, col=5, lty=2)
abline(h=0, col=1, lty=3)
legend("topleft",
   legend = c("True", "LM Regression", "LM Bias"),
   col = c(1, 4, 5),
   lty = c(1, 1, 2),
   cex = 1.5
)
dev.off()


# =====
#  END
# =====
```

A.2.2 Setup

```
# =====
# Given in the project statement
# =====

n=100

X = function(n=100) runif(n)
m = function(x) (sin(2*pi*x^3))^3
Y = function(X=runif(n)) m(X) + rnorm(length(X), 0, 0.5) # transformed into a function
```

A.2.3 Minimize

```r
library(progress)

# ---
# Only return the minimized MISE and the bandwidth associated
NW.minimizeMISE = function(Kernel = Knorm, h.test = seq(0.01, 0.15, 0.001), M = 100, ni = 100
  set.seed(122020)
  best = Inf
  best_h = 0

  pb = progress_bar$new(
    format = "[NW MISE] Best: :bmise (:bh) | Current: :cmise (:ch) [:bar] :current/:total |:p
    total = length(h.test)
  )
  Sys.sleep(0.1)
  pb$tick(0)

  for (h in h.test){

    ISE = matrix(NA, M, 1)

    for (i in 1:M){
      boot.X = runif(ni)
      boot.Y = Y(boot.X)
      boot.ISE = integrate(
        function(xi) (sapply(xi, function(x) NW.regEst(x, boot.X, boot.Y, h, Kernel)) - m(xi)
        lower=0, upper=1)$value
      ISE[i,] = boot.ISE
    }

    MISE = round(mean(ISE), 6)

    if (MISE < best){
      best = MISE
      best_h = h
    }
    pb$tick(tokens=list(bmise=best, bh=best_h, cmise=MISE, ch=h))
  }
  sprintf("[NW MISE]: %.6f with %.6f as bandwhidth", best, best_h)
  return (c(best, best_h))
}

# ---
# This one returns MSSE, bandwidth, Bias and also the mean of the regressions (for variance)
NW.minimizeMSSE = function(Kernel = Knorm, h.test = seq(0.01, 0.15, 0.001), M = 100, ni = 100
  set.seed(122020)
  boot.x = seq(0, 1, length = ni)
  res = data.frame(MSSE=Inf, h=NA, Bias=matrix(NA, ni, 1), MReg=matrix(NA, ni, 1))

  pb = progress_bar$new(
    format = "[NW MSSE] Best: :bmsse (:bh) | Current: :cmsse (:ch) [:bar] :current/:total |:p
    total = length(h.test)
  )
  Sys.sleep(0.1)
  pb$tick(0)
  for (h in h.test){
    SE = matrix(NA, M, ni)
    Reg = matrix(NA, M, ni)

    for (i in 1:M){
      boot.X = runif(ni)
      boot.Y = Y(boot.X)
      boot.reg = sapply(boot.x, function(xi) NW.regEst(xi, boot.X, boot.Y, h, Kernel))
```

```
63        Reg[i,] = boot.reg
64        SE[i,] = (boot.reg - m(boot.x))^2
65      }
66
67      MReg = colMeans(Reg)
68      Bias = MReg - m(boot.x)
69      MSE = colMeans(SE)
70      MSSE = round(mean(MSE),6)
71
72      if (MSSE < mean(res$MSSE)){
73        res$MSSE = MSSE
74        res$h = h
75        res$Bias = Bias
76        res$MReg = MReg
77      }
78      pb$tick(tokens=list(bmsse=res$MSSE, bh=res$h, cmsse=MSSE, ch=h))
79    }
80    return (res)
81  }
82
83  # ---
84  # Haven't found a way to compute MISE with LM (yet)
85  # But as MSSE ~= MISE, this is fine.
86  # This one returns MSSE, bandwidth, Bias, the mean of the M regressions
87  # And also the min and max values taken
88  # ---
89  LM.minimizeMSSE = function(poly.test = seq(1, 20, 1), M = 200){
90    set.seed(122020)
91    res = data.frame(MSSE=Inf, poly=NA, MSE=matrix(Inf, n, 1), Bias=matrix(Inf, n, 1), MReg=mat
92                     low = matrix(NA, n, 1), up = matrix(NA, n, 1))
93
94    pb = progress_bar$new(
95      format = "[LM MSSE] Best: :bmsse (:bp) | Current: :cmsse (:cp) | Bias: :bias [:bar] :curr
96      total = length(poly.test)
97    )
98    pb$tick(0)
99
100   for (p in poly.test){
101     Reg = matrix(NA, M, n)
102     SE = matrix(NA, M, n)
103
104     for (i in 1:M){
105       boot.X = sort(X(n))
106       boot.Y = Y(boot.X)
107       boot.reg = lm(boot.Y ~ poly(boot.X, degree = p, raw=T))
108       boot.pred = predict(boot.reg, newdata=poly(CP.x, degree=p), type="response")
109       Reg[i,] = boot.pred
110       SE[i,] = (boot.pred - m(CP.x))**2
111     }
112
113     # Add "confidence interval"
114     # (this is simply the min / max values taken not real CI)
115     CI = matrix(NA, n, 2)
116
117     for (i in 1:n){
118       CI[i,1] = min(Reg[,i])
119       CI[i,2] = max(Reg[,i])
120     }
121
122     MReg = colMeans(Reg)
123     Bias = MReg - m(CP.x)
124
125     MSE = colMeans(SE)
```

