Arthur Charpentier

arthur.charpentier@gmail.com

http://freakonometrics.hypotheses.org/

École Doctorale, Université de Rennes 1, March 2015

Data Science, an Overview of Classification Techniques

"*An expert is a man who has made all the mistakes which can be made, in a narrow field* " N. Bohr

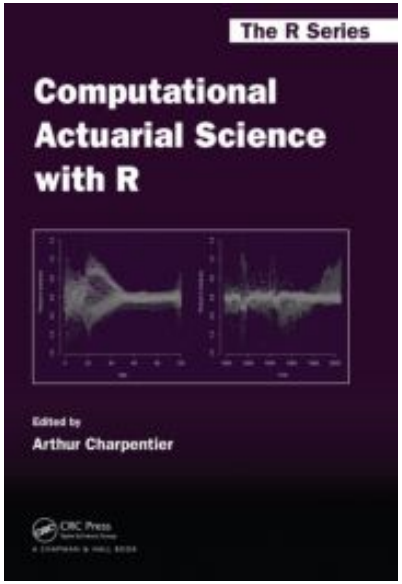
Arthur Charpentier

arthur.charpentier@gmail.com

http://freakonometrics.hypotheses.org/

École Doctorale, Université de Rennes 1, March 2015

Professor of Actuarial Sciences, Mathematics Department, UQàM
(previously Economics Department, Univ. Rennes 1 & ENSAE Paristech
actuary AXA General Insurance Hong Kong, IT & Stats FFSA)
PhD in Statistics (KU Leuven), Fellow Institute of Actuaries
MSc in Financial Mathematics (Paris Dauphine) & ENSAE
Editor of the freakonometrics.hypotheses.org's blog
Editor of Computational Actuarial Science, CRC



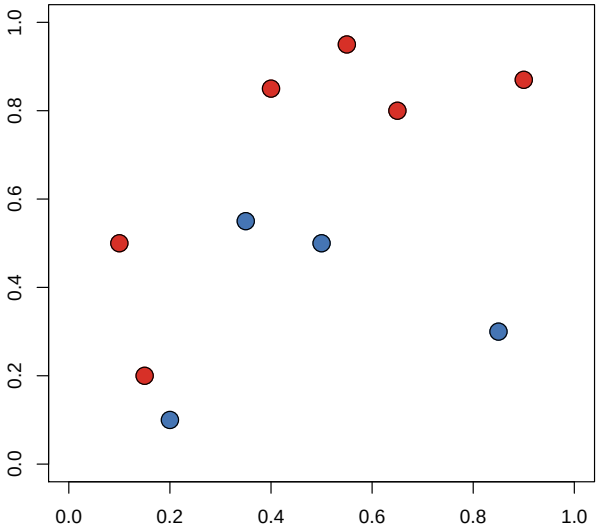
Supervised Techniques : Classification

(Linear) Discriminant Analysis

Data : $\{(\boldsymbol{x}_i, y_i) = (x_{1,i}, x_{2,i}, y_i), i = 1, \cdots, n\}$

with $y_i \in \{0, 1\}$ or $y_i \in \{-1, +1\}$ or $y_i \in \{\bullet, \bullet\}$

$$\begin{cases} \boldsymbol{X}|Y = 0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \\ \boldsymbol{X}|Y = 1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \end{cases}$$



Fisher's linear discriminant

$$\boldsymbol{\omega} \propto [\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1]^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

maximizes

$$\frac{\text{variance between}}{\text{variance within}} = \frac{[\boldsymbol{\omega} \cdot \boldsymbol{\mu}_1 - \boldsymbol{\omega} \cdot \boldsymbol{\mu}_0]^2}{\boldsymbol{\omega}^\intercal \boldsymbol{\Sigma}_1 \boldsymbol{\omega} + \boldsymbol{\omega}^\intercal \boldsymbol{\Sigma}_0 \boldsymbol{\omega}}$$

see Fisher (1936, wiley.com)

Supervised Techniques : Classification

Logistic Regression

Data : $\{(\boldsymbol{x}_i, y_i) = (x_{1,i}, x_{2,i}, y_i), i = 1, \cdots, n\}$

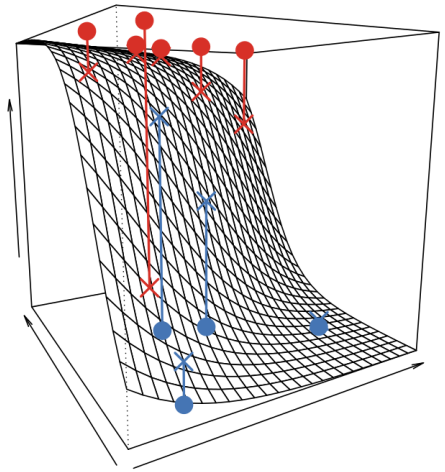
$$\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \frac{\exp[\boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}]}{1 + \exp[\boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}]}$$

Inference using maximum likelihood techniques

$$\widehat{\boldsymbol{\beta}} = \text{argmin}\left\{\sum_{i=1}^{n} \log[\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x})]\right\}$$

and the score model is then

$$s(\boldsymbol{X} = \boldsymbol{x}) = \frac{\exp[\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}]}{1 + \exp[\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}]}$$



Supervised Techniques : Classification

Logistic Regression

Historically, the idea was to model the *odds ratio*,

$$\frac{\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})}{\mathbb{P}(Y\neq 1|\boldsymbol{X}=\boldsymbol{x})}=\exp[\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta}]$$

(the odds ratio is a positive number). Hence,

$$\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})=H(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta}) \text{ where } H(\cdot)=\frac{\exp[\cdot]}{1+\exp[\cdot]}$$

is the c.d.f. of the *logistic* variable, popular in demography, see Verhulst (1845, gdz.sub.uni-goettingen.de)
cf TRISS Tauma, Boyd *et al.* (1987, journals.lww.com)

Soit p la population : représentons par dp l'accroissement infiniment petit qu'elle reçoit pendant un temps infiniment court dt. Si la population croissait en progression géométrique, nous aurions l'équation $\frac{dp}{dt}=mp$. Mais comme la vitesse d'accroissement de la population est retardée par l'augmentation même du nombre des habitans, nous devrons retrancher de mp une fonction inconnue de p; de manière que la formule à intégrer deviendra

$$\frac{dp}{dt}=mp-\varphi(p).$$

L'hypothèse la plus simple que l'on puisse faire sur la forme de la fonction φ, est de supposer $\varphi(p)=np^2$. On trouve alors pour intégrale de l'équation ci-dessus

$$t=\frac{1}{m}[\log. p-\log.(m-np)]+\text{constante},$$

et il suffira de trois observations pour déterminer les deux coefficiens constans m et n et la constante arbitraire.

En résolvant la dernière équation par rapport à p, il vient

$$p=\frac{mp'e^{mt}}{np'e^{mt}+m-np'} \quad \ldots\ldots (1)$$

en désignant par p' la population qui répond à $t=o$, et par e la base des logarithmes népériens. Si l'on fait $t=\infty$, on voit que la valeur de p correspondante est $P=\frac{m}{n}$. Telle est donc *la limite supérieure de la population*.

Supervised Techniques : Classification

Probit Regression

Bliss (1934) sciencemag.org) suggested a model such that

$$\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})=H(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta}) \text{ where } H(\cdot)=\Phi(\cdot)$$

the c.d.f. of the $\mathcal{N}(0,1)$ distribution. This is the *probit* model.

This yields a latent model, $y_i=\mathbf{1}(y_i^\star>0)$ where

$y_i^\star=\boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{\beta}+\varepsilon_i$ is a nonobservable score.

Table 3.2 Transformation of percentages to probits

%	0	1	2	3	4	5	6	7	8	9
0	—	2.67	2.95	3.12	3.25	3.36	3.45	3.52	3.59	3.66
10	3.72	3.77	3.82	3.87	3.92	3.96	4.01	4.05	4.08	4.12
20	4.16	4.19	4.23	4.26	4.29	4.33	4.36	4.39	4.42	4.45
30	4.48	4.50	4.53	4.56	4.59	4.61	4.64	4.67	4.69	4.72
40	4.75	4.77	4.80	4.82	4.85	4.87	4.90	4.92	4.95	4.97
50	5.00	5.03	5.05	5.08	5.10	5.13	5.15	5.18	5.20	5.23
60	5.25	5.28	5.31	5.33	5.36	5.39	5.41	5.44	5.47	5.50
70	5.52	5.55	5.58	5.61	5.64	5.67	5.71	5.74	5.77	5.81
80	5.84	5.88	5.92	5.95	5.99	6.04	6.08	6.13	6.18	6.23
90	6.28	6.34	6.41	6.48	6.55	6.64	6.75	6.88	7.05	7.33
—	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
99	7.33	7.37	7.41	7.46	7.51	7.58	7.65	7.75	7.88	8.09

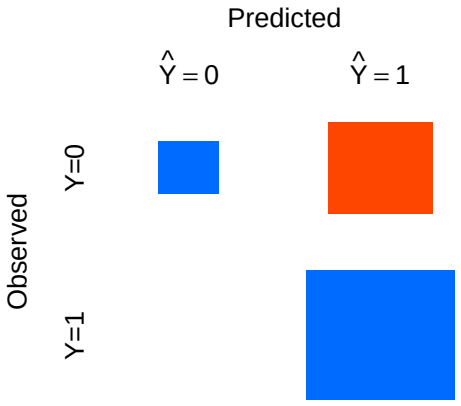
Supervised Techniques : Classification

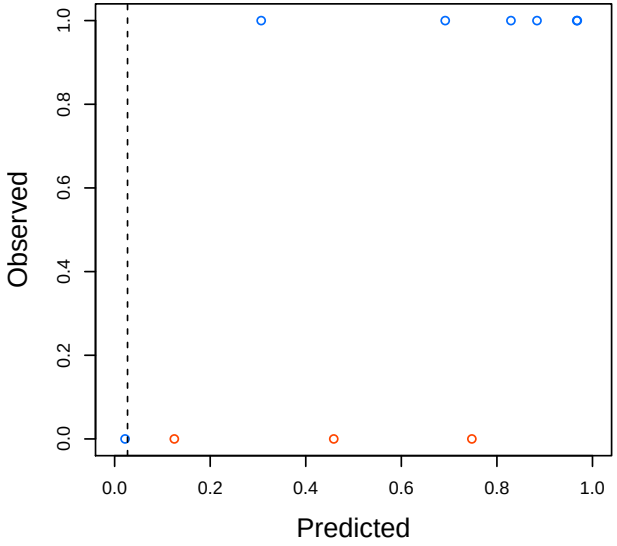
Logistic Regression

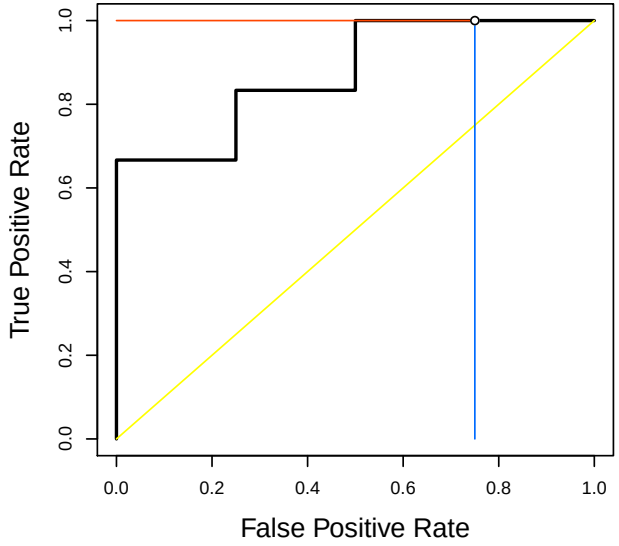
The classification function, from a score to a class:

if $s(\boldsymbol{x}) > s$, then $\widehat{Y}(\boldsymbol{x}) = 1$ and $s(\boldsymbol{x}) \leq s$, then $\widehat{Y}(\boldsymbol{x}) = 0$

Plot $TP(s) = \mathbb{P}[\widehat{Y} = 1 | Y = 1]$ vs. $FP(s) = \mathbb{P}[\widehat{Y} = 1 | Y = 0]$







Supervised Techniques : Classification

Logistic Additive Regression

Data : $-(\boldsymbol{x}_i, y_i) = (x_{1,i}, x_{2,i}, y_i), i = 1, \cdots, n''$

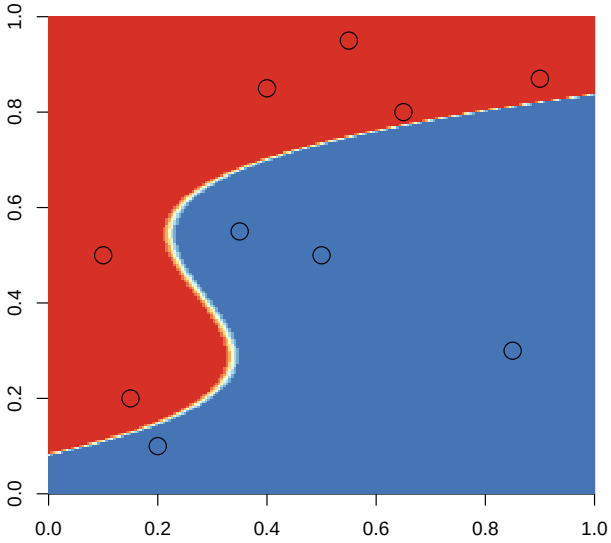
with $y_i \in -0, 1''$ or $y_i \in --1, +1''$ or $y_i \in -\bullet, \bullet''$

Instead of a linear function

$$\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \frac{\exp[\boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}]}{1 + \exp[\boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}]}$$

consider

$$\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \frac{\exp[h(\boldsymbol{x})]}{1 + \exp[h(\boldsymbol{x})]}$$



Supervised Techniques : Classification

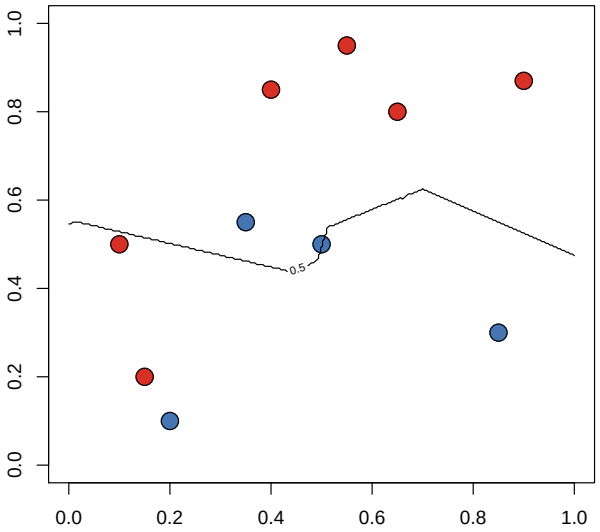
k-Nearest Neighbors

Data : $-(\boldsymbol{x}_i, y_i) = (x_{1,i}, x_{2,i}, y_i), i = 1, \cdots, n''$

with $y_i \in -0, 1''$ or $y_i \in - - 1, +1''$ or $y_i \in -$●, ●$''$

for each $\boldsymbol{x}$, consider the k nearest neighbors (for some distance $d(\boldsymbol{x}, \boldsymbol{x}_i)$) $V_k(\boldsymbol{x})$

$$s(\boldsymbol{x}) = \frac{1}{k} \sum_{i \in V_k(\boldsymbol{x})} Y_i$$



Supervised Techniques : Classification

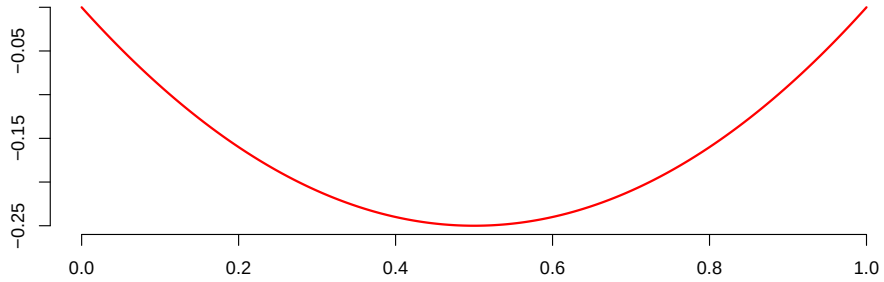
CART and Classification Tree

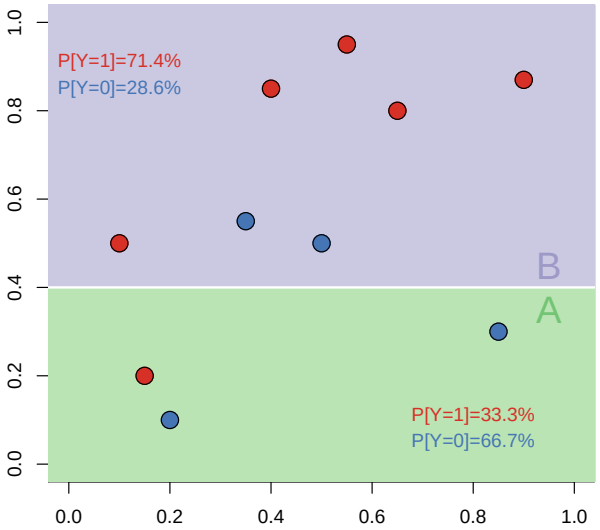
Data : $-\boldsymbol{x}_i, y_i) = (x_{1,i}, x_{2,i}, y_i), i = 1, \cdots, n''$

with $y_i \in -0, 1''$ or $y_i \in --1, +1''$ or $y_i \in -\bullet, \bullet''$

Compute some impurity criteria, e.g. Gini index,

$$- \sum_{P \in -A,B''} \underbrace{\mathbb{P}[\boldsymbol{x} \in P]}_{\text{size}} \underbrace{\mathbb{P}[Y = 1 | \boldsymbol{x} \in P]}_{p} \underbrace{\mathbb{P}[Y = 0 | \boldsymbol{x} \in P]}_{(1-p)}$$





Supervised Techniques : Classification

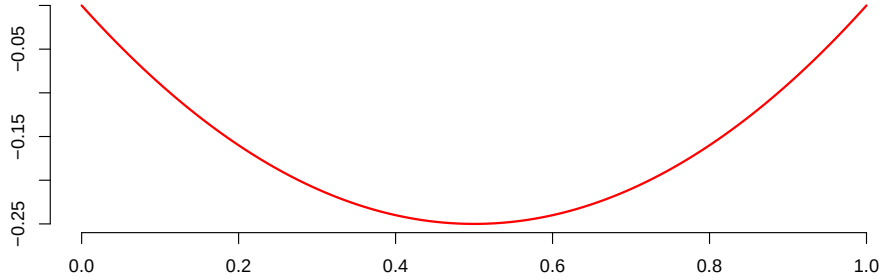
CART and Classification Tree

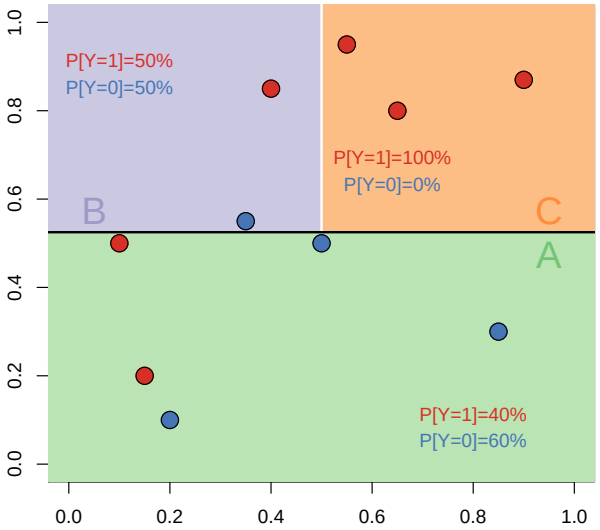
Data : $-\boldsymbol{x}_i, y_i) = (x_{1,i}, x_{2,i}, y_i), i = 1, \cdots, n''$

with $y_i \in -0, 1''$ or $y_i \in - - 1, +1''$ or $y_i \in -\bullet, \bullet''$

Given a partition, loop,

$$- \sum_{P \in -A,B,C''} \mathbb{P}[\boldsymbol{x} \in P]\mathbb{P}[Y = 0|\boldsymbol{x} \in P]\mathbb{P}[Y = 1|\boldsymbol{x} \in P]$$



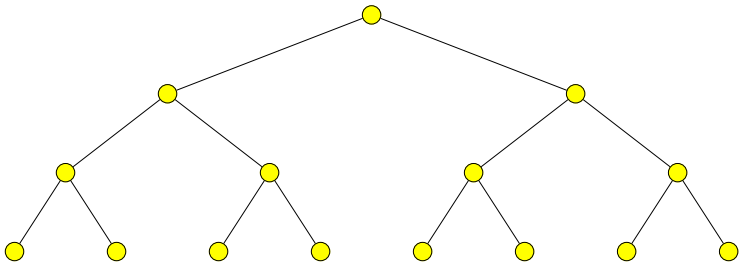


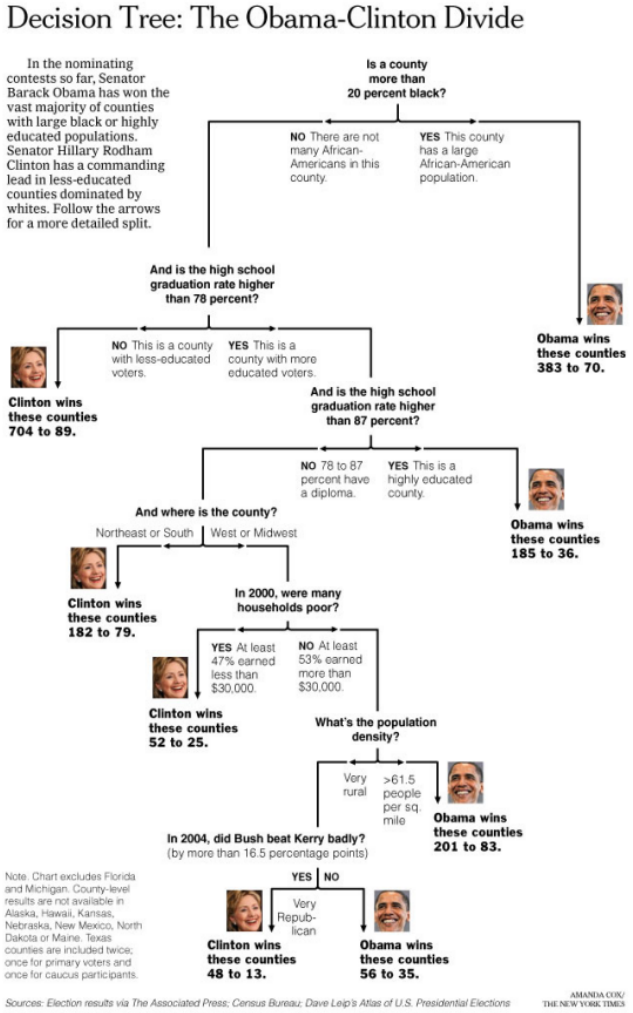
Supervised Techniques : Classification

CART and Classification Tree

Breiman *et al.* (1984, stat.berkeley.edu) developed CART (Classification and Regression Trees) algorithm

One creates a complete binary tree, and then prunning starts,





Supervised Techniques : Classification

Random Forrests and Bootstraping Technique

Data : $-(\boldsymbol{x}_i, y_i) = (x_{1,i}, x_{2,i}, y_i), i = 1, \cdots, n''$

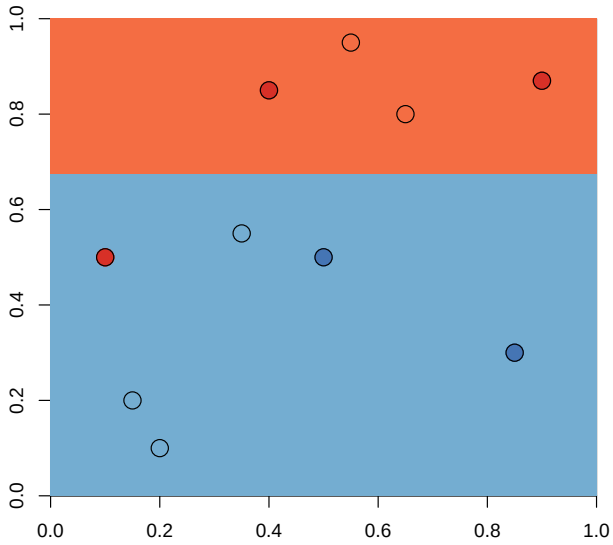
with $y_i \in -0, 1''$ or $y_i \in - - 1, +1''$ or $y_i \in -$●, ●$''$

Estimate tree on $-(\boldsymbol{x}_i^{\star,b}, y_i^{\star,b})''$ on a bootstraped sample

Estimate $s(\boldsymbol{x})$ (or $s(\boldsymbol{x}_i)$ only) $\longrightarrow \widehat{s}^b(\boldsymbol{x})$

and generate (many) other samples.

See Breiman (2001, stat.berkeley.edu)



Supervised Techniques : Classification

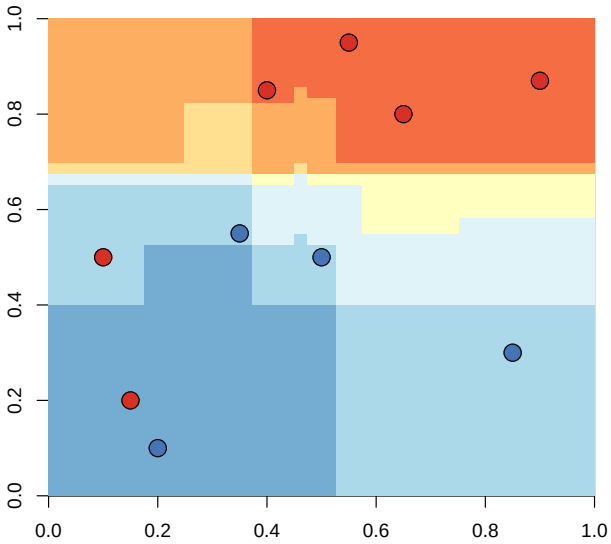
Random Forrests and Aggregation

Data : $-(\boldsymbol{x}_i, y_i) = (x_{1,i}, x_{2,i}, y_i), i = 1, \cdots, n''$

with $y_i \in -0, 1''$ or $y_i \in - - 1, +1''$ or $y_i \in -\bullet, \bullet''$

Define

$$\widehat{s}(\boldsymbol{x}) = \frac{1}{B} \sum_{b=1}^{B} \widehat{s}^b(\boldsymbol{x})$$



Supervised Techniques : Double Classification

Uplift Techniques

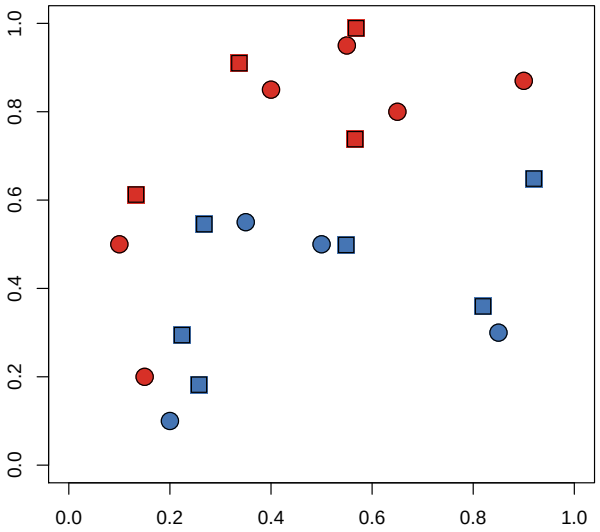
Data : $-(\boldsymbol{x}_i, y_i) = (x_{1,i}, x_{2,i}, y_i)''$ with $y_i \in -\bullet, \bullet''$

$-(\boldsymbol{x}_j, y_j) = (x_{1,j}, x_{2,j}, y_j)''$ with $y_i \in -\blacksquare, \blacksquare''$

See clinical trials, treatment vs. control group

E.g. direct mail campaign in a bank

	Control ■	Promotion ●
No Purchase	85.17%	61.60%
Purchase	14.83%	38.40%



overall uplift effect +23.57%

(see Guelman *et al.*, 2014 j.insmatheco.2014.06.009)

Consistency of Models

In supervised models, the errors related to the difference between y_i's and $\widehat{y}_i$'s.

Consider some loss function $L(y_i, \widehat{y}_i)$, e.g.

- in classification, $\mathbf{1}(y_i \neq \widehat{y}_i)$ (misclassification)
- in regression, $(y_i - \widehat{y}_i)^2$ (cf least squares, ℓ_2 norm)

Consider some statistical model, $m(\cdot)$, estimated on sample $-y_i, \boldsymbol{x}_i''$ of size n. Let $\widehat{m}_n(\cdot)$ denote that estimator, so that $\widehat{y}_i = \widehat{m}_n(\boldsymbol{x}_i)$. $\widehat{m}_n(\cdot)$ is a regression function when y is continuous, and is a classifier when y is categorical (say $-0, 1''$).

$m_n(\cdot)$ has been trained from a learning sample $(y_i, \boldsymbol{x}_i)$.

Consistency of Models

The risk is $R_n = \mathbb{E}(L) = \int L(y, \widehat{m}_n(\boldsymbol{x}))d\mathbb{P}(y, \boldsymbol{x})$

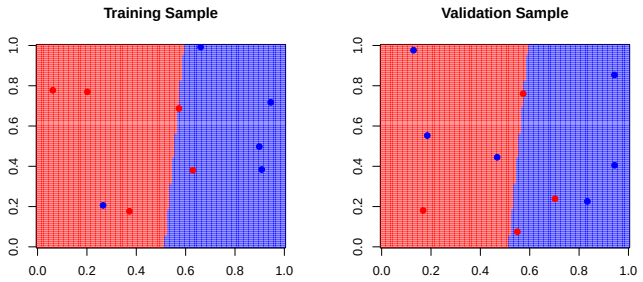
The empirical risk is $\widehat{R}_n = \dfrac{1}{n}\sum_{i=1}^{n} L(y_i, \widehat{m}_n(\boldsymbol{x}_i)))$ on a training sample $-y_i, \boldsymbol{x}_i{}''$.

The generalized risk is $\tilde{R}_{n'} = \dfrac{1}{n'}\sum_{i=1}^{n'} L(\tilde{y}_i, \widehat{m}_n(\tilde{\boldsymbol{x}}_i)))$ on a validation sample $-\tilde{y}_i, \tilde{\boldsymbol{x}}_i{}''$.

We have a consistent model if $\widehat{R}_n \to R_n$ as $n \to \infty$.

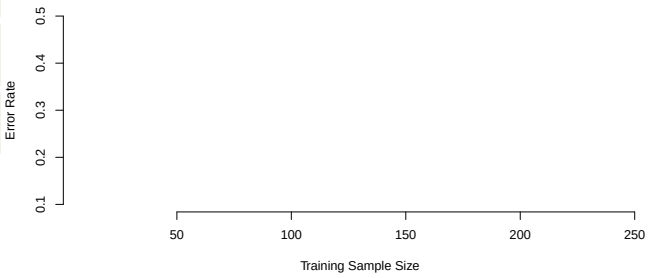
Consistency of Models

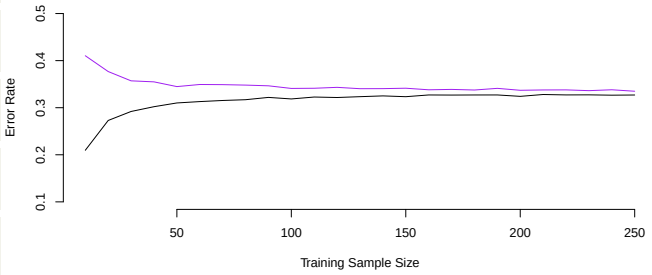
```
> U <- data.frame(X1=runif(n),X2=runif(n))
> U$Y <- rbinom(n,size=1,prob=(U[,1]+U[,2])/
    2)
> reg <- glm(Y~X1+X2,data=U,family=binomial)
> pd=function(x1,x2) predict(reg,newdata=
    data.frame(X1=x1,X2=x2),type="response")
    >.5
> MissClassU <- mean(abs(pd(U$X1,U$X2)-U$Y))
```



Consider some polynomial regression

```
> V <- data.frame(X1=runif(n),X2=runif(n))
> V$Y <- rbinom(n,size=1,prob=(V[,1]+V[,2])/
    2)
> MissClassV[s <- mean(abs(pd(V$X1,V$X2)-V$Y
    ))
```





Consistency: why is it complicated ?

