Contents

10 Smoothing Methods and Robust Model Fitting

10.1 Density Estimation

- Suppose a random sample $X_1, \cdots, X_n$ are from a population with an unknown continuous probability density function $f(x)$.
- **Goal**: estimate $f(x)$.

10.1.1 Histogram

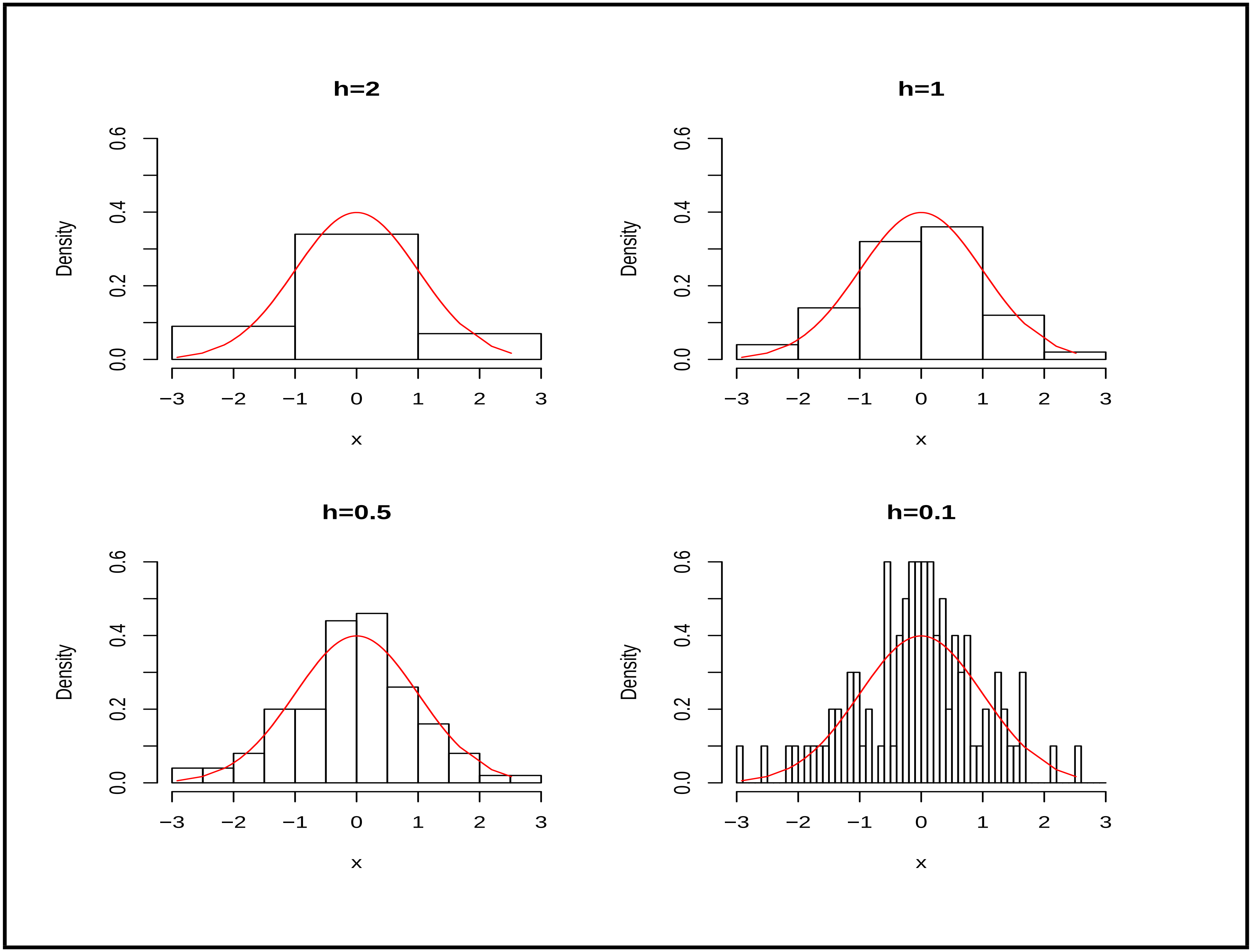
- Partition the range of data into subintervals $a_1 < a_2 < \cdots < a_k$.
- For $x \in (a_i, a_i + 1]$,

$$\hat{f}(x) = \frac{\text{No of observations } \in (a_i, a_{i+1}]}{n(a_{i+1} - a_i)}$$

- Typically, the intervals have equal length h: $a_{i+1} - a_i = 2h$ for $i = 1, \cdots, k$. The equal interval length h is often referred to as **bandwidth**. In this case, for any x within the data range,

$$\begin{aligned}\hat{f}(x) &= \frac{\text{No of observations within } h \text{ of } x}{2nh} \\ &= \frac{1}{2nh}\sum_{i=1}^{n} I(|X_i - x| \leq h).\end{aligned}$$

- Narrow intervals (smaller h): histogram has a choppy appearance, large variance.
- Large intervals (larger h): histogram may lose the local feature, large bias.
- Generally, a larger sample size n requires a smaller h, thus more intervals.
- Suggested $h = \frac{1.75}{n^{1/3}} S$, S is the sample standard deviation.


h=2
h=1
h=0.5
h=0.1
Density
0.0
0.2
0.4
0.6
−3
−2
−1
0
1
2
3
x

10.1.2 Kernel Density Estimation

- Recall the histogram estimator

$$\hat{f}(x) \quad = \quad \frac{1}{2nh}\sum_{i=1}^{n} I(|X_i - x| \leq h),$$

  where only data points within h of x are used, and counted with equal weight.

- Drawback of histogram: density estimation is piecewise constant and thus unsmooth.

- **Main idea of kernel density estimation:** instead of counting number of observations within h of x, we take a certain weighted average of data points near x to estimate $f(x)$.

- Usually we assign higher weights to data points closer to x, and lower weights to those further away from x.

