Machine Learning & Data Science for Actuaries, with R

Arthur Charpentier (Université de Rennes 1 & UQàM)

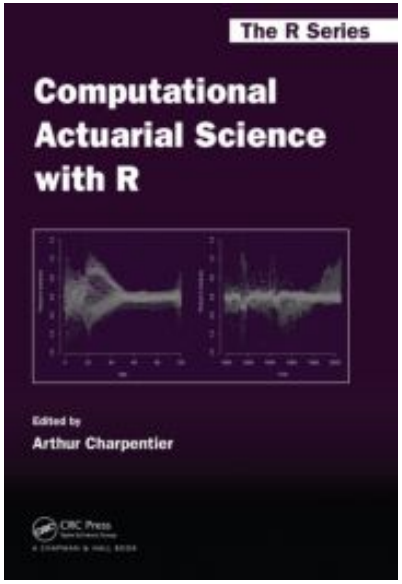
Universitat de Barcelona, April 2016.

http://freakonometrics.hypotheses.org

Machine Learning & Data Science for Actuaries, with R

Arthur Charpentier (Université de Rennes 1 & UQàM)

Professor, Economics Department, Univ. Rennes 1
In charge of Data Science for Actuaries program, IA
Research Chair *actinfo* (Institut Louis Bachelier)
(previously Actuarial Sciences at UQàM & 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's blog
Editor of Computational Actuarial Science, CRC

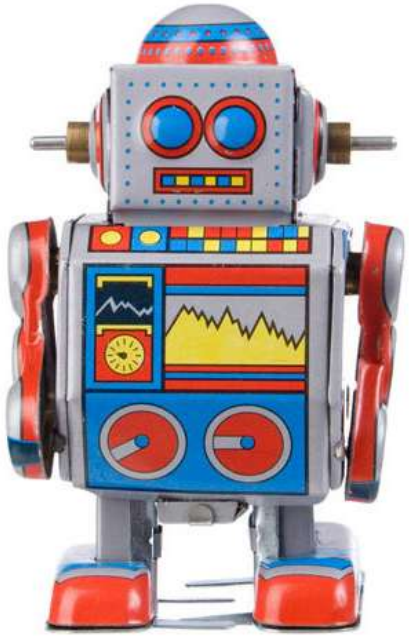


Agenda

0. **Introduction**, see slides
1. **Classification,** $y \in \{0, 1\}$
2. **Regression Models,** $y \in \mathbb{R}$
3. **Model Choice, Feature Selection, etc**
4. **Data Visualisation & Maps**

Part 1.

Classification, $y \in \{0, 1\}$



Classification?

Example: Fraud detection, automatic reading (classifying handwriting symbols), face recognition, accident occurence, death, purchase of optinal insurance cover, etc

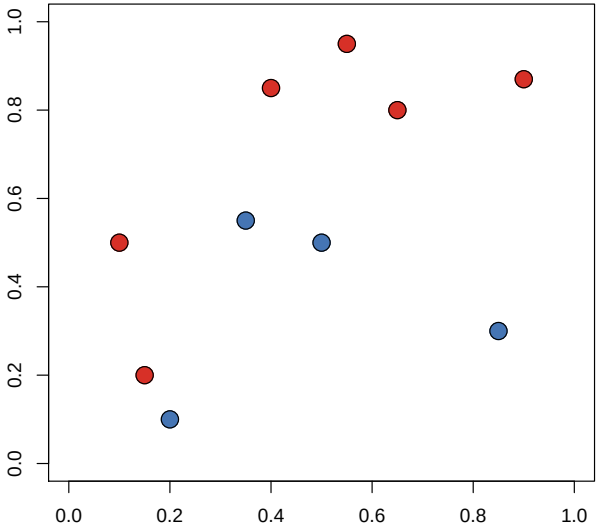
Here $y_i \in \{0, 1\}$ or $y_i \in \{-1, +1\}$ or $y_i \in \{\bullet, \bullet\}$.

We look for a (good) predictive model here.

There will be two steps,

- the score function, $s(\boldsymbol{x}) = \mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}) \in [0, 1]$

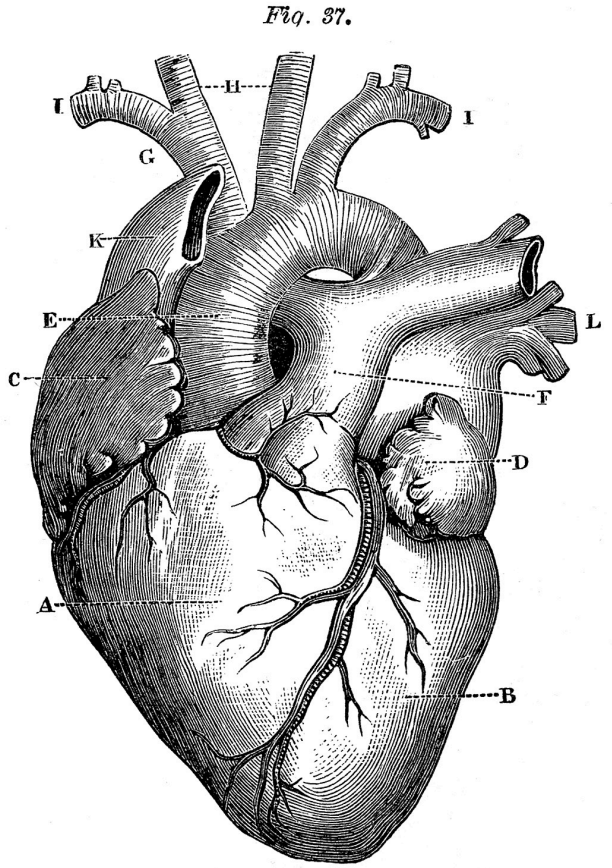
- the classification function $s(\boldsymbol{x}) \to \widehat{Y} \in \{0, 1\}$.



Modeling a 0/1 random variable

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 (`PRONO`)

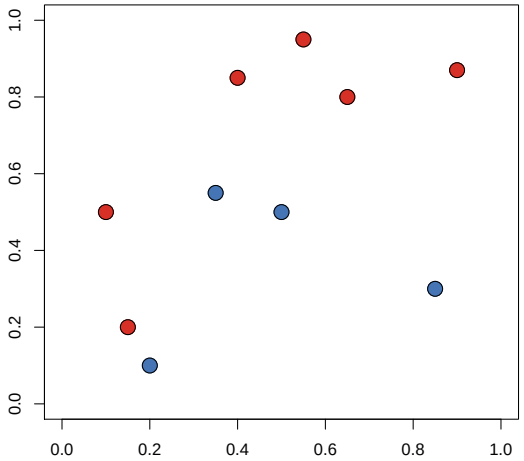



```
> myocarde=read.table("http://freakonometrics.free.fr/myocarde.csv",
    head=TRUE,sep=";")
```

Logistic Regression

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

$$\text{logit}(\pi_i) = \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta}, \text{ where } \text{logit}(\pi_i) = \log\left(\frac{\pi_i}{1-\pi_i}\right),$$



or

$$\pi_i = \text{logit}^{-1}(\boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta}) = \frac{\exp[\boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta}]}{1+\exp[\boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta}]}.$$

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

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

Logistic Regression, Output (with R)

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

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)

Number of Fisher Scoring iterations: 7
```

Logistic Regression, Output (with R)

```
> library(VGAM)
> mlogistic <- vglm(PRONO~. , data=myocarde, family=multinomial)
> summary(mlogistic)

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

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

Logistic (Multinomial) Regression

In the Bernoulli case, $y \in \{0, 1\}$,

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

In the multinomial case, $y \in \{A, B, C\}$

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

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

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

Logistic Regression, Numerical Issues

The algorithm to compute $\widehat{\boldsymbol{\beta}}$ 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, and $H(\boldsymbol{\beta})$ 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

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

Logistic Regression, Numerical Issues

and the Hessian

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

The gradient descent algorithm is then

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

From maximum likelihood properties, if $\widehat{\boldsymbol{\beta}} = \lim_{k \to \infty} \boldsymbol{\beta}_k$,

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

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

Logistic Regression, Numerical Issues

```
> X=cbind(1,as.matrix(myocarde[,1:7]))
> Y=myocarde$PRONO=="Survival"
> beta=as.matrix(lm(Y~0+X)$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
> -solve(Hessian)
> sqrt(-diag(solve(Hessian)))
```

Predicted Probability

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

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

```
> predict(logistic,type="response")[1:5]
        1         2         3         4         5
0.6013894 0.1693769 0.3289560 0.8817594 0.1424219
> predict(mlogistic,type="response")[1:5,]
      Death   Survival
1 0.3986106 0.6013894
2 0.8306231 0.1693769
3 0.6710440 0.3289560
4 0.1182406 0.8817594
5 0.8575781 0.1424219
```

Predicted Probability

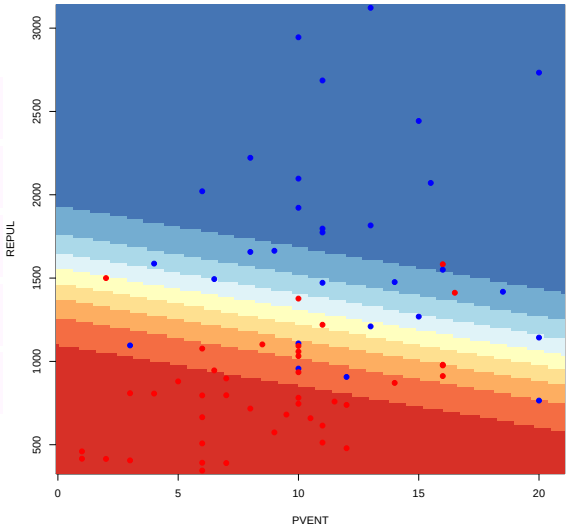
$$\widehat{m}(\boldsymbol{x}) = \frac{\exp[\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}]}{1+\exp[\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}]} = \frac{\exp[\widehat{\beta}_0+\widehat{\beta}_1 x_1+\cdots+\widehat{\beta}_k x_k]}{1+\exp[\widehat{\beta}_0+\widehat{\beta}_1 x_1+\cdots+\widehat{\beta}_k x_k]}$$

use

```
> predict(fit_glm, newdata = data, type="response")
```

e.g.

```
> GLM <- glm(PRONO ~ PVENT + REPUL, data =
    myocarde, family = binomial)
> pred_GLM = function(p,r){
+ return(predict(GLM, newdata =
+ data.frame(PVENT=p,REPUL=r), type="response")}
```

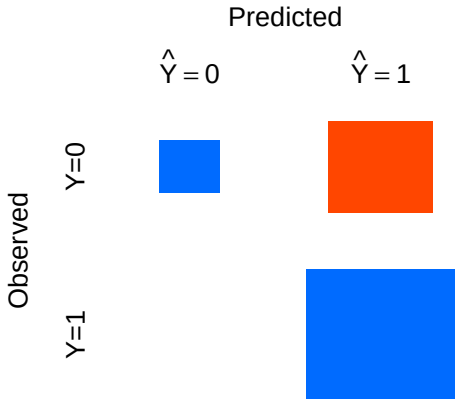


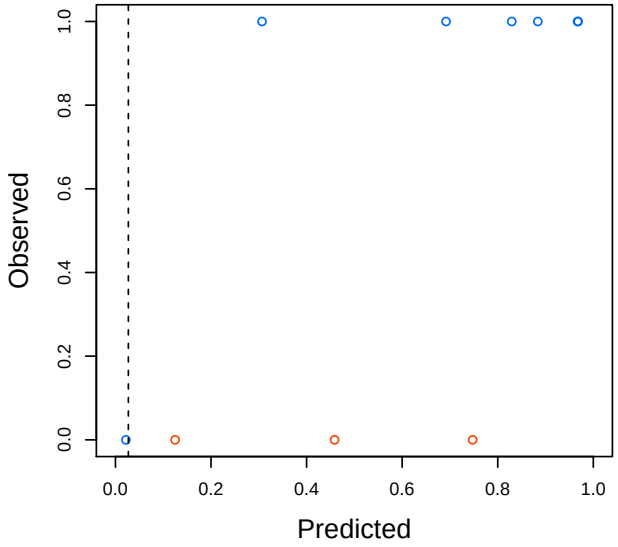
Predictive Classifier

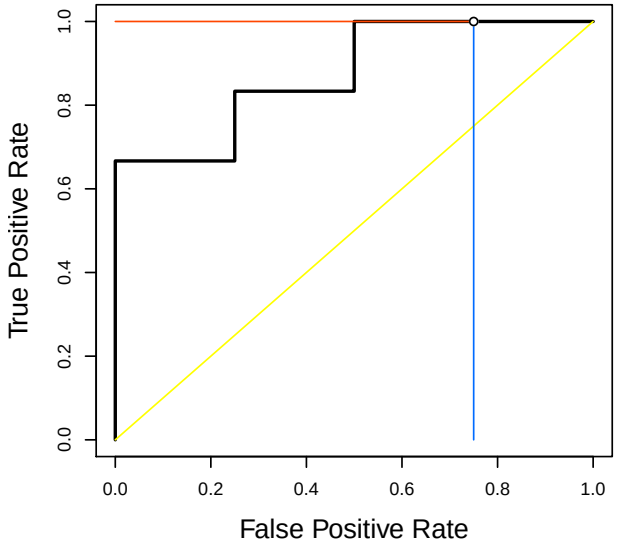
To go from a score to a class:

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

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





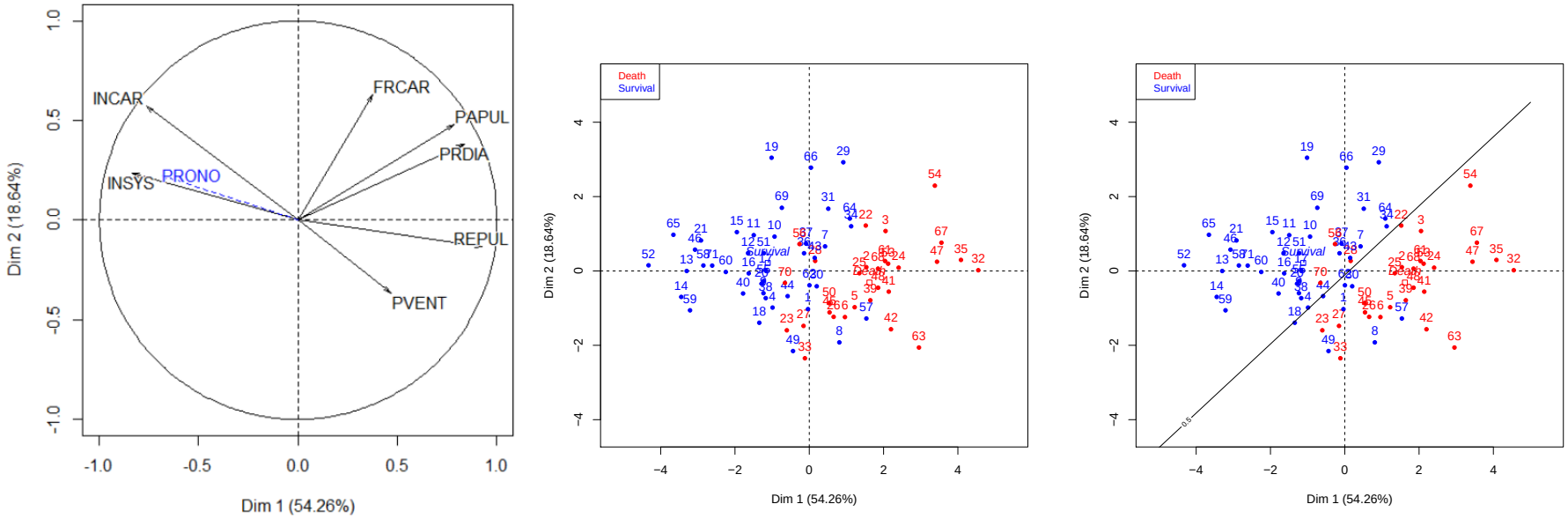


Predictive Classifier

With a threshold (e.g. $s = 50\%$) and the predicted probabilities, one can get a classifier and the confusion matrix

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

predictions Death Survival
   Death       25        3
   Survival     4       39
```

Visualization of a Classifier in Higher Dimension...



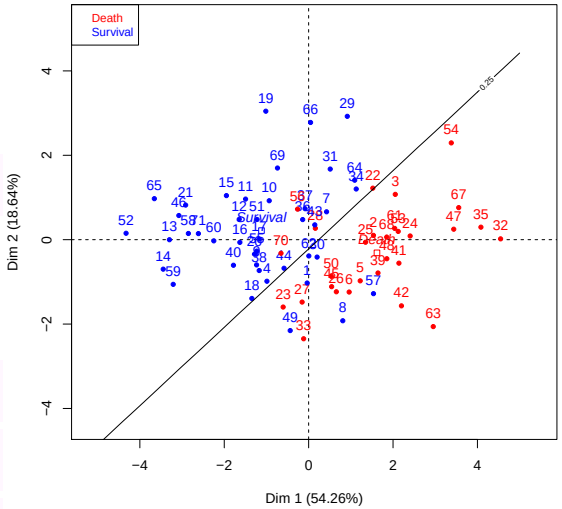
Point $\boldsymbol{z} = (z_1, z_2, 0, \cdots, 0) \longrightarrow \boldsymbol{x} = (x_1, x_2, \cdots, x_k)$.

... but be carefull about interpretation !

```
> prediction=predict(logistic,type="response")
```

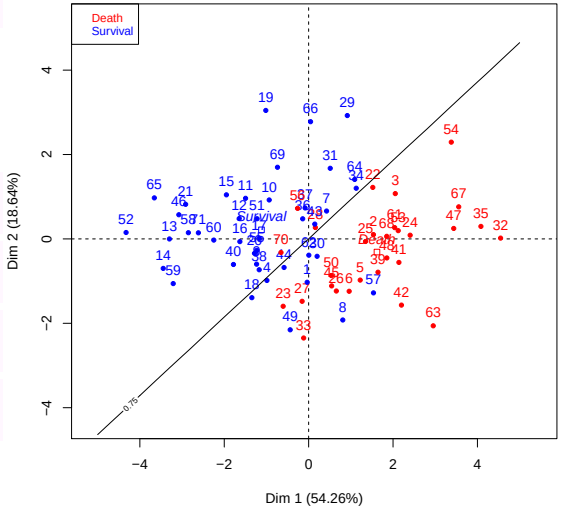
Use a 25% probability threshold

```
>  table(prediction>.25, myocarde$PRONO)
       Death Survival
FALSE     19        2
TRUE      10       40
```



or a 75% probability threshold

```
>  table(prediction>.75, myocarde$PRONO)
       Death Survival
FALSE     27        9
TRUE       2       33
```



Why a Logistic and not a Probit Regression?

Bliss (1934) 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

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

In the logistic regression, we model the odds ratio,

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

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

which is the c.d.f. of the logistic variable, see Verhulst (1845)

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

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*.

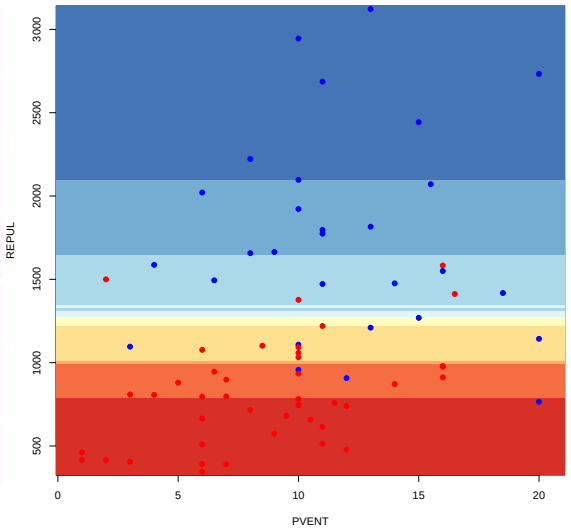
k-Nearest Neighbors (a.k.a. k-NN)

In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. (Source: wikipedia).

$$\mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}] \sim \frac{1}{k} \sum_{\|\boldsymbol{x_i}-\boldsymbol{x}\| \text{ small}} y_i$$

For k-Nearest Neighbors, the class is usually the majority vote of the k closest neighbors of $\boldsymbol{x}$.

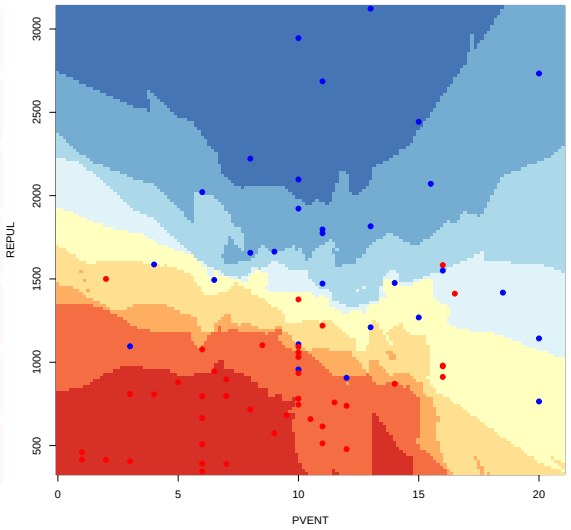
```
> library(caret)
> KNN <- knn3(PRONO ~ PVENT + REPUL, data =
    myocarde, k = 15)
>
> pred_KNN = function(p,r){
+ return(predict(KNN, newdata =
+ data.frame(PVENT=p,REPUL=r), type="prob")[,2]}
```



k-Nearest Neighbors

Distance $\|\cdot\|$ should not be sensitive to units: normalize by standard deviation

```
> sP <- sd(myocarde$PVENT); sR <- sd(myocarde$
    REPUL)
> KNN <- knn3(PRONO ~ I(PVENT/sP) + I(REPUL/sR),
    data = myocarde, k = 15)
> pred_KNN = function(p,r){
+ return(predict(KNN, newdata =
+ data.frame(PVENT=p, REPUL=r), type="prob")[,2]}
```

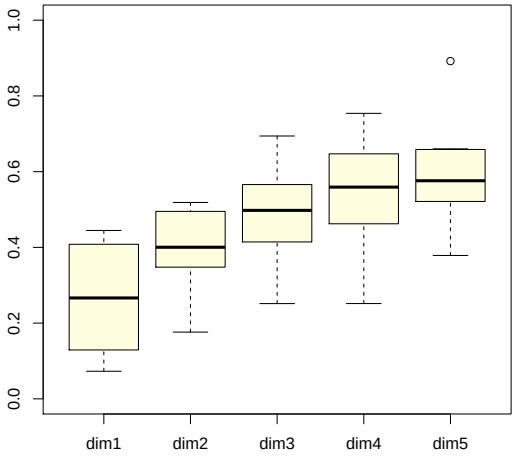


k-Nearest Neighbors and Curse of Dimensionality

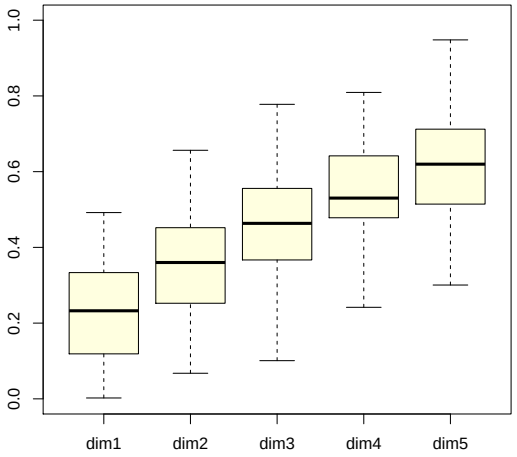
The higher the dimension, the larger the distance to the closest neigbbor

$$\min_{i\in\{1,\cdots,n\}}\{\|\boldsymbol{a},\boldsymbol{x}_i\|\},\boldsymbol{x}_i\in\mathbb{R}^d.$$

e.g. $\boldsymbol{x}$'s drawn from $\mathcal{U}([0,1])$ and $\boldsymbol{a}=\boldsymbol{0}$,



$n=10$



$n=100$

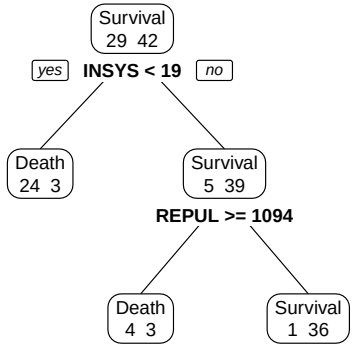
Classification (and Regression) Trees, CART

one of the predictive modelling approaches used in statistics, data mining and machine learning [...] In tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. (Source: wikipedia).

