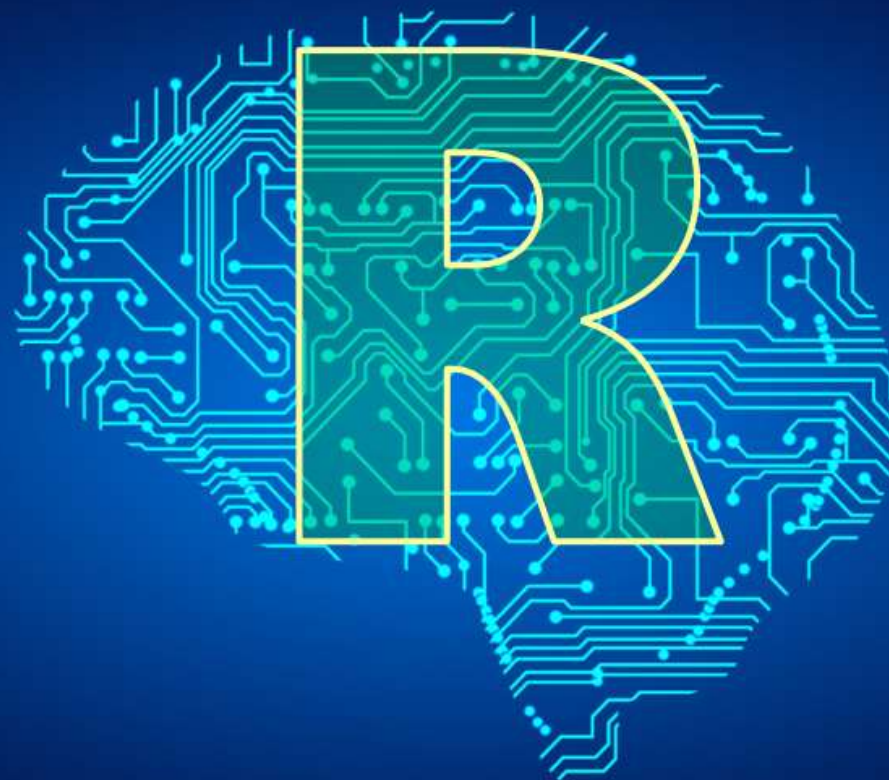




平滑技巧

吳漢銘

國立臺北大學 統計學系



■ 主題1

- 移動平均 (Moving Average)
- 曲線配適 (Fitting Curves): lowess
- 核密度估計 (Kernel Density Estimation)
- 三次樣條插值 (Cubic Spline Interpolation)

Simple Moving Average

- A moving average (移動平均) (簡稱均線) is a calculation to analyze data points by creating series of **averages** of **different subsets** of the full data set.

$$SMA = \frac{p_1 + p_2 + \cdots + p_n}{n}$$

When price is in an uptrend and subsequently, the moving average is in an uptrend, and the moving average has been tested by price and price has bounced off the moving average a few times (i.e. the moving average is serving as a support line), then a trader might **buy** on the **next pullbacks back** to the Simple Moving Average.

Moving Average Acting as Support - Potential Buy Signal



www.OnlineTradingConcepts.com - All Rights Reserved

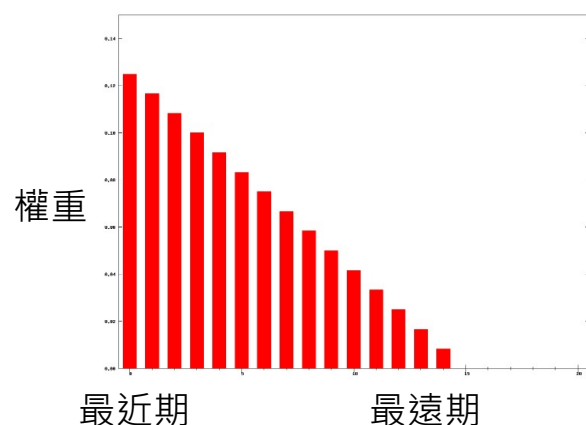
Created with TradeStation

<http://www.onlinetradingconcepts.com/TechnicalAnalysis/MASimple.html>

Moving Average Acting as Resistance - Potential Sell Signal

4/16

- At times when price is in a **downtrend** and the moving average is in a downtrend as well, and price tests the SMA above and is rejected a few consecutive times (i.e. the moving average is serving as a resistance line), then a trader might sell on the **next rally up** to the Simple Moving Average.



An n-day WMA (Weighted moving average)

$$WMA_M = \frac{np_M + (n-1)p_{M-1} + \cdots + 2p_{(M-n+2)} + p_{(M-n+1)}}{n + (n-1) + \cdots + 2 + 1}$$

www.OnlineTradingConcepts.com - All Rights Reserved

<http://www.onlinetradingconcepts.com/TechnicalAnalysis/MASimple.html>

Smoothing in R

smooth: Forecasting Using Smoothing Functions

<https://cran.r-project.org/web/packages/smooth/index.html>

es() - Exponential Smoothing;
ssarima() - State-Space ARIMA, also known as Several Seasonalities ARIMA;
ces() - Complex Exponential Smoothing;
ges() - Generalised Exponential Smoothing;
ves() - Vector Exponential Smoothing;
sma() - Simple Moving Average in state-space form;

TTR: Technical Trading Rules

<https://cran.r-project.org/web/packages/TTR/index.html>

SMA(x, n = 10, ...)
EMA(x, n = 10, wilder = FALSE, ratio = NULL, ...)
DEMA(x, n = 10, v = 1, wilder = FALSE, ratio = NULL)
WMA(x, n = 10, wts = 1:n, ...)
EVWMA(price, volume, n = 10, ...)
ZLEMA(x, n = 10, ratio = NULL, ...)
VWAP(price, volume, n = 10, ...)
VMA(x, w, ratio = 1, ...)
HMA(x, n = 20, ...)
ALMA(x, n = 9, offset = 0.85, sigma = 6, ...)