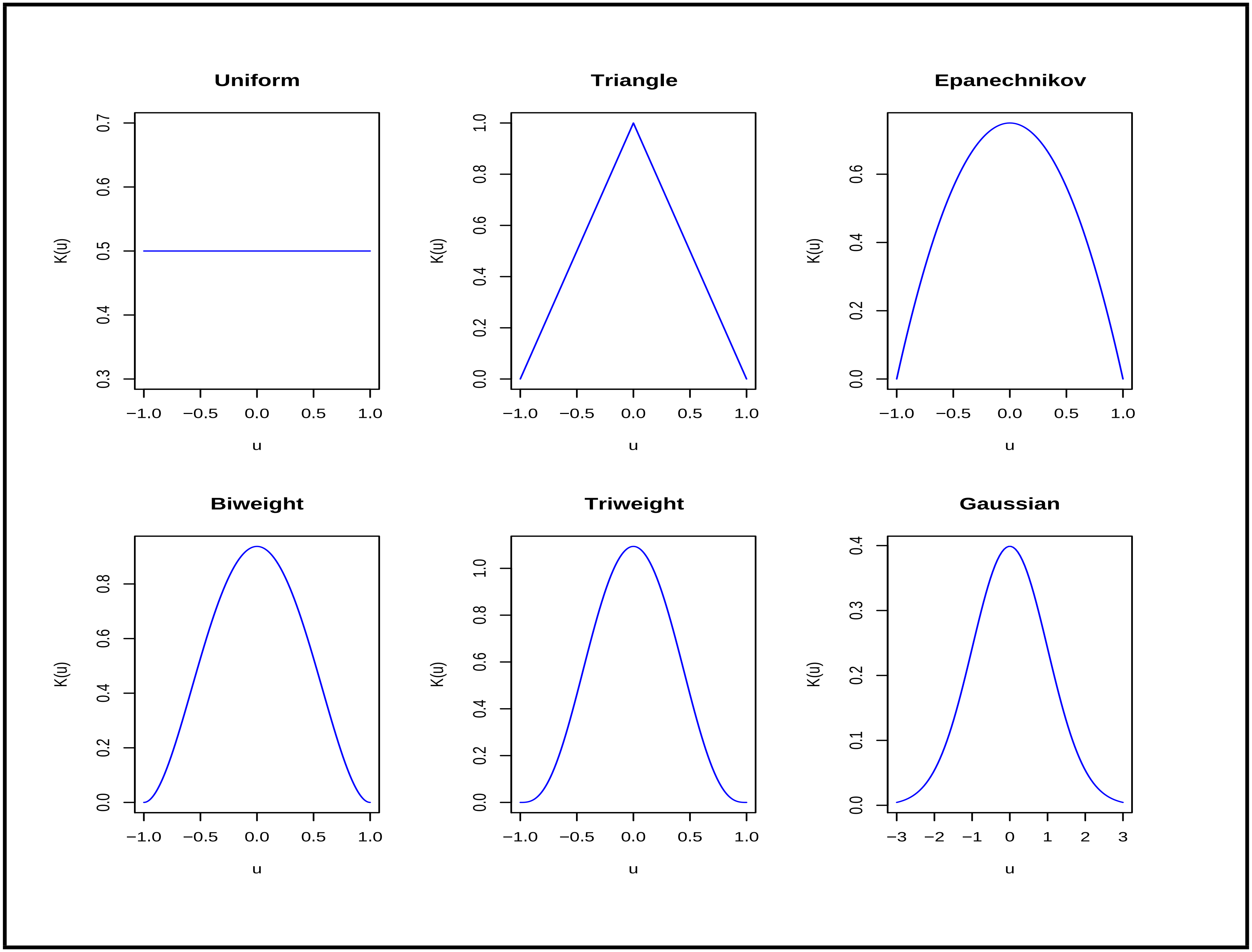
- Kernel density estimate

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

where $K(\cdot)$ is a kernel function.

Choices of kernel function

- Uniform: $K(u) = 1/2I(u \leq 1)$. Reduces to histogram
- Gaussian: $K(u) = \phi(u)$ (standard normal density)
- Triangle: $K(u) = (1 - |u|)I(|u| \leq 1)$
- Epanechnikov: $K(u) = 3/4(1 - u^2)I(|u| \leq 1)$
- Biweight: $K(u) = \frac{15}{16}(1 - u^2)^2 I(|u| \leq 1)$
- Triweight: $K(u) = \frac{35}{32}(1 - u^2)^3 I(|u| \leq 1)$
- minimum variance kernel (allowing negative weight):
  $K(u) = 1/8(3 - 5u^2)I(|u| \leq 1)$



Uniform
Triangle
Epanechnikov
Biweight
Triweight
Gaussian
K(u)
u

Choice of h:

- A rule of thumb (Hardle, 1991):

$$h = \frac{1.06}{n^{1/5}} S,$$

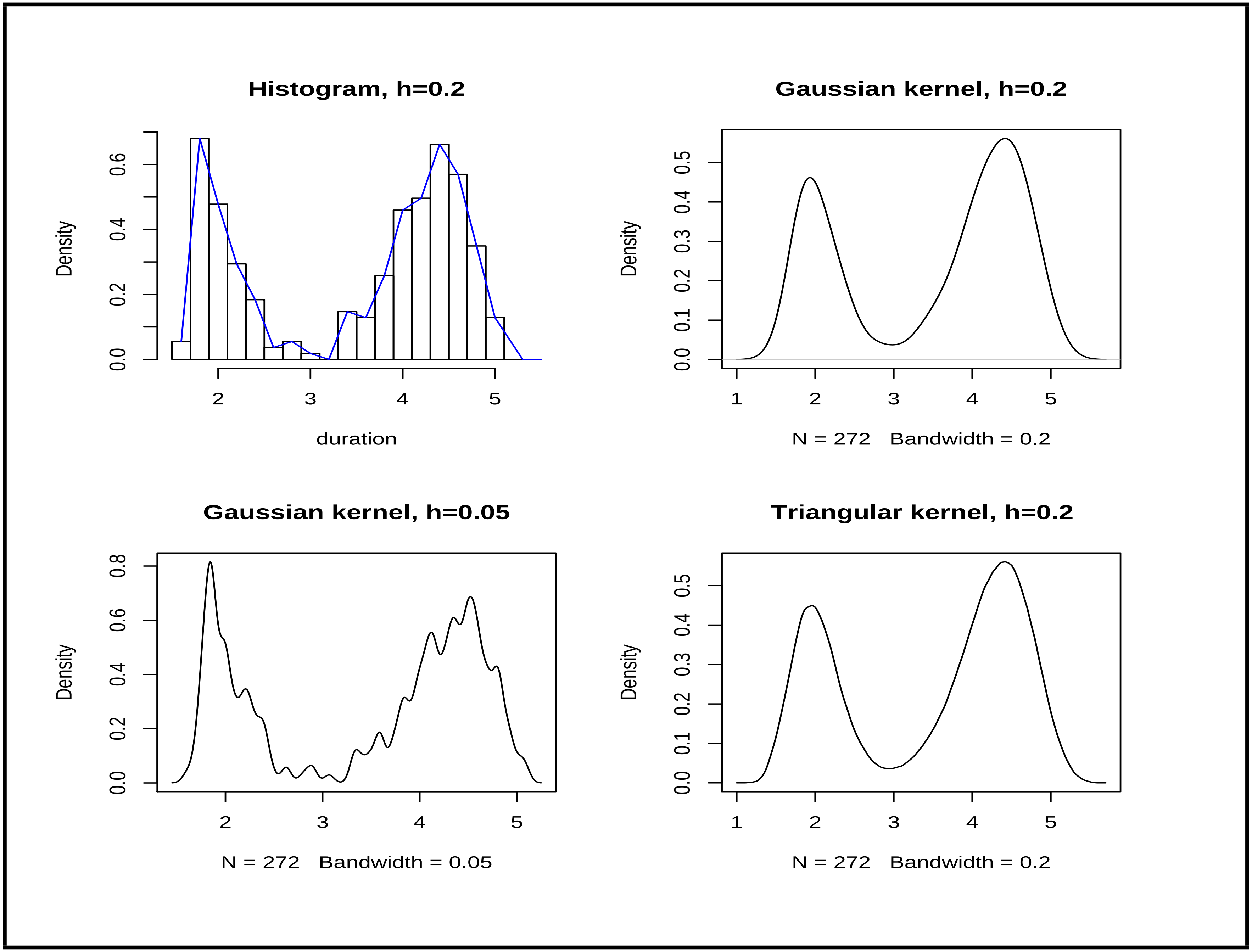
where S is the same standard deviation of the sample, or can also be taken as more robust version of standard deviation, e.g.