```
> library(rpart)
> cart<-rpart(PRONO~., data = myocarde)
> library(rpart.plot)
> library(rattle)
> prp(cart, type=2, extra=1)
```

or

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



Classification (and Regression) Trees, CART

The impurity is a function φ of the probability to have 1 at node N, i.e. $\mathbb{P}[Y = 1|$ node $N]$, and

$$\mathcal{I}(N) = \varphi(\mathbb{P}[Y = 1|\text{ node } N])$$

φ is nonnegative ($\varphi \geq 0$), symmetric ($\varphi(p) = \varphi(1-p)$), with a minimum in 0 and 1 ($\varphi(0) = \varphi(1) < \varphi(p)$), e.g.

- Bayes error: $\varphi(p) = \min\{p, 1-p\}$
- cross-entropy: $\varphi(p) = -p\log(p) - (1-p)\log(1-p)$
- Gini index: $\varphi(p) = p(1-p)$

Those functions are concave, minimum at $p = 0$ and 1, maximum at $p = 1/2$.

Classification (and Regression) Trees, CART

To split N into two $\{N_L, N_R\}$, consider

$$\mathcal{I}(N_L, N_R) = \sum_{x\in\{L,R\}} \frac{n_x}{n}\mathcal{I}(N_x)$$

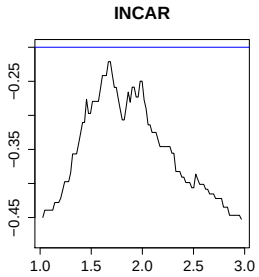
e.g. Gini index (used originally in CART, see Breiman *et al.* (1984))

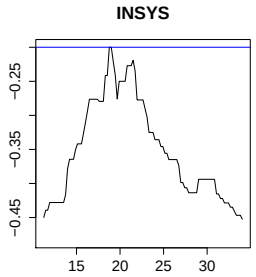
$$\text{gini}(N_L, N_R) = -\sum_{x\in\{L,R\}} \frac{n_x}{n} \sum_{y\in\{0,1\}} \frac{n_{x,y}}{n_x}\left(1 - \frac{n_{x,y}}{n_x}\right)$$

and the cross-entropy (used in C4.5 and C5.0)

$$\text{entropy}(N_L, N_R) = -\sum_{x\in\{L,R\}} \frac{n_x}{n} \sum_{y\in\{0,1\}} \frac{n_{x,y}}{n_x}\log\left(\frac{n_{x,y}}{n_x}\right)$$

Classification (and Regression) Trees, CART

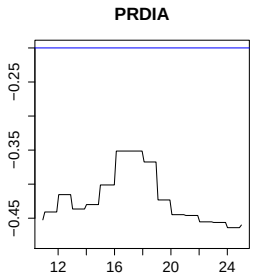


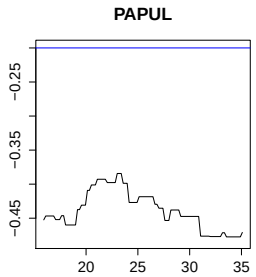


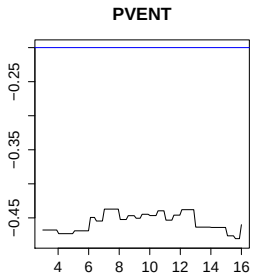
$$N_L\colon \{x_{i,j} \leq s\} \quad N_R\colon \{x_{i,j} > s\}$$

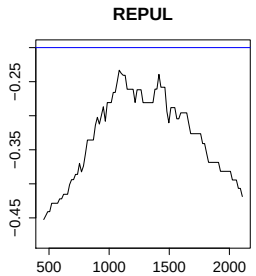
$$\text{solve} \max_{j\in\{1,\cdots,k\},s} \{\mathcal{I}(N_L, N_R)\}$$

$\longleftarrow$ first split

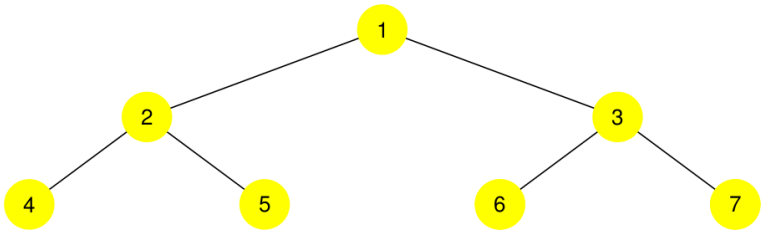


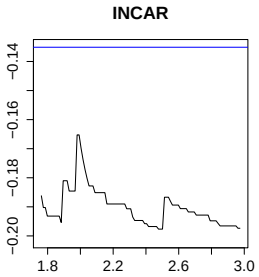


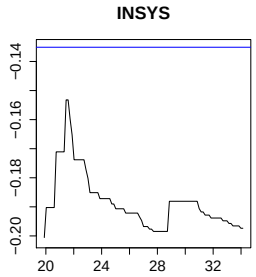


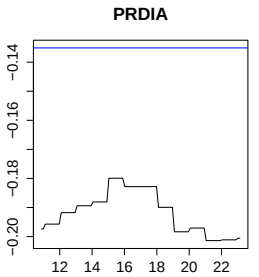


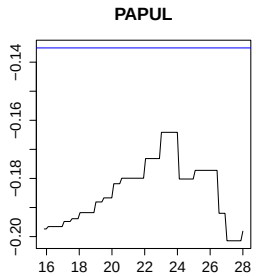
second split $\longrightarrow$

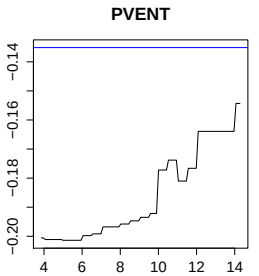


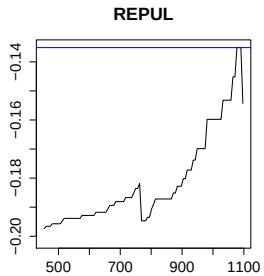












Pruning Trees

One can grow a big tree, until leaves have a (preset) small number of observations, and then possibly go back and prune branches (or leaves) that do not improve gains on good classification sufficiently.

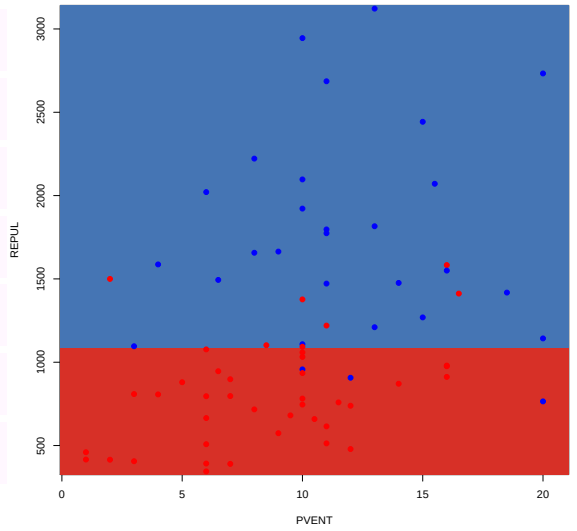
Or we can decide, at each node, whether we split, or not.

Pruning Trees

In trees, overfitting increases with the number of steps, and leaves. Drop in impurity at node N is defined as

$$\Delta\mathcal{I}(N_L, N_R) = \mathcal{I}(N) - \mathcal{I}(N_L, N_R) = \mathcal{I}(N) - \left(\frac{n_L}{n}\mathcal{I}(N_L) - \frac{n_R}{n}\mathcal{I}(N_R)\right)$$

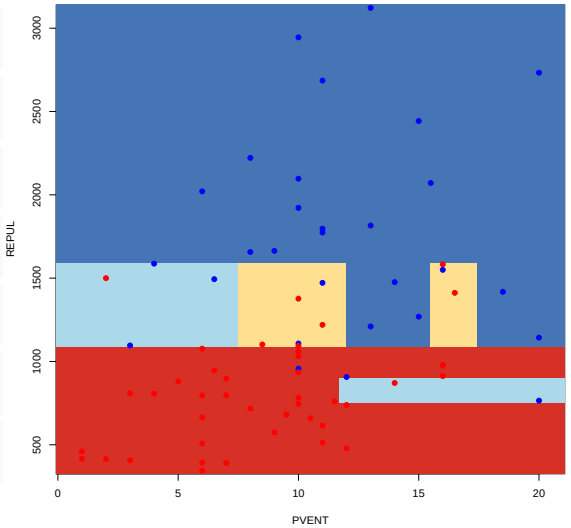
```
> library(rpart)
> CART <- rpart(PRONO ~ PVENT + REPUL, data =
    myocarde, minsplit = 20)
>
> pred_CART = function(p,r){
+ return(predict(CART, newdata =
+ data.frame(PVENT=p,REPUL=r)[,"Survival"])}
```



$\longrightarrow$ we cut if $\Delta\mathcal{I}(N_L, N_R)/\mathcal{I}(N)$ (relative gain) exceeds `cp` (complexity parameter, default 1%).

Pruning Trees

```
> library(rpart)
> CART <- rpart(PRONO ~ PVENT + REPUL, data =
    myocarde, minsplit = 5)
>
> pred_CART = function(p,r){
+ return(predict(CART, newdata =
+ data.frame(PVENT=p,REPUL=r)[,"Survival"])}
```



See also

```
> library(mvpart)
> ?prune
```

Define the missclassification rate of a tree $R(\text{tree})$

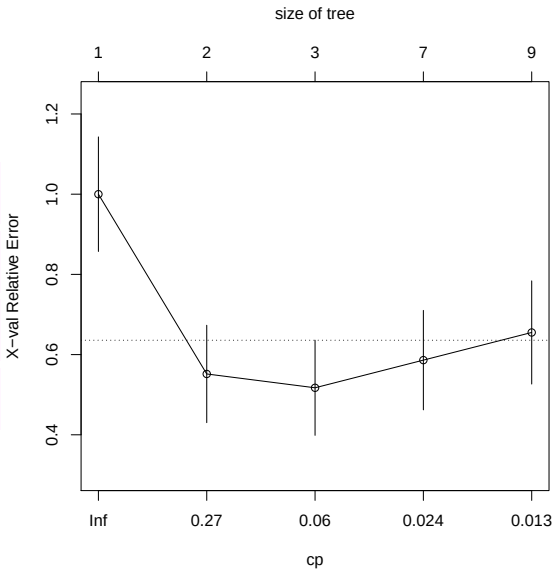
Pruning Trees

Given a cost-complexity parameter `cp` (see tunning parameter in Ridge-Lasso) define a penalized $R(\cdot)$

$$R_{\text{cp}}(\text{tree}) = \underbrace{R(\text{tree})}_{\text{loss}} + \underbrace{\text{cp}\|\text{tree}\|}_{\text{complexity}}$$

If `cp` is small the optimal tree is large, if `cp` is large the optimal tree has no leaf, see Breiman *et al.* (1984).

```
> cart <- rpart(PRONO ~ .,data = myocarde,minsplit
    =3)
> plotcp(cart)
> prune(cart, cp=0.06)
```



Bagging

Bootstrapped Aggregation (Bagging) , is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification (Source: wikipedia).

It is an ensemble method that creates multiple models of the same type from different sub-samples of the same dataset [boostrap]. The predictions from each separate model are combined together to provide a superior result [aggregation].

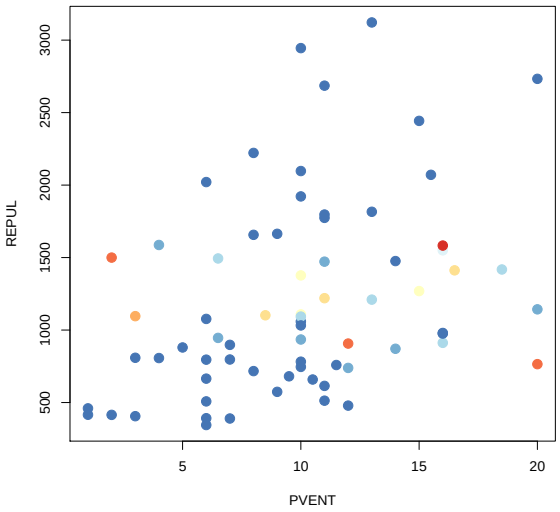
$\rightarrow$ can be used on any kind of model, but interesting for trees, see Breiman (1996)

Boostrap can be used to define the concept of margin,

$$\text{margin}_i = \frac{1}{B}\sum_{b=1}^{B} \mathbf{1}(\widehat{y}_i = y_i) - \frac{1}{B}\sum_{b=1}^{B} \mathbf{1}(\widehat{y}_i \neq y_i)$$

Remark Probability that ith raw is not selection $(1-n^{-1})^n \rightarrow e^{-1} \sim 36.8\%$, cf training / validation samples (2/3-1/3)

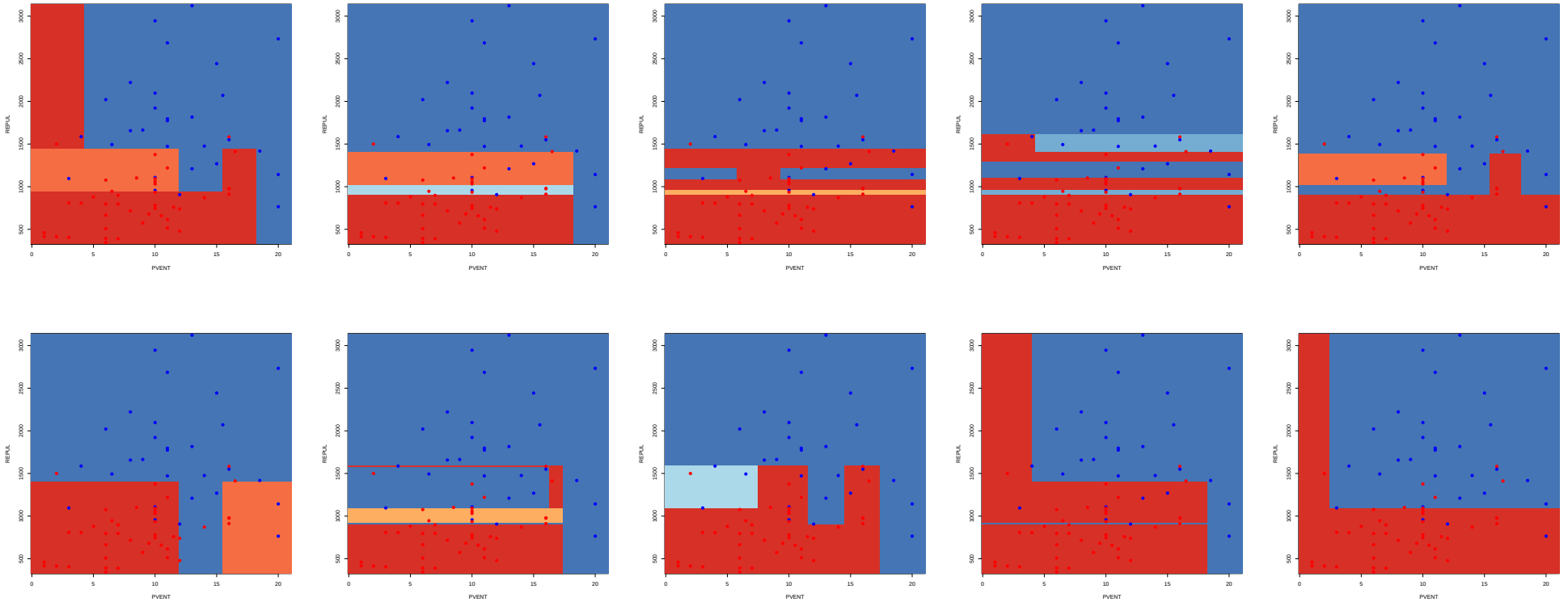
Bagging Trees




```
> margin <- matrix(NA,1e4,n)
> for(b in 1:1e4){
+ idx = sample(1:n,size=n,replace=TRUE)
> cart <- rpart(PRONO~PVENT+REPUL,
+ data=myocarde[idx,], minsplit = 5)
> margin[j,] <- (predict(cart2,newdata=
    myocarde,type="prob")[,"Survival"]>.5)!=
    (myocarde$PRONO=="Survival")
+ }
> apply(margin, 2, mean)
```

Bagging Trees

Interesting because of instability in CARTs (in terms of tree structure, not necessarily prediction)



Bagging and Variance, Bagging and Bias

Assume that $y = m(\boldsymbol{x}) + \varepsilon$. The mean squared error over repeated random samples can be decomposed in three parts Hastie *et al.* (2001)

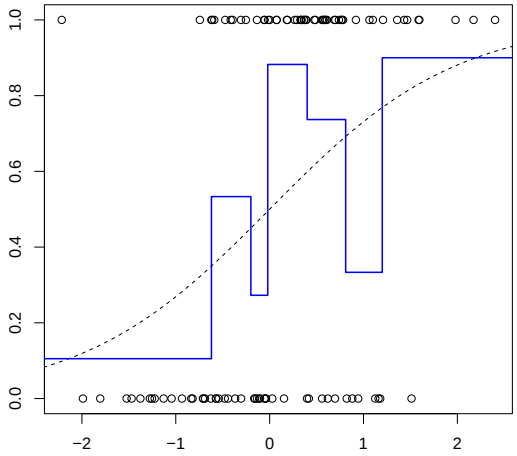
$$\mathbb{E}[(Y - \widehat{m}(\boldsymbol{x}))^2] = \underbrace{\sigma^2}_{1} + \underbrace{\left[\mathbb{E}[\widehat{m}(\boldsymbol{x})] - m(\boldsymbol{x})\right]^2}_{2} + \underbrace{\mathbb{E}\Big(\left[\widehat{m}(\boldsymbol{x}) - \mathbb{E}[(\widehat{m}(\boldsymbol{x})]\right]^2\Big)}_{3}$$

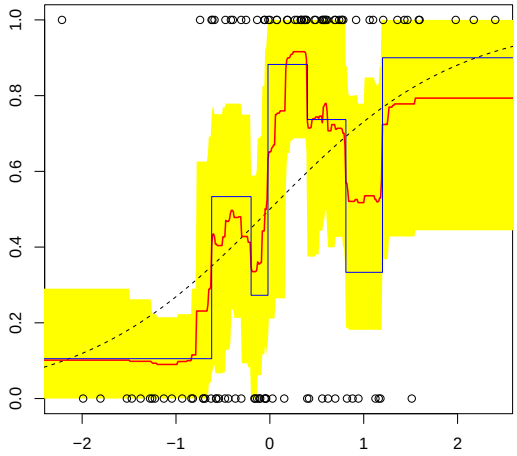
1 reflects the variance of Y around $m(\boldsymbol{x})$

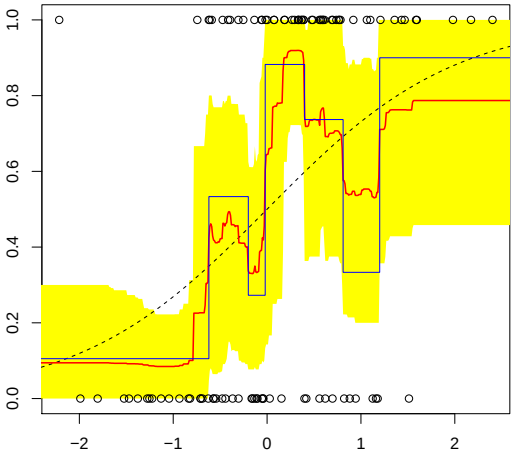
2 is the squared bias of $\widehat{m}(\boldsymbol{x})$

3 is the variance of $\widehat{m}(\boldsymbol{x})$

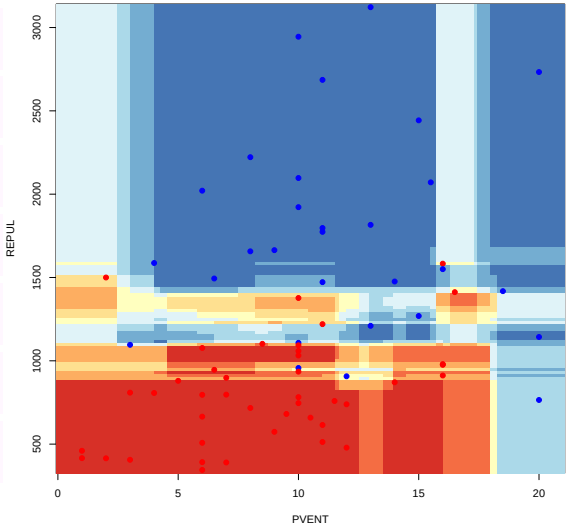
$\longrightarrow$ bias-variance tradeoff. Boostrap can be used to reduce the bias, and he variance (but be careful of outliers)








```
> library(ipred)
> BAG <- bagging(PRONO ~ PVENT + REPUL, data =
    myocarde)
>
> pred_BAG = function(p,r){
+ return(predict(BAG,newdata=
+ data.frame(PVENT=p,REPUL=r), type="prob")[,2])}
```

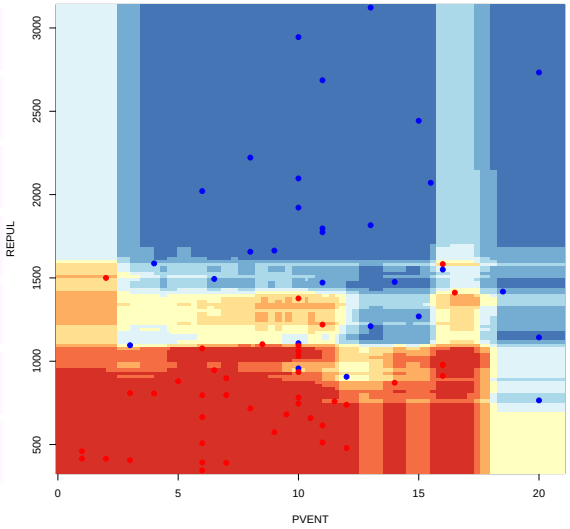


Random Forests

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)) and Amit & Geman (1997))

```
> library(randomForest)
> RF <- randomForest(PRONO ~ PVENT + REPUL, data =
      myocarde)
>
> pred_RF = function(p,r){
+ return(predict(RF,newdata=
+ data.frame(PVENT=p,REPUL=r), type="prob")[,2])}
```



Random Forest

At each node, select $\sqrt{k}$ covariates out of k (randomly).

can deal with small n large k-problems

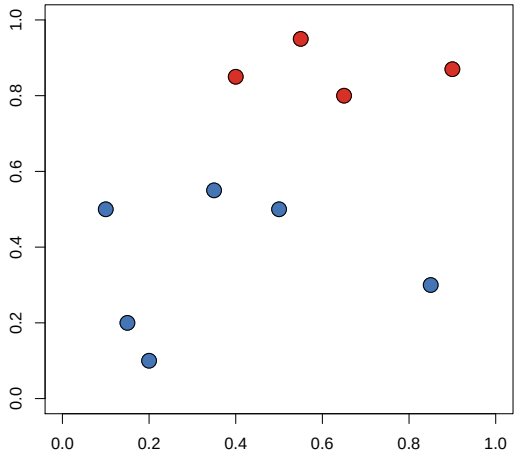
Random Forest are used not only for prediction, but also to assess variable importance (discussed later on).

Support Vector Machine

SVMs were developed in the 90's based on previous work, from Vapnik & Lerner (1963), see Vailant (1984)

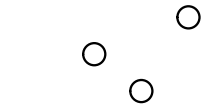
Assume that points are linearly separable, i.e. there is $\boldsymbol{\omega}$ and b such that

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

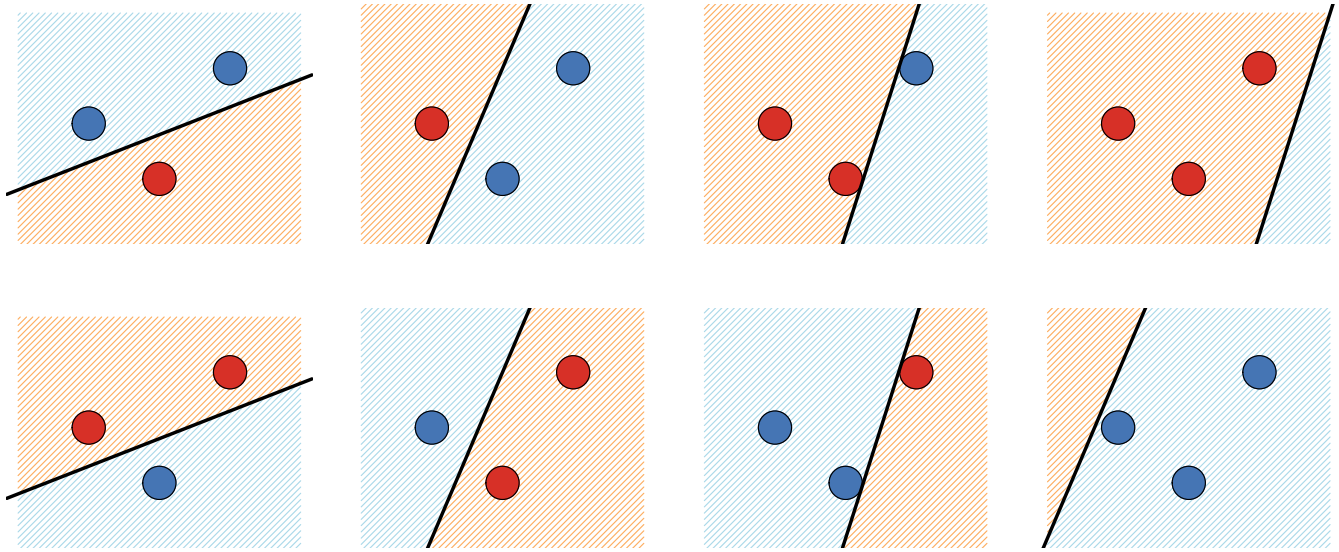


Problem: infinite number of solutions, need a good one, that separate the data, (somehow) far from the data.

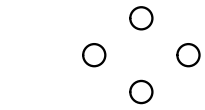
Concept : VC dimension. Let $\mathcal{H} : \{h : \mathbb{R}^d \mapsto \{-1, +1\}\}$. Then $\mathcal{H}$ is said to shatter a set of points $\boldsymbol{X}$ is all dichotomies can be achieved.
E.g. with those three points, all configurations can be achieved



Support Vector Machine



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

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

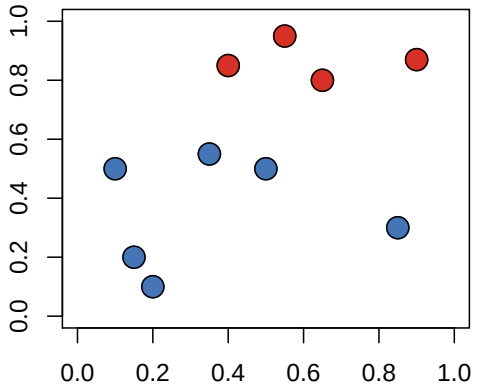
$$\text{argmin}\left\{\min_{i=1,\cdots,n} d(\boldsymbol{x}_i, H_{\boldsymbol{\omega},b})\right\}$$

Support Vector Machine

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 $2\|\boldsymbol{\omega}\|^{-1}$.

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

$$\min\left\{\frac{1}{2}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\omega}\right\} \text{ s.t. } Y_i(\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{x}_i + b) \geq 1, \ \forall i.$$

Support Vector Machine

Problem difficult to solve: many inequality constraints (n)

$\longrightarrow$ 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)

$$\max \left\{ \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i=1} \alpha_i \alpha_j Y_i Y_j \boldsymbol{x}_i^\mathsf{T} \boldsymbol{x}_j \right\} \text{ s.t. } \sum_{i=1} \alpha_i Y_i = 0.$$

which is usually written

$$\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \boldsymbol{\alpha}^\mathsf{T} \boldsymbol{Q} \boldsymbol{\alpha} - \mathbf{1}^\mathsf{T} \boldsymbol{\alpha} \right\} \text{ s.t. } \begin{cases} 0 \leq \alpha_i \ \forall i \\ \boldsymbol{y}^\mathsf{T} \boldsymbol{\alpha} = 0 \end{cases}$$

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

Now, what about the non-separable case?

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

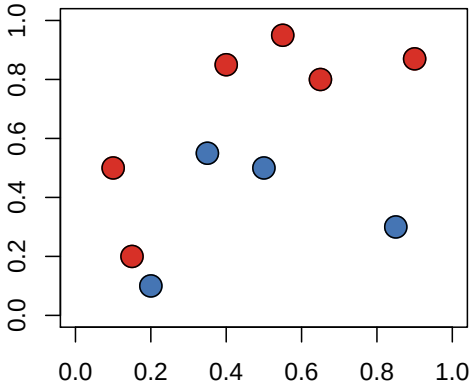
$\longrightarrow$ introduce slack variables,

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

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

The idea is then to solve

$$\min \left\{ \frac{1}{2}\boldsymbol{\omega}^\top \boldsymbol{\omega} + C\mathbf{1}^\top \mathbf{1}_{\boldsymbol{\xi}>1} \right\}, \text{ instead of } \min \left\{ \frac{1}{2}\boldsymbol{\omega}^\top \boldsymbol{\omega} \right\}$$



Support Vector Machines, with a Linear Kernel

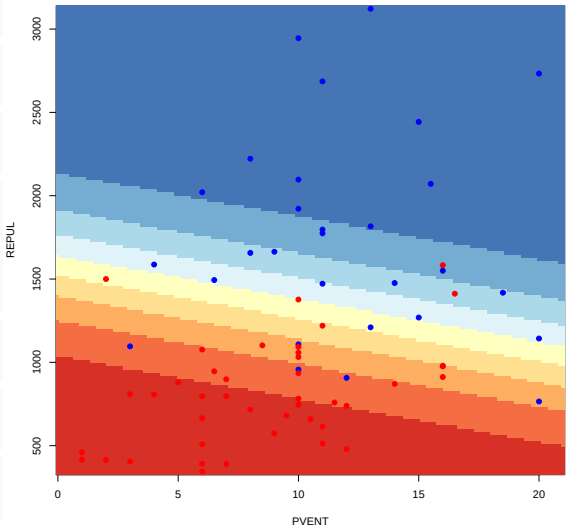
So far,

$$d(\boldsymbol{x}_0, H_{\boldsymbol{\omega},b}) = \min_{\boldsymbol{x}\in H_{\boldsymbol{\omega},b}} \{\|\boldsymbol{x}_0 - \boldsymbol{x}\|_{\ell_2}\}$$

where $\|\cdot\|_{\ell_2}$ is the Euclidean (ℓ_2) norm,

$$\|\boldsymbol{x}_0 - \boldsymbol{x}\|_{\ell_2} = \sqrt{(\boldsymbol{x}_0 - \boldsymbol{x})\cdot(\boldsymbol{x}_0 - \boldsymbol{x})} = \sqrt{\boldsymbol{x}_0\cdot\boldsymbol{x}_0 - 2\boldsymbol{x}_0\cdot\boldsymbol{x} + \boldsymbol{x}\cdot\boldsymbol{x}}$$

