

## Data Science (for Actuaries): from Small to Big Data

A. Charpentier (UQAM & Université de Rennes 1)

K.U.Leuven, June 2015.

<http://freakonometrics.hypotheses.org>



## Data Science (for Actuaries): from Small to Big Data

A. Charpentier (UQAM & Université de Rennes 1)

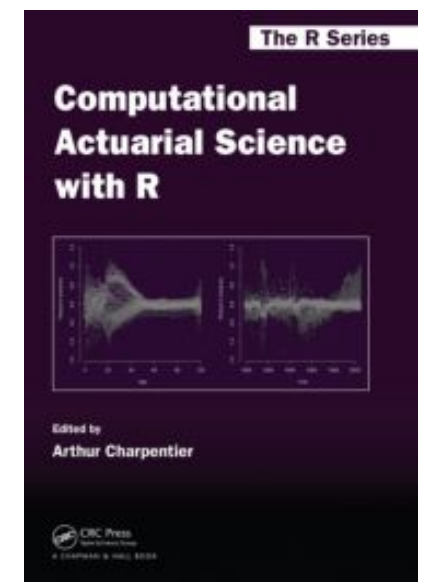
Professor of Actuarial Sciences, Mathematics Department, UQAM  
(previously Economics Department, Univ. Rennes 1 & ENSAE Paristech  
actuary in Hong Kong, IT & Stats FFSA)

PhD in Statistics (KU Leuven), Fellow Institute of Actuaries

MSc in Financial Mathematics (Paris Dauphine) & ENSAE

Editor of the [freakonometrics.hypotheses.org](http://freakonometrics.hypotheses.org)'s blog

Editor of Computational Actuarial Science, CRC



## Agenda: Small & Big Data

Actuaries (should) have a strong background on econometric models and GLMs.

What can be done on **small data**?

use of expertise  $\{y_1, \dots, y_n\}$  with  $n$  small. See also (extremely) rare event inference, [Bayesian Models](#)

## Agenda: Small & Big Data

### What can be done on **big data**?

Massive data,  $\{(y_1, \mathbf{x}_1), \dots, (y_n, \mathbf{x}_n)\}$  where  $\mathbf{x} \in \mathbb{R}^k$ .

1. the sample size  $n$  can be large (asymptotic theory,  $n \rightarrow \infty$ )
2. the number of explanatory variables  $k$  can be large: “**the more the merrier**”.  
There are  $2^k - 1$  models and submodels (hard to test all of them,  $k = 30$ , 1 billion models) and typically requires estimation of the inverse of variance matrices (complexity  $O(k^3)$ )
3. explanatory variables can be correlated

Note that massive data usually means missing values (sparsity). Answers are deletion (delete rows containing missingness), central imputation (mode, median, mean) or model based imputation.

What can we learn from **Machine Learning** theory and related techniques?



# Part 1.

## Small Data and Bayesian Philosophy

“it’s time to adopt modern Bayesian data analysis as standard procedure in our scientific practice and in our educational curriculum. Three reasons:

1. Scientific disciplines from astronomy to zoology are moving to Bayesian analysis. We should be leaders of the move, not followers.
2. Modern Bayesian methods provide richer information, with greater flexibility and broader applicability than 20th century methods. Bayesian methods are intellectually coherent and intuitive.

Bayesian analyses are readily computed with modern software and hardware.

3. Null-hypothesis significance testing (NHST), with its reliance on  $p$  values, has many problems.

There is little reason to persist with NHST now that Bayesian methods are accessible to everyone.

My conclusion from those points is that we should do whatever we can to encourage the move to Bayesian data analysis.” John Kruschke,

(quoted in Meyers & Guszczka (2013))

## Bayes vs. Frequentist, inference on heads/tails

Consider some Bernoulli sample  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ , where  $x_i \in \{0, 1\}$ .

$X_i$ 's are i.i.d.  $\mathcal{B}(p)$  variables,  $f_X(x) = p^x[1-p]^{1-x}$ ,  $x \in \{0, 1\}$ .

Standard frequentist approach

$$\hat{p} = \frac{1}{n} \sum_{i=1}^n x_i = \operatorname{argmin}_{p \in (0,1)} \left\{ \underbrace{\prod_{i=1}^n f_X(x_i)}_{\mathcal{L}(p; \mathbf{x})} \right\}$$

From the central limit theorem

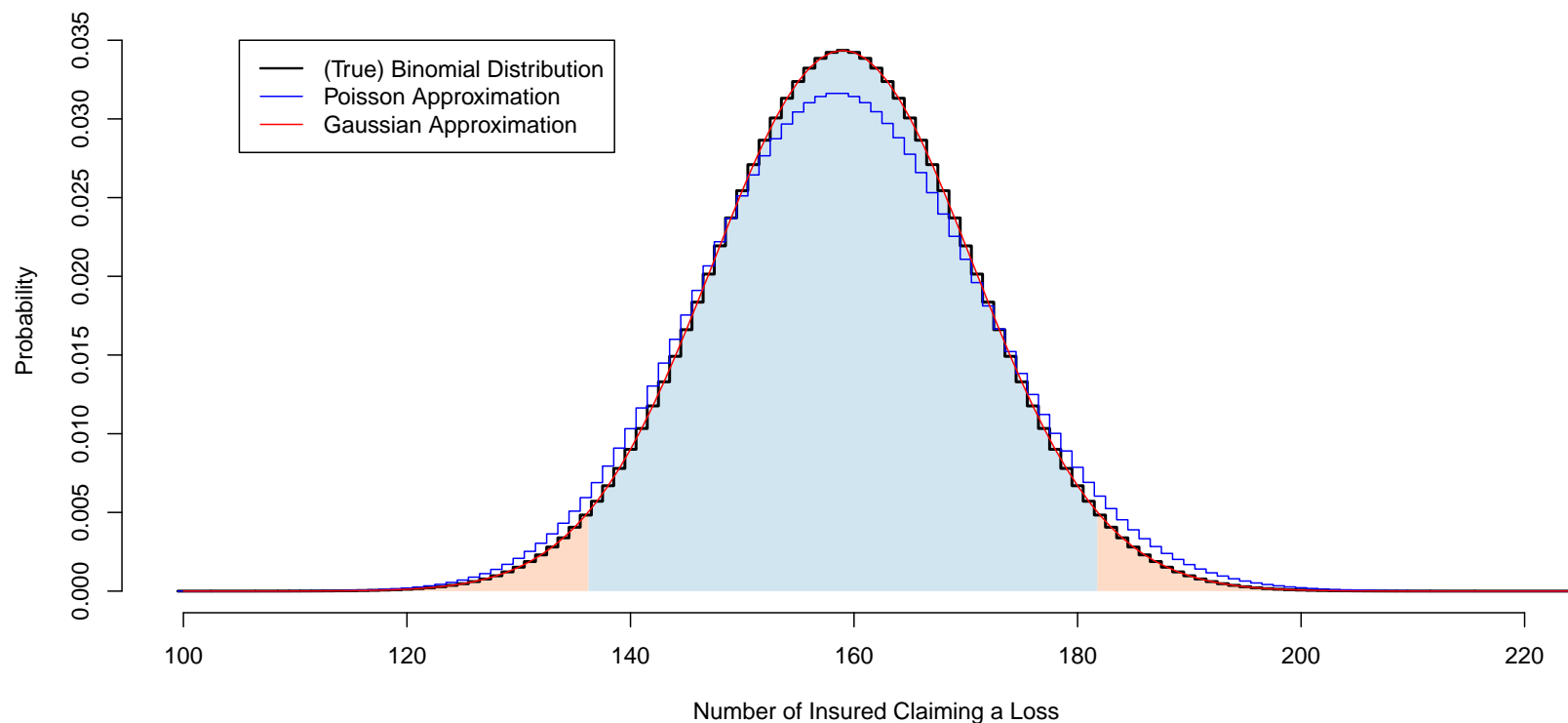
$$\sqrt{n} \frac{\hat{p} - p}{\sqrt{p(1-p)}} \xrightarrow{\mathcal{L}} \mathcal{N}(0, 1) \text{ as } n \rightarrow \infty$$

we can derive an approximated 95% confidence interval

$$\left[ \hat{p} \pm \frac{1.96}{\sqrt{n}} \sqrt{\hat{p}(1-\hat{p})} \right]$$

## Bayes vs. Frequentist, inference on heads/tails

Example out of 1,047 contracts, 159 claimed a loss








## Small Data and Black Swans

**Example** [Operational risk] What if our sample is  $\mathbf{x} = \{0, 0, 0, 0, 0\}$  ?

How would we derive a confidence interval for  $p$  ?

“INA’s chief executive officer, dressed as Santa Claus, asked an unthinkable question: Could anyone predict the probability of two planes colliding in midair? Santa was asking his chief actuary, L. H. Longley-Cook, to make a prediction based on no experience at all. There had never been a serious midair collision of commercial planes. Without any past experience or repetitive experimentation, any orthodox statistician had to answer Santa’s question with a resounding no.”

the theory   
that would   
not die   
how bayes' rule cracked  
 the enigma code,  
hunted down russian  
submarines & emerged  
triumphant from two   
centuries of controversy  
sharon bertsch mcgrayne

## *Bayes, the theory that would not die*

Liu et al. (1996) claim that “ Statistical methods with a Bayesian flavor [...] have long been used in the insurance industry”.

History of Bayesian statistics, *the theory that would not die* by Sharon Bertsch McGrayne

“[Arthur] Bailey spent his first year in New York [in 1918] trying to prove to himself that ‘all of the fancy actuarial [Bayesian] procedures of the casualty business were mathematically unsound.’ After a year of intense mental struggle, however, realized to his consternation that actuarial sledgehammering worked” [...]

## *Bayes, the theory that would not die*

[...] “ He even preferred it to the elegance of frequentism. He positively liked formulae that described ‘actual data . . . I realized that the hard-shelled underwriters were recognizing certain facts of life neglected by the statistical theorists.’ He wanted to give more weight to a large volume of data than to the frequentists small sample; doing so felt surprisingly ‘logical and reasonable’. He concluded that only a ‘suicidal’ actuary would use Fishers method of maximum likelihood, which assigned a zero probability to nonevents. Since many businesses file no insurance claims at all, Fishers method would produce premiums too low to cover future losses.”

## Bayes's theorem

Consider some hypothesis  $H$  and some evidence  $E$ , then

$$\mathbb{P}_E(H) = \mathbb{P}(H|E) = \frac{\mathbb{P}(H \cap E)}{\mathbb{P}(E)} = \frac{\mathbb{P}(H) \cdot \mathbb{P}(E|H)}{\mathbb{P}(E)}$$

Bayes rule,

$$\left\{ \begin{array}{l} \text{prior probability } \mathbb{P}(H) \\ \text{versus posterior probability after receiving evidence } E, \mathbb{P}_E(H) = \mathbb{P}(H|E). \end{array} \right.$$

In Bayesian (parametric) statistics,  $H = \{\theta \in \Theta\}$  and  $E = \{\mathbf{X} = \mathbf{x}\}$ .

Bayes' Theorem,

$$\pi(\theta|\mathbf{x}) = \frac{\pi(\theta) \cdot f(\mathbf{x}|\theta)}{f(\mathbf{x})} = \frac{\pi(\theta) \cdot f(\mathbf{x}|\theta)}{\int f(\mathbf{x}|\theta)\pi(\theta)d\theta} \propto \pi(\theta) \cdot f(\mathbf{x}|\theta)$$



## Small Data and Black Swans

Consider sample  $\mathbf{x} = \{0, 0, 0, 0, 0\}$ .