  - the scaled interquartile range $(X_{(0.75)} - X_{(0.25)})/1.34$;
  - or scaled MAD (median absolute deviation): $1.4826 Median\{|X_i - median(X_i)|\}$.

Example 10.1.1 *data set faithful contains 272 durations (mins) of the eruptions of Old Faithful geyer.*

```
#take a look at the data set (the first 5 observations)
faithful[1:5,]

duration = faithful$eruptions
```

```
n=length(duration)

#estimate the density using histogram and kernel estimation
par(mfrow=c(2,2))
#histogram
out=hist(duration, nclass=15, prob=TRUE, main="Histogram, h=0.2")
z = (out$breaks[-1] + out$breaks[-19])/2
lines(out$density~z, col="blue")

#kernel density estimation
d1 = density(duration, kernel="gaussian", bw=0.2)
plot(d1, main="Gaussian kernel, h=0.2")
d2=density(duration, kernel="gaussian", bw=0.05)
plot(d2, main="Gaussian kernel, h=0.05")
d3=density(duration, kernel="triangular", bw=0.2)
plot(d3,main="Triangular kernel, h=0.2")
```



Histogram, h=0.2
Density
0.0
0.2
0.4
0.6
2
3
4
5
duration
Gaussian kernel, h=0.2
Density
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
N = 272 Bandwidth = 0.2
Gaussian kernel, h=0.05
Density
0.0
0.2
0.4
0.6
0.8
2
3
4
5
N = 272 Bandwidth = 0.05
Triangular kernel, h=0.2
Density
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
N = 272 Bandwidth = 0.2

10.2 Nonparametric Curve Smothing

Given n pairs of data (x_i, y_i), $i = 1, \cdots, n$, we want to investigate the relationship between variables Y (response) and X (predictor).

10.2.1 Parametric regression

Regression model:

$$y_i = f(x_i) + e_i,$$

where $f(\cdot)$: unknown function, usually referred to as the regression function or curve.

Parametric Models

Assume that the form of f is known except for finitely many unknown parameters.

- Assume there is a unknown parameter vector $\beta = (\beta_1, \cdots, \beta_p)^T$ and a known function $f(\cdot, \beta)$.

- The form of f determines linear/nonlinear models
  - Linear model: $f(x) = \beta_1 + \beta_2 x$
  - Quadratic model: $f(x) = \beta_1 + \beta_2 x + \beta_3 x^2$
  - Nonlinear model: $f(x) = \beta_1 x^{\beta_2}$
- β is p-dimensional, p is finite.
- Least squares estimates (e.g. quadratic model):

$$(\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3) = \arg\min \sum_{i=1}^{n} (y_i - \beta_1 - \beta_2 x_i - \beta_3 x_i^2)^2,$$

