UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Deams
Dipartimento di
Scienze Economiche, Aziendali,
Matematiche e Statistiche "Bruno de Finetti"

Non parametric statistics

"Local" methods

Francesco Pauli

A.A. 2021/2022

Parametric v. non parametric

With respect to parametric inference, non parametric entails

- few assumptions
- infinite dimensional models

Example: estimate the distribution function from a sample $X_1, \ldots, X_n \sim F()$

- parametric: assume $F \in \mathcal{F} = \{F_\theta() : \theta \in \mathbb{R}^d\}$ estimate θ through, for example, maximum likelihood, $F_{\hat{\theta}}$ is the estimate of the distribution function.
- non parametric: assume F is a valid distribution function; a good estimate is the **empirical distribution function**.

Parametric v. non parametric

With respect to parametric inference, non parametric entails

- few assumptions
- infinite dimensional models

Example: estimate the regression function $E(Y|X = x)$ from a sample (x_i, Y_i), $i = 1, \ldots, n$

- parametric: assume $E(Y|X = x) = \beta_1 + \beta_2 x$ (or any more complicated functional form depending on a possibly multidimensional parameter θ), estimate θ through, for example, maximum likelihood.
- non parametric: assume $E(Y|X = x) = f(x)$ where f belongs to a flexible class of functions (no parameters of direct interest).

Indice

Regression problem

We have a regression problem when we observe $(Y_i, \mathbf{x}_i)$ and we are interested in

$$E(Y|\mathbf{X} = \mathbf{x})$$

Standard tools to deal with this situation are

- the linear model

$$Y|\mathbf{X} = \mathbf{x} \sim N()$$

$$E(Y|\mathbf{X} = \mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x}$$

- the generalized linear model

$$Y|\mathbf{X} = \mathbf{x} \sim < \text{member of expon family} >$$

$$g(E(Y|\mathbf{X} = \mathbf{x})) = \boldsymbol{\beta}^T \mathbf{x}$$

and extension such as (generalized) mixed models (see Torelli's course).

(Generalized) linear models limitation

The key assumption is that relationship between the covariates and

- either the conditional expectation

$$E(Y|\mathbf{X} = \mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x}$$

- or a known function of the conditional expectation

$$g(E(Y|\mathbf{X} = \mathbf{x})) = \boldsymbol{\beta}^T \mathbf{x}$$

is **linear in the parameters**.

Non parametric regression relaxes this assumption, avoiding specification of a precise shape of the relationship.

Non parametric regression

We assume that

$$E(Y|\mathbf{X} = \mathbf{x}) = f(x)$$

where f is a "regular" function (continuous with continuous derivatives up to a certain order).

Two main approaches may be taken

- "local" approach
  - if we had many observations for each value x_0 of X we could estimate $f(x_0)$ as a sample mean.
  - in general we have one observation for each value of X, we may use nearby points

Non parametric regression

We assume that

$$E(Y|\mathbf{X} = \mathbf{x}) = f(x)$$

where f is a "regular" function (continuous with continuous derivatives up to a certain order).

Two main approaches may be taken

- "local" approach: estimate $E(Y|X = x_0)$ using points near to x_0.
- "global" approach (spline)
  - we define a set of functions $f(x;\boldsymbol{\theta})$ which is flexible enough to approximate any regular function $f(\cdot)$
  - we estimate f by choosing the best fitting $f(x;\boldsymbol{\theta})$
  - (This may be more appropriately called a semiparametric approach as we have a parameter $\boldsymbol{\theta}$, it is not a parametric problem because of the dimension of $\boldsymbol{\theta}$ which may be high and variable.)

Non parametric regression

We assume that

$$E(Y|\mathbf{X} = \mathbf{x}) = f(x)$$

where f is a "regular" function (continuous with continuous derivatives up to a certain order).

Two main approaches may be taken

- "local" approach: estimate $E(Y|X = x_0)$ using points near to x_0.
- "global" approach (spline): define a flexible model $f(x; \boldsymbol{\theta})$

Non parametric regression

We assume that

$$E(Y|\mathbf{X} = \mathbf{x}) = f(x)$$

where f is a "regular" function (continuous with continuous derivatives up to a certain order).

Two main approaches may be taken

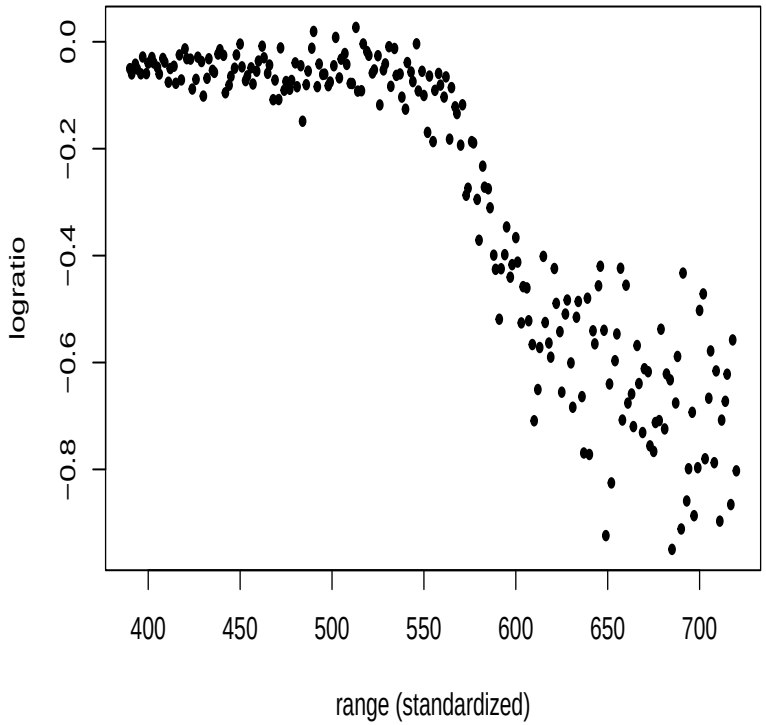
- "local" approach: estimate $E(Y|X = x_0)$ using points near to x_0.
- "global" approach (spline): define a flexible model $f(x; \boldsymbol{\theta})$

One crucial issue in both methods is to decide the degree of smoothness of the estimate, which translates into

- deciding how near is near
- deciding how flexible should the model $f(x; \boldsymbol{\theta})$ be

As we will see a trade off between bias and variance arises.

Motivating example: lidar data



LIDAR = light detection and ranging

- Is a technique to detect chemical compounds in the atmosphere
- x: distance traveled before reflection
- y: log of the ratio of received light between two laser sources

- We want to estimate

$$f(x) = E(Y|X = x)$$

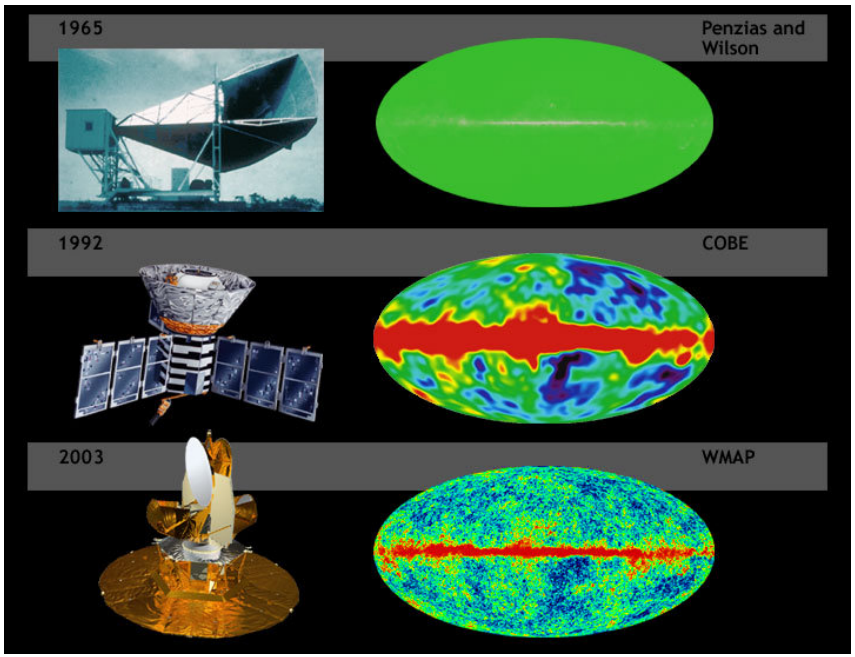
- Well known example of non linear relationship where polynomial regression does non work very well.

Cosmic microwave background

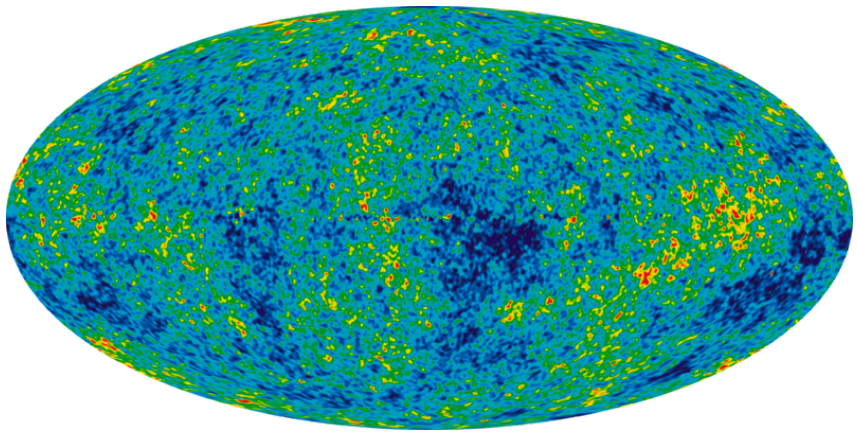
- Genovese, C. R., Miller, C. J., Nichol, R. C., Arjunwadkar, M., & Wasserman, L. (2004). *Nonparametric inference for the cosmic microwave background.* Statistical Science, 308-321.
- Cosmic microwave background is electromagnetic radiation which is observed almost uniformly in all universe.
- It was discovered in 1964 by Penzias and Wilson.
- It can be thought as the 'echo' of the events occurred at the time of the formation of the first atoms (when the universe was 380 000 years old) and shortly after.
- As a consequence, studying the characteristics of CMB today allows to verify hypotheses on the formation of the universe.

Cosmic microwave background: observations

Raw measures of CMB at increasing resolution.



CMB measures having subtracted milky way signal.

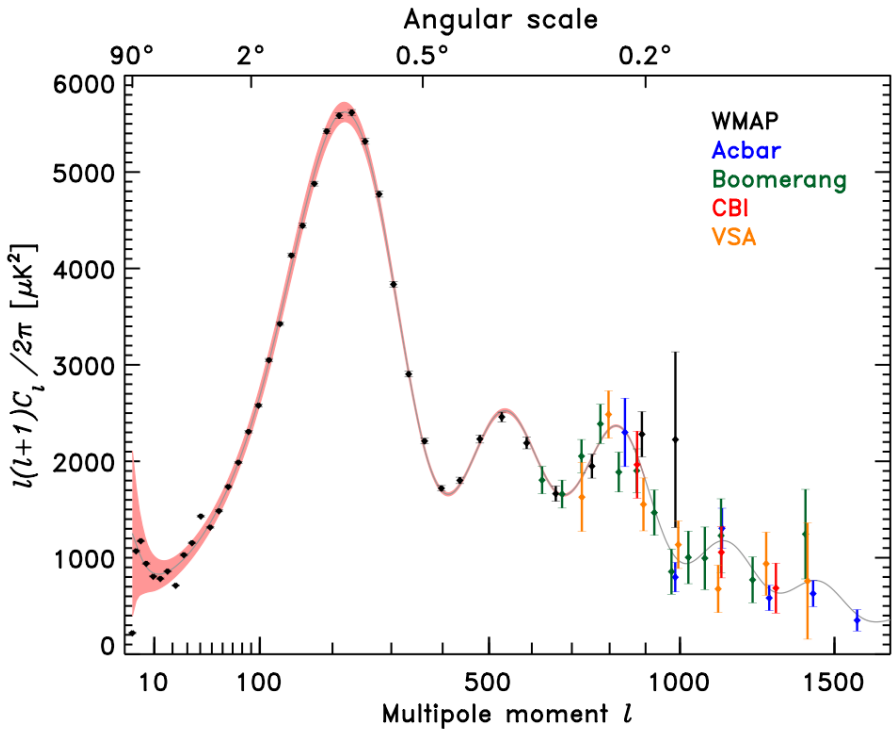


The inhomogeneities are what mostly interests to extract information on the characteristics of the universe at the beginning.