Here the likelihood is

$$\begin{cases} (x_i|\theta) = \theta^{x_i} [1 - \theta]^{1-x_i} \\ f(\mathbf{x}|\theta) = \theta^{\mathbf{x}^\top \mathbf{1}} [1 - \theta]^{n - \mathbf{x}^\top \mathbf{1}} \end{cases}$$

and we need a priori distribution  $\pi(\cdot)$  e.g.

a beta distribution

$$\pi(\theta) = \frac{\theta^\alpha [1 - \theta]^\beta}{B(\alpha, \beta)}$$

$$\pi(\theta|\mathbf{x}) = \frac{\theta^{\alpha + \mathbf{x}^\top \mathbf{1}} [1 - \theta]^{\beta + n - \mathbf{x}^\top \mathbf{1}}}{B(\alpha + \mathbf{x}^\top \mathbf{1}, \beta + n - \mathbf{x}^\top \mathbf{1})}$$

## On Bayesian Philosophy, Confidence vs. Credibility

for frequentists, a probability is a measure of the the frequency of repeated events

→ parameters are fixed (but unknown), and data are random

for Bayesians, a probability is a measure of the degree of certainty about values

→ parameters are random and data are fixed

“Bayesians : Given our observed data, there is a 95% probability that the true value of  $\theta$  falls within the credible region

vs. Frequentists : There is a 95% probability that when I compute a confidence interval from data of this sort, the true value of  $\theta$  will fall within it.” in Vanderplas (2014)

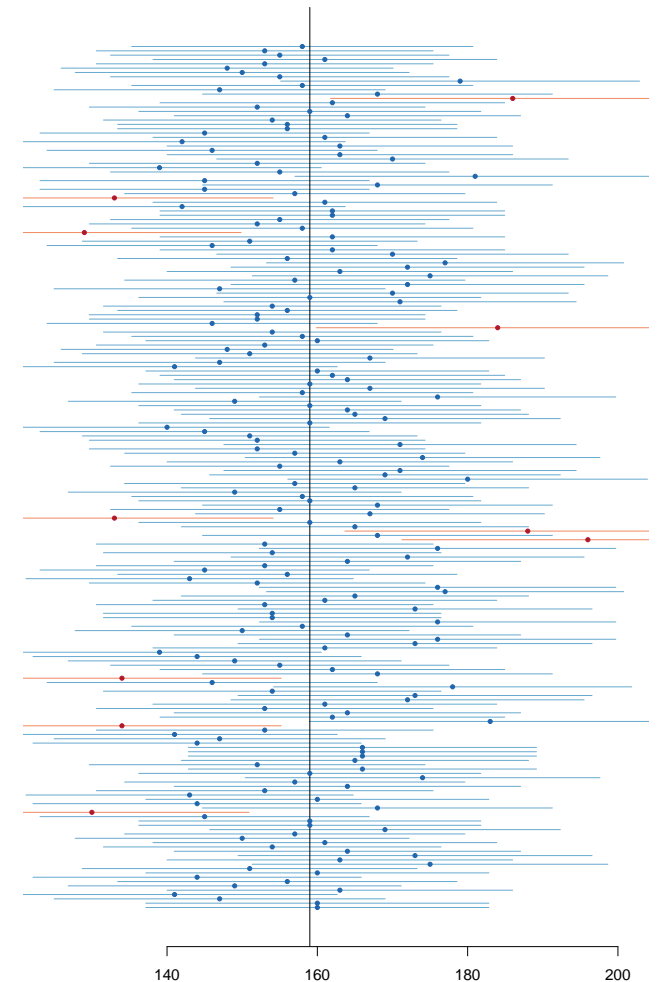
**Example** see Jaynes (1976), e.g. the truncated exponential

## On Bayesian Philosophy, Confidence vs. Credibility

**Example** What is a 95% confidence interval of a proportion ? Here  $\bar{x} = 159$  and  $n = 1047$ .

1. draw sets  $(\tilde{x}_1, \dots, \tilde{x}_n)_k$  with  $X_i \sim \mathcal{B}(\bar{x}/n)$
2. compute for each set of values confidence intervals
3. determine the fraction of these confidence interval that contain  $\bar{x}$

→ the parameter is fixed, and we guarantee that 95% of the confidence intervals will contain it.



## On Bayesian Philosophy, Confidence vs. Credibility

**Example** What is 95% credible region of a proportion ? Here  $\bar{x} = 159$  and  $n = 1047$ .

1. draw random parameters  $p_k$  with from the posterior distribution,  $\pi(\cdot|\mathbf{x})$
  2. sample sets  $(\tilde{x}_1, \dots, \tilde{x}_n)_k$  with  $X_{i,k} \sim \mathcal{B}(p_k)$
  3. compute for each set of values means  $\bar{x}_k$
  4. look at the proportion of those  $\bar{x}_k$  that are within this credible region  $[\Pi^{-1}(.025|\mathbf{x}); \Pi^{-1}(.975|\mathbf{x})]$
- the credible region is fixed, and we guarantee that 95% of possible values of  $\bar{x}$  will fall within it.

## Difficult concepts ? Difficult computations ?

We have a sample  $\mathbf{x} = \{x_1, \dots, x_d\}$  i.i.d. from distribution  $f_\theta(\cdot)$ .

In predictive modeling, we need  $\mathbb{E}(g(X)|\mathbf{x}) = \int x f_{\theta|\mathbf{x}}(x) dx$  where

$$f_{\theta|\mathbf{x}}(x) = f(x|\mathbf{x}) = \int f(x|\theta) \cdot \pi(\theta|\mathbf{x}) d\theta$$

How can we derive  $\pi(\theta|\mathbf{x})$  ?

Can we sample from  $\pi(\theta|\mathbf{x})$  (use monte carlo technique to approximate the integral) ?

Computations not that simple... until the 90's : **MCMC**

## Markov Chain

Stochastic process,  $(X_t)_{t \in \mathbb{N}_*}$ , on some discrete space  $\Omega$

$$\mathbb{P}(X_{t+1} = y | X_t = x, \underline{\mathbf{X}}_{t-1} = \underline{\mathbf{x}}_{t-1}) = \mathbb{P}(X_{t+1} = y | X_t = x) = P(x, y)$$

where  $P$  is a transition probability, that can be stored in a transition matrix,  $\mathbf{P} = [P_{x,y}] = [P(x, y)]$ .

Observe that  $\mathbb{P}(X_{t+k} = y | X_t = x) = P_k(x, y)$  where  $\mathbf{P}^k = [P_k(x, y)]$ .

Under some condition,  $\lim_{n \rightarrow \infty} \mathbf{P}^n = \mathbf{\Lambda} = [\boldsymbol{\lambda}^\top]$ ,

**Problem** given a distribution  $\boldsymbol{\lambda}$ , is it possible to generate a Markov Chain that converges to this distribution ?

## Bonus Malus and Markov Chains

Ex no-claim bonus, see [Lemaire \(1995\)](#).

### HONG KONG

Table B-9. Hong Kong System

Class	Premium	Class After		
		0	1 Claims	$\geq 2$
6	100	5	6	6
5	80	4	6	6
4	70	3	6	6
3	60	2	6	6
2	50	1	4	6
1	40	1	3	6

Starting class: 6.

Assume that the number of claims is  $N \sim \mathcal{P}(21.7\%)$ , so that  $\mathbb{P}(N = 0) = 80\%$ .

## Hastings-Metropolis

Back to our problem, we want to sample from  $\pi(\theta|\mathbf{x})$

i.e. generate  $\theta_1, \dots, \theta_n, \dots$  from  $\pi(\theta|\mathbf{x})$ .

Hastings-Metropolis sampler will generate a Markov Chain  $(\theta_t)$  as follows,

- generate  $\theta_1$
- generate  $\theta^*$  and  $U \sim \mathcal{U}([0, 1])$ ,

$$\text{compute } R = \frac{\pi(\theta^*|\mathbf{x}) P(\theta_t|\theta^*)}{\pi(\theta_t|\mathbf{x}) P(\theta^*|\theta_{t-1})}$$

if  $U < R$  set  $\theta_{t+1} = \theta^*$

if  $U \geq R$  set  $\theta_{t+1} = \theta_t$

$R$  is the acceptance ratio, we accept the new state  $\theta^*$  with probability  $\min\{1, R\}$ .



## Hastings-Metropolis

Observe that

$$R = \frac{\pi(\theta^*) \cdot f(\mathbf{x}|\theta^*)}{\pi(\theta_t) \cdot f(\mathbf{x}|\theta_t)} \frac{P(\theta_t|\theta^*)}{P(\theta^*|\theta_{t-1})}$$

In a more general case, we can have a Markov process, not a Markov chain.

E.g.  $P(\theta^*|\theta_t) \sim \mathcal{N}(\theta_t, 1)$

## Using MCMC to generate Gaussian values

```
> metrop1 <- function(n=1000,eps=0.5){  
+ vec <- vector("numeric", n)  
+ x=0  
+ vec[1] <- x  
+ for (i in 2:n) {  
+ innov <- runif(1,-eps,eps)  
+ mov <- x+innov  
+ aprob <- min(1,dnorm(mov)/dnorm(x))  
+ u <- runif(1)  
+ if (u < aprob)  
+ x <- mov  
+ vec[i] <- x  
+ }  
+ return(vec)}
```

## Using MCMC to generate Gaussian values

```
> plot.mcmc <- function(mcmc.out) {  
+ op <- par(mfrow=c(2,2))  
+ plot(ts(mcmc.out),col="red")  
+ hist(mcmc.out,30,probability=TRUE,  
+ col="light blue")  
+ lines(seq(-4,4,by=.01),dnorm(seq(-4,4,  
+ by=.01))),col="red")  
+ qqnorm(mcmc.out)  
+ abline(a=mean(mcmc.out),b=sd(mcmc.out))  
+ acf(mcmc.out,col="blue",lag.max=100)  
+ par(op) }  
  
> metrop.out<-metrop1(10000,1)  
> plot.mcmc(metrop.out)
```

## Heuristics on Hastings-Metropolis

In standard Monte Carlo, generate  $\theta_i$ 's i.i.d., then

$$\frac{1}{n} \sum_{i=1}^n g(\theta_i) \rightarrow \mathbb{E}[g(\theta)] = \int g(\theta)\pi(\theta)d\theta$$

(strong law of large numbers).

Well-behaved Markov Chains ( $\mathbf{P}$  aperiodic, irreducible, positive recurrent) can satisfy some ergodic property, similar to that LLN. More precisely,

- $\mathbf{P}$  has a unique stationary distribution  $\lambda$ , i.e.  $\lambda = \lambda \times \mathbf{P}$
- ergodic theorem

$$\frac{1}{n} \sum_{i=1}^n g(\theta_i) \rightarrow \int g(\theta)\lambda(\theta)d\theta$$

even if  $\theta_i$ 's are not independent.

## Heuristics on Hastings-Metropolis

**Remark** The conditions mentioned above are

- aperiodic, the chain does not regularly return to any state in multiples of some  $k$ .
- irreducible, the state can go from any state to any other state in some finite number of steps
- positively recurrent, the chain will return to any particular state with probability 1, and finite expected return time

## MCMC and Loss Models

**Example** A Tweedie model,  $\mathbb{E}(X) = \mu$  and  $\text{Var}(X) = \varphi \cdot \mu^p$ . Here assume that  $\varphi$  and  $p$  are given, and  $\mu$  is the unknown parameter.

→ need a predictive distribution for  $\mu$  given  $\mathbf{x}$ .

Consider the following transition kernel (a Gamma distribution)

$$\mu|\mu_t \sim \mathcal{G}\left(\frac{\mu_t}{\alpha}, \alpha\right)$$

with  $\mathbb{E}(\mu|\mu_t) = \mu_t$  and  $\text{CV}(\mu) = \frac{1}{\sqrt{\alpha}}$ .

Use some a priori distribution, e.g.  $\mathcal{G}(\alpha_0, \beta_0)$ .