and the fitted regression line

$$\hat{f}(x) = \hat{\beta}_1 + \hat{\beta}_2 x + \hat{\beta}_3 x^2.$$

Example: Motorcycle Data

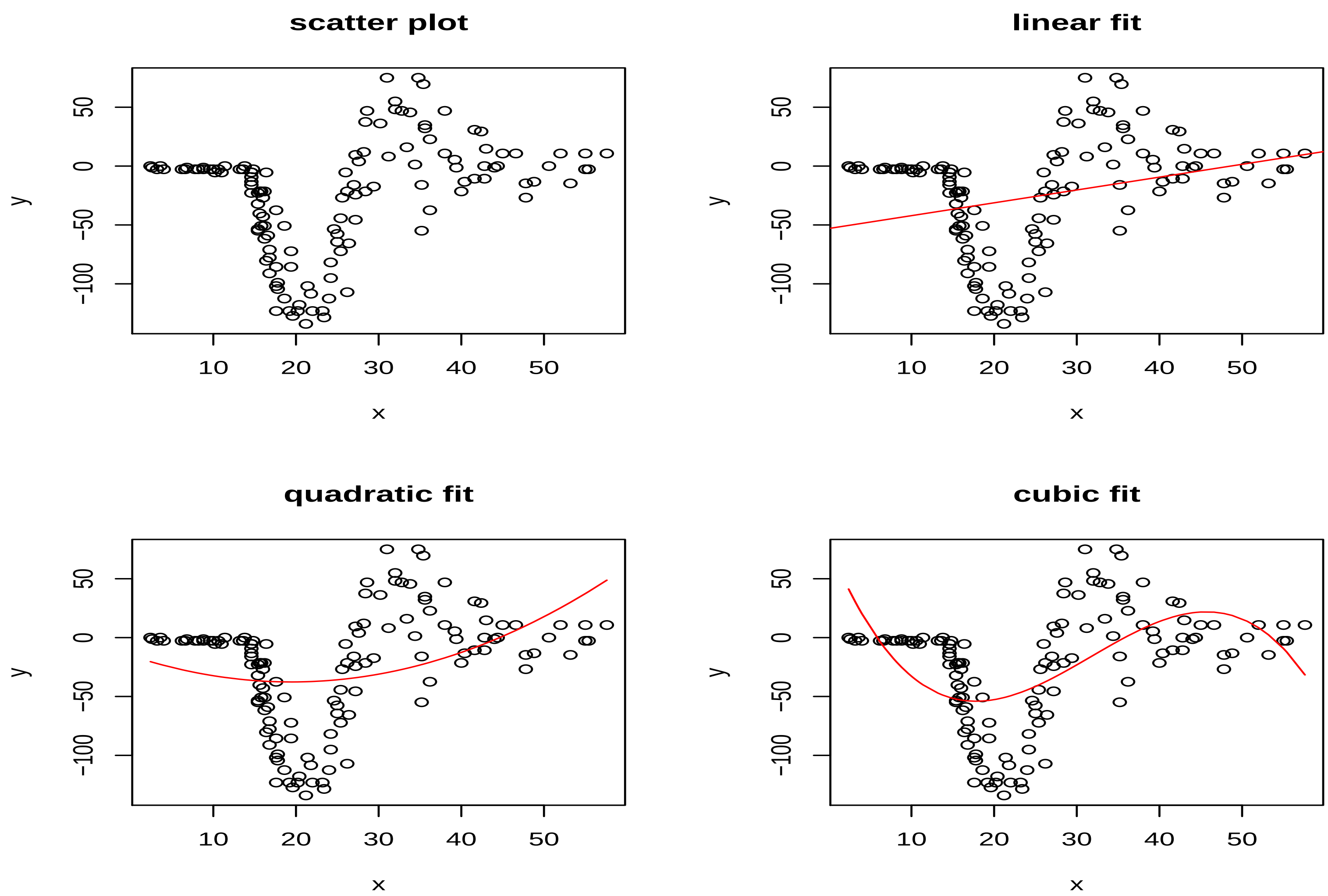
- n = 133
- x = time, y = acceleration
- Silverman (1985): Density estimation for Statistics and Data Analysis

Polynomial Fit (Least Squares Regression)

```
library(MASS)
data(mcycle)
x = mcycle[,1]
y = mcycle[,2]

par(mfrow=c(2,2))
plot(y~x, main="scatter plot")

#linear regression
plot(y~x, main="linear fit")
fit1 = lm(y~x)$fitted
lines(fit1~x, col=2)
```

```
#quadratic regression
plot(y~x, main="quadratic fit")
x2 = x^2
fit2 = lm(y~x+x2)$fitted
lines(fit2~x, col=2)

#cubic regression
x3=x^3
plot(y~x, main="cubic fit")
fit3 = lm(y~x+x2+x3)$fitted
lines(fit3~x, col=2)
```

Polynomial Regression Fit



Nonparametric Models

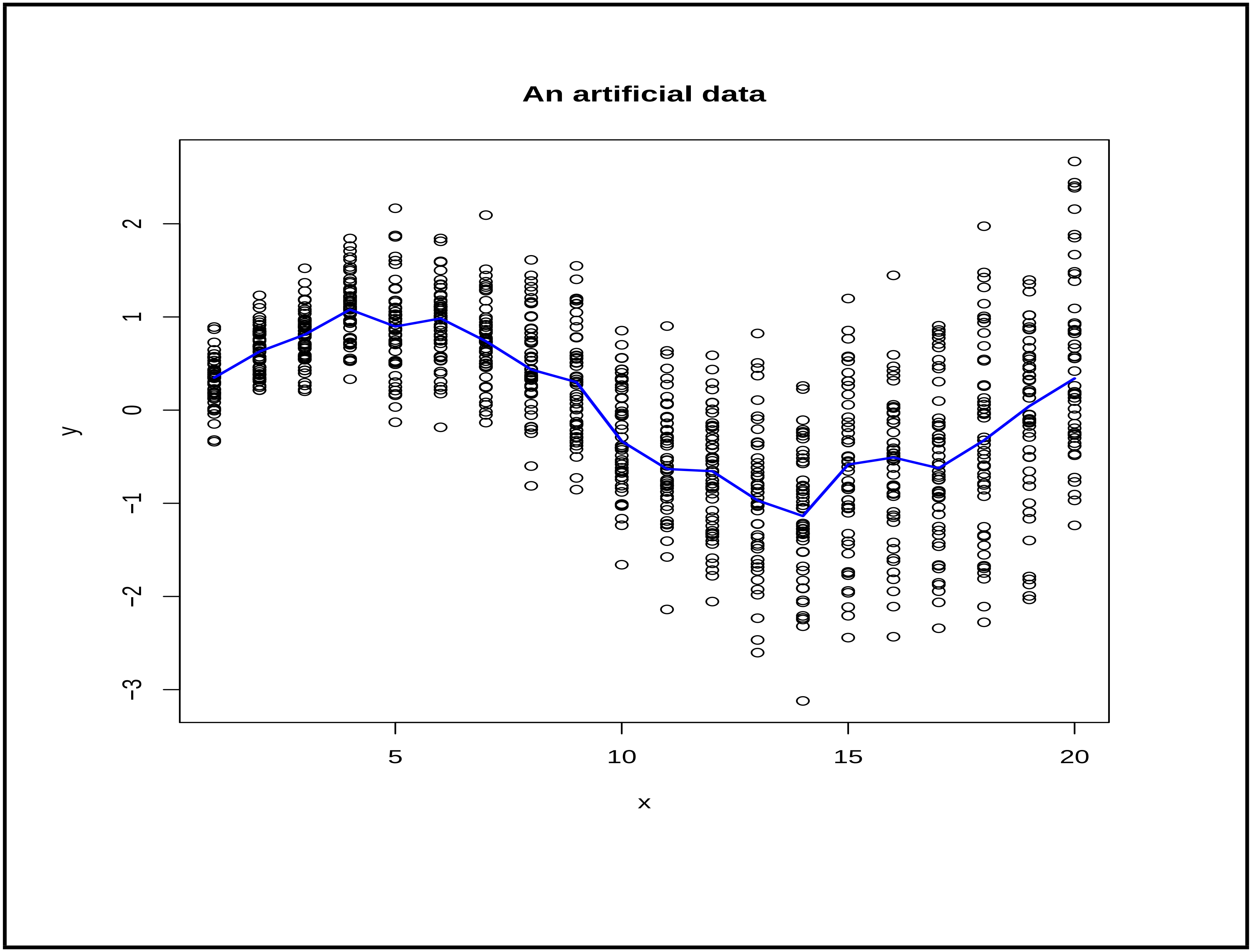
- Motivation: the underlying regression function is so complicated that no reasonable parametric model would be adequate
- Do not assume any specific form of f. More flexible.
- Infinite dimensional parameters: not no parameters
- For illustration, we shall focus on the special case of nonparametric function with $p = 1$, i.e., one predictor such as time.