Cosmic microwave background: power spectrum

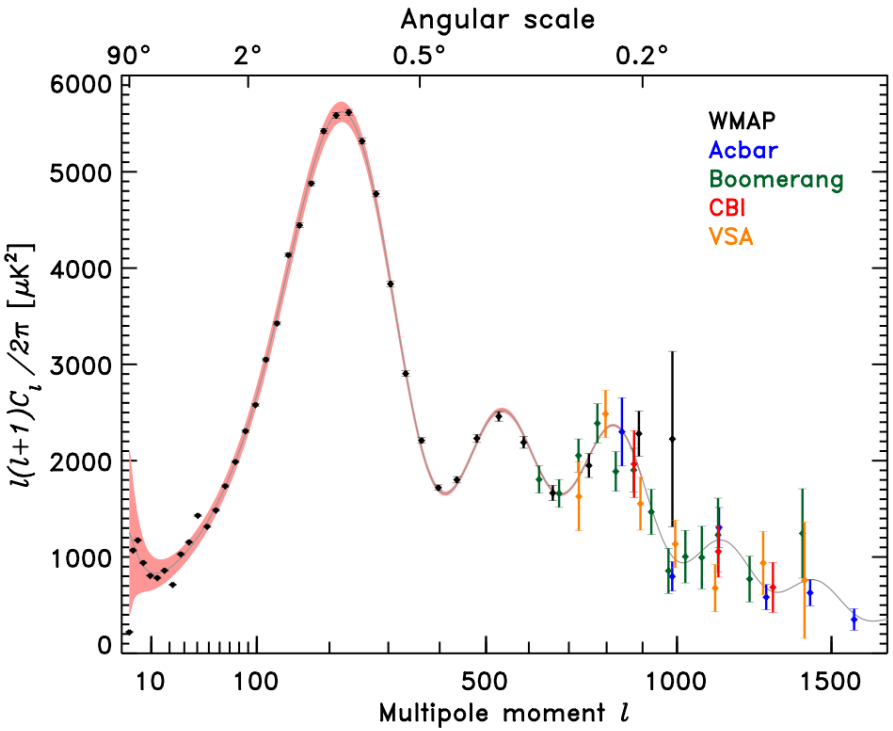
- The power spectrum is a transformation of the above data.
- The number of peaks in the spectrum gives information on the existence of *dark matter*
- (*Dark matter* is a matter which is not visible due to the fact that it does not emit nor reflect light and whose existence would explain various phenomena.)

- In particular, the existence of three or more peak in the spectrum would be coherent with the dark matter hypotheses.
- On the right, the theoeretical shape of the spectrum (with three peaks) and some observations.

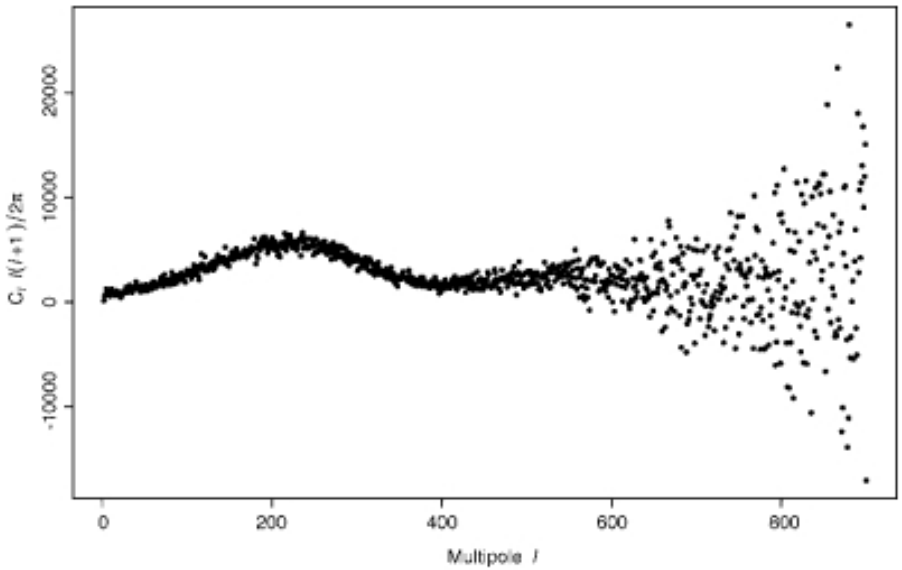


Power spectrum: theory and raw data

Theoretical three peaks spectrum



Raw data from the most recent experiment.



Are there three peaks?
Let's estimate

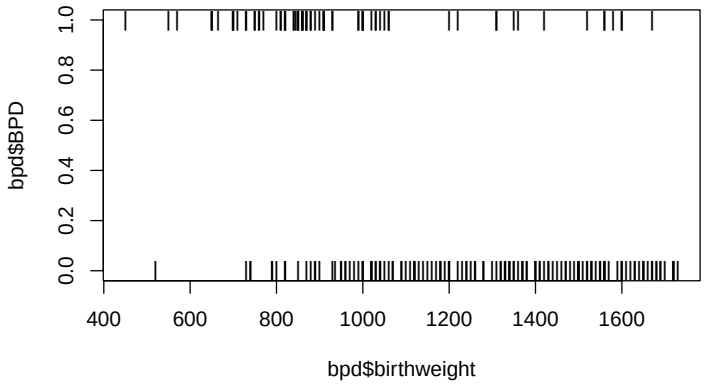
$$y = f(x) + \varepsilon$$

BPD data

Bronchopulmonary dysplasia (BPD) is a lung disease typical of premature babies, its presence may be related to birthweight.

For 223 babies we have observed

- birthweight
- presence of bronchopulmonary dysplasia (BPD)



The response is a Bernoulli r.v., so generalized linear model will have to be used.

Plan

1. Smoothing: local smothers and general issues (bias variance trade off, GCV, inference)
2. Splines: univariate and multivariate splines, GAM
3. Spline as mixed effect models

Books:

- Wasserman: Non-parametric statistics
- Azzalini, Bowman: Applied Smoothing Techniques
- Wood: Generalized Additive Models: An Introduction with R
- Ruppert, Wand, Carroll: Semiparametric regression

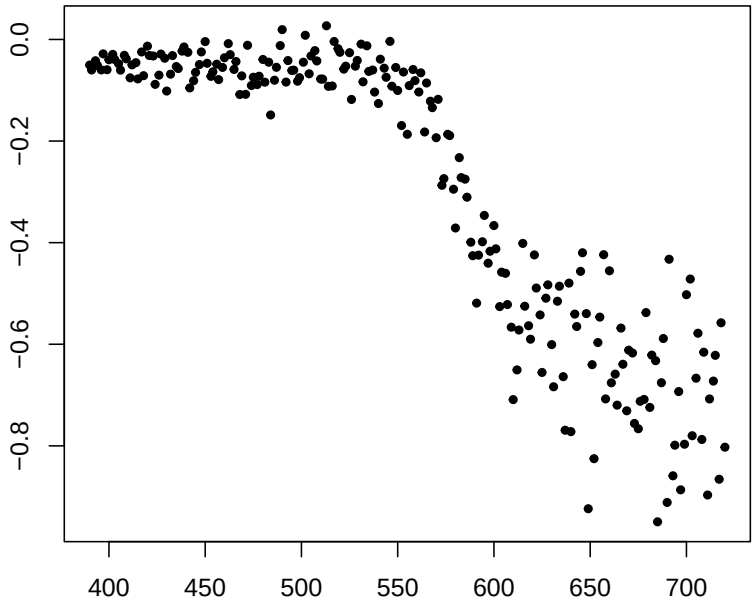
Indice

LIDAR data: linear model

Assume

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $X \in \mathcal{M}_{n\times p}$, $\boldsymbol{\beta} \in \mathbb{R}^p$, $\boldsymbol{\varepsilon} \sim N(0, \sigma^2 I)$,



Using ML

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}$$

so that the smoothed version is

$$\hat{\mathbf{y}} = X\hat{\boldsymbol{\beta}} = X(X^T X)^{-1} X^T \mathbf{y}$$

where $H = X(X^T X)^{-1} X^T$ is called the hat matrix and is the projection matrix from $\mathbb{R}^n$ in the subspace generated by the columns of X, note that

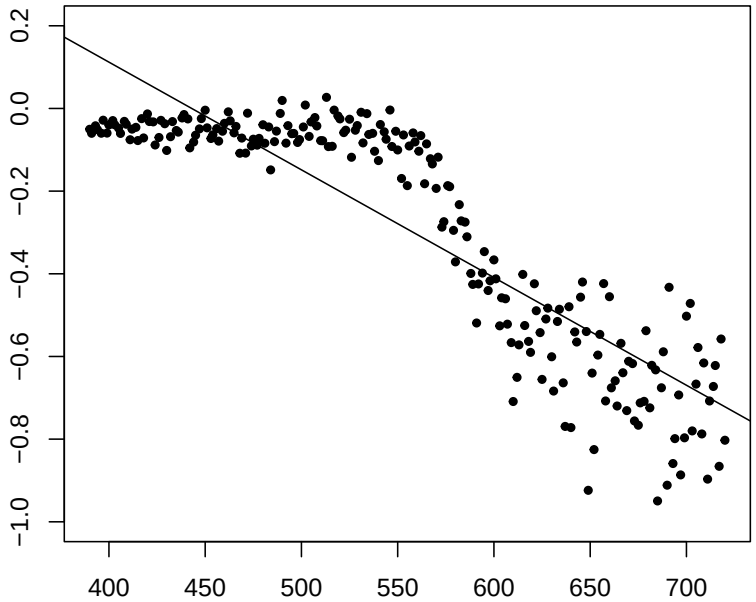
$$\text{trace} H = p$$

LIDAR data: linear model

Assume

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $X \in \mathcal{M}_{n\times p}$, $\boldsymbol{\beta} \in \mathbb{R}^p$, $\boldsymbol{\varepsilon} \sim N(0, \sigma^2 I)$,



Not very satisfying as a smoother, and does not seem to describe the relationship well.

Using ML

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}$$

so that the smoothed version is

$$\hat{\mathbf{y}} = X\hat{\boldsymbol{\beta}} = X(X^T X)^{-1} X^T \mathbf{y}$$

where $H = X(X^T X)^{-1} X^T$ is called the hat matrix and is the projection matrix from $\mathbb{R}^n$ in the subspace generated by the columns of X, note that

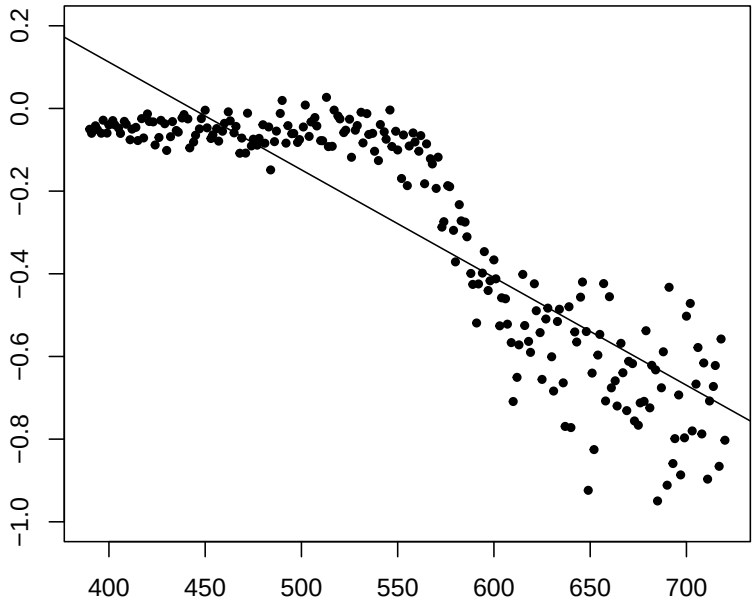
$$\text{trace} H = p$$

LIDAR data: linear model

Assume

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $X \in \mathcal{M}_{n\times p}$, $\boldsymbol{\beta} \in \mathbb{R}^p$, $\boldsymbol{\varepsilon} \sim N(0, \sigma^2 I)$,



Not very satisfying as a smoother, and does not seem to describe the relationship well.