## MCMC and Loss Models

- generate  $\mu_1$
- at step  $t$  : generate  $\mu^* \sim \mathcal{G}(\alpha^{-1}\mu_t, \alpha)$  and  $U \sim \mathcal{U}([0, 1])$ ,

$$\text{compute } R = \frac{\pi(\mu^*) \cdot f(\mathbf{x}|\mu^*)}{\pi(\mu_t) \cdot f(\mathbf{x}|\mu_t)} \frac{P_\alpha(\mu_t|\theta^*)}{P_\alpha(\theta^*|\theta_{t-1})}$$

if  $U < R$  set  $\theta_{t+1} = \theta^*$

if  $U \geq R$  set  $\theta_{t+1} = \theta_t$

where

$$f(\mathbf{x}|\mu) = \mathcal{L}(\mu) = \prod_{i=1}^n f(x_i|\mu, p, \varphi),$$

$f(x \cdot | \mu, p, \varphi)$  being the density of the Tweedie distribution, [dtweedie function](#) (`x`, `p`, `mu`, `phi`) from [library\(tweedie\)](#).

```
> p=2 ; phi=2/5
> set.seed(1) ; X <- rtweedie(50,p,10,phi)
> metrop2 <- function(n=10000,a0=10,
+ b0=1,alpha=1){
+ vec <- vector("numeric", n)
+ mu <- rgamma(1,a0,b0)
+ vec[1] <- mu
+ for (i in 2:n) {
+ mustar <- rgamma(1,vec[i-1]/alpha,alpha)
+ R=prod(dtweedie(X,p,mustar,phi)/dtweedie
+ (X,p,vec[i-1],phi))*dgamma(mustar,a0,b0)/
+ dgamma(vec[i-1],a0,b0)* dgamma(vec[i-1],
+ mustar/alpha,alpha)/dgamma(mustar,
+ vec[i-1]/alpha,alpha)
+ aprob <- min(1,R)
+ u <- runif(1)
+ ifelse(u < aprob,vec[i]<-mustar,
+ vec[i]<-vec[i-1]) }
+ return(vec)}
> metrop.output<-metrop2(10000,alpha=1)
```



## Gibbs Sampler

For a multivariate problem, it is possible to use Gibbs sampler.

**Example** Assume that the loss ratio of a company has a lognormal distribution,  $LN(\mu, \sigma^2)$ , .e.g

```
> LR <- c(0.958, 0.614, 0.977, 0.921, 0.756)
```

**Example** Assume that we have a sample  $\mathbf{x}$  from a  $\mathcal{N}(\mu, \sigma^2)$ . We want the posterior distribution of  $\boldsymbol{\theta} = (\mu, \sigma^2)$  given  $\mathbf{x}$ . Observe here that if priors are Gaussian  $\mathcal{N}(\mu_0, \tau^2)$  and the inverse Gamma distribution  $IG(a, b)$ , then

$$\begin{cases} \mu | \sigma^2, \mathbf{x} \sim \mathcal{N} \left( \frac{\sigma^2}{\sigma^2 + n\tau^2} \mu_0 + \frac{n\tau^2}{\sigma^2 + n\tau^2} \bar{x}, \frac{\sigma^2 \tau^2}{\sigma^2 + n\tau^2} \right) \\ \sigma^2 | \mu, \mathbf{x} \sim IG \left( \frac{n}{2} + a, \frac{1}{2} \sum_{i=1}^n [x_i - \mu]^2 + b \right) \end{cases}$$

More generally, we need the conditional distribution of  $\theta_k | \boldsymbol{\theta}_{-k}, \mathbf{x}$ , for all  $k$ .

```
> x <- log(LR)
```

## Gibbs Sampler

```
> xbar <- mean(x)
> mu <- sigma2=rep(0,10000)
> sigma2[1] <- 1/rgamma(1,shape=1,rate=1)
> Z <- sigma2[1]/(sigma2[1]+n*1)
> mu[1] <- rnorm(1,m=Z*0+(1-Z)*xbar,
+ sd=sqrt(1*Z))
> for (i in 2:10000){
+ Z <- sigma2[i-1]/(sigma2[i-1]+n*1)
+ mu[i] <- rnorm(1,m=Z*0+(1-Z)*xbar,
+ sd=sqrt(1*Z))
+ sigma2[i] <- 1/rgamma(1,shape=n/2+1,
+ rate <- (1/2)*(sum((x-mu[i])^2))+1)
+ }
```

## Gibbs Sampler

**Example** Consider some vector  $\mathbf{X} = (X_1, \dots, X_d)$  with independent components,  $X_i \sim \mathcal{E}(\lambda_i)$ . We sample to sample from  $\mathbf{X}$  given  $\mathbf{X}^\top \mathbf{1} > s$  for some threshold  $s > 0$ .

- start with some starting point  $\mathbf{x}_0$  such that  $\mathbf{x}_0^\top \mathbf{1} > s$
- pick up (randomly)  $i \in \{1, \dots, d\}$

$X_i$  given  $X_i > s - \mathbf{x}_{(-i)}^\top \mathbf{1}$  has an Exponential distribution  $\mathcal{E}(\lambda_i)$

draw  $Y \sim \mathcal{E}(\lambda_i)$  and set  $x_i = y + (s - \mathbf{x}_{(-i)}^\top \mathbf{1})_+$  until  $\mathbf{x}_{(-i)}^\top \mathbf{1} + x_i > s$

E.g. losses and allocated expenses

## Gibbs Sampler

```
> sim <- NULL
> lambda <- c(1,2)
> X <- c(3,3)
> s <- 5
> for(k in 1:1000){
+ i <- sample(1:2,1)
+ X[i] <- rexp(1,lambda[i])+
+ max(0,s-sum(X[-i]))
+ while(sum(X)<s){
+ X[i] <- rexp(1,lambda[i])+
+ max(0,s-sum(X[-i])) }
+ sim <- rbind(sim,X) }
```

## JAGS and STAN

Martyn Plummer developed **JAGS** *Just another Gibbs sampler* in 2007 (stable since 2013) in `library(runjags)`. It is an open-source, enhanced, cross-platform version of an earlier engine BUGS (Bayesian inference Using Gibbs Sampling).

**STAN** `library(Rstan)` is a newer tool that uses the Hamiltonian Monte Carlo (HMC) sampler.

HMC uses information about the derivative of the posterior probability density to improve the algorithm. These derivatives are supplied by algorithm differentiation in C/C++ codes.

## JAGS on the $\mathcal{N}(\mu, \sigma^2)$ distribution

```
> library(runjags)
> jags.model <- "
+ model {
+ mu ~ dnorm(mu0, 1/(sigma0^2))
+ g ~ dgamma(k0, theta0)
+ sigma <- 1 / g
+ for (i in 1:n) {
+ logLR[i] ~ dnorm(mu, g^2)
+ }
+ }"
```

```
> jags.data <- list(n=length(LR),
+ logLR=log(LR), mu0=-.2, sigma0=0.02,
+ k0=1, theta0=1)
```

```
> jags.init <- list(list(mu=log(1.2),
+ g=1/0.5^2),
+ list(mu=log(.8),
+ g=1/.2^2))
```

```
> model.out <- autorun.jags(jags.model,
+ data=jags.data, inits=jags.init,
+ monitor=c("mu", "sigma"), n.chains=2)
> traceplot(model.out$mcmc)
> summary(model.out)
```

## STAN on the $\mathcal{N}(\mu, \sigma^2)$ distribution

```
> library(rstan)
> stan.model <- "
+ data {
+   int<lower=0> n;
+   vector[n] LR;
+   real mu0;
+   real<lower=0> sigma0;
+   real<lower=0> k0;
+   real<lower=0> theta0;
+ }
+ parameters {
+   real mu;
+   real<lower=0> sigma;
+ }
+ model {
+   mu ~ normal(mu0, sigma0);
+   sigma ~ inv_gamma(k0, theta0);
+   for (i in 1:n)
+     log(LR[i]) ~ normal(mu, sigma);
+ }"
> stan.data <- list(n=length(LR), r=LR, mu0=mu0,
+ sigma0=sigma0, k0=k0, theta0=theta0)
> stan.out <- stan(model_code=stan.model,
+ data=stan.data, seed=2)
> traceplot(stan.out)
> print(stan.out, digits_summary=2)
```

## MCMC and Loss Models

**Example** Consider some simple time series of Loss Ratios,

$$LR_t \sim \mathcal{N}(\mu_t, \sigma^2) \text{ where } \mu_t = \phi\mu_{t-1} + \varepsilon_t$$

E.g. in JAGS we can define the vector  $\boldsymbol{\mu} = (\mu_1, \dots, \mu_T)$  recursively

```
+ model {  
+ mu[1] ~ dnorm(mu0, 1/(sigma0^2))  
+ for (t in 2:T) { mu[t] ~ dnorm(mu[t-1], 1/(sigma0^2)) }  
+ }
```



## MCMC and Claims Reserving

Consider the following (cumulated) triangle,  $\{C_{i,j}\}$ ,

	0	1	2	3	4	5
0	3209	4372	4411	4428	4435	4456
1	3367	4659	4696	4720	4730	4752.4
2	3871	5345	5398	5420	5430.1	5455.8
3	4239	5917	6020	6046.1	6057.4	6086.1
4	4929	6794	6871.7	6901.5	6914.3	6947.1
5	5217	7204.3	7286.7	7318.3	7331.9	7366.7

$\lambda_j$	1.3809	1.0114	1.0043	1.0018	1.0047
$\sigma_j$	0.7248	0.3203	0.04587	0.02570	0.02570

(from Markus Gesmann ' `library(ChainLadder)` ).

## A Bayesian version of Chain Ladder

	0	1	2	3	4	5
0		1.362418	1.008920	1.003854	1.001581	1.004735
1		1.383724	1.007942	1.005111	1.002119	
2		1.380780	1.009916	1.004076		
3		1.395848	1.017407			
4		1.378373				

$\lambda_j$	1.3809	1.0114	1.0043	1.0018	1.0047
$\sigma_j$	0.7248	0.3203	0.04587	0.02570	0.02570

Assume that  $\lambda_{i,j} \sim \mathcal{N}\left(\mu_j, \frac{\tau_j}{C_{i,j}}\right)$ .

We can use Gibbs sampler to get the distribution of the transition factors, as well as a distribution for the reserves,

```
> source("http://freakonometrics.free.fr/
triangleCL.R")
> source("http://freakonometrics.free.fr/
bayesCL.R")
> mcmcCL<-bayesian.triangle(PAID)
> plot.mcmc(mcmcCL$Lambda[,1])
> plot.mcmc(mcmcCL$Lambda[,2])
> plot.mcmc(mcmcCL$reserves[,6])
> plot.mcmc(mcmcCL$reserves[,7])

> library(ChainLadder)
> MCL<-MackChainLadder(PAID)
> m<-sum(MCL$FullTriangle[,6]-
+ diag(MCL$FullTriangle[,6:1]))
> stdev<-MCL$Total.Mack.S.E
> hist(mcmcCL$reserves[,7],probability=TRUE,
> breaks=20,col="light blue")
> x=seq(2000,3000,by=10)
> y=dnorm(x,m,stdev)
> lines(x,y,col="red")
```

## A Bayesian analysis of the Poisson Regression Model

In a Poisson regression model, we have a sample  $(\mathbf{x}, \mathbf{y}) = \{(x_i, y_i)\}$ ,

$$y_i \sim \mathcal{P}(\mu_i) \text{ with } \log \mu_i = \beta_0 + \beta_1 x_i.$$

In the Bayesian framework,  $\beta_0$  and  $\beta_1$  are random variables.