```
> library(kernlab)
> SVM2 <- ksvm(PRONO ~ PVENT + REPUL, data =
    myocarde,
+ prob.model = TRUE, kernel = "vanilladot")
> pred_SVM2 = function(p,r){
+ return(predict(SVM2,newdata=
+ data.frame(PVENT=p,REPUL=r), type="probabilities
    ")[,2])}
```



Support Vector Machines, with a Non Linear Kernel

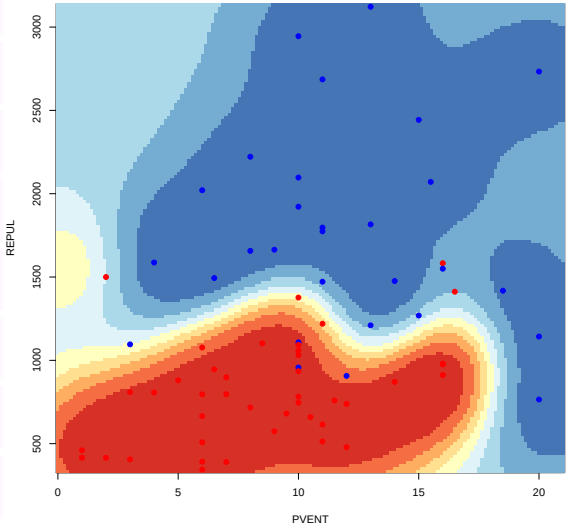
More generally,

$$d(\boldsymbol{x}_0, H_{\boldsymbol{\omega},b}) = \min_{\boldsymbol{x}\in H_{\boldsymbol{\omega},b}} \{\|\boldsymbol{x}_0 - \boldsymbol{x}\|_k\}$$

where $\|\cdot\|_k$ is some kernel-based norm,

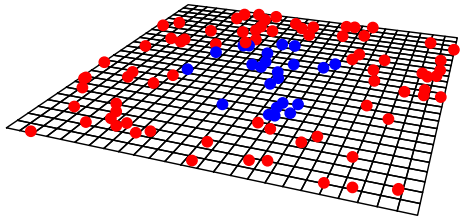
$$\|\boldsymbol{x}_0 - \boldsymbol{x}\|_k = \sqrt{k(\boldsymbol{x}_0,\boldsymbol{x}_0) - 2k(\boldsymbol{x}_0,\boldsymbol{x}) + k(\boldsymbol{x}\cdot\boldsymbol{x})}$$

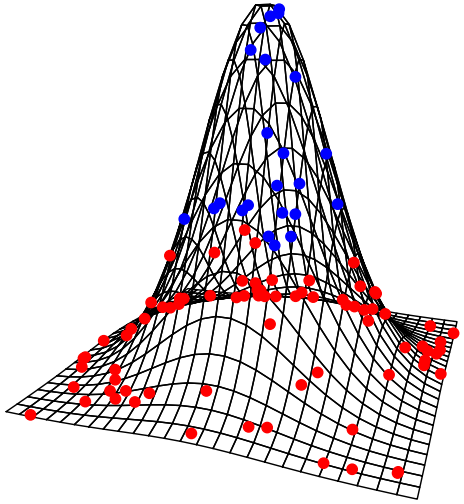
```
1 > library(kernlab)
2 > SVM2 <- ksvm(PRONO ~ PVENT + REPUL, data =
      myocarde,
3 + prob.model = TRUE, kernel = "rbfdot")
4 > pred_SVM2 = function(p,r){
5 + return(predict(SVM2,newdata=
6 + data.frame(PVENT=p,REPUL=r), type="probabilities
      ")[,2])}
```



Heuristics on SVMs

An interpretation is that data aren't linearly seperable in the original space, but might be separare by some kernel transformation,



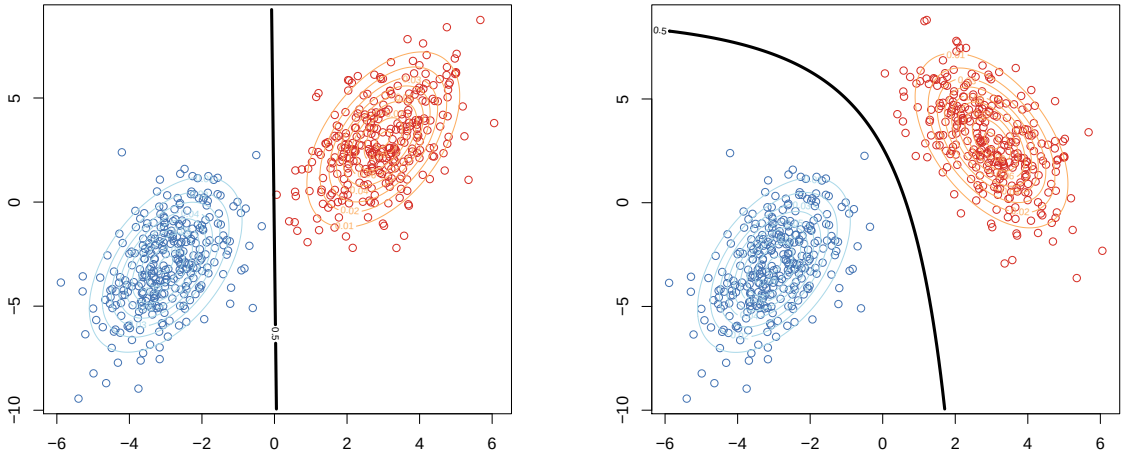


Still Hungry ?

There are still several (machine learning) techniques that can be used for classification

- Fisher's Linear or Quadratic Discrimination (closely related to logistic regression, and PCA), see Fisher (1936))

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



Still Hungry ?

- Perceptron or more generally Neural Networks In machine learning, neural networks are a family of statistical learning models inspired by biological neural networks and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. wikipedia, see Rosenblatt (1957)
- Boosting (see next section)
- Naive Bayes In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. wikipedia, see Russell & Norvig (2003)

See also the (great) package

```
> library(caret)
```

Difference in Differences

In many applications (e.g. marketing), we do need two models to analyze the impact of a treatment. We need two groups, a control and a treatment group.

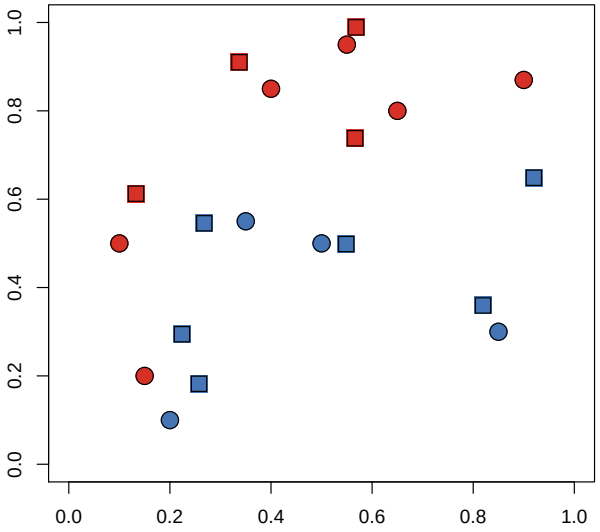
Data : $\{(\boldsymbol{x}_i, y_i)\}$ with $y_i \in \{\bullet, \bullet\}$

$\{(\boldsymbol{x}_j, y_j)\}$ 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%



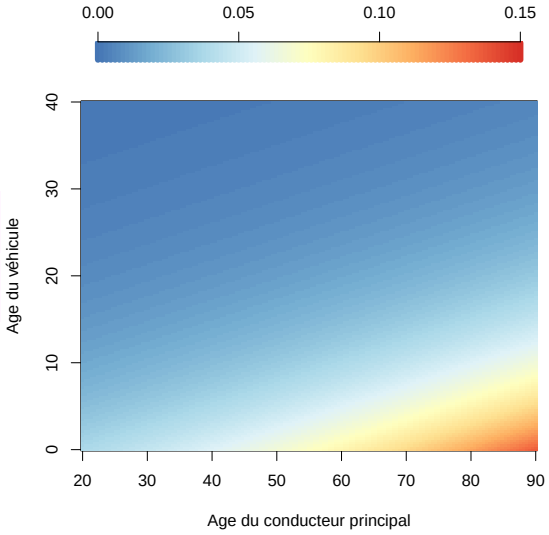
overall uplift effect +23.57%, see Guelman *et al.* (2014) for more details.

Application on Motor Insurance Claims

Consider a (standard) logistic regression, on two covariate (age of driver, and age of camping-car)

$$\pi = \text{logit}^{-1}(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$$

```
> reg_glm=glm(nombre~ageconducteur+agevehicule,
    data=camping,family=binomial)
```

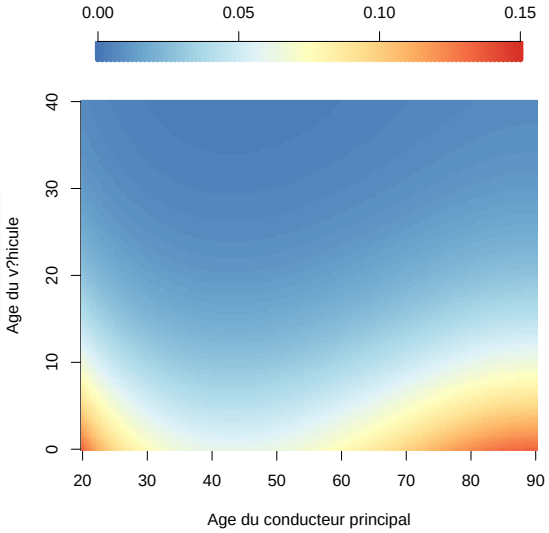


Application on Motor Insurance Claims

Consider a (standard) logistic regression, on two covariate (age of driver, and age of camping-car), smoothed with splines

$$\pi = \text{logit}^{-1}(\beta_0 + s_1(x_1) + s_2(x_2))$$

```
> reg_add=glm(nombre~bs(ageconducteur)+bs(
    agevehicule),data=camping,family=binomial)
```

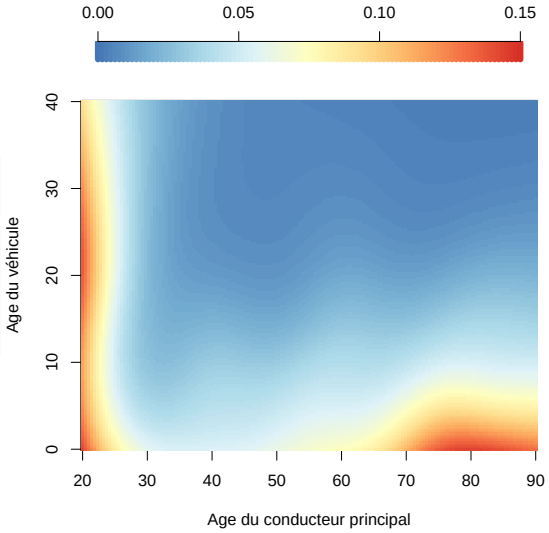


Application on Motor Insurance Claims

Consider a (standard) logistic regression, on two covariate (age of driver, and age of camping-car), smoothed with bivariate spline

$$\pi = \text{logit}^{-1}(\beta_0 + s(x_1,x_2))$$

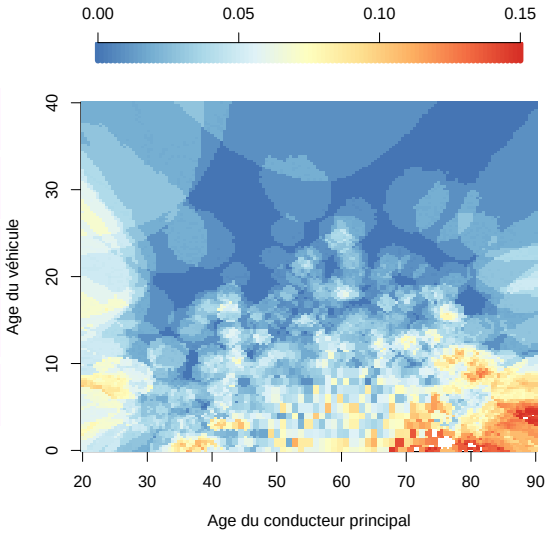
```
> library(mgcv)
> reg_gam=gam(nombre~s(ageconducteur,agevehicule),
    data=camping,family=binomial)
```



Application on Motor Insurance Claims

One can also use k-Nearest Neighbours (k-NN)

```
> library(caret)
> sc=sd(camping$ageconducteur)
> sv=sd(camping$agevehicule)
> knn=knn3((nombre==1)~I(ageconducteur/sc)+I(
    agevehicule/sv),data=camping,k=100)
```

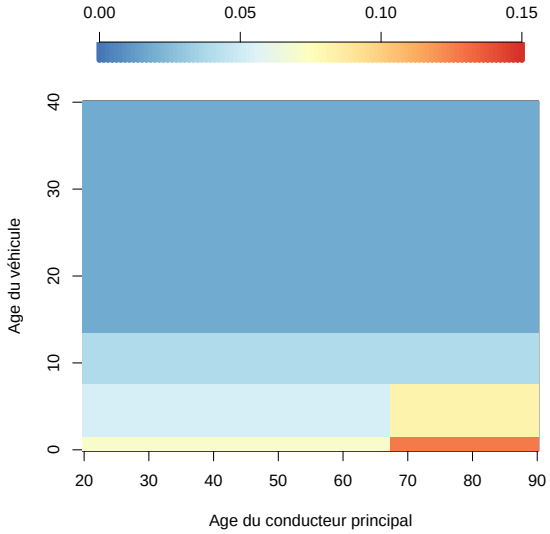


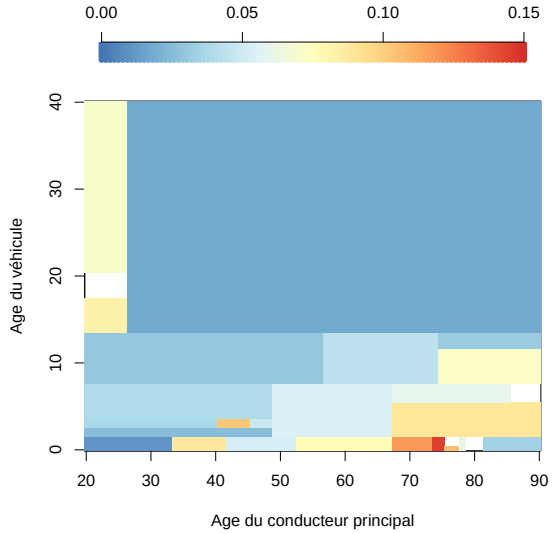
(be carefull about scaling problems)

Application on Motor Insurance Claims

We can also use a tree

```
> tree=rpart((nombre==1)~ageconducteur+agevehicule,
  data=camping,cp=7e-4)
```

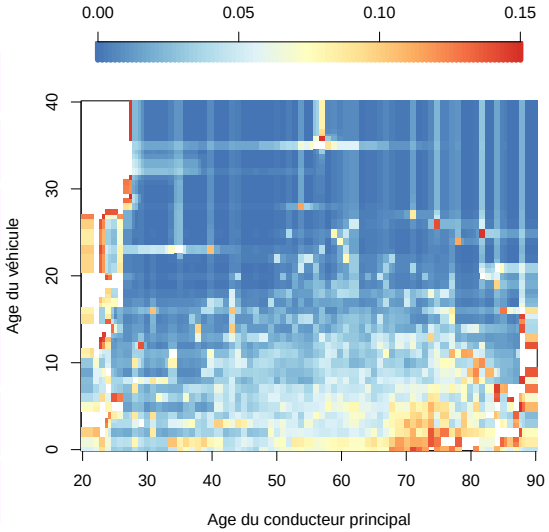




Application on Motor Insurance Claims

or bagging techniques (rather close to random forests)

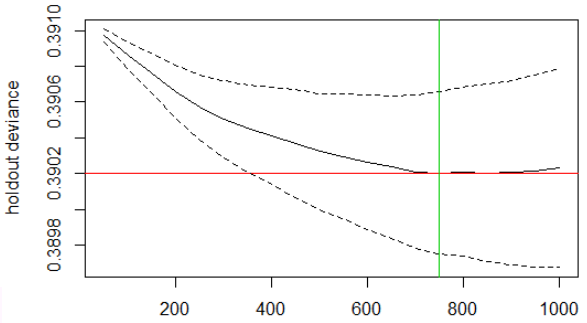
```
> library(ipred)
> bag=bagging((nombre==1)~ageconducteur+
    agevehicule,data=camping)

> library(randomForest)
> rf=randomForest((nombre==1)~ageconducteur+
    agevehicule,data=camping)
```

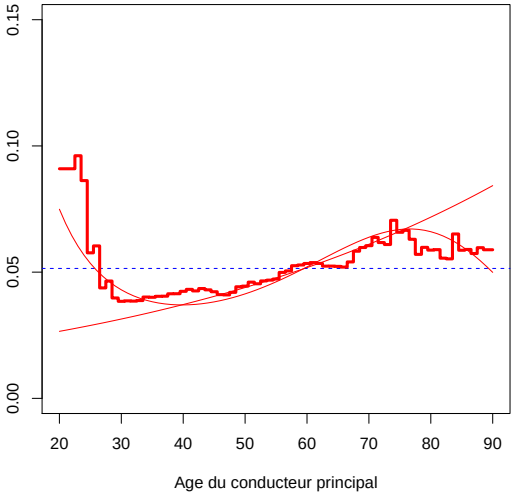


Application on Motor Insurance Claims

Boosting algorithms can also be considered (see next time)

```
> library(dismo)
> library(gbm)
> fit <- gbm.step(data=camping, gbm.x=1, gbm.y=13,
      family="bernoulli", tree.complexity=5,
     learning.rate=0.001, bag.fraction=0.5)
> predict(fit, type="response", n.trees=700)
```

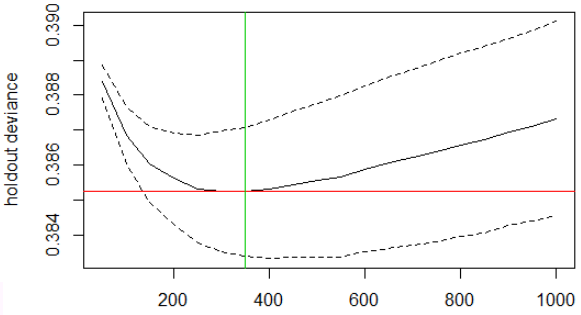


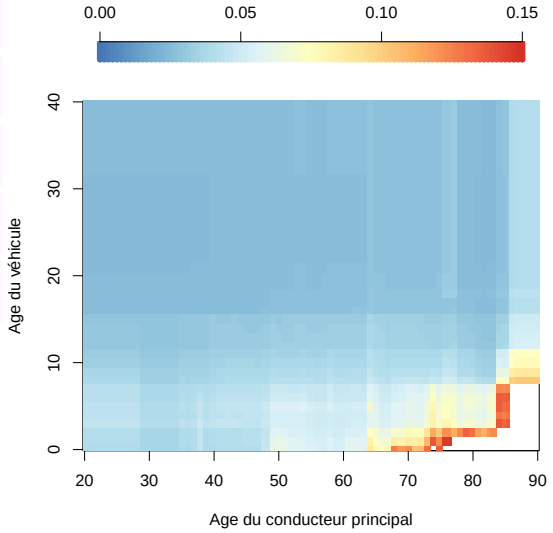


Application on Motor Insurance Claims

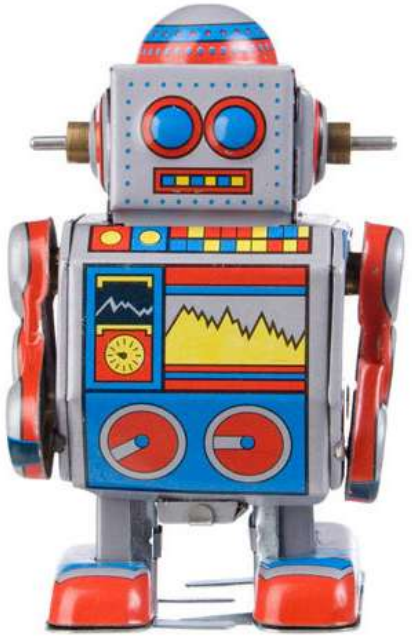
Boosting algorithms can also be considered (see next time)

```
> library(dismo)
> library(gbm)
> fit <- gbm.step(data=camping, gbm.x=c(1,7), gbm.
    y=13, family="bernoulli", tree.complexity=5,
    learning.rate=0.01, bag.fraction=0.5)
> predict(fit, type="response", n.trees=400)
```





Part 2.
Regression



Regression?

In statistics, regression analysis is a statistical process for estimating the relationships among variables [...] In a narrower sense, regression may refer specifically to the estimation of continuous response variables, as opposed to the discrete response variables used in classification. (Source: wikipedia).

Here regression is opposed to classification (as in the CART algorithm). y is either a continuous variable $y \in \mathbb{R}$ or a counting variable $y \in \mathbb{N}$.

Regression? Parametrics, nonparametrics and machine learning

In many cases in econometric and actuarial literature we *simply* want a good fit for the conditional expectation, $\mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}]$.

Regression analysis estimates the conditional expectation of the dependent variable given the independent variables (Source: wikipedia).

Example: A popular nonparametric technique, kernel based regression,

$$\widehat{m}(\boldsymbol{x}) = \frac{\sum_i Y_i \cdot K_h(\boldsymbol{X}_i - \boldsymbol{x})}{\sum_i K_h(\boldsymbol{X}_i - \boldsymbol{x})}$$

In econometric litterature, interest on asymptotic normality properties and plug-in techniques.

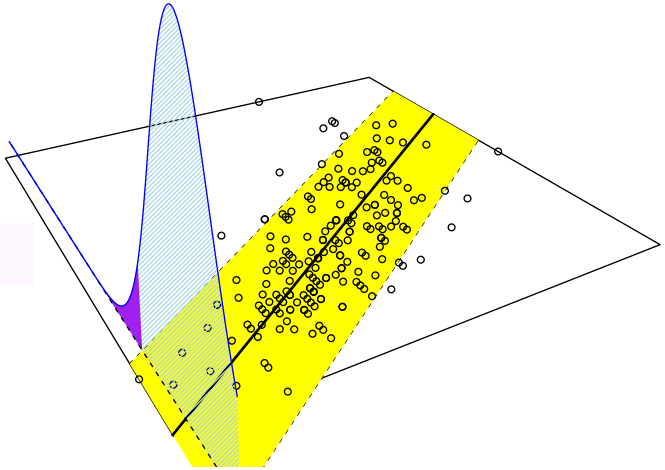
In machine learning, interest on out-of sample cross-validation algorithms.

Linear, Non-Linear and Generalized Linear

Linear Model:

- $(Y|\boldsymbol{X}=\boldsymbol{x}) \sim \mathcal{N}(\theta_{\boldsymbol{x}}, \sigma^2)$
- $\mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}] = \theta_{\boldsymbol{x}} = \boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta}$

```
1 > fit <- lm(y ~ x, data = df)
```

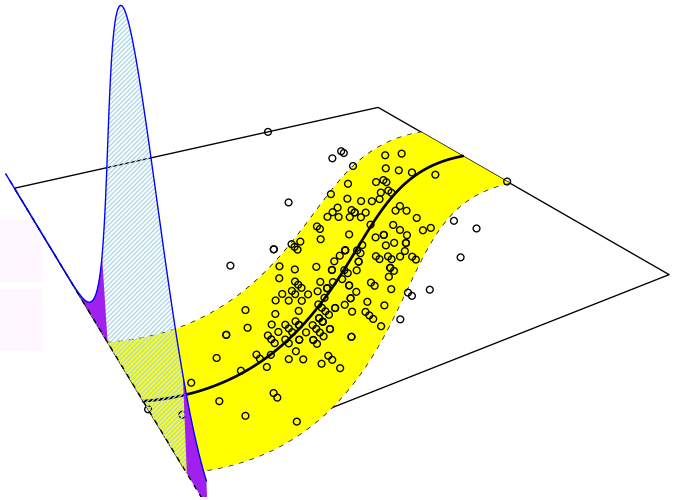


Linear, Non-Linear and Generalized Linear

NonLinear / NonParametric Model:

- $(Y|\boldsymbol{X}=\boldsymbol{x}) \sim \mathcal{N}(\theta_{\boldsymbol{x}}, \sigma^2)$
- $\mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}] = \theta_{\boldsymbol{x}} = m(\boldsymbol{x})$

```
1 > fit <- lm(y ~ poly(x, k), data = df)
2 > fit <- lm(y ~ bs(x), data = df)
```



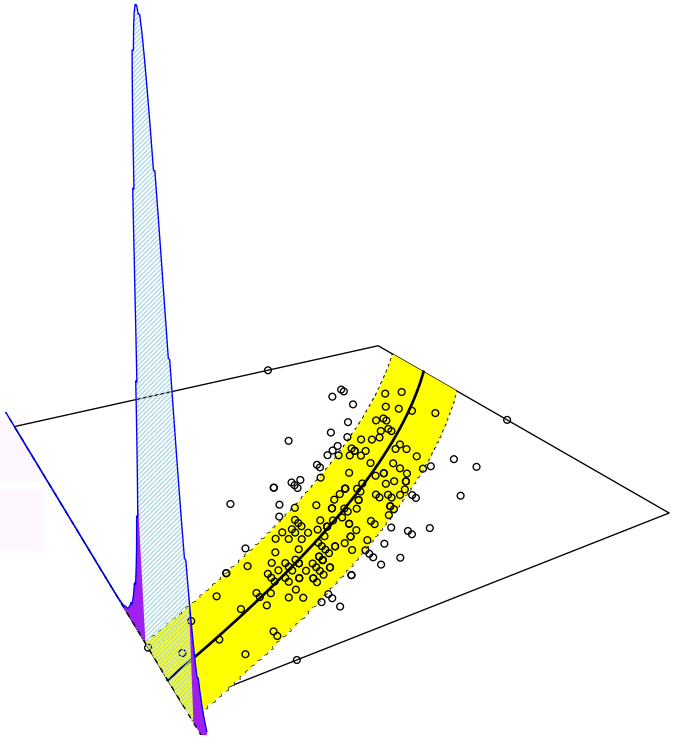
Linear, Non-Linear and Generalized Linear

Generalized Linear Model:

- $(Y|\boldsymbol{X}=\boldsymbol{x}) \sim \mathcal{L}(\theta_{\boldsymbol{x}}, \varphi)$
- $\mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}] = h^{-1}(\theta_{\boldsymbol{x}}) = h^{-1}(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta})$