Note that

$$\hat{\mathbf{y}} = X\hat{\boldsymbol{\beta}}\mathbf{y} = X(X^TX)^{-1}X^T\mathbf{y}$$

means that the estimated conditional expectation is

$$\widehat{E(Y|X=x)} = \hat{f}(x) = \sum_{i=1}^{n} h_i(x)\,Y_i$$

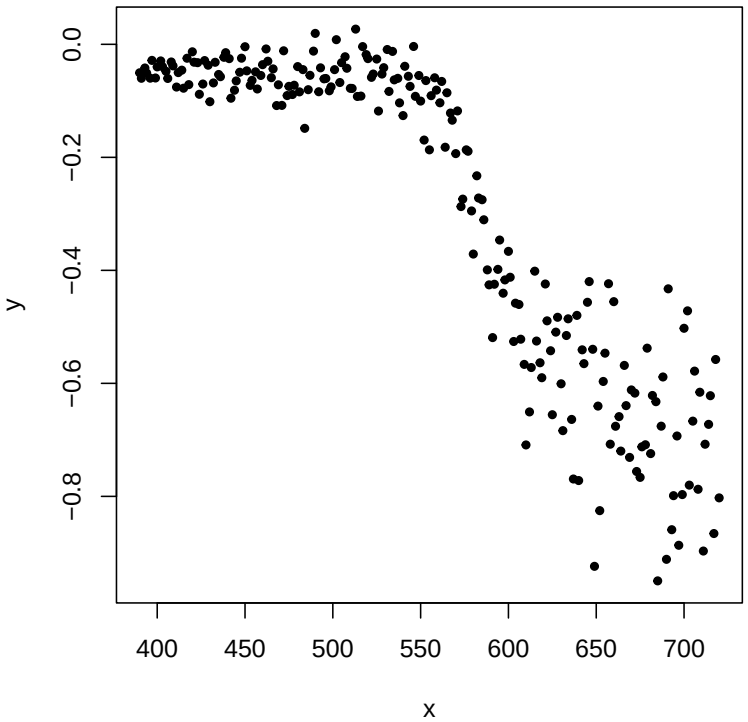
where

$$h(\mathbf{x})^T = \mathbf{x}^T(X^TX)^{-1}X^T$$

LIDAR data: bin smoother

The bin smoother is based on a partition of the covariate space, let the cut points be

$$-\infty = c_0 < c_1 < \ldots < c_{K-1} < c_K = +\infty$$



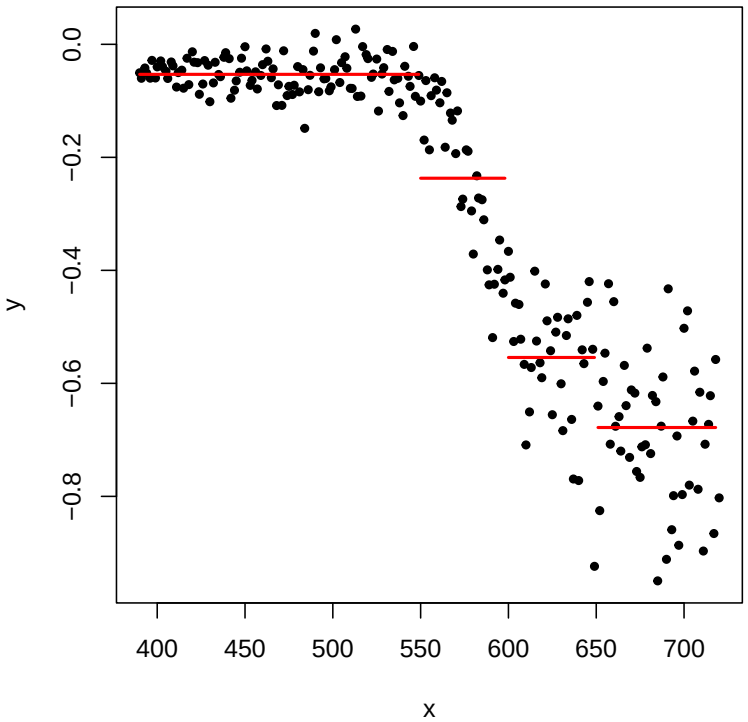
and let

$$\hat{f}(x) = \frac{\sum_{k=0}^{K-1} \sum_{i=1}^{n} y_i I_{[c_k, c_{k+1}]}(x_i)}{\sum_{k=0}^{K-1} \sum_{i=1}^{n} I_{[c_k, c_{k+1}]}(x_i)}$$

LIDAR data: bin smoother

The bin smoother is based on a partition of the covariate space, let the cut points be

$$-\infty = c_0 < c_1 < \ldots < c_{K-1} < c_K = +\infty$$



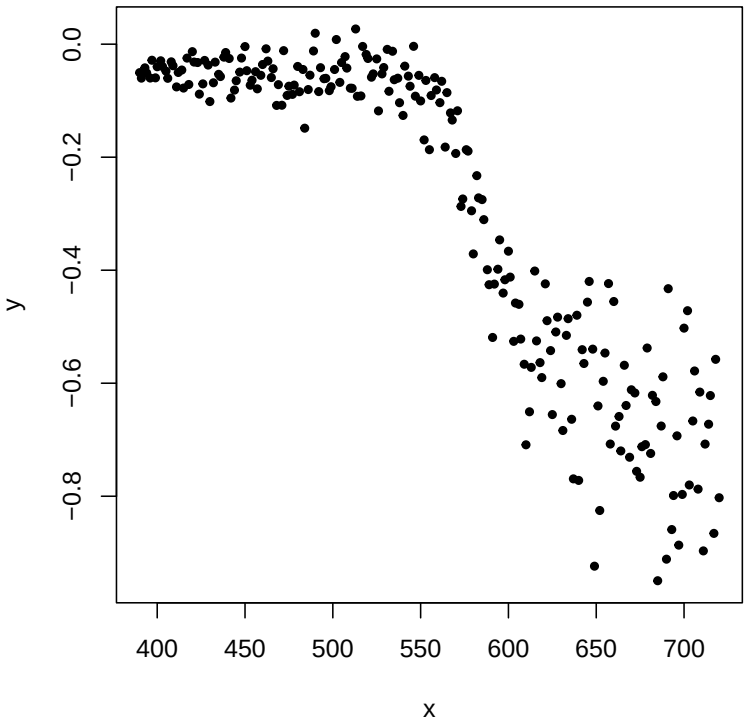
and let

$$\hat{f}(x) = \frac{\sum_{k=0}^{K-1} \sum_{i=1}^{n} y_i I_{[c_k, c_{k+1}]}(x_i)}{\sum_{k=0}^{K-1} \sum_{i=1}^{n} I_{[c_k, c_{k+1}]}(x_i)}$$

Kind of better, not really smooth, depends crucially on the choice of bins.

LIDAR data: running mean

If we are ready to assume that the function $f(x)$ is continuous, then it is reasonable to estimate $f(x)$ with the mean of those value of Y_i corresponding to x_i which lie near x.

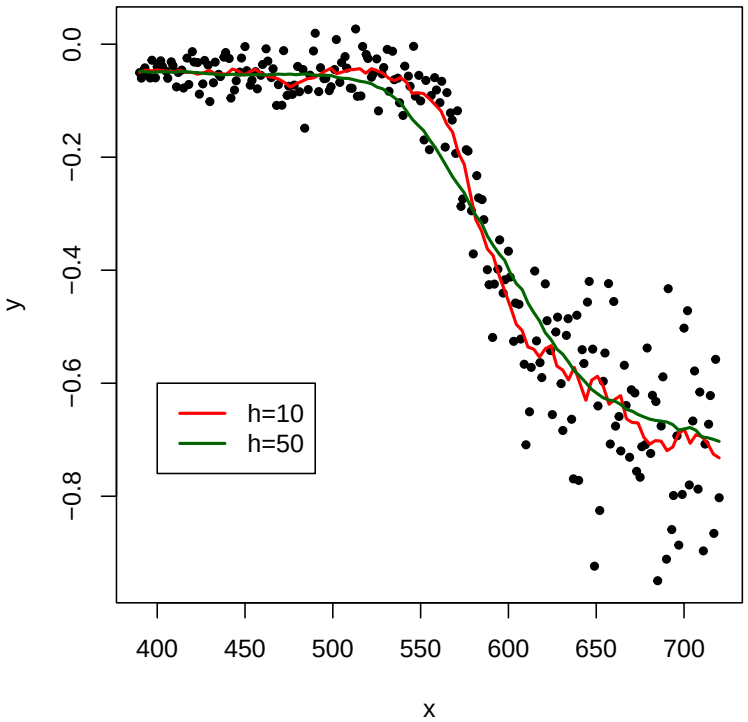


In particular we may use the x_i lying in a neighbourhood of radius h centered in x

$$\hat{f}(x) = \frac{\sum_{i=1}^{n} y_i I_h(|x - x_i|)}{\sum_{i=1}^{n} I_h(|x - x_i|)}$$

LIDAR data: running mean

If we are ready to assume that the function $f(x)$ is continuous, then it is reasonable to estimate $f(x)$ with the mean of those value of Y_i corresponding to x_i which lie near x.

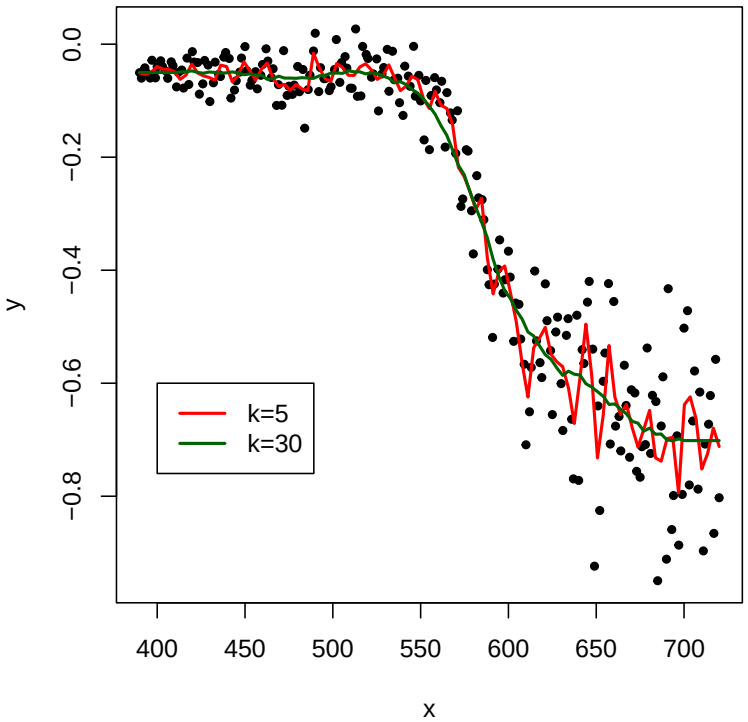


In particular we may use the x_i lying in a neighbourhood of radius h centered in x

$$\hat{f}(x) = \frac{\sum_{i=1}^{n} y_i I_h(|x - x_i|)}{\sum_{i=1}^{n} I_h(|x - x_i|)}$$

LIDAR data: running mean

If we are ready to assume that the function $f(x)$ is continuous, then it is reasonable to estimate $f(x)$ with the mean of those value of Y_i corresponding to x_i which lie near x.



Alternatively one can use the mean of the k nearest neighbours of x, let

$$N_k(x) = \{x_i : |x - x_i| \leq d_{(k)}\}$$

where $d_i = |x - x_i|$ and $d_{(1)} \leq \ldots \leq d_{(n)}$ are the ordered distances, then

$$\hat{f}(x) = \frac{1}{k} \sum_{i=1}^{n} y_i I_{N_k(x)}(x_i)$$

Practical: estimate smooths

```
plot(lidar$range,lidar$logratio,pch=20,xlab="x",ylab="y")

runmean=function(x,xx,yy,h){
  mean(yy[abs(xx-x)<h])
}
runmean2=function(x,xx,yy,h)
  mapply(runmean,x,MoreArgs=list(xx=xx,yy=yy,h=h))
curve(runmean2(x,lidar$range,lidar$logratio,h=10),
      add=TRUE,lwd=2,col="red")

hneighmean=function(x,xx,yy,h){
  mean(yy[abs(xx-x)<sort(abs(xx-x))[h]])
}
hneighmean2=function(x,xx,yy,h)
  mapply(hneighmean,x,MoreArgs=list(xx=xx,yy=yy,h=h))
curve(hneighmean2(x,lidar$range,lidar$logratio,h=5),
      add=TRUE,lwd=2,col="red")
```

Error and bias-variance trade off: estimation error at x

We need to define an error of $\hat{f}_n()$ as an estimate of $f()$, the squared error is a common choice

$$L(f(x), \hat{f}(x)) = (f(x) - \hat{f}(x))^2$$

An overall measure of the error is given by the **mean squared error**

$$MSE_x = R(f(x), \hat{f}(x)) = E(L(f(x), \hat{f}(x)))$$

which, for the quadratic error, can be decomposed as

$$\begin{aligned} &= (f(x) - E(\hat{f}(x)))^2 + V(\hat{f}(x)) \\ &= \text{bias}^2 + \text{variance} \end{aligned}$$

Error and bias-variance trade off: overall estimation error

The errors at each x can be combined to give the **mean integrated square error (MISE)**

$$MISE = \int R(f(x), \hat{f}(x))dx$$

or the **average mean square error**

$$R(f, \hat{f}) = \frac{1}{n}\sum_{i=1}^{n} R(f(x_i), \hat{f}(x_i))$$

Average MSE and prediction error

The MSE is related to the prediction error, suppose we observe a new $Y_i^* = f(x_i) + \varepsilon_i^*$ corresponding to x_i and use $\hat{f}(x_i)$ for prediction. Then the prediction error is

$$(Y_i^* - \hat{f}(x_i))^2$$

which, on average, is

$$MSE_x + V(\varepsilon_i)$$

While the predictive risk is

$$E\left(\frac{1}{n}\sum_{i=1}^{n}(Y_i^* - \hat{f}(x_i))^2\right) = R(f, \hat{f}) + \frac{1}{n}\sum_{i=1}^{n} V(\varepsilon_i^*)$$

Smoothing and error

Consider the k-neighborhood smoother

$$\hat{f}(x) = \frac{1}{k} \sum_{i=1}^{n} y_i I_{N_k(x)}(x_i)$$

where $N_k(x) = \{x_i : |x - x_i| \leq d_{(k)}\}$.

Depending on k the estimate of $\hat{f}(x)$ is based on different observations: the greater k,

- the more observations are used
- on the other hand farther observations are used

The trade off between bias and variance is a distinguishing feature of smoothers.