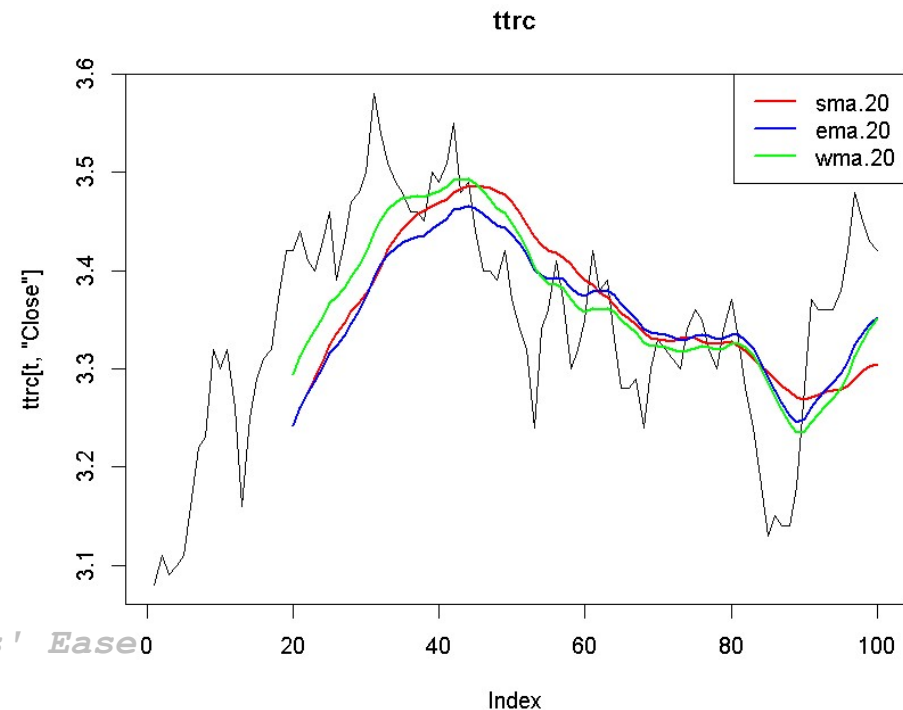
Example

ttrc {TTR}: Technical Trading Rule Composite data

Historical Open, High, Low, Close, and Volume data for the periods January 2, 1985 to December 31, 2006. Randomly generated.

```
> # install.packages("TTR")
> library(TTR)
> data(ttrc)
> dim(ttrc)
[1] 5550    6
> head(ttrc)
      Date Open High  Low Close Volume
1 1985-01-02 3.18 3.18 3.08  3.08 1870906
2 1985-01-03 3.09 3.15 3.09  3.11 3099506
3 1985-01-04 3.11 3.12 3.08  3.09 2274157
4 1985-01-07 3.09 3.12 3.07  3.10 2086758
5 1985-01-08 3.10 3.12 3.08  3.11 2166348
6 1985-01-09 3.12 3.17 3.10  3.16 3441798

> t <- 1:100
> sma.20 <- SMA(ttrc[t, "Close"], 20)
> ema.20 <- EMA(ttrc[t, "Close"], 20) # Arms' Ease
> wma.20 <- WMA(ttrc[t, "Close"], 20)
>
> plot(ttrc[t, "Close"], type="l", main="ttrc")
> lines(sma.20, col="red", lwd=2)
> lines(ema.20, col="blue", lwd=2)
> lines(wma.20, col="green", lwd=2)
> legend("topright", legend=c("sma.20", "ema.20", "wma.20"),
+       col=c("red", "blue", "green"), lty=1, lwd=2)
```