**Example:** for instance `library(arm)`, (see also `library(INLA)`)

The code is very simple : from

```
> reg<-glm(dist~speed,data=cars,family=poisson)
```

get used to

```
> regb <- bayesglm(dist~speed,data=cars,family=poisson)
```

## A Bayesian analysis of the Poisson Regression Model

```
> newd <- data.frame(speed=0:30)
> predreg <- predict(reg,newdata=
+ newd,type="response")
> plot(cars,axes)
> lines(newd$speed,predreg,lwd=2)

> library(arm)
> beta01<-coef(sim(regb))

> for(i in 1:100){
> lines(newd$speed,exp(beta01[i,1]+
> beta01[i,2]*newd$speed))}

> plot.mcmc(beta01[,1])
> plot.mcmc(beta01[,2])
```

## Other alternatives to classical statistics

Consider a regression problem,  $\mu(x) = \mathbb{E}(Y|X = x)$ , and assume that smoothed splines are used,

$$\mu(x) = \sum_{i=1}^k \beta_j h_j(x)$$

Let  $\mathbf{H}$  be the  $n \times k$  matrix,  $\mathbf{H} = [h_j(x_i)] = [\mathbf{h}(x_i)]$ , then  $\hat{\boldsymbol{\beta}} = (\mathbf{H}^\top \mathbf{H})^{-1} \mathbf{H}^\top \mathbf{y}$ , and

$$\widehat{\text{se}}(\hat{\mu}(x)) = [\mathbf{h}(x)^\top (\mathbf{H}^\top \mathbf{H})^{-1} \mathbf{h}(x)]^{\frac{1}{2}} \hat{\sigma}$$

With a Gaussian assumption on the residuals, we can derive (approximated) confidence bands for predictions  $\hat{\mu}(x)$ .

## Smoothed regression with splines

```
> dtf <- read.table(  
+ "http://freakonometrics.free.fr/  
  theftinsurance.txt", sep=";",  
+ header=TRUE)  
> names(dtf) <- c("x", "y")  
  
> library(splines)  
> reg = lm(y ~ bs(x, df=4), data=dtf)  
  
> yp = predict(reg, type="response",  
+ newdata=new, interval="confidence")
```

## Bayesian interpretation of the regression problem

Assume here that  $\beta \sim \mathcal{N}(\mathbf{0}, \tau \Sigma)$  as the priori distribution for  $\beta$ .

Then, if  $(\mathbf{x}, \mathbf{y}) = \{(x_i, y_i), i = 1, \dots, n\}$ , the posterior distribution of  $\mu(x)$  will be Gaussian, with

$$\mathbb{E}(\mu(x) | \mathbf{x}, \mathbf{y}) = \mathbf{h}(x)^\top \left( \mathbf{H}^\top \mathbf{H} + \frac{\sigma^2}{\tau} \Sigma^{-1} \right)^{-1} \mathbf{H}^\top \mathbf{y}$$

$$\text{cov}(\mu(x), \mu(x') | \mathbf{x}, \mathbf{y}) = \mathbf{h}(x)^\top \left( \mathbf{H}^\top \mathbf{H} + \frac{\sigma^2}{\tau} \Sigma^{-1} \right)^{-1} \mathbf{h}(x') \sigma^2$$

**Example**  $\Sigma = \mathbb{I}$



## Bayesian interpretation of the regression problem

```
> tau <- 100
> sigma <- summary(reg)$sigma
> H=cbind(rep(1,nrow(dtf)),matrix(bs(b$x,
+ df=4),nrow=nrow(dtf)))
> h=cbind(rep(1,nrow(new)),matrix(bs(new$x,
+ df=4),nrow=nrow(new)))
> E=h%*%solve(t(H)%*%H + sigma^2/tau*
+ diag(1,ncol(H)))%*%t(H)%*%dtf$y
> V=h%*%solve(t(H)%*%H + sigma^2/tau*
+ diag(1,ncol(H)))%*% t(h) * sigma^2
> z=E+t(chol(V))%*%rnorm(length(E))
```

## Bootstrap strategy

Assume that  $Y = \mu(x) + \varepsilon$ , and based on the estimated model, generate pseudo observations,  $y_i^* = \hat{\mu}(x_i) + \hat{\varepsilon}_i^*$ .

Based on  $(\mathbf{x}, \mathbf{y}^*) = \{(x_i, y_i^*), i = 1, \dots, n\}$ , derive the estimator  $\hat{\mu}^*(\star)$

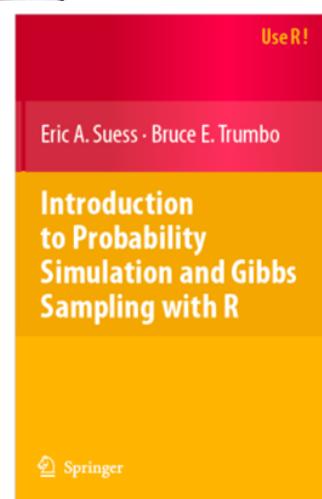
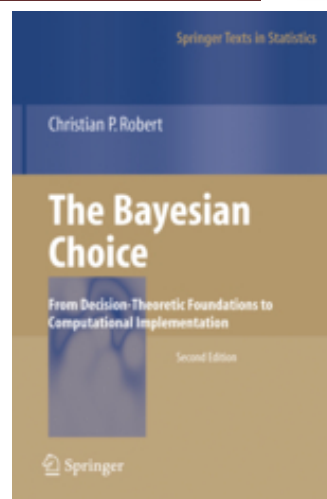
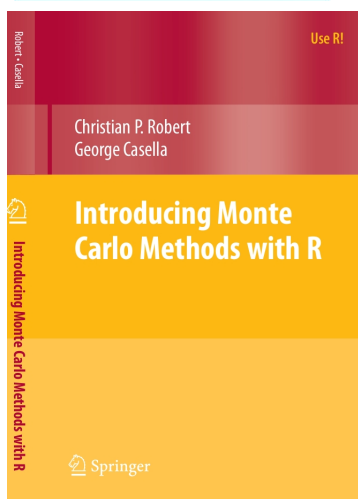
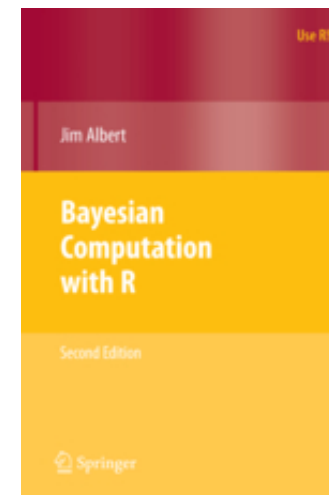
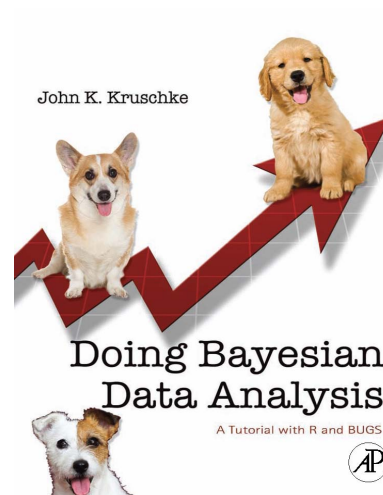
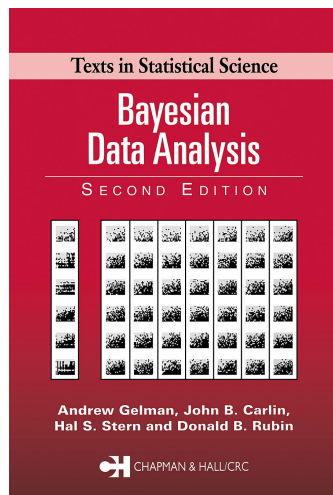
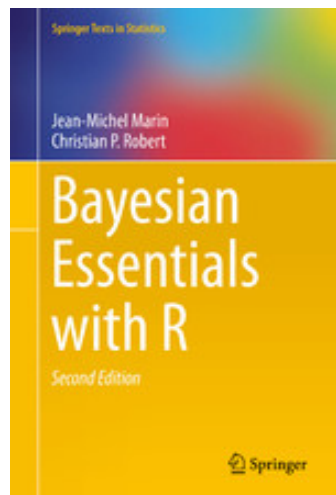
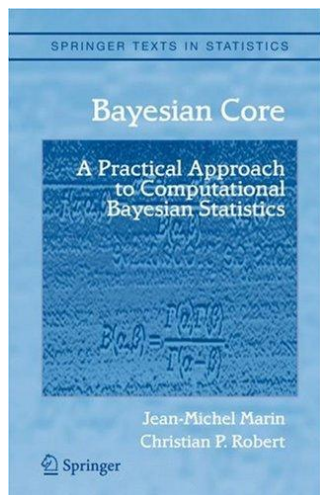
(and repeat)






## Bootstrap strategy

```
> for(b in 1:1000) {  
+ i=sample(1:nrow(dtf),size=nrow(dtf),  
+ replace=TRUE)  
+ regb=lm(y~bs(x,df=4),data=dtf[i,])  
+ ypb[,b]=predict(regb,type="response",  
+ newdata=new))  
+ }
```

Observe that the bootstrap is the Bayesian case, when  $\tau \rightarrow \infty$ .

# Some additional references (on Bayesian Modeling)



the theory  that would  
 not die   
 how bayes' rule cracked  
 the enigma code,  
 hunted down russian  
 submarines & emerged  
 triumphant from two   
 centuries of controversy  
 sharon bertsch mcgrayne

## Part 2. Big Data and Statistical/Machine Learning



## Econometric Based Models in Actuarial Science

Consider an i.i.d. sample  $\{y_1, \dots, y_n\}$  with  $y_i \in \{0, 1\}$ ,

$$\mathbb{P}(Y_i = y_i) = \pi^{y_i} [1 - \pi]^{1-y_i}, \text{ with } y_i \in \{0, 1\}.$$

where  $\pi \in [0, 1]$ , so that  $\mathbb{P}(Y_i = 1) = \pi$  and  $\mathbb{P}(Y_i = 0) = 1 - \pi$ .

The likelihood is

$$\mathcal{L}(\pi; \mathbf{y}) = \prod_{i=1}^n \mathbb{P}(Y_i = y_i) = \prod_{i=1}^n \pi^{y_i} [1 - \pi]^{1-y_i}$$

and the **log-likelihood** is

$$\log \mathcal{L}(\pi; \mathbf{y}) = \sum_{i=1}^n y_i \log[\pi] + (1 - y_i) \log[1 - \pi]$$

The first order condition is

$$\frac{\partial \log \mathcal{L}(\pi; \mathbf{y})}{\partial \pi} = \sum_{i=1}^n \frac{y_i}{\pi} - \frac{1 - y_i}{1 - \pi} = 0, \text{ i.e. } \pi^* = \bar{y}.$$

## Econometric Based Models in Actuarial Science

Assume that  $\mathbb{P}(Y_i = 1) = \pi_i$ ,

$$\text{logit}(\pi_i) = \mathbf{X}'_i \boldsymbol{\beta}, \text{ where } \text{logit}(\pi_i) = \log \left( \frac{\pi_i}{1 - \pi_i} \right),$$

or

$$\pi_i = \text{logit}^{-1}(\mathbf{X}'_i \boldsymbol{\beta}) = \frac{\exp[\mathbf{X}'_i \boldsymbol{\beta}]}{1 + \exp[\mathbf{X}'_i \boldsymbol{\beta}]}.$$

The log-likelihood is

$$\log \mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^n y_i \log(\pi_i) + (1 - y_i) \log(1 - \pi_i) = \sum_{i=1}^n y_i \log(\pi_i(\boldsymbol{\beta})) + (1 - y_i) \log(1 - \pi_i(\boldsymbol{\beta}))$$

and the first order conditions are [solved numerically](#)

$$\frac{\partial \log \mathcal{L}(\boldsymbol{\beta})}{\partial \beta_k} = \sum_{i=1}^n X_{k,i} [y_i - \pi_i(\boldsymbol{\beta})] = 0.$$

## Predictive Classification

Let  $m(\mathbf{x}) = \mathbb{E}(Y|\mathbf{X} = \mathbf{x})$ . With a logistic regression, we can get a prediction

$$\hat{m}(\mathbf{x}) = \frac{\exp[\mathbf{x}^\top \hat{\boldsymbol{\beta}}]}{1 + \exp[\mathbf{x}^\top \hat{\boldsymbol{\beta}}]}$$

```
> fit_glm <- glm(Y ~ X, family=binomial, data=df)
> m_glm <- function(x) {
+   predict( fit_glm, newdata=data.frame(X=x), type='response') }
```

Is that the ‘best’ model we can get from the data?