Smoothing and error

Consider the k-neighborhood smoother

$$\hat{f}(x) = \frac{1}{k} \sum_{i=1}^{n} y_i I_{N_k(x)}(x_i)$$

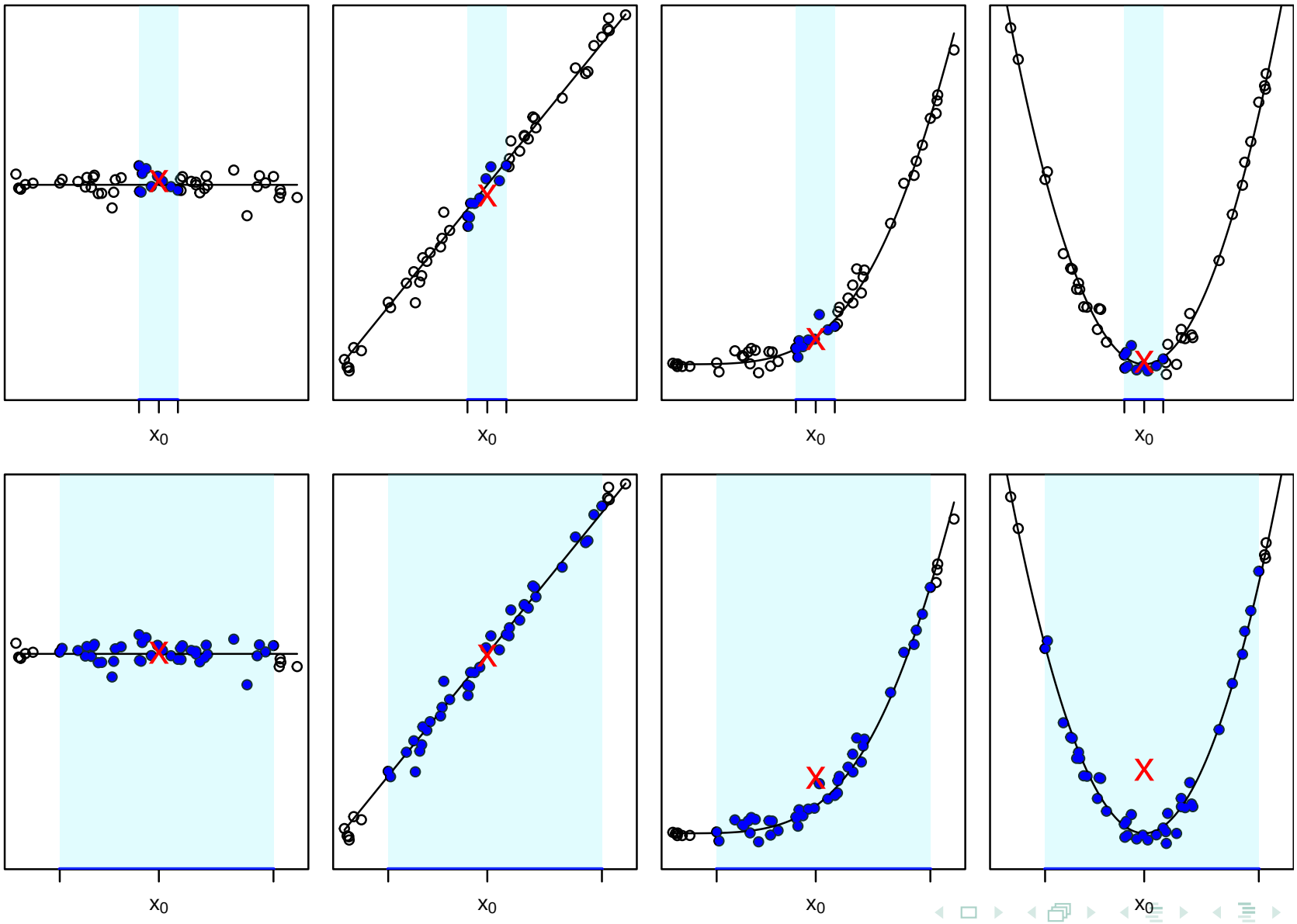
where $N_k(x) = \{x_i : |x - x_i| \leq d_{(k)}\}$.

Depending on k the estimate of $\hat{f}(x)$ is based on different observations: the greater k,

- the more observations are used
  - thus there will be less variability: lower **variance**
- on the other hand farther observations are used
  - depending on the shape of $f()$ in a neighbourhood of x, the mean of the observations may differ more markedly from $E(Y|X = x) = f(x)$: estimate will be more **biased**

The trade off between bias and variance is a distinguishing feature of smoothers.

Shape of f and bias



Smoothing and error: theoretical derivation

Let $N_k(x) = \{x_i : |x - x_i| \leq d_{(k)}\}$, the k-neighborhood smoother is

$$\hat{f}(x) = \frac{1}{k} \sum_{i=1}^{n} y_i I_{N_k(x)}(x_i)$$

Then (assuming $V(Y_i) = \sigma^2$ for all i)

$$V(\hat{f}(x)) = \frac{1}{k} \sum_{i=1}^{k} V(Y_i) = \frac{\sigma^2}{k}$$

Smoothing and error: theoretical derivation

Let $N_k(x) = \{x_i : |x - x_i| \leq d_{(k)}\}$, the k-neighborhood smoother is

$$\hat{f}(x) = \frac{1}{k} \sum_{i=1}^{n} y_i I_{N_k(x)}(x_i)$$

The bias is

$$\begin{aligned} E(\hat{f}(x)) - f(x) &= \frac{1}{k} \sum_{N_k(x)} (f(x_i) - f(x)) \\ &\approx \frac{1}{k} \sum_{N_k(x)} \left(f'(x)(x_i - x) + \frac{1}{2} f''(x)(x_i - x)^2 \right) \end{aligned}$$

assuming the covariates equispaced: $x_{i+1} - x_i = \Delta$

$$\approx \frac{2k(k+2)(k+1)}{6k} f''(x)\Delta^2$$

Smoothing and error: theoretical derivation

Let $N_k(x) = \{x_i : |x - x_i| \leq d_{(k)}\}$, the k-neighborhood smoother is

$$\hat{f}(x) = \frac{1}{k} \sum_{i=1}^{n} y_i I_{N_k(x)}(x_i)$$

Hence the MSE is

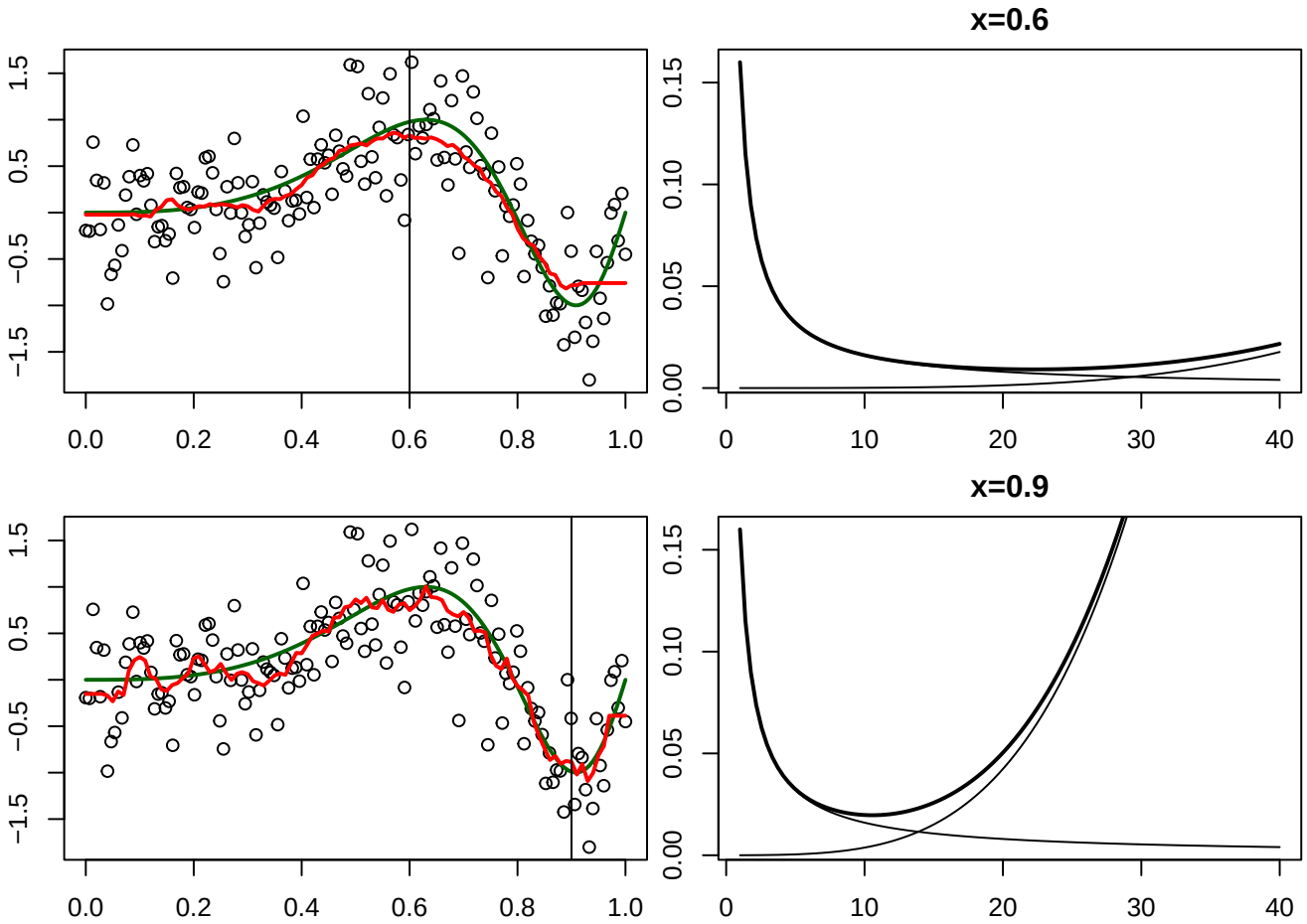
$$E((\hat{f}(x) - f(x))^2) \approx \left(\frac{2k(k+2)(k+1)}{6k} f''(x) \Delta^2 \right)^2 + \frac{\sigma^2}{k}$$

that is

- the bias grows with k and with f''
- the variance decreases with k

Smoothing and error: theoretical bias and variance example

Consider the simulated observations below, where the true f is depicted



Practical: bias and variance through simulation

```
sim=data.frame(x=seq(0,1,length=150)) #sort(runif(150,0,1)))
sim$m=sin(2*pi*sim$x^3)
sim$y=sim$m+rnorm(nrow(sim),0,0.4)
B=100
prev=error=array(NA,dim=c(nrow(sim),40,B))
for (j in 1:B){
  sim$yy=sim$m+rnorm(nrow(sim),0,0.4)
  for (i in 1:40) {
    prev[,i,j]=hneighmean2(sim$x,sim$x,sim$yy,h=i)
    error[,i,j]=prev[,i,j]-sim$m
  }}
errorA=apply(error,c(1,2),FUN=function(x) mean(x^2))
plot(1:40,errorA[135,],col="red",pch=20)
points(1:40,errorA[90,],col="green",pch=20)
errorC=apply(errorA,c(2),FUN=function(x) mean(x^2))
plot(1:40,errorC)
matplot(sim$x,prev[,2,],type="l",col=gray(0.7))
lines(sim$x,sim$m,col="darkgreen",lwd=2)
matplot(sim$x,prev[,10,],type="l",col=gray(0.7))
lines(sim$x,sim$m,col="darkgreen",lwd=2)
matplot(sim$x,prev[,20,],type="l",col=gray(0.7))
lines(sim$x,sim$m,col="darkgreen",lwd=2)
matplot(sim$x,prev[,40,],type="l",col=gray(0.7))
```

Practical 2

The same code can be used to try other smoothers

```
sim=data.frame(x=seq(0,1,length=150)) #sort(runif(150,0,1)))
sim$m=sin(2*pi*sim$x^3)
sim$y=sim$m+rnorm(nrow(sim),0,0.4)
B=100
prev=array(NA,dim=c(nrow(sim),40,B))
for (j in 1:B){
  sim$yy=sim$m+rnorm(nrow(sim),0,0.4)
  for (i in 1:40) {
    prev[,i,j]=......
  }}
prevR=apply(prev,c(1,2),FUN=function(x) mean(x^2))
plot(1:40,prevR[135,],col="red",pch=20)
points(1:40,prevR[90,],col="green",pch=20)
prevC=apply(prev,c(2),FUN=function(x) mean(x^2))
plot(1:40,prevC)
```

Estimate the MSE

In general, we can not obtain the error through simulations since we do not know the generating mechanism and the true curve (the error distribution and $f()$).

Also the theoretical formulas do not really help since they are an approximation and also require knowledge of the function f (of its second derivative actually, which is worse).

We need an estimator for the error from which we will obtain an estimator for the optimal level of smoothing.