From the law of large numbers,

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}U_i=\mathbb{E}[U]$$

when the U_i's are i.i.d., and $U_i\sim U$. Here we look for

$$\underbrace{\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}L(y_i,\widehat{m}_n(\boldsymbol{x}_i))}_{R_n}=?$$

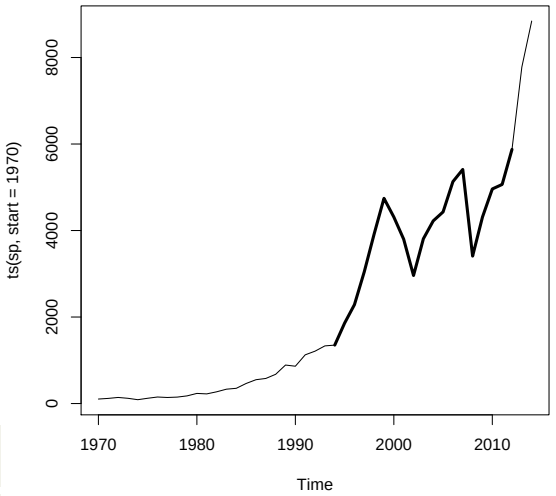
If $(y,\boldsymbol{x}_i)'s$ are independent, it is not the case for the $L(y_i,\widehat{m}_n(\boldsymbol{x}_i))$'s, because of $\widehat{m}_n(\cdot)$, also estimated on this training sample. But, on the validation sample

$$\underbrace{\lim_{n'\to\infty}\frac{1}{n'}\sum_{i=1}^{n'}L(\tilde{y}_i,\widehat{m}_n(\tilde{\boldsymbol{x}}_i))}_{\tilde{R}_n}=\mathbb{E}[R(L(Y,m(\boldsymbol{X}))]$$

Overfitting

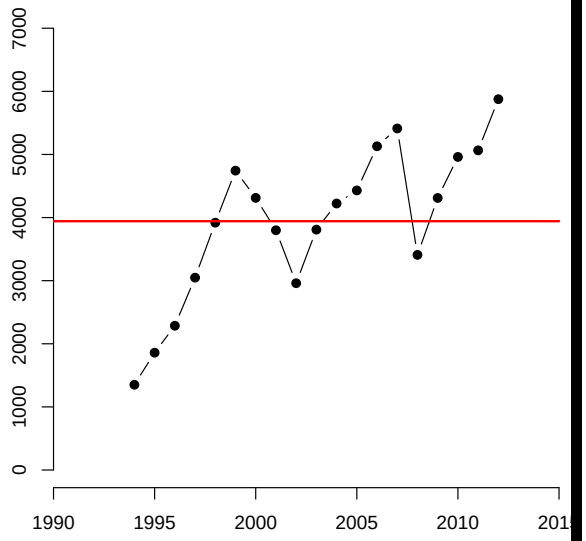
Inspired by Leinweber (1995, nerdsonwall-street.typepad.com). Consider S&P 500 yearly value, en.wikipedia.org

```
> sp <- read.table("sp.txt",header=TRUE)$SP500
> plot(ts(sp,start=1970))
> T <- 1994:2012
> X <- sp[T-1969]
```



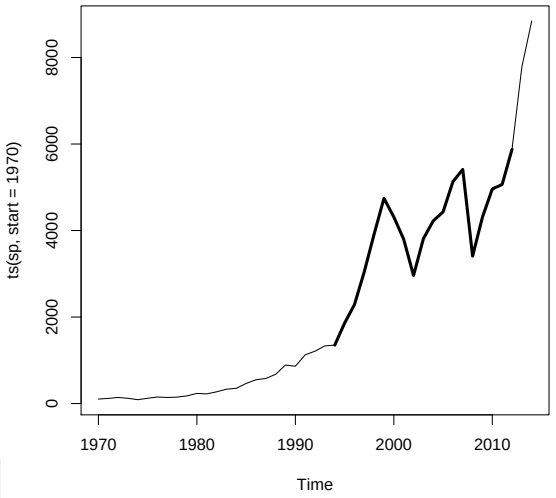
Consider some polynomial regression

```
> df <- data.frame(X,T)
> reg <- lm(X~poly(T,k),data=df)
> tp <- seq(1990,2015,by=.1)
> yp <- predict(reg,newdata=data.frame(T=tp))
```



Overfitting

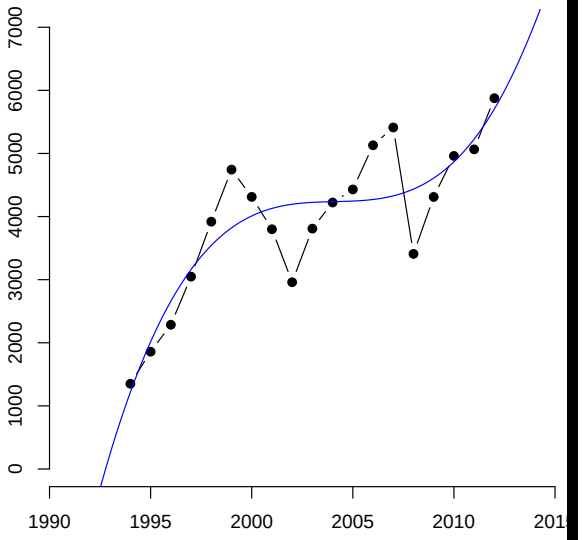
Inspired by Leinweber (1995, nerdsonwall-street.typepad.com). Consider S&P 500 yearly value, en.wikipedia.org




```
> sp <- read.table("sp.txt",header=TRUE)$SP500
> plot(ts(sp,start=1970))
> library(splines)
```

Consider some spline regression

```
> df <- data.frame(X,T)
> reg <- lm(X~bs(T,k),data=df)
> tp <- seq(1990,2015,by=.1)
> yp <- predict(reg,newdata=data.frame(T=tp))
```



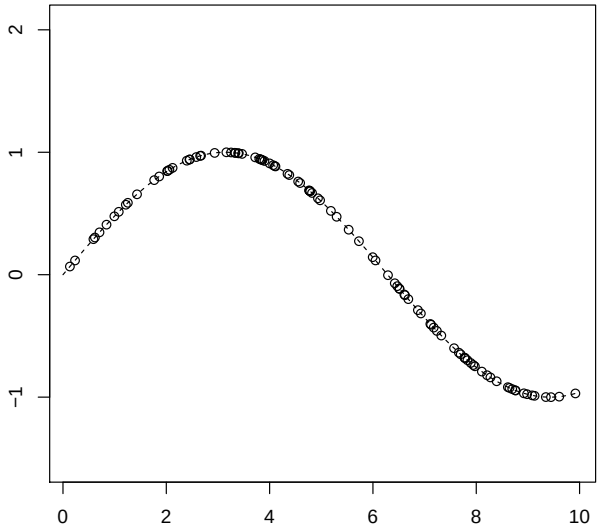
Under- and Over-fitting

Consider some simulated data

```
> set.seed(1)
> n <- 100
> x <- runif(n)*10
> y0 <- sin(x/2)
> y <- y0 + rnorm(length(x))/2
```

A (too) perfect sample (with no noise) would be

```
> plot(x,y0)
```



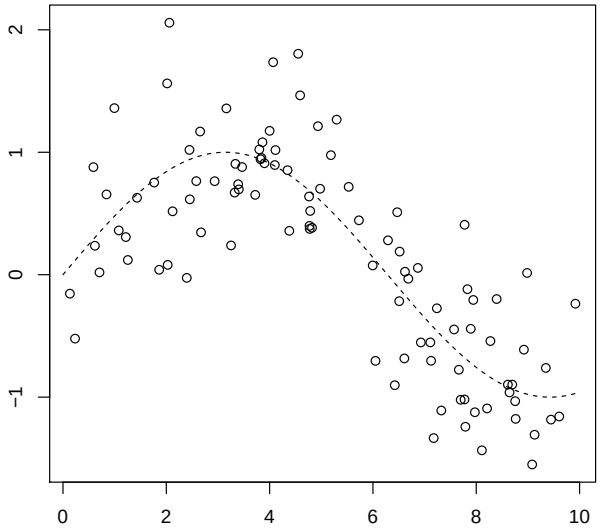
Under- and Over-fitting

Consider some simulated data

```
> set.seed(1)
> n <- 100
> x <- runif(n)*10
> y0 <- sin(x/2)
> y <- y0 + rnorm(length(x))/2
```

while our sample with some noise is

```
> plot(x,y)
```



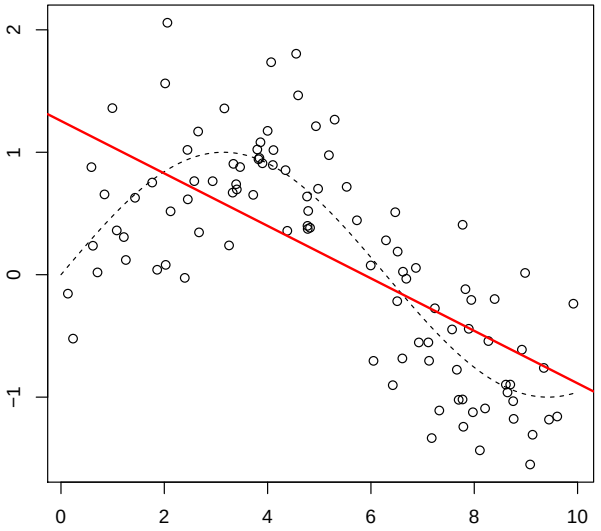
Underfitting

Consider some simulated data

```
> set.seed(1)
> n <- 100
> x <- runif(n)*10
> y0 <- sin(x/2)
> y <- y0 + rnorm(length(x))/2
```

An underfitted model is the linear model

```
> reg <- lm(y~x,data=db)
> plot(x,predict(reg))
```



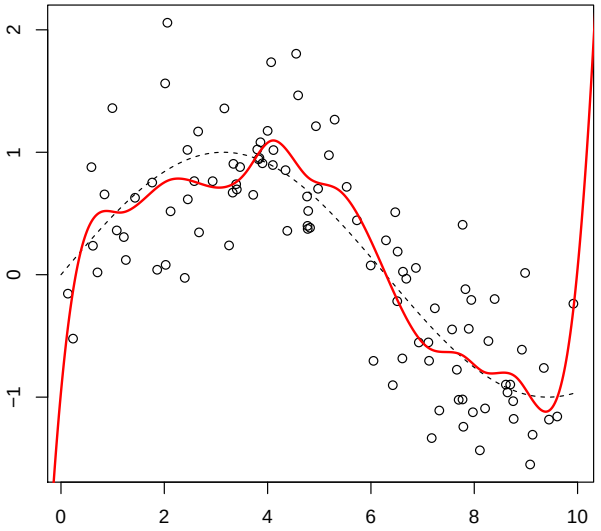
Overfitting

Consider some simulated data

```
> set.seed(1)
> n <- 100
> x <- runif(n)*10
> y0 <- sin(x/2)
> y <- y0 + rnorm(length(x))/2
```

while an overfitted model is

```
> library(splines)
> reg <- lm(y~bs(x,15),data=db)
> plot(x,predict(reg))
```



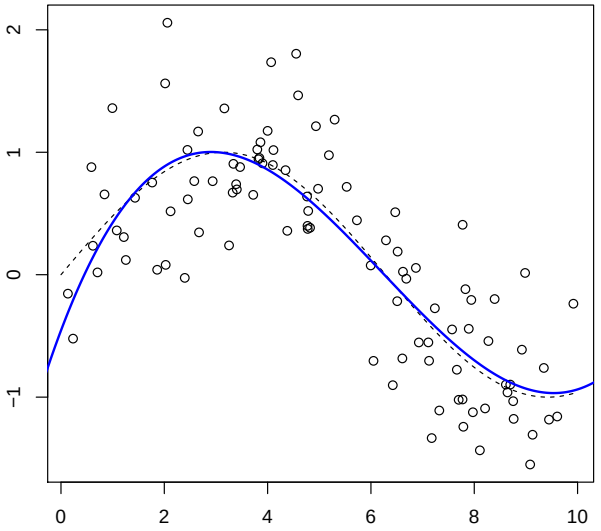
The Goldilocks Principle

Consider some simulated data

```
> set.seed(1)
> n <- 100
> x <- runif(n)*10
> y0 <- sin(x/2)
> y <- y0 + rnorm(length(x))/2
```

(too) perfect model

```
> library(splines)
> reg <- lm(y~bs(x),data=db)
> plot(x,predict(reg))
```



Overfitting and Consistency

There is a gap between R_n (on the training sample) and $\widehat{R}_n$ (on the validation sample). One can prove that

$$\widehat{R}_n \leq R_n + \sqrt{\frac{\text{VC}[\log(2n/d)+1] - \log[\alpha/4]}{n}}$$

with probability $1-\alpha$, where VC denotes the Vapnik-Chervonenkis dimension.

A process is consistent if and only if $\text{VC} < \infty$.

VC can be seen as a function of complexity of some models.

Consider polynomial regressors, of degree d, with k covariates,

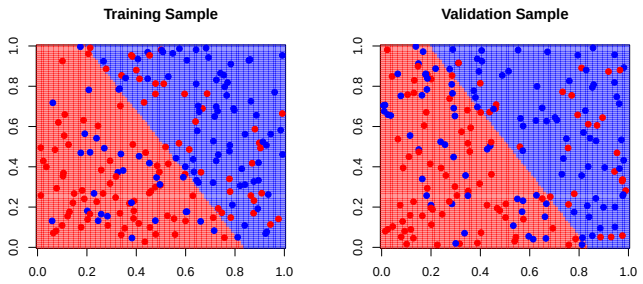
- if $k = 1$ then $\text{VC} = d+1$
- if $k = 2$ then $\text{VC} = \dfrac{(d+1)(d+2)}{2}$ (bivariate polynomials)

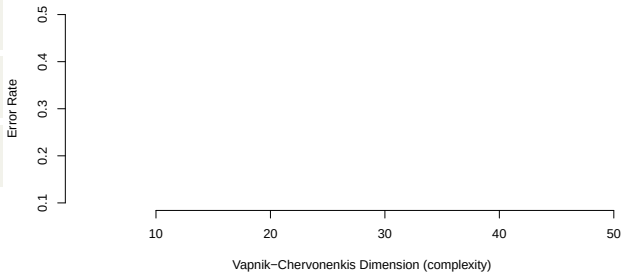
  if $k = 2$ then $\text{VC} = 2(d+1)$ (additive model, sum of (univariate) polynomials)

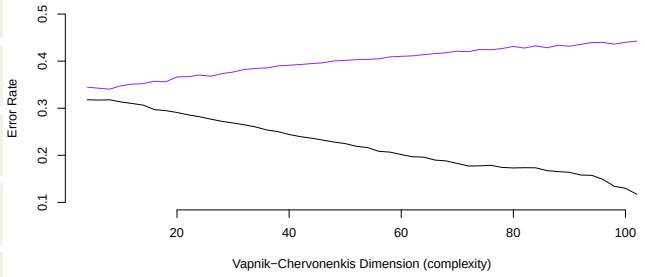
```
> U <- data.frame(X1=runif(n),X2=runif(n))
> U$Y <- rbinom(n,size=1,prob=(U[,1]+U[,2])/
    2)
> reg <- glm(Y~poly(X1,s)+poly(X2,s),data=U,
family=binomial)
> pd=function(x1,x2) predict(reg,newdata=
    data.frame(X1=x1,X2=x2),type="response")
    >.5
> MissClassU <- mean(abs(pd(U$X1,U$X2)-U$Y))
```

Consider some polynomial regression

```
> V <- data.frame(X1=runif(n),X2=runif(n))
> V$Y <- rbinom(n,size=1,prob=(V[,1]+V[,2])/
    2)
> MissClassV[s <- mean(abs(pd(V$X1,V$X2)-V$Y
    ))
```







Regression vs. Statistical Learning

In a linear regression, $y = \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta} + \varepsilon$. Parameter $\boldsymbol{\beta}$ is estimated from observations $-y_i, \boldsymbol{x}_i$''s.

The estimation is consistent if $\widehat{\boldsymbol{\beta}}_n \to \boldsymbol{\beta}$, as $n \to \infty$.

In a regression, $y = m(\boldsymbol{x}) + \varepsilon$. Function $m(\cdot)$ is estimated from observations $-y_i, \boldsymbol{x}_i$''s.

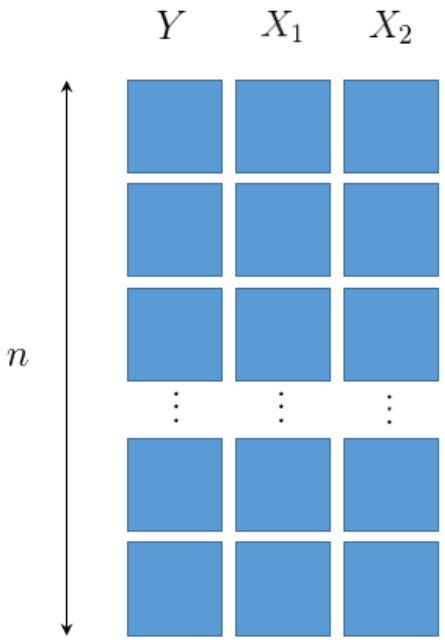
The estimation is consistent if $\widehat{m}_n(\cdot) \to m(\cdot)$, as $n \to \infty$.

In the statistical learning perspective, the interest is not really on $m(\cdot)$, but on the prediction error. $\widehat{m}_n(\cdot)$ is said to be consistent if, given a loss function L

$$\mathbb{E}[L(\widehat{m}_n(\boldsymbol{X}), Y)] \to \mathbb{E}[L(m(\boldsymbol{X}), Y)]$$

as $n \to \infty$.

Data Science, in 2015



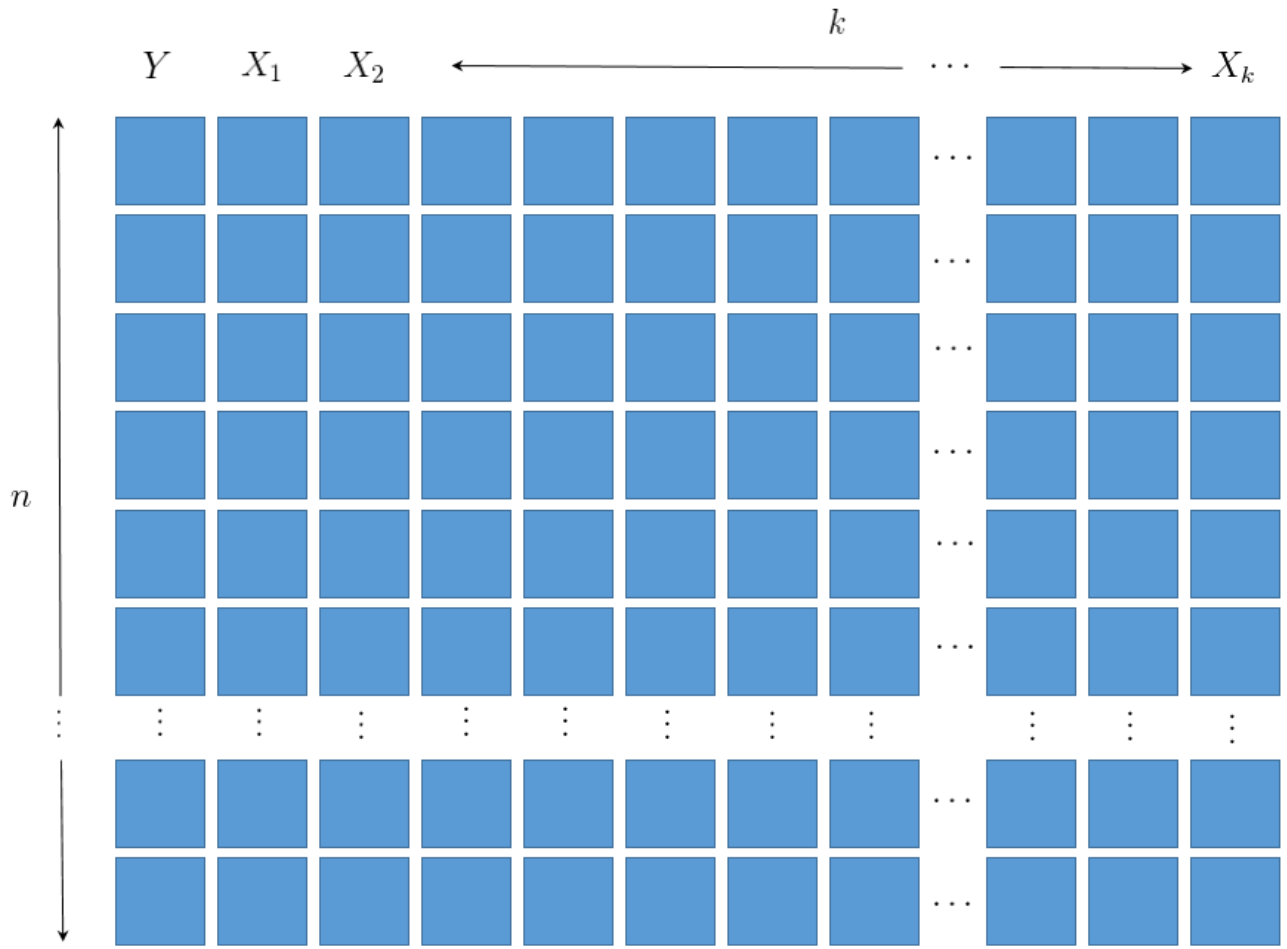
Historically, we had limited information, $(Y_i, \boldsymbol{X}_i)$, the goal was to get n as large as possible (asymptotic theory).

$\mathbb{E}(Y|X_1 = x_1) = m(x_1)$ regression line

$\mathbb{E}(Y|X_1 = x_1, X_2 = x_2) = m(x_1, x_2)$ regression surface

$\mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}) = m(\boldsymbol{x})$ regression function

Data Science, in 2015



n can be large, but limited
(portfolio size)
large variety k
large volume nk

$\rightarrow$ Feature Engineering

Regression?

Galton (1870, galton.org, 1886, galton.org) and Pearson & Lee (1896, jstor.org, 1903 jstor.org) studied genetic transmission of characterisitcs, e.g. the heigth.

On average the child of tall parents is taller than other children, but less than his parents.

"I have called this peculiarity by the name of regression', Francis Galton, 1886.

REGRESSION *towards* MEDIOCRITY *in* HEREDITARY STATURE.
By FRANCIS GALTON, F.R.S., &c.

Table 8.1. Galton's 1885 cross-tabulation of 928 adult children born of 205 midparents, by their height and their midparent's height.

Height of the mid-parent in inches	Height of the adult child													Total no. of adult children	Total no. of mid-parents	Medians
	<61.7	62.2	63.2	64.2	65.2	66.2	67.2	68.2	69.2	70.2	71.2	72.2	73.2	>73.7		
>73.0	—	—	—	—	—	—	—	—	—	—	—	1	3	—	4	5	—
72.5	—	—	—	—	—	—	1	2	1	2	7	2	4	19	6	72.2
71.5	—	—	—	1	3	4	5	5	10	4	9	2	2	45	11	69.9
70.5	1	—	1	—	1	1	3	12	18	14	7	4	3	3	68	22	69.5
69.5	—	—	1	16	4	17	27	20	33	25	20	11	4	5	183	41	68.9
68.5	1	—	7	11	16	25	31	34	48	21	18	4	3	—	219	49	68.2
67.5	—	3	5	14	15	36	38	28	38	19	11	4	—	—	211	33	67.6
66.5	—	3	3	5	2	17	17	14	13	4	—	—	—	—	78	20	67.2
65.5	1	—	9	5	7	11	11	7	7	5	2	1	—	—	66	12	66.7
64.5	1	1	4	4	1	5	5	—	2	—	—	—	—	—	23	5	65.8
<64.0	1	—	2	4	1	2	2	1	1	—	—	—	—	—	14	1	
Totals	5	7	32	59	48	117	138	120	167	99	64	41	17	14	928	205	—
Medians	—	—	66.3	67.8	67.9	67.7	67.9	68.3	68.5	69.0	69.0	70.0	—	—	—	—	—

Source: Galton (1888a).

Note: All female heights were multiplied by 1.08 before tabulation. Galton added an explanatory footnote to the table: "In calculating the Medians, the entries have been taken as referring to the middle of the squares in which they stand. The reason why the headings run 62.2, 63.2, &c., instead of 62.5, 63.5, &c., is that the observations are unequally distributed between 62 and 63, 63 and 64, &c., there being a strong bias in favour of integral inches. After careful consideration, I concluded that the headings, as adopted, best satisfied the conditions. This inequality was not apparent in the case of the Mid-parents." Galton republished these data in 1889, where they are referred to as the R.F.F. Data (Record of Family Faculties); he then noted that the first row must be in error (four children cannot have five sets of parents), but he claimed that "the bottom line, which looks suspicious, is correct" (p. 208).

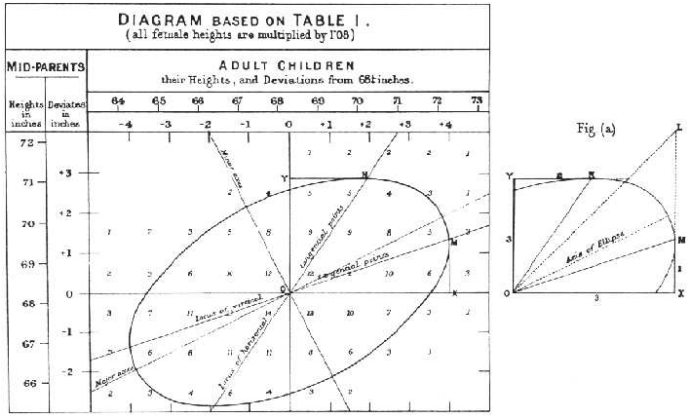
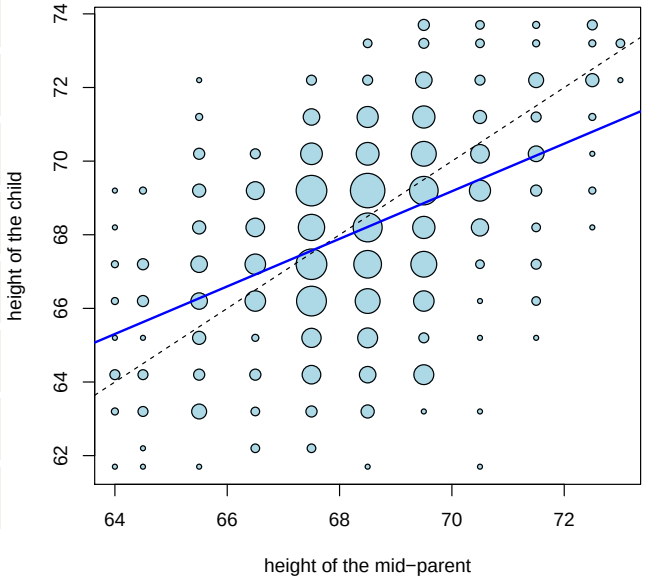


Figure 8.7. Galton's smoothed rendition of Table 8.1, with one of the "concentric and similar ellipses" drawn in. The geometric relationship of the two regression lines to the ellipse is also shown. (From Galton, 1886a.)

Regression?

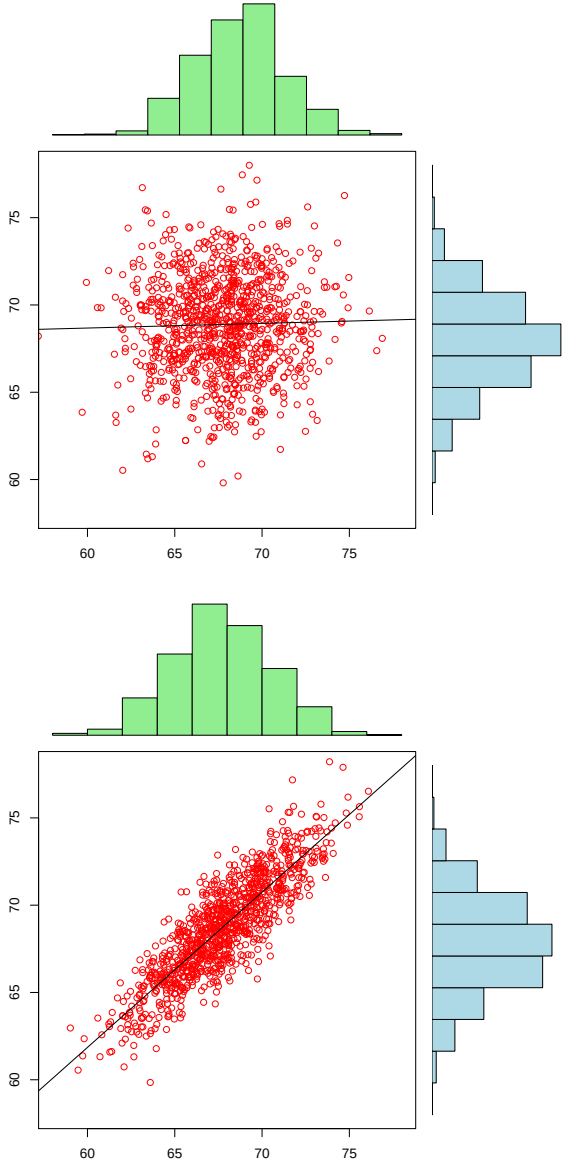
```
> library(HistData)
> attach(Galton)
> Galton$count <- 1
> df <- aggregate(Galton, by=list(parent,
    child), FUN=sum)[,c(1,2,5)]
> plot(df[,1:2],cex=sqrt(df[,3]/3))
> abline(a=0,b=1,lty=2)
> abline(lm(child~parent,data=Galton))
```



Regression?

Regression is a correlation problem

Overall, children are not smaller than parents



Least Squares?

Recall that

$$\mathbb{E}(Y) = \underset{m\in\mathbb{R}}{\text{argmin}} -\|Y-m\|_{\ell_2}^2 = \mathbb{E}\left([Y-m]^2\right) \text{ ''}$$

$$\text{Var}(Y) = \min_{m\in\mathbb{R}} -\mathbb{E}\left([Y-m]^2\right) \text{ ''} = \mathbb{E}\left([Y-\mathbb{E}(Y)]^2\right)$$

The empirical version is

$$\overline{y} = \underset{m\in\mathbb{R}}{\text{argmin}} -\sum_{i=1}^{n}\frac{1}{n}[y_i-m]^2 \text{ ''}$$

$$s^2 = \min_{m\in\mathbb{R}} -\sum_{i=1}^{n}\frac{1}{n}[y_i-m]^2 \text{ ''} = \sum_{i=1}^{n}\frac{1}{n}[y_i-\overline{y}]^2$$

The conditional version is

$$\mathbb{E}(Y|\boldsymbol{X}) = \underset{\varphi:\mathbb{R}^k\to\mathbb{R}}{\text{argmin}} -\|Y-\varphi(\boldsymbol{X})\|_{\ell_2}^2 = \mathbb{E}\left([Y-\varphi(\boldsymbol{X})]^2\right) \text{ ''}$$

$$\text{Var}(Y|\boldsymbol{X}) = \min_{\varphi:\mathbb{R}^k\to\mathbb{R}} -\mathbb{E}\left([Y-\varphi(\boldsymbol{X})]^2\right) \text{ ''} = \mathbb{E}\left([Y-\mathbb{E}(Y|\boldsymbol{X})]^2\right)$$

Errors in Regression type Models

In predictions, there are two kinds of errors

- error on the best estimate, $\widehat{Y}$
- error on the true value, Y

Recall that

$$Y = \underbrace{\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta}}_{\text{model}} + \underbrace{\varepsilon}_{\text{error}} = \underbrace{\boldsymbol{x}^{\mathsf{T}}\widehat{\boldsymbol{\beta}}}_{\text{prediction } \widehat{Y}} + \widehat{\varepsilon}.$$

- error on the best estimate, $\text{Var}(\widehat{Y}|\boldsymbol{X}=\boldsymbol{x}) = \boldsymbol{x}^{\mathsf{T}}\text{Var}(\widehat{\boldsymbol{\beta}})\boldsymbol{x}$ (inference error)
- error on the true value, $\text{Var}(Y|\boldsymbol{X}=\boldsymbol{x}) = \text{Var}(\varepsilon)$ (model error)

⚠ asymptotically (under suitable conditions), $\text{Var}(\widehat{\boldsymbol{\beta}}_n) \to \mathbf{0}$ as $n \to \infty$.