What if  $n$  and/or  $k$  are very large?

Can’t we use machine learning algorithms? What can statistical learning teach us?



## Using Information for Predictions

Suppose that the true model is

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

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

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

$$\begin{aligned}\widehat{\mathbf{b}}_1 &= (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top Y \\ &= (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top [\mathbf{X}_{1,i}\boldsymbol{\beta}_1 + \mathbf{X}_{2,i}\boldsymbol{\beta}_2 + \varepsilon] \\ &= (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top \mathbf{X}_1 \boldsymbol{\beta}_1 + (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top \mathbf{X}_2 \boldsymbol{\beta}_2 + (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top \varepsilon \\ &= \boldsymbol{\beta}_1 + \underbrace{(\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top \mathbf{X}_2 \boldsymbol{\beta}_2}_{\boldsymbol{\beta}_{12}} + \underbrace{(\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top \varepsilon}_{\nu_i}\end{aligned}$$

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

Note that if  $\mathbf{X}_1^\top \mathbf{X}_2 = \mathbf{0}$  ( $\mathbf{X}_1 \perp \mathbf{X}_2$ ),  $\mathbb{E}(\widehat{\mathbf{b}}_1) = \boldsymbol{\beta}_1$ .

## On Model Selection

$$\begin{cases} Y_i = \mathbf{X}_{1,i}\boldsymbol{\beta}_1 + \mathbf{X}_{2,i}\boldsymbol{\beta}_2 + \varepsilon_i, & (1, 2) \\ Y_i = \mathbf{X}_{1,i}\mathbf{b}_1 + \eta_i. & (1) \end{cases}$$

Here  $\text{Var}(\varepsilon) \leq \text{Var}(\eta)$ , so  $R^2_{(1,2)} \geq R^2_{(1)}$  and  $\log \mathcal{L}_{(1,2)} \geq \log \mathcal{L}_{(1)}$ .

For variable selection, we need to **penalize**. A standard technique is to penalize a criteria.

$$\log \mathcal{L}(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2) = -\frac{n}{2} [\log(2\pi) + \log[\hat{\sigma}^2]] - \frac{1}{2\hat{\sigma}^2} \underbrace{\|\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2}_{SSE}$$

in the context of Linear Regression,

$$AIC = n \log \frac{SSE}{n} + 2\text{dim}(\mathbf{X})$$

$$BIC = n \log \frac{SSE}{n} + \log[n]\text{dim}(\mathbf{X})$$

But it is also possible to derive a penalized estimator...

## Modeling and Predicting

Consider predictions obtained from a linear model and a nonlinear model, either on the training sample, or on a validation sample,

## Risk and Loss Function in Statistical Learning

Consider some loss function,  $L(\theta, \hat{\theta})$ , e.g. a quadratic loss function ( $\ell_2$  regression).

In the frequentist approach, the risk function is given by

$$R(\theta, \hat{\theta}) = \mathbb{E}_{\theta} \left( L(\theta, \hat{\theta}(\mathbf{X})) \right) = \int_{\mathbf{X}} L(\theta, \hat{\theta}(\mathbf{X})) d\mathbb{P}_{\theta}(\mathbf{x}).$$

In a Bayesian approach, the expectation is calculated using the posterior distribution  $\pi^*$  of the parameter  $\theta$

$$R(\theta, \hat{\theta}) = \int_{\Theta} L(\theta, \hat{\theta}) d\pi^*(\theta).$$

## Risk and Loss Function in Statistical Learning

Consider here the risk of a model  $\hat{m}_n(\cdot)$ .

The **true risk** is

$$R_n = \mathbb{E}(L) ([L(Y, \hat{m}_n(\mathbf{X}))]) = \int_{\mathcal{Y} \times \mathcal{X}} L(y, \hat{m}_n(\mathbf{x})) d\mathbb{P}(y, \mathbf{x})$$

The **empirical risk** is

$$\hat{R}_n = \frac{1}{n} \sum_{i=1}^n L(y_i, \hat{m}_n(\mathbf{x}_i))$$

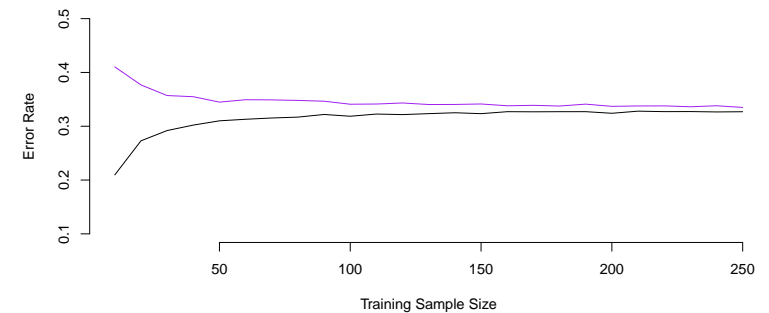
Can we say something about  $\hat{R}_n$ ?

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

## Vapnik and Consistency

Here the true model is a standard logistic

```
> 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))
```



## Vapnik and Overfitting

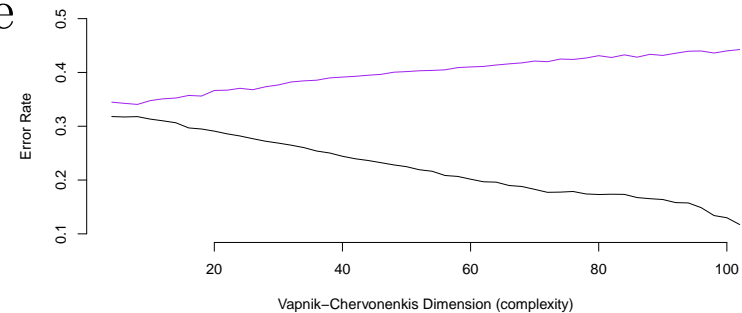
We fit some polynomial logistic regression

```
> reg <- glm(Y ~ poly(X1,s)+poly(X2,s),
+ data=U, family=binomial)
```

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

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

Here  $\text{VC} = 2(s + 1)$ .



## Penalization and Mean Square Error

Consider the quadratic loss function,  $L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$ , the risk function becomes the mean squared error of the estimate,

$$R(\theta, \hat{\theta}) = \mathbb{E}(\theta - \hat{\theta})^2 = \underbrace{[\theta - \mathbb{E}(\hat{\theta})]^2}_{\text{bias}^2} + \underbrace{\mathbb{E}(\mathbb{E}[\hat{\theta}] - \hat{\theta})^2}_{\text{variance}}$$

Get back to the initial example,  $y_i \in -0, 1$ ", with  $p = \mathbb{P}(Y = 1)$ .

Consider the estimate that minimizes the mse, that can be written  $\hat{p} = (1 - \alpha)\bar{y}$ , then

$$\text{mse}(\hat{p}) = \alpha^2 p^2 + (1 - \alpha)^2 \frac{p(1 - p)}{n}$$

$$\text{then } \alpha^* = \frac{1 - p}{1 + (n - 1)p}.$$



## Penalization and Support Vector Machines

SVMs were developed in the 90's based on previous work, from [Vapnik & Lerner \(1963\)](#), see also [Vailant \(1984\)](#).

Assume that points are [linearly separable](#), i.e. there is  $\omega$  and  $b$  such that

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

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

maximize the distance s.t.  $H_{\omega,b}$  separates  $\pm 1$  points, i.e.

$$\min -\frac{1}{2} \omega^T \omega \quad \text{s.t. } Y_i(\omega^T \mathbf{x}_i + b) \geq 1, \quad \forall i.$$

## Penalization and Support Vector Machines

Define **support vectors** as observations such that

$$|\boldsymbol{\omega}^\top \mathbf{x}_i + b| = 1$$

The margin is the distance between hyperplanes defined by support vectors. The distance from support vectors to  $H_{\boldsymbol{\omega}, b}$  is  $\|\boldsymbol{\omega}\|^{-1}$

Now, what about the **non-separable case**?

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

## Penalization and Support Vector Machines

💡 introduce **slack variables**,

$$\begin{aligned} - \quad & \omega^T \mathbf{x}_i + b \geq +1 - \xi_i \text{ when } y_i = +1 \\ & \omega^T \mathbf{x}_i + b \leq -1 + \xi_i \text{ when } y_i = -1 \end{aligned}$$

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

The idea is then to solve

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

> library(kernlab)

> fit <- ksvm(Y ~ . , data=df)

## Penalization and GLM's

The logistic regression is based on empirical risk, when  $y \in -0, 1$ "

$$-\frac{1}{n} \sum_{i=1}^n (y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \log[1 + \exp(\mathbf{x}_i^\top \boldsymbol{\beta})])$$

or, if  $y \in -1, +1$ ",

$$\frac{1}{n} \sum_{i=1}^n \log [1 + \exp(y_i \mathbf{x}_i^\top \boldsymbol{\beta})] .$$

A regularized version with the  $\ell_1$  norm is the **LASSO\*** logistic regression

$$\frac{1}{n} \sum_{i=1}^n \log [1 + \exp(y_i \mathbf{x}_i^\top \boldsymbol{\beta})] + \lambda \|\boldsymbol{\beta}\|_1$$

or more generally, with smoothing functions

$$\frac{1}{n} \sum_{i=1}^n \log [1 + \exp(y_i g(\mathbf{x}_i))] + \lambda \|g\|$$

\* Least Absolute Shrinkage and Selection Operator.

## Penalization and GLM's

We should solve

$$\hat{g} = \operatorname{argmin}_{g \in \mathcal{G}} -\frac{1}{n} \sum_{i=1}^n \log [1 + \exp(y_i g(\mathbf{x}_i))] + \lambda \|g\| "$$

Then

$$\hat{m}(\mathbf{x}) = \operatorname{sign} \left( \frac{1}{1 + \exp[-\hat{g}(\mathbf{x})]} - \frac{1}{2} \right).$$

Nothing new here.... Machine Learning is simply a “[loose confederation of themes in statistical inference \(and decision-making\)](#)”, according to Michael Jordan, with a focus on prediction.

## Using Bayes Rule on Classification

Consider the (symmetric) missclassification loss function  $L(y, \hat{y}) = \mathbf{1}(y \neq \hat{y})$ , where  $\hat{y} = m(\mathbf{x})$

The (theoretical) risk function

$$R(m) = \mathbb{E}[L(Y, m(\mathbf{X}))] = \int L(y, m(\mathbf{x})) d\mathbb{P}(y, \mathbf{x}) = \mathbb{P}(Y \neq m(\mathbf{X}))$$

The best classifier would be  $m^*$  such that

$$m^* = \operatorname{argmin}_m -\mathbb{E}[L(Y, m(\mathbf{X}))] = \operatorname{argmin}_m -\mathbb{P}(Y \neq m(\mathbf{X}))$$

which is Bayes (naive) classifier

$$m^*(\mathbf{x}) = \operatorname{argmin}_y -\mathbb{P}[Y = y | \mathbf{X} = \mathbf{x}] = \operatorname{argmin}_y -\frac{\mathbb{P}[\mathbf{X} = \mathbf{x} | Y = y]}{\mathbb{P}[\mathbf{X} = \mathbf{x}]}$$

(where  $\mathbb{P}[\mathbf{X} = \mathbf{x}]$  is the density in the continuous case).