Linear smoothers

We discuss error estimation for a class of smoothers which comprises those defined above and many others: **linear smoothers**, that is, smoothers for which there exist a vector $\ell(x) = (\ell_1(x), \ldots, \ell_n(x))^T$ for each x such that

$$\hat{f}(x) = \sum_{i=1}^{n} \ell_i(x) Y_i$$

which means that

$$\hat{\mathbf{f}} = \begin{bmatrix} \hat{f}(x_1) \\ \vdots \\ \hat{f}(x_n) \end{bmatrix} = \begin{bmatrix} \ell_1(x_1) & \cdots & \ell_n(x_1) \\ \vdots & & \\ \ell_1(x_n) & \cdots & \ell_n(x_n) \end{bmatrix} \mathbf{Y} = L\mathbf{Y}$$

The matrix L is the **smoothing matrix**, we define the **effective degrees of freedom** of the smoother as

$$\nu = tr(L)$$

Linear smoothers

Note that the previous smoothers are of the linear type, it is relevant to figure how the L matrix is, below some general indications (assuming without loss of generality that the x_i be ordered).

- For the regressogram the L matrix is a diagonal block matrix assuming value equal to the reciprocal of the number of observations in the block.
- For the k-neighbours the L matrix has a non-zero diagonal 'stripe' valued $1/k$
- For the local average the L matrix is analogous to the k-neighbours if the x_i are regularly spaced, otherwise it will be irregular.

Estimator of risk

The linear smoothers defined above (and also those which will be defined later) depend on some parameter h, whose optimal value is the minimum of

$$R(h) = E\left(\frac{1}{n}\sum_{i=1}^{n}(\hat{f}_h(x_i) - f(x_i))\right)^2$$

Since $R(h)$ is not known (as clarified above) we resort to minimize an estimate of $R(h)$.

The first guess would be the average RSS

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}_h(x_i))^2$$

but this is obviously biased downward and its use would lead to overfitting (undersmoothing).
(In fact, the point is that we use the data twice, once to estimate f, once to estimate the error.)

Loo-cv for smoothers

A better estimate for $R(h)$ is

$$CV = \hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}_{h,-i}(x_i))^2$$

where $\hat{f}_{h,-i}(x_i)$ is the smoother estimated without the i-th observation. Note that

$$\begin{aligned} E(Y_i - \hat{f}_{h,-i}(x_i))^2 &= E(Y_i - f(x_i) + f(x_i) - \hat{f}_{h,-i}(x_i))^2 \\ &= \sigma^2 + E(f(x_i) - \hat{f}_{h,-i}(x_i))^2 \\ &\approx \sigma^2 + E(f(x_i) - \hat{f}_h(x_i))^2 \end{aligned}$$

that is, $\hat{R}$ is an approximately unbiased estimator of the predictive risk

$$E(\hat{R}) \approx R + \sigma^2$$

Loo-cv for linear smoothers

If the smoother is linear with smoothing matrix L then

$$CV = \hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}_{-i}(x_i))^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{f}(x_i)}{1 - L_{ii}}\right)^2$$

So that one does not need to recompute the smoother but only to know the value of L_{ii} (which may not require to know the full matrix, otherwise the advantage would be minor).

A further simplification is the **generalized cross validation criterion** where we substitute L_{ii} with its average

$$GCV = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{f}(x_i)}{1 - \nu/n}\right)^2$$

Derivation of GCV

We did not define precisely $\hat{f}_{-i}(x_i)$, given that

$$\hat{f}(x_i) = \sum_{j=1}^{n} \ell_j(x_i) y_j$$

and assuming that $\sum_{j=1}^{n} \ell_j(x_i) = 1$ (that is, the smoother preserves constants, which is reasonable), we may define

$$\hat{f}_{-i}(x_i) = \frac{\sum_{j \neq i} \ell_j(x_i) y_j}{\sum_{j \neq i} \ell_j(x_i)} = \frac{\sum_{j \neq i} \ell_j(x_i) y_j}{1 - \ell_i(x_i)} = \frac{\sum_{j \neq i} \ell_j(x_i) y_j}{1 - L_{ii}}$$

Note that we may define $\hat{f}_{-i}()$ as the smoother re-estimated without observation (x_i, y_i), this is the same as above for the radius smoother, not for the $k-$neighborhood.

Derivation of GCV (continua)

Using the above formula we get

$$
\begin{aligned}
y_i - \hat{y}_{-i} &= y_i - \frac{1}{1 - L_{ii}} \sum_{j \neq i} \ell_j(x_i) y_j \\
&= y_i - \frac{1}{1 - L_{ii}} \left(\sum_{j=1}^{n} \ell_j(x_i) y_j - L_{ii} y_i \right) \\
&= y_i - \frac{1}{1 - L_{ii}} (\hat{y}_i - L_{ii} y_i) \\
&= \frac{1}{1 - Lii} ((1 - L_{ii}) y_i - \hat{y}_i + L_{ii} y_i) = \frac{1}{1 - Lii} (y_i - \hat{y}_i)
\end{aligned}
$$

hence the GCV formula.

Other criteria

Note that, since $(1-x)^{-2} \approx 1+2x$ in a neighbourhood of 0, the GCV is approximately the same as Mallow's C_p.

$$GCV = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{f}(x_i)}{1-\nu/n}\right)^2 \approx \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{f}(x_i)\right)^2 + \frac{2\nu\hat{\sigma}^2}{n} = C_p$$

More generally, many common bandwidth selection criteria have the form

$$B(h) = \Lambda(n,h)\frac{1}{n} + \sum_{i=1}^{n}\left(Y_i - \hat{f}(x_i)\right)^2$$

for some function $\Lambda(\cdot,\cdot)$

Asymptotic properties

Let

- $h_0 = \text{argmin} R(h)$
- $\hat{h}_0 = \text{argmin} L(h) = \text{argmin} \frac{1}{n} \sum_{i=1}^{n} (\hat{f}(x_i) - f(x_i))^2$
- $\hat{h} = \text{argmin} B(h)$

then, under appropriate conditions it can be shown that,

i. $\hat{h}$, $\hat{h}_0$, h_0 are $o(n^{-1/5})$

ii. $n^{3/10}(\hat{h} - \hat{h}_0) \to N(0, \sigma_1^2)$

iii. $n^{3/10}(h_0 - \hat{h}_0) \to N(0, \sigma_2^2)$

iv. $\frac{\hat{h} - \hat{h}_0}{\hat{h}_0} = O_p\left(\frac{n^{3/10}}{n^{1/5}}\right) = O_p\left(n^{-1/10}\right)$

v. $\frac{\hat{h} - \hat{h}_0}{h_0} = O_p\left(\frac{n^{3/10}}{n^{1/5}}\right) = O_p\left(n^{-1/10}\right)$

These results show how hard it is to estimate the optimal bandwidth!

Practical: estimating the risk and the bandwidth

```
sim=data.frame(x=seq(0,1,length=150)) #sort(runif(150,0,1)))
sim$m=sin(2*pi*sim$x^3)
sim$y=sim$m+rnorm(nrow(sim),0,0.4)
B=100
prev=error=array(NA,dim=c(nrow(sim),40,B))
for (j in 1:B){
  sim$yy=sim$m+rnorm(nrow(sim),0,0.4)
  for (i in 1:40) {
    prev[,i,j]=hneighmean2(sim$x,sim$x,sim$yy,h=i)
    error[,i,j]=prev[,i,j]-sim$m
  }}
errorC=apply(errorA,c(2),FUN=function(x) mean(x^2))

R=rep(NA,40)
for (i in 2:40){
  R[i]=mean(((sim$y-hneighmean2(sim$x,sim$x,sim$y,h=i))/(1-1/i))^2)
}
matplot(1:40,cbind(errorC,R-0.16),col=c("black","red"),pch=1)
```

Indice

Kernel regression

In the above methods as we move on the x axis we compute $\hat{f}(x)$ as a mean of different groups of y_i.

This leads to some roughness in the final estimate.

One way to reduce this roughness is to use a weighted average which gives less weight to those values which are farther from x.

Nadaraya-Watson estimator

The Nadaraya-Watson estimator is a linear smoother

$$\hat{f}(x) = \sum_{i=1}^{n} \ell_i(x) Y_i$$

in which

$$\ell_i(x) = \frac{K\left(\frac{x-x_i}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x-x_j}{h}\right)}$$

$K()$ being a kernel (see density estimation).

Kernel functions

The estimate is a rather un-smooth function, it can be made smoother by using a different kernel

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

where the kernel function K is such that

- $K(x) \geq 0$
- $\int K(x)dx = 1$
- $\int xK(x)dx = 0$
- $\int x^2 K(x)dx > 0$

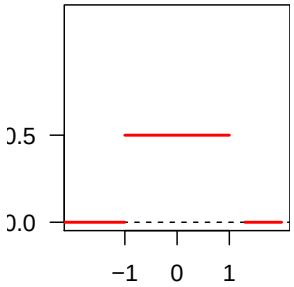
Possible kernels include

	$K(u)$
Uniform	$\frac{1}{2} I_{[-1,1]}(u)$
Triangle	$(1-\lvert u\rvert) I_{[-1,1]}(u)$
Triweight	$\frac{35}{32}(1-u^2)^3 I_{[-1,1]}(u)$
Quartic	$\frac{15}{16}(1-u^2)^2 I_{[-1,1]}(u)$
Gaussian	$\frac{1}{\sqrt{2\pi}} e^{-u^2/2}$
Epanechnikov	$\frac{3}{4}(1-u^2) I_{[-1,1]}(u)$
Cosine	$\frac{\pi}{4}\cos\left(\frac{\pi}{2}u\right) I_{[-1,1]}(u)$

Kernel functions

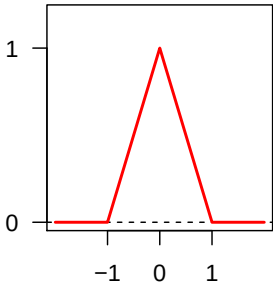
Uniform

$$\frac{1}{2} I_{[-1,1]}(u)$$



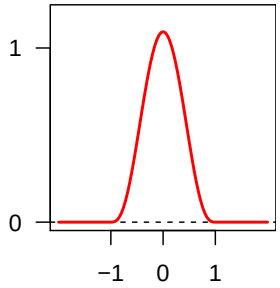
Triangle

$$(1 - |u|) I_{[-1,1]}(u)$$



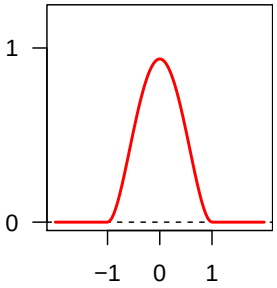
Triweight

$$\frac{35}{32}(1 - u^2)^3 I_{[-1,1]}(u)$$



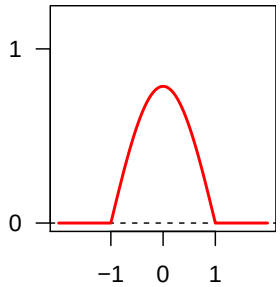
Quartic

$$\frac{15}{16}(1 - u^2)^2 I_{[-1,1]}(u)$$



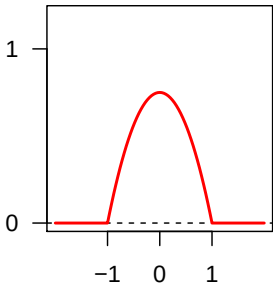
Cosine

$$\frac{\pi}{4} \cos\left(\frac{\pi}{2} u\right) I_{[-1,1]}(u)$$



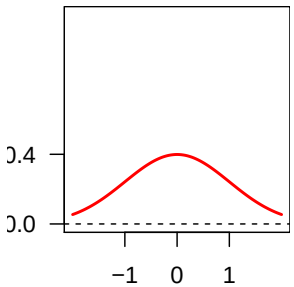
Epanechnikov

$$\frac{3}{4}(1 - u^2) I_{[-1,1]}(u)$$



Gaussian

$$\frac{1}{\sqrt{2\pi}} e^{-u^2/2}$$



Nadaraya-Watson estimator: risk

It can be shown that, assuming x_i come from the density $g()$, for $h_n \to 0$ and $nh_n \to \infty$