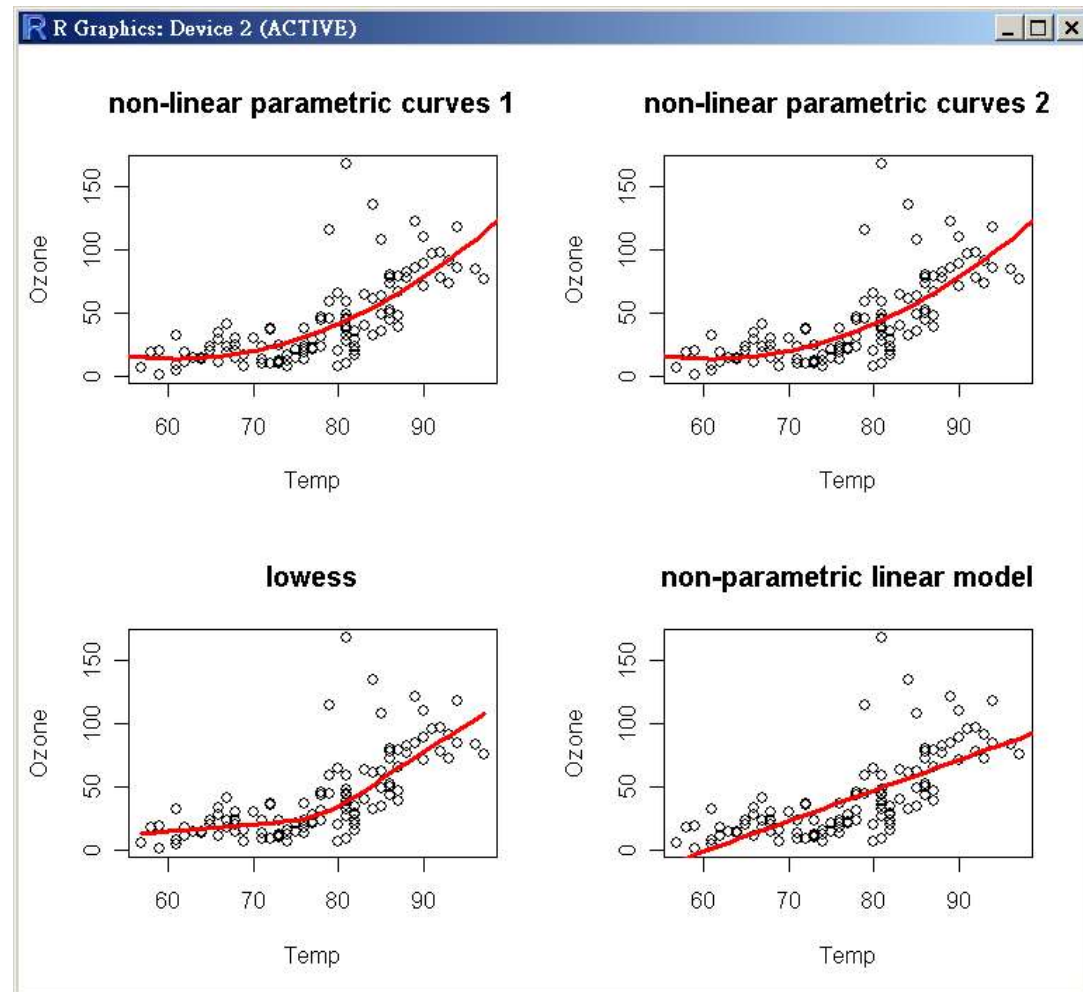


曲線配適 (Fitting Curves)

7/16

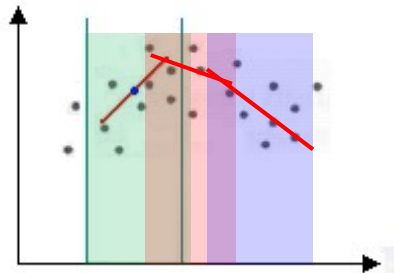
Example Methods

- non-linear parametric curves
- lowess (a non-parametric curve fitter)
- loess (a modelling tool)
- gam (fits generalized additive models)
- lm (linear model)



locally-weighted polynomial regression

Loess regression
(locally weighted polynomial regression)

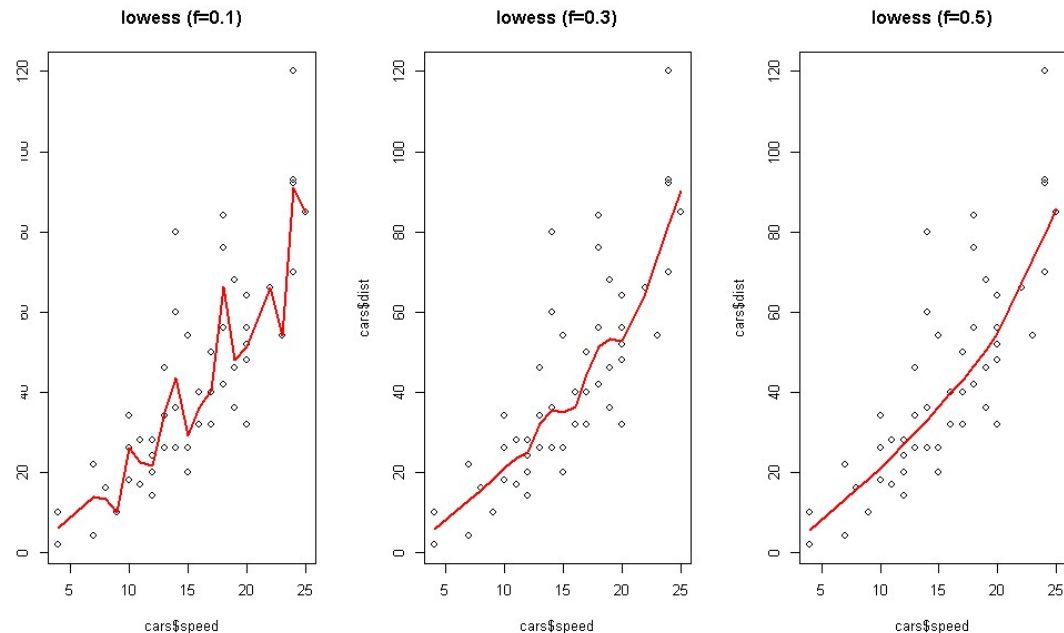


```
> data(cars)
> dim(cars)
[1] 50 2
```

```
> head(cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