```
1 > fit <- glm(y ~ x, data = df,
2 + family = poisson(link = "log")
```

e.g. $(Y|\boldsymbol{X}=\boldsymbol{x}) \sim \mathcal{P}(\exp[\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\beta}])$.



Linear Model

Consider a linear regression model, $y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \varepsilon_i$.

$\boldsymbol{\beta}$ is estimated using ordinary least squares, $\widehat{\boldsymbol{\beta}} = [\boldsymbol{X}^\top \boldsymbol{X}]^{-1} \boldsymbol{X}^\top \boldsymbol{Y}$

$\longrightarrow$ best linear unbiased estimator

Unbiased estimators in important in statistics because they have nice mathematical properties (see Cramér-Rao lower bound).

Looking for biased estimators (bias-variance tradeoff) becomes important in high-dimension, see Burr & Fry (2005)

Linear Model and Loss Functions

Consider a linear model, with some general loss function ℓ, set $\ell(x,y)=R(x-y)$ and consider,

$$\widehat{\boldsymbol{\beta}} \in \operatorname{argmin}\left\{\sum_{i=1}^{n} \ell(y_i, \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta})\right\}$$

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.

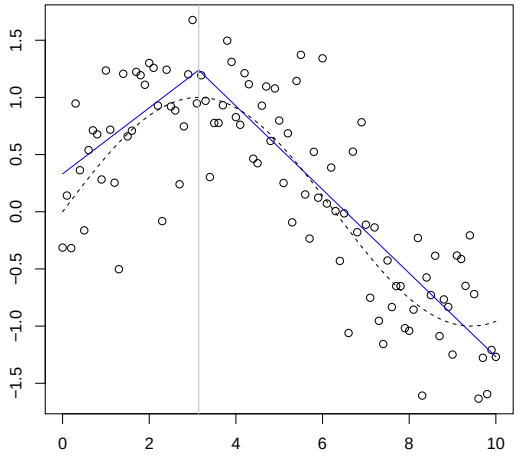
Linear Model and Loss Functions

But weights are unknown: use and iterative algorithm

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

Bagging Linear Models

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



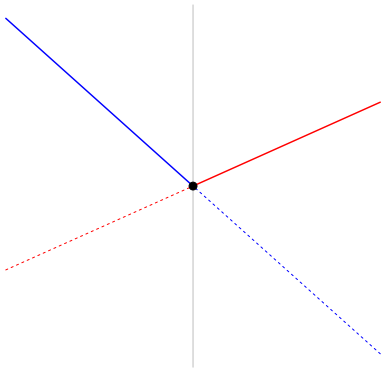
Regression Smoothers, *natura non facit saltus*

In statistical learning procedures, a key role is played by basis functions. We will see that it is common to assume that

$$m(\boldsymbol{x}) = \sum_{m=0}^{M} \beta_M h_m(\boldsymbol{x}),$$

where h_0 is usually a constant function and h_m defined basis functions.

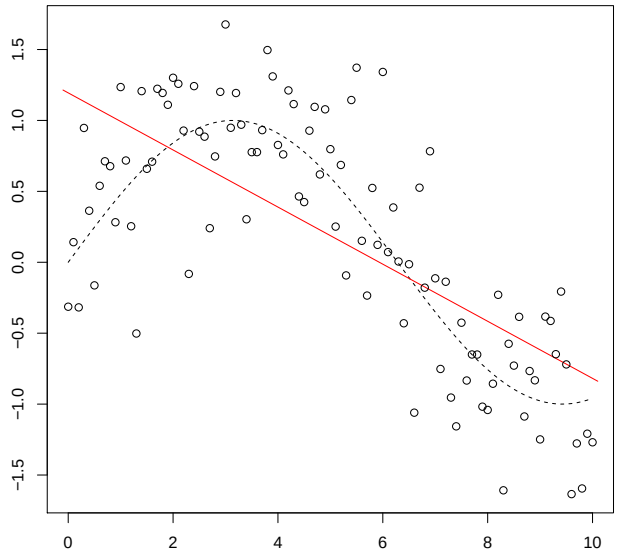
For instance, $h_m(x) = x^m$ for a polynomial expansion with a single predictor, or $h_m(x) = (x - s_m)_+$ for some knots s_m's (for linear splines, but one can consider quadratic or cubic ones).



Regression Smoothers: Polynomial Functions

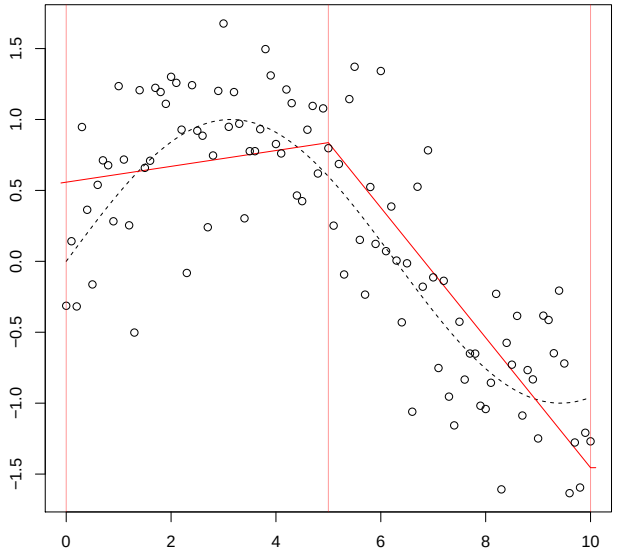
Stone-Weiestrass theorem every continuous function defined on a closed interval $[a, b]$ can be uniformly approximated as closely as desired by a polynomial function

```
> fit <- lm(Y ~ poly(X,k), data = db)
> predict(fit, newdata = data.frame(X=x))
```



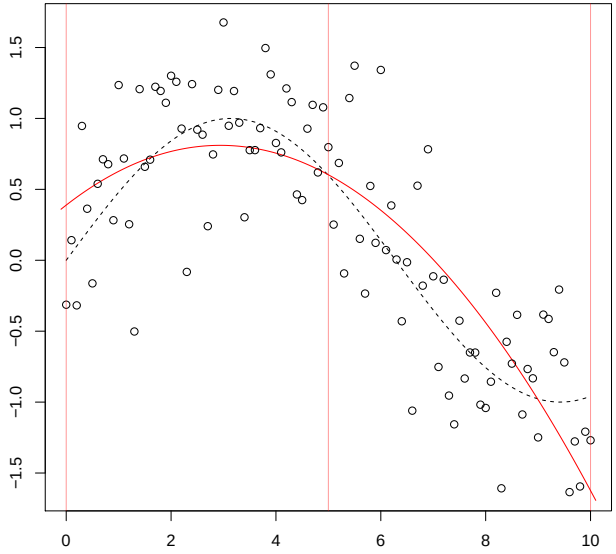
Regression Smoothers: Spline Functions

```
1 > fit <- lm(Y ~ bs(X,k,degree=1), data = db)
2 > predict(fit, newdata = data.frame(X=x))
```



Regression Smoothers: Spline Functions

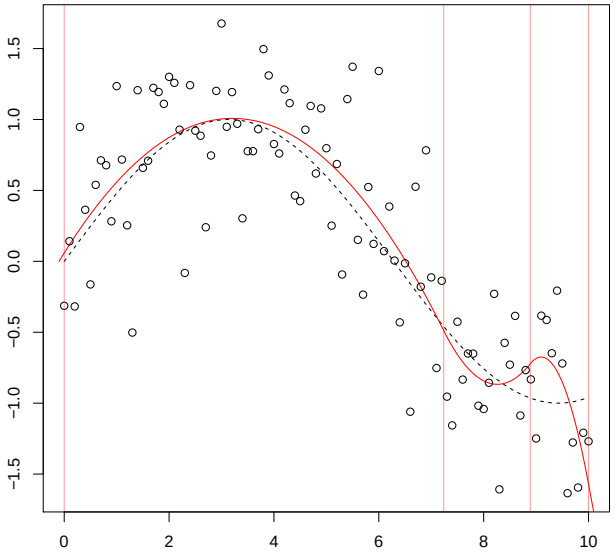
```
> fit <- lm(Y ~ bs(X,k,degree=2), data = db)
> predict(fit, newdata = data.frame(X=x))
```



see Generalized Additive Models.

Fixed Knots vs. Optimized Ones

```
1 > library(freeknotsplines)
2 > gen <- freelsgen(db$X, db$Y, degree=2,
      numknot=s)
3 > fit <- lm(Y ~ bs(X, gen@optknot, degree=2)
      , data = db)
4 > predict(fit, newdata = data.frame(X=x))
```



Interpretation of Penalty

Unbiased estimators are important in mathematical statistics, but are they the *best* estimators ?

Consider a sample, i.i.d., $\{y_1, \cdots, y_n\}$ with distribution $\mathcal{N}(\mu, \sigma^2)$. Define $\widehat{\theta} = \alpha \overline{Y}$. What is the optimal $\alpha^\star$ to get the best estimator of μ ?

- bias: $\text{bias}\left(\widehat{\theta}\right) = \mathbb{E}\left(\widehat{\theta}\right) - \mu = (\alpha - 1)\mu$
- variance: $\text{Var}\left(\widehat{\theta}\right) = \dfrac{\alpha^2\sigma^2}{n}$
- mse: $\text{mse}\left(\widehat{\theta}\right) = (\alpha - 1)^2\mu^2 + \dfrac{\alpha^2\sigma^2}{n}$

The optimal value is $\alpha^\star = \dfrac{\mu^2}{\mu^2 + \dfrac{\sigma^2}{n}} < 1.$

Linear Model

Consider some linear model $y_i = \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta} + \varepsilon_i$ for all $i = 1, \cdots, n$.

Assume that ε_i are i.i.d. with $\mathbb{E}(\varepsilon) = 0$ (and finite variance). Write

$$\underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}}_{\boldsymbol{y}, n\times 1} = \underbrace{\begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,k} \end{pmatrix}}_{\boldsymbol{X}, n\times(k+1)} \underbrace{\begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}}_{\boldsymbol{\beta}, (k+1)\times 1} + \underbrace{\begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}}_{\boldsymbol{\varepsilon}, n\times 1}.$$

Assuming $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\mathbb{I})$, the maximum likelihood estimator of $\boldsymbol{\beta}$ is

$$\widehat{\boldsymbol{\beta}} = \text{argmin}\{\|\boldsymbol{y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{\beta}\|_{\ell_2}\} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y}$$

... under the assumtption that $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ is a full-rank matrix.

What if $\boldsymbol{X}_i^\mathsf{T}\boldsymbol{X}$ cannot be inverted? Then $\widehat{\boldsymbol{\beta}} = [\boldsymbol{X}^\mathsf{T}\boldsymbol{X}]^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y}$ does not exist, but $\widehat{\boldsymbol{\beta}}_\lambda = [\boldsymbol{X}^\mathsf{T}\boldsymbol{X} + \lambda\mathbb{I}]^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y}$ always exist if $\lambda > 0$.

Ridge Regression

The estimator $\widehat{\boldsymbol{\beta}} = [\boldsymbol{X}^\mathsf{T}\boldsymbol{X} + \lambda\mathbb{I}]^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y}$ is the Ridge estimate obtained as solution of

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{argmin}} \left\{ \sum_{i=1}^{n} [y_i - \beta_0 - \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta}]^2 + \lambda \underbrace{\|\boldsymbol{\beta}\|_{\ell_2}}_{\mathbf{1}^\mathsf{T}\boldsymbol{\beta}^2} \right\}$$

for some tuning parameter λ. One can also write

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta};\|\boldsymbol{\beta}\|_{\ell_2}\leq s}{\text{argmin}} \{\|\boldsymbol{Y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{\beta}\|_{\ell_2}\}$$

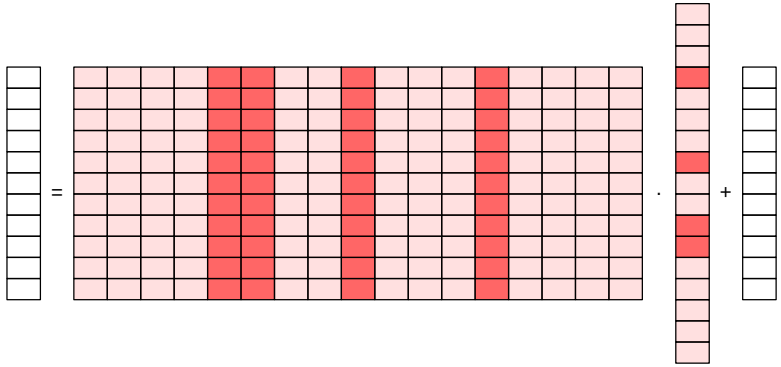
Remark Note that we solve $\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{argmin}}\{\text{objective}(\boldsymbol{\beta})\}$ where

$$\text{objective}(\boldsymbol{\beta}) = \underbrace{\mathcal{L}(\boldsymbol{\beta})}_{\text{training loss}} + \underbrace{\mathcal{R}(\boldsymbol{\beta})}_{\text{regularization}}$$

Going further on sparcity issues

In severall applications, k can be (very) large, but a lot of features are just noise: $\beta_j = 0$ for many j's. Let s denote the number of relevent features, with $s << k$, cf Hastie, Tibshirani & Wainwright (2015),

$$s = \text{card}\{\mathcal{S}\} \text{ where } \mathcal{S} = \{j; \beta_j \neq 0\}$$



The model is now $y = \boldsymbol{X}_{\mathcal{S}}^{\mathsf{T}}\boldsymbol{\beta}_{\mathcal{S}} + \varepsilon$, where $\boldsymbol{X}_{\mathcal{S}}^{\mathsf{T}}\boldsymbol{X}_{\mathcal{S}}$ is a full rank matrix.

Going further on sparcity issues

Define $\|\boldsymbol{a}\|_{\ell_0} = \sum \mathbf{1}(|a_i| > 0)$. Ici $\dim(\boldsymbol{\beta}) = s$.

We wish we could solve

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta};\|\boldsymbol{\beta}\|_{\ell_0} \leq s}{\operatorname{argmin}} \{\|\boldsymbol{Y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{\beta}\|_{\ell_2}\}$$

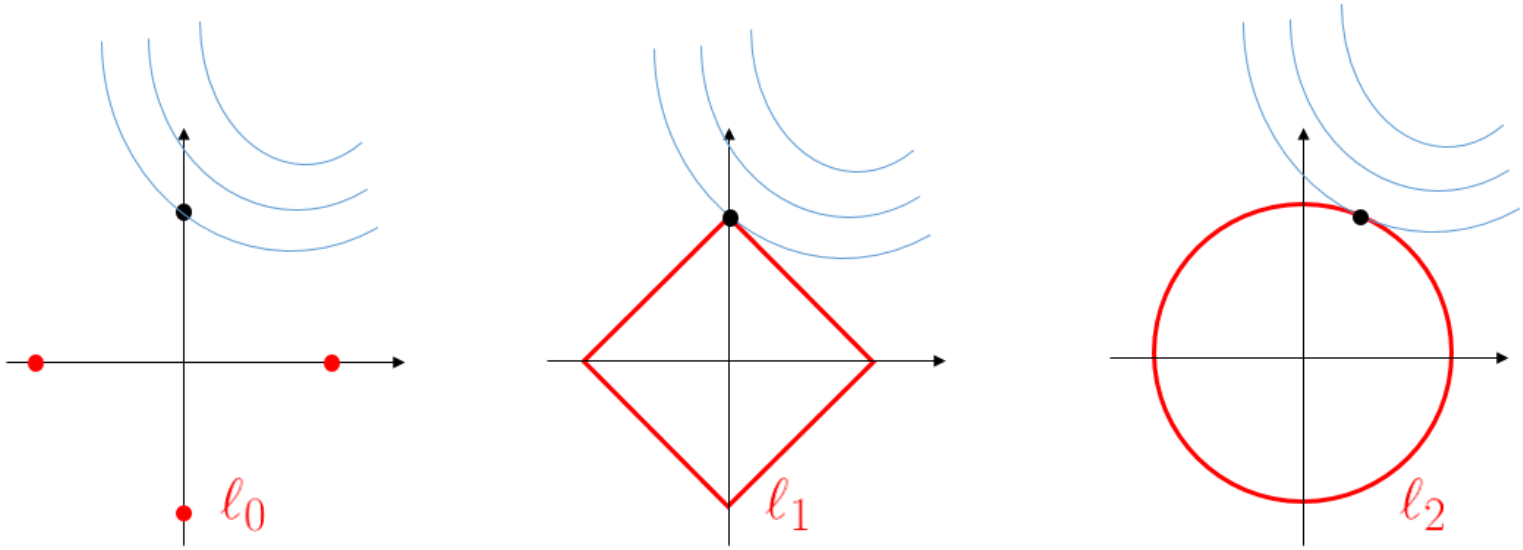
Problem: it is usually not possible to describe all possible constraints, since $\binom{s}{k}$ coefficients should be chosen here (with k (very) large).

Idea: solve the dual problem

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta};\|\boldsymbol{Y}-\boldsymbol{X}^\mathsf{T}\boldsymbol{\beta}\|_{\ell_2} \leq h}{\operatorname{argmin}} \{\|\boldsymbol{\beta}\|_{\ell_0}\}$$

where we might convexify the ℓ_0 norm, $\|\cdot\|_{\ell_0}$.

Regularization ℓ_0, ℓ_1 et ℓ_2



Optimal LASSO Penalty

Use cross validation, e.g. K-fold,

$$\widehat{\boldsymbol{\beta}}_{(-k)}(\lambda) = \text{argmin}\left\{\sum_{i\notin\mathcal{I}_k}[y_i - \boldsymbol{x}_i^\intercal\boldsymbol{\beta}]^2 + \lambda\|\boldsymbol{\beta}\|\right\}$$

then compute the sum of the squared errors,

$$Q_k(\lambda) = \sum_{i\in\mathcal{I}_k}[y_i - \boldsymbol{x}_i^\intercal\widehat{\boldsymbol{\beta}}_{(-k)}(\lambda)]^2$$

and finally solve

$$\lambda^\star = \text{argmin}\left\{\overline{Q}(\lambda) = \frac{1}{K}\sum_k Q_k(\lambda)\right\}$$

Note that this might overfit, so Hastie, Tibshiriani & Friedman (2009) suggest the largest λ such that

$$\overline{Q}(\lambda) \leq \overline{Q}(\lambda^\star) + \text{se}[\lambda^\star] \text{ with } \text{se}[\lambda]^2 = \frac{1}{K^2}\sum_{k=1}^K[Q_k(\lambda) - \overline{Q}(\lambda)]^2$$

Going further on sparcity issues

On $[-1,+1]^k$, the convex hull of $\|\boldsymbol{\beta}\|_{\ell_0}$ is $\|\boldsymbol{\beta}\|_{\ell_1}$

On $[-a,+a]^k$, the convex hull of $\|\boldsymbol{\beta}\|_{\ell_0}$ is $a^{-1}\|\boldsymbol{\beta}\|_{\ell_1}$

Hence,

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta};\|\boldsymbol{\beta}\|_{\ell_1}\leq\tilde{s}}{\operatorname{argmin}} \{\|\boldsymbol{Y}-\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}\|_{\ell_2}\}$$

is equivalent (Kuhn-Tucker theorem) to the Lagragian optimization problem

$$\widehat{\boldsymbol{\beta}} = \operatorname{argmin}\{\|\boldsymbol{Y}-\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta}\|_{\ell_2}+\lambda\|\boldsymbol{\beta}\|_{\ell_1}\}$$

LASSO *Least Absolute Shrinkage and Selection Operator*

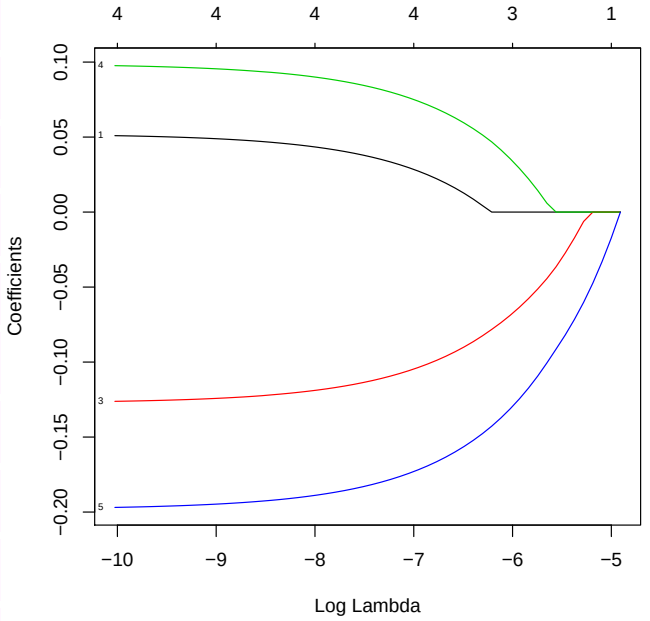
$$\widehat{\boldsymbol{\beta}} \in \text{argmin}\{\|\boldsymbol{Y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{\beta}\|_{\ell_2} + \lambda\|\boldsymbol{\beta}\|_{\ell_1}\}$$

is a convex problem (several algorithms*), but not strictly convex (no unicity of the minimum). Nevertheless, predictions $\widehat{\boldsymbol{y}} = \boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{\beta}}$ are unique

* MM, minimize majorization, coordinate descent Hunter (2003).

```
> freq = merge(contrat,nombre_RC)
> freq = merge(freq,nombre_DO)
> freq[,10]=as.factor(freq[,10])
> mx=cbind(freq[,c(4,5,6)],freq[,9]=="D",
    freq[,3]%in%c("A","B","C"))
> colnames(mx)=c(names(freq)[c(4,5,6)],"
    diesel","zone")
> for(i in 1:ncol(mx)) mx[,i]=(mx[,i]-mean(
    mx[,i]))/sd(mx[,i])
> names(mx)
[1]    puissance agevehicule ageconducteur
       diesel        zone
> library(glmnet)
> fit = glmnet(x=as.matrix(mx), y=freq[,11],
     offset=log(freq[,2]), family = "poisson
    ")
> plot(fit, xvar="lambda", label=TRUE)
```

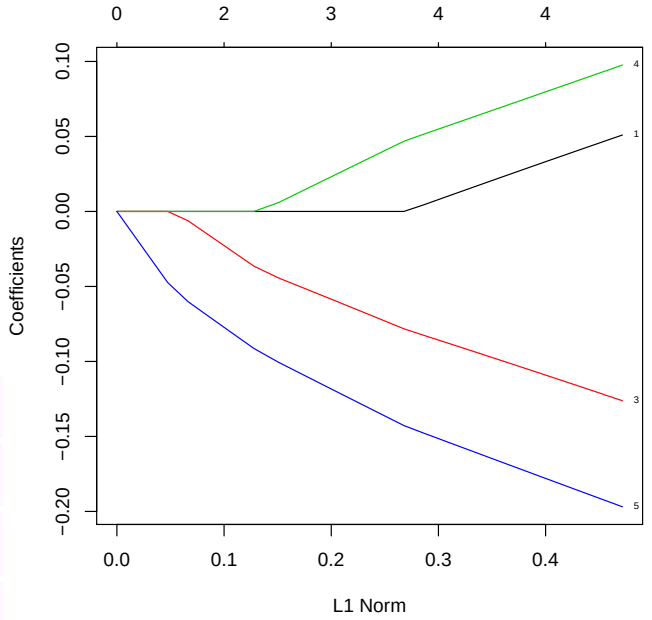
LASSO, third party

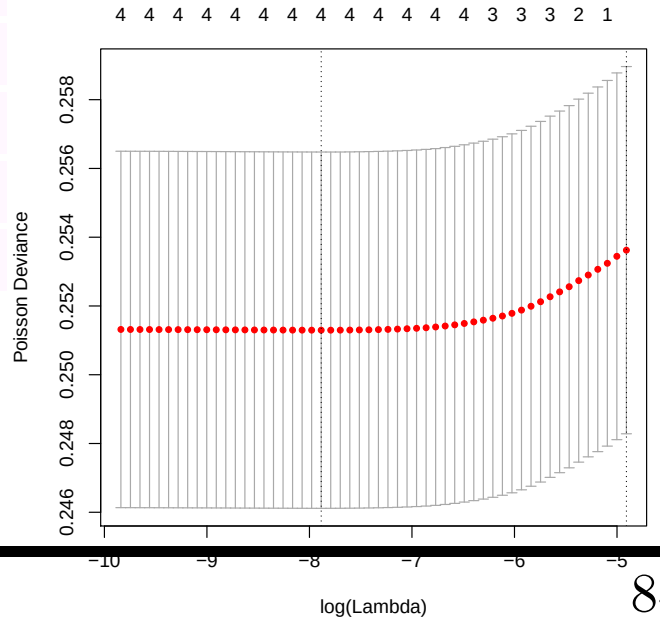


LASSO, third party

```
> plot(fit,label=TRUE)
> cvfit = cv.glmnet(x=as.matrix(mx), y=freq
    [,11], offset=log(freq[,2]),family = "
    poisson")
> plot(cvfit)
> cvfit$lambda.min
[1] 0.0002845703
> log(cvfit$lambda.min)
[1] -8.16453
```

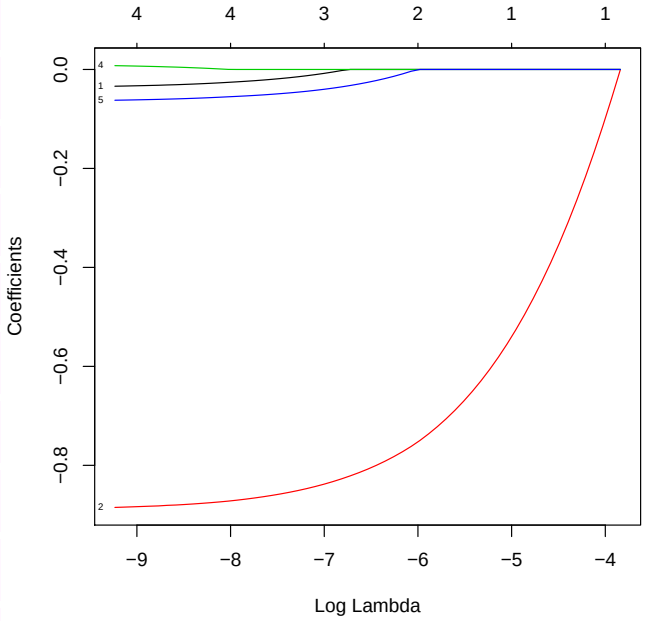
- Cross validation curve + error bars