Errors in Regression type Models

Under the Gaussian assumption, we can derive (approximated) confidence intervals. For the best estimate,

$$\left[\widehat{Y} \pm 1.96\widehat{\sigma}\sqrt{\boldsymbol{x}^{\mathsf{T}}[\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}]^{-1}\boldsymbol{x}}\right]$$

```
> predict(lm(Y~X,data=df),newdata=data.frame(X=x), interval='
    confidence')
```

⚠ this confidence interval is a statement about estimates

For the 'true value'

$$\left[\widehat{Y} \pm 1.96\widehat{\sigma}\right]$$

```
> predict(lm(Y~X,data=df),newdata=data.frame(X=x), interval='
    prediction')
```

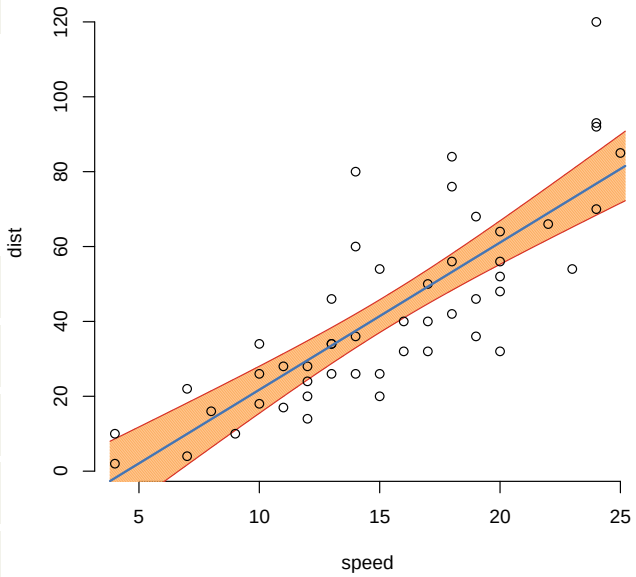
Resampling Techniques in Regression type Models

Consider some linear model

```
> plot(cars)
> reg <- lm(dist~speed,data=cars)
```

then if we plot the prediction with some confidence band

```
> u <- seq(3.8,25.2,length=251)
> v <- predict(reg,newdata=data.frame(speed=
    u),interval="confidence")
> polygon(c(u,rev(u)),c(v[,2],rev(v[,3])),
    border=NA)
> lines(u,v[,1],lwd=2)
```



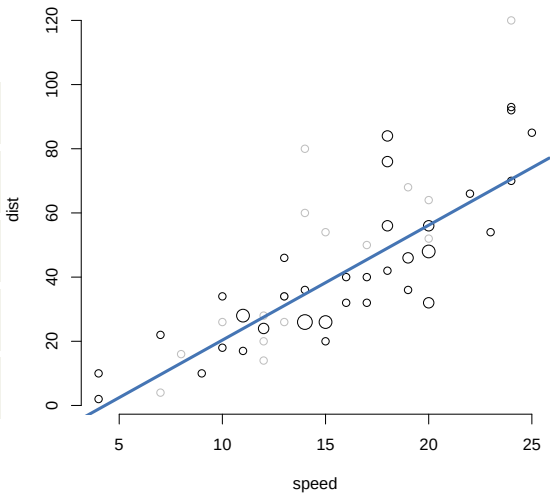
Resampling Techniques in Regression type Models

Resampling techniques can be used to generate a confidence interval.

1. Draw pairs from the sample

Resample from $-(\boldsymbol{X}_i, Y_i)''$

```
> V=matrix(NA,100,251)
> for(i in 1:100){
+ ind <- sample(1:n,size=n,replace=TRUE)
+ V[i,] <- predict(lm(dist~speed,data=cars[
    ind,]),newdata=data.frame(speed=u))}
```

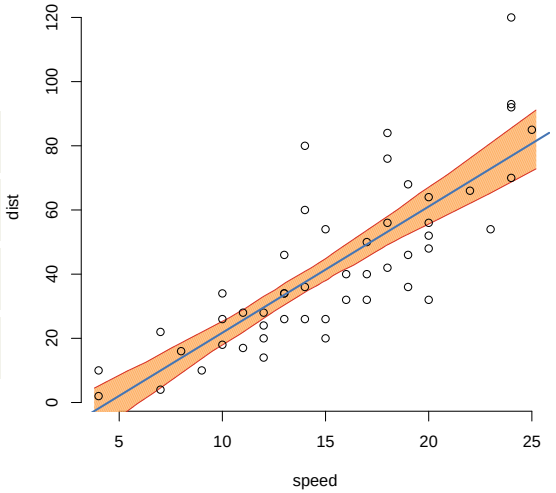


Then we can derive pointwise confidence intervals.

Resampling Techniques in Regression type Models

Confidence intervals are obtained using

```
> q1 <- apply(V,2,function(x) quantile(x
    ,.05))
> q2 <- apply(V,2,function(x) quantile(x
    ,.95))
```



Those values can be compared with the second and the third column of

```
> v <- predict(reg,newdata=data.frame(speed=u),interval="confidence")
```

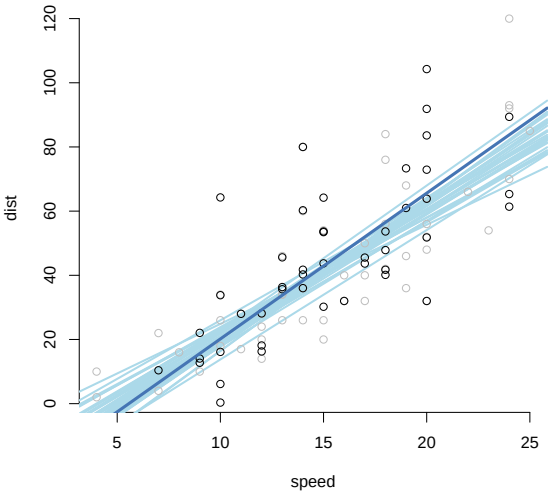
But an alternative can be to use residuals obtained from $Y_i = \underbrace{\boldsymbol{X}_i^\mathsf{T}\widehat{\boldsymbol{\beta}}}_{\widehat{Y}_i} + \widehat{\varepsilon}_i$.

Resampling Techniques in Regression type Models

2. Draw pseudo-observations using (estimated) residuals

Resample from $-(\boldsymbol{X}_i, \widehat{Y}_i + \varepsilon)''$, where ε can be obtained by resampling from (estimated) residuals

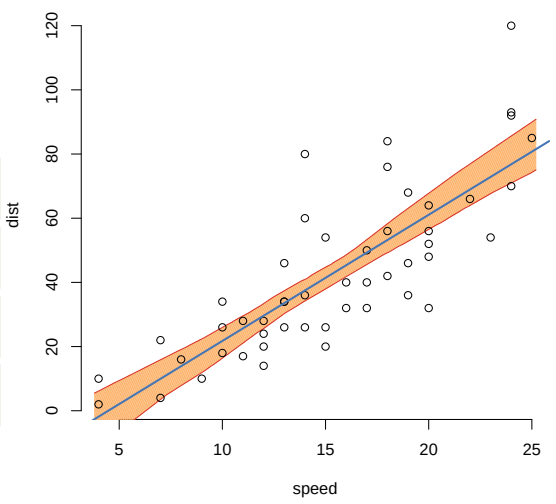
```
> V=matrix(NA,100,251)
> for(i in 1:100){
+ ind <- sample(1:50,replace=TRUE)
+ cars2 <- data.frame(dist=predict(reg),
    speed=cars$speed)
+ cars_b <- cars2[ind,]
+ cars_b$dist <- cars_b$dist+residuals(reg)[
    sample(1:50,replace=TRUE)]
+ V[i,] <- predict(lm(dist~speed,data=cars_b
    ),newdata=data.frame(speed=u))}
```



Resampling Techniques in Regression type Models

Here again, pointwise confidence intervals can be obtained

```
> q1 <- apply(V,2,function(x) quantile(x
    ,.05))
> q2 <- apply(V,2,function(x) quantile(x
    ,.95))
```



Those values can be compared with the second and the third column of

```
> v <- predict(reg,newdata=data.frame(speed=u),interval="confidence")
```

Changing the Distance in Least-Squares?

One might consider $\widehat{\boldsymbol{\beta}} \in \text{argmin}-\sum_{i=1}^{n} |Y_i - \boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta}|$ ", based on the ℓ_1-norm, and not the ℓ_2-norm.

This is the least-absolute deviation estimator, related to the median regression, since median$(X) = \text{argmin}-\mathbb{E}|X - x|$".

💡 this regression model was introduced 50 years before Gauss and Legendre ℓ_2-regression, by Boscovich.

⚠ no analytical expression, since the ℓ_1-norm is not differentiable (first order condition).

More generally, assume that, for some function $R(\cdot)$,

$$\widehat{\boldsymbol{\beta}} \in \text{argmin}-\sum_{i=1}^{n} R(Y_i - \boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta}) \text{ "}$$

Changing the Distance in Least-Squares?

If R is differentiable, the first order condition would be

$$\sum_{i=1}^{n} R'\left(Y_i - \boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta}\right) \cdot \boldsymbol{X}_i^\mathsf{T} = 0.$$

i.e.

$$\sum_{i=1}^{n} \underbrace{\omega\left(Y_i - \boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta}\right)}_{\omega_i} \cdot \left(Y_i - \boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta}\right)\boldsymbol{X}_i^\mathsf{T} = 0 \text{ with } \omega(x) = \frac{R'(x)}{x},$$

It is the first order condition of a weighted ℓ_2 regression.

⚠ weights are unknown, since they are function of residuals...

To obtain the ℓ_1-regression, observe that $\omega = |\varepsilon|^{-1}$

Changing the Distance in Least-Squares?

💡 use iterative (weighted) least-square regressions.

Start with some standard ℓ_2 regression

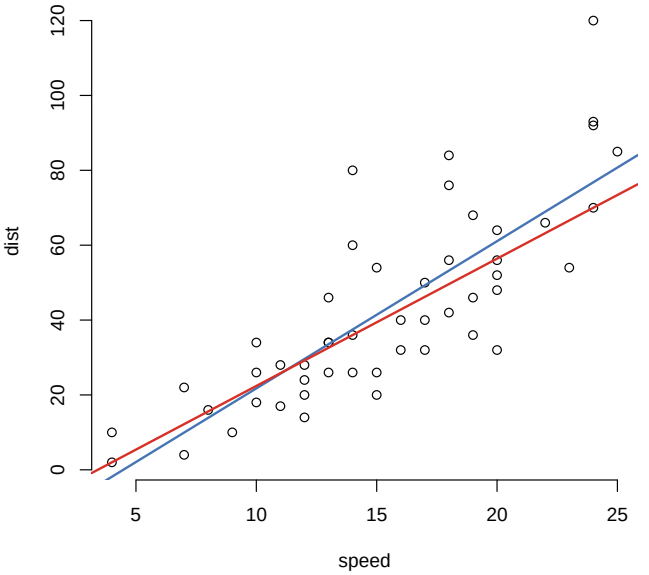
```
> reg_0 <- lm(Y~X, data=db)
```

For the ℓ_1 regression consider weight function

```
> omega <- function(e) 1/abs(e)
```

Then consider the following iterative algorithm

```
> resid <- residuals(reg_0)
> for(i in 1:100){
+ W <- omega(e)
+ reg <- lm(Y~X, data=db, weights=W)
+ e <- residualts(reg)}
```



Under- and Over-Identified Models

Suppose that the true model is

$$Y_i = \boldsymbol{X}_{1,i}\boldsymbol{\beta}_1 + \boldsymbol{X}_{2,i}\boldsymbol{\beta}_2 + \varepsilon_i,$$

but we estimate the model on $\boldsymbol{X}_1$ (only)

$$Y_i = \boldsymbol{X}_{1,i}\boldsymbol{b}_1 + \eta_i.$$

$$\begin{aligned}
\widehat{\boldsymbol{b}}_1 &= (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}Y \\
&= (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}[\boldsymbol{X}_{1,i}\boldsymbol{\beta}_1 + \boldsymbol{X}_{2,i}\boldsymbol{\beta}_2 + \varepsilon] \\
&= (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1\boldsymbol{\beta}_1 + (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_2\boldsymbol{\beta}_2 + (\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\varepsilon \\
&= \boldsymbol{\beta}_1 + \underbrace{(\boldsymbol{X}_1'\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_2\boldsymbol{\beta}_2}_{\boldsymbol{\beta}_{12}} + \underbrace{(\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1)^{-1}\boldsymbol{X}_1^\mathsf{T}\varepsilon_i}_{\nu_i}
\end{aligned}$$

i.e. $\mathbb{E}(\widehat{\boldsymbol{\alpha}}_1) = \boldsymbol{\beta}_1 + \boldsymbol{\beta}_{12}$.

⚠ if $\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_2 = \boldsymbol{0}$ ($\boldsymbol{X}_1 \perp \boldsymbol{X}_2$), $\mathbb{E}(\widehat{\boldsymbol{b}}_1) = \boldsymbol{\beta}_1$

Under-Identified Models, Simpson's paradox

Computer $\mathbb{E}(Y|X_1)$ where Y is the hospital

hospital	total	survivors	dead	survival rate	
hospitall A	1,000	800	200	80%	
hospitall B	1,000	900	100	90%	$\star$

For healthy people $\mathbb{E}(Y|X_1, X_2 = \text{healthy})$

hospital	total	survivors	dead	survival rate	
hospital A	600	590	10	98%	$\star$
hospital B	900	870	30	97%	

For sick people $\mathbb{E}(Y|X_1, X_2 = \text{sick})$

hospital	total	survivors	dead	survival rate	
hospital A	400	210	190	53%	$\star$
hospital B	100	30	70	30%	

Under- and Over-Identified Models

Conversely, assume that the true model is

$$Y = \boldsymbol{X}_1\boldsymbol{\beta}_1 + \varepsilon$$

but we estimated on (unnecessary) variables $\boldsymbol{X}_2$

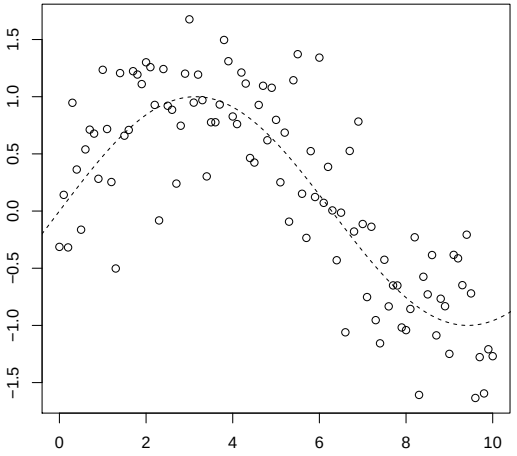
$$Y = \boldsymbol{X}_1\boldsymbol{b}_1 + \boldsymbol{X}_2\boldsymbol{b}_2 + \eta$$

Here estimation is unbiased $\mathbb{E}(\boldsymbol{b}_1) = \boldsymbol{\beta}_1$, but the estimator is not efficient...

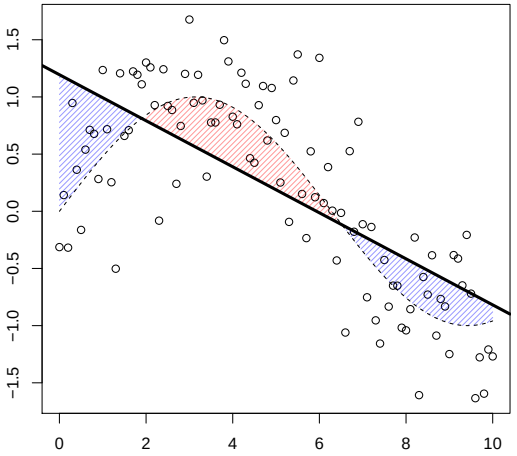
Beyond Linear Models: Polynomial

Even if $m(x) = \mathbb{E}(Y|X = x)$ is not a polynomial function, a polynomial can still be a good approximation.

From Stone-Weierstrass theorem, if $m(\cdot)$ is continuous on some interval, then there is a uniform approximation of $m(\cdot)$ by polynomial functions.



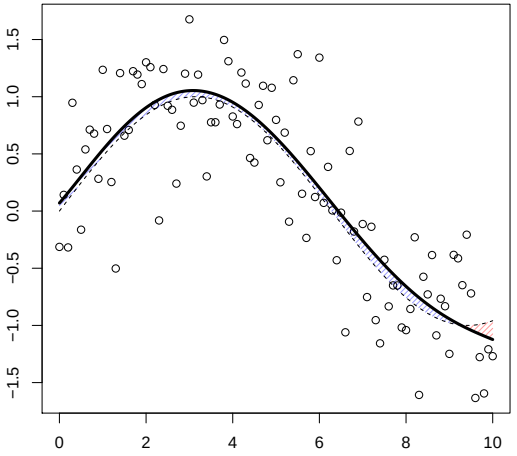

```
set.seed(1)
n=10
xr <- seq(0,n,by=.1)
yr <- sin(xr/2)+rnorm(length(xr))/2
db <- data.frame(x=xr,y=yr)
plot(db)
lines(xr,sin(xr/2),lty=2)
```

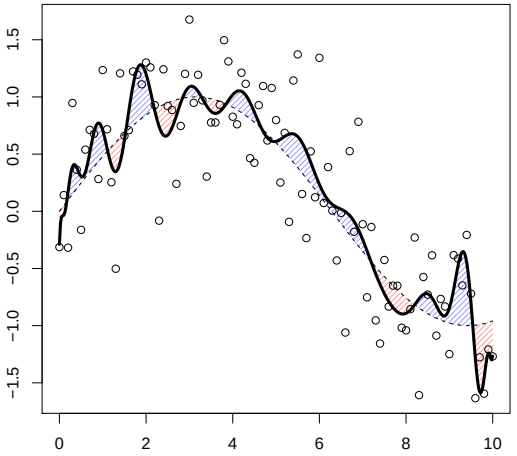


Beyond Linear Models: Polynomial

Assume that $m(x) = \mathbb{E}(Y|X=x) = \sum_{i=0}^{k} \alpha_i x^i$, where parameters $\alpha_0, \cdots, \alpha_k$ will be estimated (but not k).

```
reg <- lm(yr~poly(xr,5),data=db)
reg <- lm(yr~poly(xr,25),data=db)
```





Beyond Linear Models: Local Regression

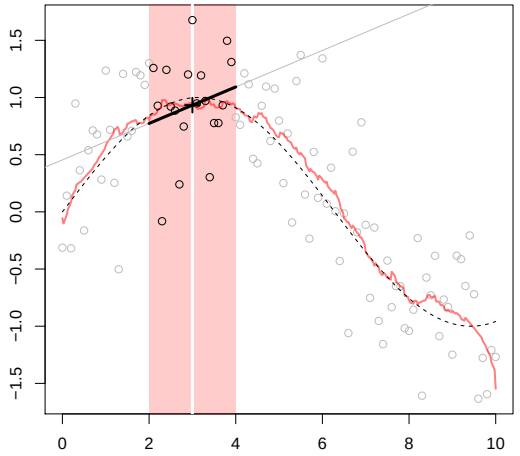
(local) regression on the subset

$$-i, |X_i - x| \leq h''$$

```
local_reg <- function(x,h){
ind <- which(abs(X-x)<h)
reg_loc <- lm(Y~X, data=db[ind,])
return(preduct(reg_loc,newdata=data.frame(X=x))) }
```

or

```
local_reg <- function(x,h){
W <- abs(X-x)<h
reg_loc <- lm(Y~X, data=db, weights=W)
return(preduct(reg_loc,newdata=data.frame(X=x))) }
```



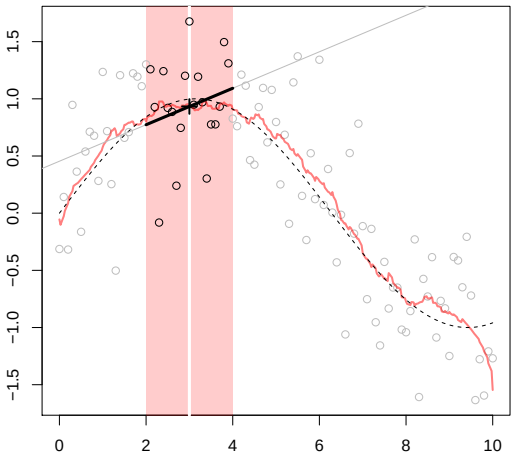
Beyond Linear Models: Local Regression

why not run a weighted linear regression with

$$\omega_i = K_h(X_i - x) = \frac{1}{h} K\left(\frac{X_i - x}{h}\right)$$

for some kernel $K(\cdot)$.

```
local_reg <- function(x,h,K){
W <- K((X-x)/h)/h
reg_loc <- lm(Y~X, data=db, weights=W)
return(preduct(reg_loc,newdata=data.frame(X=x))) }
```



One can also consider regression on the k-nearest neighbors

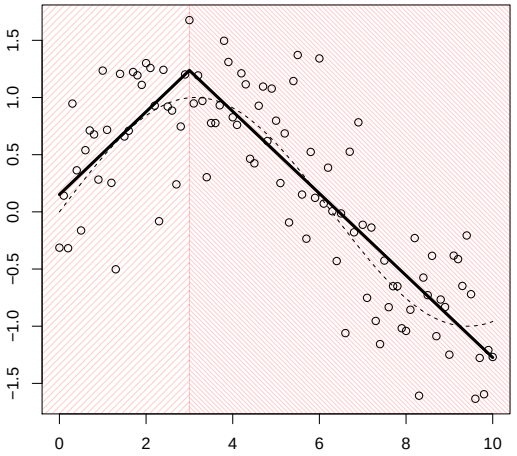
```
local_reg <- function(x,k){
D <- rank(abs(X-x))
reg_loc <- lm(Y~X, data=db[which(D<=k),])
return(preduct(reg_loc,newdata=data.frame(X=x))) }
```

Beyond Linear Models: Using Splines

for linear splines, consider

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 (X_i - s)_+ + \varepsilon_i$$

```
positive_part <- function(x) ifelse(x>0,x,0)
reg <- lm(Y~X+positive_part(X-s), data=db)
```

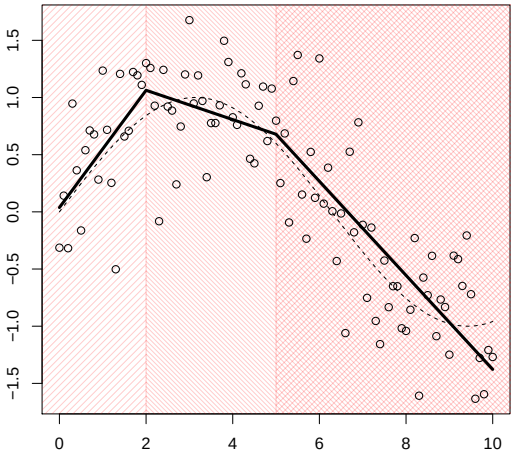


Beyond Linear Models: Using Splines

💡 for linear splines, consider

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 (X_i - s_1)_+ + \beta_3 (X_i - s_2)_+ + \varepsilon_i$$

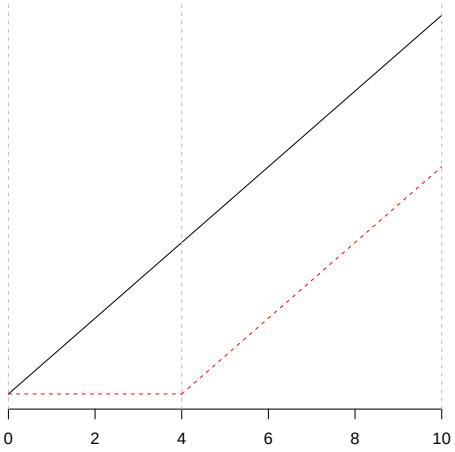
```
reg <- lm(Y~X+positive_part(X-s1)+positive_part(X-
    s2), data=db)
```

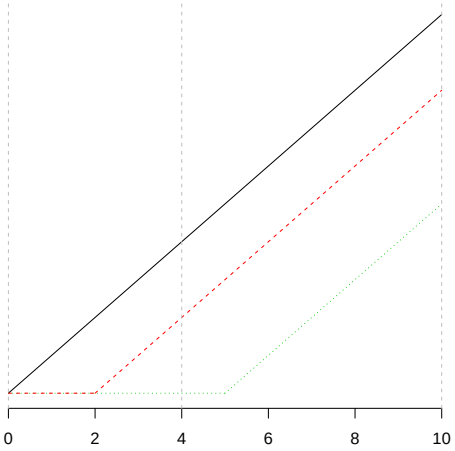


Beyond Linear Models: Using Splines

⚠ the model estimated by R is an equivalent one, see

```
> library(splines)
> B1_1 <- bs(x,knots=4,Boundary.Knots=range(x),
    degre=1)
> B2_1 <- bs(x,knots=c(2,5),Boundary.Knots=range(x
    ),degre=1)

> B1_2 <- bs(x,knots=4,Boundary.Knots=range(x),
    degre=2)
> B2_2 <- bs(x,knots=c(2,5),Boundary.Knots=range(x
    ),degre=2)
```





Smoothing Techniques, a Unified Approach

In the linear model,

$$\widehat{\boldsymbol{Y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}} = \underbrace{\boldsymbol{X}[\boldsymbol{X}^\mathsf{T}\boldsymbol{X}]^{-1}\boldsymbol{X}^\mathsf{T}}_{\boldsymbol{H}}\boldsymbol{Y}$$

⚠ $\boldsymbol{H}_{i,i}$ is the leverage of the ith element of this hat matrix.

Write

$$\widehat{Y}_i = \sum_{j=1}^{n}[\boldsymbol{X}_i^\mathsf{T}[\boldsymbol{X}^\mathsf{T}\boldsymbol{X}]^{-1}\boldsymbol{X}^\mathsf{T}]_j Y_j = \sum_{j=1}^{n}[\mathcal{H}(\boldsymbol{X}_i)]_j Y_j$$

where

$$\mathcal{H}(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}[\boldsymbol{X}^\mathsf{T}\boldsymbol{X}]^{-1}\boldsymbol{X}^\mathsf{T}$$

The prediction is

$$m(\boldsymbol{x}) = \mathbb{E}(Y|\boldsymbol{X}=\boldsymbol{x}) = \sum_{j=1}^{n}[\mathcal{H}(\boldsymbol{x})]_j Y_j$$

Smoothing Techniques, a Unified Approach

More generally, a predictor m is said to be linear if for all $\boldsymbol{x}$ if there is $\mathcal{S}(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$m(\boldsymbol{x}) = \sum_{j=1}^{n} \mathcal{S}(\boldsymbol{x})_j Y_j$$

Conversely, given $\widehat{Y}_1, \cdots, \widehat{Y}_n$, there is a matrix $\boldsymbol{S}$ $n \times n$ such that

$$\widehat{\boldsymbol{Y}} = \boldsymbol{S}\boldsymbol{Y}$$

For the linear model, $\boldsymbol{S} = \boldsymbol{H}$.

⚠ $\text{trace}(\boldsymbol{H}) = \dim(\boldsymbol{\beta})$: degrees of freedom

⚠ $\dfrac{\boldsymbol{H}_{i,i}}{1 - \boldsymbol{H}_{i,i}}$ is related to Cook's distance (detect outliers)

Smoothing Techniques, a Unified Approach

For a kernel regression model, with kernel K and bandwidth h

$$S_{i,j}^{(K,h)} = \frac{K_h(X_i - X_j)}{\sum_{k=1}^{n} K_h(X_k - X_j)}$$

where $K_h(\cdot) = K(\cdot/h)$, while

$$\mathcal{S}^{(K,h)}(\boldsymbol{x})_j = \frac{K_h(\boldsymbol{x} - X_j)}{\sum_{k=1}^{n} K_h(X_k - \boldsymbol{x})}$$

Smoothing Techniques, a Unified Approach

For a k-nearest neighbor,

$$S_{i,j}^{(k)} = \frac{1}{k}\mathbf{1}(j \in \mathcal{I}_{\boldsymbol{X}_i})$$

where $\mathcal{I}_{\boldsymbol{X}_i}$ are the k nearest observations to $\boldsymbol{X}_i$, while

$$\mathcal{S}^{(K,h)}(\boldsymbol{x})_j = \frac{1}{k}\mathbf{1}(j \in \mathcal{I}_{\boldsymbol{x}})$$

Testing for Linearity

Observe that trace(S) is usually seen as a degree of smoothness.

Do we have to smooth? Isn't linear model sufficent?

Define

$$T = \frac{\|\boldsymbol{SY} - \boldsymbol{HY}\|}{\text{trace}([\boldsymbol{S} - \boldsymbol{H}]^{\mathsf{T}}[\boldsymbol{S} - \boldsymbol{H}])}$$

If the model is linear, then T has a Fisher distribution.

On Optimal Smoothing

- local regression and kernel regression: choice of bandwidth h
- splines: choice of number and values of knots $s_1, \cdots, s_k$

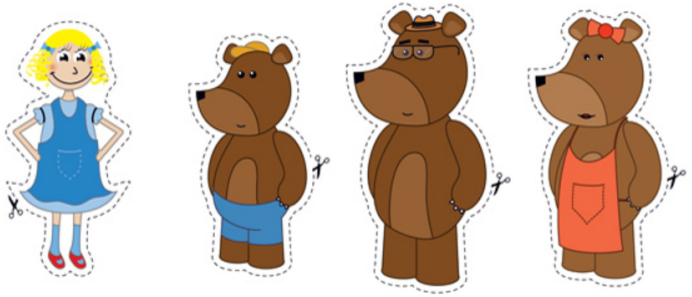
$$R_h = \mathbb{E}\left([Y - \widehat{m}_h(\boldsymbol{X})]^2\right) + \sigma^2 + \underbrace{\mathbb{E}\left([m(\boldsymbol{x}) - \widehat{m}_h(\boldsymbol{x})]^2\right)}_{\text{Mean Squared Error}}$$

We can rewrite the Mean Squared Error as

$$MSE_h(\boldsymbol{x}) = \int \text{bias}(\widehat{m}_h(\boldsymbol{x}))^2 d\boldsymbol{x} + \int \text{Var}(\widehat{m}_h(\boldsymbol{x})) d\boldsymbol{x}$$

where $\text{bias}(\widehat{m}_h(\boldsymbol{x}) = \mathbb{E}\left(\widehat{m}_h(\boldsymbol{x}) - m(\boldsymbol{x})\right)$ and $\text{Var}(\widehat{m}_h(\boldsymbol{x}) = \text{Var}(\widehat{m}_h(\boldsymbol{x})$.

Usual bias-variance tradeoff, or Goldilock principle: h should be neither too small, nor too large



- undersmoothed: bias too large, variance too small
- oversmoothed: variance too large, bias too small

The empirical version is

$$\widehat{R}_h = \frac{1}{n}\sum_{1=n}(Y_i - \widehat{m}_h(\boldsymbol{X}_i))^2$$

Use Leave-one-out Cross Validation

$$\widehat{R}_h = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{m}_{h(-i)}(\boldsymbol{X}_i))^2$$

where $\widehat{m}_{h(-i)}(\cdot)$ is the estimator obtained by omitting the ith observation.

Optimal Smoothing, Nadaraya-Watson Kernel estimator

Use a rectangular kernel $K(x) = \frac{1}{2}\mathbf{1}(|x| \leq 1)$ or a Gaussian one

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

💡 the choice of the kernel is not (that) important,

but the bandwidth is much more important.