```
> par(mfrow=c(1, 3))
> for(i in c(0.1, 0.3, 0.5)){
+   plot(cars$dist ~ cars$speed, main=paste0("lowess (f=", i, ")"))
+   lines(lowess(cars$dist ~ cars$speed, f = i), col="red", lwd=2)
+ }
```



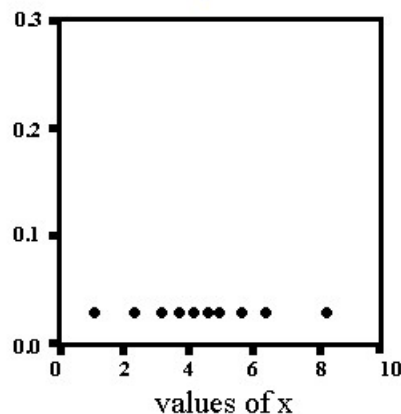
`cars {datasets}`:

The data give the speed of cars and the distances taken to stop.
Note that the data were recorded in the 1920s.

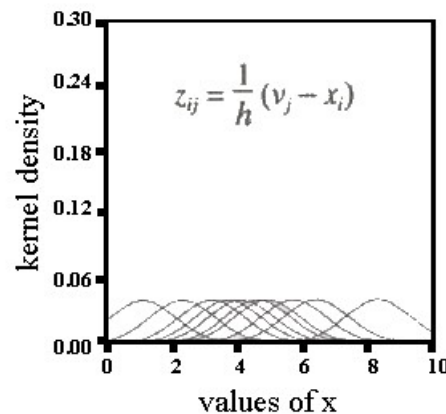
Density Plots (Smoothed Histograms) (1/3) 9/16

Constructing a Smoothed Histogram (Jacoby, 1997)

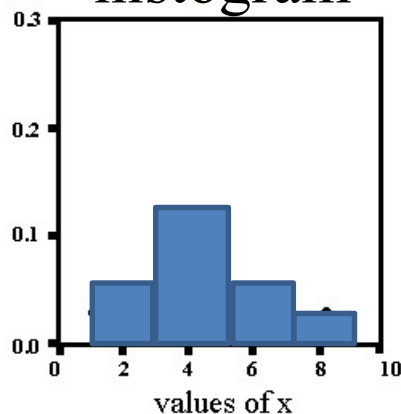
A. Unidimensional scatterplot of 10 data points



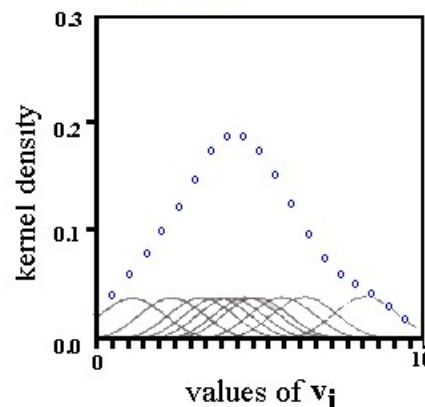
B. Data points shown as kernel densities



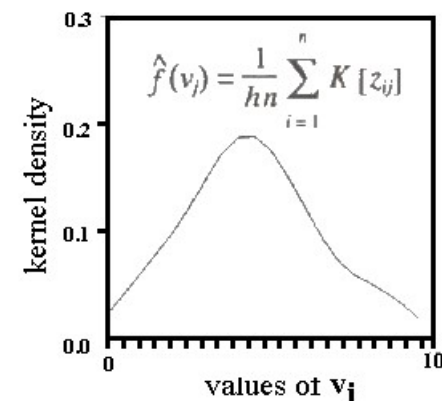
histogram