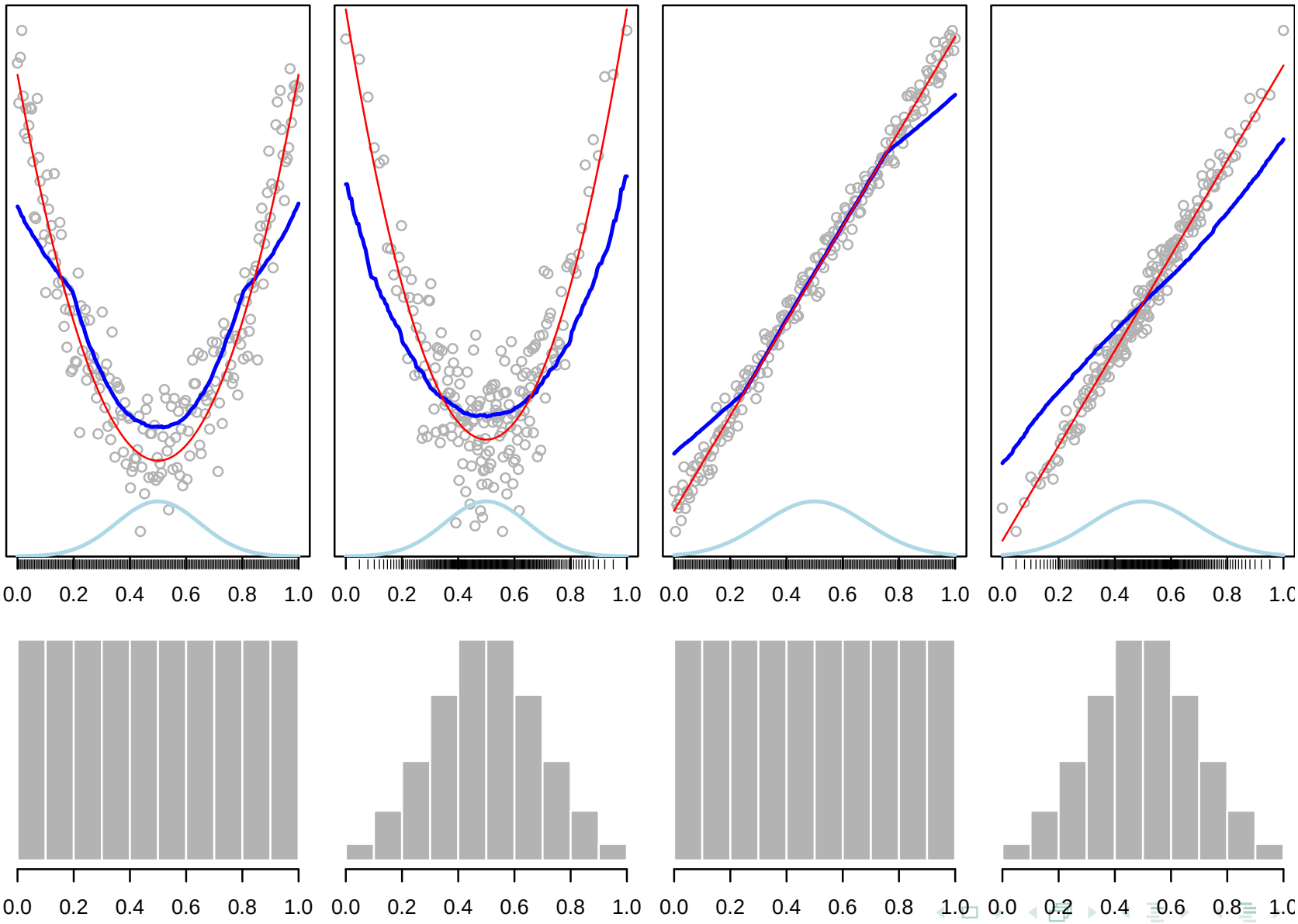
$$R = \frac{h_n^4}{4}\left(\int u^2 K(u)du\right)^2 \int \left(f''(x) + 2f'(x)\frac{g'(x)}{g(x)}\right)^2 dx + \frac{\sigma^2 \int K^2(u)du}{nh_n}\int \frac{1}{g(x)}dx + o(nh_n^{-1}) + o(h_n^4)$$

We note

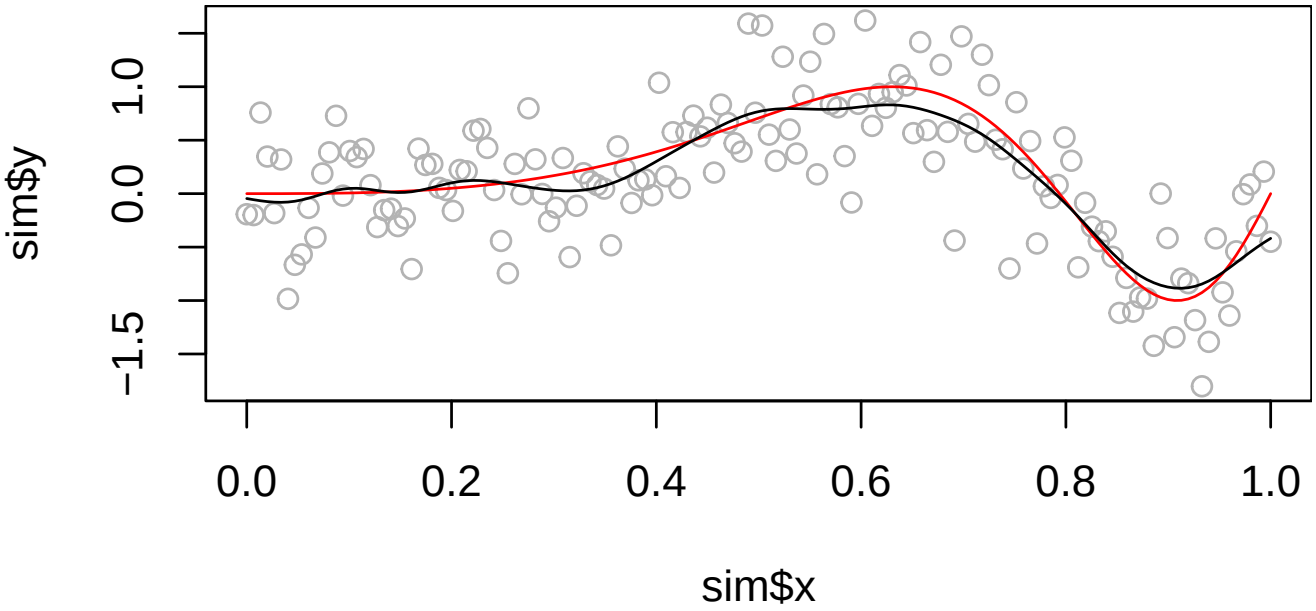
- variance decreases with h
- bias increases with h^4
- bias increases with f''
- bias increases with $f'(x)\frac{g'(x)}{g(x)}$: **design bias**

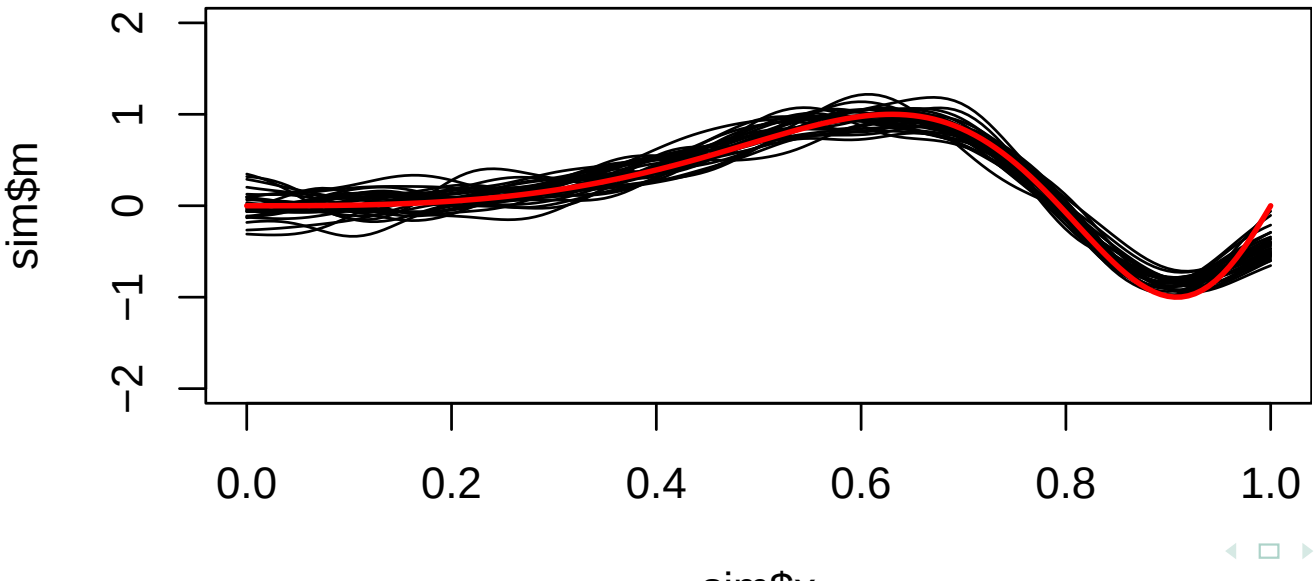
By obtaining the optimal h and substituting back in the expression for R we can see that the risk is $O(n^{-4/5})$, a slower rate than (most) parametric model ($O(n^{-1})$), the slower rate being the price for making less assumptions.

Design bias and boundary bias



Boundary bias





N-W as a minimizer

We note that the N-W estimator at x, $\hat{f}(x)$, is the solution of

$$\underset{a}{\operatorname{argmin}} \sum_{i=1}^{n} K_i \left(\frac{x_i - x}{h} \right) (Y_i - a)^2$$

that is, the N-W estimator is obtained locally as a weighted least square estimator.

The idea is then to employ weighted least squares with a polynomial rather than a constant, for any value of x we approximate $f()$ in a neighbourhood of x by the polynomial

$$p_x(u; \mathbf{a}) = a_0 + a_1(u - x) + \frac{a_2}{2!}(u - x)^2 + \ldots + \frac{a_p}{p!}(u - x)^p$$

N-W as a minimizer → local polynomials

The idea is then to employ weighted least squares with a polynomial rather than a constant, for any value of x we approximate $f()$ in a neighbourhood of x by the polynomial

$$p_x(u; \mathbf{a}) = a_0 + a_1(u - x) + \frac{a_2}{2!}(u - x)^2 + \ldots + \frac{a_p}{p!}(u - x)^p$$

We estimate $\mathbf{a}(x)$ (making the dependence on x explicit) by minimizing the weighted sum of squares

$$\hat{\mathbf{a}}(x) = \underset{a}{\operatorname{argmin}} \sum_{i=1}^{n} K_i \left(\frac{x_i - x}{h} \right) (Y_i - p_x(X_i; \mathbf{a}))^2$$

and define the estimator of $f(x)$ as

$$\hat{f}(x) = p_x(x, \hat{\mathbf{a}}) = \hat{a}_0(x)$$

Local polynomial estimator: matrix notation

Let

$$X_x = \begin{bmatrix} 1 & x_1 - x & \cdots & \frac{1}{p!}(x_1 - x)^p \\ \vdots & \vdots & & \vdots \\ 1 & x_n - x & \cdots & \frac{1}{p!}(x_n - x)^p \end{bmatrix}$$

$$W_x = \text{diag}\left\{ K_i\left(\frac{x_i - x}{h}\right), i = 1, \ldots, n \right\}$$

then the weighted sum of squares is

$$(\mathbf{Y} - X_x\mathbf{a})^T W_x (\mathbf{Y} - X_x\mathbf{a})$$

and

$$\hat{\mathbf{a}} = (X_x^T W_x X_x)^T X_x^T W_x \mathbf{Y}$$

Local polynomial estimator: matrix notation

$$\hat{\mathbf{a}} = (X_x^T W_x X_x)^T X_x^T W_x \mathbf{Y}$$

The estimator $\hat{f}(x) = \hat{a}_0(x)$ is then

$$\hat{f}(x) = e_1^T (X_x^T W_x X_x)^T X_x^T W_x \mathbf{Y}$$

where $e_1^T = (1, 0, \ldots, 0)$.
In other terms $\hat{f}(x)$ is a linear smoother

$$\hat{f}(x) = \sum_{i=1}^{n} \ell_i(x) Y_i$$

where

$$\ell(x)^T = (\ell_1(x), \ldots, \ell_n(x))^T = e_1^T (X_x^T W_x X_x)^T X_x^T W_x$$

Local linear smoothing

Assuming $p = 1$, we get the local linear smoother, which has

$$\ell_i(x) = \frac{b_i(x)}{\sum_{j=1}^{n} b_j(x)}$$

where

$$b_i(x) = K\left(\frac{x_i - x}{h}\right)(S_{n,2}(x) - (x_i - x)S_{n,1}(x))$$

$$S_{n,j}(x) = \sum_{i=1}^{n} K\left(\frac{x_i - x}{h}\right)(x_i - x)^j, \quad j = 1, 2$$

Local linear smoother: bias and variance

It can be shown under regularity assumption that the risk at x for the local linear smother is

$$R_x = \frac{h_n^4}{4}\left(\int u^2 K(u)du\right)^2 f''(x)^2 + \frac{\sigma^2 \int K^2(u)du}{g(x)nh_n} + o(nh_n^{-1}) + o(h_n^4)$$

Local linear smoother: bias and variance

It can be shown under regularity assumption that the risk at x for the local linear smother is

