台灣人工智慧學校

經理人週末研修班

資料處理方法

平滑技巧/遺失值處理
資料轉換/重抽法則

吳漢銘

國立臺北大學 統計學系

http://www.hmwu.idv.tw

Simple Moving Average

- In statistics, a moving average (移動平均) (rolling average or running average) (簡稱均線) is a calculation to analyze data points by creating series of averages of different subsets of the full data set.
- 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.

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

$$SMA_{t1,n} = SMA_{t0,n} - \frac{p_1}{n} + \frac{p_{n+1}}{n}$$

Moving Average Acting as Support - Potential Buy Signal

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

Moving Average Acting as Resistance - Potential Sell Signal

- 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.

http://www.onlinetradingconcepts.com/TechnicalAnalysis/MASimple.ht

An n-day WMA (Weighted moving average)

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

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

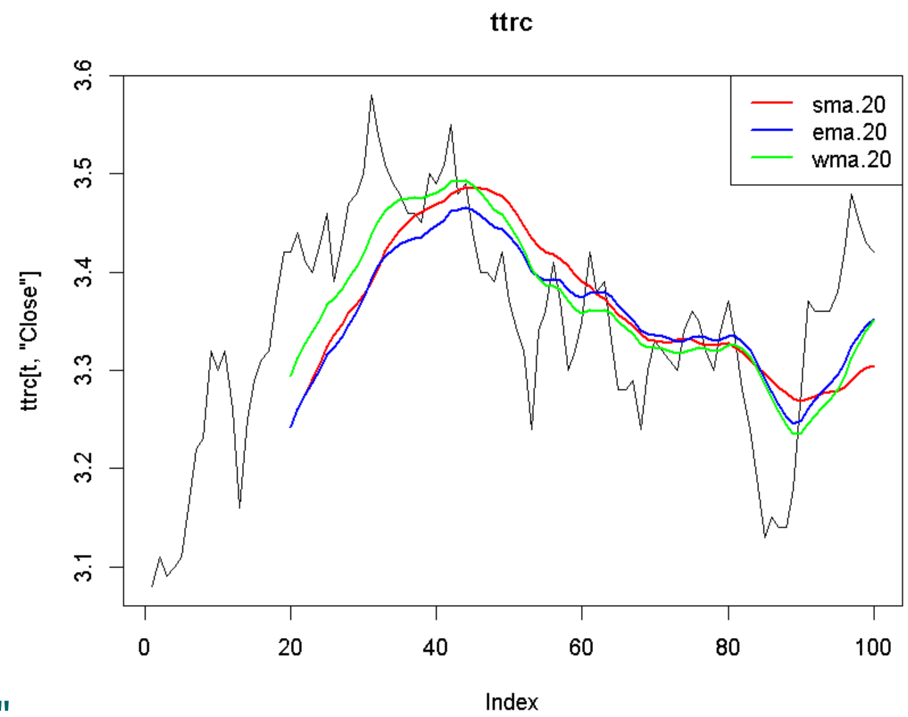
```
SMA(x, n = 10, ...)
EMA(x, n = 10, wilder = FALSE, ratio = NULL, ...)
DEMA(x, n = 10, v = 1, wilder = FALSE, ratio = NULL)
WMA(x, n = 10, wts = 1:n, ...)
EVWMA(price, volume, n = 10, ...)
ZLEMA(x, n = 10, ratio = NULL, ...)
VWAP(price, volume, n = 10, ...)
VMA(x, w, ratio = 1, ...)
HMA(x, n = 20,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

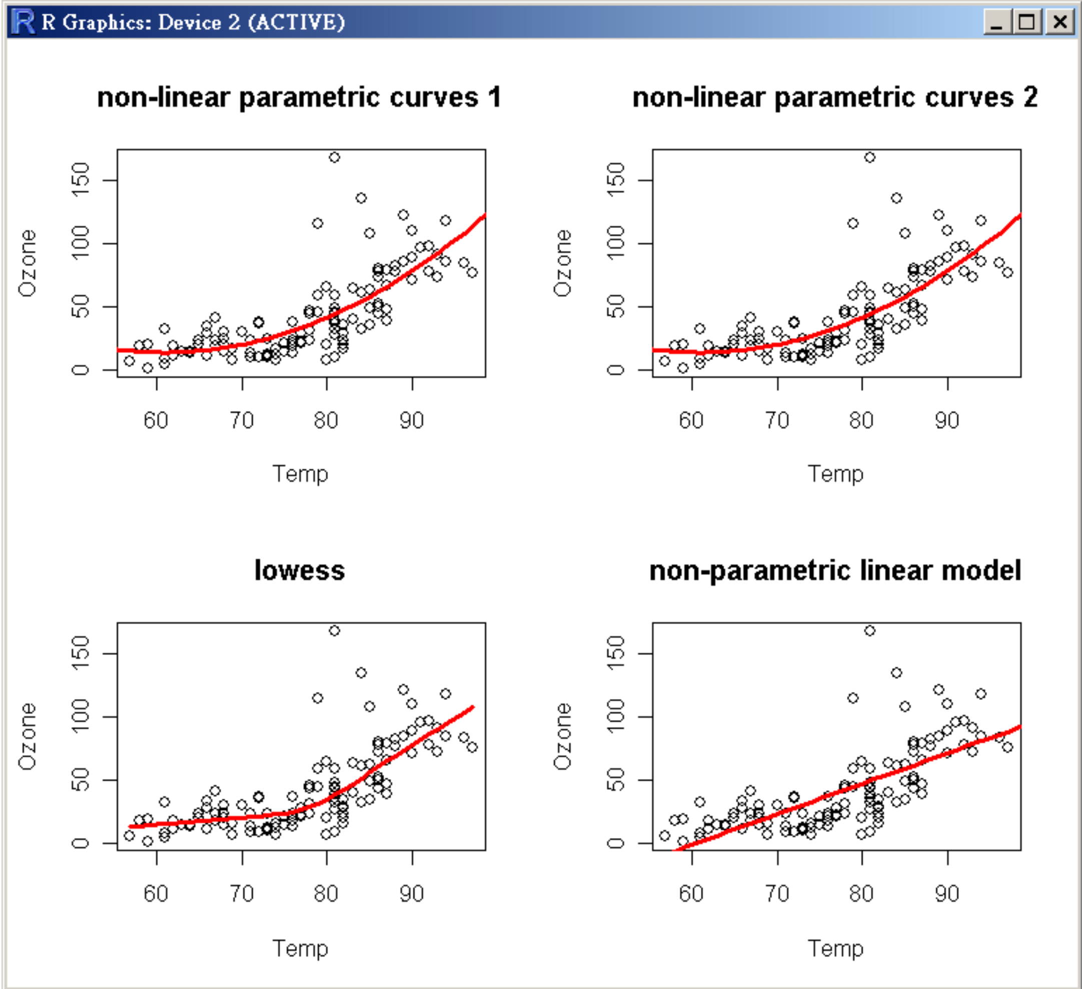
```
> # install.packages("TTR")
> library(TTR)
> data(ttrc)
> dim(ttrc)
[1] 5550    6
> head(ttrc)
        Date Open High  Low Close  Volume
1 1985-01-02 3.18 3.18 3.08  3.08 1870906
2 1985-01-03 3.09 3.15 3.09  3.11 3099506
3 1985-01-04 3.11 3.12 3.08  3.09 2274157
4 1985-01-07 3.09 3.12 3.07  3.10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

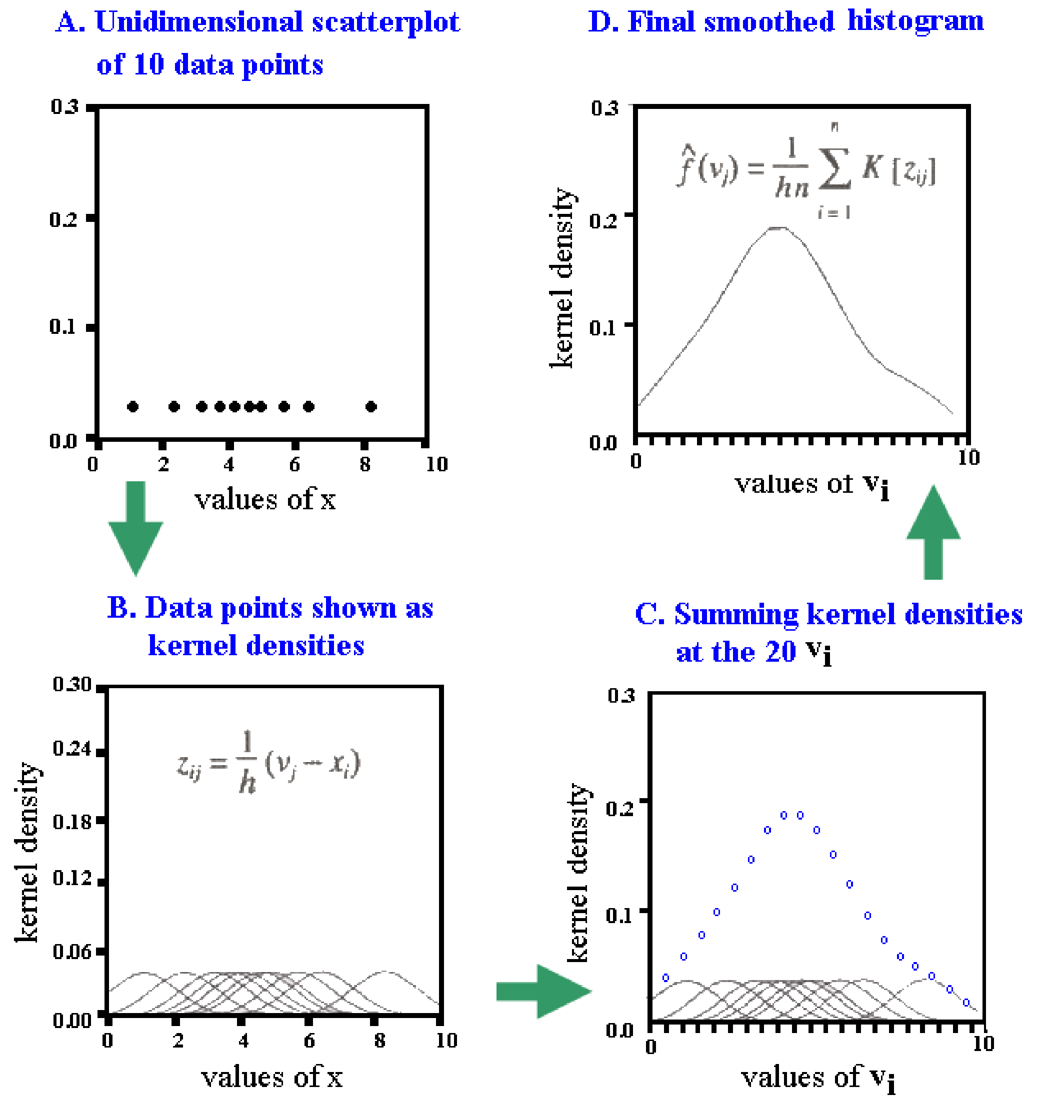
Example Methods

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



Density Plots (Smoothed Histograms) (1/3)

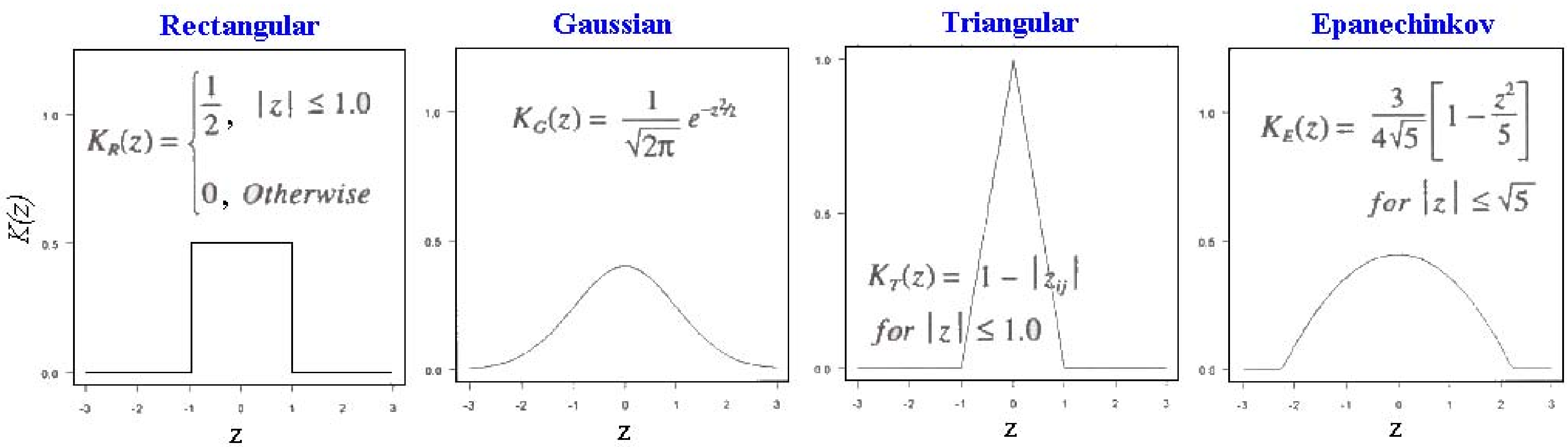
Constructing a Smoothed Histogram (Jacoby, 1997)



Kernel Density Estimation

- Selection of kernels
- Selection of bandwidth

Figures modified from Jacoby (1997)



nonparametric regression

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

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

kernel regression

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

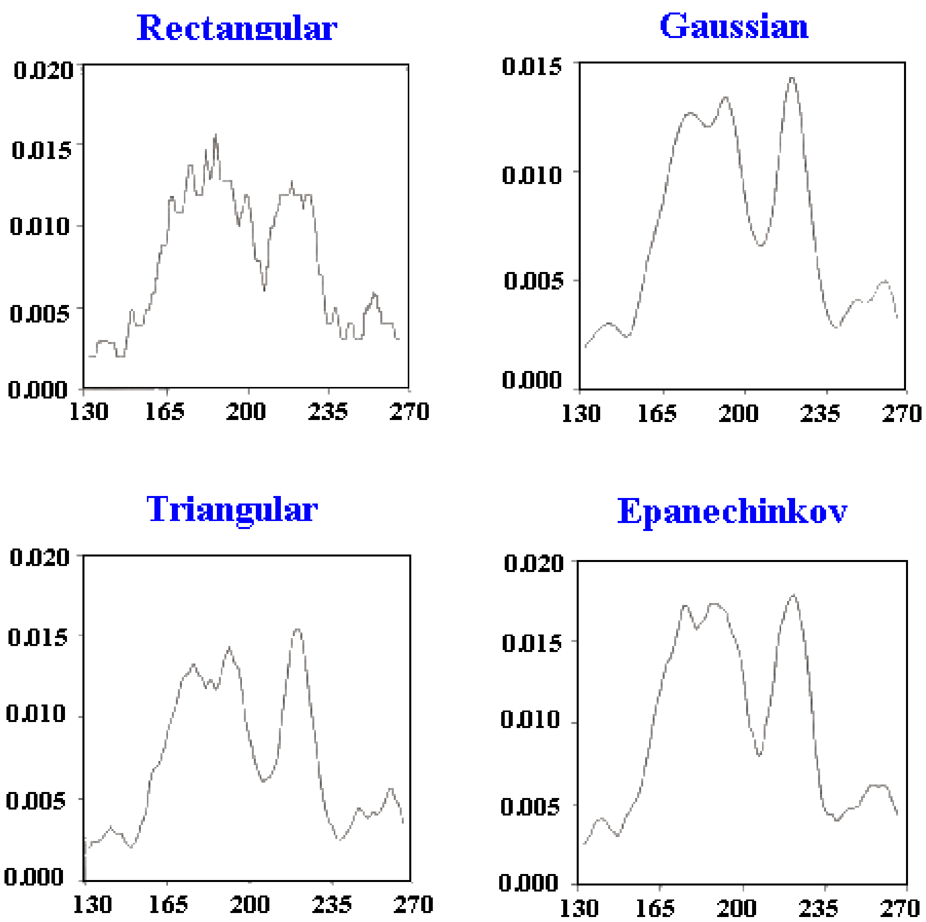
$$\hat{f}(v_j) = \frac{1}{hn} \sum_{i=1}^{n} K[z_{ij}]$$

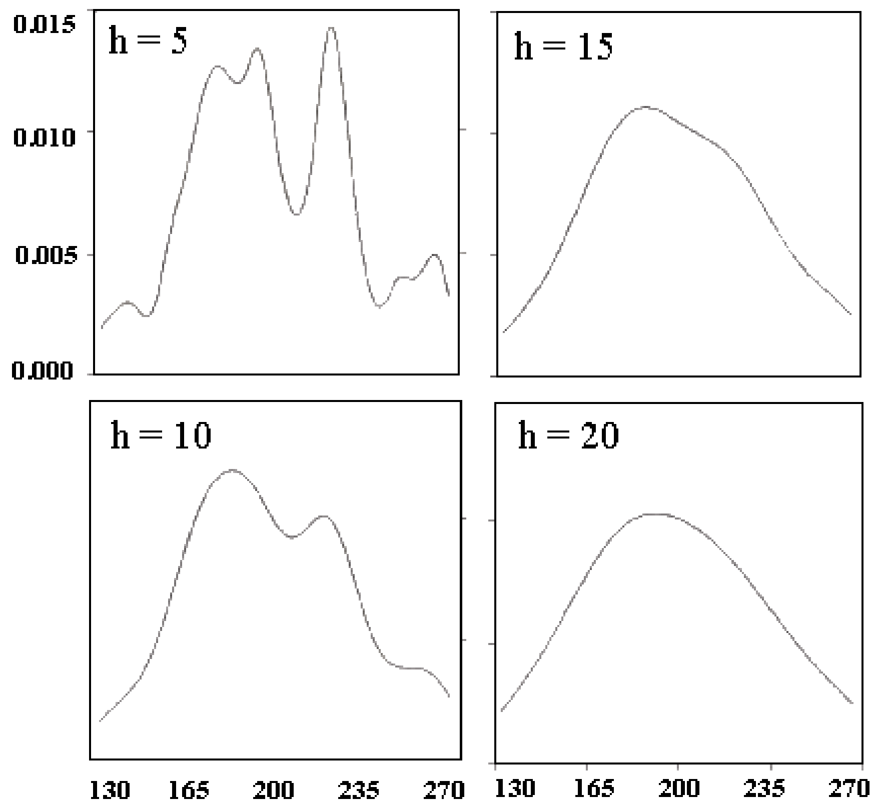
$$z_{ij} = \frac{1}{h}(v_j - x_i)$$

k-nearest-neighbors regression.