C. Summing kernel densities at the 20 v_i



D. Final smoothed histogram

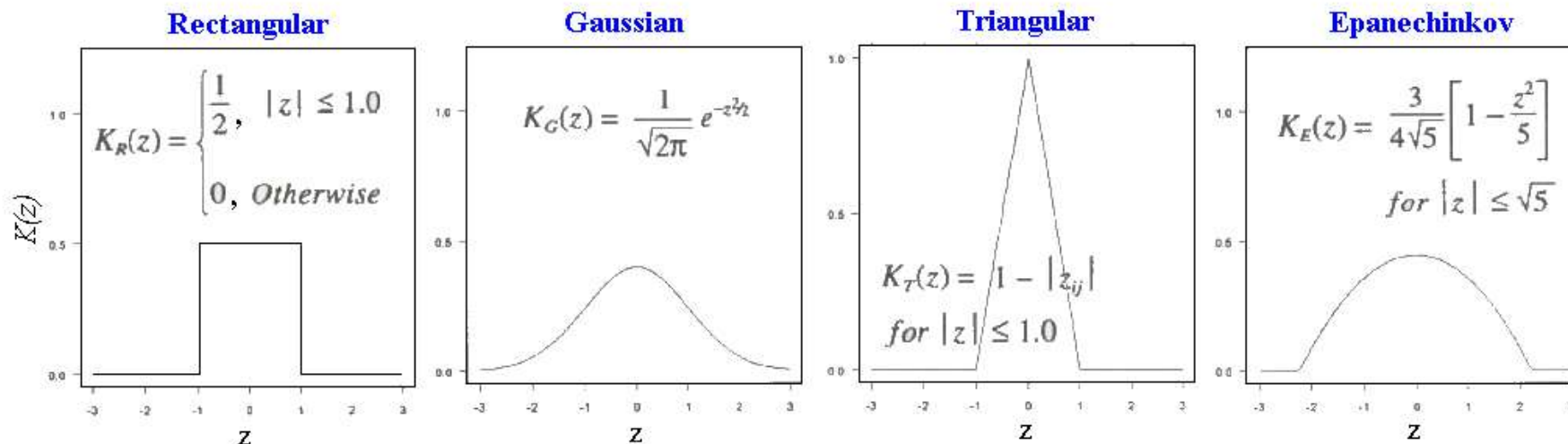


Kernel Density Estimation

10/16

- Selection of **kernels (K)**
- Selection of **bandwidth (h)**

Figures modified from Jacoby (1997)



nonparametric regression

$$y_i = f_0(x_i) + \epsilon_i, \quad i = 1, \dots, n,$$

$\epsilon_1, \dots, \epsilon_n$ are still i.i.d. random errors with $\mathbb{E}(\epsilon_i) = 0$

$$\hat{f}(v_i) = \frac{1}{hn} \sum_{j=1}^n K[z_{ij}]$$

$$z_{ij} = \frac{1}{h}(v_j - x_i)$$

kernel regression

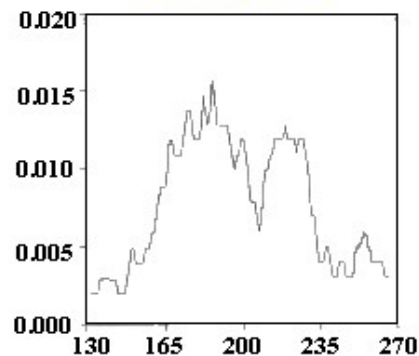
$$\hat{f}(x) = \frac{\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) y_i}{\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)}$$

Kernel Density Estimation

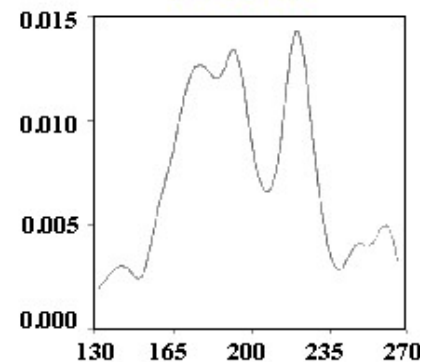