## Using Bayes Rule on Classification

In the case where  $y$  takes two values,

$$m^*(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbb{E}(Y|\mathbf{X} = \mathbf{x}) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

and the set

$$\mathcal{D}_S = \{ \mathbf{x}, \mathbb{E}(Y|\mathbf{X} = \mathbf{x}) = \frac{1}{2} \}$$

is called the decision boundary.

$$m^*(\mathbf{x}) = \begin{cases} 1 & \text{if } r_1^2 < r_0^2 + 2 \log \frac{\mathbb{P}(Y = 1)}{\mathbb{P}(Y = 0)} + \log \frac{|\Sigma_0|}{|\Sigma_1|} \\ 0 & \text{otherwise} \end{cases}$$

where  $r_y^2$  is the Mahalanobis distance,  $r_y^2 = [\mathbf{X} - \boldsymbol{\mu}_y]^\top \boldsymbol{\Sigma}_y^{-1} [\mathbf{X} - \boldsymbol{\mu}_y]$ .

## Using Bayes Rule on Classification

If

$$\delta_y(\mathbf{x}) = -\frac{1}{2} \log |\boldsymbol{\Sigma}_y| - \frac{1}{2} [\mathbf{X} - \boldsymbol{\mu}_y]^\top \boldsymbol{\Sigma}_y^{-1} [\mathbf{X} - \boldsymbol{\mu}_y] + \log \mathbb{P}(Y = y)$$

the decision boundary of this classifier is

$$-\mathbf{x} \text{ such that } \delta_0(\mathbf{x}) = \delta_1(\mathbf{x})''$$

which is quadratic in  $\mathbf{x}$ .

this is the quadratic discriminant analysis.

If  $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1$ , then

$$\delta_y(\mathbf{x}) = [\mathbf{x}]^\top \boldsymbol{\Sigma}^{-1} [\boldsymbol{\mu}_y] - \frac{1}{2} [\boldsymbol{\mu}_y]^\top \boldsymbol{\Sigma}^{-1} [\boldsymbol{\mu}_y] + \log \mathbb{P}(Y = y)$$

which is linear in  $\mathbf{x}$ .

this is the linear discriminant analysis.



## 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)
> pL <- function(u,v) predict(fitQ,
+ newdata=data.frame(x1=u,x2=v)
+ )$posterior[,"1"])
```

## In sample, out of sample, and cross-validation

The training-validation paradigm is well known in statistics, see kernel density estimation and optimal bandwidth.

The **mean squared error** for  $\hat{m}_h(\mathbf{x})$  is  $\mathbb{E} \left[ (\hat{m}_h(\mathbf{x}) - m(\mathbf{x}))^2 \right]$ , for some meta-parameter  $h$ .

The **mean integrated squared error** is  $\int \text{mse}[\hat{m}_h(\mathbf{x})] d\mathbb{P}(\mathbf{x})$  can be approximated using its empirical version

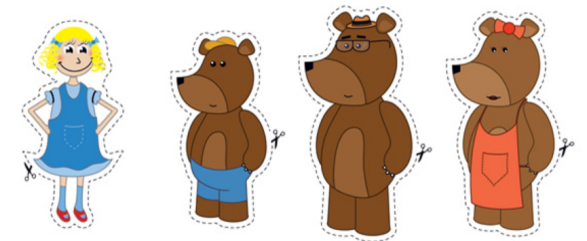
$$\widehat{\text{mise}}[\hat{m}_h] = \frac{1}{n} \sum_{i=1}^n \text{mse}[\hat{m}_h(\mathbf{X}_i)] = \frac{1}{n} \sum_{i=1}^n \text{Var}[\hat{m}_h(\mathbf{X}_i)] + \text{bias}^2[\hat{m}_h(\mathbf{X}_i)].$$

The **optimal**  $h$  would be  $h^* = \text{argmin}_h \widehat{\text{mise}}[\hat{m}_h]$ "

## In sample, out of sample, and cross-validation

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



**Problem**  $m_h(\cdot)$  is unknown, and  $\hat{m}_h(\mathbf{X}_i)$ 's are not independent.

In the **Leave-one-out Cross Validation**, we use instead  $\hat{m}_{h(-i)}(\mathbf{X}_i)$ 's. We solve

$$h^* = \frac{1}{n} \sum_{i=1}^n [Y_i - \hat{m}_{h(-i)}(\mathbf{X}_i)]^2$$

## Trees, Forests and Boosting

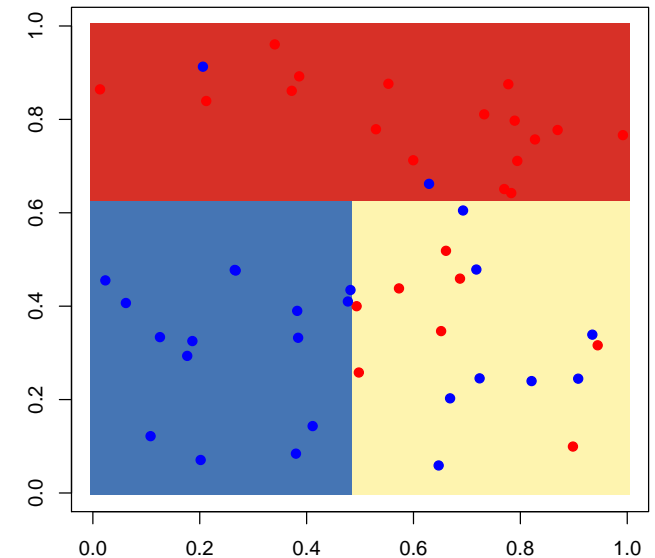
Create a partition of  $\mathcal{X} = \mathbb{R}^k$ ,  $-C_1, \dots, C_q$  and define

$$m_j^* = \operatorname{argmax}_y - \frac{1}{\#C_j} \sum_{\mathbf{x}_i \in C_j} \mathbf{1}(Y_i = y) "$$

so that

$$\hat{m}(\mathbf{x}) = \sum_j m_j^* \mathbf{1}(\mathbf{x} \in C_j)$$

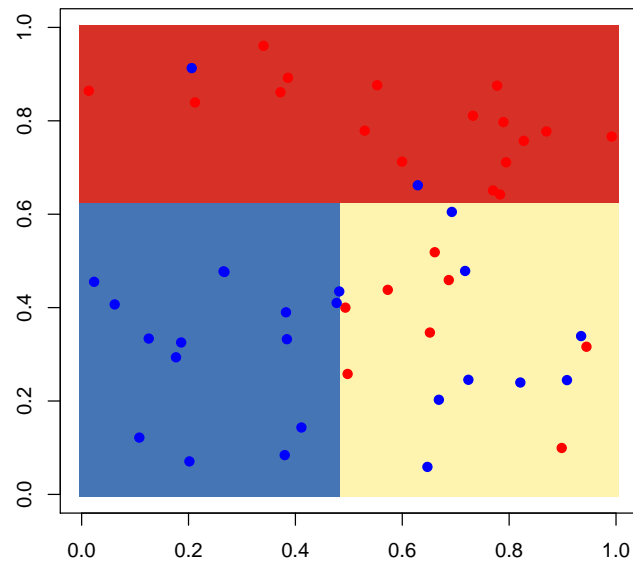
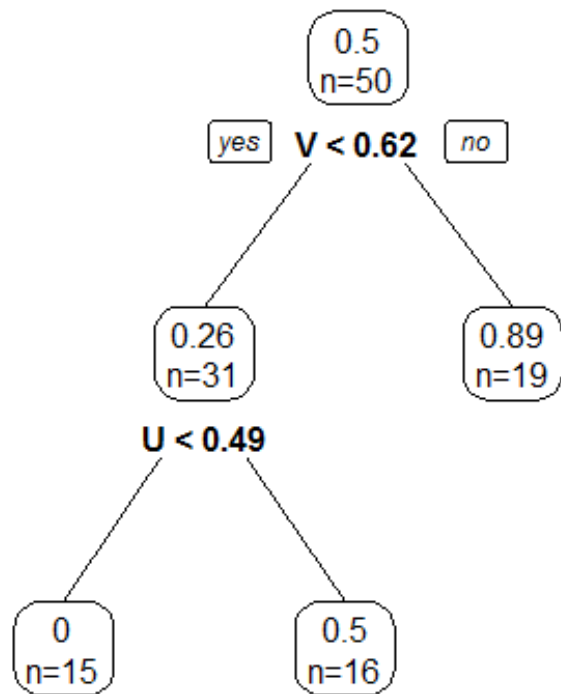
Here we seek the optimal partition  $-C_1, \dots, C_q$ .



**CART** algorithm is based on a simple (and fast) technique, for some impurity index, e.g. **Gini index**.

# Trees, Forests and Boosting

$$\text{Gini}(P) = - \sum_{P \in -A, B, C''} \underbrace{\mathbb{P}[\mathbf{x} \in P]}_{\text{weight}} \underbrace{\mathbb{P}[Y = 0 | \mathbf{x} \in P] \cdot \mathbb{P}[Y = 1 | \mathbf{x} \in P]}_{\text{impurity}}$$



## Trees

```
> gini <- 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  
+ g=-sum(omega*pxy*  
+ (1-pxy))  
+ return(g)}
```

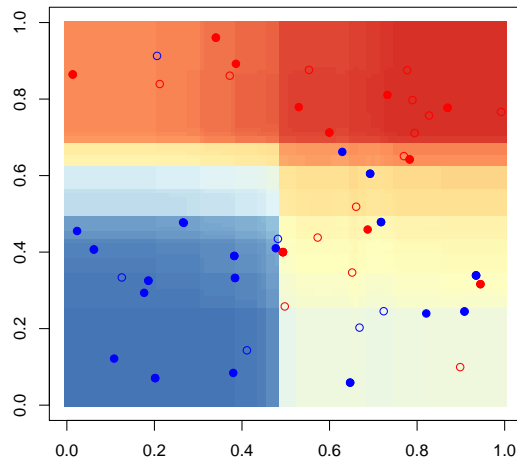
## Bagging : Bootstrap Aggregation