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i$$

Kernel Density Estimation

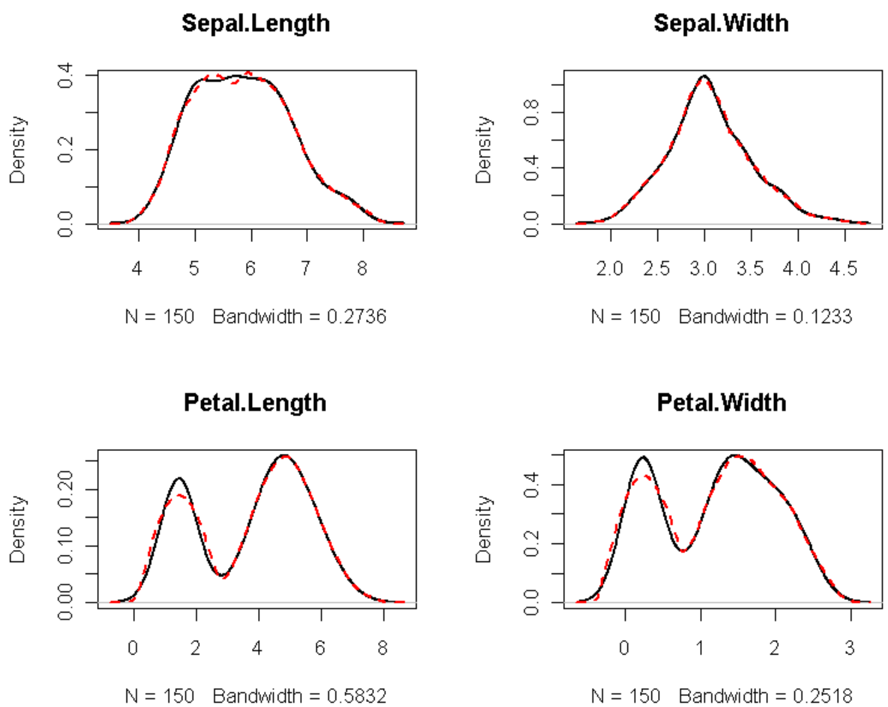




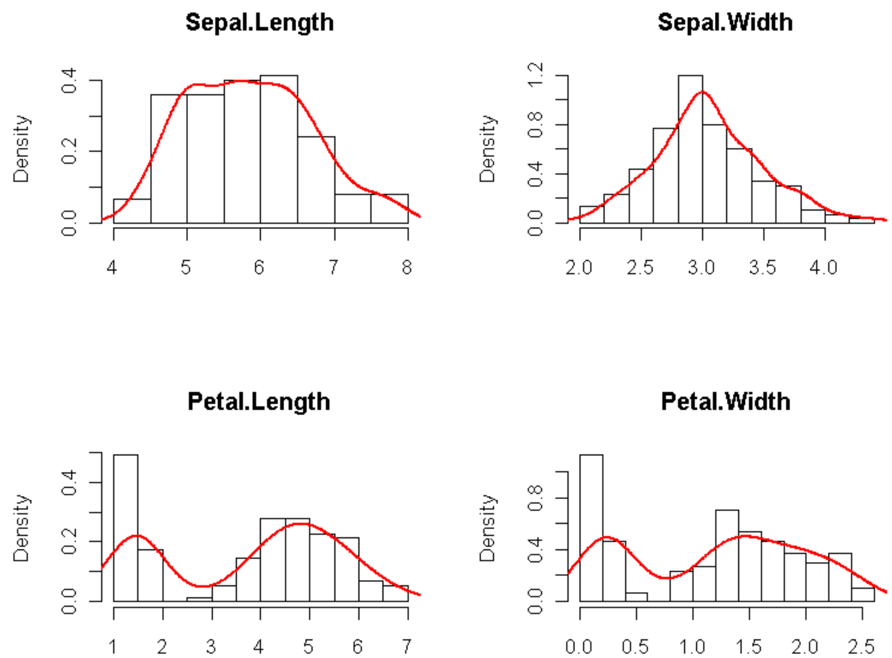
Kernel Density Estimation


```
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))
```

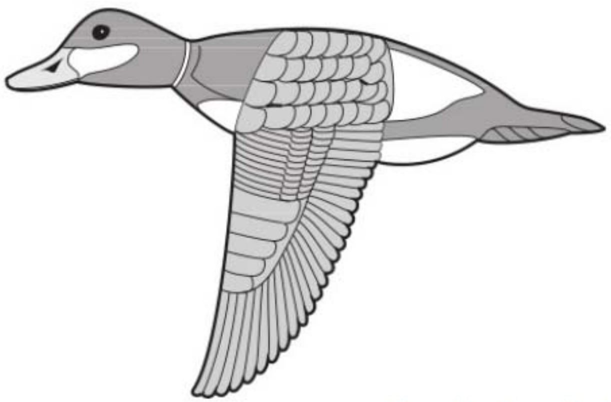


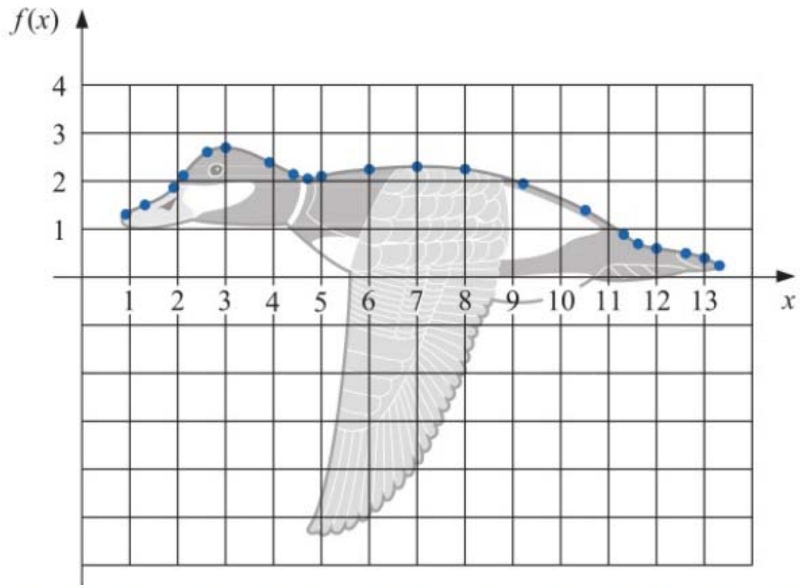

```
> hist(iris$Sepal.Length, prob=T)
> lines(density(iris$Sepal.Length), col="red")
```



Spline approximate to the top profile of the ruddy duck

ruddy duck (棕硬尾鴨) (雄)





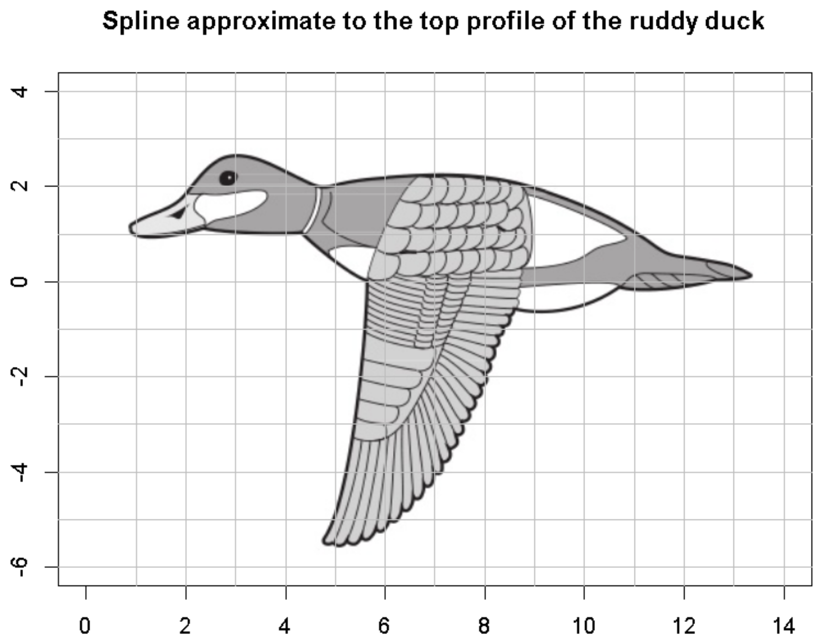
x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3
$f(x)$	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25

smooth.spline {stats}: Fit a Smoothing Spline

Usage

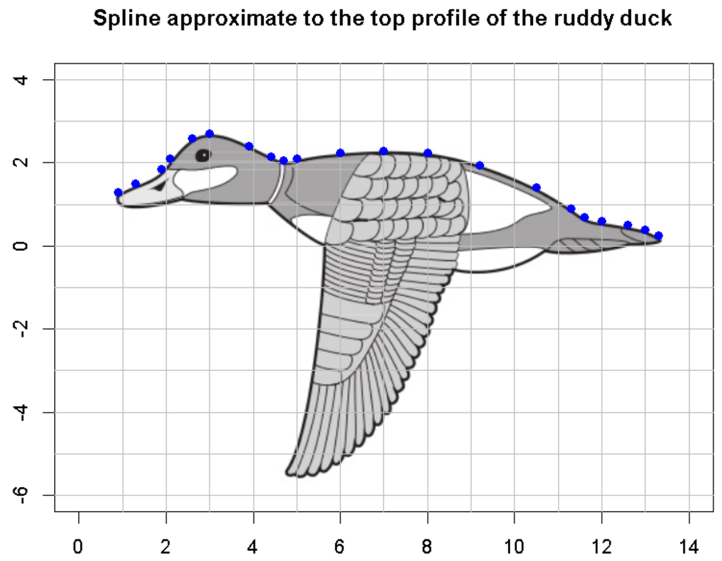
```
smooth.spline(x, y = NULL, w = NULL, df, spar = NULL, lambda = NULL, cv = FALSE,
              all.knots = FALSE, nknots = .nknots.smspl,
              keep.data = TRUE, df.offset = 0, penalty = 1,
              control.spar = list(), tol = 1e-6 * IQR(x), keep.stuff = FALSE)
```

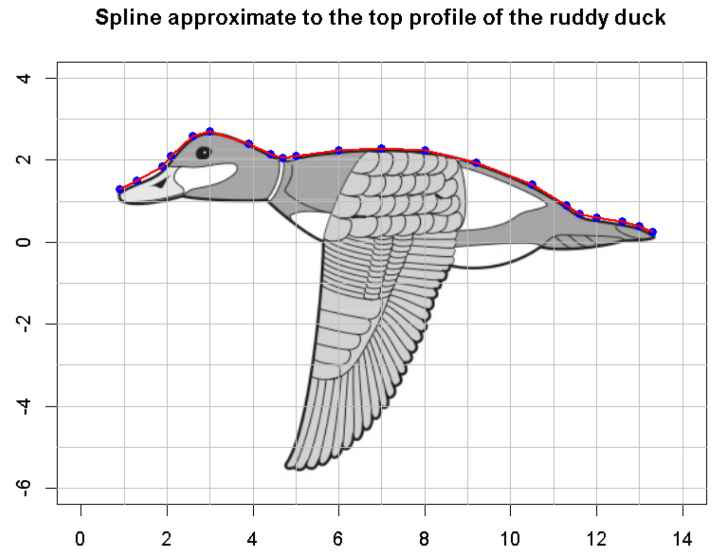
```
> #install.packages("jpeg")
> library(jpeg)
> ruddyduck.img <- readJPEG("ruddyduck.jpg")
> plot(0, xlim=c(0, 14), ylim=c(-6, 4), type='n', xlab="", ylab="",
+      main="Spline approximate to the top profile of the ruddy duck")
> rasterImage(ruddyduck.img, 0.6, -6, 13.8, 3.3)
> abline(v=1:14, h=-6:4,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



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)
```





Cubic Spline Interpolation

Cubic Splines Interpolant

Definition 3.10

Given a function f defined on $[a,b]$ and a set of nodes $a = x_0 < x_1 < \cdots < x_n = b$, 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 $S_j(x_{j+1}) = f(x_{j+1})$, $j = 0, 1, \cdots, n-1$;

(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 $j = 0, 1, \cdots, n-2$;

(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 $x_0 < x_1 < \cdots < x_n$, satisfying $S''(x_0) = S''(x_n) = 0$:

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

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

(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 $x_j \le x \le x_{j+1}$.)

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

Step 2 For $i = 1, 2, \ldots, n-1$ 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 $i = 1, 2, \ldots, n-1$
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 $j = n-1, n-2, \ldots, 0$
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 $j = 0, 1, \ldots, n-1)$;
STOP.

遺失值 (Missing Data)

- When data are missing for a variable for all cases: **latent** or **unobserved**.
- When data are missing for all variables for a given case: **unit non-response**.
- Missing data (missing values for certain variables for certain cases): **item non-response**.

	A	B	C	D	E	F	G
1	ID	C	Y	X1	X2	X3	X4
2	s1	1	78.3	69.6	74.3	NA	5.22
3	s2	2	77	69.9	72.54	NA	3.98
4	s3	3	72.2	65.7	69.74	NA	4.89
5	s4	1	33.4	NA	30.97	NA	21.54
6	s5	2	32.65	28.35	30.54	NA	9.82
7	s6	3	35.45	28.5	32.01	NA	19.81
8	s7	1	424	378	403.55	NA	12.98
9	s8	2	NA	NA	NA	NA	NA
10	s9	3	355	312.5	339.96	NA	14.14
11	s10	1	18.2	15.5	17.19	NA	13.93
12	s11	2	18.3	15.3	16.38	NA	6.92
13	s12	3	16.1	13.9	14.92	NA	10.15
14	s13	1	23.75	20.2	22.19	NA	32.81

Missing Values in R

- **NA**: a missing value ("not available"), **"NA"**: a string.
- **x[1]== NA** is not a valid logical expression and will not return **FALSE** as one would expect but will return **NA**.

```
> myvector <- c(10, 20, NA, 30, 40)
> myvector
[1] 10 20 NA 30 40
> mycountry <- c("Austria", "Australia", NA, NA, "Germany", "NA")
> mycountry
[1] "Austria"   "Australia" NA          NA          "Germany"   "NA"
> is.na(myvector)
[1] FALSE FALSE  TRUE FALSE FALSE
> which(is.na(myvector))
[1] 3
> x <- c(1, 4, 7, 10)
> x[4] <- NA # sets the 4th element to NA
> x
[1]  1  4  7 NA
> is.na(x) <- 1 # sets the first element t
> x
[1] NA  4  7 NA
```

```
#Recoding Values to Missing
mydata$v1[mydata$v1==99] <- NA
```

```
> set.seed(12345)
> mydata <- matrix(round(rnorm(20), 2), ncol=5)
> mydata[sample(1:20, 3)] <- NA
> mydata
      [,1]  [,2]  [,3]  [,4]  [,5]
[1,]  0.59  0.61    NA  0.37    NA
[2,]  0.71 -1.82 -0.92  0.52 -0.33
[3,] -0.11  0.63 -0.12 -0.75  1.12
[4,] -0.45 -0.28  1.82    NA  0.30
> which(colSums(is.na(mydata)) > 0)
[1] 3 4 5
```

NOTE: **NULL** denotes something which never existed and cannot exist at all.

NA in Summary Functions

- Most of the statistical summary functions (**mean, var, sum, min, max**, etc.) accept an argument called **na.rm**, which can be set to **TRUE** if you want missing values to be removed before the summary is calculated. (default : **FALSE**)

- For functions that don't provide such an argument, the negation operator (**!**) can be used in an expression like **x[!is.na(x)]** to create a vector which contains only the nonmissing values in **x**.

```
> x <- c(1, 4, NA, 10)
>  summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
    1.0     2.5     4.0     5.0     7.0    10.0       1 
>  mean(x)
[1] NA
>  sd(x)
[1] NA
>  mean(x, na.rm=TRUE)
[1] 5
>  sd(x, na.rm=TRUE)
[1] 4.582576
> x[!is.na(x)]
[1]  1  4 10
```

Other Special Values in R

- `NaN` : "not a number" which can arise for example when we try to compute the undeterminate 0/0.

- `Inf` which results from computations like 1/0.