× Understanding the Variance and Bias Tradeoff

Let us generate new samples, to understand this tradeoff

```
> nsim <- 500
> mat_sample_y <- matrix(sin(xr/2)+rnorm(n*nsim)/2,n,nsim)
```

For each x, we now have 500 $\widehat{m}_s(x)$

```
vec_mh=function(x,h){
sapply(1:nsim,function(s){
```

```
db_s=data.frame(yr=mat_sample_y[,s],xr=xr)
reg_s = lm(yr ~ 1,weights=dnorm(xr,mean=x,sd=h),data=db_s)
predict(reg_s,newdata=data.frame(xr=x))})}
```

Based on those values, we can approximate $\int \text{bias}^2(\widehat{m}_h(\boldsymbol{x}))d\boldsymbol{x}$

```
bias2_h=function(h) mean(Vectorize(function(x)  mean((vec_mh(x,h)-m(x
    ))^2))(xr))
```

and $\int \text{Var}(\widehat{m}_h(\boldsymbol{x}))d\boldsymbol{x}$

```
var_h=function(h) mean(Vectorize(function(x) var(vec_mh(x,h)))(xr))
```

Define finally

```
mise_h=function(h)  bias2_h(h)+var_h(h)
```

- **Cross Validation to estimate $\widehat{R}_h$**

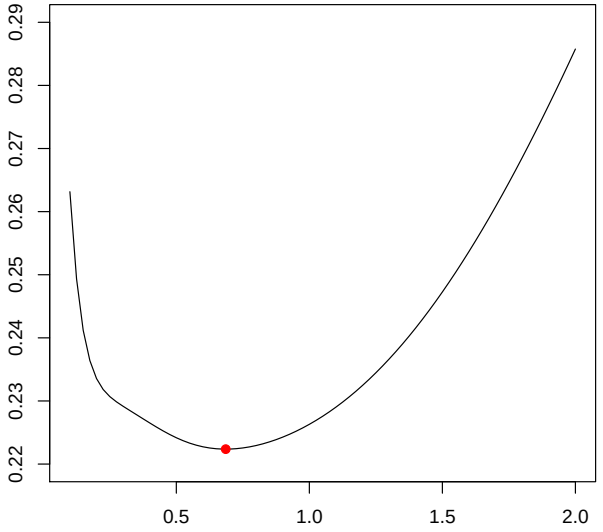
In real life, we cannot generate 500 other samples.

```
pred_i =function(i,h){
x=db$xr[i]
reg_i = lm(yr ~ 1,weights=dnorm(xr,mean=x,sd=h),data=db,subset=(1:
    nrow(db))[-i])
predict(reg_i,newdata=data.frame(xr=x)) }
```

```
mse_h=function(h){
mean((db$yr-Vectorize(function(i) pred_i(i,h))(1:n))^2) }
```

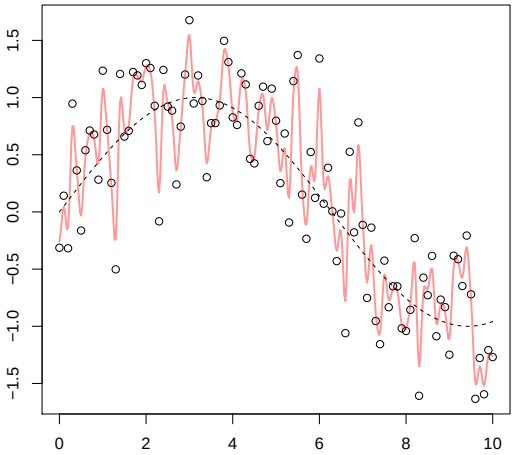
The optimal bandwidth parameter $h^\star$ is then

```
> optimize(mse_h,interval=c(.1,2))
$minimum
[1] 0.6857169

$objective
[1] 0.2223652
```



Optimal Bandwidth $h^\star$

```
library(np)
h <- 1
nw <- npreg(yr~xr, data=db, bws=h, ckertype="
    gaussian")
u=seq(0,10,by=.025)
v=predict(nw,newdata=data.frame(xr=u))
lines(u,v,col=rgb(1,0,0,.4),lwd=2)
```



× Theoretical Results (Briefly)

One can prove that, as $n \to \infty$ and $nh \to 0$, R_h can be approximated by

$$\frac{h^4}{4}\left(\int x^2 K(\boldsymbol{x})d\boldsymbol{x}\right)^2 \int \left(m''(\boldsymbol{x}) + 2m'(\boldsymbol{x})\frac{f'(\boldsymbol{x})}{f(\boldsymbol{x})}\right) d\boldsymbol{x} + \frac{1}{nh}\sigma^2 \int K^2(x)dx \int \frac{d\boldsymbol{x}}{f(\boldsymbol{x})}$$

where f is the density of $\boldsymbol{x}$'s. Thus the optimal h is

$$h^\star = n^{-\frac{1}{5}} \left(\frac{\sigma^2 \int K^2(x)dx \int \frac{d\boldsymbol{x}}{f(\boldsymbol{x})}}{\left(\int x^2 K(\boldsymbol{x})d\boldsymbol{x}\right)^2 \int \left(\int m''(\boldsymbol{x}) + 2m'(\boldsymbol{x})\frac{f'(\boldsymbol{x})}{f(\boldsymbol{x})}\right)^2 d\boldsymbol{x}} \right)^{\frac{1}{5}}$$

(hard to get a simple rule of thumb... up to a constant, $h^\star \sim n^{-\frac{1}{5}}$)

One can also use (local polynomial)

```
> library(locfit)
```

but it is not the same smoothing technique. But here also, cross validation techniques can be used to get the optimal smoothing parameter.

Cross-Validation

randomly split the sample into k chuncks of (approximately) equal size, and then compute the classifier estimated with chunk κ and predict data in chunk κ.

```
library(boot)
model <- glm(y~x,data=db)
cv.glm(db,model,K=10)
```

for some 10-fold cross validation. When omiting, it is the standard leave-one-out cross validation technique.

```
cv.glm(db,model)
```

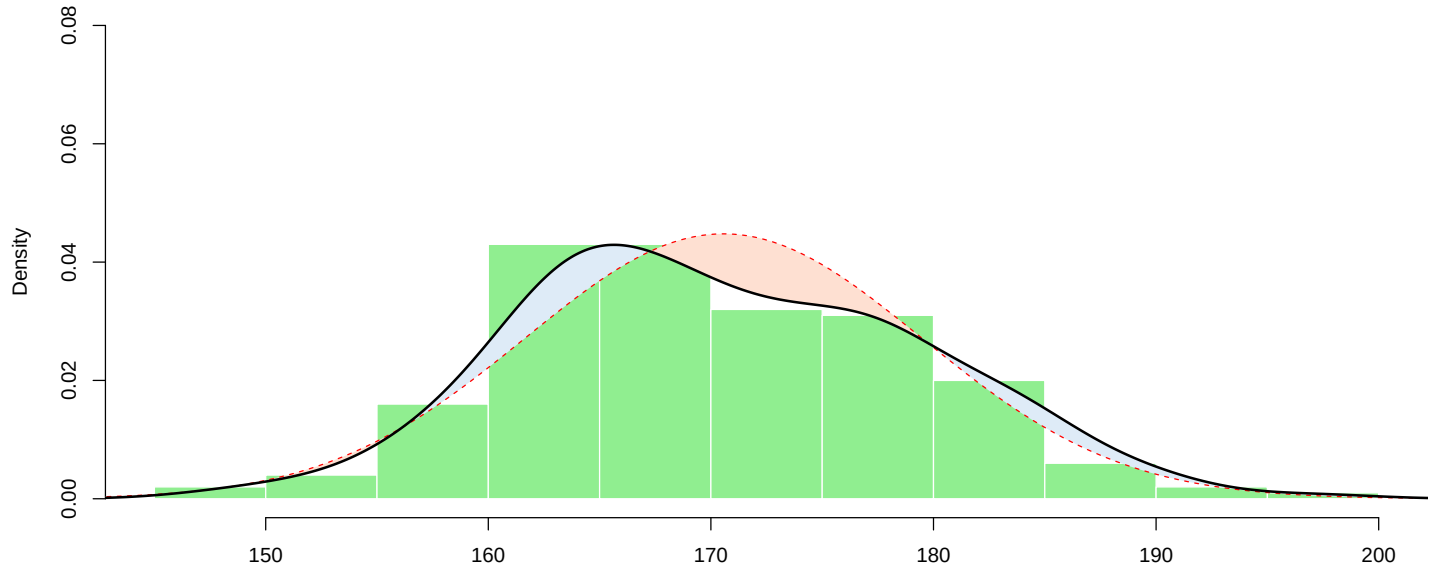
k-Fold Cross-Validation is faster to run.

Unsupervised vs. Supervised Techniques

The difference between mixture and regression models is simple

Let $-z_1, \cdots, z_n$" be a sample with the height of some insured.

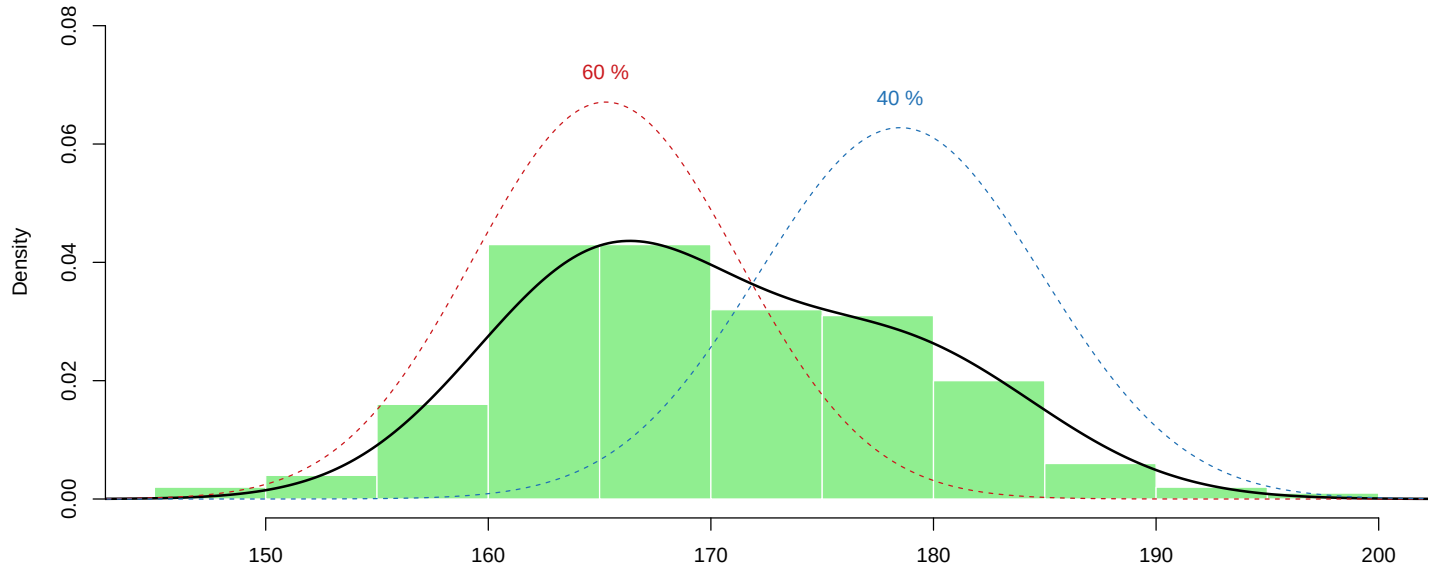
Observe that $Z \not\sim \mathcal{N}(\mu, \sigma^2)$



Unsupervised vs. Supervised Techniques

Why not a mixture model for $-x_i$"

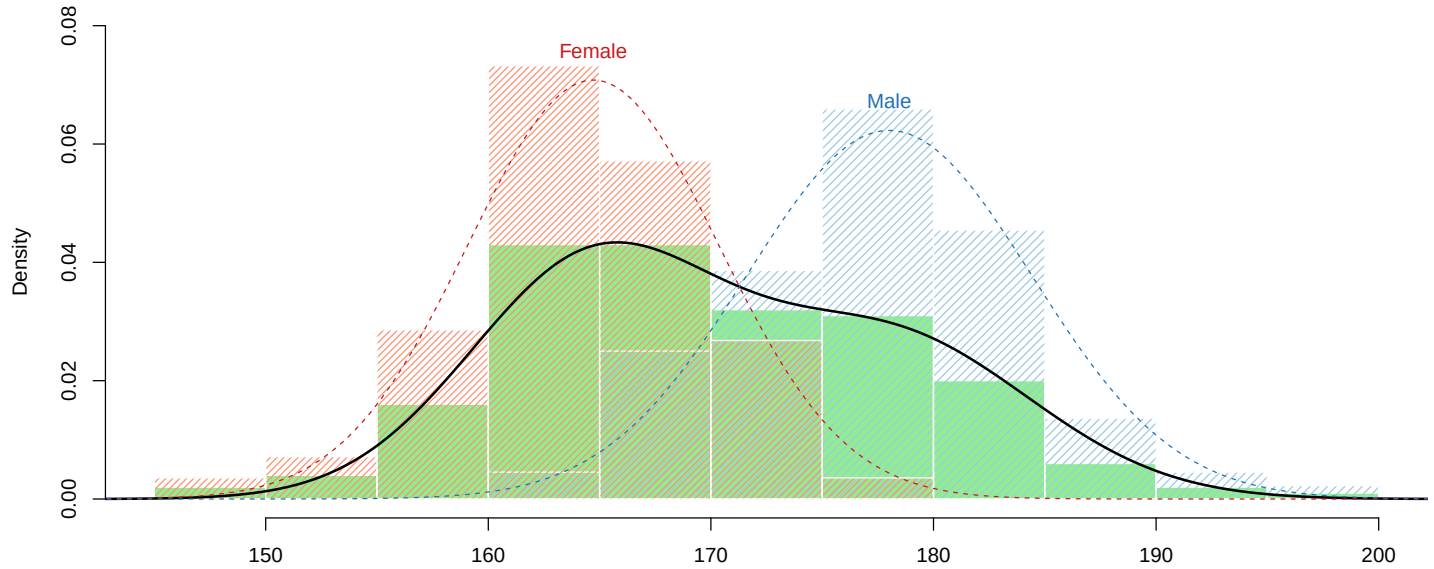
$$X \sim - \begin{cases} \mathcal{N}(\mu_1, \sigma_1^2) \text{ with probability } p_1, \text{ i.e. with a latent variable } \Theta = 1 \\ \mathcal{N}(\mu_2, \sigma_2^2) \text{ with probability } p_2, \text{ i.e. with a latent variable } \Theta = 2 \end{cases}$$



Unsupervised vs. Supervised Techniques

Why not a regression model for $-y_i, x_i"$

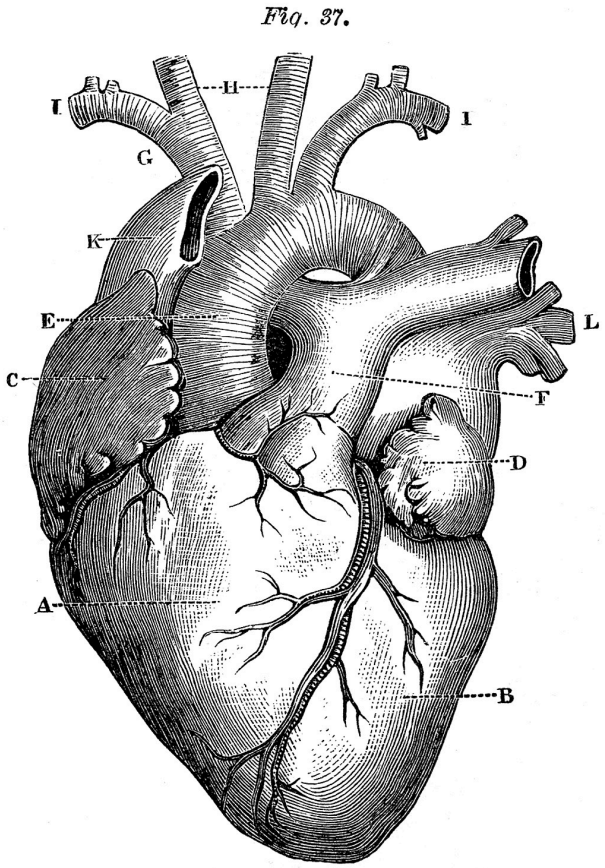
$$Y \sim - \begin{cases} \mathcal{N}(\mu_M, \sigma_M^2) \text{ when } X = \text{'male'}, \text{ with probability } p_M \\ \mathcal{N}(\mu_F, \sigma_F^2) \text{ when } X = \text{'female'}, \text{ with probability } p_F \end{cases}$$



Classification: Description of the Dataset

Myocardial infarction of patients admited in E.R.

- heart rate (FRCAR),
- heart index (INCAR)
- stroke index (INSYS)
- diastolic pressure (PRDIA)
- pulmonary arterial pressure (PAPUL)
- ventricular pressure (PVENT)
- lung resistance (REPUL)
- death or survival



Classification: Description of the Dataset

```
> myocarde=read.table("myocarde.csv",head=TRUE,sep=";")
> head(myocarde)
  FRCAR INCAR INSYS PRDIA PAPUL PVENT REPUL   PRONO
1    90  1.71  19.0    16  19.5  16.0   912 SURVIE
2    90  1.68  18.7    24  31.0  14.0  1476  DECES
3   120  1.40  11.7    23  29.0   8.0  1657  DECES
4    82  1.79  21.8    14  17.5  10.0   782 SURVIE
5    80  1.58  19.7    21  28.0  18.5  1418  DECES
6    80  1.13  14.1    18  23.5   9.0  1664  DECES
```

Training and Validation Samples

⚠ in those slides, to present the technique, we fit the model on the complete dataset

To validate a model, we usually split the sample in two(or three)

- a sample to estimate the model (training)
- a sample used for cross-validation (if necessary)
- a sample to validate the model (validation)

⚠ In practice, we allocate as follows

- $\sim 50\%, 60\%, 80\%$ of sample to estimate the model
- $\sim 50\%, 40\%, 20\%$ of sample to validate the model

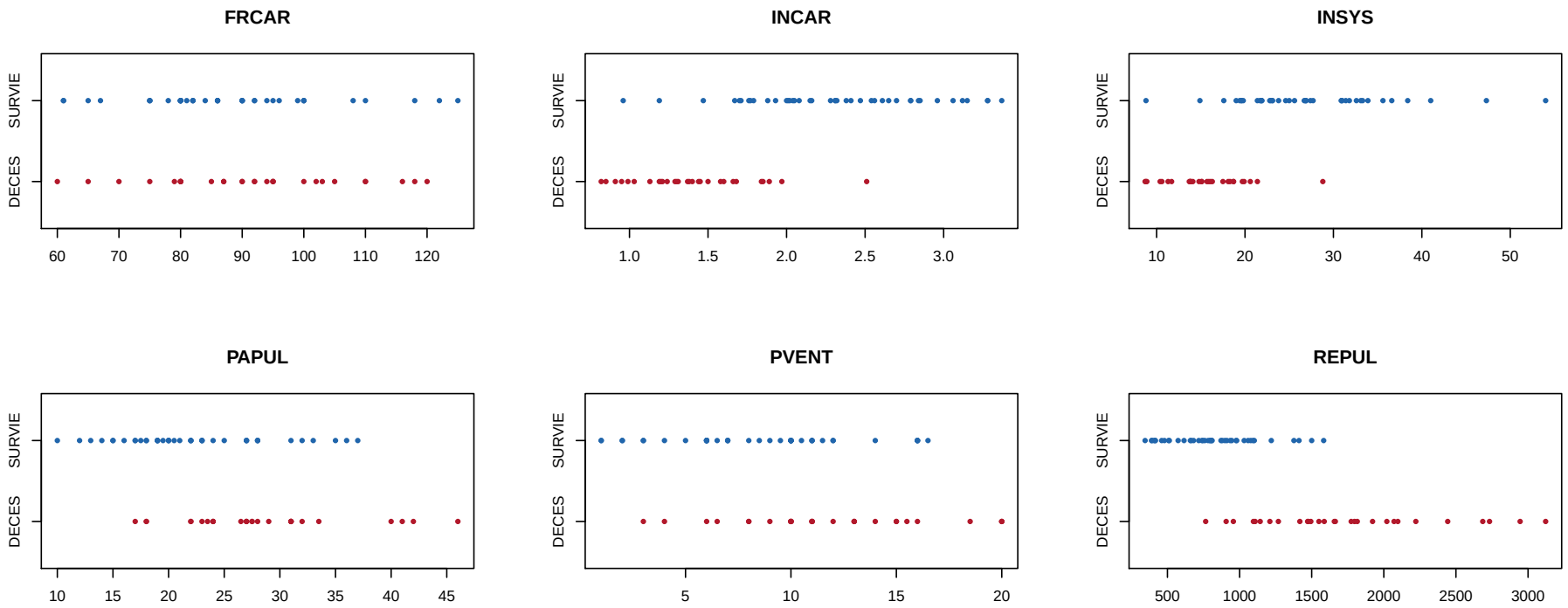
Generating Samples

```
splitdf <- function(dataframe, ratio=.5, seed=NULL) {
 if (!is.null(seed)) set.seed(seed)
index <- 1:nrow(dataframe)
trainindex <- sample(index, trunc(length(index)*ratio))
trainset <- dataframe[trainindex, ]
testset <- dataframe[-trainindex, ]
return(list(trainset=trainset,testset=testset))}
```

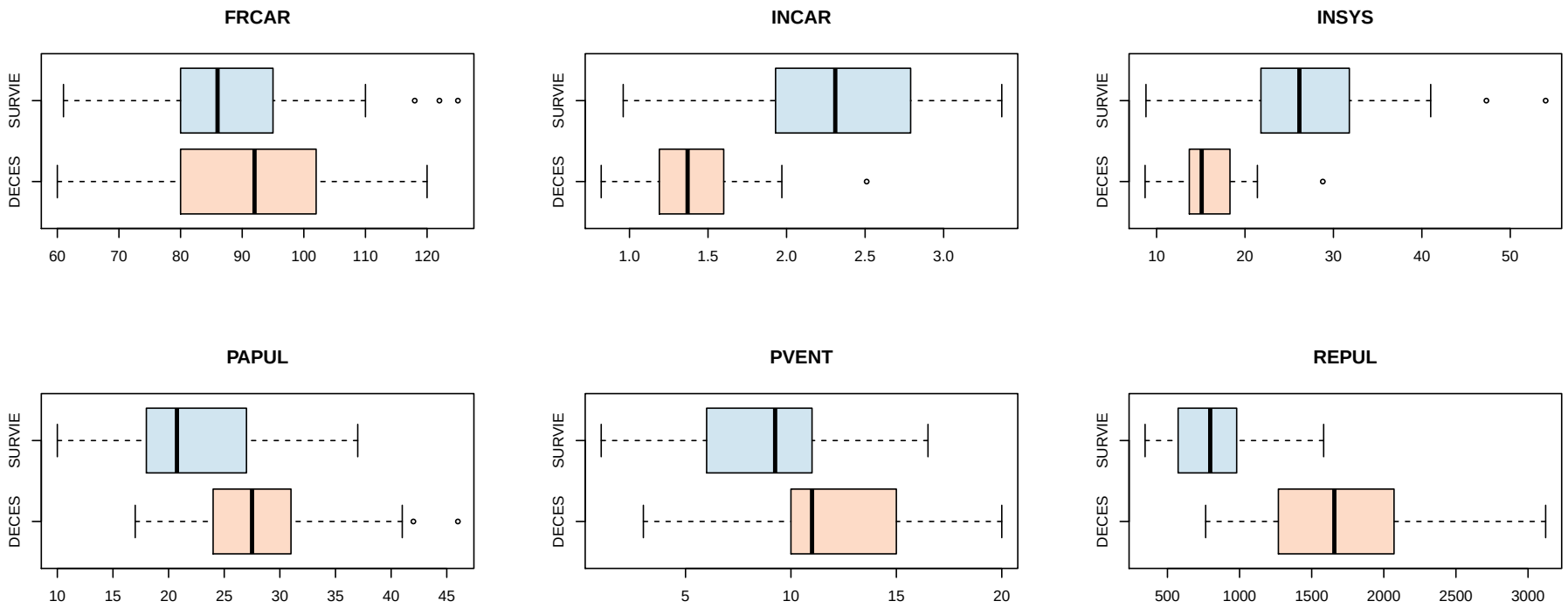
There are (at least) two generators of random numbers in SAS see sas.com

- Fishman and Moore (1982, jstor.org) used for function RANUNI
- Mersenne-Twister used for the RAND function, based on Matsumoto and Nishimura (1997, math.sci.hiroshima-u.ac.jp)

Modeling a 0/1 Variable



Modeling a 0/1 Variable



Logistic Regression

```
> logistic <- glm(PRONO~.  , data=myocarde, family=binomial)
> summary(logistic)

Call:
glm(formula = PRONO ~ ., family = binomial, data = myocarde)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.24911  -0.41613   0.05261   0.39611   2.25781

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -10.187642  11.895227  -0.856    0.392
FRCAR         0.138178   0.114112   1.211    0.226
INCAR        -5.862429   6.748785  -0.869    0.385
INSYS         0.717084   0.561445   1.277    0.202
PRDIA        -0.073668   0.291636  -0.253    0.801
```

```
PAPUL          0.016757    0.341942    0.049     0.961
PVENT         -0.106776    0.110550   -0.966     0.334
REPUL         -0.003154    0.004891   -0.645     0.519

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 96.033  on 70  degrees of freedom
Residual deviance: 41.043  on 63  degrees of freedom
AIC: 57.043

Number of Fisher Scoring iterations: 7
```

Logistic Regression

```
> logistic <- glm(PRONO=="DECES"~.  , data=myocarde, family=binomial)
> summary(logistic)

Call:
glm(formula = PRONO == "DECES" ~ ., family = binomial, data =
    myocarde)

Deviance Residuals:
     Min       1Q   Median       3Q      Max
-2.25781  -0.39611  -0.05261   0.41613   2.24911

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 10.187642  11.895227   0.856    0.392
FRCAR       -0.138178   0.114112  -1.211    0.226
INCAR        5.862429   6.748785   0.869    0.385
INSYS       -0.717084   0.561445  -1.277    0.202
```

```
PRDIA          0.073668    0.291636    0.253      0.801
PAPUL         -0.016757    0.341942   -0.049      0.961
PVENT          0.106776    0.110550    0.966      0.334
REPUL          0.003154    0.004891    0.645      0.519

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 96.033  on 70  degrees of freedom
Residual deviance: 41.043  on 63  degrees of freedom
AIC: 57.043

Number of Fisher Scoring iterations: 7
```

Multiple Logistic Regression

```
> library(VGAM)
> mlogistic <- vglm(PRONO~.  , data=myocarde, family=multinomial)
> levels(myocarde$PRONO)
[1] "DECES"  "SURVIE"
> summary(mlogistic)

Call:
vglm(formula = PRONO ~ ., family = multinomial, data = myocarde)

Pearson residuals:
                              Min       1Q   Median       3Q      Max
log(mu[,1]/mu[,2]) -3.4334 -0.28577 -0.03718 0.3007 3.3966

Coefficients:
              Estimate Std. Error   z value
(Intercept) 10.1876411 11.8941581  0.856525
FRCAR       -0.1381781  0.1141056 -1.210967
```

```
INCAR          5.8624289   6.7484319   0.868710
INSYS         -0.7170840   0.5613961  -1.277323
PRDIA          0.0736682   0.2916276   0.252610
PAPUL         -0.0167565   0.3419255  -0.049006
PVENT          0.1067760   0.1105456   0.965901
REPUL          0.0031542   0.0048907   0.644939

Number of linear predictors:  1

Name of linear predictor: log(mu[,1]/mu[,2])

Dispersion Parameter for multinomial family:    1

Residual deviance: 41.04314 on 63 degrees of freedom

Log-likelihood: -20.52157 on 63 degrees of freedom

Number of iterations: 7
```

Fisher's Gradient Descent

The algorithm to estimate $\widehat{\boldsymbol{\beta}}$ is

1. start with some initial value $\boldsymbol{\beta}_0$
2. define $\boldsymbol{\beta}_k = \boldsymbol{\beta}_{k-1} - H(\boldsymbol{\beta}_{k-1})^{-1}\nabla \log \mathcal{L}(\boldsymbol{\beta}_{k-1})$

where $\nabla \log \mathcal{L}(\boldsymbol{\beta})$is the gradient, qnd $H(\boldsymbol{\beta})$ the Hessian matrix, also called Fisher's score.