10.2.2 Scatterplot smooth: categorical predictors

- Assume $x \in \{1, 2, \cdots, c\}$ are discrete, and we obtain a fairly large number of y at each possible value of x.

- Question: how to smooth y in this simplest situation?

- One solution: take the mean/median of y in each category


An artificial data
y
x
2
1
0
-1
-2
-3
5
10
15
20

10.2.3 Local regression

- Continuous predictor: no enough replicates at each x value
- Goal: estimate y at $x = x_0$, i.e. $f(x_0)$.
- Local averaging (running mean): mimic the categorical averaging. In order to estimate $f(x_0)$: we take the average/median of the y_i's corresponding to those x_i's that are close to x_0
  - **Nearest neighbor smoother**: $x_{(1)} \leq \cdots \leq x_{(n)}$. To estimate $f(x_{(j)})$, find the k values of x_i's that are nearest to x_0. Then take $\hat{f}(x_0)$ as the average of y_i's corresponding to these k x's. That is, if we let $N_k(x_0)$ denote this set of k points, then

$$\hat{f}(x_0) = \frac{1}{k} \sum_{i:x_i \in N_k(x_0)} y_i$$

- k: window width. Smaller k: less smooth, less bias, more variance.

- **Running line/Loess regression**: within each window, fit a linear regression function $l(x) = \beta_1 + \beta_2(x - x_0)$. Then estimate $f(x_0)$ by $\hat{\beta}_1$, the least squares estimate of β_1 obtained by using paired data from each window, i.e. data points such that $x_i \in N_k(x_0)$.

- Some generalization. Let $W(u), 0 \leq u \leq 1$ be a nonnegative weighting function with mode at $u = 0$, for example,
  - $W(u) = (1 - u^3)^3$: weighting points closer to x_0 more;
  - $W(u) = 1$: unweighted, the same as running line.
  - R function loess().

We estimate $l(x) = \beta_1 + \beta_2(x - x_0)$ by minimizing

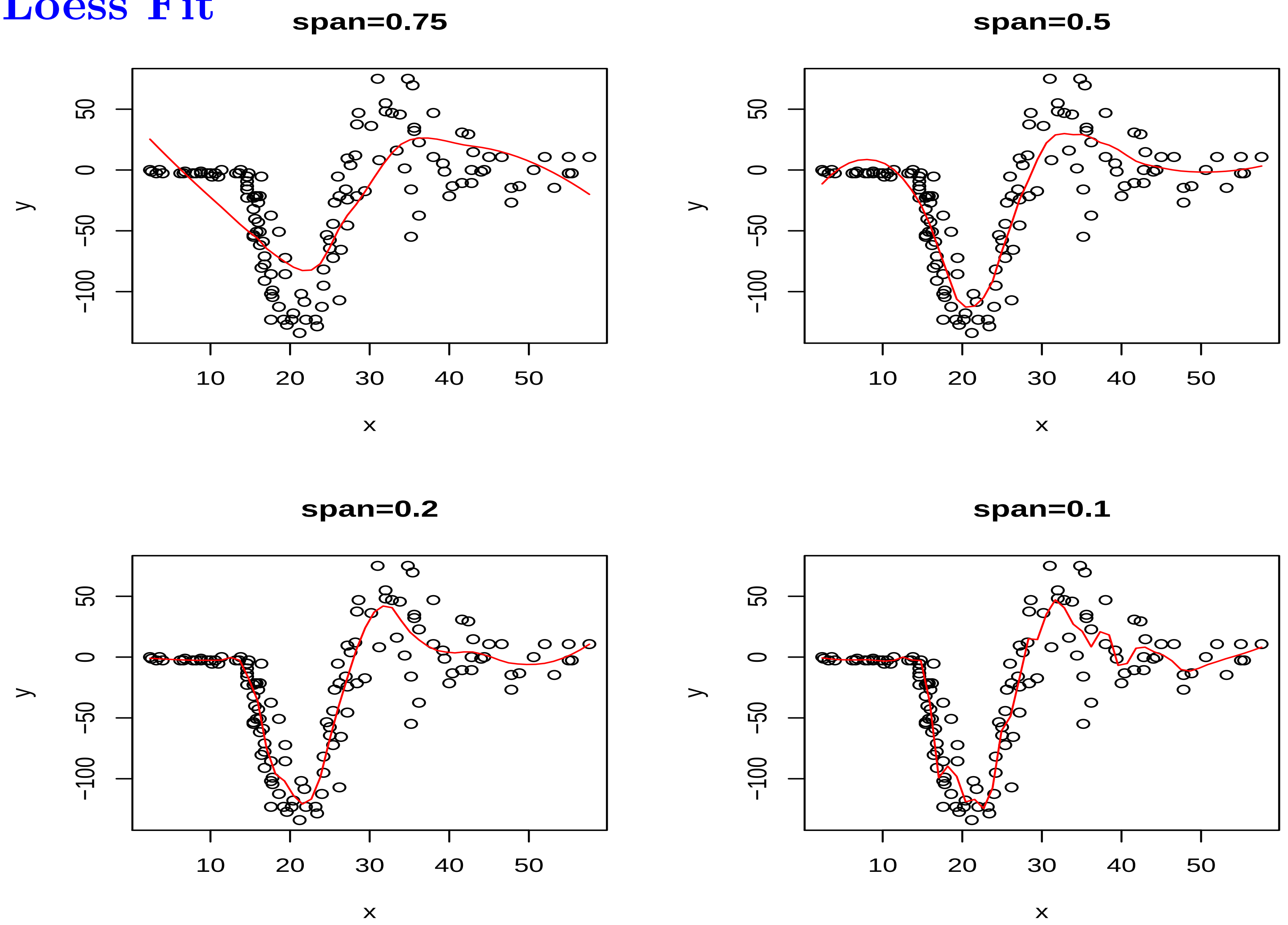
$$\sum_{i:x_i \in N_k(x_0)} \{y_i - l(x_i)\}^2 \, W\left\{\frac{|x_0 - x_i|}{\Delta_{x_0}}\right\},$$