- Using the functions `is.finite()` and `is.infinite()` we can determine whether a number is finite or not.

```
> x <- c(1, 0, 10)
> x/x
[1]   1 NaN   1
> is.nan(x/x)
[1] FALSE  TRUE FALSE
```

```
> 1/x
[1] 1.0 Inf 0.1
> is.finite(1/x)
[1]  TRUE FALSE  TRUE
>
> -10/x
[1]  -10 -Inf   -1
> is.infinite(-10/x)
[1] FALSE  TRUE FALSE
```

```
> exp(-Inf)
[1] 0
> 0/Inf
[1] 0
> Inf - Inf
[1] NaN
> Inf/Inf
[1] NaN
```

R Packages for Dealing With Missing Values

https://cran.r-project.org/web/packages/package-name/

- `Amelia (Amelia II)`: A Program for Missing Data
- `hot.deck:` Multiple Hot-Deck Imputation
- `HotDeckImputation`: Hot Deck Imputation Methods for Missing Data
- `impute`: (Bioconductor) Imputation for Microarray Data
- `mi`: Missing Data Imputation and Model Checking
- `mice`: Multivariate Imputation by Chained Equations
- `missForest:` Nonparametric Missing Value Imputation using Random Forest
- `missMDA`: Handling Missing Values with Multivariate Data Analysis (e.g., imputePCA, imputeMCA,)
- `mitools`: Tools for Multiple Imputation of Missing Data
- `norm:` Analysis of Multivariate Normal Datasets with Missing Values
- `VIM`: Visualization and Imputation of Missing Values
- R packages support for missing values imputation.
  - `Hmisc:` Harrell Miscellaneous
  - `survey`: analysis of complex survey samples
  - `Zelig`: Everyone's Statistical Software
  - `rfImpute{randomForest}`: Imputations by randomForest
  - `imputation{rminer}`: Data Mining Classification and Regression Methods, Missing data imputation (e.g. substitution by value or hotdeck method).
  - `impute.svd{bcv}`: Cross-Validation for the SVD (Bi-Cross-Validation), Missing value imputation via a low-rank SVD approximation estimated by the EM algorithm.
  - `mlr`: Machine Learning in R provides several imputation methods.
    https://mlr-org.github.io/mlr-tutorial/release/html/index.html

Package "`imputation`" was removed from the CRAN. (Archived on 2014-01-14)

R Package: MICE

- `mice`: Multivariate Imputation by Chained Equations in R by Stef van Buuren.

- Imputing missing values on mixed data.
  - Continuous data: Predictive mean matching, Bayesian linear regression, Linear regression ignoring model error, Unconditional mean imputation etc.
  - Binary data: Logistic Regression, Logistic regression with bootstrap
  - Categorical data (More than 2 categories) - Polytomous logistic regression, Proportional odds model etc.
  - Mixed data (Can work for both Continuous and Categorical) - CART, Random Forest, Sample (Random sample from the observed values).

Source: http://www.listendata.com/2015/08/missing-imputation-with-mice-package-in.html

Imputation using MICE Package

```
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> dim(airquality)
[1] 153   6
> mydata <- airquality
> mydata[4:10,3] <- rep(NA,7)
> mydata[1:5,4] <- NA
>
> # Use numerical variables as examples here.
> # Ozone is the variable with the most missing datapoints.
> summary(mydata)
     Ozone            Solar.R           Wind             Temp           Month            Day      
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :57.00   Min.   :5.000   Min.   : 1.0  
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:73.00   1st Qu.:6.000   1st Qu.: 8.0  
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00   Median :7.000   Median :16.0  
 Mean   : 42.13   Mean   :185.9   Mean   : 9.806   Mean   :78.28   Mean   :6.993   Mean   :15.8  
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00   3rd Qu.:8.000   3rd Qu.:23.0  
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.   :9.000   Max.   :31.0  
 NA's   :37       NA's   :7       NA's   :7        NA's   :5                                      
```

Sourec: http://www.r-bloggers.com/imputing-missing-data-with-r-mice-package/

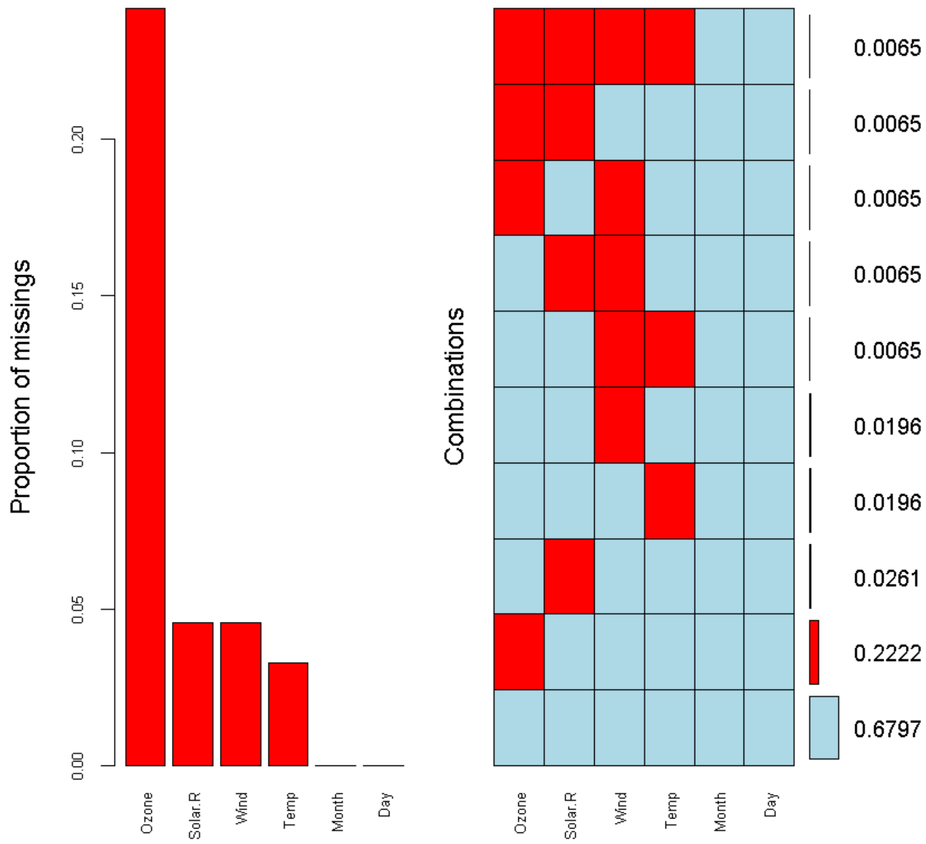
Visualizing the Pattern of Missing Data

```
> library(mice)
> md.pattern(mydata)
    Month Day Temp Solar.R Wind Ozone
104     1   1    1       1    1     1  0
 34     1   1    1       1    1     0  1
  4     1   1    1       0    1     1  1
  3     1   1    1       1    0     1  1
  3     1   1    0       1    1     1  1
  1     1   1    1       0    1     0  2
  1     1   1    1       1    0     0  2
  1     1   1    1       0    0     1  2
  1     1   1    0       1    0     1  2
  1     1   1    0       0    0     0  4
        0   0    5       7    7    37 56
```

```
> library(VIM)
> mydata.aggrplot <- aggr(mydata,
col=c('lightblue','red'), numbers=TRUE,
prop = TRUE, sortVars=TRUE,
labels=names(mydata), cex.axis=.7, gap=3)

 Variables sorted by number of missings:
 Variable      Count
    Ozone 0.24183007
  Solar.R 0.04575163
     Wind 0.04575163
     Temp 0.03267974
    Month 0.00000000
      Day 0.00000000
```

#104 samples are complete, 34 samples miss only the Ozone measurement, 4 samples miss only the Solar.R value and so on.



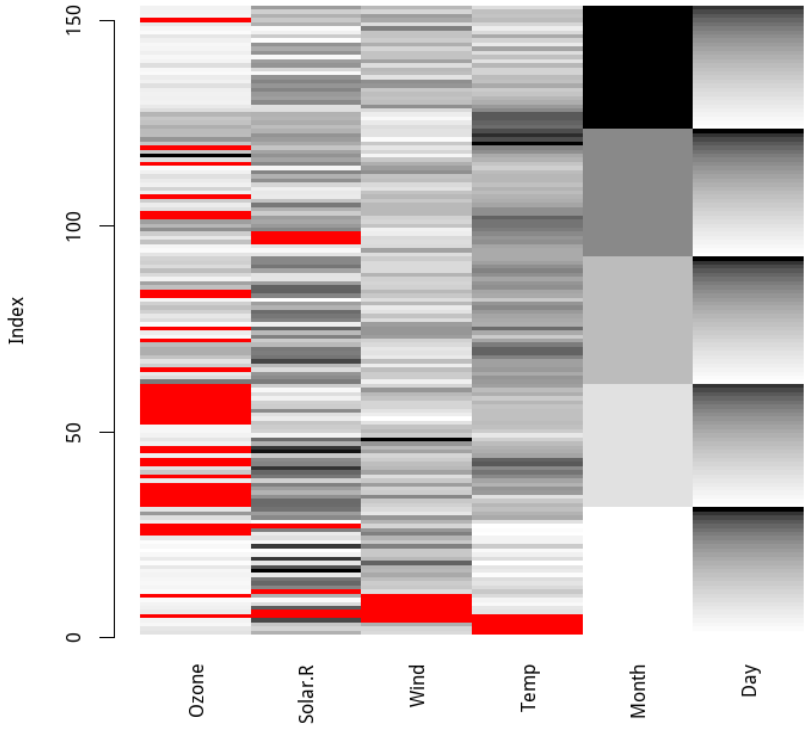
Aggregation Plot

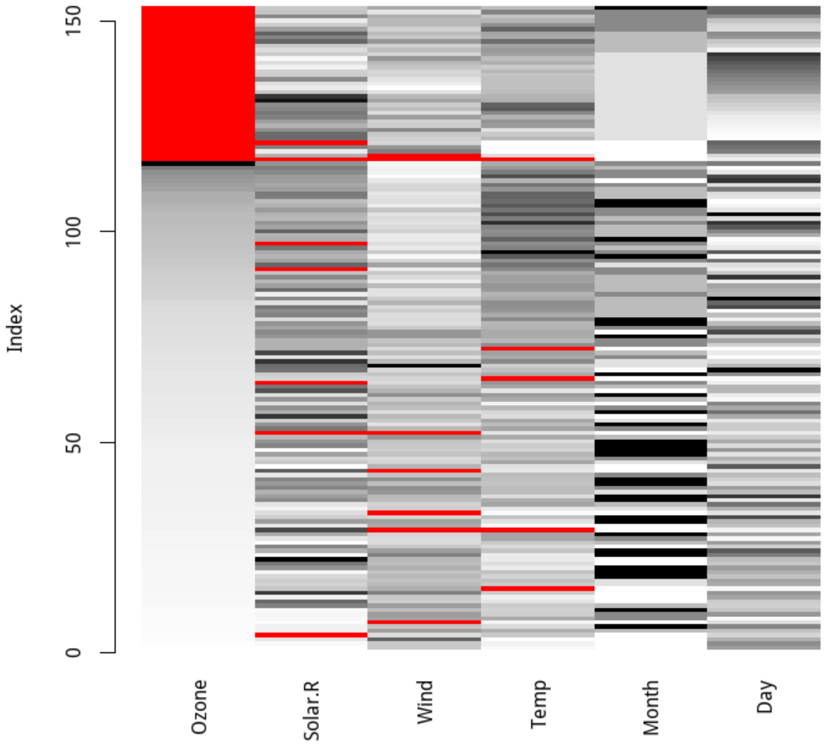
Matrix Plot

```
> matrixplot(mydata)

Click in a column to sort by the corresponding variable.
To regain use of the VIM GUI and the R console, click outside the plot region.

Matrix plot sorted by variable 'Ozone'.
```





(T1) List-wise Deletion

- Also called the **complete case analysis**.
- All units with missing data for a variable are removed and the analysis is performed with the remaining units (complete cases).
- This is the default approach in most statistical packages.
- The use of this method is only justified if the missing data generation mechanism is **MCAR**.
- In R, using the function `na.omit()` or extract complete observations using the function `complete.cases()`.

```
> mdata <- matrix(rnorm(15), nrow=5)
> mdata[sample(1:15, 4)] <- NA
> mdata <- as.data.frame(mdata)
> mdata
           V1         V2          V3
1 -0.62222501  1.0807983          NA
2  0.07124865  0.5216675 -0.08334454
3  1.70707399  0.1004917  0.88197789
4          NA -0.6595201 -0.08387860
5          NA  1.6138847          NA
> (x1 <- na.omit(mdata))
          V1        V2          V3
2 0.07124865 0.5216675 -0.08334454
3 1.70707399 0.1004917  0.88197789
> (x2 <- mdata[complete.cases(mdata),])
          V1        V2          V3
2 0.07124865 0.5216675 -0.08334454
3 1.70707399 0.1004917  0.88197789
> mdata[!complete.cases(mdata),]
         V1         V2         V3
1 -0.622225  1.0807983         NA
4        NA -0.6595201 -0.0838786
5        NA  1.6138847         NA
```

快速分析一下，得知資料大概狀況

(T2) Pairwise Deletion

- To compute a covariance matrix, each two cases will be used for which the values of both corresponding variables are available. In R,
  - `use="everything"` (default): use all observations will result in a covariance matrix most likely consisting of NAs.
  - `use="all.obs"`: the presence of missing observations will produce an error.
  - `use="complete.obs"`: missing values are handled by list-wise deletion (and if there are no complete cases, an error appears).
  - `use="pairwise.complete.obs"`: the covariance between each pair of variables is computed using all complete pairs of observations on those variables.
- This can result in covariance or correlation matrices which are not positive semi-definite, as well as NA entries if there are no complete pairs for the given pair of variables.

```
> cov(mdata)
   V1        V2 V3
V1 NA        NA NA
V2 NA 0.7694197 NA
V3 NA        NA NA
> cov(mdata, use = "all.obs")
Error in cov(mdata, use = "all.obs") :
missing observations in cov/cor
> cov(mdata, use = "complete.obs")
           V1          V2         V3
V1  1.3379623 -0.34448500  0.7895494
V2 -0.3444850  0.08869452 -0.2032852
V3  0.7895494 -0.20328521  0.4659237
```

```
> cov(mdata, use = "na.or.complete")
           V1          V2         V3
V1  1.3379623 -0.34448500  0.7895494
V2 -0.3444850  0.08869452 -0.2032852
V3  0.7895494 -0.20328521  0.4659237
> cov(mdata, use = "pairwise")
           V1          V2         V3
V1  1.4304107 -0.56002326 0.78954945
V2 -0.5600233  0.76941970 0.05468712
V3  0.7895494  0.05468712 0.31078774
```

(T4) Mean Substitution

- A very simple but popular approach is to substitute means for the missing values.
- The method preserves sample size and does not reduce the statistical power associated with sample size in comparison with list-wise or pairwise deletion.
- This method produces biased estimates and can severely distort the distribution of the variable in which missing values are substituted.
- This results in underestimates of the standard deviations and **distorts relationships between variables** (estimates of the correlation are pulled toward zero).

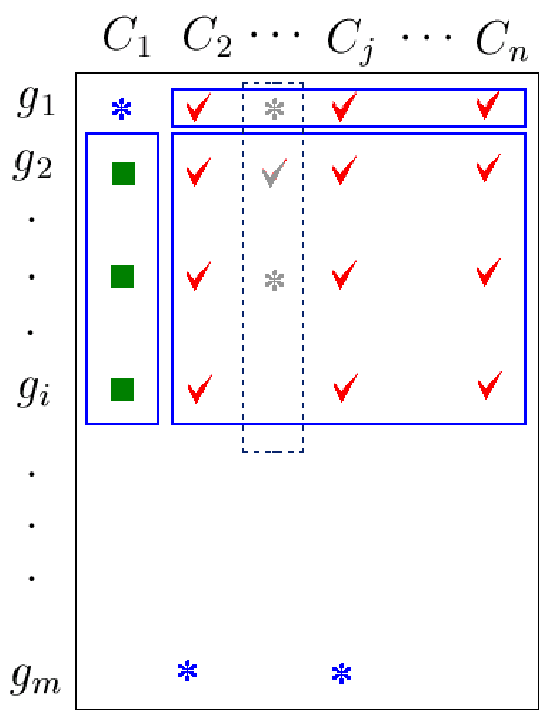
Due to these **distributional problems**, it is often recommended to ignore missing values rather than impute values by mean substitution (Little and Rubin, 1989.)

```
mean.subst <- function(x) {
   x[is.na(x)] <- mean(x, na.rm = TRUE)
   x
}
```

```
> mdata
           V1         V2          V3
1 -0.62222501  1.0807983          NA
2  0.07124865  0.5216675 -0.08334454
3  1.70707399  0.1004917  0.88197789
4          NA -0.6595201 -0.08387860
5          NA  1.6138847          NA
> mdata.mip <- apply(mdata, 2, mean.subst)
> mdata.mip
              V1         V2          V3
[1,] -0.62222501  1.0807983  0.23825158
[2,]  0.07124865  0.5216675 -0.08334454
[3,]  1.70707399  0.1004917  0.88197789
[4,]  0.38536588 -0.6595201 -0.08387860
[5,]  0.38536588  1.6138847  0.23825158
```

(A1) K-Nearest Neighbour Imputation

- KNN imputation searches for the k-nearest observations (respective to the observation which has to be imputed) and replaces the missing value with the mean of the found *k* observations.
- It is recommended to use the (weighted) median instead of the arithmetic mean.
- **KNN** minimize data modeling assumptions and take advantage of the correlation structure of the data.



KNNimpute

Model:

$$\{g_{(k)}, k = 1, 2, \cdots, K\} = \underset{k}{\text{args}} \ \max_{i \in \mathrm{C}} \mathrm{Corr}(g_1, g_i)$$

$$\{g_{(k)}, k = 1, 2, \cdots, K\} = \underset{k}{\text{args}} \ \min_{i \in \mathrm{C}} \mathrm{Dist}(g_1, g_i)$$

C: Observed C_i's without missing values

Imputation:

Average $\widehat{C_1(g_1)} = \frac{1}{K}\sum_{k=1}^{K} C_1(g_k)$

Weighted Average $\widehat{C_1(g_1)} = \frac{\sum_{k=1}^{K} w_k C_1(g_k)}{\sum_{k=1}^{K} w_k}$

$$w_k = \frac{1}{\sum_{j \in \mathrm{C}} [C_j(g_k) - C_1(g_1)]^2}$$

kNN {VIM}:
k-Nearest Neighbour Imputation

Description

k-Nearest Neighbour Imputation based on a variation of the Gower Distance for numerical, categorical, ordered and semi-continous variables.