The generic term of the Hessian is

$$\frac{\partial^2 \log \mathcal{L}(\boldsymbol{\beta})}{\partial \beta_k \partial \beta_\ell} = \sum_{i=1}^{n} X_{k,i} X_{\ell,i} [y_i - \pi_i(\boldsymbol{\beta})]$$

Define $\boldsymbol{\Omega} = [\omega_{i,j}] = \text{diag}(\widehat{\pi}_i(1-\widehat{\pi}_i))$ so that the gradient is writen

$$\nabla \log \mathcal{L}(\boldsymbol{\beta}) = \frac{\partial \log \mathcal{L}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{X}'(\boldsymbol{y} - \boldsymbol{\pi})$$

and the Hessian

$$H(\boldsymbol{\beta}) = \frac{\partial^2 \log \mathcal{L}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}'} = -\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{X}$$

The gradient descent algorithm is then

$$\boldsymbol{\beta}_k = (\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{Z} \text{ where } \boldsymbol{Z} = \boldsymbol{X}\boldsymbol{\beta}_{k-1} + \boldsymbol{X}'\boldsymbol{\Omega}^{-1}(\boldsymbol{y} - \boldsymbol{\pi}),$$

```
> beta=as.matrix(lm(Y~0+X)$coefficients,ncol=1)
> for(s in 1:9){
+    pi=exp(X%*%beta[,s])/(1+exp(X%*%beta[,s]))
+    gradient=t(X)%*%(Y-pi)
+    omega=matrix(0,nrow(X),nrow(X));diag(omega)=(pi*(1-pi))
+    Hessian=-t(X)%*%omega%*%X
+    beta=cbind(beta,beta[,s]-solve(Hessian)%*%gradient)}
> beta
                    [,1]        [,2]        [,3]       [,4]       [,5]
XConstant   0.05172274 -2.4401950 -1.2761448  2.9062584  7.9451328
XFRCAR      0.00316378  0.0113589 -0.0170704 -0.0723398 -0.1212657
```

XINCAR	−0.26677926	−0.7002255	0.3982835	2.9929899	5.2128794
XINSYS	0.00475193	0.0128323	−0.0992147	−0.3553974	−0.6146837
XPRDIA	0.00632701	0.0332371	0.0828177	0.0976728	0.0863094
XPAPUL	−0.00500955	−0.0486975	−0.0884098	−0.0864954	−0.0465313
XPVENT	0.00938577	0.0402316	0.0662682	0.0866649	0.1006702
XREPUL	0.00033615	0.0020090	0.0032057	0.0038279	0.0035192
	[,6]	[,7]	[,8]	[,9]	[,10]
XConstant	9.9356684	10.184312	10.1876410	10.1876416	10.1876416
XFRCAR	−0.1364497	−0.138156	−0.1381781	−0.1381781	−0.1381781
XINCAR	5.8045084	5.861785	5.8624289	5.8624290	5.8624290
XINSYS	−0.7062320	−0.716945	−0.7170839	−0.7170840	−0.7170840
XPRDIA	0.0752166	0.073687	0.0736681	0.0736681	0.0736681
XPAPUL	−0.0206140	−0.016810	−0.0167565	−0.0167565	−0.0167565
XPVENT	0.1061793	0.106769	0.1067760	0.1067760	0.1067760
XREPUL	0.0032053	0.003154	0.0031541	0.0031541	0.0031541

After 7 iterations, convergence was reached. See

```
> summary(logistic)$coefficients[,1]
```

```
 (Intercept)        FRCAR        INCAR        INSYS        PRDIA
10.187641685 -0.138178119  5.862429035 -0.717084018  0.073668171
       PAPUL        PVENT        REPUL
-0.016756506  0.106776012  0.003154187
 -0.073668171  0.016756506 -0.106776012 -0.003154187
```

From maximm likelihood properties,

$$\sqrt{n}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \xrightarrow{\mathcal{L}} \mathcal{N}(\mathbf{0}, I(\boldsymbol{\beta})^{-1}).$$

From a numerical point of view, this asymptotic variance $I(\boldsymbol{\beta})^{-1}$ satisfies $I(\boldsymbol{\beta})^{-1} = -H(\boldsymbol{\beta})$,

```
> -solve(Hessian)
             Constant       FRCAR       INCAR       INSYS
Constant 141.50029499 -1.1120707443 42.29308470 -5.792019e+00
FRCAR     -1.11207074  0.0130218132 -0.68328148  6.113030e-02
INCAR     42.29308470 -0.6832814845 45.54684575 -3.245723e+00
INSYS     -5.79201872  0.0611303045 -3.24572286  3.152285e-01
```

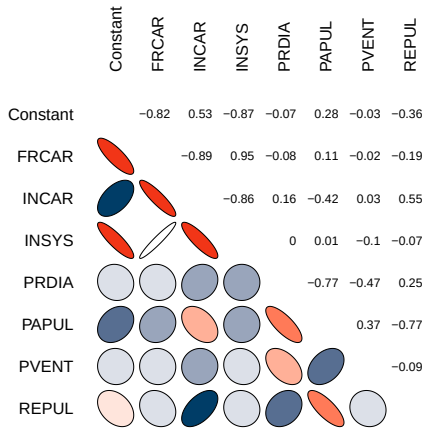
```
PRDIA    -0.25859731  -0.0026468213   0.30880800   5.050851e-05
PAPUL     1.15456409   0.0044720856  -0.97012941   2.087264e-03
PVENT    -0.03304634  -0.0002828814   0.02340121  -6.141984e-03
REPUL    -0.02103416  -0.0001058035   0.01811208  -1.791229e-04
                PRDIA         PAPUL         PVENT         REPUL
Constant -2.585973e-01  1.154564088 -3.304634e-02 -2.103416e-02
FRCAR    -2.646821e-03  0.004472086 -2.828814e-04 -1.058035e-04
INCAR     3.088080e-01 -0.970129411  2.340121e-02  1.811208e-02
INSYS     5.050851e-05  0.002087264 -6.141984e-03 -1.791229e-04
PRDIA     8.505231e-02 -0.076531702 -1.516899e-02  3.594489e-04
PAPUL    -7.653170e-02  0.116926256  1.390261e-02 -1.290423e-03
PVENT    -1.516899e-02  0.013902608  1.222148e-02 -4.677069e-05
REPUL     3.594489e-04 -0.001290423 -4.677069e-05  2.392144e-05
```

```
> sqrt(-diag(solve(Hessian)))
    Constant         FRCAR         INCAR         INSYS         PRDIA
11.895389653  0.114113159  6.748840326  0.561452093  0.291637287
       PAPUL         PVENT         REPUL
 0.341944815  0.110550787  0.004890955
```

```
> summary(logistic)$coefficients[,2]
  (Intercept)         FRCAR         INCAR         INSYS         PRDIA
11.895227082   0.114112163   6.748785486   0.561444684   0.291636009
         PAPUL         PVENT         REPUL
 0.341942251   0.110550096   0.004890917
```

One can also visualise the correlation matrix of $\widehat{\beta}$

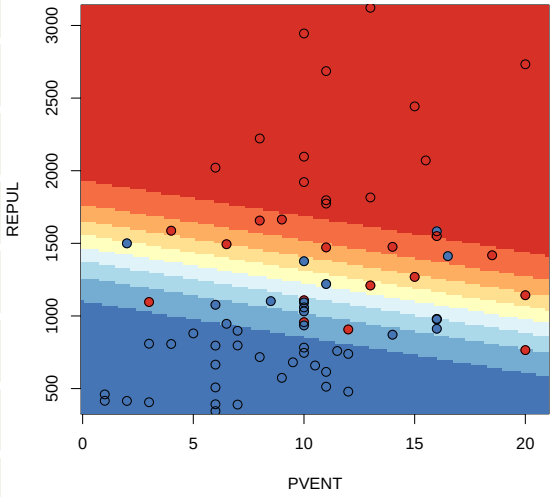
```
> sigma=sqrt(diag(-solve(Hessian)))
> C <- -solve(Hessian)/(sigma %*% t(sigma))
```



visualising a logistic regression

Consider a logistic regression with 2 covariates

```
> mlogistic2 <- vglm(PRONO~PVENT+REPUL , data=
    myocarde, family=multinomial)
> pred_logistic = function(p,r){
  return(predict(mlogistic2,newdata=data.frame(
    PVENT=p,REPUL=r),type="response")[,1])}
> vpvent=seq(min(myocarde$PVENT)-1,max(myocarde$
    PVENT)+1,length=151)
> vrepul=seq(min(myocarde$REPUL)-10,max(myocarde$
    REPUL)+10,length=151)
> image(vpvent,vrepul,outer(vpvent,vrepul,pred_
    logistic),col=CL2palette,xlab="PVENT",ylab="
    REPUL")
```



The logistic regression

```
> probabilities <- predict(logistic, myocarde, type="response")
> predictions <- levels(myocarde$PRONO)[(probabilities<.5)+1]
```

or for a more general classification

```
> probabilities <- predict(mlogistic, myocarde, type="response")
> probabilities[1:3,]
          [,1]      [,2]
[1,] 0.3986106 0.6013894
[2,] 0.8306231 0.1693769
[3,] 0.6710440 0.3289560
> pred <- levels(myocarde$PRONO)[apply(probabilities,1,which.max)]
```

```
> table(pred, myocarde$PRONO)

pred          DECES SURVIE
     DECES       25       3
     SURVIE       4      39
```

From $-0,1''$ to $-A,B,C''$

The Bernoulli case can (easily) be extended to the multinomial distribution. In the $-0,1''$ case

$$\log\frac{\mathbb{P}(Y=1)}{1-\mathbb{P}(Y=1)}=\log\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=0)}=\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}$$

so that

$$\mathbb{P}(Y=1)=\frac{e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}}}{1+e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}}}=\frac{p_1}{p_0+p_1}\text{ and }\mathbb{P}(Y=0)=\frac{1}{1+e^{\boldsymbol{X}^{\mathsf{T}}}}=\frac{p_0}{p_0+p_1}$$

In the case where the response variable takes values $-A,B,C''$,

$$\log\frac{\mathbb{P}(Y=A)}{\mathbb{P}(X=C)}=\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_A\text{ and }\log\frac{\mathbb{P}(Y=B)}{\mathbb{P}(X=C)}=\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_B$$

From "0, 1" to "A, B, C"

so that

$$\mathbb{P}(X=A)=\frac{p_A}{p_A+p_B+p_C} \text{ i.e. } \mathbb{P}(X=A)=\frac{e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_A}}{e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_B}+e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_B}+1}$$

$$\mathbb{P}(X=B)=\frac{p_B}{p_A+p_B+p_C} \text{ i.e. } \mathbb{P}(X=B)=\frac{e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_B}}{e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_A}+e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_B}+1}$$

and

$$\mathbb{P}(X=C)=\frac{p_C}{p_A+p_B+p_C} \text{ i.e. } \mathbb{P}(X=C)=\frac{1}{e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_A}+e^{\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}_B}+1}$$

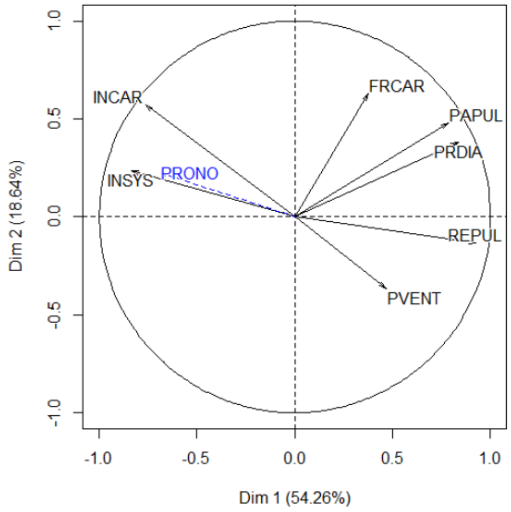
Visualisation of a Classification

Consider here some logistic regression on k variables. Can we visualise it ?

On our myocarde dataset, consider some principal component analysis, including the pronostic variable,

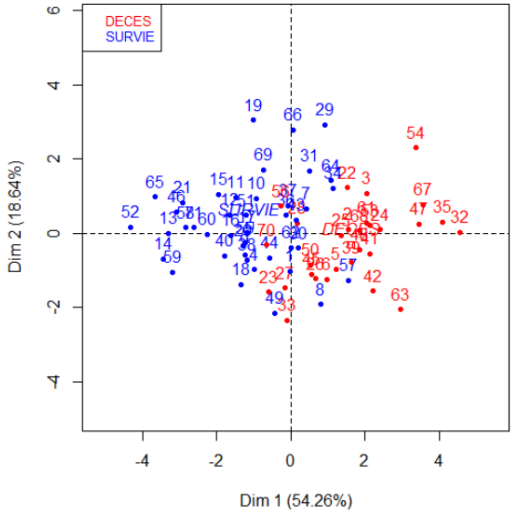
either as a continuous variable,

```
> library(FactoMineR)
> MYOCARDE2 <- MYOCARDE
> MYOCARDE2$PRONO <- (MYOCARDE2$PRONO=="SURVIE")*1
> acp <- PCA(MYOCARDE2, quanti.sup=8,graph=TRUE)
```



or as a qualitative variable,

```
> acp <- PCA(MYOCARDE, quali.sup=8,graph=TRUE)
> plot(acp, habillage = 8,col.hab=c("red","blue"))
```



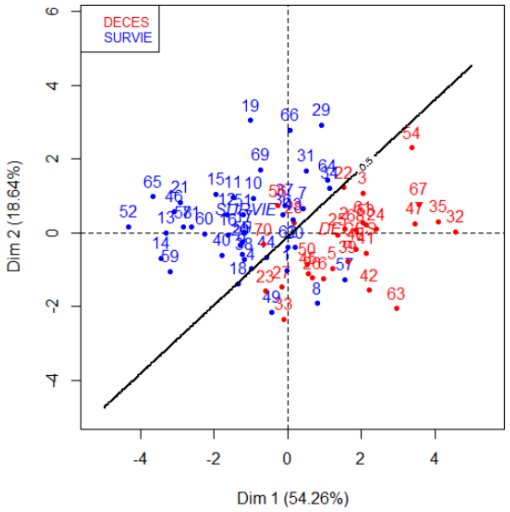
To get from the first two component hyperplane to $\mathbb{R}^k$, use

```
> X <- MYOCARDE[,1:7]
> acp <- PCA(X,ncp=ncol(X))
> M <- acp$var$coord
> m <- apply(X,2,mean)
> s <- apply(X,2,sd)
> pred <- function(d1,d2,Mat,reg){
+   z <- Mat %*% c(d1,d2,rep(0,ncol(X)-2))
+   newd <- data.frame(t(z*s+m))
+   names(newd) <- names(X)
```

```
10 +   predict(reg,newdata=newd,type="response") }
```

Hence
or as a qualitative variable,

```
1 > reg_tot <- glm(PRONO~.,data=MYOCARDE,family=
      binomial)
2 > Minv <- solve(M)
3 > p <- Vectorize(function(d1,d2) pred(d1,d2,Minv,
      reg_tot))
4 > zgrid <- outer(xgrid,ygrid,p)
5 > acp2 <- PCA(MYOCARDE,quali.sup=8,graph=TRUE)
6 > plot(acp2, habillage = 8,col.hab=c("red","blue")
      )
7 > contour(xgrid,ygrid,zgrid,add=TRUE,levels=.5)
```



Linear Discriminant Analysis

LDA is a classification method that finds a linear combination of data attributes that best separate the data into classes.

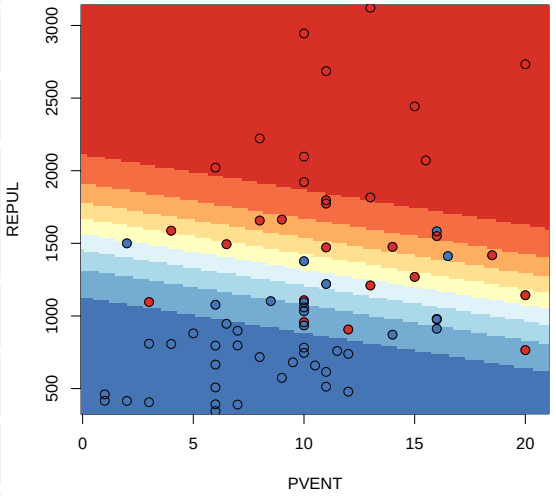
```
> library(MASS)
> fit_lda <- lda(PRONO ~. , data=myocarde)
> fit_lda
Call:
lda(PRONO ~ ., data = myocarde)

Prior probabilities of groups:
    DECES     SURVIE
0.4084507 0.5915493

Group means:
              FRCAR     INCAR     INSYS     PRDIA     PAPUL     PVENT
    REPUL
DECES  91.55172 1.397931 15.53103 21.44828 28.43103 11.844828
    1738.6897
```

```
SURVIE 87.69048 2.318333 27.20238 15.97619 22.20238  8.642857
    817.2143

Coefficients of linear discriminants:
               LD1
FRCAR -0.012743116
INCAR  1.074534545
INSYS -0.019139867
PRDIA -0.025483955
PAPUL  0.020177505
PVENT -0.037804074
REPUL -0.001353977
> predictions <- predict(fit_lda, myocarde)$class
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
     DECES     25      2
     SURVIE     4     40
```

visualising a LDA

Consider a logistic regression with 2 covariates

```
fit_lda2 <- lda(PRONO~PVENT+REPUL , data=myocarde)
pred_lda = function(p,r){
  return(predict(fit_lda2,newdata=
                   data.frame(PVENT=p,REPUL=r),
                 type="response")$posterior[,"
    DECES"])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_lda),
     col=CL2palette,
       xlab="PVENT",ylab="REPUL")
```



Partial Least Square Discriminant Analysis

Partial Least Squares Discriminate Analysis is the application of LDA on a dimension-reducing projection of the input data (partial least squares).

```
> library(caret)
> X=as.matrix(myocarde[,1:7])
> Y=myocarde$PRONO
> fit_plsda <- plsda(X, Y, probMethod="Bayes")
> predictions <- predict(fit_plsda, myocarde[,1:7])
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES    24      3
      SURVIE    5     39
```

Assume that

$$\boldsymbol{X}|Y=0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}) \text{ and } \boldsymbol{X}|Y=1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

then

$$\log\frac{\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})}{\mathbb{P}(Y=0|\boldsymbol{X}=\boldsymbol{x})}=[\boldsymbol{x}]^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}[\boldsymbol{\mu}_y]-\frac{1}{2}[\boldsymbol{\mu}_1-\boldsymbol{\mu}_0]^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}[\boldsymbol{\mu}_1-\boldsymbol{\mu}_0]+\log\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=0)}$$

which is linear in $\boldsymbol{x}$

$$\log\frac{\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})}{\mathbb{P}(Y=0|\boldsymbol{X}=\boldsymbol{x})}=\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta}$$

When each groups have Gaussian distributions with identical variance matrix, then LDA and the logistic regression lead to the same classification rule.

⚠ there is a slight difference in the way parameters are estimated.

Mixture Discriminant Analysis

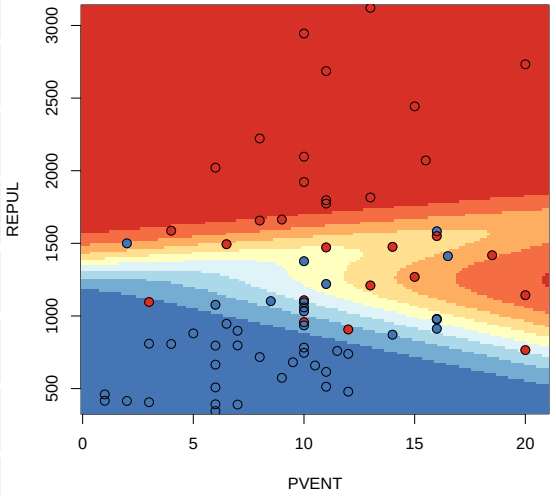
```
> library(mda)
> fit_mda <- mda(PRONO ~. , data=myocarde)
> fit_mda
Call:
mda(formula = PRONO ~ ., data = myocarde)

Dimension: 5

Percent Between-Group Variance Explained:
    v1     v2     v3     v4     v5
 82.43  97.16  99.45  99.88 100.00

Degrees of Freedom (per dimension): 8

Training Misclassification Error: 0.14085 ( N = 71 )

Deviance: 46.203
```

```
> predictions <- predict(fit_mda,myocarde)
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES     24      5
      SURVIE     5     37
```

visualising a MDA

Consider a MDA with 2 covariates

```
fit_mda <- mda(PRONO~PVENT+REPUL , data=myocarde)
pred_mda = function(p,r){
  return(predict(fit_mda,newdata=
                   data.frame(PVENT=p,REPUL=r),
    type="posterior")[,"DECES"])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_mda),
    col=CL2palette,
      xlab="PVENT",ylab="REPUL")
```



Quadratic Discriminant Analysis

QDA seeks a quadratic relationship between attributes that maximises the distance between the classes.

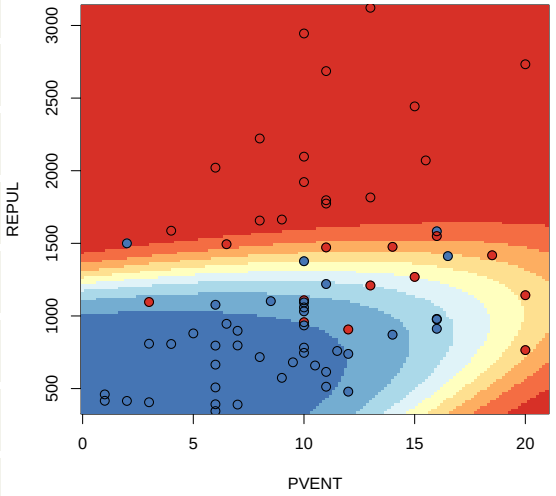
```
> library(MASS)
> fit_dqa <- qda(PRONO ~.  , data=myocarde)
> fit_dqa
Call:
qda(PRONO ~ ., data = myocarde)

Prior probabilities of groups:
    DECES     SURVIE
0.4084507 0.5915493

Group means:
            FRCAR     INCAR     INSYS     PRDIA     PAPUL     PVENT
    REPUL
DECES  91.55172 1.397931 15.53103 21.44828 28.43103 11.844828
    1738.6897
```

```
SURVIE 87.69048 2.318333 27.20238 15.97619 22.20238  8.642857
    817.2143
> predictions <- predict(fit_dqa, myocarde)$class
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
     DECES     24      5
     SURVIE     5     37
```

visualising a QDA

Consider a QDA with 2 covariates

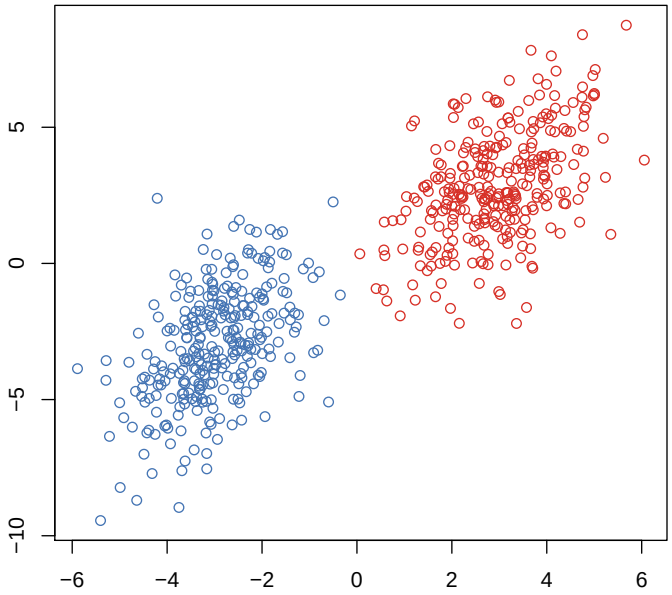
```
fit_qda <- qda(PRONO~PVENT+REPUL , data=myocarde)
pred_qda = function(p,r){
  return(predict(fit_qda,newdata=
                   data.frame(PVENT=p,REPUL=r))$
    posterior[,"DECES"])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_qda),
    col=CL2palette,
      xlab="PVENT",ylab="REPUL")
```

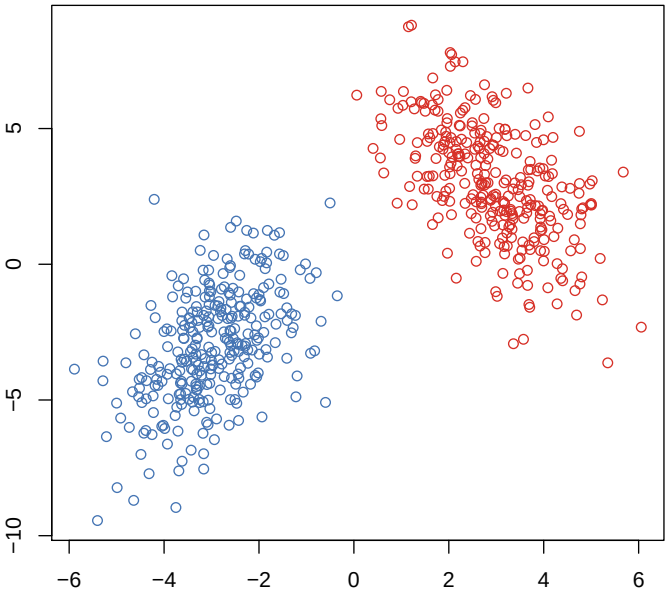


The difference between LDA and QDA

```
S<-matrix(c(1,.5*2,.5*2,4),2,2)

X1<-rmnorm(300,c(-1,-1)*3,S)
X2<-rmnorm(300,c(1,1)*3,S)
```

```
S1<-matrix(c(1,.5*2,.5*2,4),2,2)
S2<-matrix(c(1,-.5*2,-.5*2,4),2,2)
X1<-rmnorm(300,c(-1,-1)*3,S1)
X2<-rmnorm(300,c(1,1)*3,S2)
```

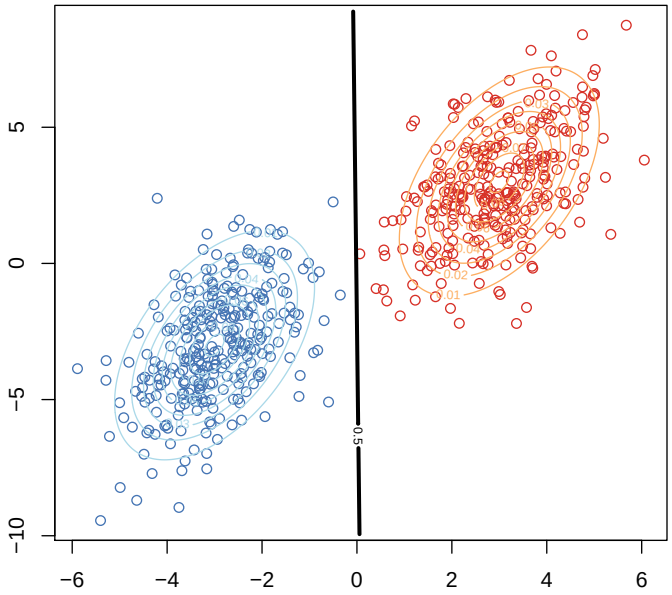


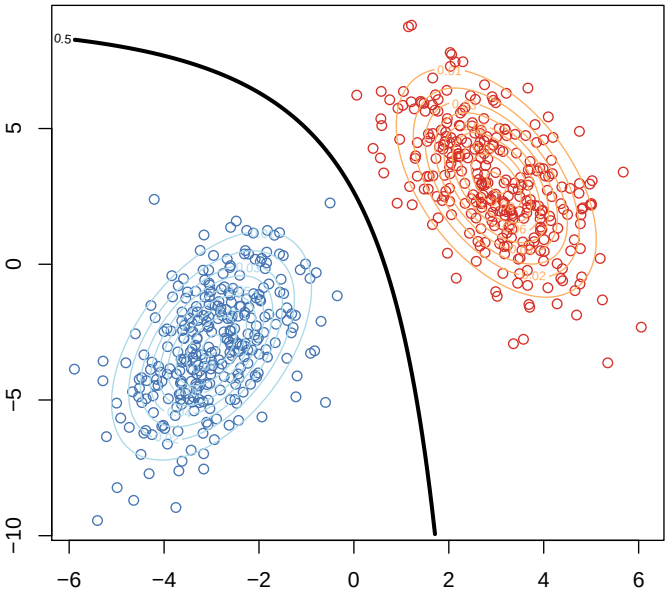


The difference between LDA and QDA

```
fitL<-lda(y ~ x1 + x2, data=df)
pL<-function(u,v) predict(fitL,
newdata=data.frame(x1=u,x2=v)
)$posterior[,"1"])
```

```
fitQ<-qda(y ~ x1 + x2, data=df)
pQ<-function(u,v) predict(fitQ,
newdata=data.frame(x1=u,x2=v)
)$posterior[,"1"])
```





k-Nearest Neighbor and Classification

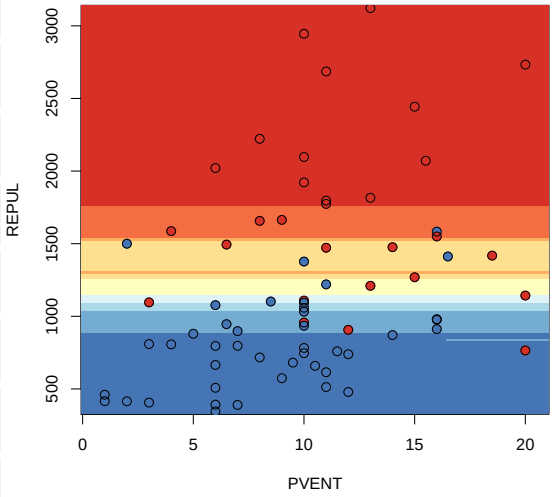
The k-Nearest Neighbor (kNN) method makes predictions by locating similar cases to a given data instance (using a similarity function) and returning the average or majority of the most similar data instances.

```
> library(caret)
> kNN <- knn3(PRONO~., data=myocarde, k=5)
> predictions <- predict(kNN,myocarde, type="class")
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES     24      5
      SURVIE     5     37
```

Visualising a k-Nearest Neighbor Classification

Consider a k-NN with 2 covariates

```
kNN2 <- knn3(PRONO~PVENT+REPUL, data=myocarde, k=10)
pred_kNN = function(p,r){
  return(predict(kNN2,newdata=
  data.frame(PVENT=p,REPUL=r), type="prob")[,1])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_kNN),
    col=CL2palette,
xlab="PVENT",ylab="REPUL")
```



Naive Bayes Classifier

Naive Bayes uses Bayes Theorem to model the conditional relationship of each attribute to the class variable.

```
> library(e1071)
> NB <- naiveBayes(PRONO~., data=myocarde)
> NB

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    DECES     SURVIE
0.4084507 0.5915493

Conditional probabilities:
```

```
            FRCAR
Y                 [,1]      [,2]
  DECES    91.55172  15.28441
  SURVIE   87.69048  14.58949

            INCAR
Y                 [,1]      [,2]
  DECES    1.397931  0.3808954
  SURVIE   2.318333  0.5743880
```

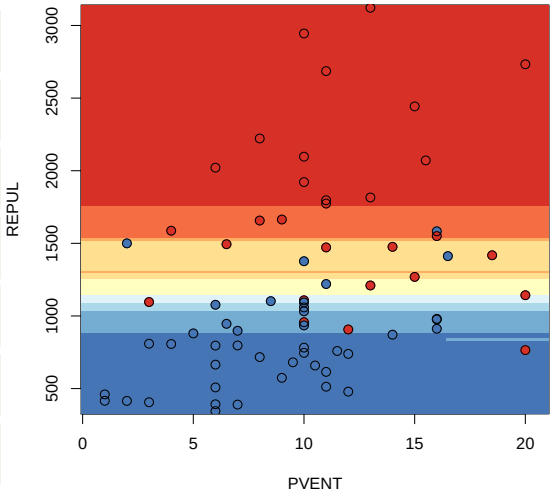
etc

```
> predictions <- predict(NB, myocarde)
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES     25      4
      SURVIE     4     38
```

Visualising a Naive Bayes Classification

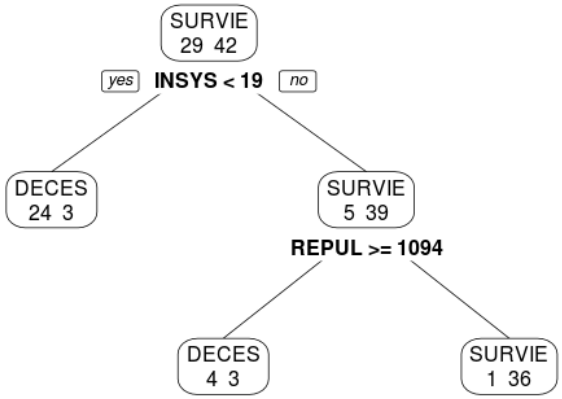
Consider a NB with 2 covariates

```
> NB2 <- naiveBayes(PRONO~PVENT+REPUL, data=
    myocarde)
> pred_NB = function(p,r){
+ return(predict(NB2,newdata=
+ data.frame(PVENT=p,REPUL=r), type="raw")[,1])}
> image(vpvent,vrepul,outer(vpvent,vrepul,pred_NB)
    ,col=CL2palette, xlab="PVENT",ylab="REPUL")
```



Visualising a Classification Tree

```
> MYOCARDE=read.table(
+ "http://freakonometrics.free.fr/saporta.csv",
+ head=TRUE,sep=";")
> library(rpart)
> cart<-rpart(PRONO~.,data=MYOCARDE)
> library(rpart.plot)
> library(rattle)
> prp(cart,type=2,extra=1)
```



Visualising a Classification Tree

If X is splitted in 2 classes, denoted A and B, use either Gini index

$$\text{gini}(Y|X) = -\sum_{x\in -A,B''} \frac{n_x}{n} \sum_{y\in -0,1''} \frac{n_{x,y}}{n_x}\left(1-\frac{n_{x,y}}{n_x}\right)$$

```
> gini=function(y,classe){
+ T=table(y,classe)
+ nx=apply(T,2,sum)
+ pxy=T/matrix(rep(nx,each=2),nrow=2)
+ omega=matrix(rep(nx,each=2),nrow=2)/sum(T)
+ return( -sum(omega*pxy*(1-pxy)) )}
```

Hence,

```
> CLASSE=MYOCARDE[,1]<=2.5
> gini(y=MYOCARDE$PRONO, classe=CLASSE)
[1] -0.4832375
```

Visualising a Classification Tree

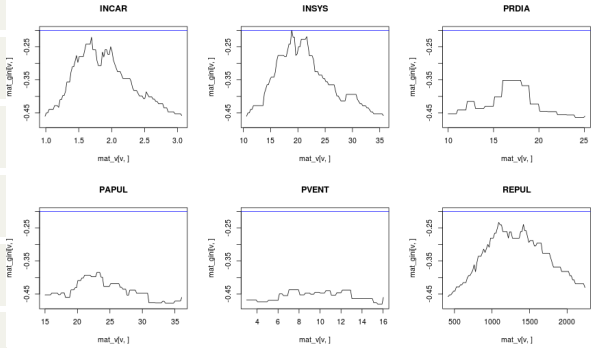
or the entropy index,

$$\text{entropy}(Y|X) = -\sum_{x\in -A,B,C''} \frac{n_x}{n} \sum_{y\in -0,1''} \frac{n_{x,y}}{n_x} \log\left(\frac{n_{x,y}}{n_x}\right)$$

```
> entropie=function(y,classe){
+    T=table(y,classe)
+    nx=apply(T,2,sum)
+    n=sum(T)
+    pxy=T/matrix(rep(nx,each=2),nrow=2)
+    omega=matrix(rep(nx,each=2),nrow=2)/n
+    return( sum(omega*pxy*log(pxy)) )}
```