```
> freq = merge(contrat,nombre_RC)
> freq = merge(freq,nombre_DO)
> freq[,10]=as.factor(freq[,10])
> mx=cbind(freq[,c(4,5,6)],freq[,9]=="D",
    freq[,3]%in%c("A","B","C"))
> colnames(mx)=c(names(freq)[c(4,5,6)],"
    diesel","zone")
> for(i in 1:ncol(mx)) mx[,i]=(mx[,i]-mean(
    mx[,i]))/sd(mx[,i])
> names(mx)
[1]    puissance agevehicule ageconducteur
       diesel        zone
> library(glmnet)
> fit = glmnet(x=as.matrix(mx), y=freq[,12],
     offset=log(freq[,2]), family = "poisson
    ")
> plot(fit, xvar="lambda", label=TRUE)
```

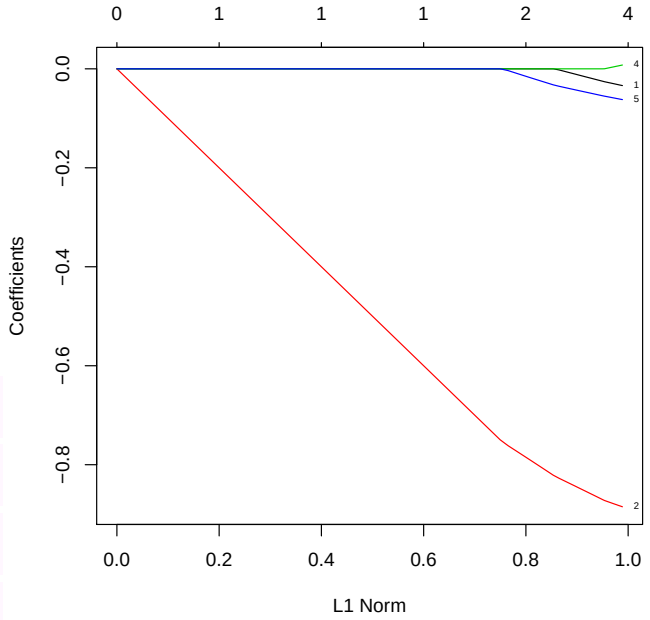
LASSO, Fréquence DO

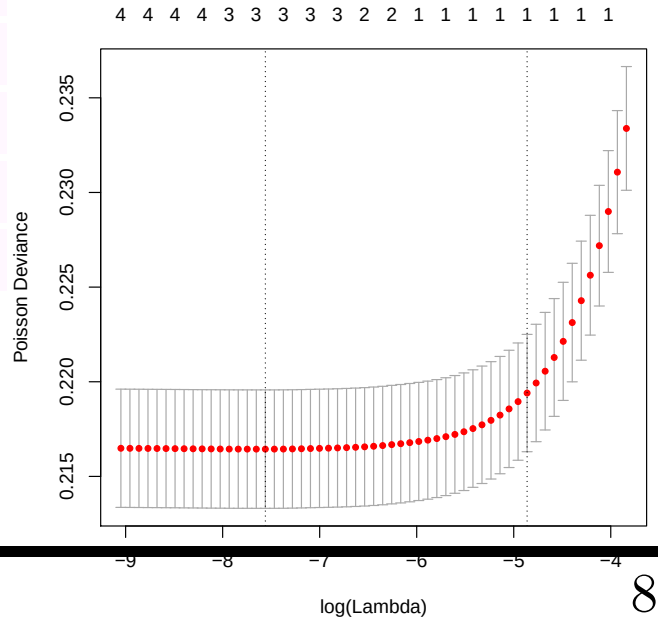


LASSO, material

```
> plot(fit,label=TRUE)
> cvfit = cv.glmnet(x=as.matrix(mx), y=freq
    [,12], offset=log(freq[,2]),family = "
    poisson")
> plot(cvfit)
> cvfit$lambda.min
[1] 0.0004744917
> log(cvfit$lambda.min)
[1] -7.653266
```

- Cross validation curve + error bars





Some thoughts about Tuning parameters

Regularization is a key issue in machine learning, to avoid overfitting.

In (traditional) econometrics are based on plug-in methods: see Silverman bandwith rule in Kernel density estimation,

$$h^\star = \left(\frac{4\widehat{\sigma}^5}{3n}\right) \sim 1.06\widehat{\sigma}n^{-1/5}.$$

In machine learning literature, use on out-of-sample cross-validation methods for choosing amount of regularization.

Optimal LASSO Penalty

Use cross validation, e.g. K-fold,

$$\widehat{\boldsymbol{\beta}}_{(-k)}(\lambda) = \text{argmin} \left(\{ \sum_{i \notin \mathcal{I}_k} [y_i - \boldsymbol{x}_i^\intercal \boldsymbol{\beta}]^2 + \lambda \sum_k |\beta_k| \right\}$$

then compute the sum or the squared errors,

$$Q_k(\lambda) = \sum_{i \notin \mathcal{I}_k} [y_i - \boldsymbol{x}_i^\intercal \widehat{\boldsymbol{\beta}}_{(-k)}(\lambda)]^2$$

and finally solve

$$\lambda^\star = \text{argmin} \left\{ \overline{Q}(\lambda) = \frac{1}{K} \sum_k Q_k(\lambda) \right\}$$

Note that this might overfit, so Hastie, Tibshiriani & Friedman (2009) suggest the largest λ such that

$$\overline{Q}(\lambda) \leq \overline{Q}(\lambda^\star) + \text{se}[\lambda^\star] \text{ with } \text{se}[\lambda]^2 = \frac{1}{K^2} \sum_{k=1}^K [Q_k(\lambda) - \overline{Q}(\lambda)]^2$$

Big Data, Oracle and Sparcity

Assume that k is large, and that $\boldsymbol{\beta} \in \mathbb{R}^k$ can be partitioned as $\boldsymbol{\beta} = (\boldsymbol{\beta}_{\text{imp}}, \boldsymbol{\beta}_{\text{non-imp}})$, as well as covariates $\boldsymbol{x} = (\boldsymbol{x}_{\text{imp}}, \boldsymbol{x}_{\text{non-imp}})$, with important and non-important variables, i.e. $\boldsymbol{\beta}_{\text{non-imp}} \sim \mathbf{0}$.

Goal : achieve variable selection and make inference of $\boldsymbol{\beta}_{\text{imp}}$

Oracle property of high dimensional model selection and estimation, see Fan and Li (2001). Only the oracle knows which variables are important...

If sample size is large enough $(n >> k_{\text{imp}} \left(1 + \log \frac{k}{k_{\text{imp}}}\right))$ we can do inference *as if* we knew which covariates were important: we can ignore the selection of covariates part, that is not relevant for the confidence intervals. This provides cover for ignoring the shrinkage and using regularstandard errors, see Athey & Imbens (2015).

Why Shrinkage Regression Estimates ?

Interesting for model selection (alternative to peanlized criterions) and to get a good balance between bias and variance.

In decision theory, an admissible decision rule is a rule for making a decision *such that there is not any other rule that is always better than it.*

When $k \geq 3$, ordinary least squares are not admissible, see the improvement by James–Stein estimator.

Regularization and Scalability

What if k is (extremely) large? never trust ols with more than five regressors (attributed to Zvi Griliches in Athey & Imbens (2015))

Use regularization techniques, see Ridge, Lasso, or subset selection

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} [y_i - \beta_0 - \boldsymbol{x}_i^\mathsf{T} \boldsymbol{\beta}]^2 + \lambda \|\boldsymbol{\beta}\|_{\ell_0} \text{ where } \|\boldsymbol{\beta}\|_{\ell_0} = \sum_k \mathbf{1}(\beta_k \neq 0). \right\}$$

Penalization and Splines

In order to get a sufficiently smooth model, why not penalyse the sum of squares of errors,

$$\sum_{i=1}^{n}[y_i - m(\boldsymbol{x}_i)]^2 + \lambda \int [m''(\boldsymbol{t})]^2 d\boldsymbol{t}$$

for some tuning parameter λ. Consider some cubic spline basis, so that

$$m(\boldsymbol{x}) = \sum_{j=1}^{J} \theta_j N_j(\boldsymbol{x})$$

then the optimal expression for m is obtained using

$$\widehat{\boldsymbol{\theta}} = [\boldsymbol{N}^\mathsf{T}\boldsymbol{N} + \lambda\Omega]^{-1}\boldsymbol{N}^\mathsf{T}\boldsymbol{y}$$

where $\boldsymbol{N}_{i,j}$ is the matrix of $N_j(\boldsymbol{X}_i)$'s and $\Omega_{i,j} = \int N_i''(\boldsymbol{t})N_j''(\boldsymbol{t})d\boldsymbol{t}$

Smoothing with Multiple Regressors

Actually

$$\sum_{i=1}^{n}[y_i - m(\boldsymbol{x}_i)]^2 + \lambda \int [m''(\boldsymbol{t})]^2 d\boldsymbol{t}$$

is based on some multivariate penalty functional, e.g.

$$\int [m''(\boldsymbol{t})]^2 d\boldsymbol{t} = \int \left[\sum_i \left(\frac{\partial^2 m(\boldsymbol{t})}{\partial \boldsymbol{t}_i^2} \right)^2 + 2 \sum_{i,j} \left(\frac{\partial^2 m(\boldsymbol{t})}{\partial \boldsymbol{t}_i \partial \boldsymbol{t}_j} \right)^2 \right] d\boldsymbol{t}$$

Regression Trees

The partitioning is sequential, one covariate at a time (see adaptative neighbor estimation).

Start with $Q = \sum_{i=1}^{n} [y_i - \overline{y}]^2$

For covariate k and threshold t, split the data according to $\{x_{i,k} \leq t\}$ (L) or $\{x_{i,k} > t\}$ (R). Compute

$$\overline{y}_L = \frac{\sum_{i,x_{i,k} \leq t} y_i}{\sum_{i,x_{i,k} \leq t} 1} \text{ and } \overline{y}_R = \frac{\sum_{i,x_{i,k} > t} y_i}{\sum_{i,x_{i,k} > t} 1}$$

and let

$$m_i^{(k,t)} = \begin{cases} \overline{y}_L \text{ if } x_{i,k} \leq t \\ \overline{y}_R \text{ if } x_{i,k} > t \end{cases}$$

Regression Trees

Then compute $(k^\star, t^\star) = \text{argmin}\left\{\sum_{i=1}^{n}[y_i - m_i^{(k,t)}]^2\right\}$, and partition the space intro two subspace, whether $x_{k^\star} \leq t^\star$, or not.

Then repeat this procedure, and minimize

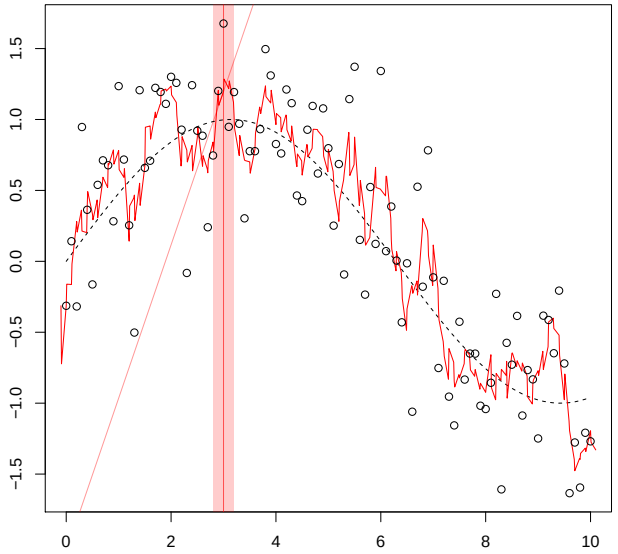
$$\sum_{i=1}^{n}[y_i - m_i]^2 + \lambda \cdot \#\{\text{leaves}\},$$

(cf LASSO).

One can also consider random forests with regression trees.

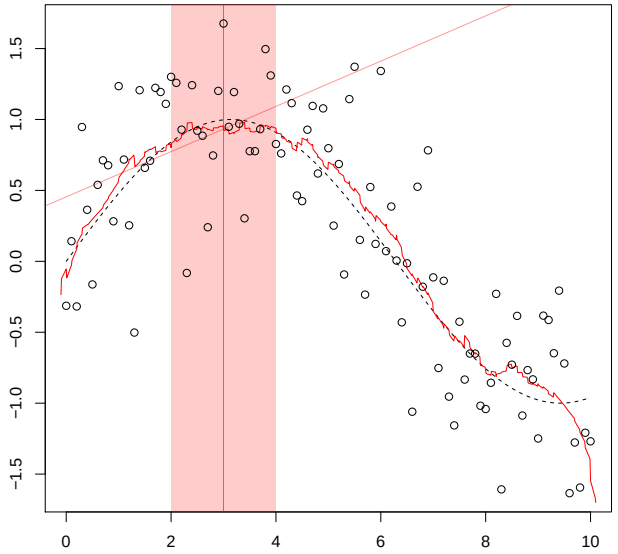
Local Regression

```
> W <- ( abs(db$X-x)<h )*1
> fit <- lm(Y ~ X, data = db, weights = W)
> predict(fit, newdata = data.frame(X=x))
```



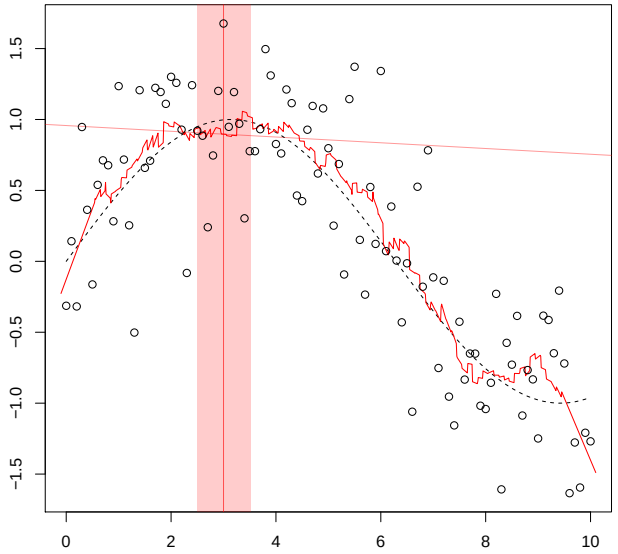
Local Regression

```
> W <- ( abs(db$X-x)<h )*1
> fit <- lm(Y ~ X, data = db, weights = W)
> predict(fit, newdata = data.frame(X=x))
```



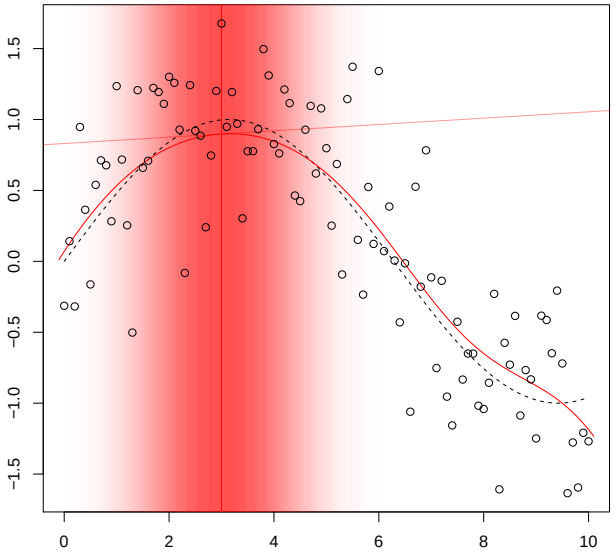
Local Regression : Nearest Neighbor

```
> W <- (rank( abs(db$X-x)<h ) <= k)*1
> fit <- lm(Y ~ X, data = db, weights = W)
> predict(fit, newdata = data.frame(X=x))
```



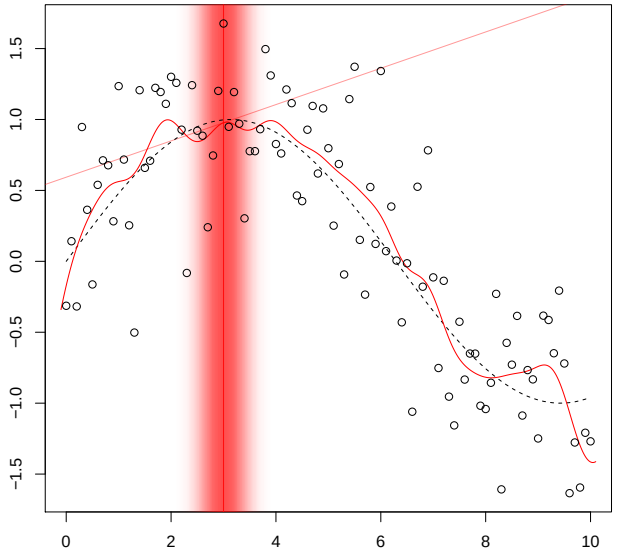
Local Regression : Kernel Based Smoothing

```
> library(KernSmooth)
> W <- dnorm( abs(db$X-x)<h )/h
> fit <- lm(Y ~ X, data = db, weights = W)
> predict(fit, newdata = data.frame(X=x))
> library(KernSmooth)
> library(sp)
```



Local Regression : Kernel Based Smoothing

```
> library(np)
> fit <- npreg(Y ~ X, data = db, bws = h,
+ ckertype = "gaussian")
> predict(fit, newdata = data.frame(X=x))
```



From Linear to Generalized Linear Models

The (Gaussian) Linear Model and the logistic regression have been extended to the wide class of the exponential family,

$$f(y|\theta,\phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right),$$

where $a(\cdot)$, $b(\cdot)$ and $c(\cdot)$ are functions, θ is the natural - canonical - parameter and ϕ is a nuisance parameter.

The Gaussian distribution $\mathcal{N}(\mu,\sigma^2)$ belongs to this family

$$\underbrace{\theta = \mu}_{\theta \leftrightarrow \mathbb{E}(Y)}, \quad \underbrace{\phi = \sigma^2}_{\phi \leftrightarrow \mathrm{Var}(Y)}, \; a(\phi) = \phi, \; b(\theta) = \theta^2/2$$

From Linear to Generalized Linear Models

The Bernoulli distribution $\mathcal{B}(p)$ belongs to this family

$$\underbrace{\theta = \log \frac{p}{1-p}}_{\theta = g_\star(\mathbb{E}(Y))}, \ a(\phi) = 1, \ b(\theta) = \log(1+\exp(\theta)), \text{ and } \phi = 1$$

where the $g_\star(\cdot)$ is some link function (here the logistic transformation): the canonical link.

Canonical links are

```
binomial(link = "logit")
gaussian(link = "identity")
Gamma(link = "inverse")
inverse.gaussian(link = "1/mu^2")
poisson(link = "log")
quasi(link = "identity", variance = "constant")
quasibinomial(link = "logit")
quasipoisson(link = "log")
```

From Linear to Generalized Linear Models

Observe that

$$\mu = \mathbb{E}(Y) = b'(\theta) \text{ and } \text{Var}(Y) = b''(\theta) \cdot \phi = \underbrace{b''([b']^{-1}(\mu)) \cdot \phi}_{\text{variance function } V(\mu)}$$

$\longrightarrow$ distributions are characterized by this variance function, e.g. $V(\mu) = 1$ for the Gaussian family (homoscedastic models), $V(\mu) = \mu$ for the Poisson and $V(\mu) = \mu^2$ for the Gamma distribution, $V(\mu) = \mu^3$ for the inverse-Gaussian family.

Note that $g_\star(\cdot) = [b']^{-1}(\cdot)$ is the canonical link.

Tweedie (1984) suggested a power-type variance function $V(\mu) = \mu^\gamma \cdot \phi$. When $\gamma \in [1, 2]$, then Y has a compound Poisson distribution with Gamma jumps.

```
> library(tweedie)
```

From the Exponential Family to GLM's

So far, there no regression model. Assume that

$$f(y_i|\theta_i,\phi)=\exp\left(\frac{y_i\theta_i-b(\theta_i)}{a(\phi)}+c(y_i,\phi)\right) \text{ where } \theta_i=g_\star^{-1}(g(\boldsymbol{x}_i^\top\boldsymbol{\beta}))$$

so that the log-likelihood is

$$\mathcal{L}(\boldsymbol{\theta},\phi|\boldsymbol{y})=\prod_{i=1}^n f(y_i|\theta_i,\phi)=\exp\left(\frac{\sum_{i=1}^n y_i\theta_i-\sum_{i=1}^n b(\theta_i)}{a(\phi)}+\sum_{i=1}^n c(y_i,\phi)\right).$$

To derive the first order condition, observe that we can write

$$\frac{\partial\log\mathcal{L}(\boldsymbol{\theta},\phi|\boldsymbol{y}_i)}{\partial\boldsymbol{\beta}_j}=\boldsymbol{\omega}_{i,j}\boldsymbol{x}_{i,j}[y_i-\mu_i]$$

for some $\boldsymbol{\omega}_{i,j}$ (see e.g. Müller (2004)) which are simple when $g_\star=g$.

From the Exponential Family to GLM's

The first order conditions can be writen

$$\boldsymbol{X}^\top \boldsymbol{W}^{-1}[\boldsymbol{y}-\boldsymbol{\mu}]=\boldsymbol{0}$$

which are first order conditions for a weighted linear regression model.

As for the logistic regression, $\boldsymbol{W}$ depends on unkown $\boldsymbol{\beta}$'s : use an iterative algorithm

1. Set $\widehat{\boldsymbol{\mu}}_0=\boldsymbol{y}$, $\boldsymbol{\theta}_0=g(\widehat{\boldsymbol{\mu}}_0)$ and

$$\boldsymbol{z}_0=\boldsymbol{\theta}_0+(\boldsymbol{y}-\widehat{\boldsymbol{\mu}}_0)g'(\widehat{\boldsymbol{\mu}}_0).$$

Define $\boldsymbol{W}_0=\text{diag}[g'(\widehat{\boldsymbol{\mu}}_0)^2\text{Var}(\widehat{\boldsymbol{y}})]$ and fit a (weighted) lineare regression of $\boldsymbol{Z}_0$ on $\boldsymbol{X}$, i.e.

$$\widehat{\boldsymbol{\beta}}_1=[\boldsymbol{X}^\top\boldsymbol{W}_0^{-1}\boldsymbol{X}]^{-1}\boldsymbol{X}^\top\boldsymbol{W}_0^{-1}\boldsymbol{z}_0$$

2. Set $\widehat{\boldsymbol{\mu}}_k=\boldsymbol{X}\widehat{\boldsymbol{\beta}}_k$, $\boldsymbol{\theta}_k=g(\widehat{\boldsymbol{\mu}}_k)$ and

$$\boldsymbol{z}_k=\boldsymbol{\theta}_k+(\boldsymbol{y}-\widehat{\boldsymbol{\mu}}_k)g'(\widehat{\boldsymbol{\mu}}_k).$$

From the Exponential Family to GLM's

Define $\boldsymbol{W}_k = \text{diag}[g'(\widehat{\boldsymbol{\mu}}_k)^2 \text{Var}(\widehat{\boldsymbol{y}})]$ and fit a (weighted) lineare regression of $\boldsymbol{Z}_k$ on $\boldsymbol{X}$, i.e.

$$\widehat{\boldsymbol{\beta}}_{k+1} = [\boldsymbol{X}^\top \boldsymbol{W}_k^{-1} \boldsymbol{X}]^{-1} \boldsymbol{X}^\top \boldsymbol{W}_k^{-1} \boldsymbol{Z}_k$$

and loop... until changes in $\widehat{\boldsymbol{\beta}}_{k+1}$ are (sufficiently) small. Then set $\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}_\infty$

Under some technical conditions, we can prove that $\widehat{\boldsymbol{\beta}} \xrightarrow{\mathbb{P}} \boldsymbol{\beta}$ and

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

where numerically $I(\boldsymbol{\beta}) = \phi \cdot [\boldsymbol{X}^\top \boldsymbol{W}_\infty^{-1} \boldsymbol{X}])$.

From the Exponential Family to GLM's

We estimate (see linear regression estimation) ϕ by

$$\widehat{\phi} = \frac{1}{n - \dim(\boldsymbol{X})} \sum_{i=1}^{n} \boldsymbol{\omega}_{i,i} \frac{[y_i - \widehat{\mu}_i]^2}{\text{Var}(\widehat{\boldsymbol{\mu}}_i)}$$

This asymptotic expression can be used to derive confidence intervals, or tests. But is might be a poor approximation when n is small. See use of boostrap in claims reserving.

Those are theorerical results: in practice, the algorithm may fail to converge

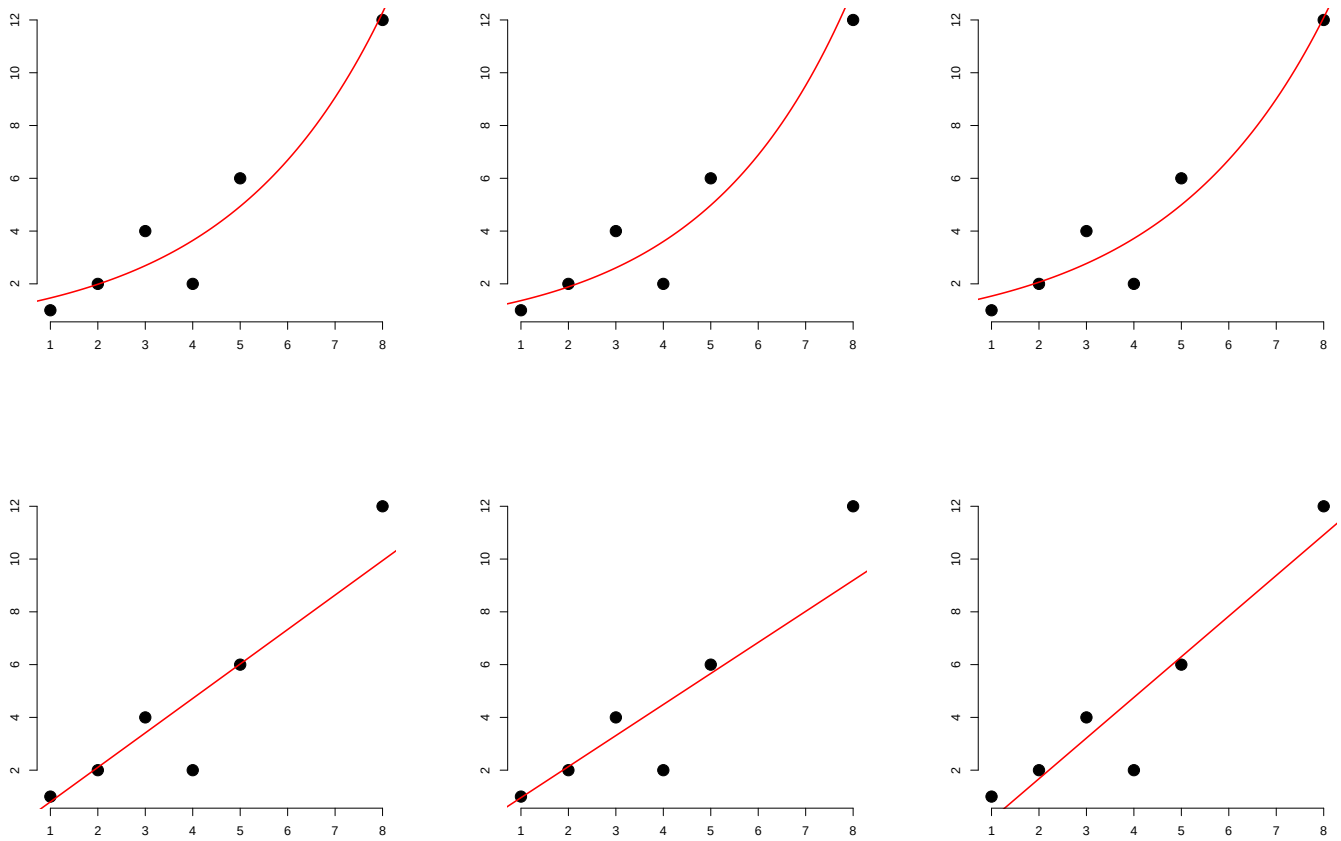
GLM's outside the Exponential Family?

Actually, it is possible to consider more general distributions, see Yee (2014))