11/16

Different kernels

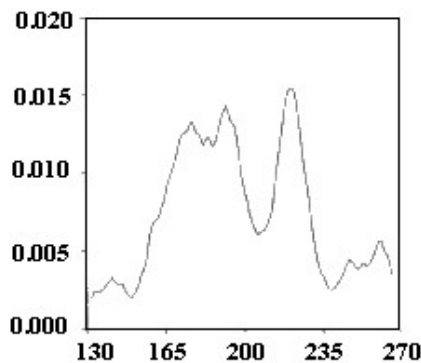
Rectangular



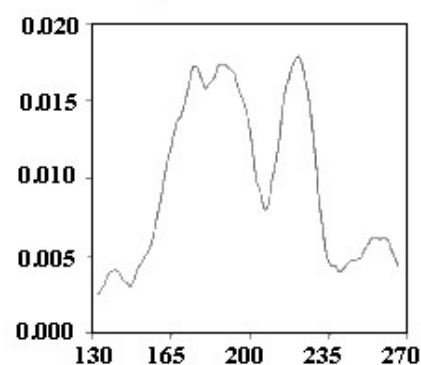
Gaussian



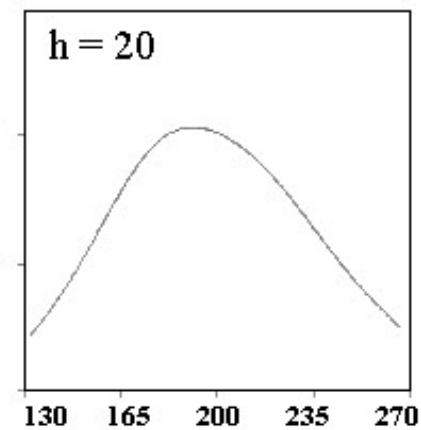
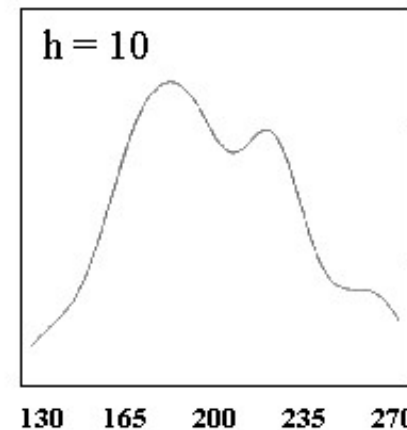
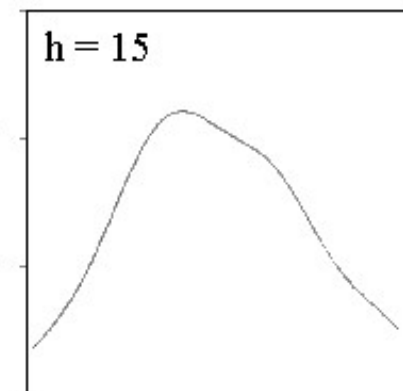
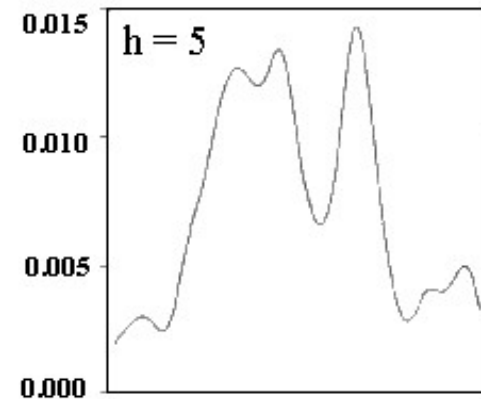
Triangular



Epanechnikov



Different bandwidth



Kernel Density Estimation in R

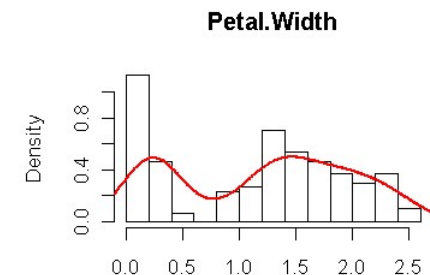
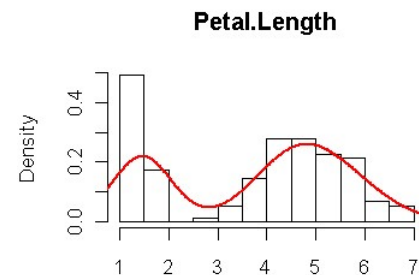
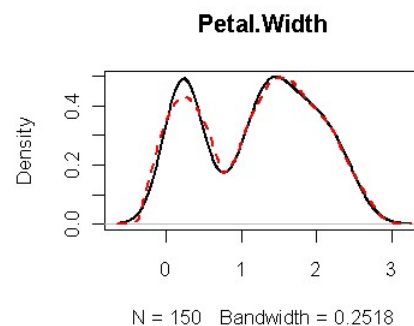
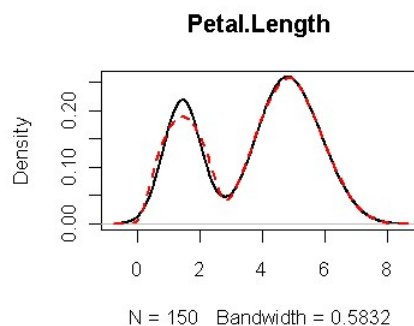
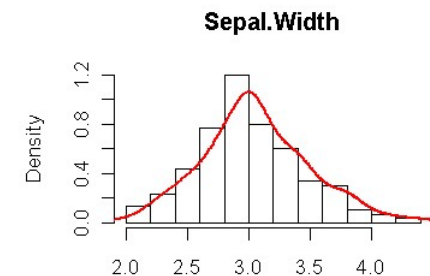
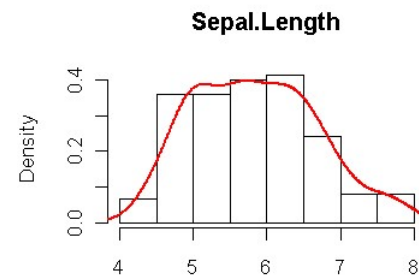
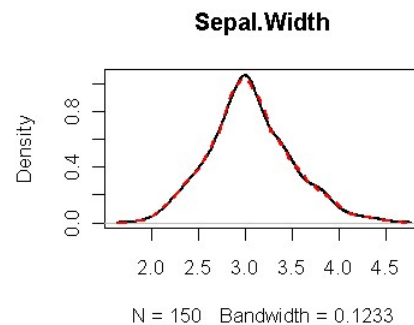
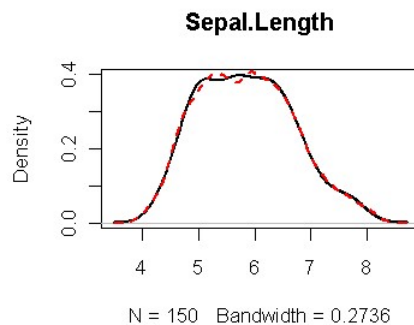
12/16

— gaussian
- - - epanechnikov