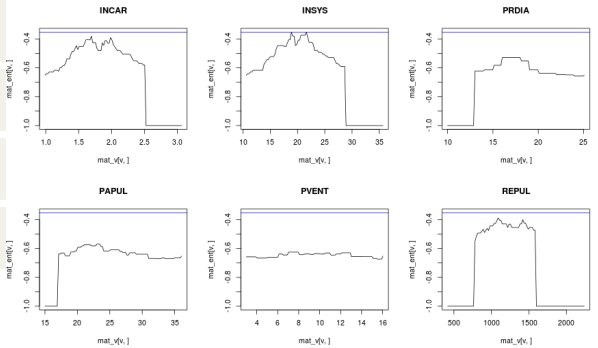
Visualising a Classification Tree

```
> mat_gini=mat_v=matrix(NA,7,101)
> for(v in 1:7){
+     variable=MYOCARDE[,v]
+     v_seuil=seq(quantile(MYOCARDE[,v],
+ 6/length(MYOCARDE[,v])),
+ quantile(MYOCARDE[,v],1-6/length(
+ MYOCARDE[,v])),length=101)
+     mat_v[v,]=v_seuil
+     for(i in 1:101){
+ CLASSE=variable<=v_seuil[i]
+ mat_gini[v,i]=
+     gini(y=MYOCARDE$PRONO, classe=CLASSE)}}
```



Visualising a Classification Tree

```
> par(mfrow=c(2,3))
> for(v in 2:7){
+     plot(mat_v[v,],mat_gini[v,],type="l",
+     ylim=range(mat_gini),
+     main=names(MYOCARDE)[v])
+     abline(h=max(mat_gini),col="blue")
+ }
```

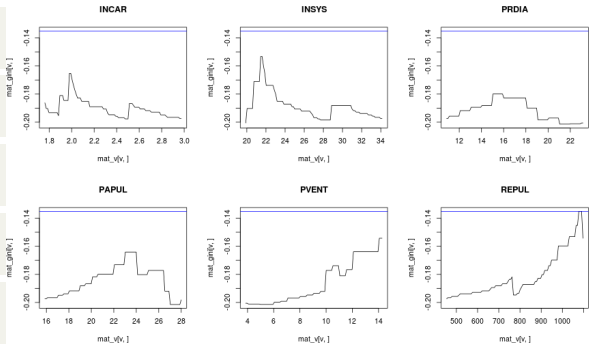


or we can use the entropy index.

```
> idx=which(MYOCARDE$INSYS>=19)
```

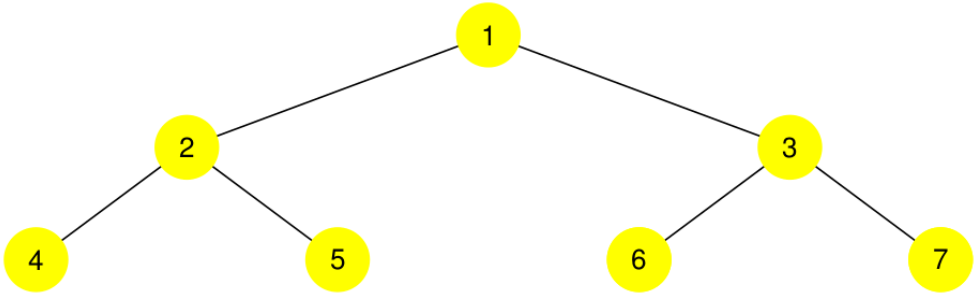
Visualising a Classification Tree

```
> mat_gini=mat_v=matrix(NA,7,101)
> for(v in 1:7){
+   variable=MYOCARDE[idx,v]
+   v_seuil=seq(quantile(MYOCARDE[idx,v],
+ 6/length(MYOCARDE[idx,v])),
+ quantile(MYOCARDE[idx,v],1-6/length(
+ MYOCARDE[idx,v])), length=101)
+  mat_v[v,]=v_seuil
+  for(i in 1:101){
+   CLASSE=variable<=v_seuil[i]
+   mat_gini[v,i]=
+   gini(y=MYOCARDE$PRONO[idx],
+     classe=CLASSE)}}
```



Classification and Regression Trees (CART)

```
> library(rpart)
> cart <- rpart(PRONO~. , data=myocarde)
```



Classification and Regression Trees (CART)

```
> summary(cart)
Call:
rpart(formula = PRONO ~ ., data = myocarde)
  n= 71

          CP nsplit  rel error     xerror       xstd
1 0.72413793      0 1.0000000 1.0000000 0.1428224
2 0.03448276      1 0.2758621 0.4827586 0.1156044
3 0.01000000      2 0.2413793 0.5172414 0.1186076

Variable importance
INSYS REPUL INCAR PAPUL PRDIA FRCAR PVENT
   29    27    23     8     7     5     1
```

Classification and Regression Trees (CART)

```
Node number 1: 71 observations,    complexity param=0.7241379
  predicted class=SURVIE  expected loss=0.4084507  P(node) =1
    class counts:    29    42
   probabilities: 0.408 0.592
  left son=2 (27 obs) right son=3 (44 obs)
  Primary splits:
      INSYS < 18.85   to the left,  improve=20.112890, (0 missing)
      REPUL < 1094.5  to the right, improve=19.021400, (0 missing)
      INCAR < 1.69    to the left,  improve=18.615510, (0 missing)
      PRDIA < 17      to the right, improve= 9.361141, (0 missing)
      PAPUL < 23.25   to the right, improve= 7.022101, (0 missing)
  Surrogate splits:
      REPUL < 1474    to the right, agree=0.915, adj=0.778, (0 split)
      INCAR < 1.665   to the left,  agree=0.901, adj=0.741, (0 split)
      PAPUL < 28.5    to the right, agree=0.732, adj=0.296, (0 split)
      PRDIA < 18.5    to the right, agree=0.718, adj=0.259, (0 split)
      FRCAR < 99.5    to the right, agree=0.690, adj=0.185, (0 split)
```

Classification and Regression Trees (CART)

```
Node number 2: 27 observations
  predicted class=DECES   expected loss=0.1111111  P(node) =0.3802817
    class counts:    24     3
   probabilities: 0.889 0.111
```

Classification and Regression Trees (CART)

```
Node number 3: 44 observations,    complexity param=0.03448276
  predicted class=SURVIE  expected loss=0.1136364  P(node) =0.6197183
    class counts:     5    39
   probabilities: 0.114 0.886
  left son=6 (7 obs) right son=7 (37 obs)
  Primary splits:
      REPUL < 1094.5 to the right, improve=3.489119, (0 missing)
      INSYS < 21.55  to the left,  improve=2.122460, (0 missing)
      PVENT < 13     to the right, improve=1.651281, (0 missing)
      PAPUL < 23.5   to the right, improve=1.641414, (0 missing)
      INCAR < 1.985  to the left,  improve=1.592803, (0 missing)
  Surrogate splits:
      INCAR < 1.685  to the left,  agree=0.886, adj=0.286, (0 split)
      PVENT < 17.25  to the right, agree=0.864, adj=0.143, (0 split)
```

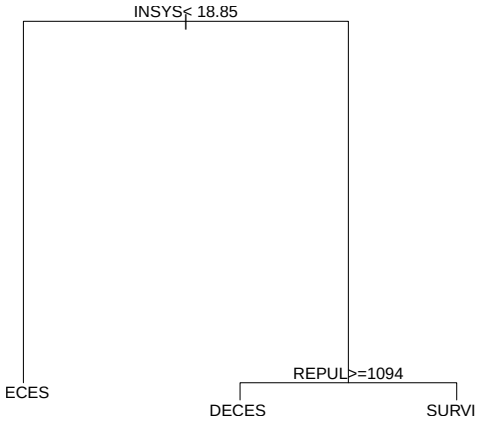
Classification and Regression Trees (CART)

```
1 Node number 6: 7 observations
2   predicted class=DECES   expected loss=0.4285714  P(node)
      =0.09859155
3     class counts:     4     3
4    probabilities: 0.571 0.429
5
6 Node number 7: 37 observations
7   predicted class=SURVIE  expected loss=0.02702703  P(node)
      =0.5211268
8     class counts:     1    36
9    probabilities: 0.027 0.973
```

Vizualize a Trees (CART)

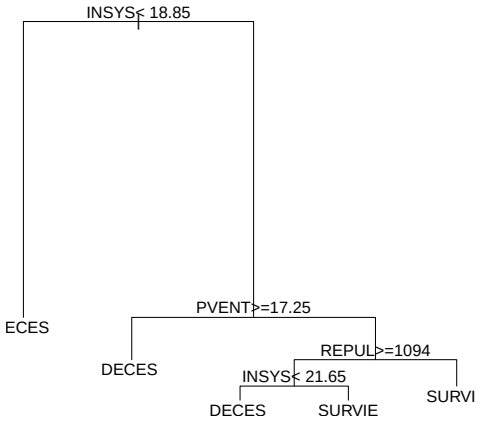
A basic viz' of the tree

```
> cart <- rpart(PRONO~., data=myocarde)
> plot(cart)
> text(cart)
```



Each leaf contains, at least, 20 observations. But we can ask for less

```
> cart <- rpart(PRONO~., data=myocarde,minsplit =
    5)
> plot(cart)
> text(cart)
```



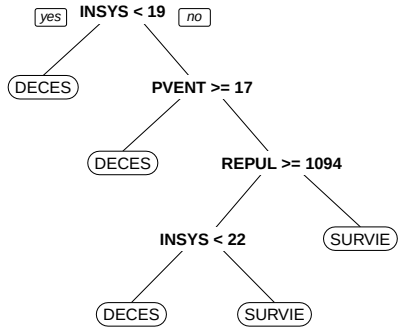
Vizualize a Trees (CART)

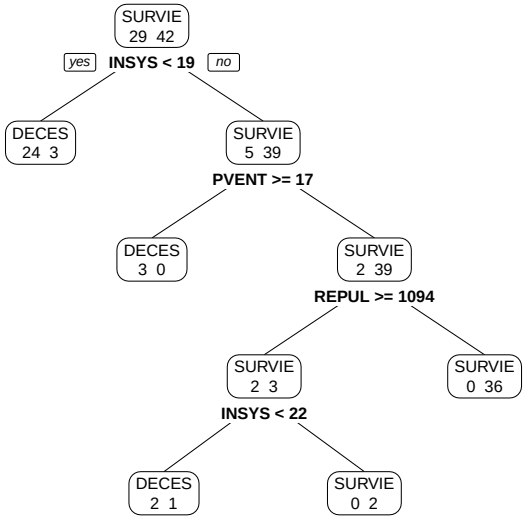
A basic viz' of the tree

```
> library(rpart.plot)
> prp(cart)
```

or

```
> prp(cart,type=2,extra=1)
```

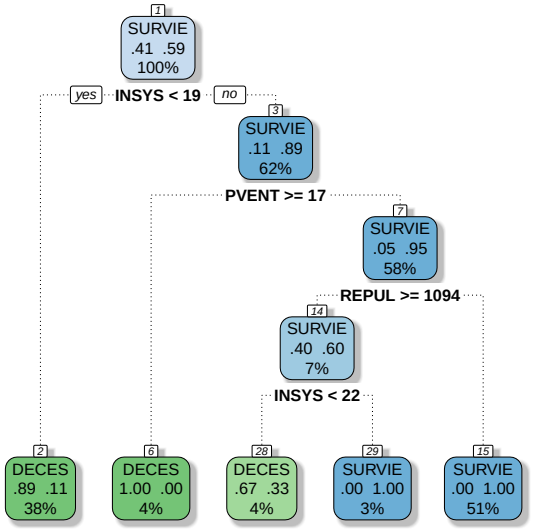




Vizualize a Trees (CART)

A basic viz' of the tree

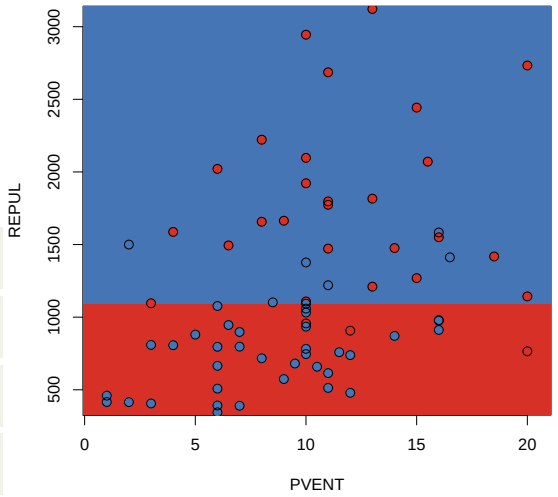
```
> library(rattle)
> fancyRpartPlot(cart,sub="")
```



Vizualize a Trees (CART)

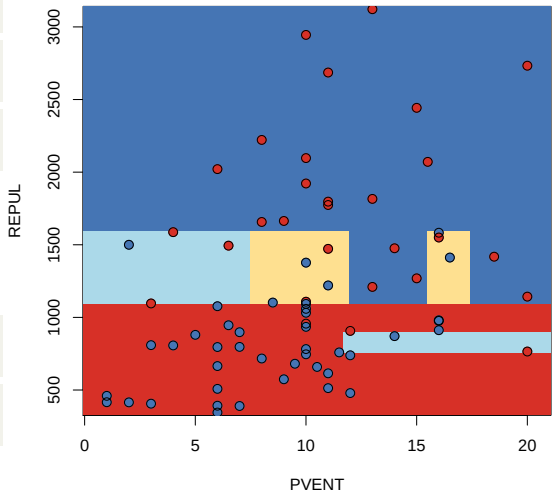
A basic viz' of the tree

```
cart2 <- rpart(PRONO~PVENT+REPUL , data=myocarde,
    minsplit = 20)
pred_cart = function(p,r){return(predict(cart2,
    newdata=data.frame(PVENT=p,REPUL=r))[,"SURVIE"
    ])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_cart)
    ,col=CL2palette,
      xlab="PVENT",ylab="REPUL")
```



or

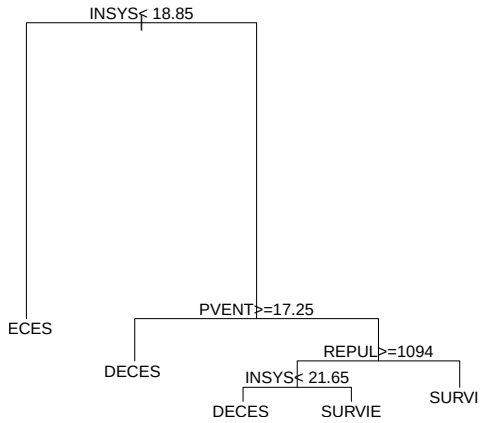
```
cart2 <- rpart(PRONO~PVENT+REPUL , data=myocarde,
    minsplit = 5)
```



Trees (CART), Gini vs. Entropy

Tree based in Gini impurity index

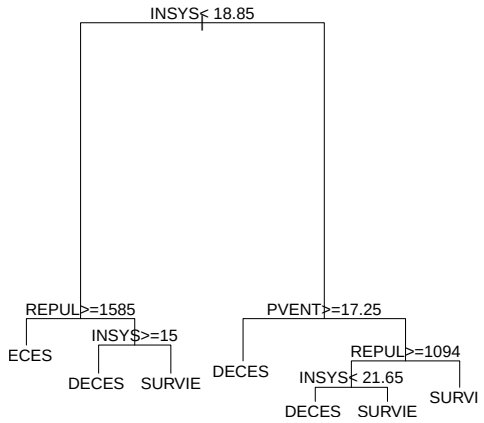
```
> cart_gini <- rpart(PRONO~.  , data=myocarde,
    minsplit = 5,parms=list(split="gini"))
> summary(cart_gini)

           CP nsplit  rel error    xerror       xstd
1 0.72413793      0 1.0000000 1.0000000 0.1428224
2 0.10344828      1 0.2758621 0.5862069 0.1239921
3 0.01724138      2 0.1724138 0.4827586 0.1156044
4 0.01000000      4 0.1379310 0.4482759 0.1123721
> plot(cart_gini)
> text(cart_gini)
```



Trees (CART), Gini vs. Entropy

Tree based in the entropy impurity index

```
> cart_entropy <- rpart(PRONO~. , data=myocarde,
    minsplit = 5,parms=list(split="information"))
> summary(cart_entropy)

           CP nsplit  rel error     xerror       xstd
1 0.72413793      0 1.00000000 1.0000000 0.1428224
2 0.10344828      1 0.27586207 0.5862069 0.1239921
3 0.03448276      2 0.17241379 0.4827586 0.1156044
4 0.01724138      4 0.10344828 0.4482759 0.1123721
5 0.01000000      6 0.06896552 0.4482759 0.1123721
> plot(cart_entropy)
> text(cart_entropy)
```



Trees (CART), Gini vs. Entropy

Minority class $\min\{p, 1-p\}$ is the error rate: it measures the proportion of misclassified examples if the leaf was labelled with the majority class

Gini index $2p(1-p)$ – this is the expected error if we label examples in the leaf randomly: positive with probability p and negative with probability 1-p.

entropy $-p\log p-(1-p)\log(1-p)$ – this is the expected information

Observe that Gini index is related to the variance of a Bernoulli distribution. With two leaves

$$\frac{n_1}{n}\underbrace{p_1(1-p_1)}_{\text{var}_1}+\frac{n_2}{n}\underbrace{p_2(1-p_2)}_{\text{var}_2}$$

is a weigthed average variance.

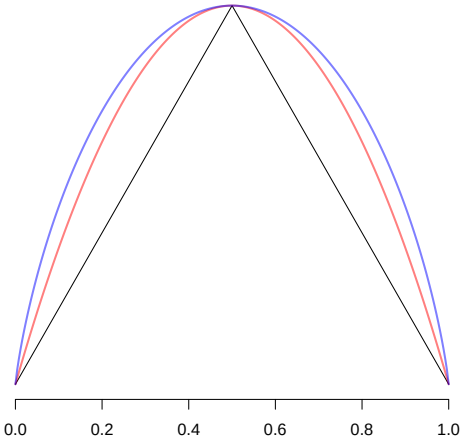
⚠ Regression tree will be obtained by replacing the impurity measure by the variance.

Trees (CART), Gini vs. Entropy

⚠ Dietterich, Kearns and Mansour (1996) suggested to use $\sqrt{\text{Gini}}$ as a measure of impurity. Entropy and Gini index are sensitive to fluctuations in the class distribution, $\sqrt{\text{Gini}}$ isn't. See Drummond & Holte (cs.alberta.ca, 2000) for a discussion on (in)sensitivity of decision tree splitting criteria.

But standard criteria yield (usually) similar trees (rapid-i.com or quora.com)

- misclassification rate $1-\max -p, 1-p''$
- entropy $-[p\log p+(1-p)\log(1-p)]$
- Gini index $1-[p^2+(1-p)^2]=2p(1-p)$

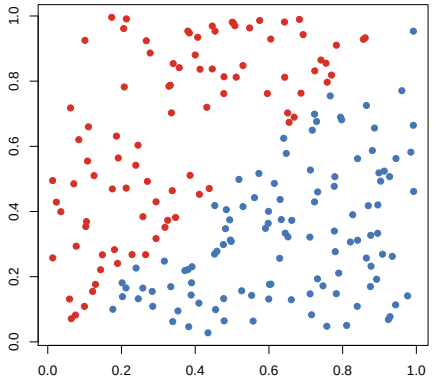


Prunning a Tree

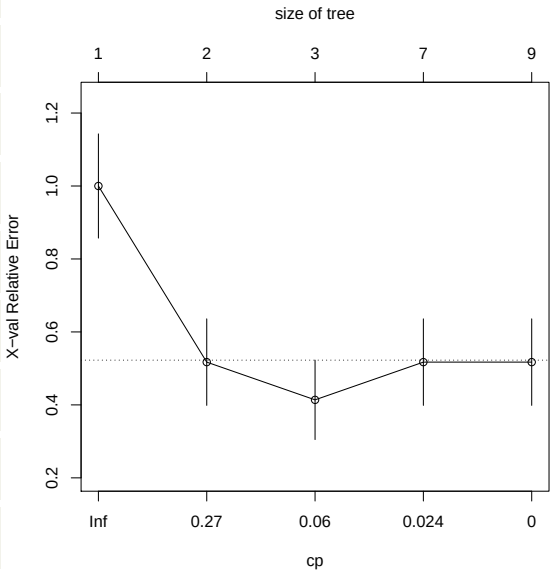
Drop in impurity at node N is defined as

$$\Delta i(N) = i(N) - \mathbb{P}(\text{left})i(N_{\text{left}}) - \mathbb{P}(\text{right})i(N_{\text{right}})$$

If we continue to grow the tree fully until each leaf node corresponds to the lowest impurity, we will overfit. If splitting is stopped too early, error on training data is not sufficiently low and performance will suffer. Thus, stop splitting when the best candidate split at a node reduces the impurity by less than the preset amount, or when a node has a small number of observations



```
> cart <- rpart(PRONO~. , data=myocarde,minsplit =
    2,parms=list(split="gini"),cp=0)
> printcp(cart)

Variables actually used in tree construction:
[1] FRCAR INCAR INSYS PVENT REPUL

Root node error: 29/71 = 0.40845

n= 71

        CP nsplit rel error  xerror     xstd
1 0.724138      0  1.000000 1.00000 0.14282
2 0.103448      1  0.275862 0.51724 0.11861
3 0.034483      2  0.172414 0.41379 0.10889
4 0.017241      6  0.034483 0.51724 0.11861
5 0.000000      8  0.000000 0.51724 0.11861
> plotcp(cart)
```



Classification Pronostic

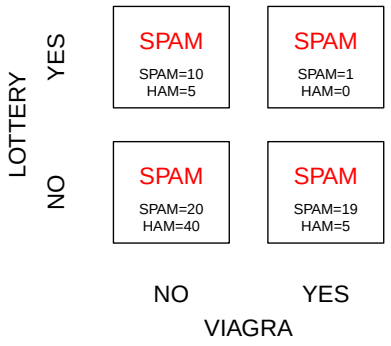
For pronostics, consider the confusion matrix

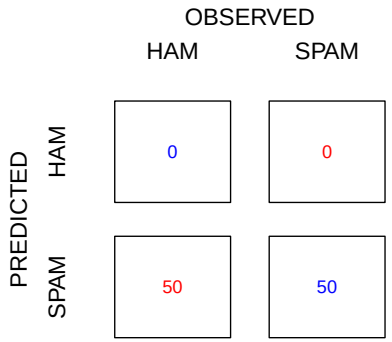
```
> cart <- rpart(PRONO~., data=myocarde)
> predictions <- predict(cart, myocarde, type="class")
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES     28      6
      SURVIE     1     36
```

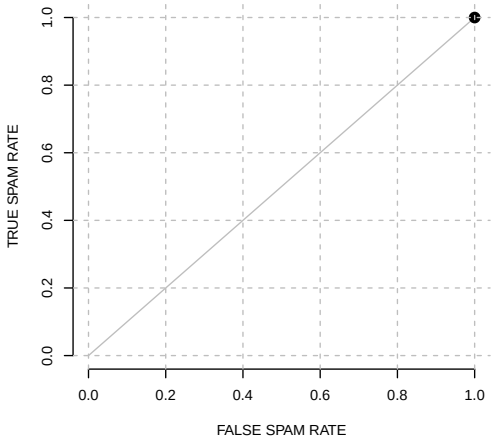
Classification with Categorical Variables

Consider a spam classifier, based on two keywords, viagra and lottery.

```
> load("spam.RData")
> head(db,4)
      Y viagra lottery
27 spam      0       1
37  ham      0       1
57 spam      0       0
89  ham      0       0
```



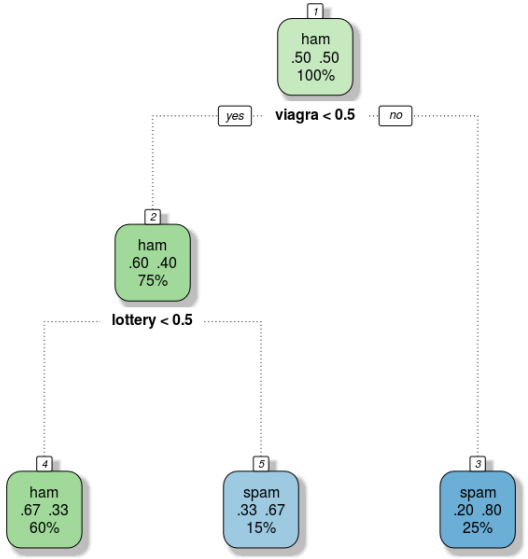




Classification with Categorical Variables

Consider e.g. a tree classifier

```
library(rpart)
library(rattle)
ctrl <- rpart.control(maxdepth=3)
model <- rpart(Y~viagra+lottery,
data=db, control=ctrl)
fancyRpartPlot(model)
```



Classification with C4.5 algorithm

The C4.5 algorithm is an extension of the ID3 algorithm and constructs a decision tree to maximize information gain (difference in entropy).

```
> library(RWeka)
> C45 <- J48(PRONO~.  , data=myocarde)
> summary(C45)

=== Summary ===

Correctly Classified Instances          66               92.9577 %
Incorrectly Classified Instances         5                7.0423 %
Kappa statistic                          0.855
Mean absolute error                      0.1287
Root mean squared error                  0.2537
Relative absolute error                 26.6091 %
Root relative squared error             51.6078 %
Coverage of cases (0.95 level)          97.1831 %
```

```
Mean rel. region size (0.95 level)          69.0141 %
Total Number of Instances                   71

=== Confusion Matrix ===

   a   b   <-- classified as
  27   2 |   a = DECES
   3  39 |   b = SURVIE
```

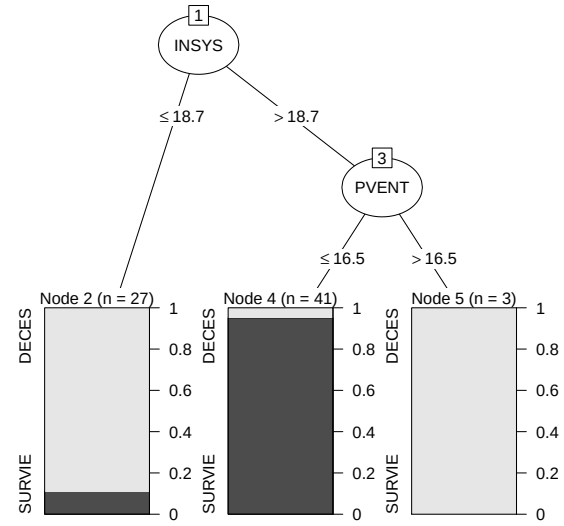
```
> predictions <- predict(C45, myocarde)
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES     27      3
      SURVIE     2     39
```

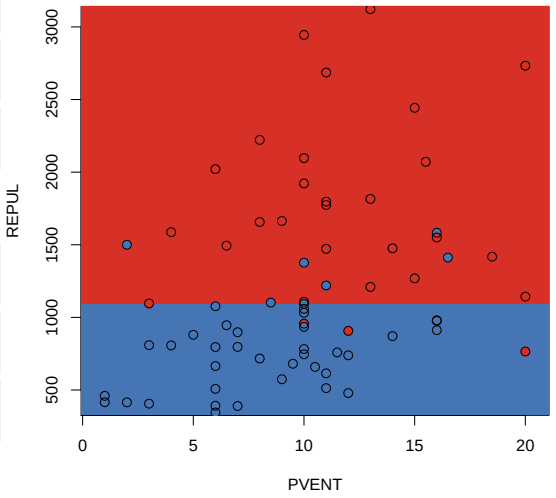
Classification with C4.5 algorithm

To visualise this tree, use

```
C452 <- J48(PRONO~PVENT+REPUL, data=myocarde)
pred_C45 = function(p,r){
  return(predict(C452,newdata=
                   data.frame(PVENT=p,REPUL=r),
    type="probability")[,1])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_C45),
    col=CL2palette,
      xlab="PVENT",ylab="REPUL")
```

```
> library(partykit)
> plot(C45)
> plotcp(cart)
```





From C4.5 algorithm to PART

PART is a rule system that creates pruned C4.5 decision trees for the data set and extracts rules and those instances that are covered by the rules are removed from the training data. The process is repeated until all instances are covered by extracted rules.

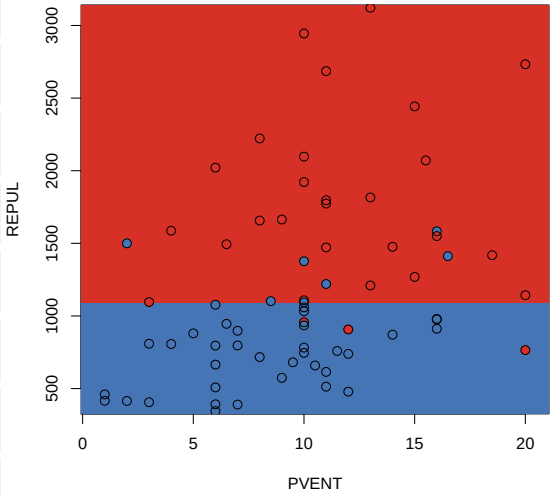
```
> library(RWeka)
> fit_PART <- PART(PRONO~.  , data=myocarde)
> summary(fit_PART)

=== Summary ===

Correctly Classified Instances          69               97.1831 %
Incorrectly Classified Instances         2                2.8169 %
Kappa statistic                          0.9423
Mean absolute error                      0.0488
Root mean squared error                  0.1562
Relative absolute error                 10.0944 %
```

```
Root relative squared error                 31.7864 %
Coverage of cases (0.95 level)             100      %
Mean rel. region size (0.95 level)          60.5634 %
Total Number of Instances                   71

=== Confusion Matrix ===

  a  b   <-- classified as
 29  0 |  a = DECES
  2 40 |  b = SURVIE
```

From C4.5 algorithm to PART

The code to visualise this tree is

```
PART2 <- PART(PRONO~PVENT+REPUL, data=myocarde)
pred_PART = function(p,r){
  return(predict(PART2,newdata=
                   data.frame(PVENT=p,REPUL=r),
    type="probability")[,1])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_PART)
    ,col=CL2palette,
      xlab="PVENT",ylab="REPUL")
```



Boostrap and Bagging

Bootstrapped Aggregation (Bagging) is an ensemble method that creates multiple models of the same type from different sub-samples of the same dataset. The predictions from each separate model are combined together to provide a superior result.

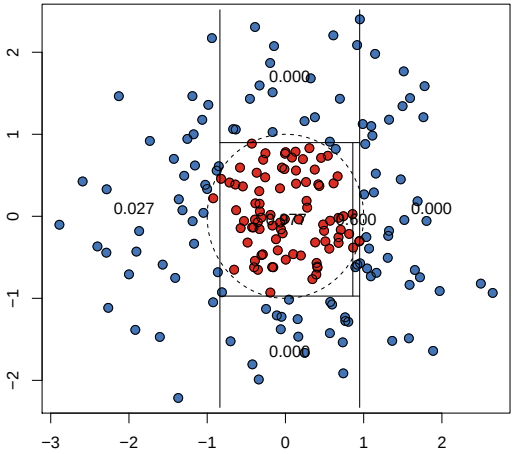
⚠ the probability that a particular data point is not selected for a bootstrap sample of size n as $(1-1/n)^n \to e^{-1} \sim 36.8\%$: each bootstrap sample is likely to leave out about a third of the data points.