Usage

```
kNN(data, variable = colnames(data), metric = NULL, k = 5,
  dist_var = colnames(data), weights = NULL, numFun = median,
  catFun = maxCat, makeNA = NULL, NAcond = NULL, impNA = TRUE,
  donorcond = NULL, mixed = vector(), mixed.constant = NULL,
  trace = FALSE, imp_var = TRUE, imp_suffix = "imp", addRandom = FALSE,
  useImputedDist = TRUE, weightDist = FALSE)
```

```
> names(airquality)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
> airquality.imp.median <- kNN(airquality[1:4], k=5)
> head(airquality.imp.median)
  Ozone Solar.R Wind Temp Ozone_imp Solar.R_imp Wind_imp Temp_imp
1    41     190  7.4   67     FALSE       FALSE    FALSE    FALSE
2    36     118  8.0   72     FALSE       FALSE    FALSE    FALSE
3    12     149 12.6   74     FALSE       FALSE    FALSE    FALSE
4    18     313 11.5   62     FALSE       FALSE    FALSE    FALSE
5    35      92 14.3   56      TRUE        TRUE    FALSE    FALSE
6    28     242 14.9   66     FALSE        TRUE    FALSE    FALSE
```

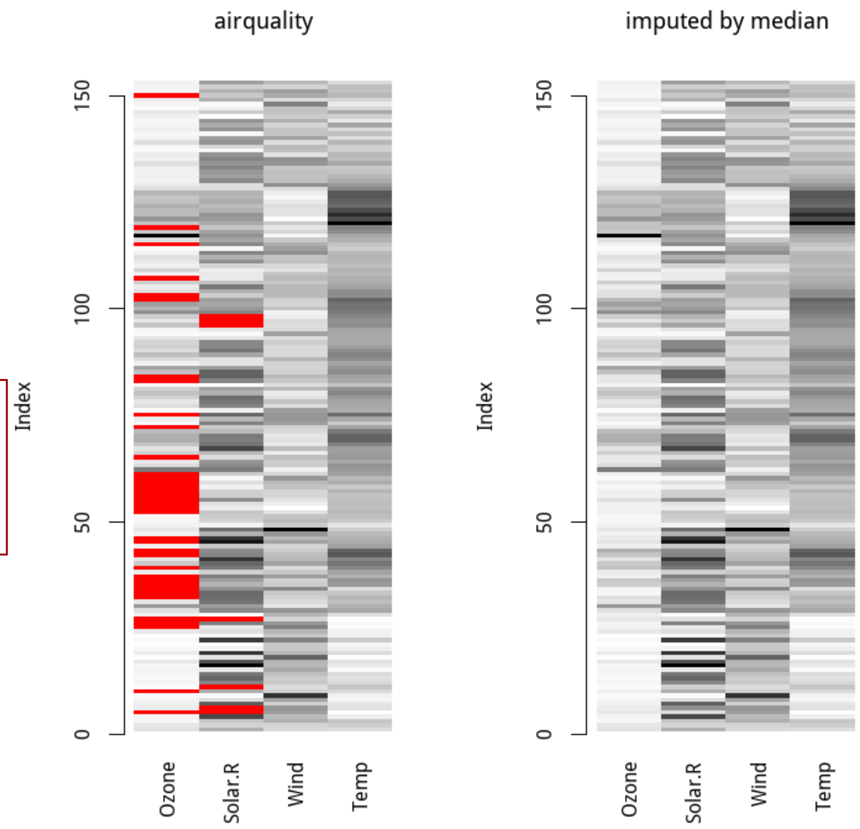

- Gower JC, 1971, A General Coefficient of Similarity and Some of Its Properties. Biometrics, 857–871.
- Alexander Kowarik and Matthias Templ, 2016, Imputation with the R Package VIM, Journal of Statistical Software, Volume 74, Issue 7.

kNN {VIM}: k-Nearest Neighbour Imputation

```
> matrixplot(airquality[1:4], interactive = F, main="airquality")
> matrixplot(airquality.imp.median[1:4], interactive = F, main="imputed by median")
```

```
trim_mean <- function(x){
   mean(x, trim = 0.1)
}
```

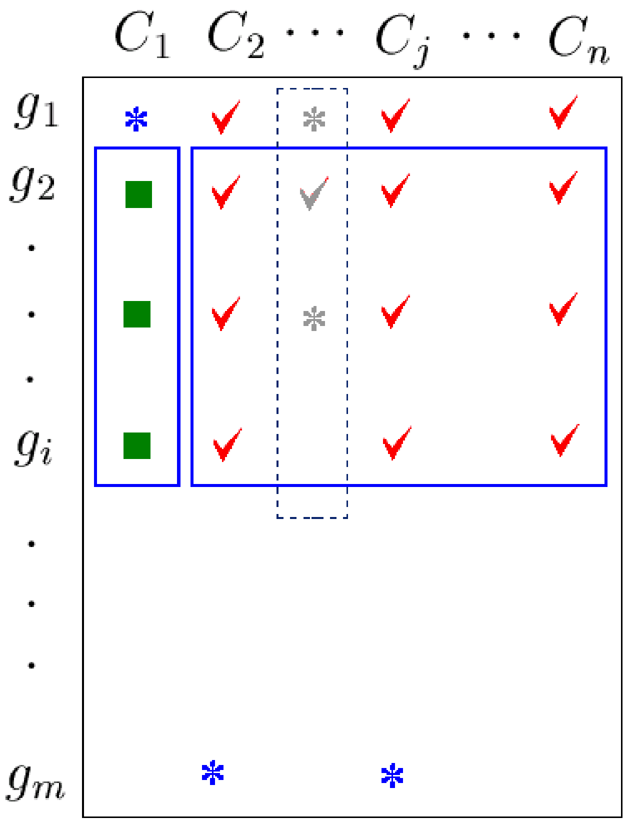
```
> airquality.imp.mean <- kNN(airquality[1:4],
+   k=5, metric=dist, numFun=mean)
> airquality.imp.tmean <- kNN(airquality[1:4],
+   k=5, numFun=trim_mean)
```




```
> airquality.imp.mean <- kNN(airquality[1:4], k=5, metric=dist, numFun=mean)
Warning messages:
1: In `[<-.data.table`(`*tmp*`, indexNA2s[, variable[j]], variable[j],  :
  Coerced 'double' RHS to 'integer' to match the column's type; may have trur
```

(A2) Regression Methods

- Using fitted regression values to replace missing values.
- The model must be chosen so that it does not yields invalid fitted values. e.g., negative values.
- This technique might be more accurate than simply substituting a measure of central tendency, since the imputed value is based on other input variables.



Regression

Model:

$$C_1 = \beta_0 + \sum_{j \in \mathrm{C}} \beta_j C_j$$

C: Observed C_i's without missing values

Imputation:

$$\widehat{C_1(g_1)} = \hat{\beta}_0 + \sum_{j \in \mathrm{C}} \hat{\beta}_j C_j(g_1)$$

regressionImp {VIM}: Regression Imputation

Description

Impute missing values based on a regression model.

Usage

```
regressionImp(formula, data, family = "AUTO", robust = FALSE,
  imp_var = TRUE, imp_suffix = "imp", mod_cat = FALSE)
```

```
> airquality.imp.lm <- regressionImp(Ozone ~ Wind + Temp, data=airquality)
Error in regressionImp_work(formula = formula, data = data, family = family,  :
  找不到物件 'nLev'
>
> data(sleep)
> summary(sleep)
    BodyWgt            BrainWgt            NonD            Dream            Sleep
 Min.   :   0.005   Min.   :   0.14   Min.   : 2.100   Min.   :0.000   Min.   : 2.60
 1st Qu.:   0.600   1st Qu.:   4.25   1st Qu.: 6.250   1st Qu.:0.900   1st Qu.: 8.05
 Median :   3.342   Median :  17.25   Median : 8.350   Median :1.800   Median :10.45
 Mean   : 198.790   Mean   : 283.13   Mean   : 8.673   Mean   :1.972   Mean   :10.53
 3rd Qu.:  48.203   3rd Qu.: 166.00   3rd Qu.:11.000   3rd Qu.:2.550   3rd Qu.:13.20
 Max.   :6654.000   Max.   :5712.00   Max.   :17.900   Max.   :6.600   Max.   :19.90
                                      NA's   :14       NA's   :12      NA's   :4
      Span              Gest             Pred            Exp             Danger
 Min.   :  2.000   Min.   : 12.00   Min.   :1.000   Min.   :1.000   Min.   :1.000
 1st Qu.:  6.625   1st Qu.: 35.75   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000
 Median : 15.100   Median : 79.00   Median :3.000   Median :2.000   Median :2.000
 Mean   : 19.878   Mean   :142.35   Mean   :2.871   Mean   :2.419   Mean   :2.613
 3rd Qu.: 27.750   3rd Qu.:207.50   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :100.000   Max.   :645.00   Max.   :5.000   Max.   :5.000   Max.   :5.000
 NA's   :4         NA's   :4
```

regressionImp {VIM}: Regression Imputation

```
> sleep.imp.lm <- regressionImp(Dream + NonD ~ BodyWgt + BrainWgt, data=sleep)
> summary(sleep.imp.lm)
    BodyWgt            BrainWgt            NonD              Dream             Sleep
 Min.   :   0.005   Min.   :   0.14   Min.   :-11.733   Min.   :-0.6897   Min.   : 2.60
 1st Qu.:   0.600   1st Qu.:   4.25   1st Qu.:  6.525   1st Qu.: 1.0000   1st Qu.: 8.05
 Median :   3.342   Median :  17.25   Median :  8.500   Median : 1.9312   Median :10.45
 Mean   : 198.790   Mean   : 283.13   Mean   :  8.335   Mean   : 1.9326   Mean   :10.53
 3rd Qu.:  48.203   3rd Qu.: 166.00   3rd Qu.: 10.550   3rd Qu.: 2.2750   3rd Qu.:13.20
 Max.   :6654.000   Max.   :5712.00   Max.   : 17.900   Max.   : 6.6000   Max.   :19.90
                                                                          NA's   :4
      Span              Gest             Pred            Exp            Danger
 Min.   :  2.000   Min.   : 12.00   Min.   :1.000   Min.   :1.000   Min.   :1.000
 1st Qu.:  6.625   1st Qu.: 35.75   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000
 Median : 15.100   Median : 79.00   Median :3.000   Median :2.000   Median :2.000
 Mean   : 19.878   Mean   :142.35   Mean   :2.871   Mean   :2.419   Mean   :2.613
 3rd Qu.: 27.750   3rd Qu.:207.50   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :100.000   Max.   :645.00   Max.   :5.000   Max.   :5.000   Max.   :5.000
 NA's   :4         NA's   :4
 Dream_imp         NonD_imp
 Mode :logical    Mode :logical
 FALSE:50         FALSE:48
 TRUE :12         TRUE :14
 NA's :0          NA's :0
```

Which Imputation Method?

- KNN is the most widely-used.

- Characteristics of data that may affect choice of imputation method:
  - dimensionality
  - percentage of values missing
  - experimental design (time series, case/control, etc.)
  - patterns of correlation in data

- Suggestion!!
  - add (**same percentage**) artificial missing values to your (**complete cases**) data set
  - impute them with various methods
  - see which is best (since you know the real value)

Classical (Numerical) Data Table

*j*th variables

UID	alpha0	alpha7	alpha14	alpha21	alpha28	alpha35	alpha42
YAR007C	-0.48	-0.42	0.87	0.92	0.67	-0.18	-0.35
YBL035C	-0.39	-0.58	1.08	1.21	0.52	-0.33	[illegible]
YBR023C	0.87	0.25	-0.17	0.18	-0.13	-0.44	-0.13
YBR067C	1.57	1.03	1.22	0.31	0.16	-0.49	-1.02
YBR088C	-1.15	-0.86	1.21	1.62	1.12	0.16	-0.44
YBR278W	0.04	-0.12	0.31	0.16	0.17	-0.06	0.08
YCL055W	2.95	0.45	-0.4	-0.66	-0.59	-0.38	-0.76
YDL003W	-1.22	-0.74	1.34	1.5	0.63	0.29	-0.55
YDL055C	-0.73	-1.06	-0.79	-0.02	0.16	0.44	0.03
YDL102W	-0.58	-0.4	0.13	0.58	-0.09	0.02	-0.45
YDL164C	-0.5	-0.42	0.66	1.05	0.68	0.06	0.01
YDL197C	-0.86	-0.29	0.42	0.46	0.3	0.1	-0.63
YDL227C	-0.16	0.2877	0.17	-0.28	-0.02	-0.55	-0.04
YDR052C	-0.36	-0.03	-0.03	-0.08	-0.23	-0.25	-0.21
YDR097C	-0.72	-0.85	0.54	1.04	0.84	0.24	-0.64
YDR113C	-0.78	-0.52	0.26	0.2	0.48	0.48	0.27
YDR309C	0.6	-0.55	0.41	0.45	0.18	-0.66	-1.02
YDR356W	-0.2	-0.67	0.13	0.1	0.38	0.44	0.05
YER001W	-2.29	-0.635739	0.77	1.6	0.53	0.55	-0.38
YER070W	-1.46	-0.76	1.08	1.5	0.74	0.47	-0.7
YER095W	-0.57	0.42	1.03	1.35	0.64	0.42	-0.4
YGL163C	-0.11	0.13	0.41	0.6	0.23	0.31	0.19
YGL225W	-1.08	-0.99	-0.16	0.2	0.61	0.37	0.1
YGR109C	-1.79	0.9449	2.13	1.75	0.23	0.15	-0.66

subjects

*i*th sample

transformation
for each row

transformation
for each column

transformation
for both rows
and columns

Why Data Transformations?

- Many statistical procedures make two assumptions that are relevant to data transformation:
  - (a) the variables (or their error terms) are **normally distributed**.
  - (b) homoscedasticity or **homogeneity of variance**, meaning that the variance of the variable remains constant over the observed range of some other variable.

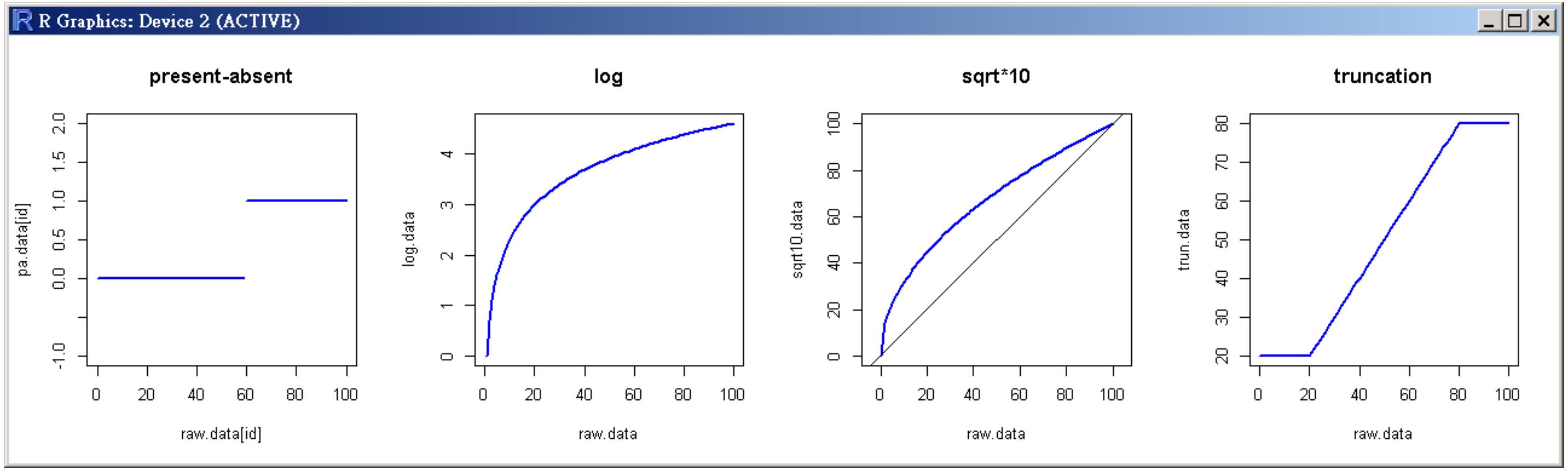
- In regression analyses the assumption (b) is that the variance around the regression line is constant across the entire observed range of data.
- In ANOVA analyses, the assumption (b) is that the variance in one cell is not significantly different from that of other cells.

- In some cases, transforming the data will make it **fit the assumptions** better.