```
> library(randomForest)
> reg <- randomForest(y~x1+x2,data=df)
```

For classes,  $\tilde{m}(\mathbf{x}) = \operatorname{argmax}_y \sum_{b=1}^B \mathbf{1}(y = \hat{m}^{(b)})$ .

For probabilities,

$$\tilde{m}(\mathbf{x}) = \frac{1}{n} \sum_{b=1}^B \hat{m}^{(b)}(\mathbf{x}) = \frac{1}{n} \sum_{b=1}^B \sum_{j=1}^{k_b} y_j \mathbf{1}(\mathbf{x}_i \in C_j).$$



## Nonlinearities: Gradient Boosting vs. Splines

A regression problem can be formulated as

$$m^* = \operatorname{argmin}_{m \in \mathcal{M}} -\mathbb{E} (L(Y, m(\mathbf{X}))) "$$

With a parametric model, solve

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^k} -\mathbb{E} (L(Y, m_{\boldsymbol{\theta}}(\mathbf{X}))) "$$

usually using numerical algorithms...

Consider here an incremental form,  $\boldsymbol{\theta}^* = \boldsymbol{\theta}_0^* + \boldsymbol{\theta}_1^* + \dots + \boldsymbol{\theta}_M^*$ .

Standard algorithm is the (steepest) gradient descent, based on the empirical risk.



## Nonlinearities: Gradient Boosting vs. Splines

Start from  $\boldsymbol{\theta}_0^*$ . At step  $j$ ,  $\boldsymbol{\theta}^{(j)*} = \boldsymbol{\theta}_0^* + \boldsymbol{\theta}_1^* + \dots + \boldsymbol{\theta}_j^*$ , then compute

$$\boldsymbol{\theta}_{j+1}^* = -\nabla R_n(\boldsymbol{\theta}^{(j)*}) \text{ where } \nabla R_n(\boldsymbol{\theta}) = \left[ \frac{\partial R_n}{\partial \boldsymbol{\theta}_i} \right]$$

and then update  $\boldsymbol{\theta}^{(j+1)*} = \boldsymbol{\theta}^{(j)*} + \boldsymbol{\theta}_{j+1}^*$ . Finally,  $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(M)*}$ .

In a more general setting, we can do the same to get  $m^*(\mathbf{x})$ . Start from  $m_0^*(\mathbf{x})$ .

At step  $j$ ,  $m^{(j)*}(\mathbf{x}) = m_0^*(\mathbf{x}) + m_1^*(\mathbf{x}) + \dots + m_j^*(\mathbf{x})$ , then compute

$m_{j+1}^*(\mathbf{x}) = -\nabla R_n(m^{(j)*}(\mathbf{x}))$  (or sort of).

Here optimization is in a function space. Assume that  $-\nabla R_n(m^{(j)*}(\boldsymbol{\theta}))$  is expressed in a parametric family of **bases learner** functions,  $h(\cdot, \boldsymbol{\theta})$ ,

$$\boldsymbol{\theta}_j^* = \operatorname{argmin}_{\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^k} -\mathbb{E} \left( L(Y, m^{(j)*}(\mathbf{x}) + h(\mathbf{x}, \boldsymbol{\theta})) \right) "$$

Then update  $m^{(j+1)*}(\mathbf{x}) = m^{(j)*}(\mathbf{x}) + h(\mathbf{x}, \boldsymbol{\theta}_j^*)$ .

## Nonlinearities: Gradient Boosting vs. Splines

For the  $\ell_2$  loss function, it is simply based on residual (re)fitting since

$$\left[ \frac{\partial R_n}{\partial \boldsymbol{\theta}_k} \right] = \sum_{i=1}^n \omega_i [Y_i - m^{(j)*}(\mathbf{X}_i)]$$

In practice, learning should be weak, e.g. tree based learners  $h(\cdot, \boldsymbol{\theta})$ .

But it is also possible to consider splines smoothers to obtain also good model.

## Nonlinearities

```
> library(dismo)
> reg <- gbm.step(data=db,gbm.x=1,gbm.y=2,
+ family="gaussian",tree.complexity=5,
+ learning.rate=0.01,bag.fraction=0.5)
```

💡 for linear splines, consider

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

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

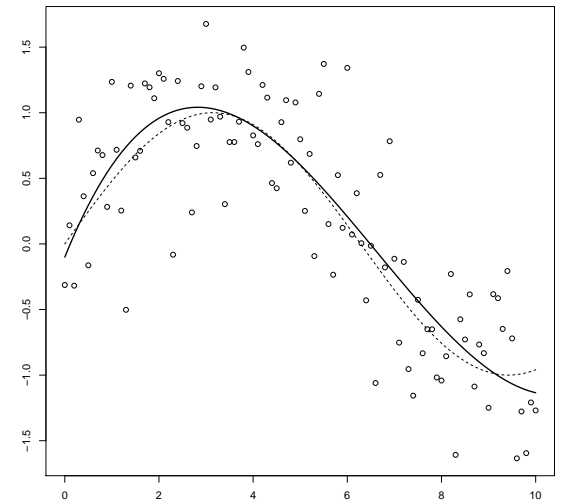
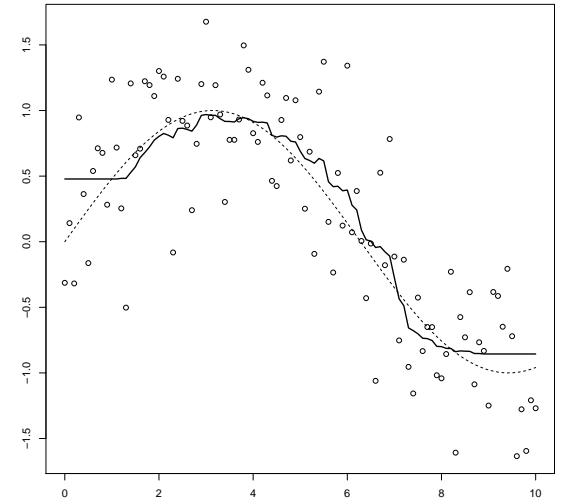
## Nonlinearities

```
> library(dismo)
> reg <- gbm.step(data=db,gbm.x=1,gbm.y=2,
+ family="gaussian",tree.complexity=5,
+ learning.rate=0.01,bag.fraction=0.5)
```

💡 for linear splines, consider

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

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



## Comparing Various Models

Consider simulated data, based on the following

$m(x_1, x_2)$  function

```
> m <- function(x1, x2) { sin(x1+x2)/(x1+x2) }
```

with some additional Gaussian noise

```
> df <- data.frame(x1=(runif(n, min=1, max=6)),
```

```
+ x2=(runif(n, min=1, max=6)))
```

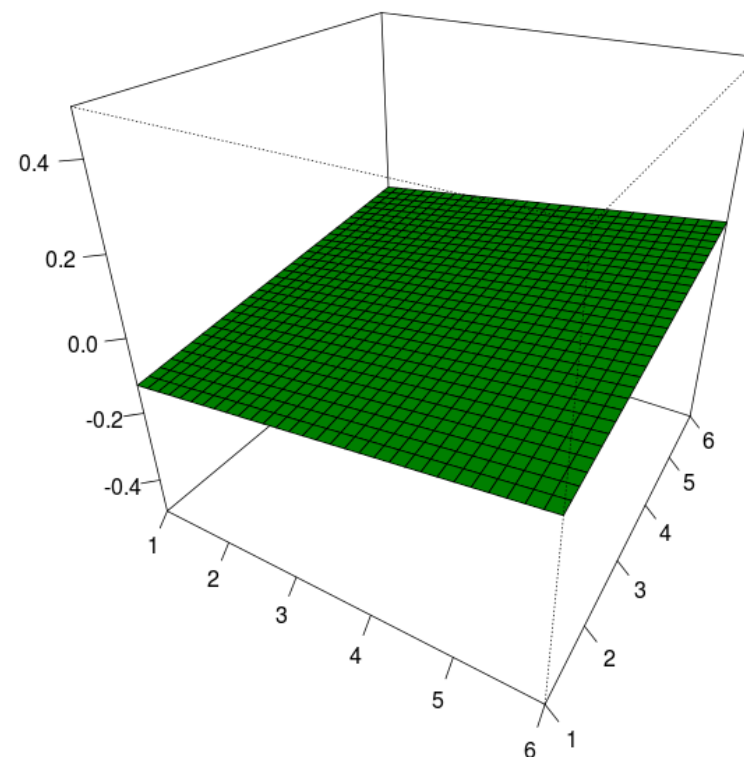
```
> df$m=m(df$x1, df$x2)
```

```
> df$y=df$m+rnorm(n,sd=.07)
```

## Comparing Various Models

A standard regression model is not good

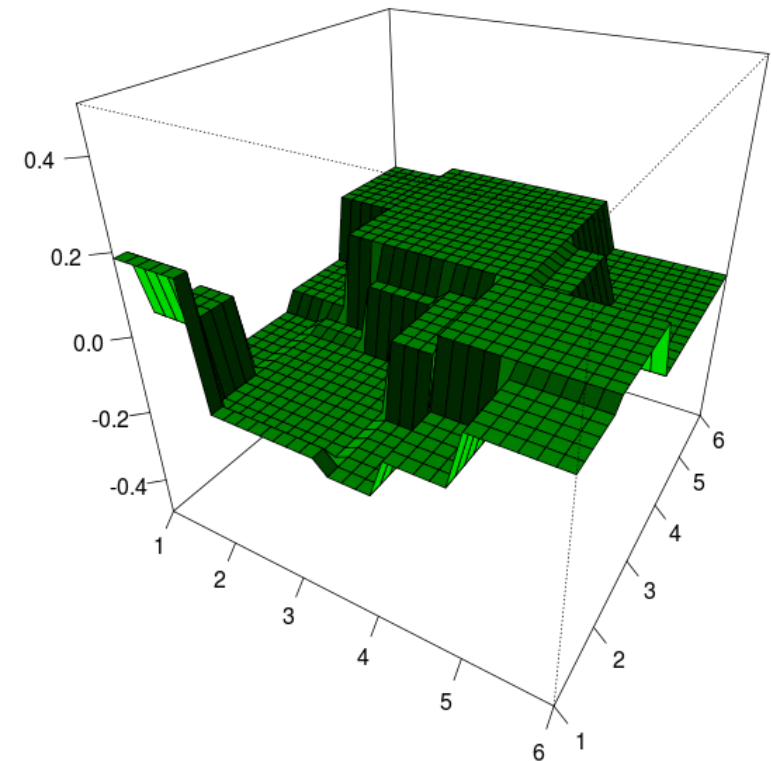
```
> reg <- lm(y~x1+x2,data=df)
```



## Comparing Various Models

... but neither is a [regression tree](#)

```
> reg <- rpart(y~x1+x2,data=df,method="anova")
```

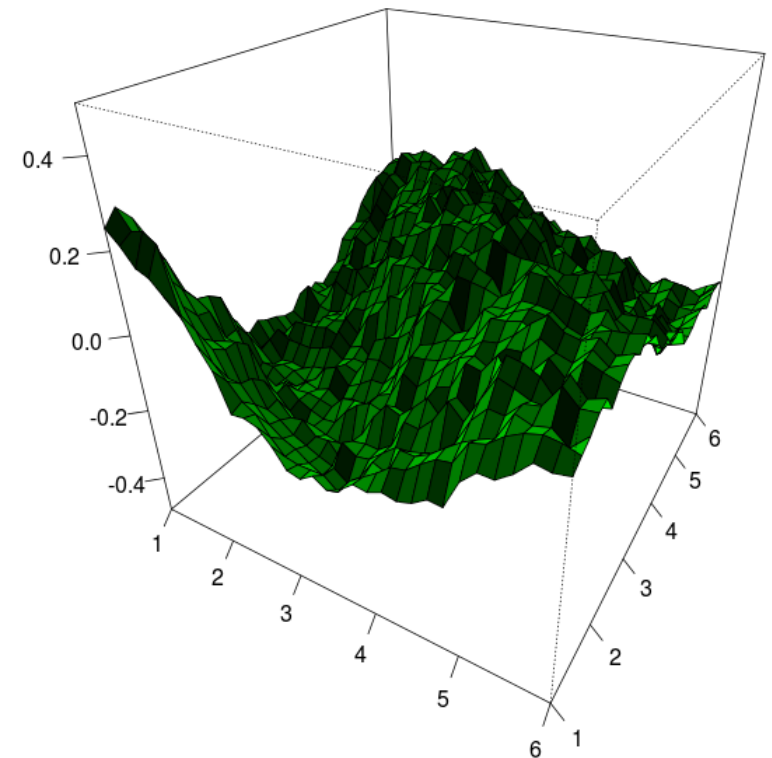


## Comparing Various Models

but **random forests** are nice