where $\Delta_{x_0} = \max_{x_i \in N_k(x_0)} |x_i - x_0|$ is the maximum distance of x_0 to elements in $N_k(x_0)$.

```
library(MASS)
data(mcycle)
x = mcycle[,1]
y = mcycle[,2]
par(mfrow=c(2,2))
lo <- loess(y ~ x, span = 0.75)
#span controls the degree of smoothing
#neighbourhood includes proportion span of the points
#i.e. k=n*span if span<1; k=n is span>1.

plot(y~x, main="span=0.75")
newx = seq(min(x), max(x), length=50)
pred =predict(lo, data.frame(x = newx))
```

```
lines(pred ~newx, col=2)

lo <- loess(y ~ x, span = 0.5)
plot(y~x, main="span=0.5")
newx = seq(min(x), max(x), length=50)
pred =predict(lo, data.frame(x = newx))
lines(pred ~newx, col=2)

lo <- loess(y ~ x, span = 0.2)
plot(y~x, main="span=0.2")
newx = seq(min(x), max(x), length=50)
pred =predict(lo, data.frame(x = newx))
lines(pred ~newx, col=2)

lo <- loess(y ~ x, span = 0.1)
plot(y~x, main="span=0.1")
newx = seq(min(x), max(x), length=50)
pred =predict(lo, data.frame(x = newx))
lines(pred ~newx, col=2)
```

Loess Fit



loess smoothers

- observations within the same neighborhood receives nonzero weights.
- the weights drops off abruptly to zero outside the NN of x_0. This accounts for jagged appearance of the fit.
- remedy: give smooth weights. Assign weights to observations close to x_0, and let weights smoothly decrease as we move further away from x_0

10.2.4 Kernel regression

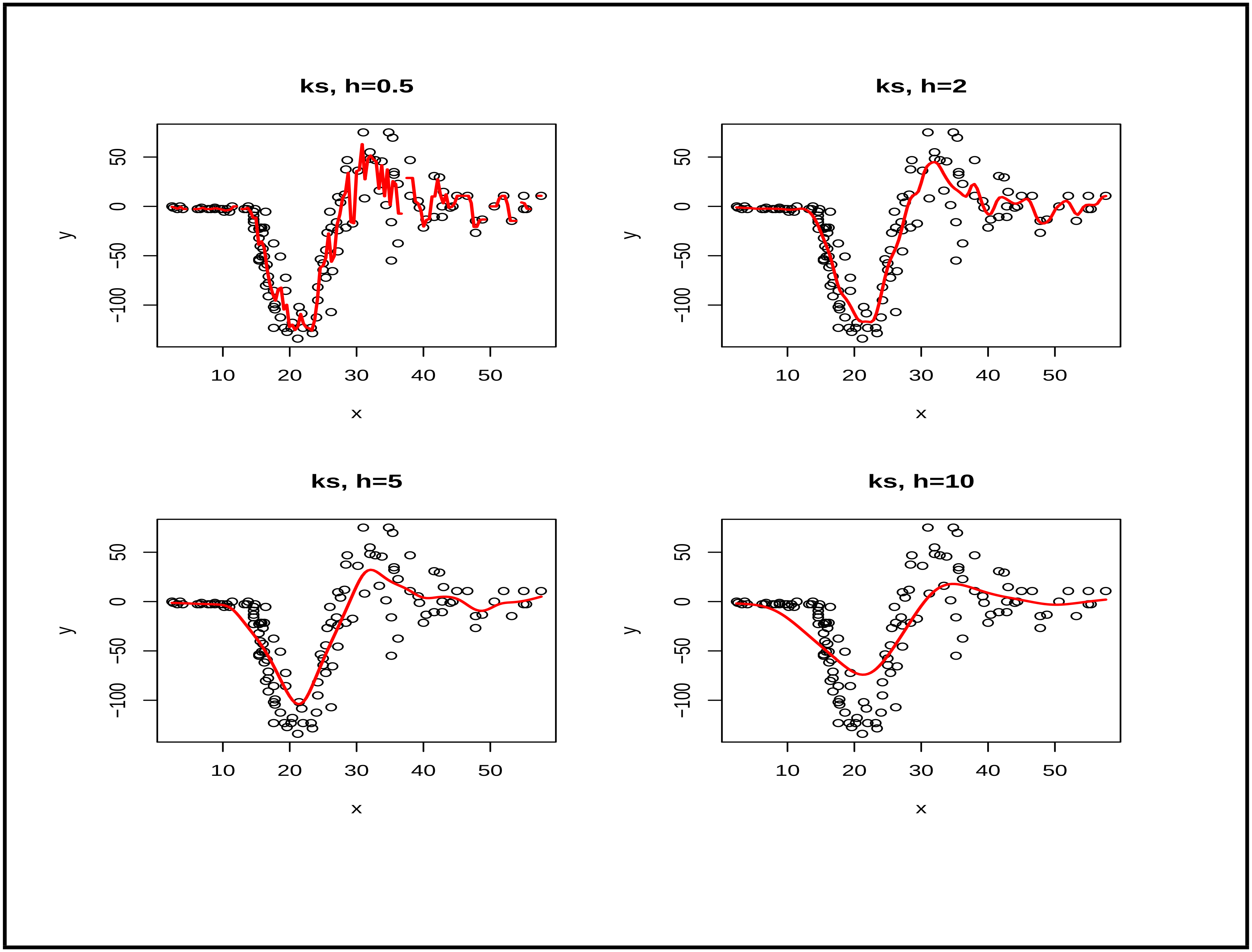
- Nadaraya (1964) and Watson (1964)
- Estimate $f(x)$ as a locally weighted average, using a kernel as a weighting function.
- The Nadaraya-Watson estimator is:

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

  where
  - $h > 0$ is the bandwidth parameter, plays the same role as k in loess,
  - $K(\cdot)$ is a kernel function; points closer to x receive larger weights.
- Does not specify the number of points in the neighborhood, but specify the width of the neighborhood.