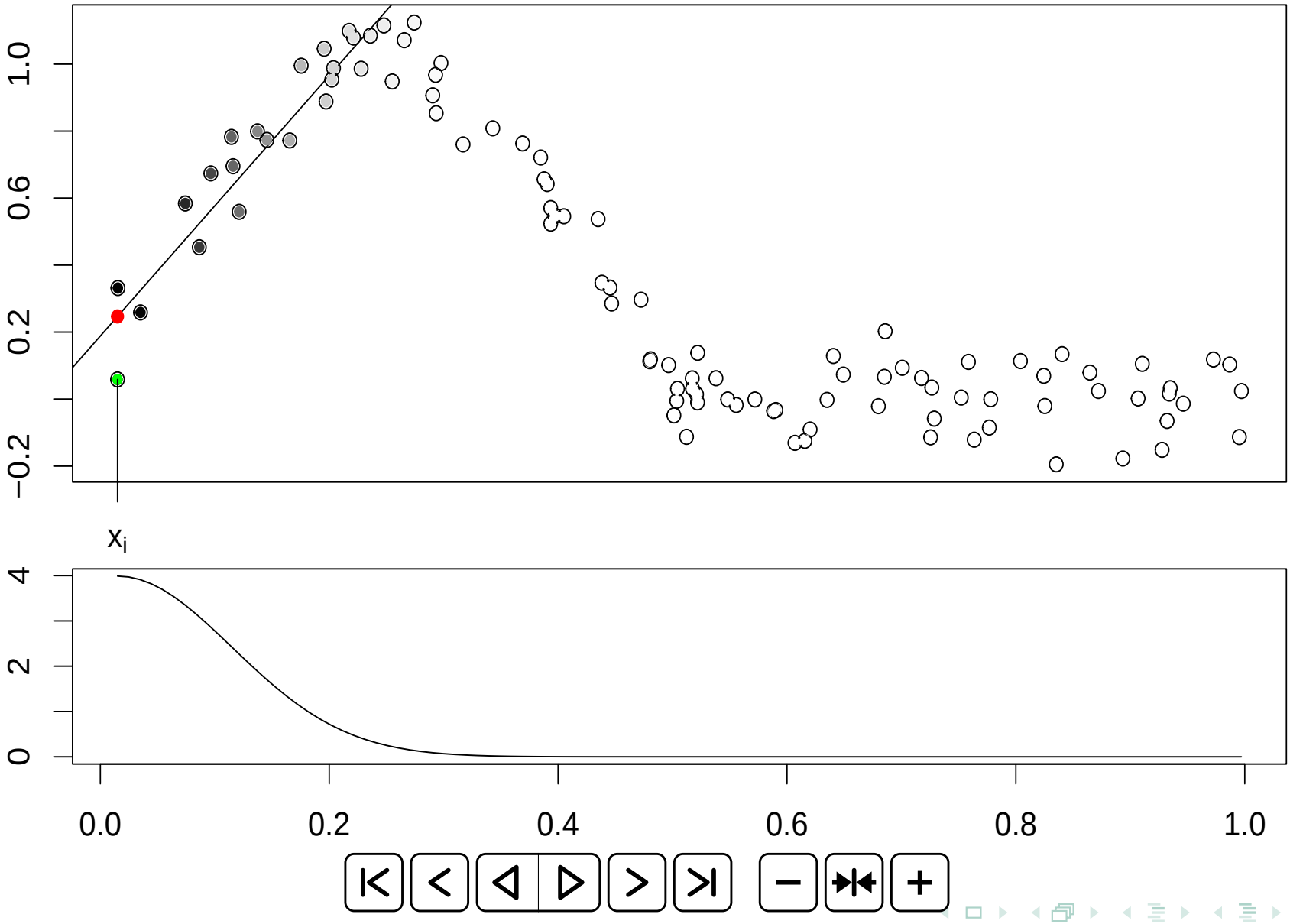
$$R_x = \frac{h_n^4}{4}\left(\int u^2 K(u)du\right)^2 f''(x)^2 + \frac{\sigma^2 \int K^2(u)du}{g(x)nh_n} + o(nh_n^{-1}) + o(h_n^4)$$

If we compare this to the risk for the N-W estimator we note that there is no design bias.

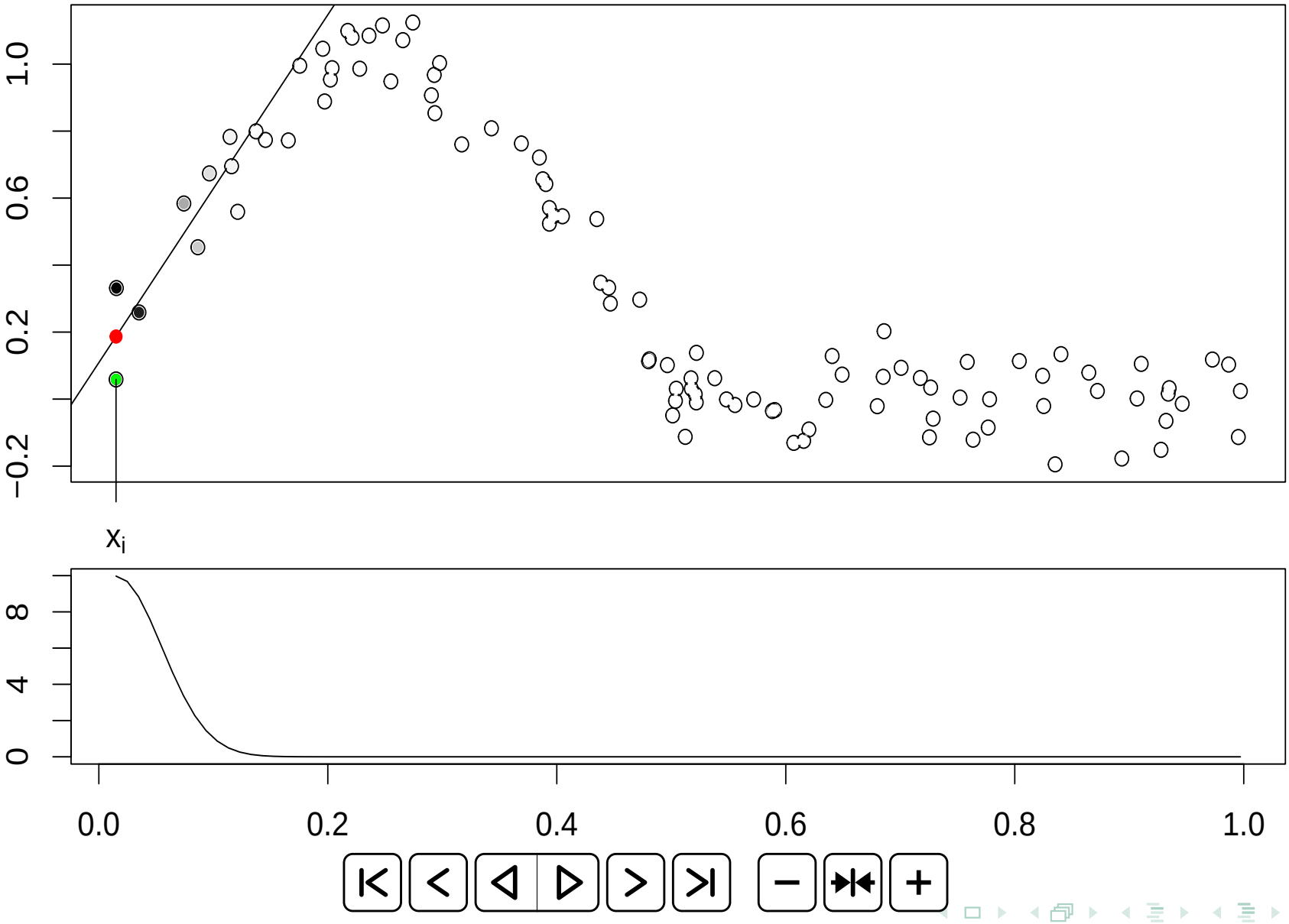
$$\begin{aligned} R =& \frac{h_n^4}{4}\left(\int u^2 K(u)du\right)^2 \int \left(f''(x) + 2f'(x)\frac{g'(x)}{g(x)}\right)^2 dx \\ &+ \frac{\sigma^2 \int K^2(u)du}{nh_n}\int \frac{1}{g(x)}dx + o(nh_n^{-1}) + o(h_n^4) \end{aligned}$$

Furthermore, the asymptotic bias at the boundaries is $o(h_n^2)$ rather than $o(h_n)$.