Common Transformations (1/3)

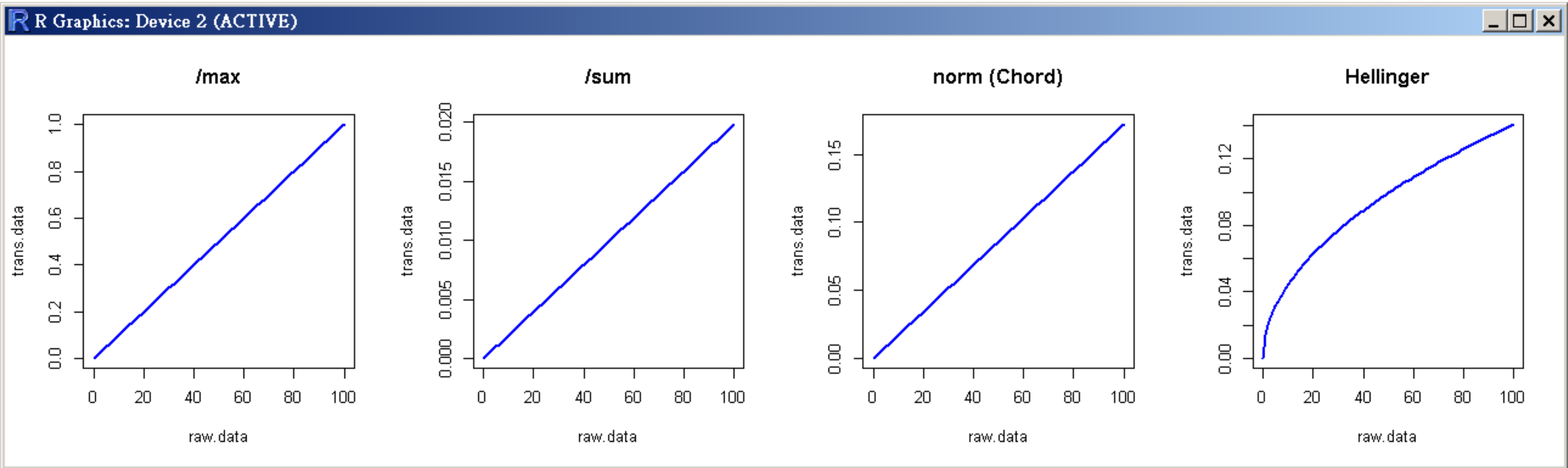
```
> par(mfrow=c(1,4))
> raw.data <- 0:100
> pa.data <- ifelse(raw.data >= 60, 1, 0)
> id <- which(pa.data==1)
> plot(raw.data[id], pa.data[id], main="present-absent",
+ type="l", lwd=2, col="blue", ylim=c(-1, 2), xlim=c(0, 100))
> points(raw.data[-id], pa.data[-id], type="l", lwd=2, col="blue")
>
> log.data <- log(raw.data)
> plot(raw.data, log.data, main="log", type="l", lwd=2, col="blue")
>
> sqrt10.data <- sqrt(raw.data)*10
> plot(raw.data, sqrt10.data, main="sqrt*10", type="l", lwd=2, col="blue", asp=1)
> abline(a=0, b=1)
>
> trun.data <- ifelse(raw.data >= 80, 80, ifelse(raw.data < 20, 20, raw.data))
> plot(raw.data, trun.data, main="truncation", type="l", lwd=2, col="blue")
```

NOTE:
```
apply(raw.data.matrix, 2, log)
apply(raw.data.matrix, 2, function(x) sqrt(x)*10)
apply(raw.data.matrix, 2, myfun)
```



Common Transformations (2/3)

```
> par(mfrow=c(1,4))
> raw.data <- 0:100
> trans.data <- raw.data/max(raw.data)
> plot(raw.data, trans.data, main="/max", type="l", lwd=2, col="blue")
>
> trans.data <- raw.data/sum(raw.data)  #Species profile transformation
> plot(raw.data, trans.data, main="/sum", type="l", lwd=2, col="blue")
>
> trans.data <- raw.data/sqrt(sum(raw.data^2)) #length of 1, Chord transformation
> plot(raw.data, trans.data, main="norm (Chord)", type="l", lwd=2, col="blue")
>
> trans.data <- sqrt(raw.data/sum(raw.data)) #Hellinger transformation
> plot(raw.data, trans.data, main="Hellinger", type="l", lwd=2, col="blue")
```



Other Transformations for community composition data: Chi-square distance transformation, Chi-square metric transformation

範例: Software Inspection Data

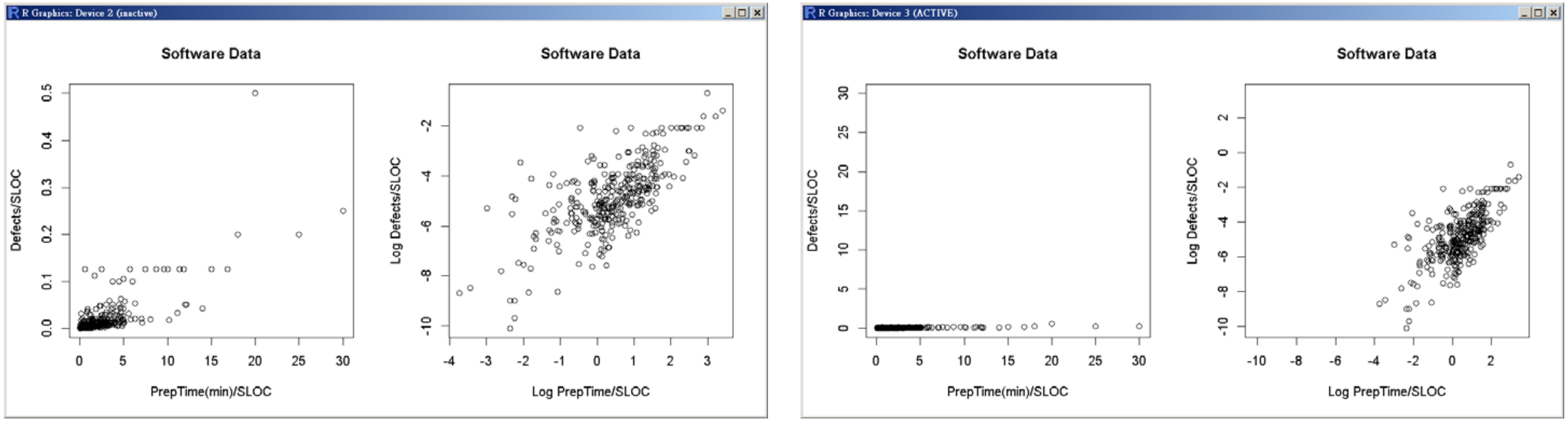
- The data were collected in response to efforts for process improvement in software testing by code inspection.
- First they look for inconsistencies, logical errors, etc., and decide what they perceive as defects. The defect types include compatibility, design, human-factors, standards, and others.
- The variables are normalized by the size of the inspection (the number of pages or SLOC (single lines of code): the preparation time in minutes (**prepage, prepsloc**), the total work hours in minutes for the meeting (**mtgsloc**), and the number of defects found (**defpage**, **defsloc**).
- Interested in: understanding the relationship between the inspection time and the number of defects found.

```
> library('R.matlab')
> data <- readMat("software.mat")
> print(data)
...
> str(data)
List of 5
 $ prepsloc: num [1:426, 1] 0.485 0.54 0.54 0.311 0.438 ...
 $ defsloc : num [1:426, 1] 0.005 0.002 0.002 0.00328 0.00278 ...
 $ mtgsloc : num [1:426, 1] 0.075 0.06 0.06 0.2787 0.0417 ...
 $ prepage : num [1:491, 1] 6.15 1.47 1.47 5.06 5.06 ...
 $ defpage : num [1:491, 1] 0.0385 0.0267 0.0133 0.0128 0.0385 ...
```

Log Transformations (1/3)

- The data are skewed, and the relationship between the variables is difficult to understand.
- We apply a log transform to both variables.
- In any application of EDA, the analyst should go back to the subject area and consult domain experts to verify and help interpret the results.

```
par(mfrow=c(1,2))
xylim <- range(data$prepsloc, data$defsloc)
plot(data$prepsloc, data$defsloc, xlab="PrepTime(min)/SLOC", ylab="Defects/SLOC",
main="Software Data", xlim=xylim, ylim=xylim)
logxylim <- range(log(data$prepsloc), log(data$defsloc))
plot(log(data$prepsloc), log(data$defsloc), xlab="Log PrepTime/SLOC",
ylab="Log Defects/SLOC", main="Software Data", xlim=logxylim, ylim=logxylim)
```



Box-Cox Transformations (1/3)

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0; \\ \log y, & \text{if } \lambda = 0. \end{cases}$$

Box and Cox(1964)

$$y(\boldsymbol{\lambda}) = \begin{cases} \frac{(y+\lambda_2)^{\lambda_1} - 1}{\lambda_1}, & \text{if } \lambda_1 \neq 0; \\ \log(y + \lambda_2), & \text{if } \lambda_1 = 0. \end{cases}$$

$\boldsymbol{\lambda} = (\lambda_1, \lambda_2)'$

choose λ_2 such that $y + \lambda_2 > 0$ for any y.

- The aim of the Box-Cox transformations is to ensure the usual assumptions for Linear Model hold.

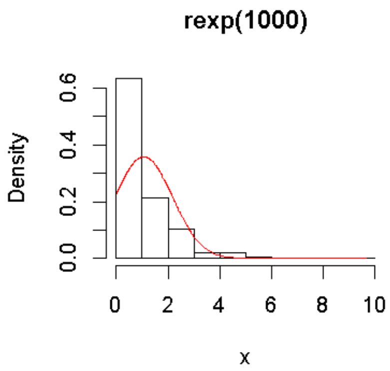
$$\boldsymbol{y} \sim \mathrm{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n)$$

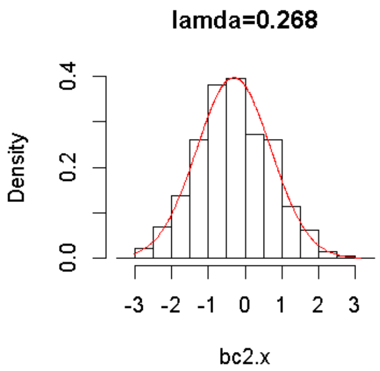
- Clearly not all data could be power-transformed to Normal. Draper and Cox (1969) studied this problem and conclude that even in cases that no power-transformation could bring the distribution to exactly normal, the usual estimates of lambda will lead to a distribution that satisfies certain restrictions on the first 4 moments, thus will be usually symmetric.

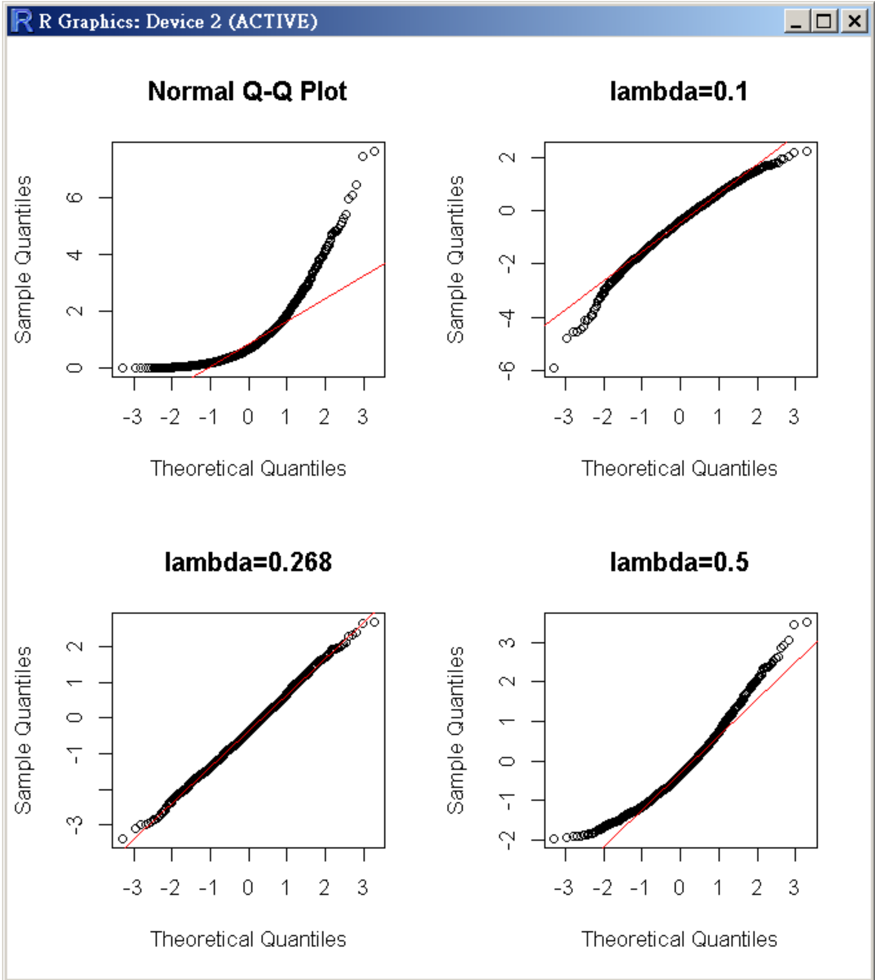
Source: Box-Cox Transformations: An Overview, Pengfei Li, Department of Statistics, University of Connecticut, Apr 11, 2005

Box-Cox Transformations (3/3)

```
> x <- rexp(1000)
> bc <- function(y, lambda){
+     (y^lambda -1)/lambda
+ }
> bc1.x <- bc(x, 0.1)
> bc2.x <- bc(x, 0.268)
> bc3.x <- bc(x, 0.5)
> par(mfrow=c(2, 2))
> qqnorm(x); qqline(x, col="red")
> qqnorm(bc1.x, main="lambda=0.1")
> qqline(bc1.x, col="red")
> qqnorm(bc2.x, main="lambda=0.268")
> qqline(bc2.x, col="red")
> qqnorm(bc3.x, main="lambda=0.5")
> qqline(bc3.x, col="red")
```







$$\left(\Phi^{-1}\left(\frac{i-0.5}{n}\right), x_{(i)}\right), \qquad \text{for } i = 1, 2, \ldots, n,$$

Source: Box-Cox Transformations: An Overview, Pengfei Li, Department of Statistics, University of Connecticut, Apr 11, 2005

Modified Box-Cox Transformations

Manly(1971)

$$y(\lambda) = \begin{cases} \frac{e^{\lambda y}-1}{\lambda}, & \text{if } \lambda \neq 0; \\ y, & \text{if } \lambda = 0. \end{cases}$$

Negative y's could be allowed. The transformation was reported to be successful in transform unimodal skewed distribution into normal distribution, but is not quite useful for **bimodal** or **U-shaped distribution**.

John and Draper(1980) "Modulus Transformation"

$$y(\lambda) = \begin{cases} \text{Sign}(y)\frac{(|y|+1)^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0; \\ \text{Sign}(y)\log(|y|+1), & \text{if } \lambda = 0, \end{cases}$$

$$\text{Sign}(y) = \begin{cases} 1, & \text{if } y \geq 0; \\ -1, & \text{if } y < 0. \end{cases}$$

Bickel and Doksum(1981)

$$y(\lambda) = \frac{|y|^{\lambda}\text{Sign}(y) - 1}{\lambda}, \quad \text{for } \lambda > 0,$$

Yeo and Johnson(2000)

$$y(\lambda) = \begin{cases} \frac{(y+1)^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0,\ y \geq 0; \\ \log(y+1), & \text{if } \lambda = 0,\ y \geq 0; \\ \frac{(1-y)^{2-\lambda}-1}{\lambda-2}, & \text{if } \lambda \neq 2,\ y < 0; \\ -\log(1-y), & \text{if } \lambda = 2,\ y < 0. \end{cases}$$

Source: Box-Cox Transformations: An Overview, Pengfei Li, Department of Statistics, University of Connecticut, Apr 11, 2005

Standardization

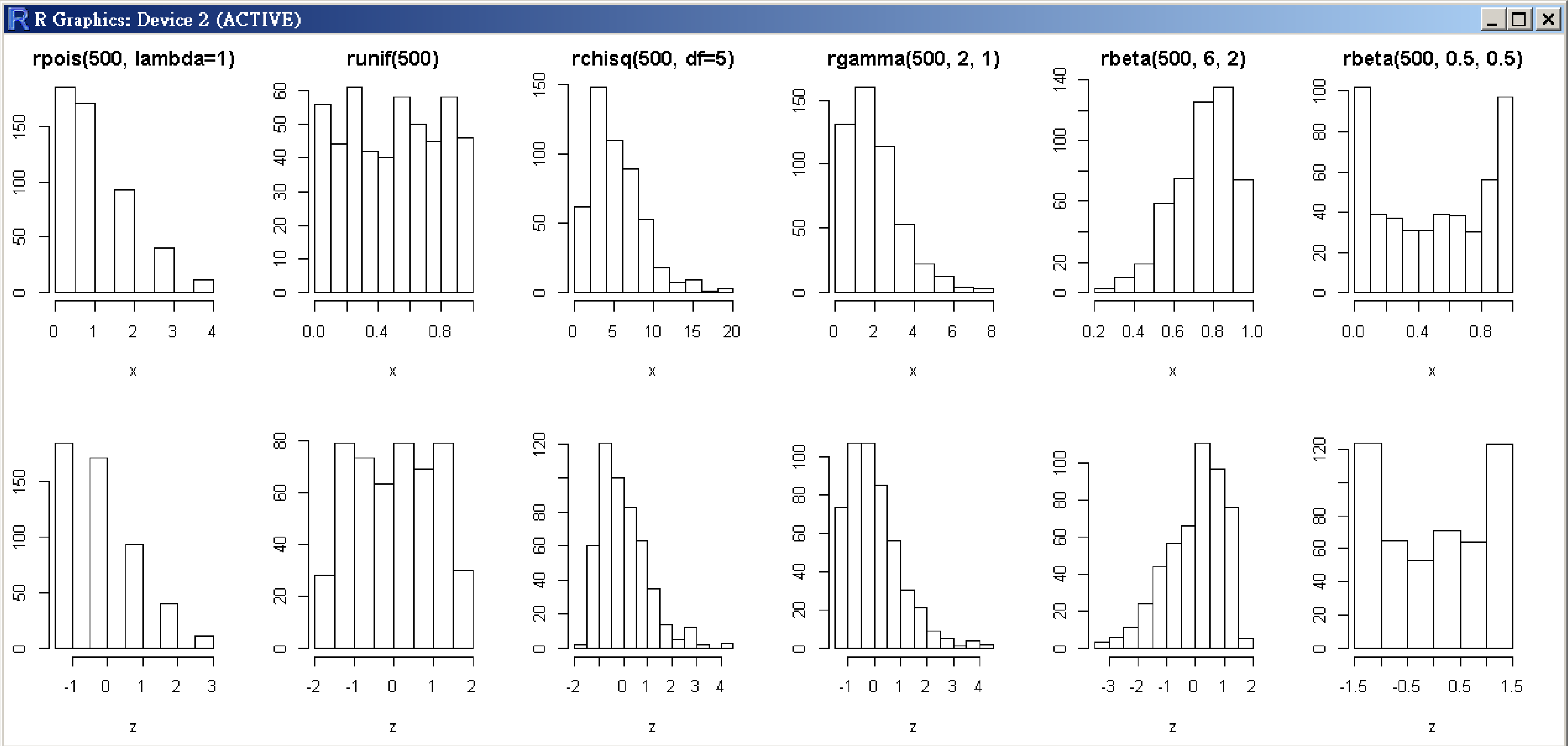
- Standardization, $z = (x\text{-}x.bar)/s$, (called z-score): the new variate z will have a mean of zero and a variance of one. (also called centering and scaling.)