- R function ksmooth()

```
#analyze the motocycle data
par(mfrow=c(2,2))
plot(y~x, main="ks, h=0.5")
lines(ksmooth(x, y, "normal", bandwidth=0.5), col=2, lwd=2)

plot(y~x, main="ks, h=2")
lines(ksmooth(x, y, "normal", bandwidth=2), col=2, lwd=2)

plot(y~x, main="ks, h=5")
lines(ksmooth(x, y, "normal", bandwidth=5), col=2, lwd=2)

plot(y~x, main="ks, h=10")
lines(ksmooth(x, y, "normal", bandwidth=10), col=2, lwd=2)
```



ks, h=0.5
ks, h=2
ks, h=5
ks, h=10
y
x
50
0
−50
−100
10
20
30
40
50

10.3 Robust Regression—M-Estimation

Assume the regression model:

$$Y_i = f(X_i) + \epsilon_i, i = 1, \cdots, n.$$

For instance, $f(X_i) = \beta_0 + \beta_1 X_i$ for linear regression model.

- Least squares regression estimates $f(x_i)$ by

  $$\min \sum_{i=1}^{n} \{y_i - f(x_i)\}^2 .$$

  Drawback: sensitive to outliers in Y.

- Median regression (least absolute deviation regression) estimates $f(x_i)$ by

  $$\min \sum_{i=1}^{n} |y_i - f(x_i)|.$$

  R function: rq() in package quantreg with quantile level tau=0.5.

- A more general procedure—M estimation, estimates $f(x_i)$ by

$$\min \sum_{i=1}^{n} \rho \left\{ \frac{y_i - f(x_i)}{\hat{\sigma}_i} \right\},$$

  - $\hat{\sigma}_i$ is an estimate of the standard deviation of ϵ_i;
  - $\rho(x)$ is a symmetric function with a unique minimum at $x = 0$;
  - Tukey bisquare function:

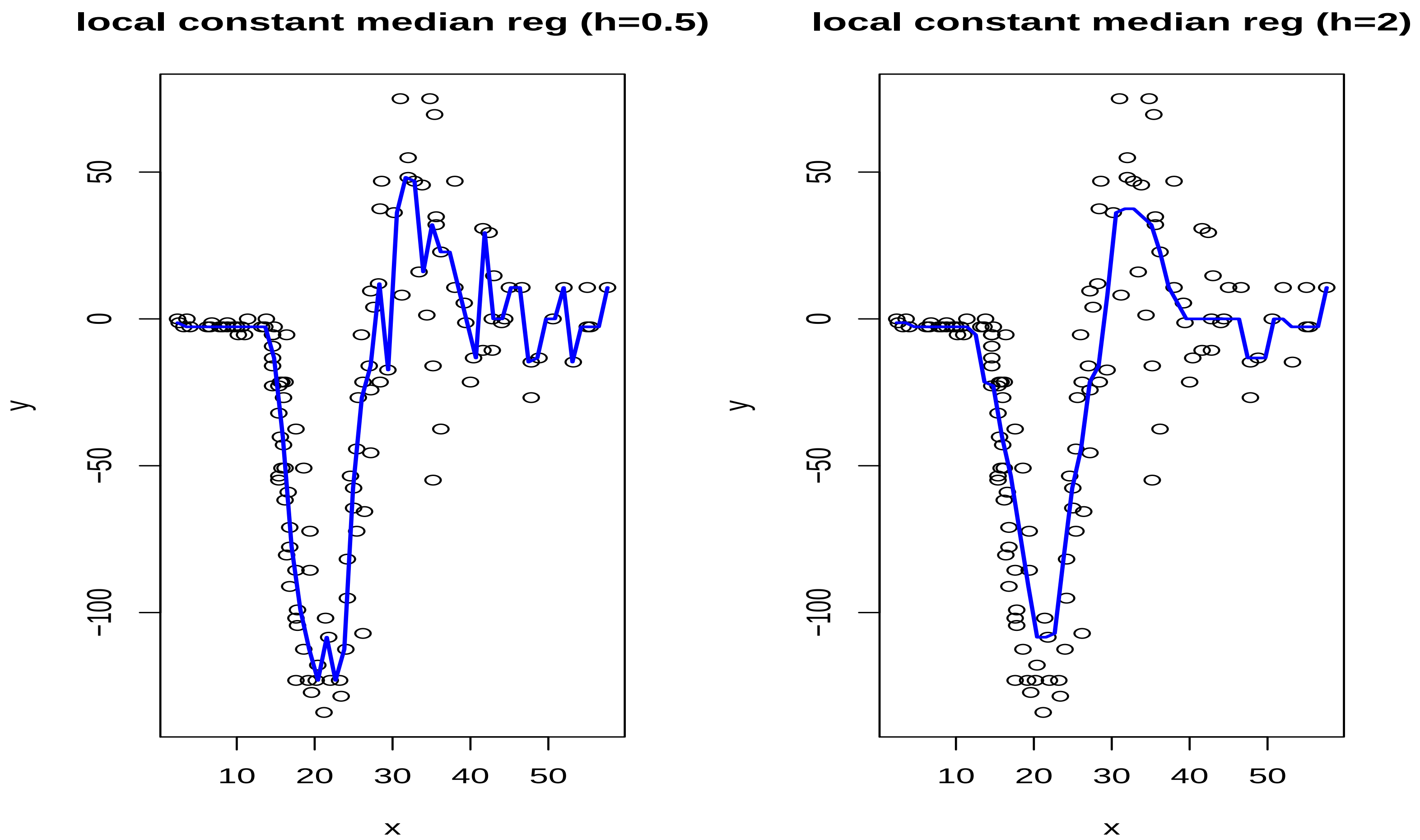
$$\begin{aligned} \rho(x) &= \left(\frac{x}{c}\right)^6 - 3\left(\frac{x}{c}\right)^4 + 3\left(\frac{x}{c}\right)^2, \quad |x| \le c \\ &= 1, \quad |x| > c. \end{aligned}$$

  - R function for robust linear regression: rlm() in package MASS.
  - Options of objective function: psi= psi.huber, psi.hampel or psi.bisquare.
- Local robust regression with Tukey bisquare objective function: R function loess() with option: family="symmetric".