```
> library(VGAM)
> vglm(y ~ x, family = Makeham)
> vglm(y ~ x, family = Gompertz)
> vglm(y ~ x, family = Erlang)
> vglm(y ~ x, family = Frechet)
> vglm(y ~ x, family = pareto1(location=100))
```

Those functions can also be used for a multivariate response $\boldsymbol{y}$

GLM: Link and Distribution



GLM: Distribution?

From a computational point of view, the Poisson regression is not (really) related to the Poisson distribution.

Here we solve the first order conditions (or normal equations)

$$\sum_i [Y_i - \exp(\boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta})] X_{i,j} = 0 \ \forall j$$

with unconstraint $\boldsymbol{\beta}$, using Fisher's scoring technique $\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k - \boldsymbol{H}_k^{-1}\nabla_k$

$$\text{where } \boldsymbol{H}_k = -\sum_i \exp(\boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta}_k)\boldsymbol{X}_i\boldsymbol{X}_i^\mathsf{T} \text{ and } \nabla_k = \sum_i \boldsymbol{X}_i^\mathsf{T}[Y_i - \exp(\boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta}_k)]$$

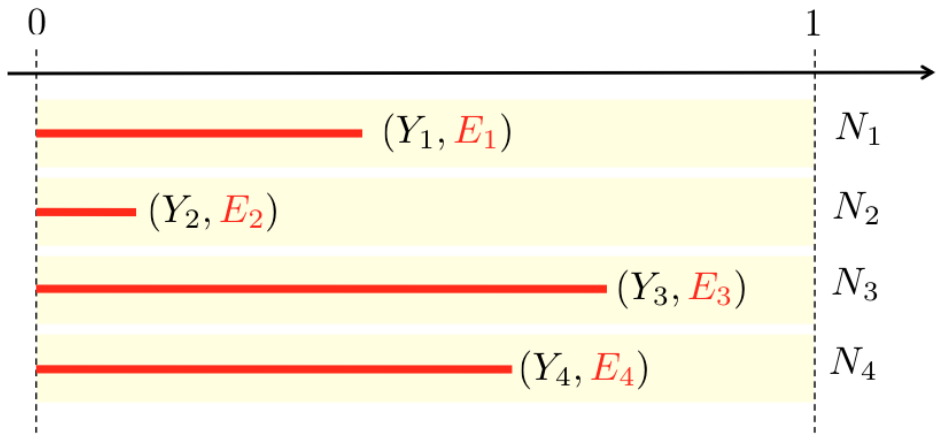
$\longrightarrow$ There is no assumption here about $Y \in \mathbb{N}$: it is possible to run a Poisson regression on non-integers.

The Exposure and (Annual) Claim Frequency

In General Insurance, we should predict blueyearly claims frequency. Let N_i denote the number of claims over one year for contrat i.

We did observe only the contract for a period of time E_i

Let Y_i denote the observed number of claims, over period $[0, E_i]$.



The Exposure and (Annual) Claim Frequency

Assuming that claims occurence is driven by a Poisson process of intensity λ, if $N_i \sim \mathcal{P}(\lambda)$, then $Y_i \sim \mathcal{P}(\lambda \cdot E_i)$, where N is the annual frequency.

$$\mathcal{L}(\lambda, \boldsymbol{Y}, \boldsymbol{E}) = \prod_{i=1}^{n} \frac{e^{-\lambda E_i} [\lambda E_i]^{Y_i}}{Y_i!}$$

the first order condition is

$$\frac{\partial}{\partial \lambda} \log \mathcal{L}(\lambda, \boldsymbol{Y}, \boldsymbol{E}) = -\sum_{i=1}^{n} E_i + \frac{1}{\lambda} \sum_{i=1}^{n} Y_i = 0$$

for

$$\widehat{\lambda} = \frac{\sum_{i=1}^{n} Y_i}{\sum_{i=1}^{n} E_i} = \sum_{i=1}^{n} \omega_i \frac{Y_i}{E_i} \text{ where } \omega_i = \frac{E_i}{\sum_{i=1}^{n} E_i}$$

The Exposure and (Annual) Claim Frequency

Assume that

$$Y_i \sim \mathcal{P}(\lambda_i \cdot E_i) \text{ where } \lambda_i = \exp[\boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta}].$$

Here $\mathbb{E}(Y_i|\boldsymbol{X}_i) = \text{Var}(Y_i|\boldsymbol{X}_i) = \lambda_i = \exp[\boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta} + \log E_i]$.

$$\log \mathcal{L}(\boldsymbol{\beta}; \boldsymbol{Y}) = \sum_{i=1}^{n} Y_i \cdot [\boldsymbol{X}_i^\mathsf{T}\boldsymbol{\beta} + \log E_i] - (\exp[\boldsymbol{X}_i\boldsymbol{\beta}] + \log E_i) - \log(Y_i!)$$

```
> model <- glm(y~x,offset=log(E),family=poisson)
> model <- glm(y~x + offset(log(E)),family=poisson)
```

Taking into account the exposure in other models is difficult...

Boosting

Boosting is a machine learning ensemble meta-algorithm for reducing bias primarily and also variance in supervised learning, and a family of machine learning algorithms which convert weak learners to strong ones. (source: Wikipedia)

The heuristics is simple: we consider an iterative process where we keep modeling the errors.

Fit model for $\boldsymbol{y}$, $m_1(\cdot)$ from $\boldsymbol{y}$ and $\boldsymbol{X}$, and compute the error, $\boldsymbol{\varepsilon}_1 = \boldsymbol{y} - m_1(\boldsymbol{X})$.

Fit model for $\boldsymbol{\varepsilon}_1$, $m_2(\cdot)$ from $\boldsymbol{\varepsilon}_1$ and $\boldsymbol{X}$, and compute the error, $\boldsymbol{\varepsilon}_2 = \boldsymbol{\varepsilon}_1 - m_2(\boldsymbol{X})$, etc. Then set

$$m(\cdot) = \underbrace{m_1(\cdot)}_{\sim y} + \underbrace{m_2(\cdot)}_{\sim \varepsilon_1} + \underbrace{m_3(\cdot)}_{\sim \varepsilon_2} + \cdots + \underbrace{m_k(\cdot)}_{\sim \varepsilon_{k-1}}$$

Boosting

With (very) general notations, we want to solve

$$m^\star = \text{argmin}\{\mathbb{E}[\ell(Y, m(\boldsymbol{X}))]\}$$

for some loss function ℓ.

It is an iterative procedure: assume that at some step k we have an estimator $m_k(\boldsymbol{X})$. Why not constructing a new model that might improve our model,

$$m_{k+1}(\boldsymbol{X}) = m_k(\boldsymbol{X}) + h(\boldsymbol{X}).$$

What $h(\cdot)$ could be?

Boosting

In a perfect world, $h(\boldsymbol{X}) = \boldsymbol{y} - m_k(\boldsymbol{X})$, which can be interpreted as a residual.

Note that this residual is the gradient of $\frac{1}{2}[y - m(\boldsymbol{x})]^2$

A gradient descent is based on Taylor expansion

$$\underbrace{f(\boldsymbol{x}_k)}_{\langle f, \boldsymbol{x}_k \rangle} \sim \underbrace{f(\boldsymbol{x}_{k-1})}_{\langle f, \boldsymbol{x}_{k-1} \rangle} + \underbrace{(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})}_{\alpha} \underbrace{\nabla f(\boldsymbol{x}_{k-1})}_{\langle \nabla f, \boldsymbol{x}_{k-1} \rangle}$$

But here, it is different. We claim we can write

$$\underbrace{f_k(\boldsymbol{x})}_{\langle f_k, \boldsymbol{x} \rangle} \sim \underbrace{f_{k-1}(\boldsymbol{x})}_{\langle f_{k-1}, \boldsymbol{x} \rangle} + \underbrace{(f_k - f_{k-1})}_{\beta} \underbrace{?}_{\langle f_{k-1}, \nabla \boldsymbol{x} \rangle}$$

where ? is interpreted as a 'gradient'.

Boosting

Here, f_k is a $\mathbb{R}^d \to \mathbb{R}$ function, so the gradient should be in such a (big) functional space $\to$ want to approximate that function.

$$m_k(\boldsymbol{x}) = m_{k-1}(\boldsymbol{x}) + \underset{f \in \mathcal{F}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \ell(Y_i, m_{k-1}(\boldsymbol{x}) + f(\boldsymbol{x})) \right\}$$

where $f \in \mathcal{F}$ means that we seek in a class of weak learner functions.

If learner are two strong, the first loop leads to some fixed point, and there is no learning procedure, see linear regression $y = \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta} + \varepsilon$. Since $\varepsilon \perp \boldsymbol{x}$ we cannot learn from the residuals.

Boosting with some Shrinkage

Consider here some quadratic loss function.

In order to make sure that we learn weakly, we can use some shrinkage parameter ν (or collection of parameters ν_j) so that

$$\mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}]=m(\boldsymbol{x})\sim m_M(\boldsymbol{x})=\sum_{j=1}^{M}\nu_j h_j(\boldsymbol{x})$$

The problem is always the same. At stage j, we should solve

$$\min_{h(\cdot)}\left\{\sum_{i=1}^{n}[\underbrace{y_i-m_{j-1}(\boldsymbol{x}_i)}_{\varepsilon_{i,j-1}}-h(\boldsymbol{x}_i)]^2\right\}$$

Boosting with some Shrinkage

The algorithm is then

- start with some (simple) model $\boldsymbol{y} = h_1(\boldsymbol{x})$
- compute the residuals (including ν), $\boldsymbol{\varepsilon}_1 = \boldsymbol{y} - \nu h_1(\boldsymbol{x})$

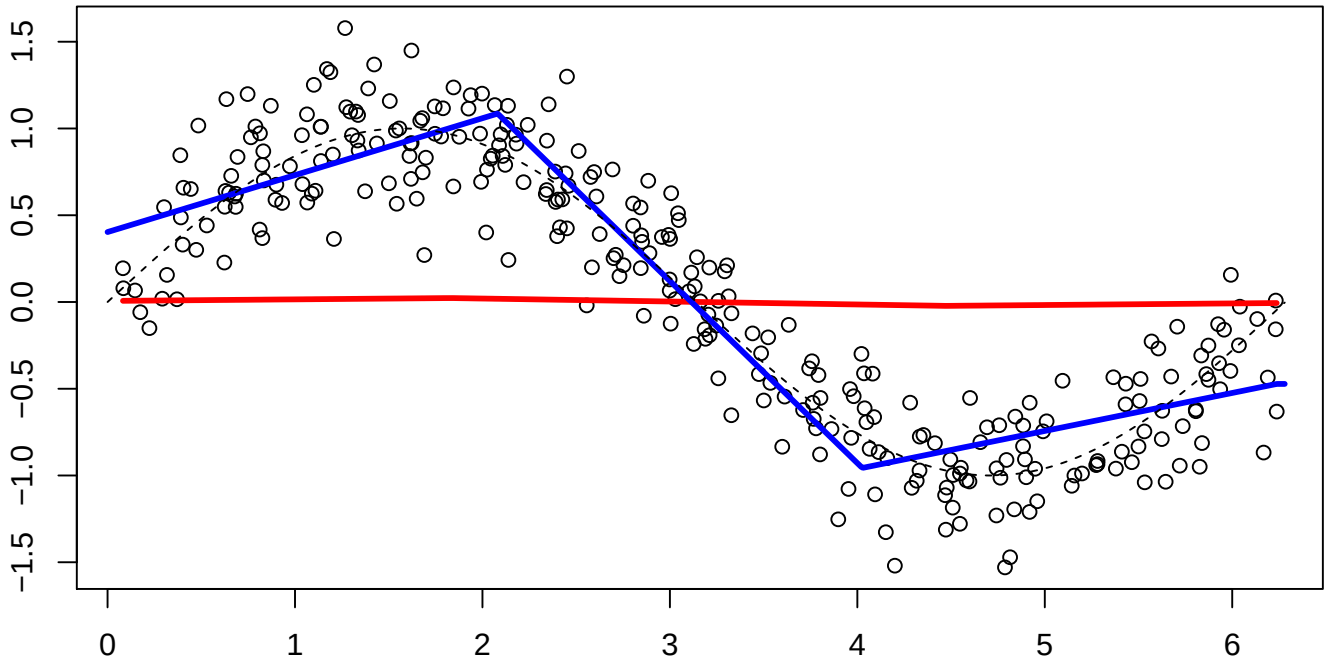
and at step j,

- consider some (simple) model $\boldsymbol{\varepsilon}_j = h_j(\boldsymbol{x})$
- compute the residuals (including ν), $\boldsymbol{\varepsilon}_{j+1} = \boldsymbol{\varepsilon}_j - \nu h_j(\boldsymbol{x})$

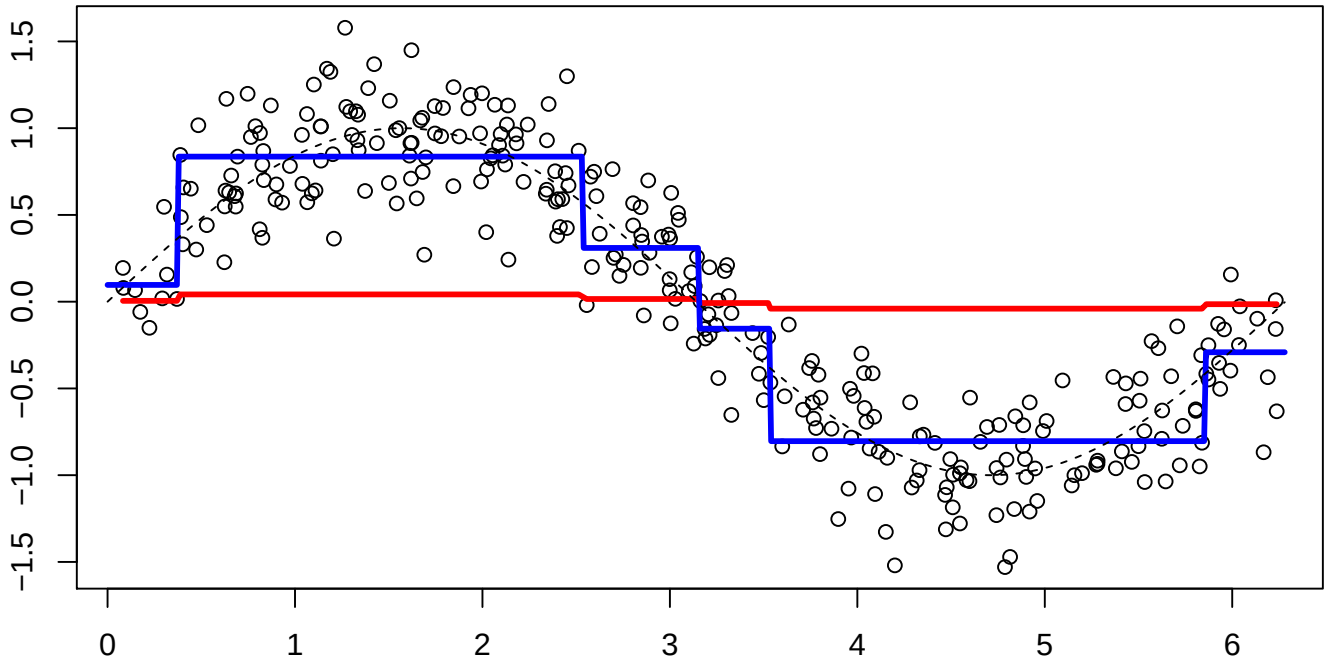
and loop. And set finally

$$\widehat{\boldsymbol{y}} = \sum_{j=1}^{M} \nu h_j(\boldsymbol{x})$$

Boosting with Piecewise Linear Spline Functions



Boosting with Trees (Stump Functions)



Boosting for Classification

Still seek $m^\star(\cdot) = \text{argmin}\{\mathbb{E}[\ell(Y, m(\boldsymbol{X}))]\}$

Here $y \in \{-1, +1\}$, and use $\ell(y, m(\boldsymbol{x})) = e^{-y \cdot m(\boldsymbol{x})}$: AdaBoot algorithm.

Note that

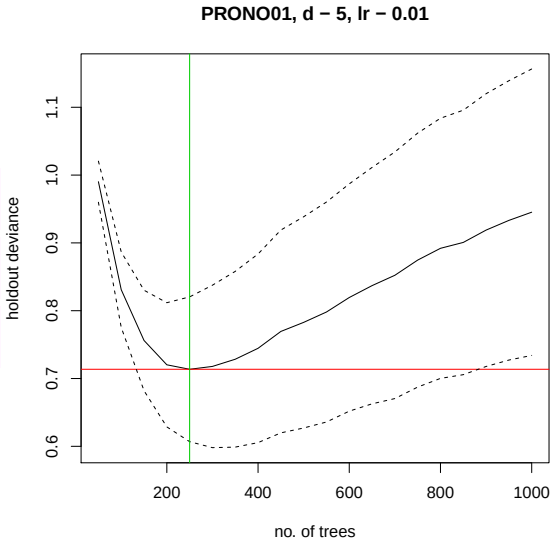
$$\mathbb{P}[Y = +1 | \boldsymbol{X} = \boldsymbol{x}] = \frac{1}{1 + e^{2m^\star \boldsymbol{x}}}$$

cf probit transform... Can be seen as iteration on weights. At step k solve

$$\underset{h(\cdot)}{\text{argmin}} \left\{ \sum_{i=1}^{n} \underbrace{e^{y_i \cdot m_k(\boldsymbol{x}_i)}}_{\omega_{i,k}} \cdot e^{y_i \cdot h(\boldsymbol{x}_i)} \right\}$$

Boosting for Classification

```
> gbm.step(data=myocarde, gbm.x = 1:7, gbm.y = 8,
+ family = "bernoulli", tree.complexity = 5,
+ learning.rate = 0.01, bag.fraction = 0.5)
```



Exponential distribution, deviance, loss function, residuals, etc

- Gaussian distribution $\longleftrightarrow$ ℓ_2 loss function

Deviance is $\sum_{i=1}^{n}(y_i - m(\boldsymbol{x}_i))^2$, with gradient $\widehat{\varepsilon}_i = y_i - m(\boldsymbol{x}_i)$

- Laplace distribution $\longleftrightarrow$ ℓ_1 loss function

Deviance is $\sum_{i=1}^{n}|y_i - m(\boldsymbol{x}_i))|$, with gradient $\widehat{\varepsilon}_i = \text{sign}(y_i - m(\boldsymbol{x}_i))$

Exponential distribution, deviance, loss function, residuals, etc

- Bernoullli $\{-1,+1\}$ distribution $\longleftrightarrow$ ℓ_{adaboost} loss function

Deviance is $\sum_{i=1}^{n} e^{-y_i m(\boldsymbol{x}_i)}$, with gradient $\widehat{\varepsilon}_i = -y_i e^{-[y_i] m(\boldsymbol{x}_i)}$

- Bernoullli $\{0,1\}$ distribution

Deviance $2\sum_{i=1}^{n}[y_i \cdot \log\left(\frac{y_i}{m(\boldsymbol{x}_i)}\right)(1-y_i)\log\left(\frac{1-y_i}{1-m(\boldsymbol{x}_i)}\right)$ with gradient

$$\widehat{\varepsilon}_i = y_i - \frac{\exp[m(\boldsymbol{x}_i)]}{1+\exp[m(\boldsymbol{x}_i)]}$$

- Poisson distribution

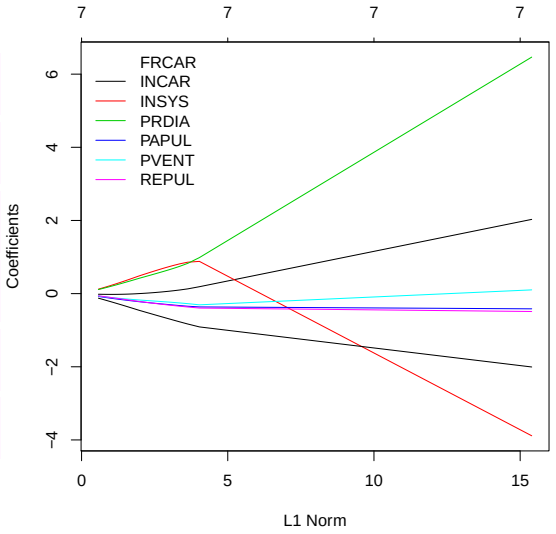
Deviance $2\sum_{i=1}^{n}\left(y_i \cdot \log\left(\frac{y_i}{m(\boldsymbol{x}_i)}\right) - [y_i - m(\boldsymbol{x}_i)]\right)$ with gradient $\widehat{\varepsilon}_i = \frac{y_i - m(\boldsymbol{x}_i)}{\sqrt{m(\boldsymbol{x}_i)}}$

Regularized GLM

In Regularized GLMs, we introduced a penalty in the loss function (the deviance), see e.g. ℓ_1 regularized logistic regression

$$\max\left\{\sum_{i=1}^{n}\left(y_i[\beta_0+\boldsymbol{x}_i^\top\boldsymbol{\beta}-\log[1+e^{\beta_0+\boldsymbol{x}_i^\top\boldsymbol{\beta}}]]\right)-\lambda\sum_{j=1}^{k}|\beta_j|\right\}$$

```
> library(glmnet)
> y <- myocarde$PRONO
> x <- as.matrix(myocarde[,1:7])
> glm_ridge <- glmnet(x, y, alpha=0, lambda=seq
    (0,2,by=.01), family="binomial")
> plot(lm_ridge)
```



Collective vs. Individual Model

Consider a Tweedie distribution, with variance function power $p \in (0,1)$, mean μ and scale parameter ϕ, then it is a compound Poisson model,

- $N \sim \mathcal{P}(\lambda)$ with $\lambda = \dfrac{\phi\mu^{2-p}}{2-p}$
- $Y_i \sim \mathcal{G}(\alpha,\beta)$ with $\alpha = -\dfrac{p-2}{p-1}$ and $\beta = \dfrac{\phi\mu^{1-p}}{p-1}$

Consversely, consider a compound Poisson model $N \sim \mathcal{P}(\lambda)$ and $Y_i \sim \mathcal{G}(\alpha,\beta)$,

- variance function power is $p = \dfrac{\alpha+2}{\alpha+1}$
- mean is $\mu = \dfrac{\lambda\alpha}{\beta}$
- scale parameter is $\phi = \dfrac{[\lambda\alpha]^{\frac{\alpha+2}{\alpha+1}-1}\beta^{2-\frac{\alpha+2}{\alpha+1}}}{\alpha+1}$

seems to be equivalent... but it's not.

Collective vs. Individual Model

In the context of regression

$$N_i \sim \mathcal{P}(\lambda_i) \text{ with } \lambda_i = \exp[\boldsymbol{X}_i^{\mathsf{T}}\boldsymbol{\beta}_\lambda]$$

$$Y_{j,i} \sim \mathcal{G}(\mu_i, \phi) \text{ with } \mu_i = \exp[\boldsymbol{X}_i^{\mathsf{T}}\boldsymbol{\beta}_\mu]$$

Then $S_i = Y_{1,i} + \cdots + Y_{N,i}$ has a Tweedie distribution

- variance function power is $p = \dfrac{\phi+2}{\phi+1}$
- mean is $\lambda_i\mu_i$
- scale parameter is $\dfrac{\lambda_i^{\frac{1}{\phi+1}-1}}{\mu_i^{\frac{\phi}{\phi+1}}}\left(\dfrac{\phi}{1+\phi}\right)$

There are $1+2\text{dim}(\boldsymbol{X})$ degrees of freedom.

Collective vs. Individual Model

Note that the scale parameter should not depend on i. A Tweedie regression is

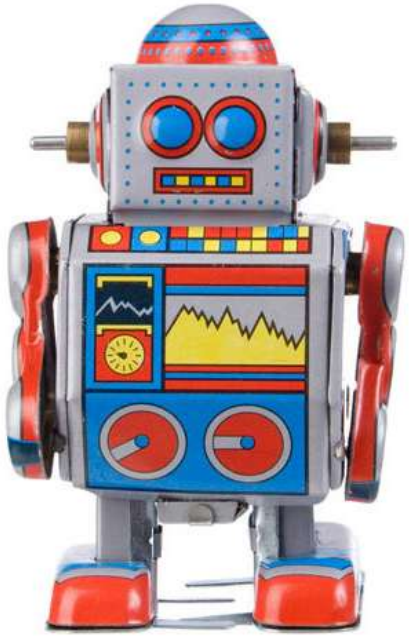
- variance function power is $p = \in (0, 1)$
- mean is $\mu_i = \exp[\boldsymbol{X}_i^{\mathsf{T}} \boldsymbol{\beta}_{\text{Tweedie}}]$
- scale parameter is ϕ

There are $2 + \dim(\boldsymbol{X})$ degrees of freedom.

Note that oone can easily boost a Tweedie model