- If the variables are measurements along a **different scale** or if the standard deviations for the variables are different from one another, then one variable might **dominate** the distance (or some other similar calculation) used in the analysis:
- Standardization is useful for comparing variables expressed in different units.

- In some multivariate contexts, the transformations may be applied to each variable separately.
  - **Standardization makes no difference to the shape of a distribution.**

Standardization

- Standardization makes no difference to the shape of a distribution.




```
x <- rpois(500, lambda=1)
hist(x, main="rpois(500, lambda=1)"); z <- scale(x); hist(z, main="")
```

Standardization

airquality {datasets}

New York Air Quality Measurements: Daily air quality measurements in New York, May to September 1973.

A data frame with 154 observations on 6 variables.

[1] Ozone: Ozone (ppb)

[2] Solar.R: Solar R (lang)

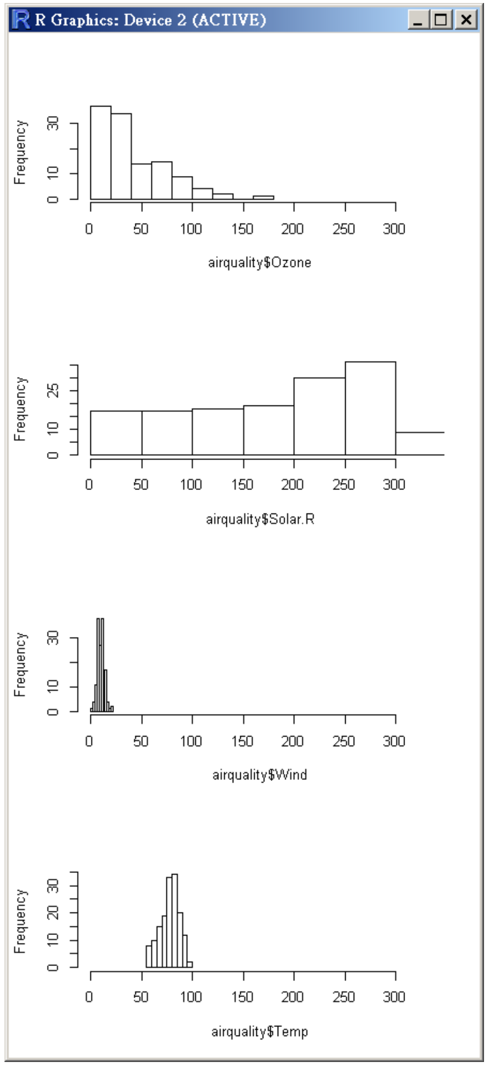
[3] Wind: Wind (mph)

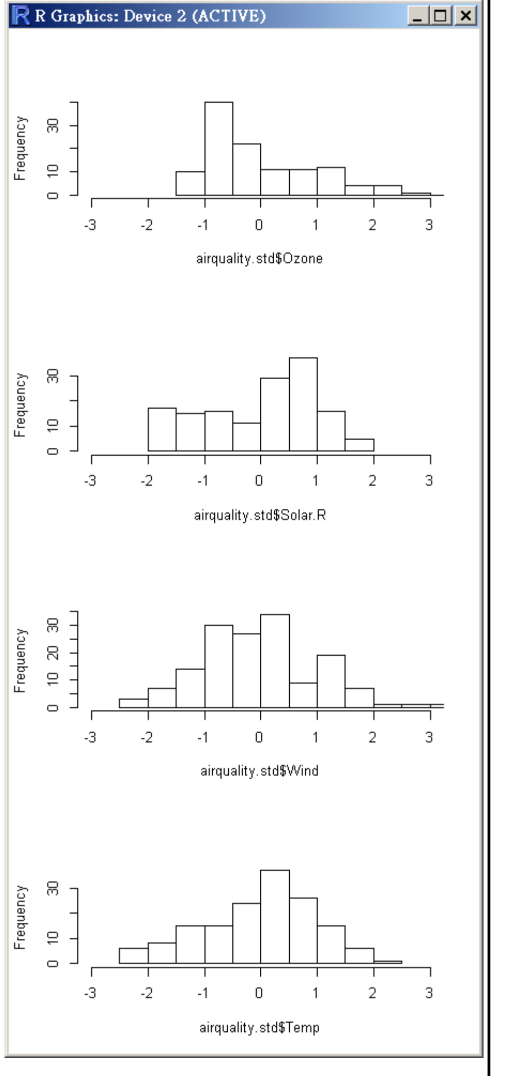
[4] Temp: Temperature (degrees F)

[5] Month: Month (1--12)

[6] Day: Day of month (1--31)

```
> head(airquality )
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> r <- range(airquality[,1:4], na.rm = T)
> hist(airquality$Ozone , xlim = r)
> hist(airquality$Solar.R, xlim = r)
> hist(airquality$Wind, xlim = r)
> hist(airquality$Temp, xlim = r)
>
> airquality.std <- as.data.frame(
apply(airquality, 2, scale))
> r.std <- c(-3, 3)
> hist(airquality.std$Ozone, xlim = r.std)
> hist(airquality.std$Solar.R, xlim = r.std)
> hist(airquality.std$Wind, xlim = r.std)
> hist(airquality.std$Temp, xlim = r.std)
```





Which Transformation?

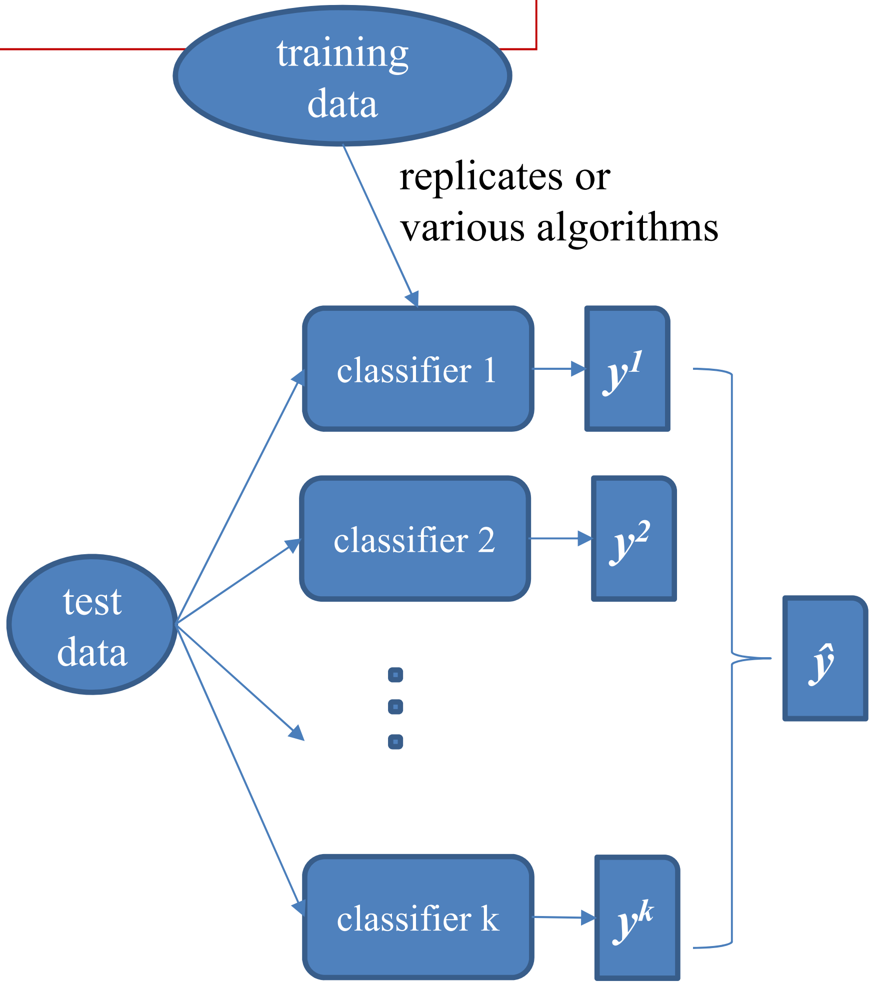
- Use a transformation that other researchers **commonly use in your field**.
- Remember that your data don't have to be perfectly normal and homoscedastic; parametric tests aren't extremely **sensitive** to deviations from their assumptions.
- It is also important that you decide which transformation to use before you do the statistical test. Trying different transformations until you find one that gives you a significant result **is cheating**. (?)
- If you have a **large** number of observations, compare the effects of different transformations on the normality and the homoscedasticity of the variable.
- If you have a **small** number of observations, you may not be able to see much effect of the transformations on the normality and homoscedasticity; in that case, you should use whatever transformation people in your field routinely use for your variable.

http://www.biostathandbook.com/transformation.html

Why Ensemble Learning?

```
prediction.accuracy.rate <- function(no.classifier=1, accuracy.rate=0.5){
   c(no.classifiers=no.classifier,
     at.least.one.accuracy=1-(1-accuracy.rate)^no.classifier)
}
```

```
> prediction.accuracy.rate()
       no.classifiers at.least.one.accuracy
                  1.0                   0.5
> t(sapply(1:10, prediction.accuracy.rate))
      no.classifiers at.least.one.accuracy
 [1,]              1             0.5000000
 [2,]              2             0.7500000
 [3,]              3             0.8750000
 [4,]              4             0.9375000
 [5,]              5             0.9687500
 [6,]              6             0.9843750
 [7,]              7             0.9921875
 [8,]              8             0.9960938
 [9,]              9             0.9980469
[10,]             10             0.9990234
```



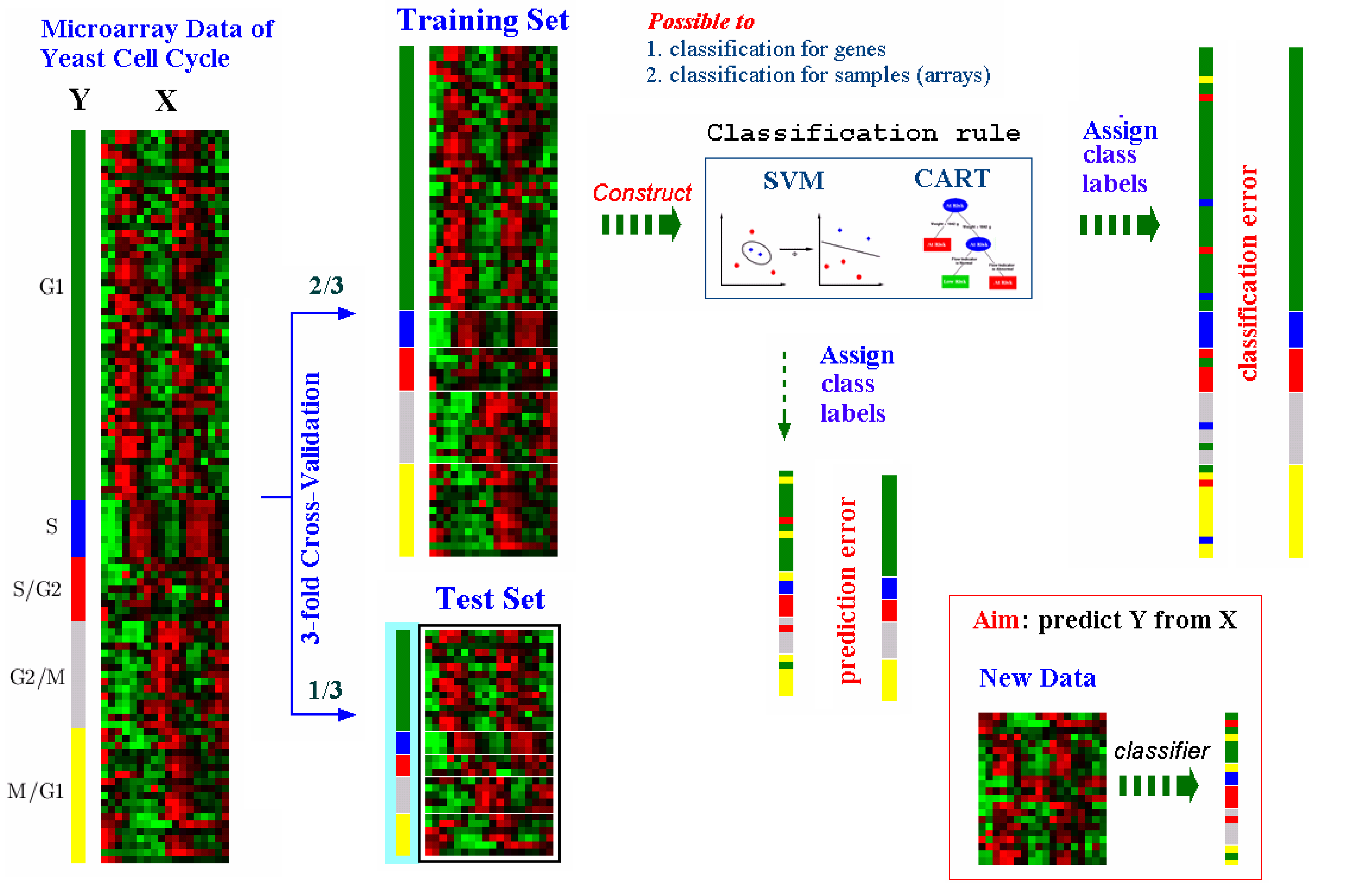
Why Resampling?

- Resampling is any of a variety of methods for:
  - Estimating the precision of sample statistics (medians, variances, percentiles) by using subsets of available data (**jackknifing**) or drawing randomly with replacement from a set of data points (**bootstrapping**).
  - Exchanging labels on data points when performing significance tests (**permutation tests**, **randomization tests**)
  - Validating models by using random subsets (bootstrapping, cross validation)

  https://en.wikipedia.org/wiki/Resampling_(statistics)

- This single sample method can serve as a **mini population**, from which repeated small samples are drawn with replacement over and over again.
- As well as saving time and money, bootstrapped samples can be **quite good approximations** for population parameters.

k-fold Cross-Validation Error Rates



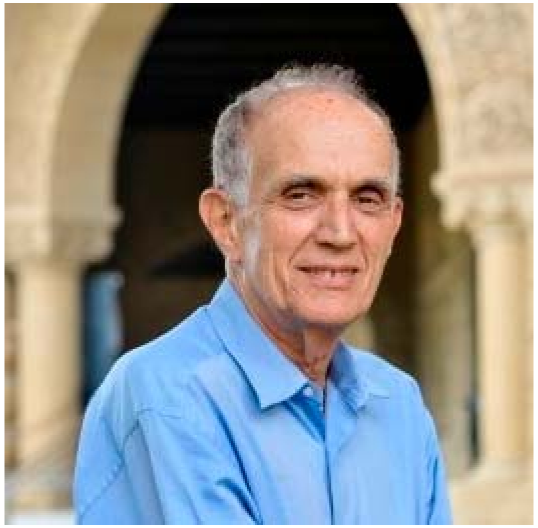
Split Data into Training Set and Test Set

```
> id <- sample(2, nrow(iris), replace = TRUE, prob = c(0.9, 0.1))
> id
  [1] 1 1 2 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
 [38] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
[112] 1 1 1 1 1 1 1 2 1 2 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 2 2 2 1 1
[149] 1 1
> train.data <- iris[id == 1, ]
> dim(train.data)
[1] 131   5
> test.data <- iris[id == 2, ]
> dim(test.data)
[1] 19  5
```

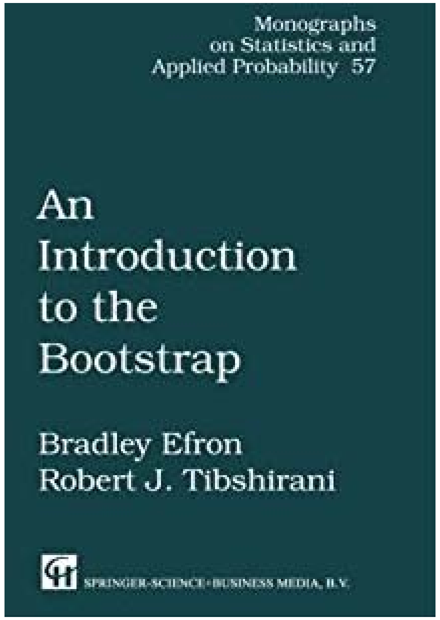
```
> id <- sample(nrow(iris), floor(nrow(iris) * 0.9))
> id
  [1]  39  27  96  33   4  98  12   3  32  48   2  22  18  24 126  93 140  85 110  60
 [21]  62  91 131  35 134 143  29 108 114  50  19  43  45  66  36  90 105  76 127  92
 [41]  68  57  65 147  69  41 130  82  31  20  51  17 149  61 107  70 139   5 115  72
 [61]  78 118 117  38  15  74 120 111 106  11 104  67  13  21 133  42  87 121 122  40
 [81]  84 135 123  77  83  97  52 116  55  88 142  16   7  49 125 112  34  10  56  26
[101]  99  63  37  46 144   9 141  59 138  80 101 132 129 113  73  30  44 136 119  79
[121]  95  64 109 148  28  14  86 150 137  81  94  75 128 102 124
> train.data <- iris[id, ]
> dim(train.data)
[1] 135   5
> test.data <- iris[-id, ]
> dim(test.data)
[1] 15  5
```

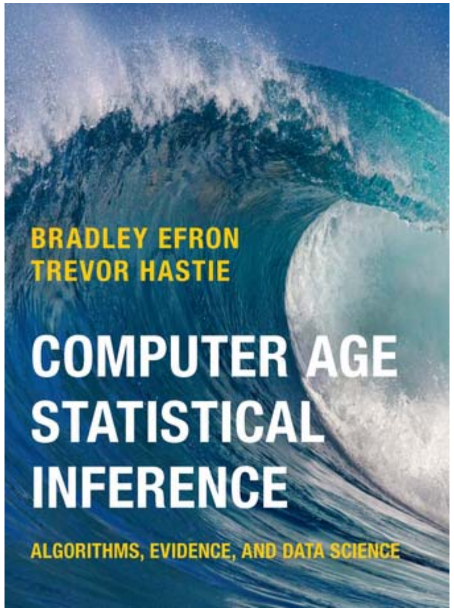
Bootstrap Methods

- Bootstrapping is a statistical method for estimating the sampling distribution of an estimator by sampling with replacement from the original sample, of the same size as the original sample.
- The name "bootstrapping" comes from the phrase: "To lift himself up by his bootstraps".
- This refers to something that is preposterous and impossible.
- Try as hard as you can, you cannot lift yourself into the air by tugging at pieces of leather on your boots.