```
density(x, bw = "nrd0", adjust = 1,  
        kernel = c("gaussian", "epanechnikov", "rectangular",  
                   "triangular", "biweight",  
                   "cosine", "optcosine"),  
        weights = NULL, window = kernel, width,  
        give.Rkern = FALSE,  
        n = 512, from, to, cut = 3, na.rm = FALSE, ...)
```

```
> plot(density(iris$Sepal.Length))
```

```
> hist(iris$Sepal.Length, prob=T)  
> lines(density(iris$Sepal.Length), col="red")
```

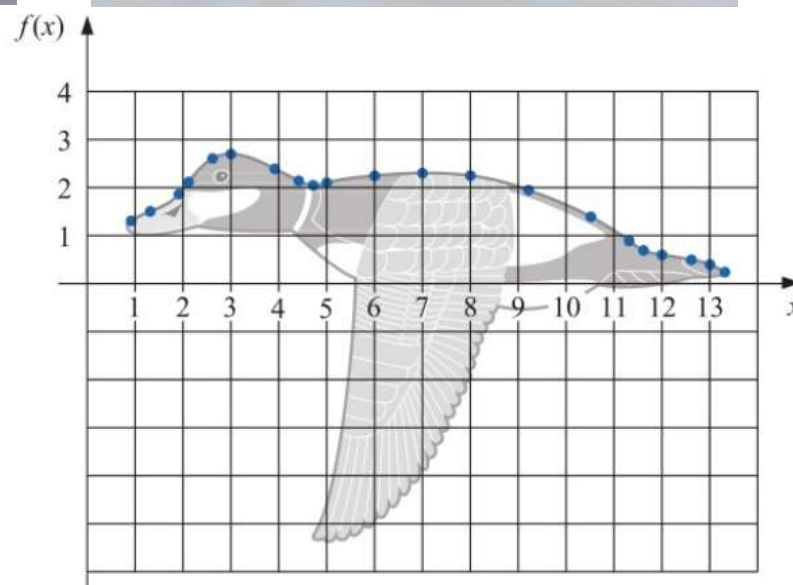
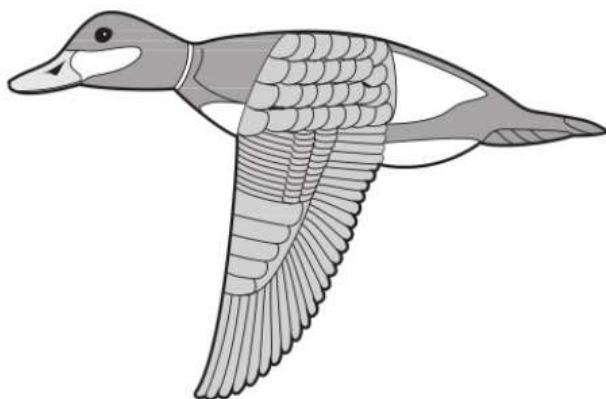
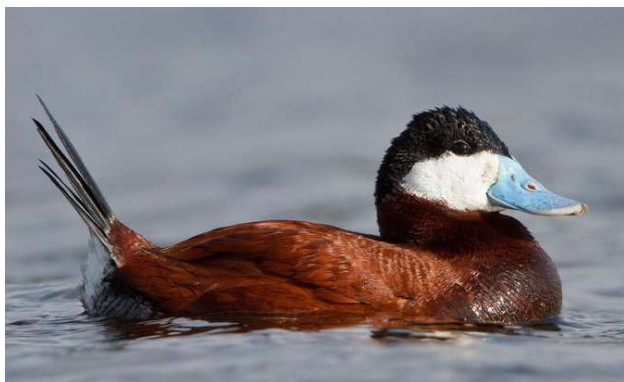


Spline approximate to the top profile of the ruddy duck

13/16

Ruddy duck (棕硬尾鴨) (雄)

"棕硬尾鴨棲息在北美洲的沼澤湖及池中，在南美洲的安地斯山脈也有分布。" <https://zh.wikipedia.org/wiki/棕硬尾鴨>



x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3
$f(x)$	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25

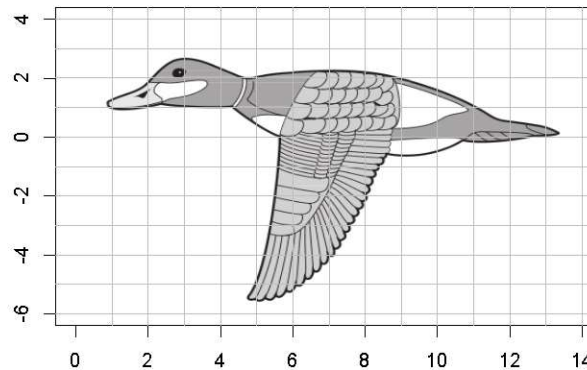
三次樣條插值法

Usage

```
smooth.spline(x, y = NULL, w = NULL, df, spar = NULL, lambda = NULL, cv = FALSE,  
              all.knots = FALSE, nknots = .nknots.smspl,  
              keep.data = TRUE, df.offset = 0, penalty = 1,  
              control.spar = list(), tol = 1e-6 * IQR(x), keep.stuff = FALSE)
```