```
> library(TDboost)
```

Part 3.
Model Choice, Feature Selection, etc.



AIC, BIC

AIC and BIC are both maximum likelihood estimate driven and penalize useless parameters(to avoid overfitting)

$$AIC = -2\log[\text{likelihood}] + 2k \text{ and } BIC = -2\log[\text{likelihood}] + \log(n)k$$

AIC focus on overfit, while BIC depends on n so it might also avoid underfit

BIC penalize complexity more than AIC does.

Minimizing AIC $\Leftrightarrow$ minimizing cross-validation value, Stone (1977).

Minimizing BIC $\Leftrightarrow$ k-fold leave-out cross-validation, Shao (1997), with $k = n[1 - (\log n - 1)]$

$\longrightarrow$ used in econometric stepwise procedures

Cross-Validation

Formally, the leave-one-out cross validation is based on

$$CV = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \widehat{m}_{-i}(\boldsymbol{x}_i))$$

where $\widehat{m}_{-i}$ is obtained by fitting the model on the sample where observation i has been dropped, e.g.

$$CV = \frac{1}{n}\sum_{i=1}^{n} [y_i, \widehat{m}_{-i}(\boldsymbol{x}_i)]^2$$

The Generalized cross-validation, for a quadratic loss function, is defined as

$$GCV = \frac{1}{n}\sum_{i=1}^{n} \left[\frac{y_i - \widehat{m}_{-i}(\boldsymbol{x}_i)}{1 - \text{trace}(\boldsymbol{S})/n}\right]^2$$

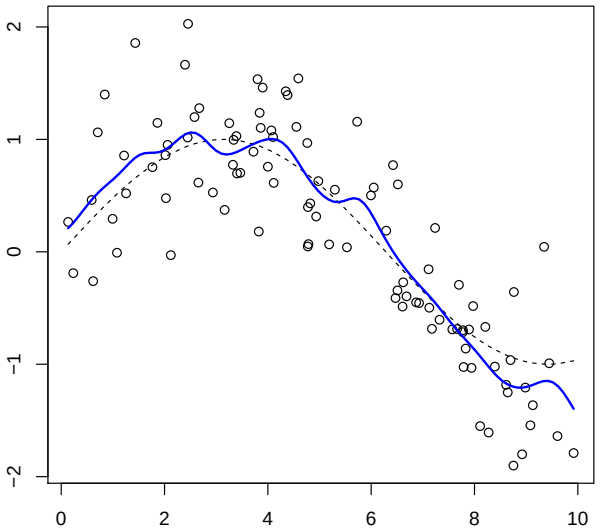
Cross-Validation for kernel based local regression

Econometric approach

Define $\widehat{m}(x) = \widehat{\beta}_0^{[x]} + \widehat{\beta}_1^{[x]} x$ with

$$(\widehat{\beta}_0^{[x]}, \widehat{\beta}_1^{[x]}) = \underset{(\beta_0,\beta_1)}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \omega_{h^\star}^{[x]} [y_i - (\beta_0 + \beta_1 x_i)]^2 \right\}$$

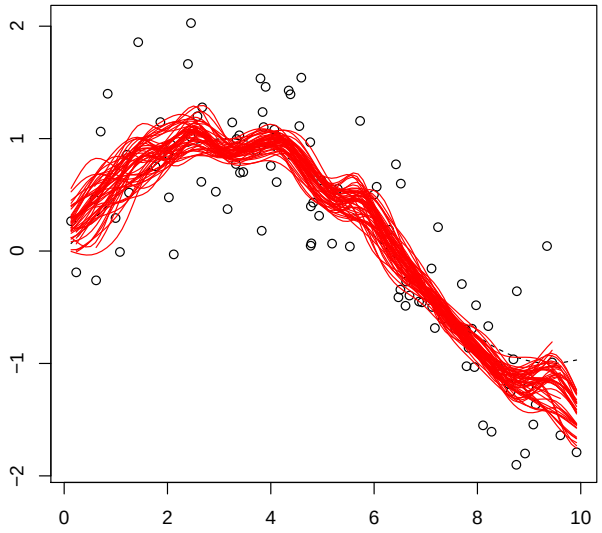
where $h^\star$ is given by some rule of thumb (see previous discussion).

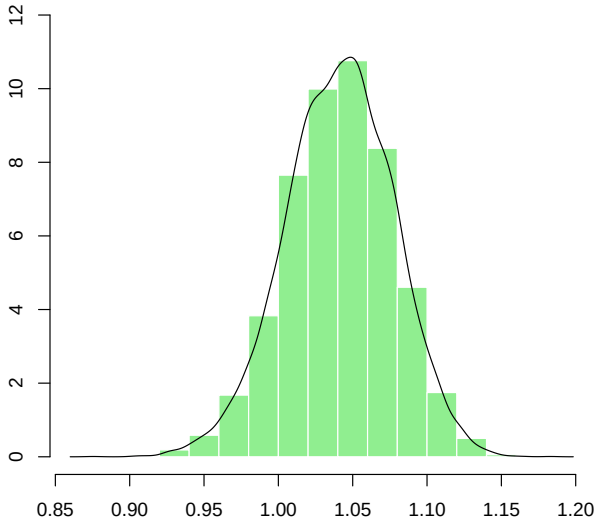


Cross-Validation for kernel based local regression

Bootstrap based approach

Use bootstrap samples, compute $h_b^\star$, and get $\widehat{m}_b(x)$'s.





Cross-Validation for kernel based local regression

Statistical learning approach (Cross Validation (leave-one-out))

Given $j \in \{1, \cdots, n\}$, given h, solve

$$(\widehat{\beta}_0^{[(i),h]}, \widehat{\beta}_1^{[(i),h]}) = \underset{(\beta_0,\beta_1)}{\text{argmin}} \left\{ \sum_{j \neq i} \omega_h^{(i)} [Y_j - (\beta_0 + \beta_1 x_j)]^2 \right\}$$

and compute $\widehat{m}_{(i)}^{[h]}(x_i) = \widehat{\beta}_0^{[(i),h]} + \widehat{\beta}_1^{[(i),h]} x_i$. Define

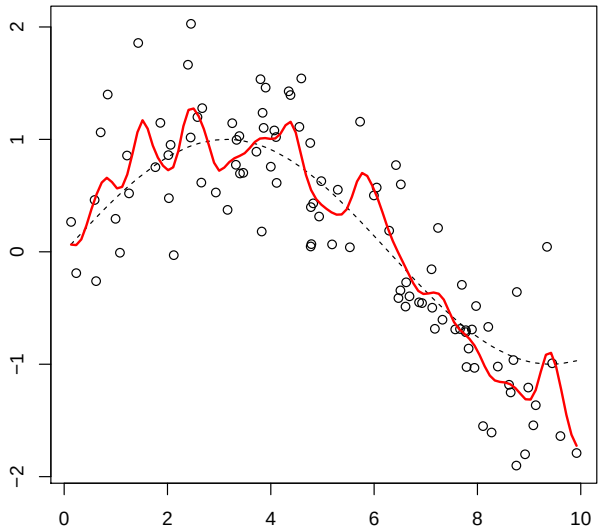
$$\text{mse}(h) = \sum_{i=1}^{n} [y_i - \widehat{m}_{(i)}^{[h]}(x_i)]^2$$

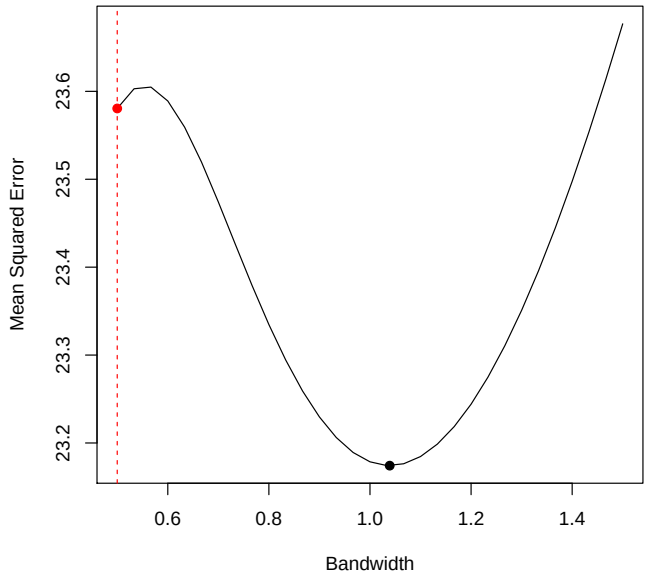
and set $h^\star = \text{argmin}\{\text{mse}(h)\}$.

Then compute $\widehat{m}(x) = \widehat{\beta}_0^{[x]} + \widehat{\beta}_1^{[x]} x$ with

$$(\widehat{\beta}_0^{[x]}, \widehat{\beta}_1^{[x]}) = \underset{(\beta_0,\beta_1)}{\text{argmin}} \left\{ \sum_{i=1}^{n} \omega_{h^\star}^{[x]} [y_i - (\beta_0 + \beta_1 x_i)]^2 \right\}$$

Cross-Validation for kernel based local regression





Cross-Validation for kernel based local regression

Statistical learning approach (Cross Validation (k-fold))

Given $\mathcal{I} \in \{1, \cdots, n\}$, given h, solve

$$(\widehat{\beta}_0^{[(\mathcal{I}),h]}, \widehat{\beta}_1^{[x_i,h]}) = \underset{(\beta_0,\beta_1)}{\text{argmin}} \left\{ \sum_{j \notin \mathcal{I}} \omega_h^{(\mathcal{I})} [y_j - (\beta_0 + \beta_1 x_j)]^2 \right\}$$

and compute $\widehat{m}_{(\mathcal{I})}^{[h]}(x_i) = \widehat{\beta}_0^{[(i),h]} + \widehat{\beta}_1^{[(i),h]} x_i$, $\forall i \in \mathcal{I}$. Define

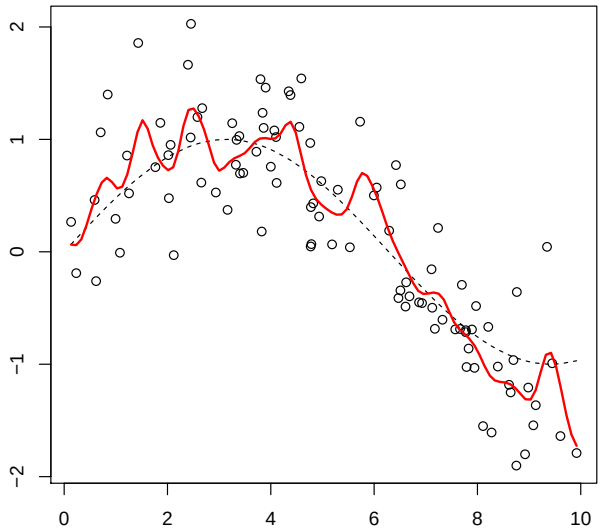
$$\text{mse}(h) = \sum_{\mathcal{I}} \sum_{i \in \mathcal{I}} [y_i - \widehat{m}_{(\mathcal{I})}^{[h]}(x_i)]^2$$

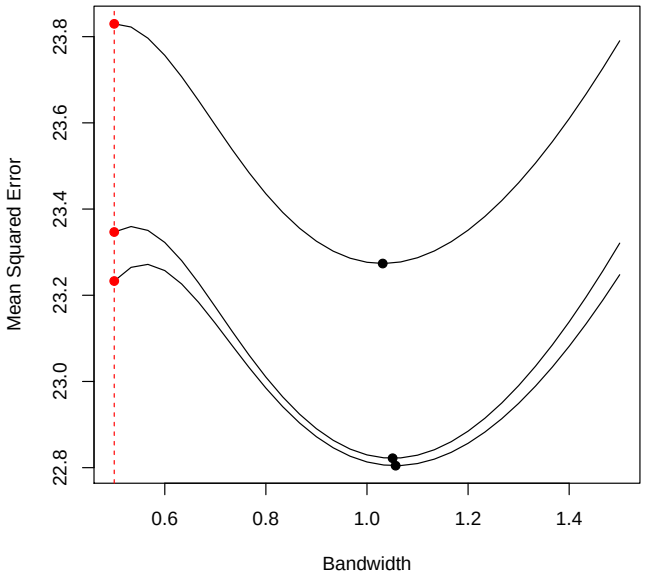
and set $h^\star = \text{argmin}\{\text{mse}(h)\}$.

Then compute $\widehat{m}(x) = \widehat{\beta}_0^{[x]} + \widehat{\beta}_1^{[x]} x$ with

$$(\widehat{\beta}_0^{[x]}, \widehat{\beta}_1^{[x]}) = \underset{(\beta_0,\beta_1)}{\text{argmin}} \left\{ \sum_{i=1}^{n} \omega_{h^\star}^{[x]} [y_i - (\beta_0 + \beta_1 x_i)]^2 \right\}$$

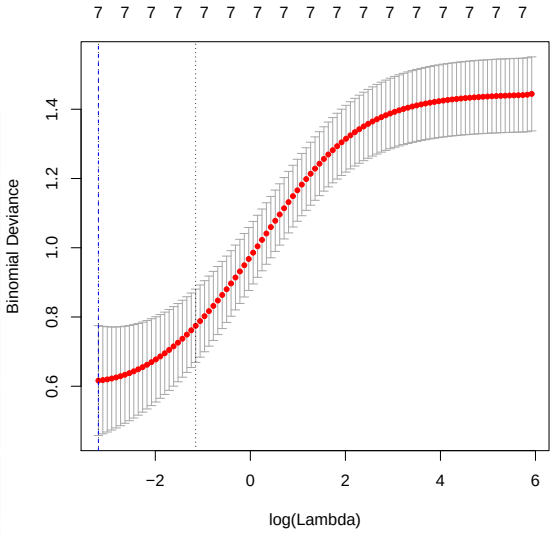
Cross-Validation for kernel based local regression

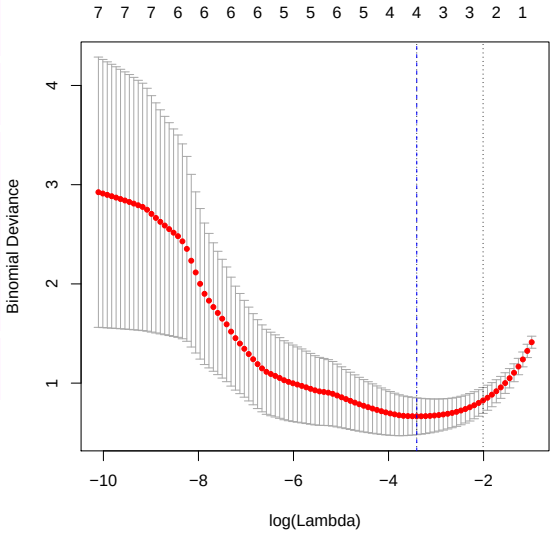




Cross-Validation for Ridge & Lasso

```
> library(glmnet)
> y <- myocarde$PRONO
> x <- as.matrix(myocarde[,1:7])
> cvfit <- cv.glmnet(x, y, alpha=0, family =
+ "binomial", type = "auc", nlambda = 100)
> cvfit$lambda.min
[1] 0.0408752
> plot(cvfit)
> cvfit <- cv.glmnet(x, y, alpha=1, family =
+ "binomial", type = "auc", nlambda = 100)
> cvfit$lambda.min
[1] 0.03315514
> plot(cvfit)
```



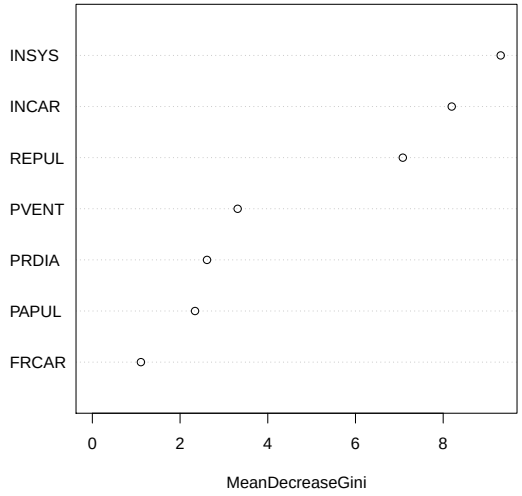


Variable Importance for Trees

Given some random forest with M trees, set $I(X_k) = \frac{1}{M} \sum_m \sum_t \frac{N_t}{N} \Delta i(t)$

where the first sum is over all trees, and the second one is over all nodes where the split is done based on variable X_k.

```
> RF=randomForest(PRONO ~ .,data = myocarde)
> varImpPlot(RF,main="")
> importance(RF)
      MeanDecreaseGini
FRCAR         1.107222
INCAR         8.194572
INSYS         9.311138
PRDIA         2.614261
PAPUL         2.341335
PVENT         3.313113
REPUL         7.078838
```

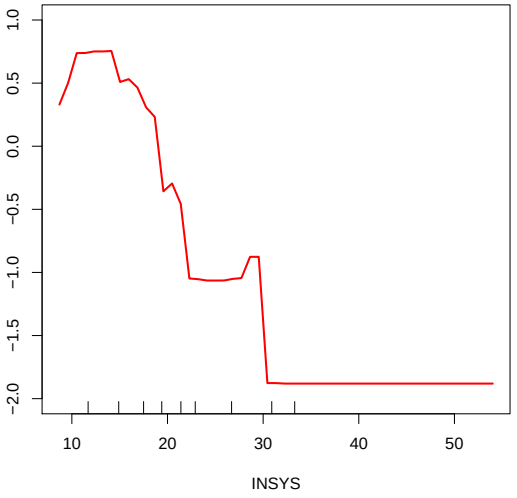


Partial Response Plots

One can also compute Partial Response Plots,

$$x \mapsto \frac{1}{n}\sum_{i=1}^{n} \widehat{\mathbb{E}}[Y|X_k = x, \boldsymbol{X}_{i,(k)} = \boldsymbol{x}_{i,(k)}]$$

```
> importanceOrder <- order(-RF$importance)
> names <- rownames(RF$importance)[importanceOrder
    ]
> for (name in names)
+ partialPlot(RF, myocarde, eval(name), col="red",
     main="", xlab=name)
```

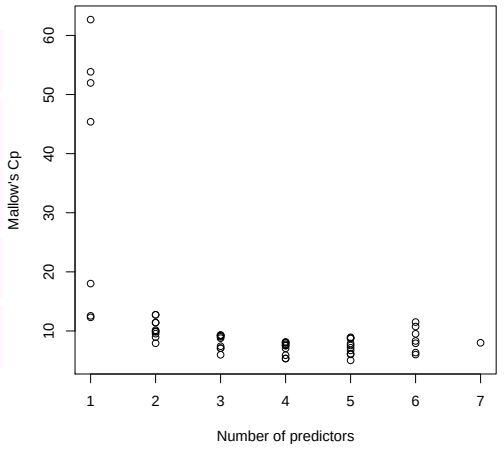


Feature Selection

Use Mallow's C_p, from Mallow (1974) on all subset of predictors, in a regression

$$C_p = \frac{1}{S^2}\sum_{i=1}^{n}[Y_i - \widehat{Y}_i]^2 - n + 2p,$$

```
> library(leaps)
> y <- as.numeric(train_myocarde$PRONO)
> x <- data.frame(train_myocarde[,-8])
> selec = leaps(x, y, method="Cp")
> plot(selec$size-1, selec$Cp)
```

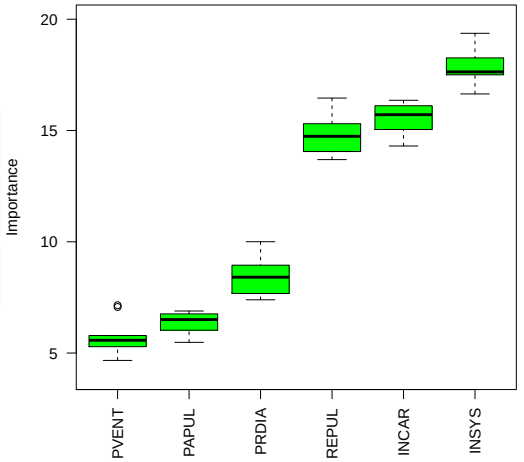


Feature Selection

Use random forest algorithm, removing some features at each iterations (the less relevent ones).

The algorithm uses shadow attributes (obtained from existing features by shuffling the values).

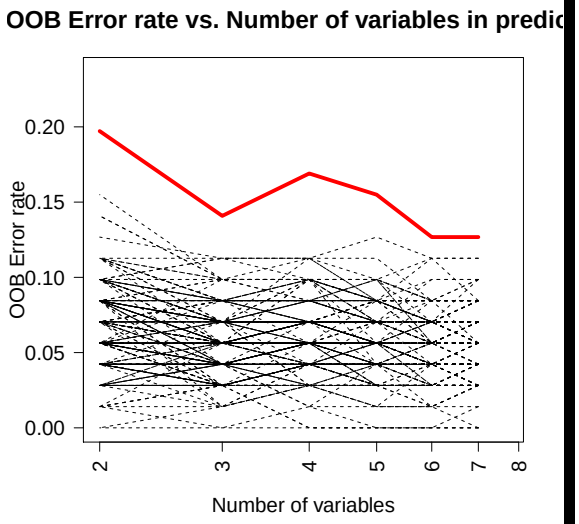
```
> library(Boruta)
> B <- Boruta(PRONO ~ ., data=myocarde, ntree=500)
> plot(B)
```



Feature Selection

Use random forests, and variable importance plots

```
> library(varSelRFBoot)
> X <- as.matrix(myocarde[,1:7])
> Y <- as.factor(myocarde$PRONO)
> library(randomForest)
> rf <- randomForest(X, Y, ntree = 200, importance
    = TRUE)
> V <- randomVarImpsRF(X, Y, rf, usingCluster =
  FALSE)
> VB <- varSelRFBoot(X, Y,usingCluster = FALSE)
> plot(VB)
```

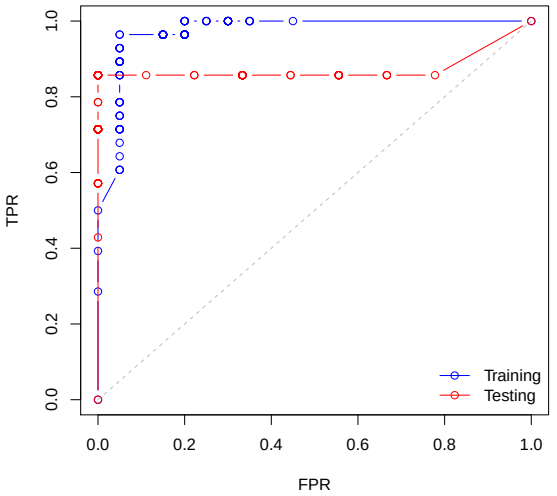


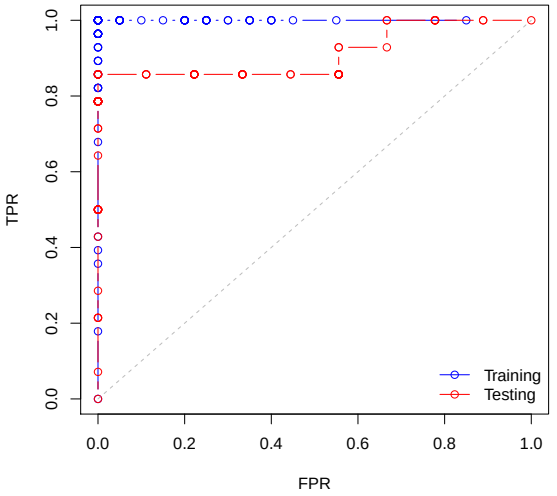
ROC (and beyond)

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

Comparing Classifiers: ROC Curves

```
> library(randomForest)
> fit=randomForest(PRONO~.,data=train_myocarde)
> train_Y=(train_myocarde$PRONO=="Survival")
> test_Y =(test_myocarde$PRONO=="Survival")
> train_S=predict(fit,type="prob",newdata=train
    _myocarde)[,2]
> test_S=predict(fit,type="prob",newdata=test_
    myocarde)[,2]
> vp=seq(0,1,length=101)
> roc_train=t(Vectorize(function(u) roc.curve(
    train_Y,train_S,s=u))(vp))
> roc_test=t(Vectorize(function(u) roc.curve(
    test_Y,test_S,s=u))(vp))
> plot(roc_train,type="b",col="blue",xlim=0:1,
    ylim=0:1)
```





Comparing Classifiers: ROC Curves

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, see Swets, Dawes & Monahan (2000)

Many other quantities can be computed, see

```
> library(hmeasures)
> HMeasure(Y,S)$metrics[,1:5]
Class labels have been switched from (Death,Survival) 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), Gini and AUC, as well as the area under the convex hull (AUCH).

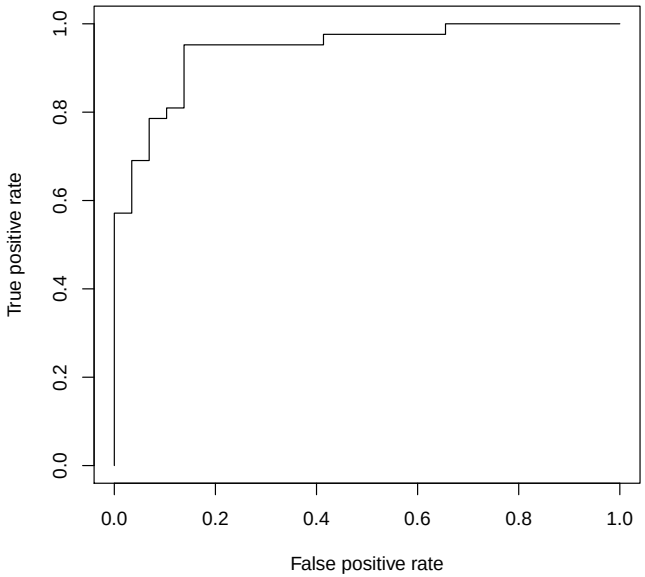
Comparing Classifiers: ROC Curves

Consider our previous logistic regression (on heart attacks)

```
> logistic <- glm(PRONO~. , data=myocarde,
    family=binomial)
> Y <- myocarde$PRONO
> S <- predict(logistic, type="response")
```

For a standard ROC curve

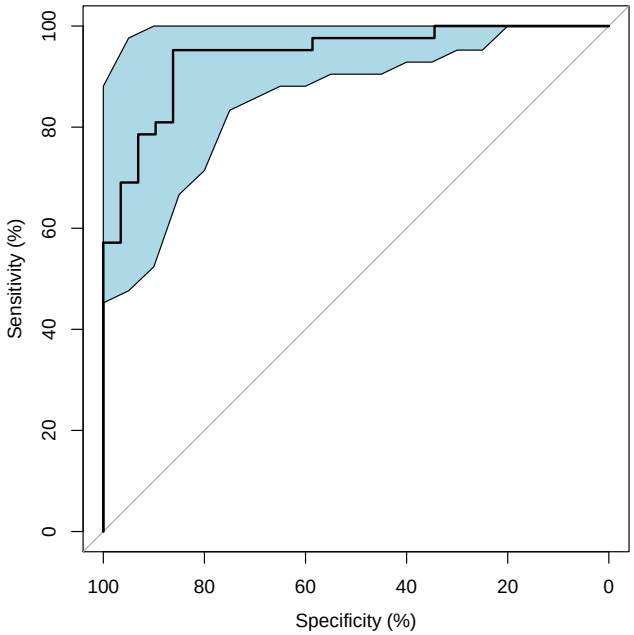
```
> library(ROCR)
> pred <- prediction(S,Y)
> perf <- performance(pred, "tpr", "fpr")
> plot(perf)
```