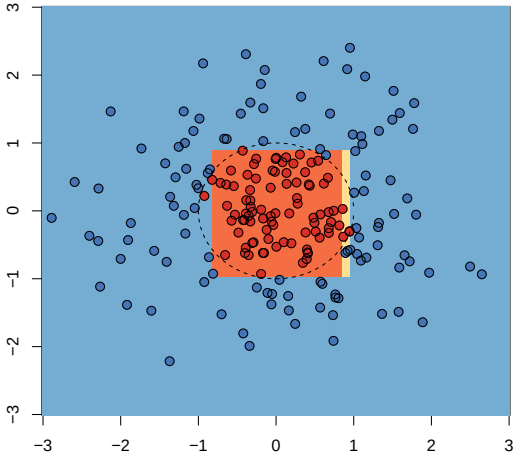
The Intuition of Bagging Algorithm

Tree cannot be used in the context of nonlinear pattern (without considering nonlinear transformation of variables)

```
> set.seed(1)
> X1 <- rnorm(200)
> X2 <- rnorm(200)
> Y <- (X1^2+X2^2)<=1
> df <- data.frame(Y,X1,X2)
```

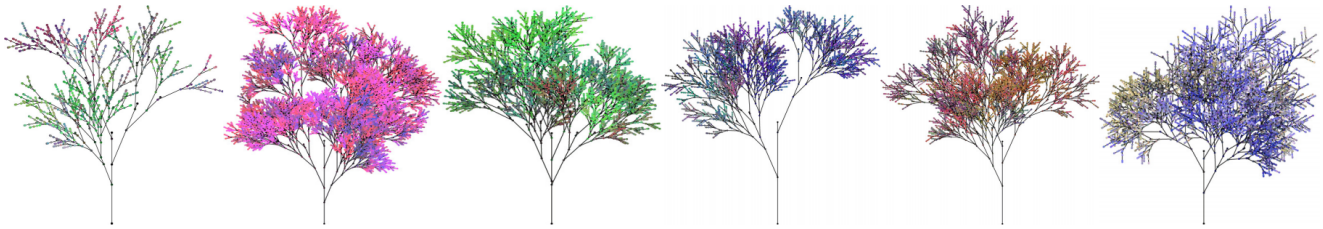
A strategy can be to aggregate several trees...

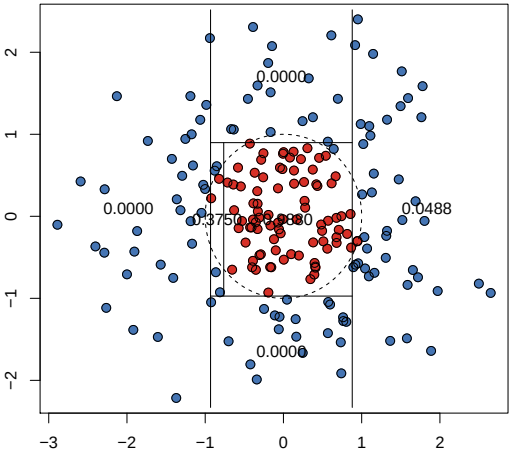


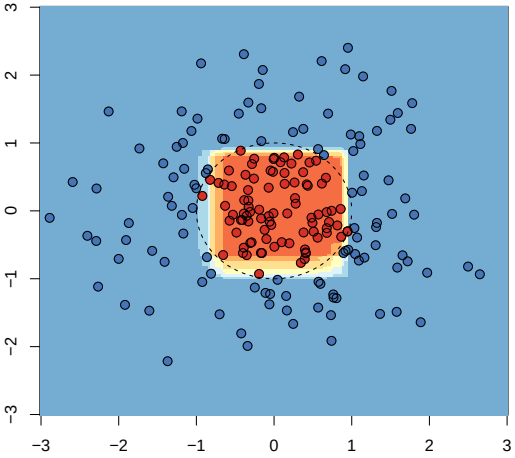


The Intuition of Bagging Algorithm

```
Y_pred <- matrix(NA,length(Y),100)
for(b in 1:100){
ind <- sample(1:n,size=n,replace=TRUE)
tree <- rpart(Y~X1+X2, data=df[ind,])
Y_pred[,b] <- predict(tree)}
Y_agg <- apply(Y_pred,1,mean)
```







From bagging to Random Forest

Strictly speaking, when boostrapping among observations, and aggregating, we use a bagging algorithm.

In the random forest algorithm, we combine Breiman's bagging idea and the random selection of features, introduced independently by Ho (1995, cm.bell-labs.com) and Amit & Geman (1997, cis.jhu.edu)

This process is sometimes called feature bagging : typically, for a dataset with p features, $\sqrt{p}$ features are used in each split.

Bagging for Classification

This approach has shown participially effective for high-variance methods such as decision trees.

```
> library(ipred)
> fit_bag <- bagging(PRONO~. , data=myocarde)
> summary(fit_bag)

Bagging classification trees with 25 bootstrap replications
```

Several trees are generated

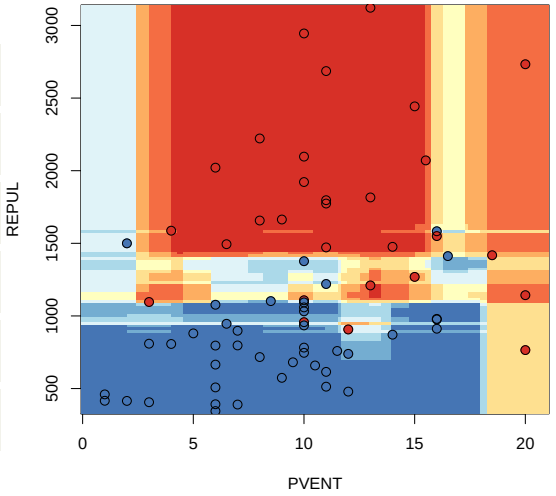
```
$mtrees
$mtrees[[1]]
$bindx
 [1] 13 17 44 28 33 20  2 42 43 46 28 48 27  2 23  4  2 10 14 69 33
    67 42 38 52 42 16 41 54 28 32 63  3 27 49 68 12 47 52
[40]  1 57 20 60 15  7 71 21 53 57 29 40 14 22 68 54 57 30 55 35 31
    42  4 31 27 13 48 13 42  2 44  9
```

```
6
7  $btree
8  n= 71
9
10 node), split, n, loss, yval, (yprob)
11       * denotes terminal node
12
13  1) root 71 33 SURVIE (0.46478873 0.53521127)
14    2) INSYS< 18.85 36  3 DECES (0.91666667 0.08333333)
15      4) INSYS>=8.85 32  0 DECES (1.00000000 0.00000000) *
16      5) INSYS< 8.85 4  1 SURVIE (0.25000000 0.75000000)
17       10) FRCAR< 106.5 1  0 DECES (1.00000000 0.00000000) *
18       11) FRCAR>=106.5 3  0 SURVIE (0.00000000 1.00000000) *
19    3) INSYS>=18.85 35  0 SURVIE (0.00000000 1.00000000) *
20
21 attr(,"class")
22    class
23 "sclass"
```

```
> predictions <- predict(fit_bag, myocarde, type="class")
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES     28      0
      SURVIE     1     42
```

The output is

```
bag2 <- bagging(PRONO~PVENT+REPUL, data=myocarde)
pred_bag = function(p,r){
return(predict(bag2,newdata=
data.frame(PVENT=p,REPUL=r),type="prob")[,1])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_bag),
    col=CL2palette)
```



Random Forests for Classification

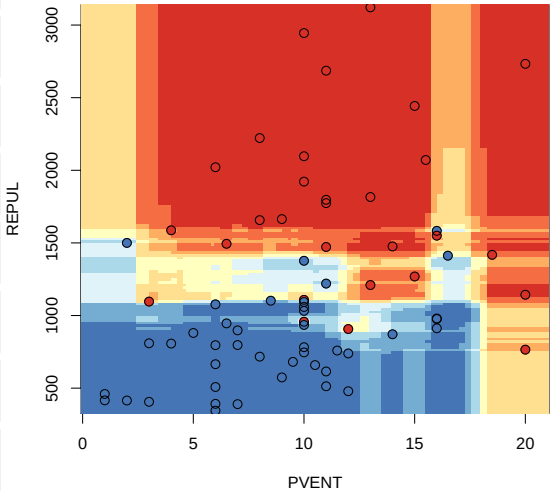
To grow forest of trees, one can consider

```
> library(randomForest)
> RF <- randomForest(PRONO~.  , data=myocarde)
> summary(RF)
                Length  Class   Mode
call               3    -none-  call
type               1    -none-  character
predicted         71    factor  numeric
err.rate        1500    -none-  numeric
confusion          6    -none-  numeric
votes            142    matrix  numeric
oob.times         71    -none-  numeric
classes            2    -none-  character
importance         7    -none-  numeric
importanceSD       0    -none-  NULL
localImportance    0    -none-  NULL
proximity          0    -none-  NULL
```

```
ntree                1   -none- numeric
mtry                 1   -none- numeric
forest              14   -none- list
y                   71   factor numeric
test                 0   -none- NULL
inbag                0   -none- NULL
terms                3   terms  call
> predictions <- predict(RF, myocarde, type="class")
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES    29      0
      SURVIE    0     42
```

Random Forests for Classification

```
RF2 <- randomForest(PRONO~PVENT+REPUL, data=
    myocarde)
pred_RF = function(p,r){
  return(predict(RF2,newdata=
                   data.frame(PVENT=p,REPUL=r),
    type="prob")[,1])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_RF),
    col=CL2palette)
```



Gradient Boosting for Classification

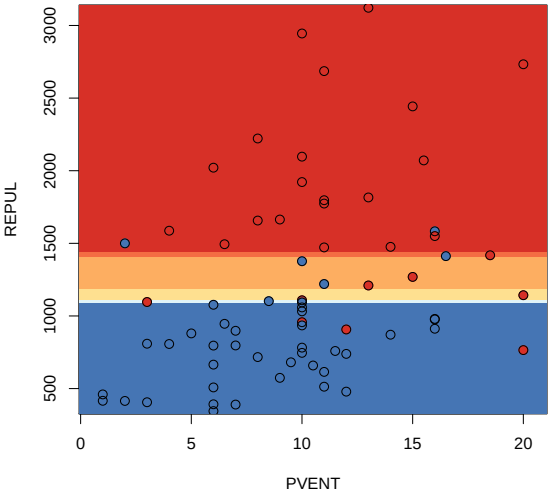
There is a dedicated package for Gradient Boosting

```
> library(gbm)
> fit_gbm <- gbm(PRONO~.  , data=myocarde, distribution="multinomial")
> print(fit_gbm)
gbm(formula = PRONO ~ ., distribution = "multinomial", data =
    myocarde)
A gradient boosted model with multinomial loss function.
100 iterations were performed.
There were 7 predictors of which 3 had non-zero influence.
```

This technique will be explained in slides #4.

Gradient Boosting for Classification

```
gbm2 <- gbm(PRONO~PVENT+REPUL, data=myocarde,
    distribution="multinomial")
pred_gbm = function(p,r){proba=predict(gbm2,
    newdata=
data.frame(PVENT=p,REPUL=r),n.trees=100,type="
    response")
  return(matrix(proba,ncol=2)[,1])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_gbm),
    col=CL2palette)
```

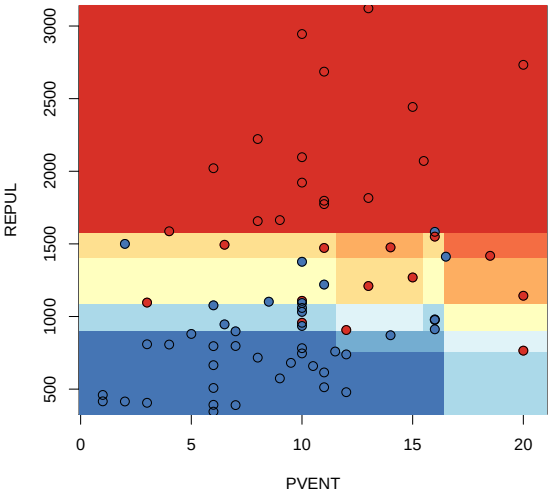


C5.0 for Classification

```
> library(C50)
> C50 <- C5.0(PRONO~.  , data=myocarde, trials=10)
> print(C50)

Call:
C5.0.formula(formula = PRONO ~ ., data = myocarde, trials = 10)

Classification Tree
Number of samples: 71
Number of predictors: 7

Number of boosting iterations: 10
Average tree size: 4.3

Non-standard options: attempt to group attributes

> predictions <- predict(C50, myocarde, type="class")
```

```
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES     29      0
      SURVIE     0     42
```

```
C502 <- C5.0(PRONO~PVENT+REPUL, data=myocarde,
    trials=10)
pred_C50 = function(p,r){proba=predict(C502,
    newdata=
                        data.frame(PVENT=p,REPUL=
    r),type="prob")
                        return(proba[,1])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_C50),
    col=CL2palette)
```

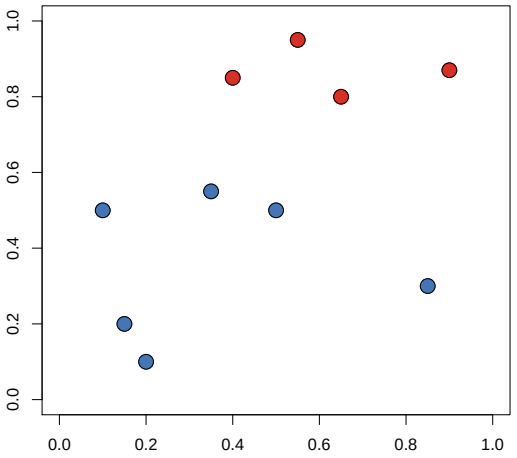


Support Vector Machine and Vapnik

SVMs were developed in the 90's based on previous work, from Vapnik & Lerner (1963, cs.iastate.edu, see Vailant (1984, people.mpi-inf.mpg.de)

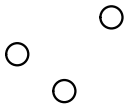
Assume that points are linearly separable, i.e. there is $\boldsymbol{\omega}$ and b such that

$$Y = \begin{cases} +1 \text{ if } \boldsymbol{\omega}^\top \boldsymbol{x} + b > 0 \\ -1 \text{ if } \boldsymbol{\omega}^\top \boldsymbol{x} + b < 0 \end{cases}$$

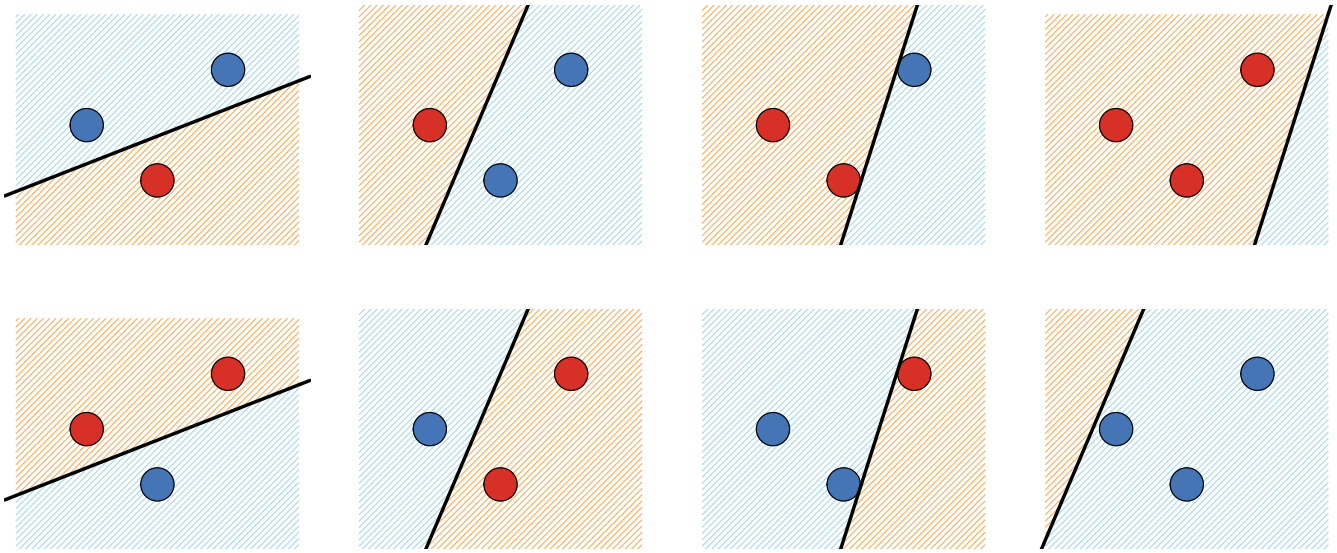


Problem: infinite number of solutions, need a good one, that separate the data, (somehow) far from the data.

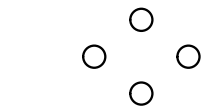
Concept : VC dimension. Let $\mathcal{H} : -h : \mathbb{R}^d \mapsto --1, +1''''$. Then $\mathcal{H}$ is said to shatter a set of points $\boldsymbol{X}$ is all dichotomies can be achieved.
E.g. with those three points, all configurations can be achieved



Support Vector Machine and Vapnik



E.g. with those four points, several configurations cannot be achieved (with some linear separator, but they can with some quadratic one)



Support Vector Machine and Vapnik

Vapnik's (VC) dimension is the size of the largest shattered subset of $\boldsymbol{X}$.

This dimension is intersting to get an upper bound of the probability of miss-classification (with some complexity penalty, function of $\text{VC}(\mathcal{H})$).

Now, in practice, where is the optimal hyperplane ?

The distance from $\boldsymbol{x}_0$ to the hyperplane $\boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}+b$ is

$$d(\boldsymbol{x}_0, H_{\boldsymbol{\omega},b}) = \frac{\boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}_0+b}{\|\boldsymbol{\omega}\|}$$

and the optimal hyperplane (in the separable case) is

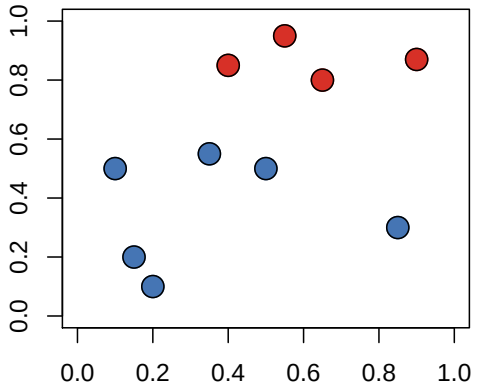
$$\text{argmin} - \min_{i=1,\cdots,n} d(\boldsymbol{x}_i, H_{\boldsymbol{\omega},b}) \text{ "}$$

Support Vector Machine and Vapnik

Define support vectors as observations such that

$$|\boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}_i + b| = 1$$

The margin is the distance between hyperplanes defined by support vectors.



The distance from support vectors to $H_{\boldsymbol{\omega},b}$ is $\|\boldsymbol{\omega}\|^{-1}$, and the margin is then $2\|\boldsymbol{\omega}\|^{-1}$.

💡 the algorithm is to minimize the inverse of the margins s.t. $H_{\boldsymbol{\omega},b}$ separates ± 1 points, i.e.

$$\min -\frac{1}{2}\boldsymbol{\omega}^\mathsf{T}\boldsymbol{\omega} \text{ '' s.t. } Y_i(\boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}_i + b) \geq 1, \ \forall i.$$

Support Vector Machine and Vapnik

Problem difficult to solve: many inequality constraints (n)

💡 solve the dual problem...

In the primal space, the solution was

$$\boldsymbol{\omega} = \sum \alpha_i Y_i \boldsymbol{x}_i \text{ with } \sum_{i=1} \alpha_i Y_i = 0.$$

In the dual space, the problem becomes (hint: consider the Lagrangian)

$$\max -\sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i=1} \alpha_i \alpha_j Y_i Y_j \boldsymbol{x}_i^\mathsf{T} \boldsymbol{x}_j \text{ '' s.t. } \sum_{i=1} \alpha_i Y_i = 0.$$

which is usually written

$$\min_{\boldsymbol{\alpha}} -\frac{1}{2} \boldsymbol{\alpha}^\mathsf{T} \boldsymbol{Q} \boldsymbol{\alpha} - \mathbf{1}^\mathsf{T} \boldsymbol{\alpha} \text{ '' s.t. } - \begin{matrix} 0 \leq \alpha_i \ \forall i \\ \boldsymbol{y}^\mathsf{T} \boldsymbol{\alpha} = 0 \end{matrix}$$

where $\boldsymbol{Q} = [\boldsymbol{Q}_{i,j}]$ and $\boldsymbol{Q}_{i,j} = y_i y_j \boldsymbol{x}_i^\mathsf{T} \boldsymbol{x}_j$.

Support Vector Machine and Vapnik

Now, what about the non-separable case?

Here, we cannot have $y_i(\boldsymbol{\omega}^\intercal \boldsymbol{x}_i + b) \geq 1 \ \forall i$.

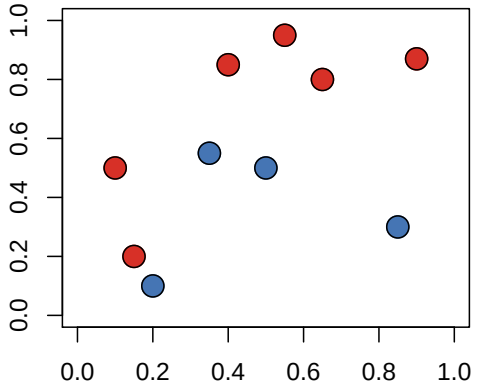
💡 introduce slack variables,

$$\begin{cases} \boldsymbol{\omega}^\intercal \boldsymbol{x}_i + b \geq +1 - \xi_i \text{ when } y_i = +1 \\ \boldsymbol{\omega}^\intercal \boldsymbol{x}_i + b \leq -1 + \xi_i \text{ when } y_i = -1 \end{cases}$$

where $\xi_i \geq 0 \ \forall i$. There is a classification error when $\xi_i > 1$.

The idea is then to solve

$$\min -\frac{1}{2}\boldsymbol{\omega}^\intercal\boldsymbol{\omega} + C\mathbf{1}^\intercal\mathbf{1}_{\boldsymbol{\xi}>1} \text{ ", instead of } \min -\frac{1}{2}\boldsymbol{\omega}^\intercal\boldsymbol{\omega} \text{ "}$$



Support Vector Machine and Vapnik

Here C is related to some - standard - tradeoff

- large C will penalize errors,
- small C will penalize complexity.

Note that the dual problem here is the same as the one before, with additional constraint, $0 \leq \alpha_i \leq C$.

$$\min_{\boldsymbol{\alpha}} -\frac{1}{2}\boldsymbol{\alpha}^\intercal \boldsymbol{Q}\boldsymbol{\alpha} - \mathbf{1}^\intercal\boldsymbol{\alpha} \text{ '' s.t. } - \begin{array}{l} 0 \leq \alpha_i \leq C \ \forall i \\ \boldsymbol{y}^\intercal\boldsymbol{\alpha} = 0 \end{array}$$

with $C \geq 0$ and $\boldsymbol{Q} = [\boldsymbol{Q}_{i,j}]$ where $\boldsymbol{Q}_{i,j} = y_i y_j \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$.

⚠ it is possible to consider some more general function here, instead of $\boldsymbol{x}_i^\intercal \boldsymbol{x}_j$

Support Vector Machine and C-classification

$$\min_{\boldsymbol{\alpha}} -\frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}\boldsymbol{\alpha} - \mathbf{1}^\top\boldsymbol{\alpha} \text{ '' s.t. } - \begin{array}{l} 0 \leq \alpha_i \leq C \ \forall i \\ \boldsymbol{y}^\top\boldsymbol{\alpha} = 0 \end{array}$$

where $C \geq 0$ is the upper bound, K is a Kernel, e.g.

- linear $K(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top\boldsymbol{v}$
- polynomial $K(\boldsymbol{u}, \boldsymbol{v}) = \gamma[\boldsymbol{u}^\top\boldsymbol{v} + c_0]^d$
- radial basis $K(\boldsymbol{u}, \boldsymbol{v}) = \exp(-\gamma\|\boldsymbol{u} - \boldsymbol{v}\|^2$

and

$$\boldsymbol{Q} = [\boldsymbol{Q}_{i,j}] \text{ where } \boldsymbol{Q}_{i,j} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- **Support Vector Machine and ν-classification**

$$\min_{\boldsymbol{\alpha}} -\frac{1}{2}\boldsymbol{\alpha}^\intercal \boldsymbol{Q}\boldsymbol{\alpha} \text{ '' s.t. } - \begin{cases} 0 \leq \alpha_i \leq \dfrac{1}{d} \ \forall i \\ \boldsymbol{y}^\intercal \boldsymbol{\alpha} = 0 \\ \mathbf{1}^\intercal \boldsymbol{\alpha} \geq \nu \end{cases}$$

with $\nu \in (0, 1]$

- **Support Vector Machine and one class-classification**

$$\min_{\boldsymbol{\alpha}} -\frac{1}{2}\boldsymbol{\alpha}^\intercal \boldsymbol{Q}\boldsymbol{\alpha} \text{ '' s.t. } - \begin{cases} 0 \leq \alpha_i \leq \dfrac{1}{\nu d} \ \forall i \\ \mathbf{1}^\intercal \boldsymbol{\alpha} = 1 \end{cases}$$

- **Support Vector Machine and ϵ-regression**

$$\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^\star} -\frac{1}{2}[\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star]^\mathsf{T}\boldsymbol{Q}[\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star]+\epsilon\mathbf{1}^\mathsf{T}[\boldsymbol{\alpha}+\boldsymbol{\alpha}^\star]+\mathbf{1}^\mathsf{T}[\boldsymbol{y}\cdot(\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star)]\ ''$$

subject to

$$\begin{aligned} & 0\leq\alpha_i,\alpha_i^\star\leq C\ \forall i \\ - & \\ & \mathbf{1}^\mathsf{T}[\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star]=0 \end{aligned}$$

- **Support Vector Machine and ν-regression**

$$\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^\star} -\frac{1}{2}[\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star]^\mathsf{T}\boldsymbol{Q}[\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star]+\boldsymbol{z}^\mathsf{T}[(\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star)]\ '' \text{ s.t. } \begin{aligned} & 0\leq\alpha_i,\alpha_i^\star\leq C\ \forall i \\ - & \\ & \mathbf{1}^\mathsf{T}[\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star]=0 \\ & \mathbf{1}^\mathsf{T}[\boldsymbol{\alpha}+\boldsymbol{\alpha}^\star]=C\nu \end{aligned}$$

From Support Vector Machine to Perceptron

SVM's belong to the class of linear classifiers.

A linear classifier is define as

$$Y^\star(\boldsymbol{x}) = \begin{cases} +1 \text{ if } B(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta} > 0 \\ -1 \text{ otherwise} \end{cases}$$

Data as linearly separable if there is a hyperplane that separate (perfectly) the two classes.

- observations such that $y_i B(\boldsymbol{x}_i) \geq 0$ are correctly classified.
- observations such that $y_i B(\boldsymbol{x}_i) \leq 0$ are misclassified.

Consider the separating hyperplane such that

$$B^\star = \text{argmin} -- \sum_{\text{misclassified}} y_i B(\boldsymbol{x}_i) \text{ ''}$$

From Support Vector Machine to Perceptron

The perceptron algorithm, introduced by Rosenblatt starts with some initial values, and then, we update

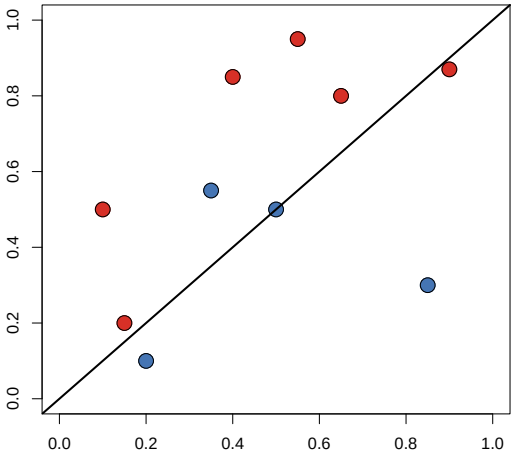
$$\begin{cases} \beta_0 \leftarrow \beta_0 + Y_i \\ \boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + Y_i \cdot \boldsymbol{X}_i \end{cases}$$

⚠ The convergence of this algorithm depends on starting values

In case of convergence, the resulting hyperplane is the maximum margin hyperplane, and points on the boundary of the margins are called support vector.

From Support Vector Machine to Perceptron

```
x=c(.4,.55,.65,.9,.1,.35,.5,.15,.2,.85)
y=c(.85,.95,.8,.87,.5,.55,.5,.2,.1,.3)
z=c(1,1,1,1,1,0,0,1,0,0)
z_sign=z*2-1
beta=c(0,-1,1)
for(i in 1:k){
beta=beta+c(z_sign[i],z_sign[i]*x[i],z_sign[i]*y[i
    ])}
```



Support Vector Machines (SVM) are a method that uses points in a transformed problem space that best separate classes into two groups. Classification for multiple classes is supported by a one-vs-all method. SVM also supports regression by modeling the function with a minimum amount of allowable error.

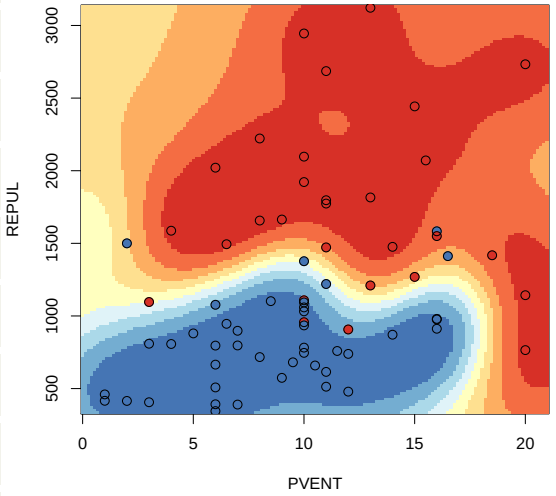
```
> library(kernlab)
> SVM <- ksvm(PRONO~., data=myocarde)
Using automatic sigma estimation (sigest) for RBF or laplace kernel
> SVM
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.146414435486797

Number of Support Vectors : 41

```

```
Objective Function Value : -23.9802
Training error : 0.070423
> predictions <- predict(SVM, myocarde, type="response")
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES     25      1
      SURVIE     4     41
```

Visualising a SVM

Consider a SVM with 2 covariates

```
SVM2 <- ksvm(PRONO~PVENT+REPUL, data=myocarde,prob.model=TRUE)
pred_SVM = function(p,r){
  return(predict(SVM2,newdata=
                   data.frame(PVENT=p,REPUL=r),type="probabilities")[,1])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_SVM),
      col=CL2palette,
        xlab="PVENT",ylab="REPUL")
```



A short remark on kernels

A kernel is a function $k(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$ that calculates a dot product in some feature space, but without constructing the feature vectors $\varphi(x)$ explicitly.

Consider the Euclidean ℓ_2 distance

$$d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_{\ell_2} = \sqrt{(\boldsymbol{x} - \boldsymbol{y}) \cdot (\boldsymbol{x} - \boldsymbol{y})} = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x} - 2\boldsymbol{x} \cdot \boldsymbol{y} + \boldsymbol{y} \cdot \boldsymbol{y}}$$

More generally, define

$$d_k(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{k(\boldsymbol{x}, \boldsymbol{x}) - 2k(\boldsymbol{x}, \boldsymbol{y}) + k(\boldsymbol{y}, \boldsymbol{y})}$$

Neural Network and Classification

A Neural Network (NN) is a graph of computational units that recieve inputs and transfer the result into an output that is passed on. The units are ordered into layers to connect the features of an input vector to the features of an output vector. With training, such as the Back-Propagation algorithm, neural networks can be designed and trained to model the underlying relationship in data.

```
> library(nnet)
> NN <- nnet(PRONO~., data=myocarde, size=10, decay=0.0001, maxit
    =500)
# weights:  91
initial  value 48.811990
iter  10 value 48.017252
iter  20 value 30.627189
iter  30 value 30.216558
iter  40 value 26.974742
```

...