```
> #install.packages("jpeg")  
> library(jpeg)  
> ruddyduck.img <- readJPEG("ruddyduck.jpg")  
> plot(0, xlim=c(0, 14), ylim=c(-6, 4), type='n', xlab="", ylab="",  
+      main="Spline approximate to the top profile of the ruddy duck")  
> rasterImage(ruddyduck.img, 0.6, -6, 13.8, 3.3)  
> abline(v=1:14, h=-6:4, col=gray)
```

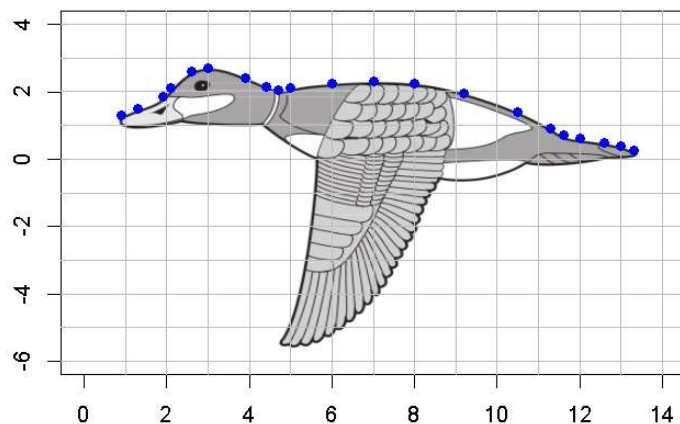
Spline approximate to the top profile of the ruddy duck



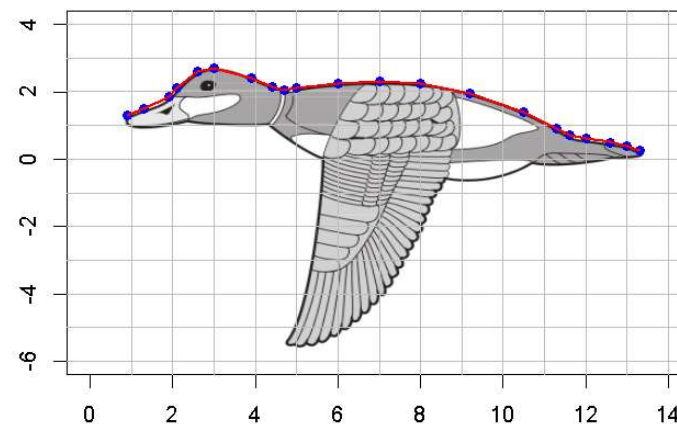
smooth.spline {stats}: Fit a Smoothing Spline

```
> ruddyduck.dat <- read.table("ruddyduck.txt", header=T, sep="\t")
> head(ruddyduck.dat)
      x    fx
1 0.9 1.30
2 1.3 1.50
3 1.9 1.85
4 2.1 2.10
5 2.6 2.60
6 3.0 2.70
> points(ruddyduck.dat, col="blue", pch=16)
>
> duck.spl <- smooth.spline(ruddyduck.dat$fx ~ ruddyduck.dat$x)
> lines(duck.spl, col = "red", lwd=2)
```

Spline approximate to the top profile of the ruddy duck



Spline approximate to the top profile of the ruddy duck



Cubic Spline Interpolation

16/16

Cubic Splines Interpolant

Definition 3.10

Given a function f defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 < \dots < x_n = b$, a cubic spline interpolant S for f is a function that satisfies the following conditions:

- (a) $S(x)$ is a cubic polynomial ($S_j(x)$) on $[x_j, x_{j+1}]$.
- (b) $S_j(x_j) = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$, $j = 0, 1, \dots, n-1$;
- (c) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$; (d) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$;
- (e) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0, 1, \dots, n-2$;
- (f) One of the following sets of boundary conditions is satisfied:
 - (i) $S''(x_0) = S''(x_n) = 0$ (natural or free boundary);
 - (ii) $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (clamped boundary).

Construction of a Cubic Spline (conti.)

- (12) This system involves only the $\{c_j\}_{j=0}^n$ as unknowns.
- (13) The values of $\{h_j\}_{j=0}^{n-1}$ and $\{a_j\}_{j=0}^n$ are given, respectively, by the spacing of the nodes $\{x_j\}_{j=0}^n$ and the values of f at the nodes.
- (14) So once the values of $\{c_j\}_{j=0}^n$ are determined, it is a simple matter to find the remainder of the constants $\{b_j\}_{j=0}^{n-1}$ from Eq. (3.20) and $\{d_j\}_{j=0}^{n-1}$ from Eq. (3.17)
- (15) The major question that arises in connection with this construction is whether the values of $\{c_j\}_{j=0}^n$ can be found using the system of equations given in (3.21) and, if so, whether these values are unique.

ALGORITHM 034: Natural Cubic Spline

To construct the cubic spline interpolant S for the function f , defined at the numbers $x_0 < x_1 < \dots < x_n$, satisfying $S''(x_0) = S''(x_n) = 0$:

INPUT $n; x_0, x_1, \dots, x_n; a_0 = f(x_0), a_1 = f(x_1), \dots, a_n = f(x_n)$.

OUTPUT a_j, b_j, c_j, d_j for $j = 0, 1, \dots, n-1$.

(Note: $S(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$ for $x_j \leq x \leq x_{j+1}$.)

Step 1 For $i = 0, 1, \dots, n-1$ set $h_i = x_{i+1} - x_i$.

Step 2 For $i = 1, 2, \dots, n-1$ set

$$\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}).$$

Step 3 Set $l_0 = 1$; (Steps 3, 4, 5, and part of Step 6 solve a tridiagonal linear system using a method described in Algorithm 6.7.)

$$\mu_0 = 0;$$
$$z_0 = 0.$$

ALGORITHM 034: Natural Cubic Spline (conti.)

Step 4 For $i = 1, 2, \dots, n-1$

$$\text{set } l_i = 2(x_{i+1} - x_{i-1}) - h_{i-1}\mu_{i-1};$$

$$\mu_i = h_i/l_i;$$

$$z_i = (\alpha_i - h_{i-1}z_{i-1})/l_i.$$

Step 5 Set $l_n = 1$;

$$z_n = 0;$$

$$c_n = 0.$$

Step 6 For $j = n-1, n-2, \dots, 0$

$$\text{set } c_j = z_j - \mu_j c_{j+1};$$

$$b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3;$$

$$d_j = (c_{j+1} - c_j)/(3h_j).$$

Step 7 **OUTPUT** (a_j, b_j, c_j, d_j) for $j = 0, 1, \dots, n-1$;
STOP.