```
> library(randomForest)
```

```
> reg <- randomForest(y~x1+x2,data=df)
```

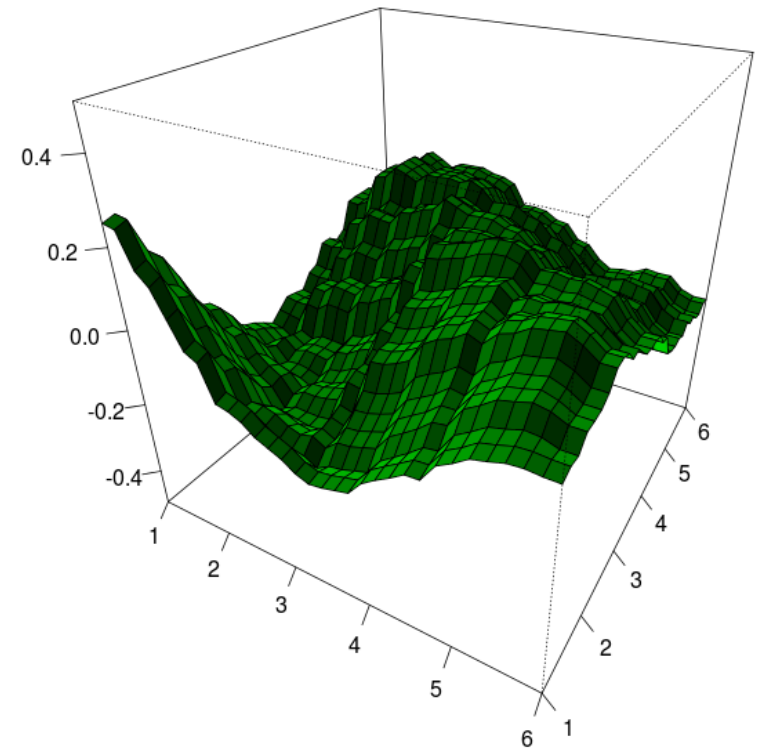




## Comparing Various Models

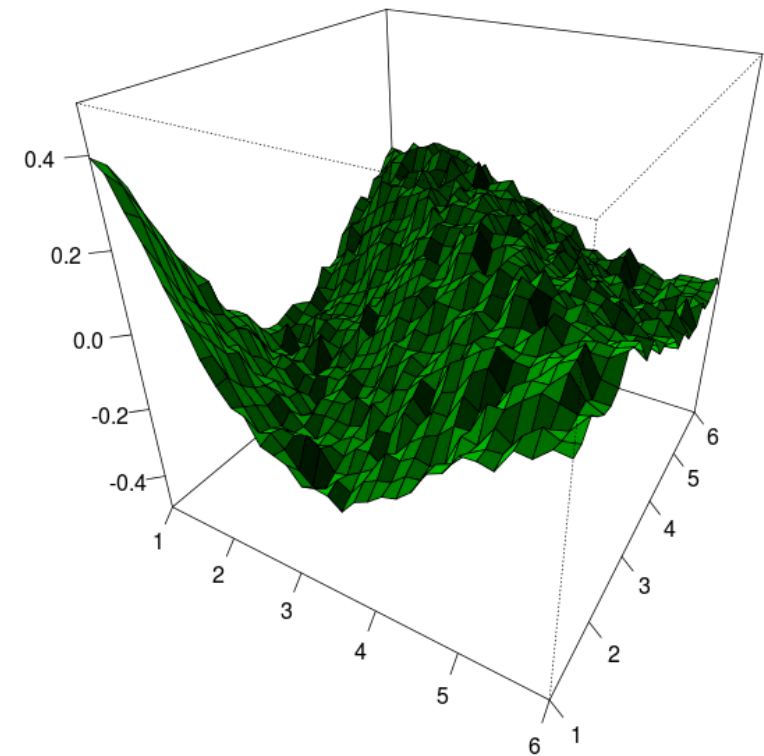
and gradient boosting algorithms too

```
> library(dismo)
> reg <- gbm.step(data=df, gbm.x = 1:2, gbm.y = 4,
+ family = "gaussian", tree.complexity = 5,
+ learning.rate = 0.01, bag.fraction = 0.5)
```



## Comparing Various Models

But one can also get a nice model with a simple *k*-nearest neighbour

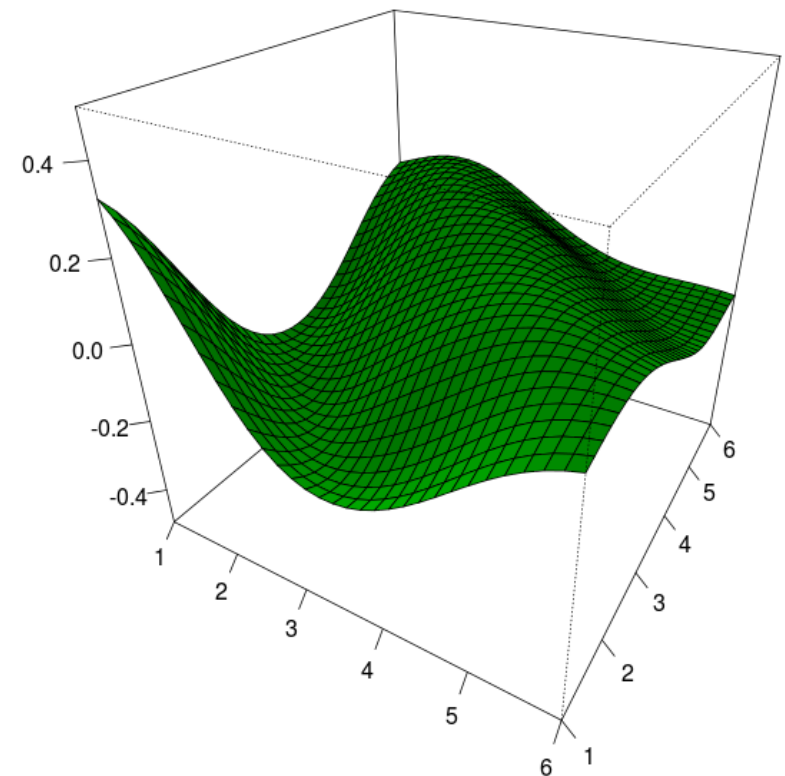


## Comparing Various Models

... or using **bivariate splines**

```
> library(mgcv)
```

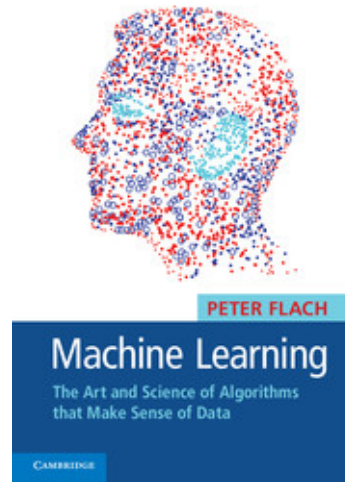
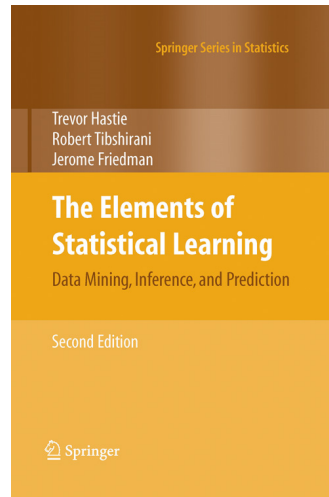
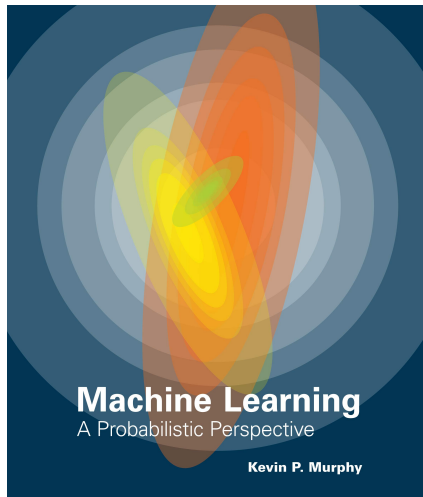
```
> reg <- gam(y~s(x1,x2),data=df)
```



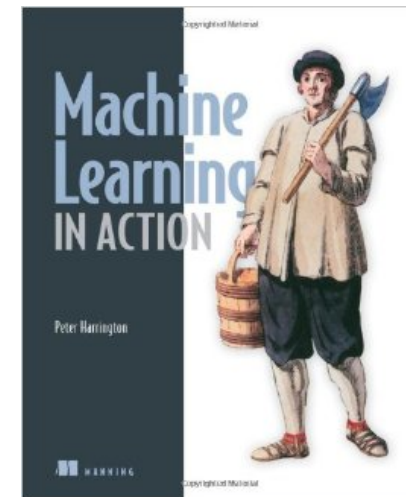
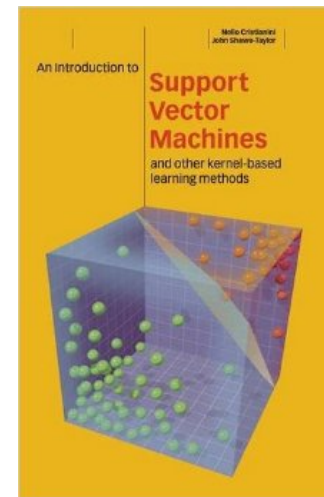
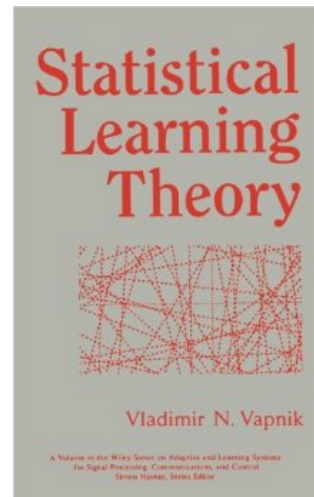
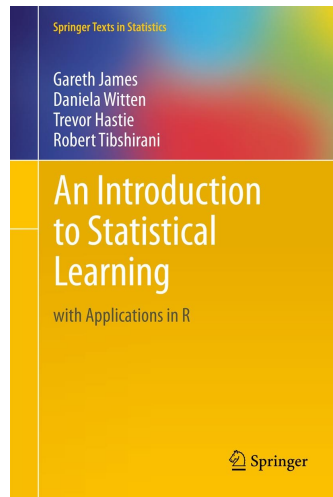
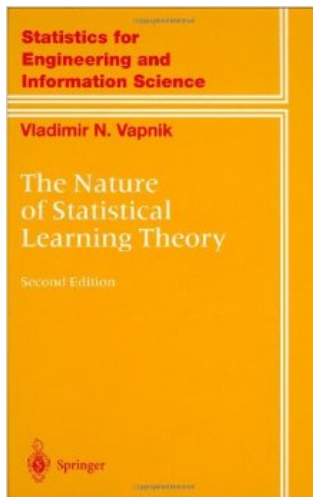
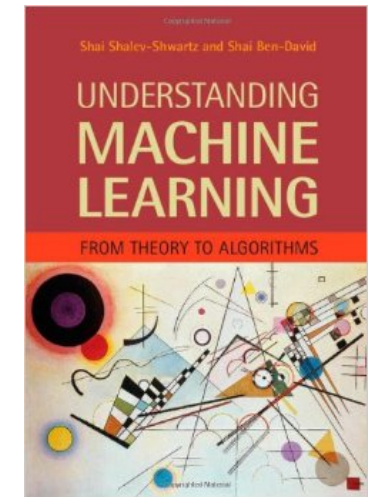
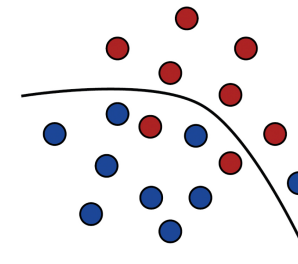
## Comparing Various Models

Econometric models also perform well on a validation sample (and not only on the training sample).

# Some additional references (on Statistical Learning)



Foundations of Machine Learning



## Take-Away Conclusion

“People rarely succeed unless they have fun in what they are doing ” D. Carnegie

- on very small datasets, it is possible to use **Bayesian technique** to derive robust predictions,
- on extremely large datasets, it is possible to use ideas developed in **machine learning**, on regression models (e.g. bootstrapping and aggregating)
- all those techniques require **computational skills**

“the numbers have no way of speaking for themselves. We speak for them. ... Before we demand more of our data, we need to demand more of ourselves ” N. Silver, in **Silver (2012)**.