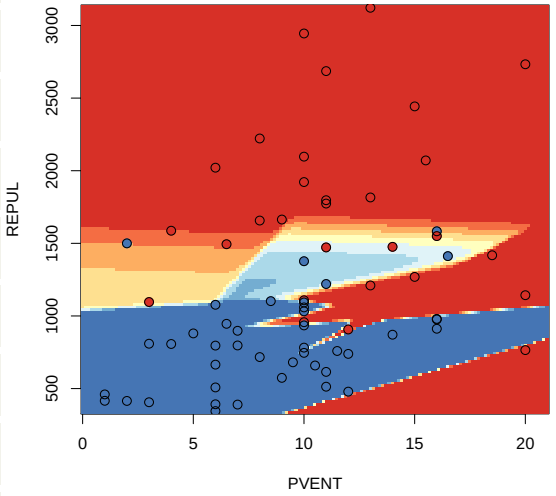
```
iter 490 value 6.801462
iter 500 value 6.797179
final  value 6.797179
stopped after 500 iterations
```

```
> predictions <- predict(NN, myocarde, type="class")
> table(predictions, myocarde$PRONO)

predictions DECES SURVIE
      DECES     27      1
      SURVIE     2     41
```

Visualising a Neural Network

Consider a NN with 2 covariates

```
NN2 <- nnet(PRONO~PVENT+REPUL , data=myocarde, size
    =20, decay=0.0001, maxit=500)
pred_NN = function(p,r){
  return(1-predict(NN2,newdata=
                     data.frame(PVENT=p,REPUL=r),
                   type="raw")[,1])}
image(vpvent,vrepul,outer(vpvent,vrepul,pred_NN),
    col=CL2palette,
      xlab="PVENT",ylab="REPUL")
```



Classification Output : terminology

	$Y=0$	$Y=1$		prevalence P=
$\widehat{Y}=0$	true negative N_{00}	false negative (type II) N_{01}	negative predictive value NPV= $\frac{N_{00}}{N_{\cdot 0}}$	false omission rate FOR= $\frac{N_{00}}{N_{\cdot 0}}$
$\widehat{Y}=1$	false positive (type I) N_{10}	true positive N_{11}	false discovery rate FDR= $\frac{N_{00}}{N_{\cdot 0}}$	positive predictive value PPV= $\frac{N_{00}}{N_{\cdot 0}}$ (precision)
negative likelihood ratio LR-=FNR/TNR	true negative rate TNR= $\frac{N_{00}}{N_{\cdot 0}}$ (specificity)	false negative rate FNR= $\frac{N_{00}}{N_{\cdot 0}}$		
positive likelihood ratio LR+=TPR/FPR	false positive rate FPR= $\frac{N_{00}}{N_{\cdot 0}}$ (fall out)	true positive rate TPR= $\frac{N_{00}}{N_{\cdot 0}}$ (sensitivity)		
diagnostic odds ratio = LR+/LR-				

Measures and Curves

- sensitivity or true positive rate (TPR) eqv. with hit rate, recall

$$TPR = TP/P = TP/(TP + FN)$$

- specificity (SPC) or True Negative Rate

$$SPC = TN/N = TN/(FP + TN)$$

- precision or positive predictive value (PPV)

$$PPV = TP/(TP + FP)$$

- negative predictive value (NPV)

$$NPV = TN/(TN + FN)$$

- fall-out or false positive rate (FPR)

$$FPR = FP/N = FP/(FP + TN) = 1 - SPC$$

Measures and Curves

- false discovery rate (FDR)

$$FDR = FP/(FP + TP) = 1 - PPV$$

- miss Rate or False Negative Rate (FNR)

$$FNR = FN/P = FN/(FN + TP)$$

- accuracy (ACC)

$$ACC = (TP + TN)/(P + N)$$

A ROC Curve

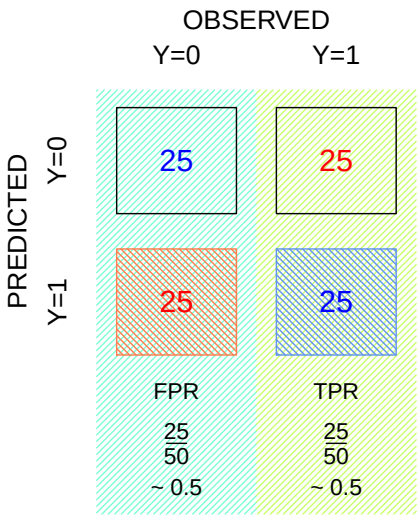
Given a classifier (and a score function), the ROC curve (receiver operating characteristic) is obtained by plotting the true positive rate against the false positive rate at various threshold s settings

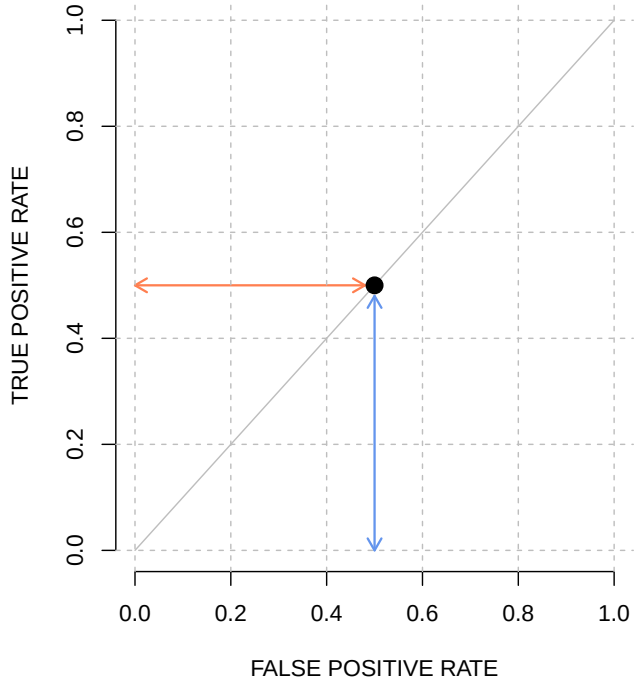
```
> roc.curve <- function(Y,S,s,print=FALSE){
 Ps <- (S>s)*1
 FP <- sum((Ps==1)*(Y==0))/sum(Y==0)
 TP <- sum((Ps==1)*(Y==1))/sum(Y==1)
 if(print==TRUE){
 print(table(Observed=Y,Predicted=Ps)) }
 vect=c(FP,TP)
 names(vect)=c("FPR","TPR")
 return(vect) }
```

ROC = sensitivity as a function of fall-out

A ROC Curve

Construction of ROC curve points given a confusion matrix





The Area Under the Curve, AUC, can be interpreted as the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one

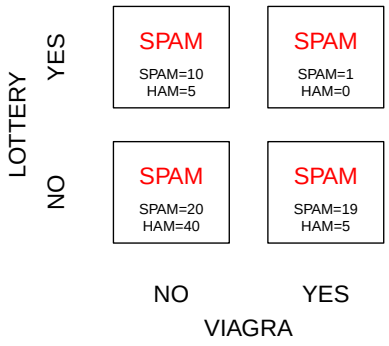
💡 AUC is related to Gini coefficient, since $G = 2\text{AUC} - 1$.

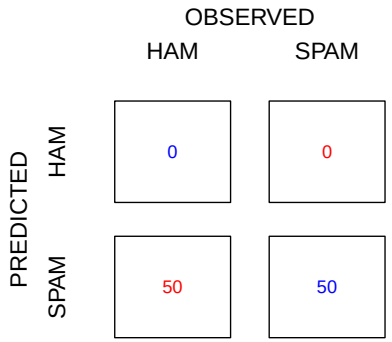
See Swets, Dawes & Monahan (ist-socrates.berkeley.edu, 2000)

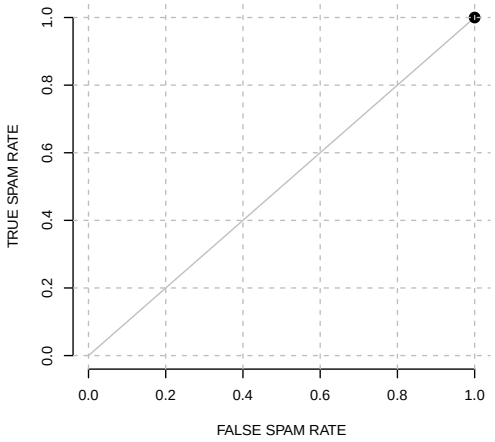
Classification with Categorical Variables

Consider a spam classifier, based on two keywords, viagra and lottery.

```
> load("spam.RData")
> head(db,4)
      Y viagra lottery
27 spam      0       1
37  ham      0       1
57 spam      0       0
89  ham      0       0
```



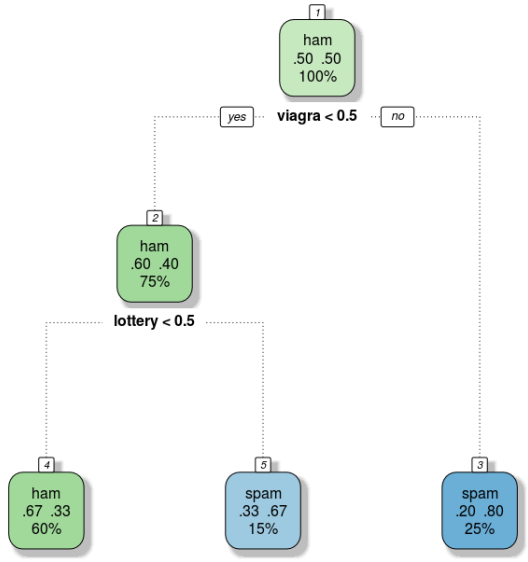




Classification with Categorical Variables

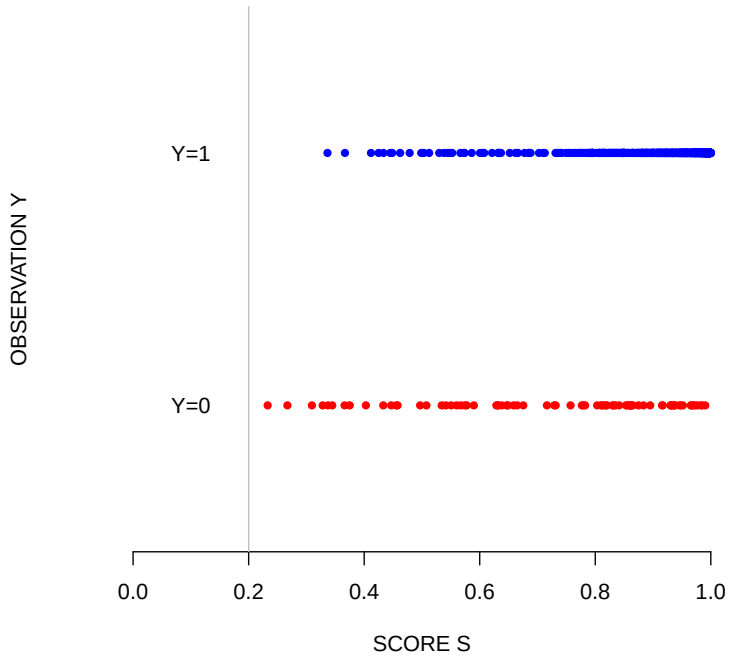
Consider e.g. a tree classifier

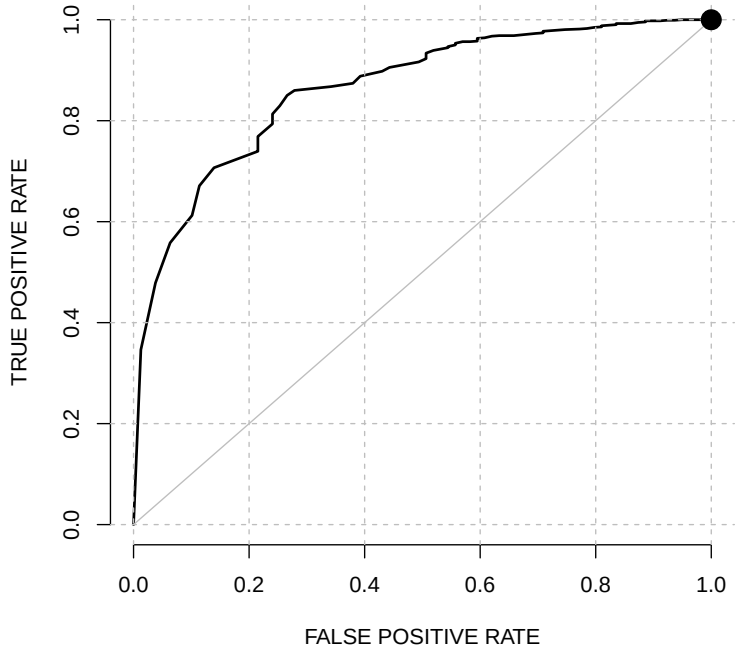
```
library(rpart)
library(rattle)
ctrl <- rpart.control(maxdepth=3)
model <- rpart(Y~viagra+lottery,
data=db, control=ctrl)
fancyRpartPlot(model)
```



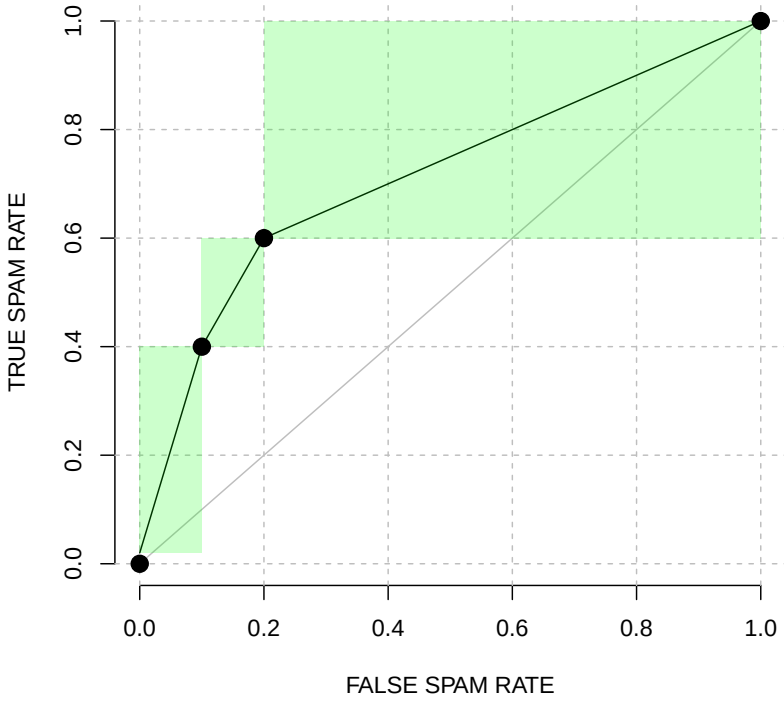
Whatever the classifier consider ROC curve will go through 4 very specific points.

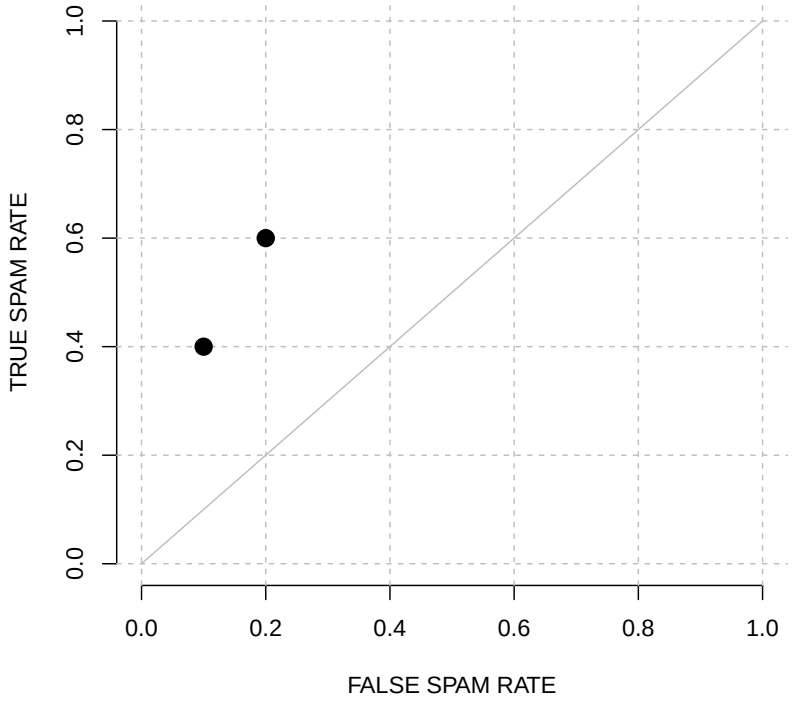
Classification with Continuous Variables





Classification with Categorical Variables





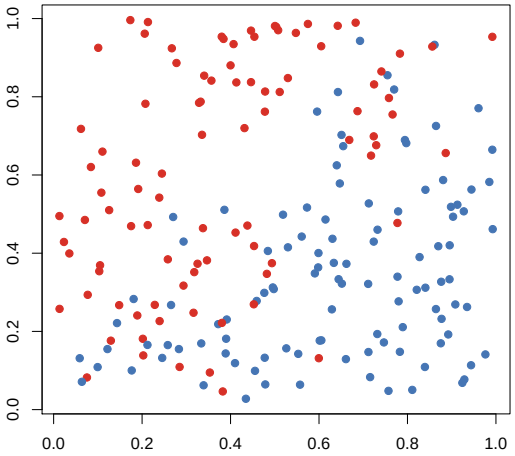
💡 interpolation between those points have different meanings

see convexification of ROC curves, see Flach (cs.bris.ac.uk, 2012)

Comparing ROC Curves

Consider a simple simulated dataset (just to illustrate)

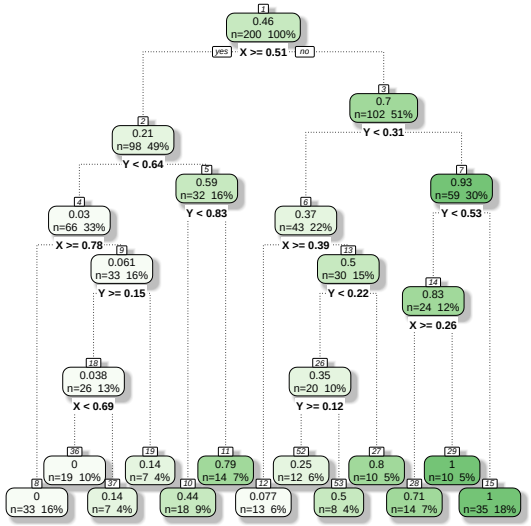
```
> X <- runif(200)
> Y <- runif(200)
> Z <- (Y>(X+rnorm(200)/4))
> df <- data.frame(X,Y,Z)
> plot(X,Y,col=c("red","blue")[1+Z*1],pch=19)
```

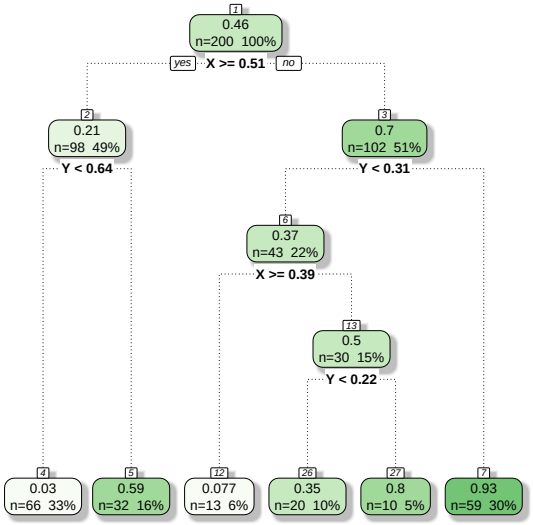


Comparing ROC Curves

On that dataset, consider two trees

```
> library(rpart)
> model1 <- rpart(Z~X+Y,data=df,control=rpart.
    control(cp = 0.00035, maxcompete = 2))
> model2 <- rpart(Z~X+Y,data=df,control=rpart.
    control(cp = 0.025, maxcompete = 2))
> library(rattle)
> fancyRpartPlot(model1)
> fancyRpartPlot(model2)
> df$s1 <- predict(modelprune1)
> df$s2 <- predict(modelprune2)
```

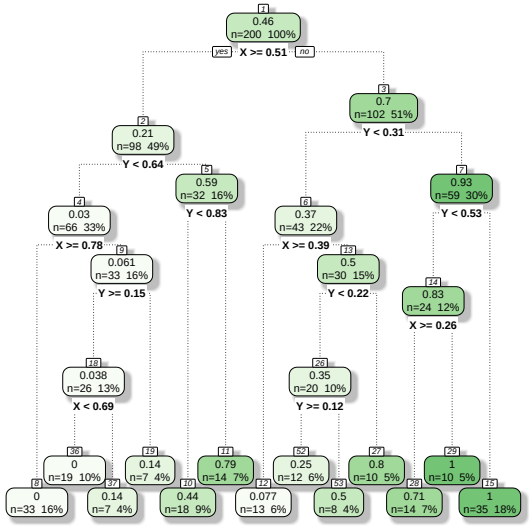


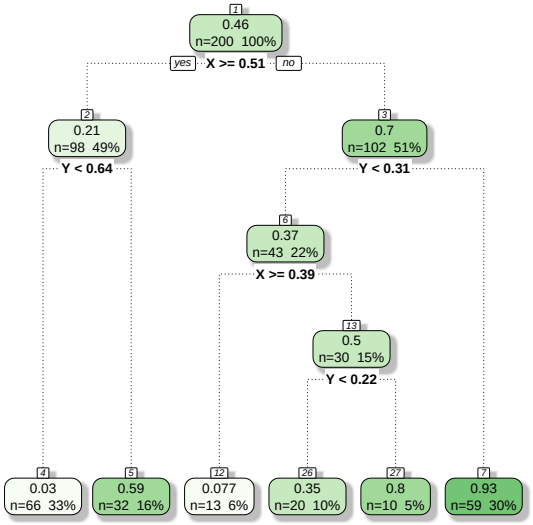


Comparing ROC Curves

On that dataset, consider two trees

```
> library(rpart)
> model1 <- rpart(Z~X+Y,data=df,control=rpart.
    control(cp = 0.00035, maxcompete = 2))
> model2 <- rpart(Z~X+Y,data=df,control=rpart.
    control(cp = 0.025, maxcompete = 2))
> library(rattle)
> fancyRpartPlot(model1)
> fancyRpartPlot(model2)
> df$s1 <- predict(modelprune1)
> df$s2 <- predict(modelprune2)
```





Comparing ROC Curves

Consider two thresholds

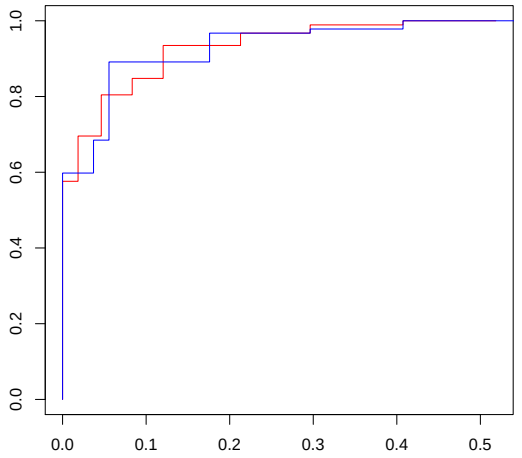
```
> s1 <- .5
> Ps1 <- (df$s1>s)*1
> Y <- df$Z
> FP=sum((Ps1==1)*(Y==0))/
sum(Y==0)
> TP=sum((Ps1==1)*(Y==1))/
sum(Y==1)
> table(Observed=Y,Predicted=P1))
          Predicted
Observed   0   1
   FALSE 99   9
   TRUE  18  74
```

```
> s2 <- .5
> Ps2 <- (df$s2>s)*1
> Y <- df$Z
> FP=sum((Ps2==1)*(Y==0))/
sum(Y==0)
> TP=sum((Ps2==1)*(Y==1))/
sum(Y==1)
> table(Observed=Y,Predicted=Ps2)
          Predicted
Observed   0   1
   FALSE 89  19
   TRUE  10  82
```

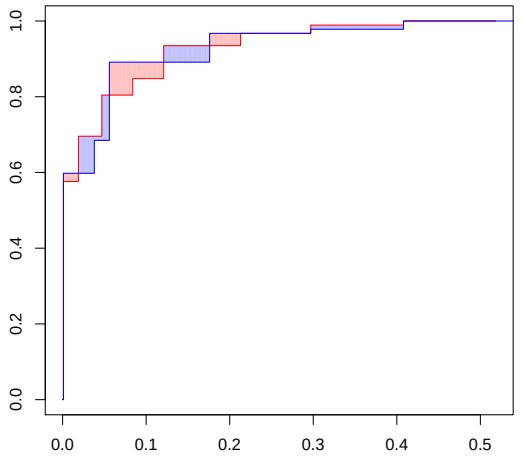
We have a (standard) tradeoff between type I and type II errors.

Comparing ROC Curves

```
> plot(roc(Z,s1,data=df),col="red")
> plot(roc(Z,s2,data=df),col="blue")
```



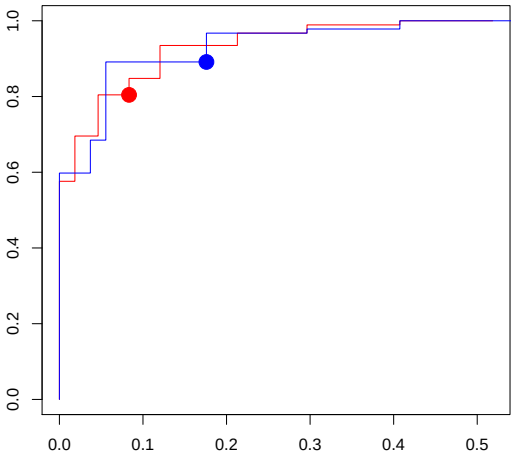
In the case of trees, this comparison is artificial, since only (lower) corners of the curves can be reached.

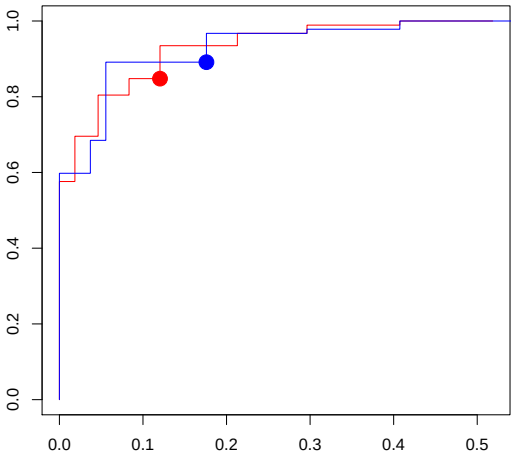


Comparing ROC Curves

```
> s1 <- .5
> s2 <- .5
> table(Observed=Y,
    Predicted=Ps1)
          Predicted
Observed   0   1
   FALSE  99   9
   TRUE   18  74
> table(Observed=Y,
    Predicted=Ps2)
          Predicted
Observed   0   1
   FALSE  89  19
   TRUE   10  82
```

```
> s1 <- .49
> s2 <- .49
> table(Observed=Y,
    Predicted=Ps1)
          Predicted
Observed   0   1
   FALSE  95  13
   TRUE   14  78
> table(Observed=Y,
    Predicted=Ps2)
          Predicted
Observed   0   1
   FALSE  89  19
   TRUE   10  82
```





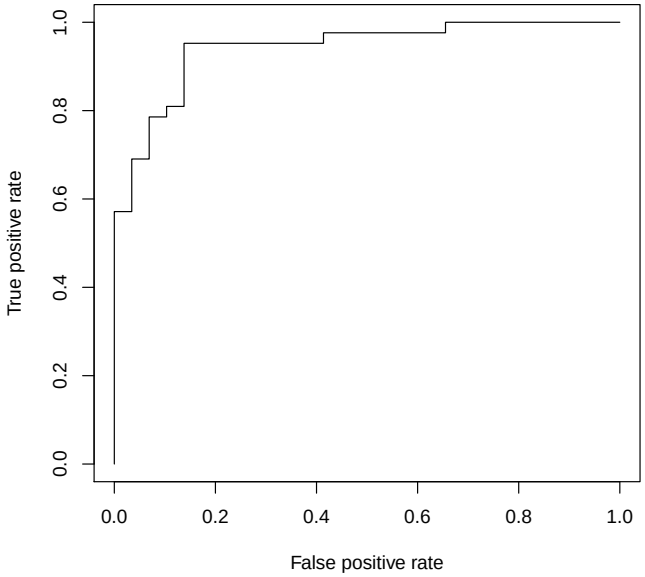
R packages for ROC curves

Consider our previous logistic regression (on heart attacks)

```
> myocarde <- read.table("myocarde.csv",head=
    TRUE,sep=";")
> logistic <- glm(PRONO~.  , data=myocarde,
    family=binomial)
> Y <- myocarde$PRONO
> S <- predict(logistic,type="response")
```

For a standard ROC curve

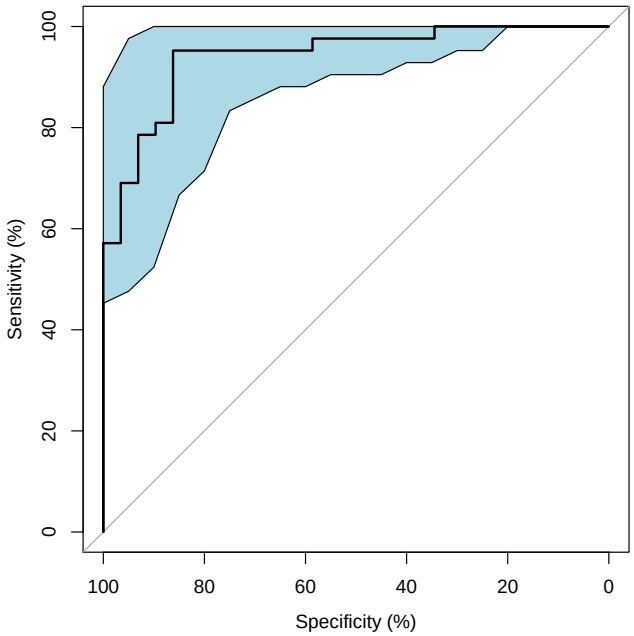
```
> library(ROCR)
> pred <- prediction(S,Y)
> perf <- performance(pred,"tpr", "fpr")
> plot(perf)
```



R packages for ROC curves

On can get confidence bands (obtained using bootstrap procedures)

```
> library(pROC)
> roc <- plot.roc(Y,S,main="", percent=TRUE, ci
    =TRUE)
> roc.se <- ci.se(roc,specificities=seq(0, 100,
     5))
> plot(roc.se, type="shape", col="light blue")
```



see also for Gains and Lift curves

```
> library(gains)
```

Standard Quantities Derived from a ROC curve

A standard quantity derived from the ROC surve is the AUC, Area Under the Curve, but many other quantities can be computed, see

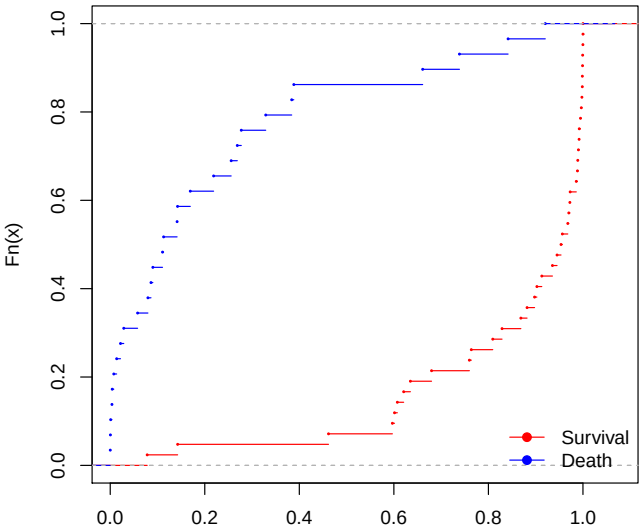
```
> library(hmeasures)
> HMeasure(Y,S)$metrics[,1:5]
Class labels have been switched from (DECES,SURVIE) to (0,1)
                H      Gini       AUC      AUCH        KS
scores 0.7323154 0.8834154 0.9417077 0.9568966 0.8144499
```

with the H-measure (see hmeasure.net), Gini and AUC, as well as the area under the convex hull (AUCH).

Standard Quantities Derived from a ROC curve

One can compute Kolmogorov-Smirnov statistics on the two conditional distributions of the score function (given either $Y = 1$ or $Y = 0$)

```
> plot(ecdf(S[Y=="SURVIE"]),main="",xlab="",
    pch=19,cex=.2,col="red")
> plot(ecdf(S[Y=="DECES"]),, pch=19,cex=.2, col
    ="blue",add=TRUE)
> max(perfy.values[[1]]-perfx.values[[1]])
[1] 0.8144499
```




```
> HMeasure(Y,S)$metrics[,6:10]
Class labels have been switched from (DECES,SURVIE) to (0,1)
                MER       MWL Spec.Sens95 Sens.Spec95        ER
scores 0.08450704 0.08966475     0.862069    0.6904762 0.09859155
```

with the minimum error rate (MER), the minimum cost-weighted error rate, etc.