```
    MSSE = round(mean(MSE),6)

    if (mean(MSE) < mean(res$MSE)){
      res$MSE = MSE
      res$MSSE = MSSE
      res$poly = p
      res$Bias = Bias
      res$MReg = MReg
      res$low = CI[,1]
      res$up = CI[,2]
    }

    pb$tick(tokens=list(bias=sum(abs(Bias)), bmsse=res$MSSE, bp=res$poly, cmsse=MSSE, cp=p))
  }
  return (res)
}
```

A.2.4 Investigate

```
# ---
# Compute the MSE by M Monte-carlo simulations

NW.investigation = function(M=5000, n=100){
  SE = matrix(NA, M, length(CP.to_check))

  for (i in 1:M){
    boot.X = runif(n)
    boot.Y = Y(boot.X)
    SE[i,] = (sapply(CP.to_check,function(xi) NW.regEst(xi, boot.X, boot.Y, NW.h, Knorm))- m(
  }

  MSE = colMeans(SE)
  return(MSE)
}

LM.investigation = function(M=5000, n=100){
  SE = matrix(NA, M, length(CP.to_check))

  for (i in 1:M){
    boot.X = sort(runif(n))
    boot.Y = Y(boot.X)
    boot.reg = lm(boot.Y ~ poly(boot.X, degree = LM.poly, raw=ifelse(n<50, T, F)))
    boot.pred = fitted(boot.reg)[c(1,CP.to_check*n)]
    SE[i,] = (boot.pred - m(CP.to_check))^2
  }

  MSE = colMeans(SE)
  return(MSE)
}

# ---
# Create the dataframe for each sample size

CP.investigation.toframe = function(investigation_method){
  set.seed(122020)
  res = NULL
  pb = progress_bar$new(
    format = "[Investigation] Sample: :sample | [:bar] :current/:total |:percent | :elapsed",
    total = length(CP.investigation.samples)
  )
  pb$tick(0)
  Sys.sleep(0.1)
  for (s in CP.investigation.samples){
```

```
    pb$tick(tokens=list(sample=s))
    res = cbind(res, investigation_method(n=s))
  }
  return (res)
}
```

A.2.5 Draw

```
# ---
# Draw the comparison of LM and NW

CP.drawComparison = function(BestNW, BestLM, n=100, save=F){
  set.seed(122020)
  if(save) {jpeg("./plots/comparison.jpg", quality = 100, width = 1080, height = 720)}
  plot(CP.X, CP.Y, pch="+", col = "grey", xlab="X", ylab="Y")
  lines(CP.x, BestNW, col = 2, lwd=ifelse(save, 2, 1))
  lines(CP.x, BestLM, col=4, lwd=ifelse(save, 2, 1), pch=19)
  lines(CP.x, m(CP.x), col = 1, lwd=ifelse(save, 2, 1))
  legend("topleft",
         legend = c("True", "NW", "LM"),
         col = c(1, 2, 4),
         lty = c(1, 1, 1),
         #pch = c(NA, 19, 19),
         cex = ifelse(save, 1.5, 0.8)
  )
  title("NW v.s LM")
  if(save) {dev.off()}
}

# ---
# Draw the evolution of the MSE

CP.drawMSE = function(investigation_frame, title, save=F){
  set.seed(122020)
  middle = nrow(investigation_frame)-1
  if(save) {jpeg(paste0("./plots/investigation/",title,".jpg"), quality = 100, width = 1080,
  plot(CP.investigation.samples,
       head(investigation_frame, 1), col = 1, lwd = ifelse(save, 2, 1), lty=2,
       type="l", xlab="Sample size", ylab="MSE", ylim = c(-0.0001, min(max(investigation_fram
  )

  for (i in 2:middle){
    lines(CP.investigation.samples, investigation_frame[i,], col = i, lwd = ifelse(save, 2, 1
  }

  lines(CP.investigation.samples, tail(investigation_frame, 1), col = 1, lwd = ifelse(save, 2
  abline(h = 0, lty = 2)
  legend("topright",
         legend = CP.to_check,
         col = c(1, 2:middle, 1),
         lty = c(2, rep(1,middle-1), 3),
         lwd = 2,
         cex = ifelse(save, 1.5, 0.8)
  )
  title(title)
  if(save) {dev.off()}
}
```