Bradley Efron 1938~
Department of Statistics
Stanford University





Bootstrapping

Real World

Unknown probability distribution

Observed random sample

$$P \longrightarrow X = (X_1, \ldots, X_n)$$

$$\downarrow$$

$$\hat{\theta} = s(X)$$

Statistic of interest

sampling with replacement

Bootstrap World

Empirical distribution

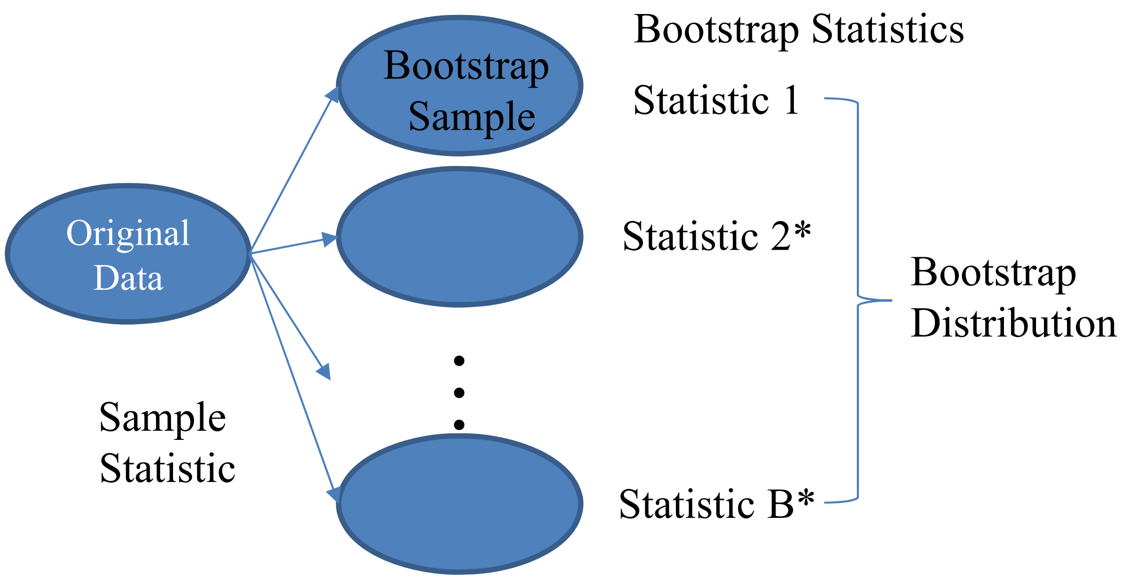
Bootstrap sample

$$\hat{P} \longrightarrow X^* = (X_1^*, \ldots, X_n^*)$$

$$\downarrow$$

$$\hat{\theta}^* = s(X^*)$$

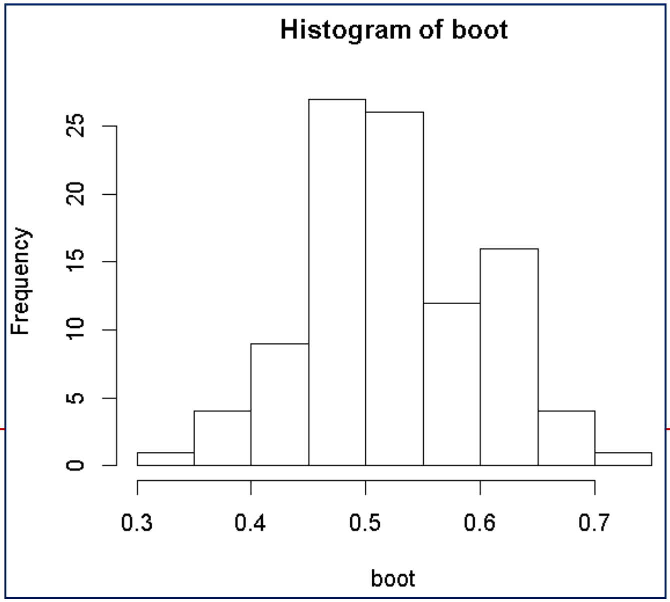
Bootstrap replication



- Types of bootstrap scheme: Case resampling, Bayesian bootstrap, Smooth bootstrap, Parametric bootstrap, Resampling residuals, Gaussian process regression bootstrap, Wild bootstrap, Block bootstrap.
- An empirical bootstrap sample is drawn from observations.
- A parametric bootstrap sample is drawn from a parameterized distribution (e.g. a normal distribution).

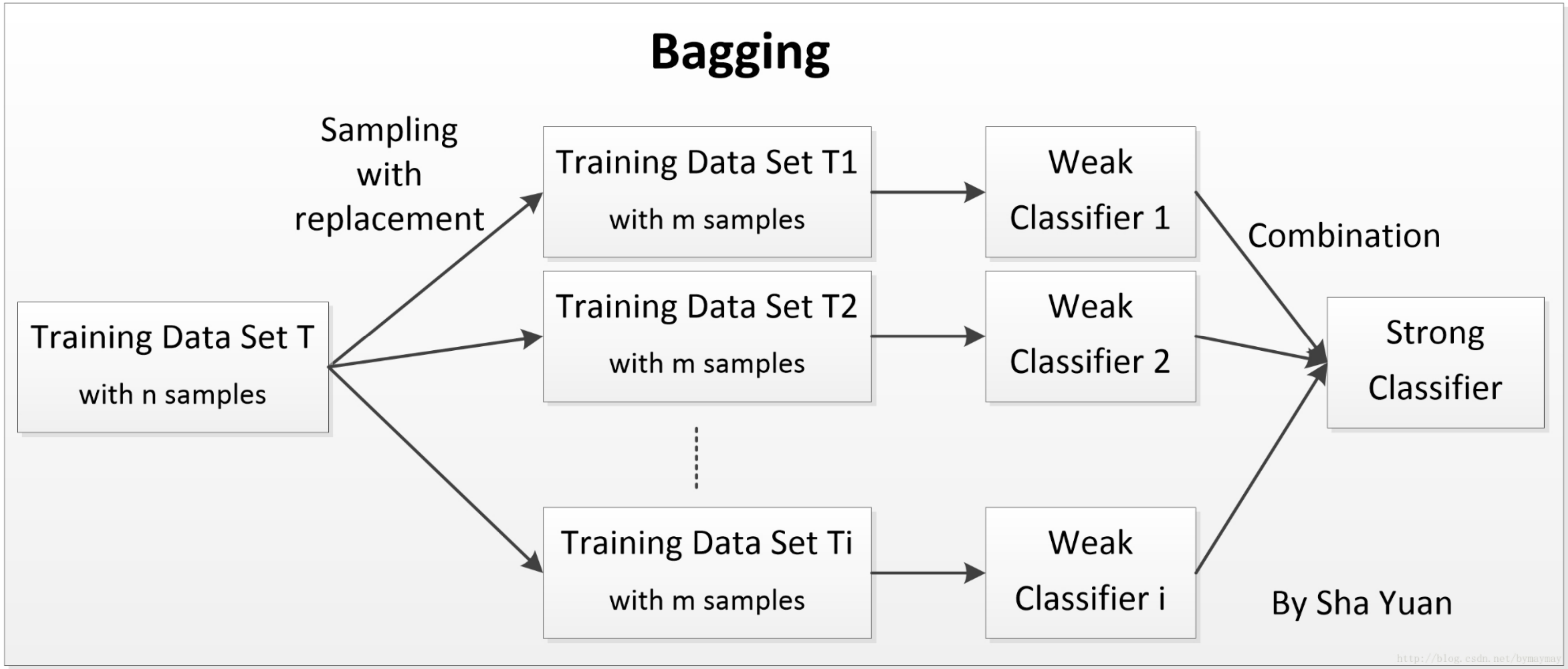
Example: Bootstrap Estimate the Coefficient of Variation

$$\mathrm{CV} = \sqrt{\mathrm{Var}} / \overline{x}$$




```
> set.seed(12345)
> x <- runif(30)
> CV <- function(x) sqrt(var(x))/mean(x)
> CV(x)
[1] 0.5380304
> CV(sample(x, replace=T)) # a single bootstrap sample
[1] 0.5459389
> boot <- replicate(n=100, expr=CV(sample(x, replace=T)))
> boot
  [1] 0.5044811 0.5286011 0.4634611 0.5605438 0.4835447 0.5374531 0.4857342 0.4342565
 ...
[89] 0.5297020 0.5121274 0.4938053 0.5479498 0.5262306 0.6095145 0.5322045 0.6069263
 [97] 0.5374840 0.4921430 0.4674226 0.4573680
> mean(boot)
[1] 0.5251909
> var(boot)
[1] 0.006107636
> hist(boot)
```

Bagging: Bootstrap Aggregating



http://blog.csdn.net/bymaymay/article/details/77824574

- Breiman, L. (1996). Bagging predictors, Machine Learning, Vol. 26, pp. 123-140.
- Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm, Proceedings of the Thirteenth International Conference, Machine Learning.

Boosting

$$W_{x_i}^{(1)} = N^{-1} \text{ for all } x_i.$$

a bootstrap sample $\mathcal{L}_t^{(B)}$

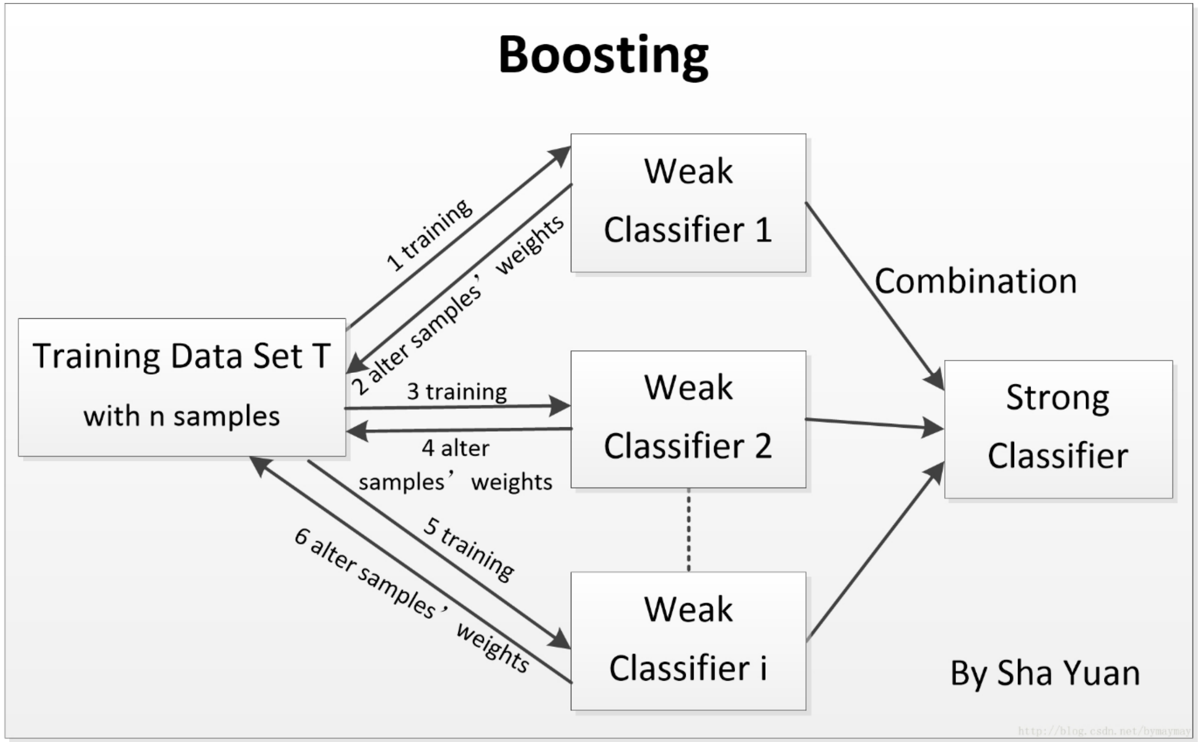
error ϵ_t of classifier $\varphi_t(\mathbf{x})$

$$\epsilon_t = \sum_{\{i:\varphi_t(x_i)\neq y_i\}} W_{x_i}^{(t)}.$$

$$\beta_t = (1-\epsilon_t)\epsilon_t^{-1}$$

$$W_{x_i}^{(t+1)} = \frac{W_{x_i}^{(t)}\beta_t^{d(i)}}{\sum_i W_{x_i}^{(t)}\beta_t^{d(i)}},$$

boosted classifier



http://blog.csdn.net/bymaymay/article/details/77824574

$d(i) = 1$ if ith case is classified incorrectly,
$d(i) = 0$, otherwise

$$\varphi_B(x_i) = arg\ max_j \sum_{t=1}^{T} \log \beta_t I[\varphi_t(x_i) = j]$$

Ad-Boost.M1 (Freund and Schapire, 1996)

Example: Apply rpart to Vehicle Data

```
> library(rpart); library(mlbench); library(adabag)
> data(Vehicle)
> dim(Vehicle)
[1] 846  19
> head(Vehicle)
 Comp Circ D.Circ Rad.Ra Pr.Axis.Ra Max.L.Ra Scat.Ra Elong Pr.Axis.Rect Max.L.Rect Sc.Var.Maxis
1   95   48     83    178         72       10     162    42           20        159          176
...
  Sc.Var.maxis Ra.Gyr Skew.Maxis Skew.maxis Kurt.maxis Kurt.Maxis Holl.Ra Class
1          379    184         70          6         16        187     197   van
...
6          957    264         85          5          9        181     183   bus
> table(Vehicle$Class)
 bus opel saab  van
 218  212  217  199
```

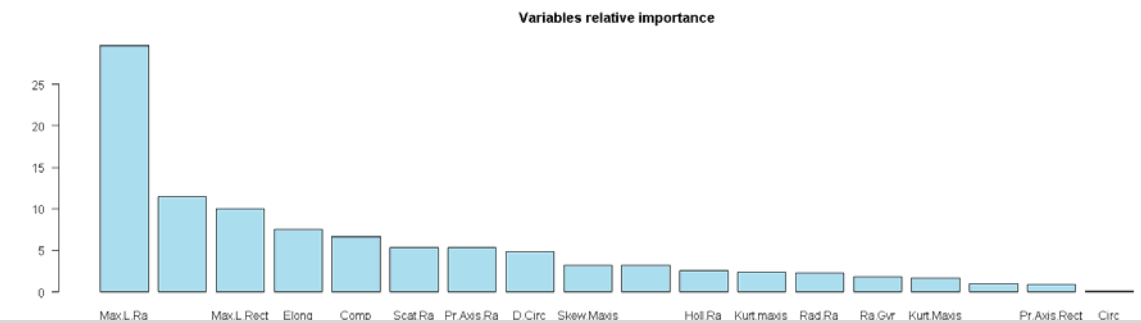
```
> n <- nrow(Vehicle)
> sub <- sample(1:n, 2*n/3)
> Vehicle.train <- Vehicle[sub, ]
> Vehicle.test <- Vehicle[-sub, ]
```

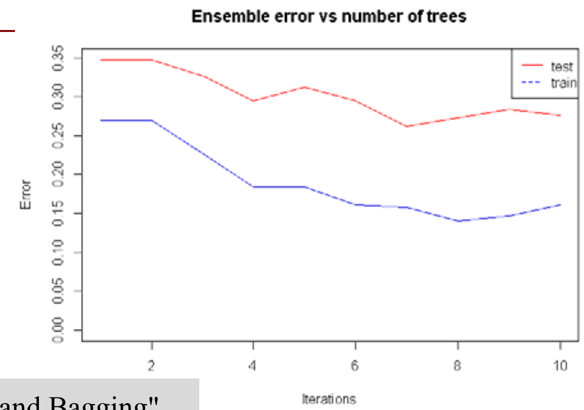
```
> mfinal <- 10 #Defaults to mfinal=100 iterations
> maxdepth <- 5
> Vehicle.rpart <- rpart(Class ~ ., data = Vehicle.train, maxdepth = maxdepth)
> Vehicle.rpart.pred <- predict(Vehicle.rpart, newdata = Vehicle.test, type = "class")
> (tb <- table(Vehicle.rpart.pred, Observed.Class=Vehicle.test$Class))
                  Observed.Class
Vehicle.rpart.pred bus opel saab van
              bus   69   10    6   2
              opel   1   25   13   3
              saab   1   34   37   8
              van    2    7    5  59
> (error.rpart <- 1 - (sum(diag(tb)) / sum(tb)))
[1] 0.3262411
```

adabag: An R Package for Classification with Boosting and Bagging

```
> library(adabag)
> Vehicle.adaboost <- boosting(Class ~., data = Vehicle.train, mfinal = mfinal,
+                              control = rpart.control(maxdepth=maxdepth))
> Vehicle.adaboost.pred <- predict.boosting(Vehicle.adaboost, newdata = Vehicle.test)
> Vehicle.adaboost.pred$confusion
               Observed Class
Predicted Class bus opel saab van
           bus   69    2    3   1
           opel   1   30   16   4
           saab   1   38   39   1
           van    2    6    3  66
> Vehicle.adaboost.pred$error
[1] 0.2765957
> importanceplot(Vehicle.adaboost)
>
> # comparing error evolution in training and test set
> evol.train <- errorevol(Vehicle.adaboost, newdata = Vehicle.train)
> evol.test <- errorevol(Vehicle.adaboost, newdata = Vehicle.test)
> plot.errorevol(evol.test, evol.train)
```

```
> sort(Vehicle.adaboost$importance, dec=T)[1:5]
  Max.L.Ra Sc.Var.maxis    Max.L.Rect
 29.623783    11.473254      9.956137
     Elong         Comp
  7.570798     6.656360
```