Local constant median regression

```
lcrq <- function(x, y, h, m = 50, tau = 0.5) {
 xx <- seq(min(x), max(x), length = m)
 fv <- xx  # estimates of f
 for (i in 1:length(xx)) {
 z <- x - xx[i]
 wx <- dnorm(z/h)
 r <- rq(y ~1, weights = wx, tau = tau, ci = FALSE)
 fv[i] <- r$coef[1]
 }
 return(list(fv=fv, xx=xx))
}
par(mfrow=c(1,2))
plot(y~x, main="local constant median reg (h=0.5)")
fit1 = lcrq(x, y, h=0.5, m=50, tau=0.5)
lines(fit1$fv~fit1$xx, col="blue", lwd=2)

plot(y~x, main="local constant median reg (h=2)")
fit1 = lcrq(x, y, h=2, m=50, tau=0.5)
lines(fit1$fv~fit1$xx, col="blue", lwd=2)
```

Local constant median

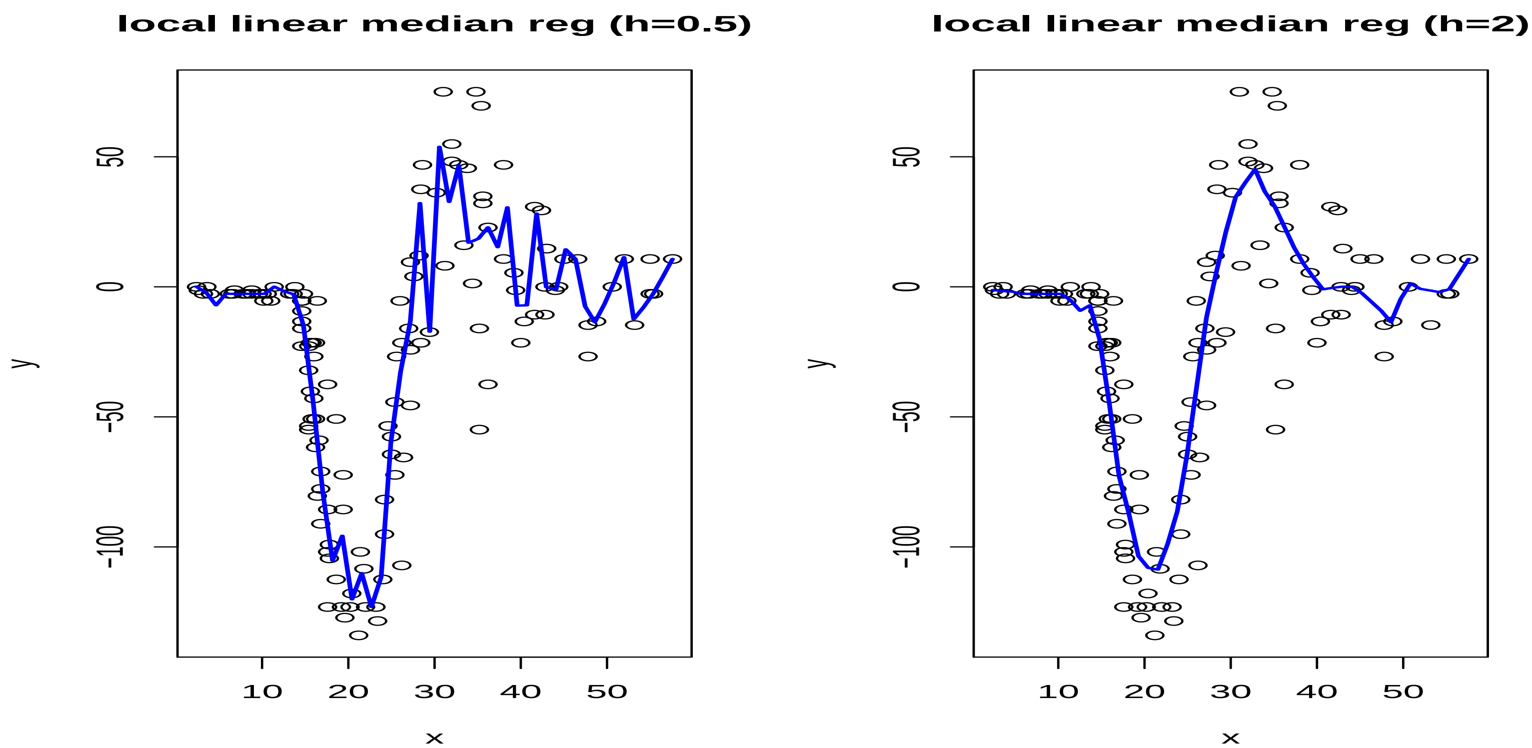


Local linear median regression

```
llrq <- function(x, y, h, m = 50, tau = 0.5) {
 xx <- seq(min(x), max(x), length = m)
 fv <- xx  # estimates of f
 dv <- xx  # estimate of derivative of f
 for (i in 1:length(xx)) {
 z <- x - xx[i]
 wx <- dnorm(z/h)
 r <- rq(y ~ z, weights = wx, tau = tau, ci = FALSE)
 fv[i] <- r$coef[1]
 dv[i] <- r$coef[2]
 }
 return(list(fv=fv, dv=dv, xx=xx))
}
par(mfrow=c(1,2))
plot(y~x, main="local linear median reg (h=0.5)")
fit1 = llrq(x, y, h=0.5, m=50, tau=0.5)
lines(fit1$fv~fit1$xx, col="blue", lwd=2)

plot(y~x, main="local linear median reg (h=2)")
```

```
fit1 = llrq(x, y, h=2, m=50, tau=0.5)
lines(fit1$fv~fit1$xx, col="blue", lwd=2)
```

Local linear median regression



How to choose the smoothing parameter h

- smaller h: rougher estimates, relying heavily on the data near x, smaller bias, larger variance

- larger h: more averaging range, smoother estimates, larger bias, smaller variance

Bandwidth Selection (m-fold cross validation)

- Randomly divide the data into m non-overlapped and roughly equal-sized parts $D_1, \cdots, D_m$.

- For the ith part, fit the model using the data from the test data, predict and calculate the prediction error as

$$\sum_{j \in D_i} \rho \left\{ Y_j - \hat{f}(x_j)_{-D_i} \right\}.$$

- Repeat this procedure for $i = 1, \cdots, m$, and calculate the averaged prediction error.

- Select h with the smallest averaged prediction error.