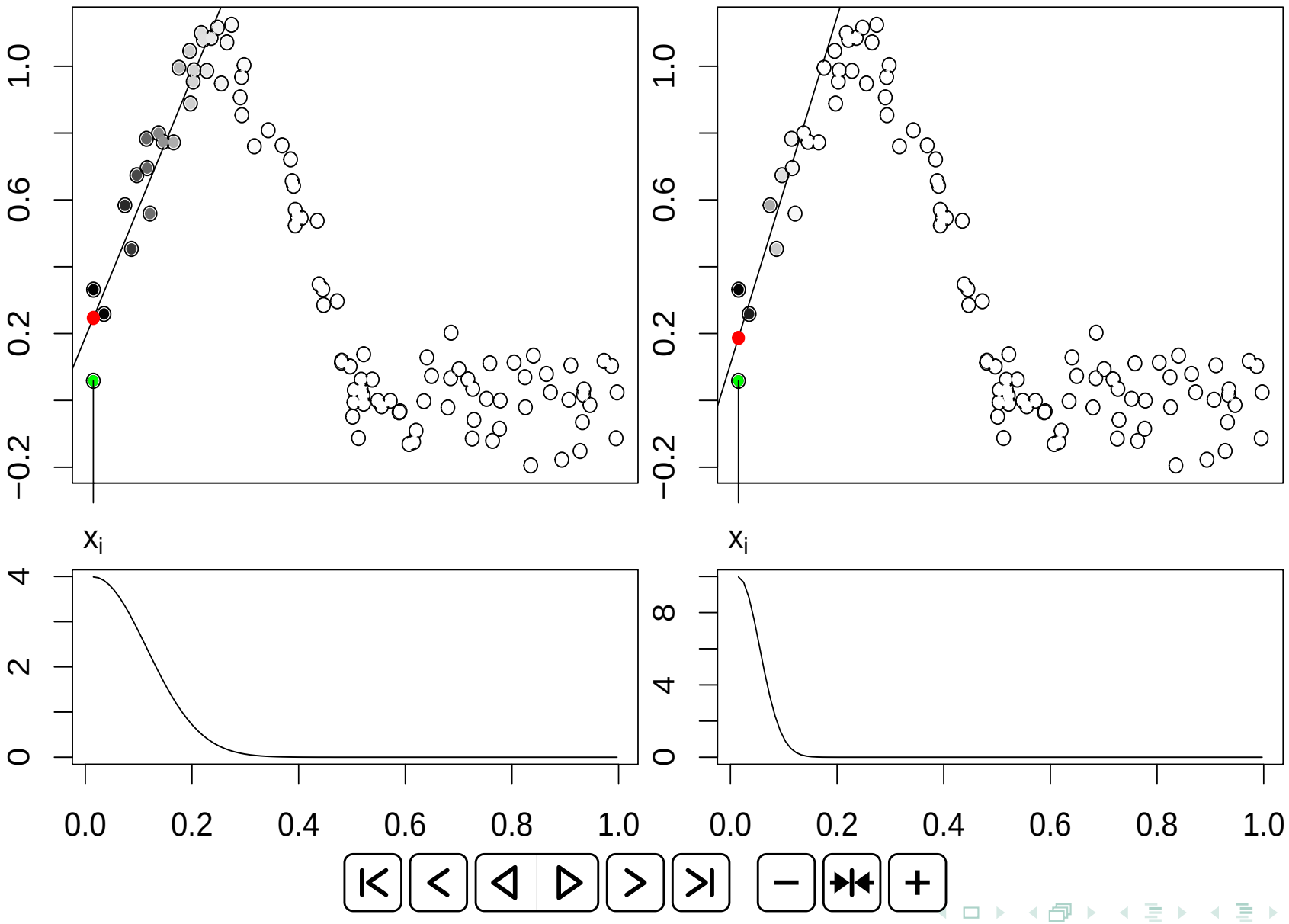
Graphical representation



Graphical representation

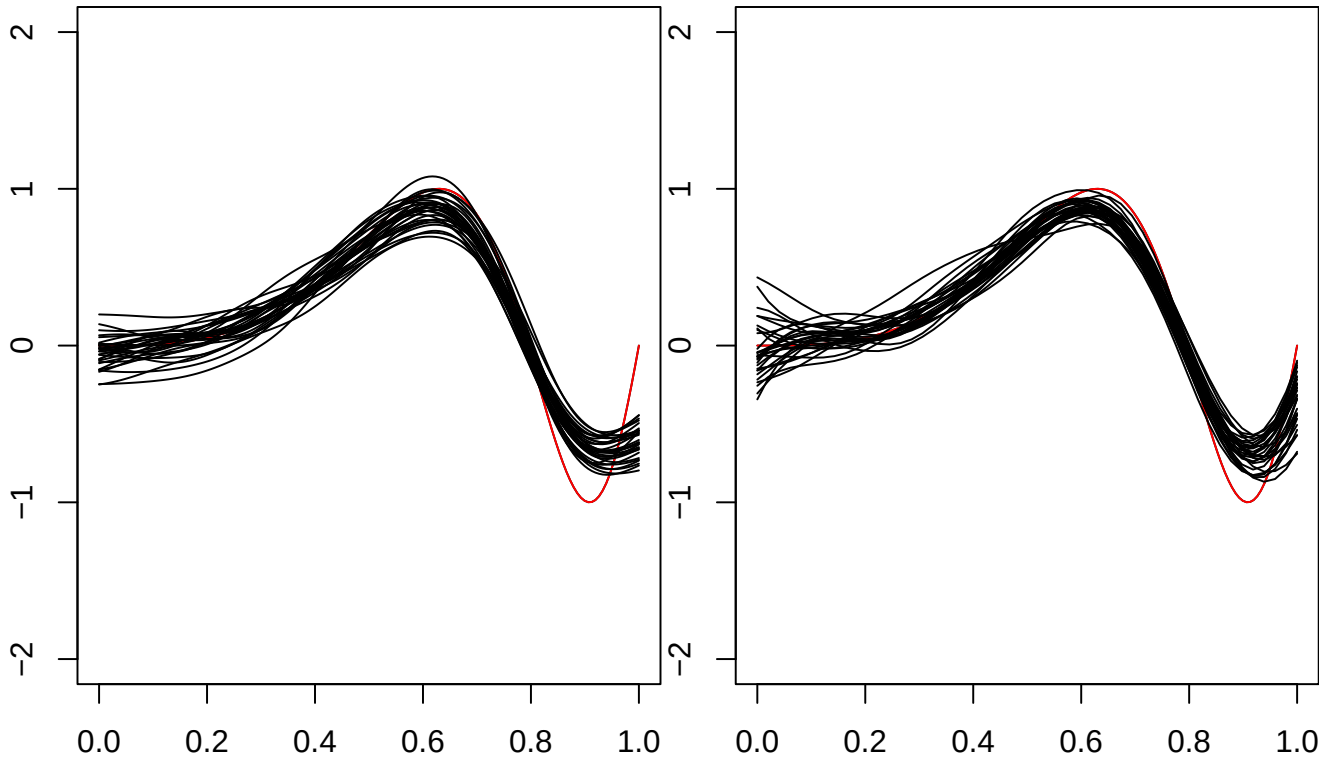


Graphical representation: comparison

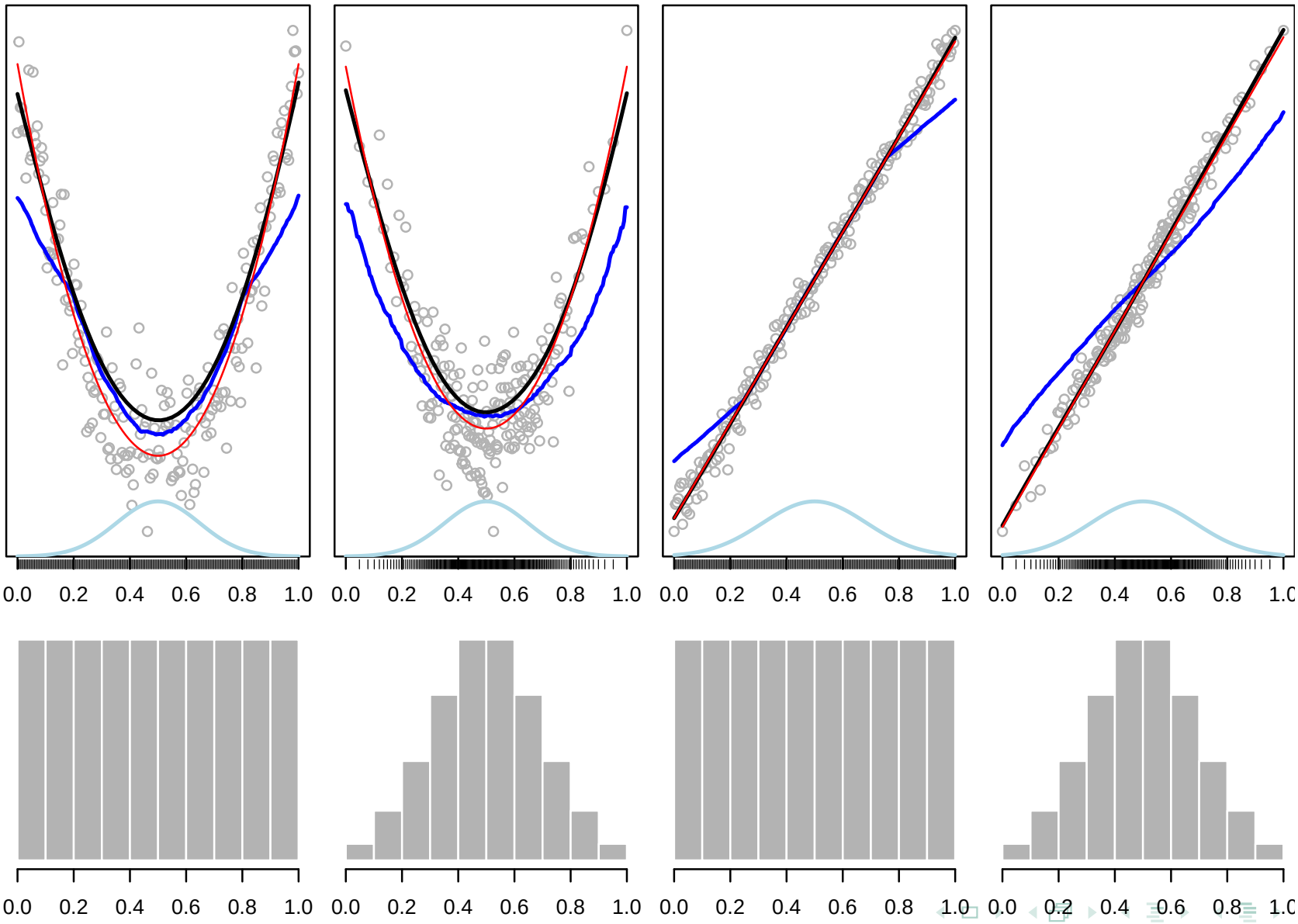


Boundary bias

N-W versus local linear, same bandwith



Design bias and boundary bias



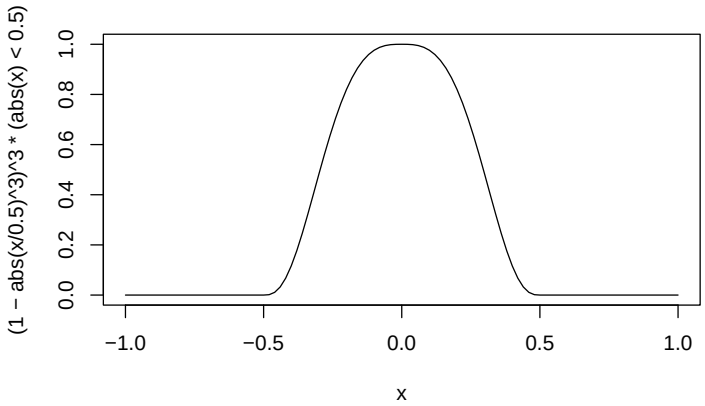
Varying bandwith

The local estimator $\hat{f}(x)$ with a kernel function with a bounded support (for instance the tricube which is non zero on $[-h, h]$) is based on a varying number of points depending on the distribution of the x_i.

A variant involves choosing the scale of the kernel so that a fixed proportion α of points have non zero weight, that is

$$h_i = d_k(x)$$

where $d_k(x)$ is the distance of the k-th nearest neighbour from x.



Tricube kernel

$$K(x) = \left(1 - \left(\frac{x}{h}\right)^3\right)^3 I_{[-h,h]}(x)$$

Indice

Variability bands

In principle, confidence bands for $f(x)$ may be obtained from the approximate result

$$\frac{\hat{f}(x) - E(\hat{f}(x))}{V(\hat{f}(x))} = \frac{\hat{f}(x) - f(x) - \text{bias}_x}{V(\hat{f}(x))} \sim N(0,1)$$

this, however, requires knowing the bias, which would make things complicated.

A common strategy is to ignore bias, so that the bands are actually variability bands around $E(\hat{f}(x))$, which affects interpretation.

Bootstrap and variability bands

Bootstrap strategies, either non parametric or semi parametric can be used to assess the variability of $\hat{f}(x)$.

- In non parametric bootstrap a bootstrap sample is obtained by resmpling the pairs (x_i, Y_i).
- In semi parametric bootstrap we
  - estimate $\hat{f}(x)$ and compute $\varepsilon_i = y_i - \hat{f}(x_i)$
  - sample the residuals ε_i to obtain $(\varepsilon_1^*, \ldots, \varepsilon_n^*)$
  - Obtain the bootstrap sample as

$$(x_i, Y_i^* = \hat{f}(x_i) + \varepsilon_i^*)$$

Whatever the sampling strategy, we repeat the procedure B times and obtain a sample

$$\hat{f}^{*1}(), \ldots, \hat{f}^{*B}()$$

Note that these reflect variability around $\hat{f}(x)$ (we do not allow for bias).

Practical: bootstrap

Simulate the sample

```
x=sort(runif(100,0,1))
m=sin(2*pi*x^3)
y=m+rnorm(length(x),0,0.4)
```

Use the two bootstrap strategies and an estimator to obtain variability bands.

- use different strategies for the bandwidth
- using fixed bandwith at different levels note that bias is not allowed for

Model comparison

Suppose that we want to compare the estimated $\hat{f}(x)$ to the hypothesis

$$H_0 : f(x) = f_0$$

Under H_0 the estimate is $\hat{f}_0(x) = \bar{y}$, the model fits may be compared through the residual sums of squares

$$\begin{aligned} RSS_0 &= \sum_{i=1}^{n} (y_i - \bar{y})^2 \\ RSS_1 &= \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2 \end{aligned}$$

using the statistic

$$F = \frac{\frac{RSS_0 - RSS_1}{\nu_0 - \nu_1}}{\frac{RSS_1}{\nu_1}} \quad \text{where } \nu_j \text{ are the edf.}$$

Model comparison

$$F = \frac{\frac{RSS_0 - RSS_1}{\nu_0 - \nu_1}}{\frac{RSS_1}{\nu_1}} \quad \text{where } \nu_j \text{ are the edf.}$$

Unlike in linear model, however, the distribution of F is not easily determined.

A possible strategy is then to perform a permutation test:

1. We randomly pair the observed x_i and Y_i leading to a sample (x_i, Y_i^*)
2. The two models are estimated on the (x_i, Y_i^*) sample obtaining the corresponding $\hat{f}_0^*$, $\hat{f}^*$ estimates and the F^* statistic.
3. Steps 1 and 2 are repeated B times obtaining a sample from the distribution of F under the null $\{F^{*1}, \ldots, F^{*B}\}$, the empirical p-value of the test is obtained based on this.

Reference bands

Related to the variability band is the idea of reference bands to compare the non parametric estimate with a base model.

The $\hat{f}^{*j}(x)$ obtained through the permutation strategy above may be used to provide a band where the non parametric regression curve is expected to lie if the null model is correct.

Practical

- simulate a sample where $f(x) = m_0$

```
x=runif(100,0,1)
y=rnorm(100,0,2)
```

- use the above strategy to test the hypothesis $H_0 : f(x) = m_0$

Repeat the experiment simulating

- from a linear model
- from a trigonometric function

```
x=runif(100,0,1)
y=rnorm(100,sin(2*pi*x),2)
```

Practical

- simulate a sample where $f(x) = m_0$

```
x=runif(100,0,1)
y=rnorm(100,0,2)
```

- use the above strategy to test the hypothesis $H_0 : f(x) = m_0$

Repeat the experiment simulating

- from a linear model
- from a trigonometric function

```
x=runif(100,0,1)
y=rnorm(100,sin(2*pi*x),2)
```

The `sm.regression` function can be used to perform the procedure

```
sm.regression(x, y, model = "no effect", se = TRUE)
```

Practical

- simulate a sample where $f(x) = m_0$

```
x=runif(100,0,1)
y=rnorm(100,0,2)
```

- use the above strategy to test the hypothesis $H_0 : f(x) = m_0$

Repeat the experiment simulating

- from a linear model
- from a trigonometric function

```
x=runif(100,0,1)
y=rnorm(100,sin(2*pi*x),2)
```

The `sm.regression` function does also perform the test and bands for linearity

```
sm.regression(x, y, model = "linear", se = TRUE)
```

R functions: ksmooth (stats)

The Nadaraya-Watson kernel regression estimate.

```
ksmooth(x, y, kernel = c("box", "normal"),
        bandwidth = 0.5,
        range.x = range(x),
        n.points = max(100L, length(x)), x.points)
```

Note that

dwidth The kernels are scaled so that their quartiles (viewed as probability densities) are at +/- 0.25*bandwidth.

R functions: sm.regression (sm)

Performs local linear regression

```
sm.regression(x, y, h, design.mat = NA, model = "none",
              weights = NA,
              group = NA, ...)
```

Try the panel option (requires additional packages)

```
sm.regression(sim$x,sim$y,h = 0.07,panel=TRUE)
```

R functions: loess (stats)

Performs local polynomial regression

```
loess(formula, data, weights, subset, na.action, model = FALSE,
      span = 0.75, enp.target, degree = 2,
      parametric = FALSE, drop.square = FALSE, normalize = TRUE
      family = c("gaussian", "symmetric"),
      method = c("loess", "model.frame"),
      control = loess.control(...), ...)
```

Fitting is done locally. That is, for the fit at point x, the fit is made using points in a neighbourhood of x, weighted by their distance from x (with differences in parametric ... variables being ignored when computing the distance). The size of the neighbourhood is controlled by α (set by `span` or `enp.target`). For $\alpha < 1$, the neighbourhood includes proportion α of the points, and these have tricubic weighting (proportional to `{(1-(dist/maxdist)^3)^3}`). For $\alpha > 1$, all points are used, with the maximum distance assumed to be $\alpha^{(1/p)}$ times the actual maximum distance for p explanatory variables.