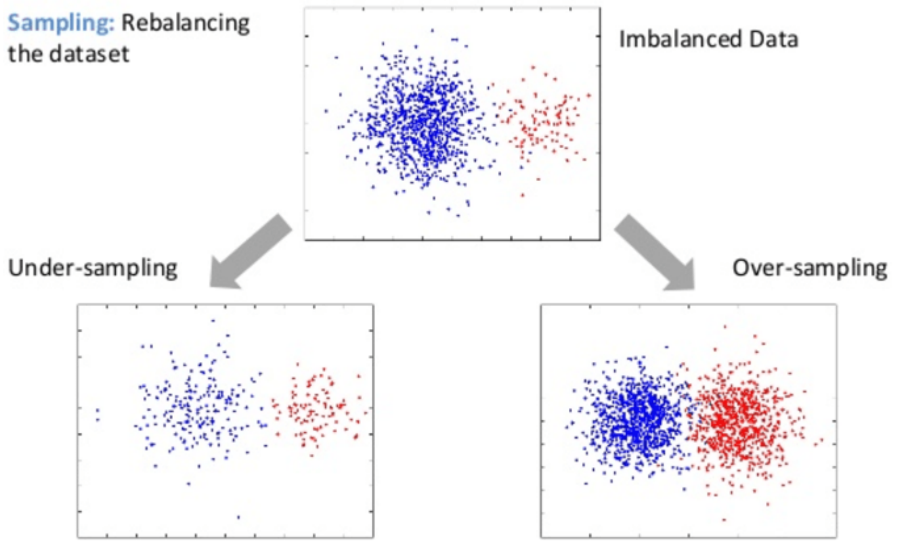




Alfaro, E., Gamez, M. and Garcia, N. (2013): "adabag: An R Package for Classification with Boosting and Bagging". Journal of Statistical Software, 54(2), 1–35.

The Imbalanced Data Problem

- A dataset is said to be unbalanced when the class of interest (minority class) is much rarer than normal behaviour (majority class).
- The cost of missing a minority class is typically much higher that missing a majority class. Most learning systems are not prepared to cope with unbalanced data and several techniques have been proposed.
- **Example**: 5% of the target class represents fraudulent transactions, 95% of the target class represents legitimate transactions.



http://www.srutisj.in/blog/research/statisticalmodeling/balancing-techniques-for-unbalanced-datasets-in-python-r/

unbalanced

Racing for Unbalanced Methods Selection

Re-balance or remove noisy instances in unbalanced datasets.

ubBalance {unbalanced}

Usage

```
ubBalance(X, Y, type="ubSMOTE", positive=1,
          percOver=200, percUnder=200,
          k=5, perc=50, method="percPos", w=NULL, verbose=FALSE)
```

Arguments

X: the input variables of the unbalanced dataset.
Y: the response variable of the unbalanced dataset.
type: the balancing technique to use (**ubOver, ubUnder, ubSMOTE, ubOSS, ubCNN, ubENN, ubNCL, ubTomek**).
positive: the majority class of the response variable.
percOver: parameter used in **ubSMOTE**
percUnder: parameter used in **ubSMOTE**
k: parameter used in **ubOver, ubSMOTE, ubCNN, ubENN, ubNCL**
perc: parameter used in **ubUnder**
method: parameter used in **ubUnder**
w: parameter used in **ubUnder**
verbose: print extra information (TRUE/FALSE)

ubSMOTE {unbalanced}: synthetic minority over-sampling technique

Usage

```
ubSMOTE(X, Y, perc.over = 200, k = 5, perc.under = 200, verbose = TRUE)
```

NOTE: **imbalance**: Preprocessing Algorithms for Imbalanced Datasets, Imbalanced Classification in R: **ROSE** (Random Over Sampling Examples) and **DMwR** (Data Mining with R).

The Balancing Technique

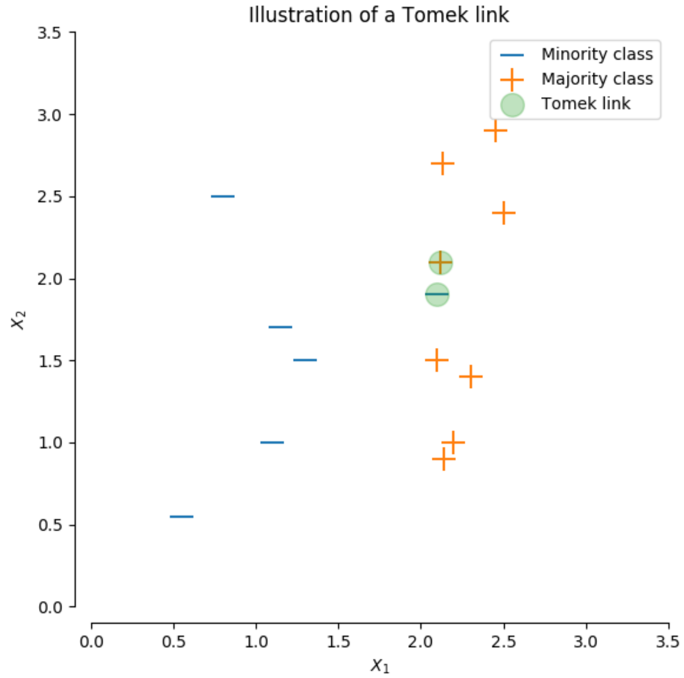
- **ubOver**: replicates randomly some instances from the minority class in order to obtain a final dataset with the same number of instances from the two classes.
- **ubUnder**: removes randomly some instances from the majority (negative) class and keeps all instances in the minority (positive) class in order to obtain a more balanced dataset.

- **ubCNN**: Condensed Nearest Neighbor selects the subset of instances that are able to correctly classifying the original datasets using a one-nearest neighbor rule.
- **ubENN**: Edited Nearest Neighbor removes any example whose class label differs from the class of at least two of its three nearest neighbors.
- **ubNCL**: Neighborhood Cleaning Rule modifies the Edited Nearest Neighbor method by increasing the role of data cleaning.
  - Firstly, NCL removes negatives examples which are misclassified by their 3-nearest neighbors.
  - Secondly, the neighbors of each positive examples are found and the ones belonging to the majority class are removed.

The Balancing Technique

- **ubTomek**: finds the points in the dataset that are tomek link using 1-NN and then removes only majority class instances that are tomek links.

x's nearest neighbor is y
y's nearest neighbor is x
x and y are different classes



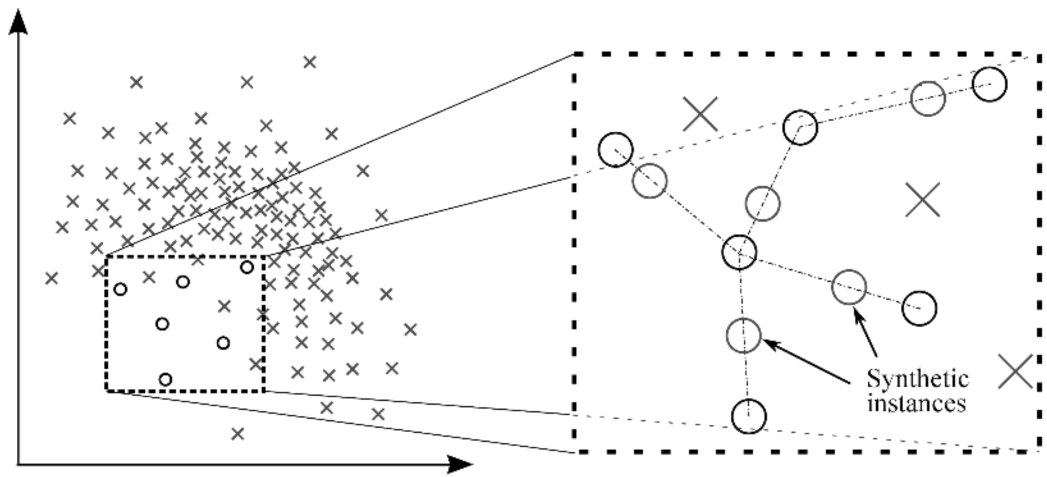
http://contrib.scikit-learn.org/imbalanced-learn/stable/auto_examples/under-sampling/plot_illustration_tomek_links.html

- **ubOSS**: One Side Selection is an undersampling method resulting from the application of Tomek links followed by the application of Condensed Nearest Neighbor.

The Balancing Technique

- **ubSMOTE**: synthetic minority over-sampling technique generates new examples by filling empty areas among the positive instances

N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, *Journal Of Artificial Intelligence Research*, Volume 16, pages 321-357, 2002.(由 NV Chawla 著作 - 2002 - 被引用 5161 次)



- **ubRacing**: the Racing algorithm for selecting the best technique to re-balance or remove noisy instances in unbalanced datasets.

Ionosphere dataset
ubIonosphere {unbalanced}

- The datasets is a modification of Ionosphere dataset contained in "mlbench" package.

```
> data(ubIonosphere)
> dim(ubIonosphere)
[1] 351  33
> head(ubIonosphere)
       V3       V4          V34 Class
1 0.99539 -0.05889 ... -0.45300     0
...
6 0.02337 -0.00592 ...  0.12011     1
> table(ubIonosphere$Class)
  0   1
225 126
```

```
> # install.packages("unbalanced")
> library(unbalanced)
> p <- ncol(ubIonosphere)
> y <- ubIonosphere$Class
> x <- ubIonosphere[ ,-p]
> data <- ubBalance(X=x, Y=y, type="ubOver", k=0)
> overData <- data.frame(data$X, Class=data$Y)
> table(overData$Class)
  0   1
225 225
> data <- ubBalance(X=x, Y=y, type="ubUnder", perc=50, method="percPos")
> underData <- data.frame(data$X, Class=data$Y)
> table(underData$Class)
  0   1
126 126
> bdata <- ubBalance(X=x, Y=y, type="ubSMOTE", percOver=300, percUnder=150, verbose=TRUE)
Proportion of positives after ubSMOTE : 47.06 % of 1071 observations
> str(bdata)
List of 3
 $ X    :'data.frame':  1071 obs. of  32 variables:
  ..$ V3 : num [1:1071] -0.787 1 1 0.5 1 ...
...
..$ V34: num [1:1071] -0.576 0.714 -0.243 0.174 -0.892 ...
 $ Y    : Factor w/ 2 levels "0","1": 2 1 1 1 1 2 1 2 1 2 ...
 $ id.rm: logi NA
> table(bdata$Y)
  0   1
567 504
```

K=0: sample with replacement from the minority class until we have the same number of instances in each class.
If K>0: sample with replacement from the minority class until we have k-times the orginal number of minority instances

`per.over/100` : number of new instances generated for each rare instance
`perc.under/100`: number of "normal" (majority class) instances that are randomly selected for each smoted observation.

Compare the Performances using SVM

```
> set.seed(12345)
> n <- nrow(ubIonosphere) # 351
> no.train <- floor(0.5*n) # 175, keep half for training and half for testing
> id <- sample(1:n, no.train)
> x.train  <- x[id, ]  # 175 x 32
> y.train <- y[id]
> x.test <- x[-id, ] # 176  32
> y.test <- y[-id]
>
> library(e1071)
> model1 <- svm(x.train, y.train)
> y.pred1 <- predict(model1, x.test)
> table(y.pred1, y.test)
       y.test
y.pred1   0   1
      0 113  10
      1   4  49
>
> # rebalance the training set before building a model
> balancedData <- ubBalance(X=x.train, Y=y.train, type="ubSMOTE",
                            percOver=200, percUnder=150)
> table(balancedData$Y)
  0   1
201 201
```

```
> model2 <- svm(balancedData$X, balancedData$Y)
> y.pred2 <- predict(model2, x.test)
> table(y.pred2, y.test)
       y.test
y.pred2   0   1
      0 112   8
      1   5  51
```

ubRacing {unbalanced}
Racing for Strategy Selection

Selecting the best technique to re-balance or remove noisy instances in unbalanced datasets.

```
ubRacing(formula, data, algo, positive=1, ncore=1, nFold=10, maxFold=10, maxExp=100,
         stat.test="friedman", metric="f1", ubConf, verbose=FALSE, ...)
```

Arguments

- **algo**: the classification algorithm to use with the mlr package.
- **positive**: label of the positive (minority) class.
- **ncore**: the number of core (parallel execution) to use in the Race.

```
> set.seed(1234)
> # load(url("http://www.ulb.ac.be/di/map/adalpozz/data/creditcard.Rdata"))
> load("creditcard.Rdata")
> str(creditcard)
'data.frame':   284807 obs. of  31 variables:
 $ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
 $ V1    : num  -1.36 1.192 -1.358 -0.966 -1.158 ...
...
 $ V28   : num  -0.0211 0.0147 -0.0598 0.0615 0.2152 ...
 $ Amount: num  149.62 2.69 378.66 123.5 69.99 ...
 $ Class : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
> table(creditcard$Class)

     0      1
284315    492
> # configuration of the sampling method used in the race
> ubConf <- list(percOver=200, percUnder=200, k=2, perc=50, method="percPos", w=NULL)
> # Race with 5 trees in the Random Forest
> results <- ubRacing(Class ~., creditcard, "randomForest",
+                      positive=1, metric="auc", ubConf=ubConf, ntree=5)
```

The function **ubRacing** compares the 8 unbalanced methods (ubUnder, ubOver, ubSMOTE, ubOSS, ubCNN, ubENN, ubNCL, ubTomek) against the unbalanced distribution.

Racing for Strategy Selection

```
Racing for unbalanced methods selection in 10 fold CV
Number of candidates.........................................9
Max number of folds in the CV...............................10
Max number of experiments..................................100
Statistical test...............................Friedman test

                            Markers:
                               x No test is performed.
                               - The test is performed and
                                 some candidates are discarded.
                               = The test is performed but
                                 no candidate is discarded.

+-+-----------+-----------+-----------+-----------+-----------+
| |       Fold|      Alive|       Best|  Mean best| Exp so far|
+-+-----------+-----------+-----------+-----------+-----------+
|x|          1|          9|          4|     0.9543|          9|
|=|          2|          9|          3|     0.9433|         18|
|-|          3|          3|          4|     0.9567|         27|
|-|          4|          2|          4|     0.9566|         30|
|=|          5|          2|          4|     0.9582|         32|
|=|          6|          2|          4|     0.9546|         34|
|=|          7|          2|          4|     0.9531|         36|
|=|          8|          2|          4|     0.9539|         38|
|=|          9|          2|          4|     0.9531|         40|
|=|         10|          2|          4|     0.9529|         42|
+-+-----------+-----------+-----------+-----------+-----------+
Selected candidate: ubSMOTE      metric: auc     mean value: 0.9529
```

Racing for Strategy Selection

```
> # Race using 4 cores and 500 trees (default)
> results <- ubRacing(Class ~., creditcard, "randomForest",
                      positive=1, metric="auc", ubConf=ubConf, ncore=4)
> library(e1071)
> results <- ubRacing(Class ~., creditcard, "svm",
                      positive=1, ubConf=ubConf)
> library(rpart)
> results <- ubRacing(Class ~., creditcard, "rpart",
                      positive=1, ubConf=ubConf)
```

```
> results
$best
[1] "ubSMOTE"

$avg
[1] 0.9529177

$sd
[1] 0.009049014

$N.test
[1] 42

$Gain
[1] 53

$Race
          unbal    ubOver   ubUnder   ubSMOTE     ubOSS     ubCNN     ubENN     ubNCL   ubTomek
 [1,] 0.8844582 0.9138946 0.9354739 0.9543104 0.8957273 0.9139340 0.9024656 0.9014143 0.9048642
 [2,] 0.9116642 0.9104928 0.9511485 0.9507221 0.9037491 0.9104840 0.9139047 0.9094542 0.9105558
 [3,] 0.8979478 0.9013642 0.9502417 0.9649361 0.9092505 0.9081796 0.9103668 0.9036617 0.9058917
 [4,]        NA        NA 0.9503782 0.9564226        NA        NA 0.8999928        NA        NA
 [5,]        NA        NA 0.9537802 0.9647722        NA        NA        NA        NA        NA
 [6,]        NA        NA 0.9494913 0.9362763        NA        NA        NA        NA        NA
 [7,]        NA        NA 0.9411979 0.9440379        NA        NA        NA        NA        NA
 [8,]        NA        NA 0.9576971 0.9594249        NA        NA        NA        NA        NA
 [9,]        NA        NA 0.9530119 0.9473722        NA        NA        NA        NA        NA
[10,]        NA        NA 0.9633438 0.9509024        NA        NA        NA        NA        NA
```