Comparing Classifiers: ROC Curves

On can get econfidence 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)
```

Comparing Classifiers: Accuracy and Kappa

Kappa statistic κ compares an Observed Accuracy with an Expected Accuracy (random chance), see Landis & Koch (1977).

	$Y=0$	$Y=1$	
$\widehat{Y}=0$	TN	FN	TN+FN
$\widehat{Y}=1$	FP	TP	FP+TP
	TN+FP	FN+TP	n

See also Obsersed and Random Confusion Tables

	$Y=0$	$Y=1$	
$\widehat{Y}=0$	25	3	28
$\widehat{Y}=1$	4	39	43
	29	42	71

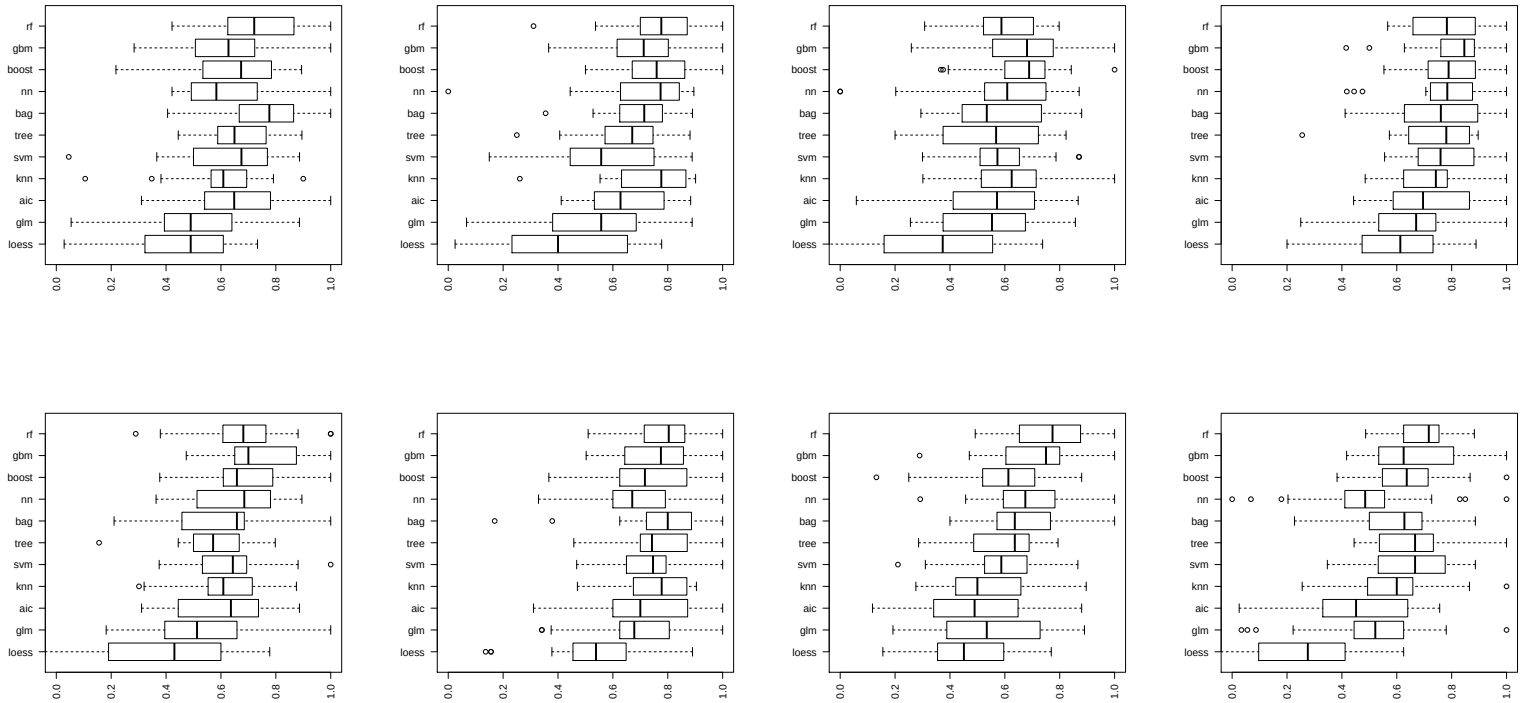
	$Y=0$	$Y=1$	
$\widehat{Y}=0$	11.44	16.56	28
$\widehat{Y}=1$	17.56	25.44	43
	29	42	71

$$\text{total accuracy} = \frac{TP+TN}{n} \sim 90.14\%$$

$$\text{random accuracy} = \frac{[TN+FP]\cdot[TP+FN]+[TP+FP]\cdot[TN+FN]}{n^2} \sim 51.93\%$$

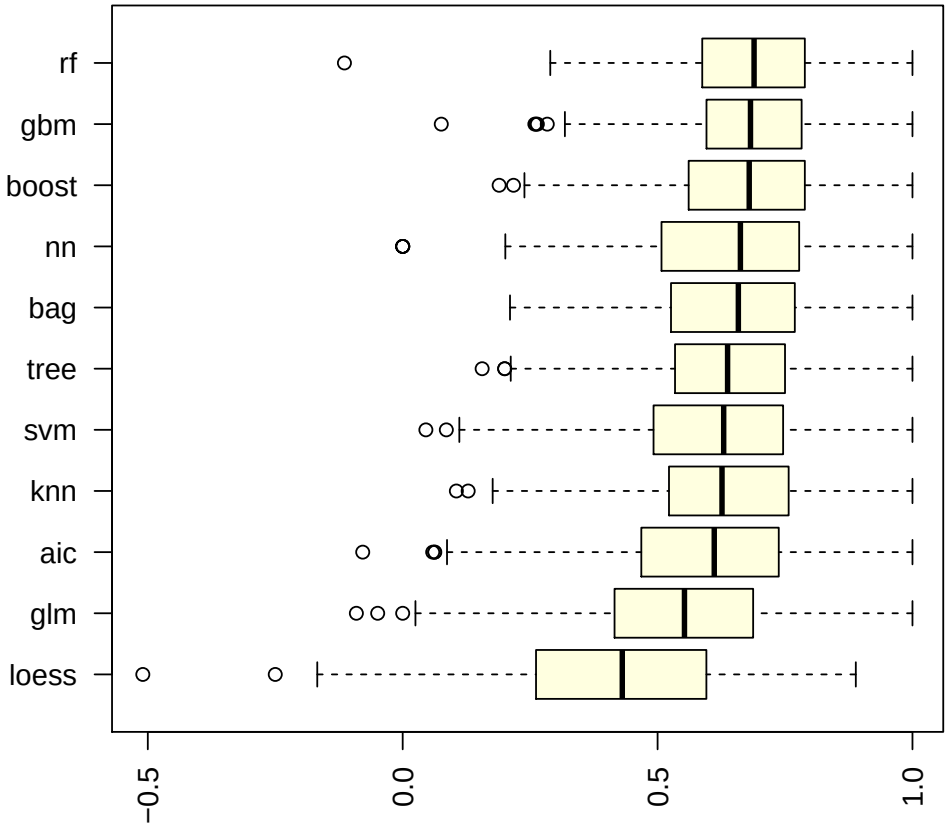
$$\kappa = \frac{\text{total accuracy} - \text{random accuracy}}{1 - \text{random accuracy}} \sim 79.48\%$$

Comparing Models on the myocarde Dataset



Comparing Models on the myocarde Dataset

If we average over all training samples

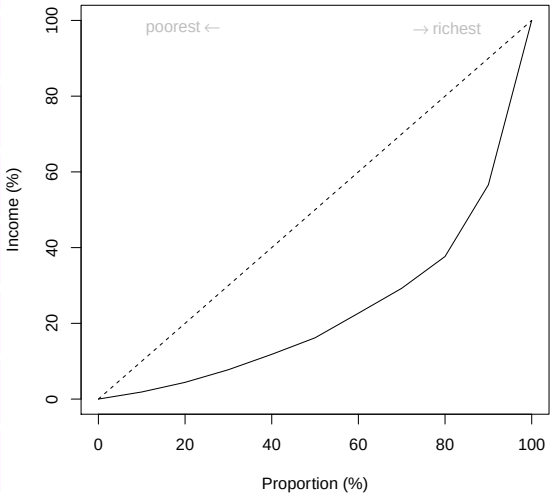


Gini and Lorenz Type Curves

Consider an ordered sample $\{y_1, \cdots, y_n\}$ of incomes, with $y_1 \leq y_2 \leq \cdots \leq y_n$, then Lorenz curve is

$$\{F_i, L_i\} \text{ with } F_i = \frac{i}{n} \text{ and } L_i = \frac{\sum_{j=1}^{i} y_j}{\sum_{j=1}^{n} y_j}$$

```
> L <- function(u,vary="income"){
+   base=base[order(base[,vary],decreasing=FALSE),]
+   base$cum=(1:nrow(base))/nrow(base)
+   return(sum(base[base$cum<=u,vary])/
+            sum(base[,vary]))}
> vu <- seq(0,1,length=nrow(base)+1)
> vv <- Vectorize(function(u) L(u))(vu)
> plot(vu, vv, type = "l")
```



Gini and Lorenz Type Curves

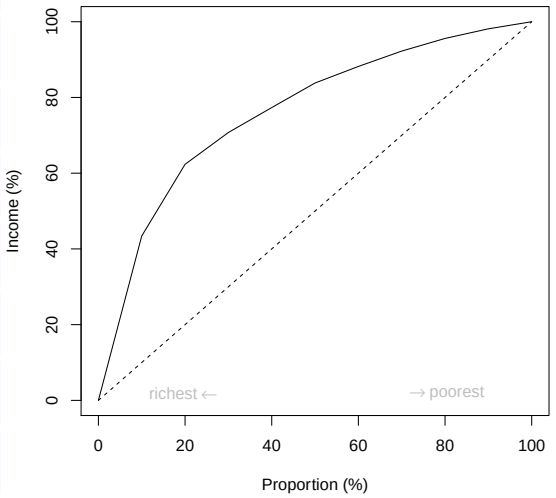
The theoretical curve, given a distribution F, is

$$u \mapsto L(u) = \frac{\int_{-\infty}^{F^{-1}(u)} t dF(t)}{\int_{-\infty}^{+\infty} t dF(t)}$$

see Gastwirth (1972).

One can also sort them from high to low incomes, $y_1 \geq y_2 \geq \cdots \geq y_n$

```
> L <- function(u,vary="income"){
+   base=base[order(base[,vary],decreasing=TRUE),]
+   base$cum=(1:nrow(base))/nrow(base)
+   return(sum(base[base$cum<=u,vary])/
+            sum(base[,vary]))}
> vu <- seq(0,1,length=nrow(base)+1)
> vv <- Vectorize(function(u) L(u))(vu)
> plot(vu, vv, type = "l")
```



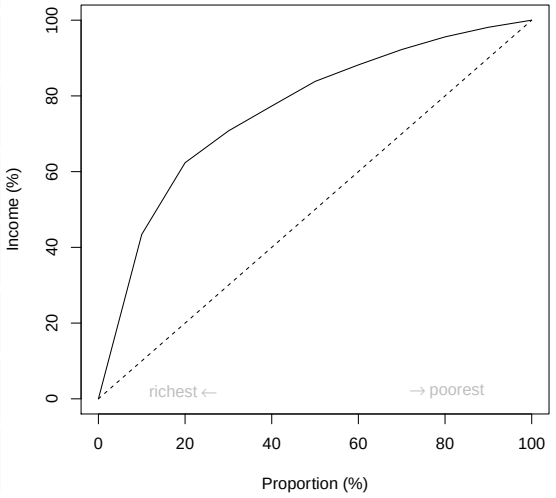
Gini and Lorenz Type Curves

We want to compare two regression models, $\widehat{m}_1(\cdot)$ and $\widehat{m}_2(\cdot)$, in the context of insurance pricing, see Frees, Meyers & Cummins (2014). We have observed losses y_i and premiums $\widehat{m}(\boldsymbol{x}_i)$. Consider an ordered sample by the model,

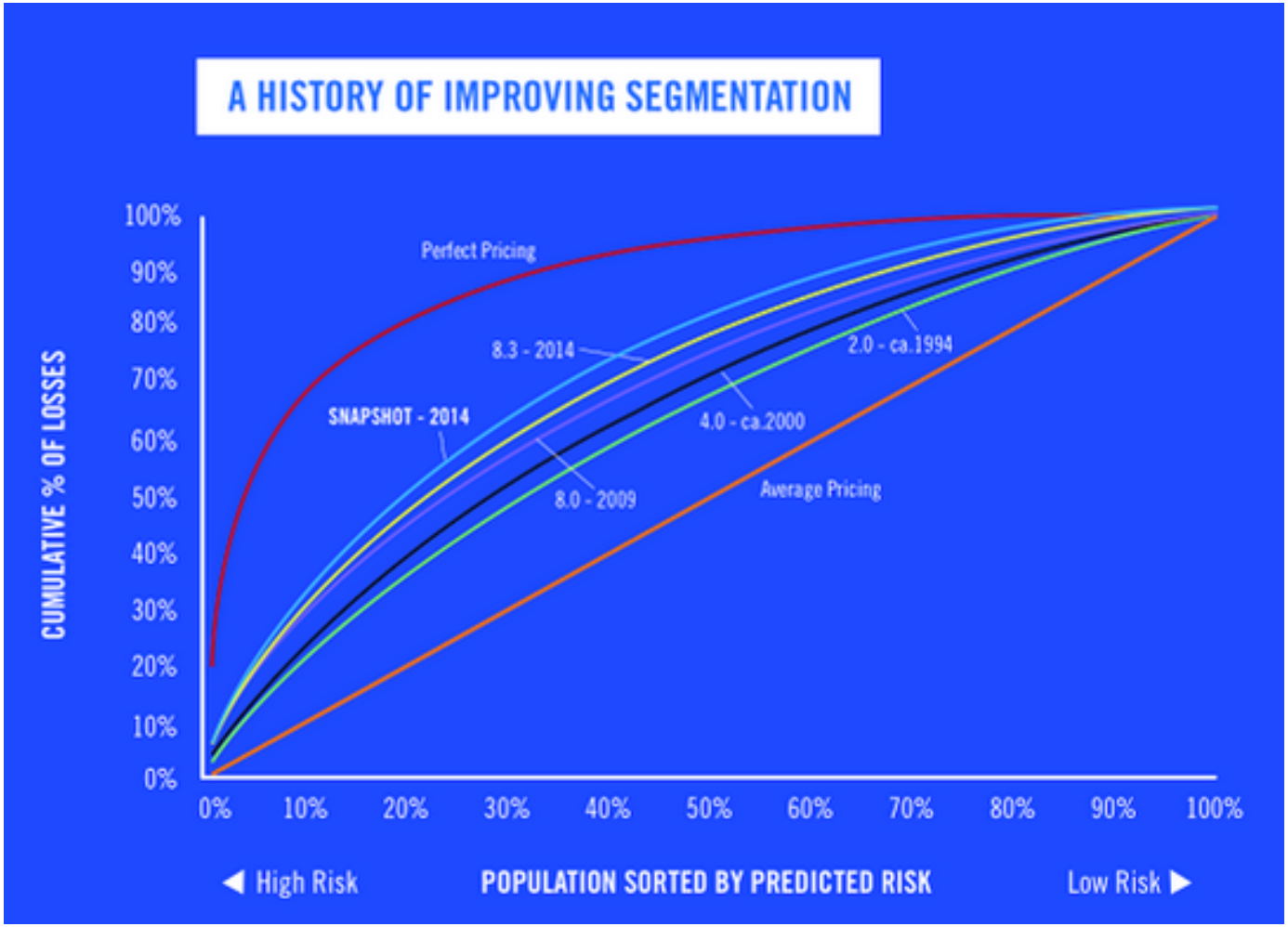
$$\widehat{m}(\boldsymbol{x}_1) \geq \widehat{m}(\boldsymbol{x}_2) \geq \cdots \geq \widehat{m}(\boldsymbol{x}_n)$$

then plot $\{F_i, L_i\}$ with $F_i = \dfrac{i}{n}$ and $L_i = \dfrac{\sum_{j=1}^{i} y_j}{\sum_{j=1}^{n} y_j}$

```
> L <- function(u,varx="premium",vary="losses"){
+   base=base[order(base[,varx],decreasing=TRUE),]
+   base$cum=(1:nrow(base))/nrow(base)
+   return(sum(base[base$cum<=u,vary])/
+            sum(base[,vary]))}
> vu <- seq(0,1,length=nrow(base)+1)
> vv <- Vectorize(function(u) L(u))(vu)
> plot(vu, vv, type = "l")
```



Gini and Lorenz Type Curves



See Frees *et al.* (2010) or Tevet (2013).

Model Selection, or Aggregation?

We have k models, $\widehat{m}_1(\boldsymbol{x}), \cdots, \widehat{m}_k(\boldsymbol{x})$ for the same y-variable, that can be trees, vsm, regression, etc.

Instead of selecting the best model, why not consider

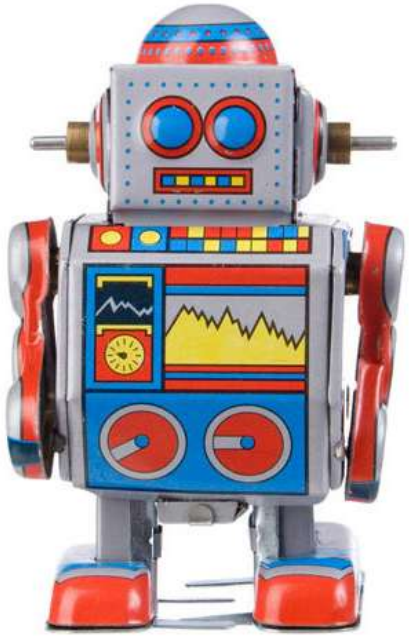
$$\widehat{m}^\star(\boldsymbol{x}) = \sum_{\kappa=1}^{k} \omega_\kappa \widehat{m}_\kappa(\boldsymbol{x})$$

for some weights ω_κ.

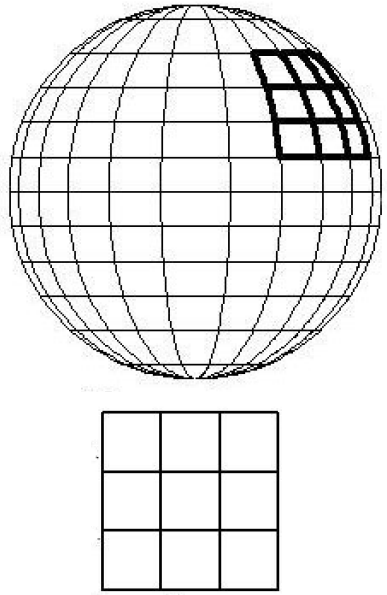
New problem: solve $\min_{\omega_1,\cdots,\omega_k} \left\{ \sum_{i=1}^{n} \ell\left(y_i - \sum_{\kappa=1}^{k} \omega_\kappa \widehat{m}_\kappa(\boldsymbol{x}_i) \right) \right\}$

Note that it might be interesting to regularize, by adding a Lasso-type penalty term, based on $\lambda \sum_{\kappa=1}^{k} |\omega_\kappa|$

Part 4.
Visualization and Maps



Projection(s)



"*When people thought the earth was flat, they were wrong. When people thought the earth was spherical, they were wrong. But if you think that thinking the earth is spherical is just as wrong as thinking the earth is flat, then your view is wronger than both of them put together.*" Isaac Asimov, cited in kjordahl.github.io

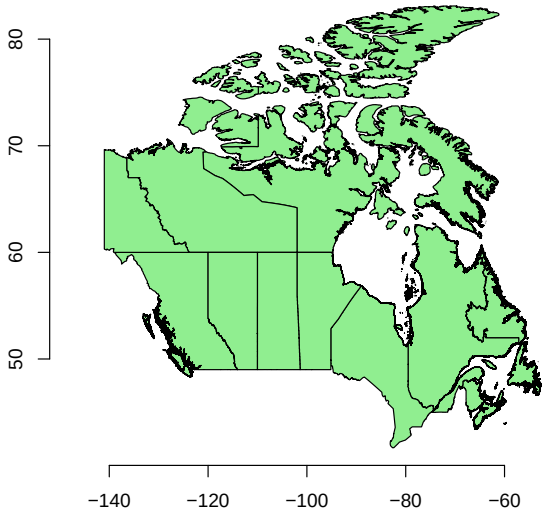
Need some Spatial Reference Systems

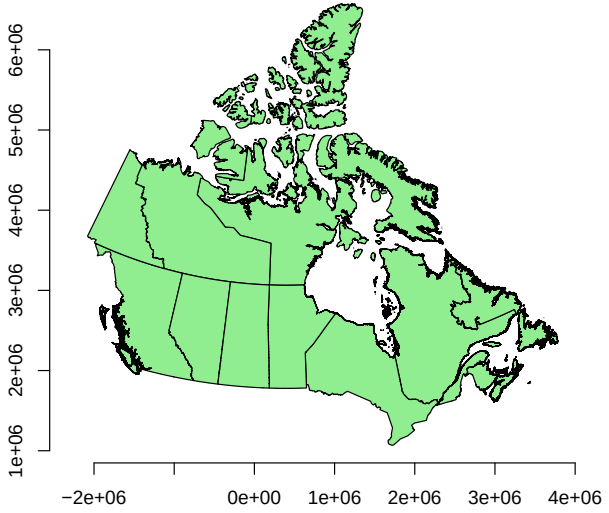
E.g. EPSG:4326 latitude, longitude in WGS-84 coordinate system

EPSG:900913 and EPSG:3857: Google spherical Mercator

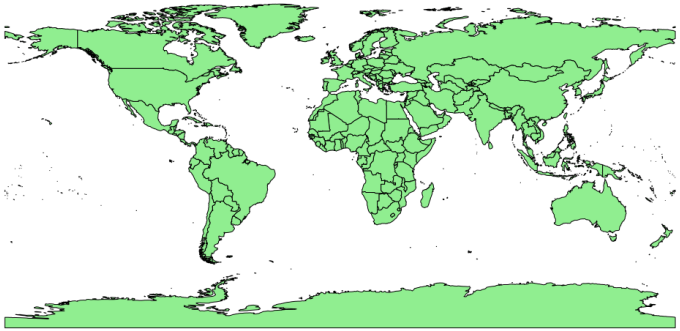
ESRI:102718: and NAD 1983 State Plane New York Long Island

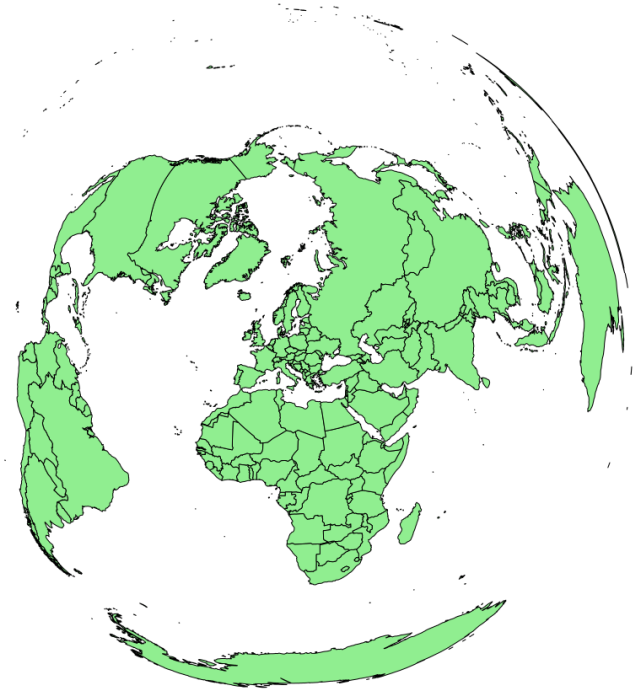
```
1 library(rgdal)
2 library(maptools)
3 Proj <- CRS("+proj=longlat +datum=WGS84")
4 CAN_shp <- readShapePoly("CAN_adm1.shp",
      verbose=TRUE, proj4string=Proj)
5 plot(CAN_shp)
6
7 new_CAN_shp <- spTransform(CAN_shp, CRS("+
      init=epsg:26978"))
8 plot(new_CAN_shp)
```



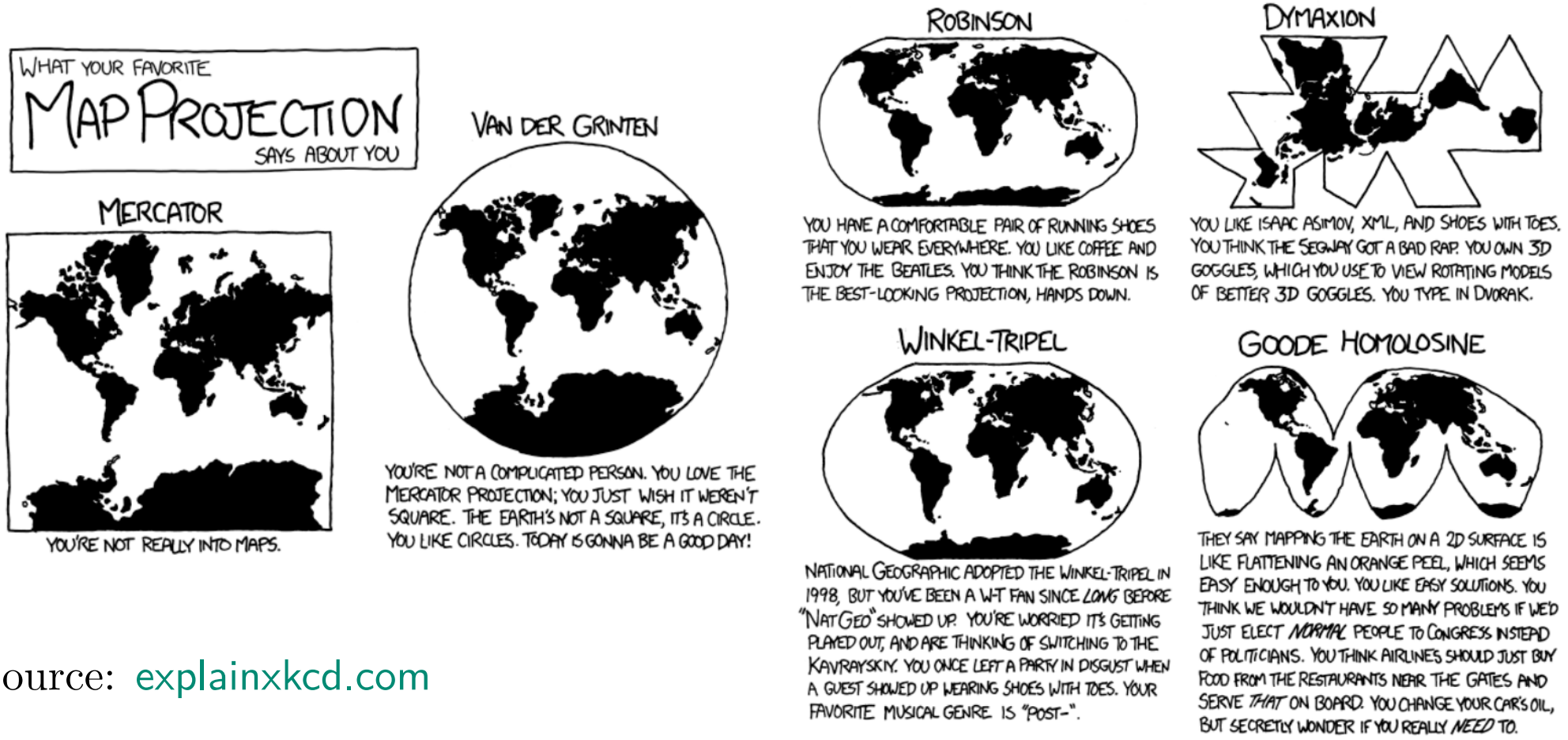


```r
library(sp)
library(rworldmap)
data(countriesLow)

crs.WGS84 <- CRS("+proj=longlat +datum=WGS84")
countries.WGS84 <- spTransform(countriesLow, crs.WGS84)
plot(countries.WGS84,col="light green")

crs.laea <- CRS("+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +units=m +no_defs")
countries.laea <- spTransform(countriesLow, crs.laea)
plot(countries.laea,col="light green")
```





Projection(s)



Source: explainxkcd.com

Maps and Polygons

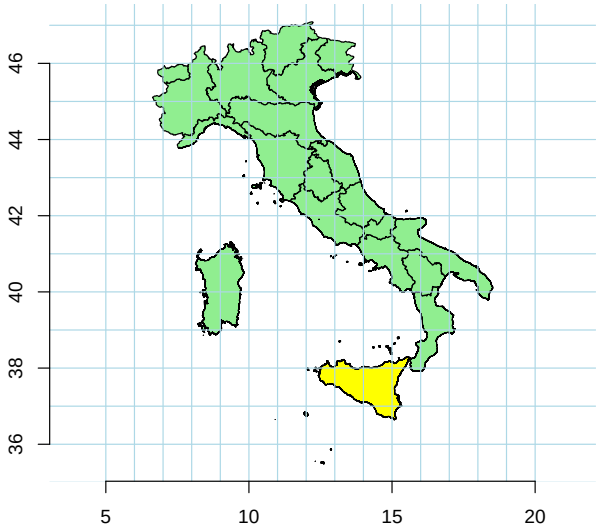
```
1 > library(rgdal)
2 > ita1<-readOGR(dsn="/Italy",layer="ITA_adm1
      ",verbose=FALSE)
3 > plot(ita1,col="light green")
```

Maps and Polygons

```
> ita1@data[,"NAME_1"][1:6]
[1]   Abruzzo   Apulia   Basilicata   Calabria
[5] Campania Emilia-Romagna
> pos<-which(ita1@data[,"NAME_1"]=="Sicily")
> Poly_Sicily<-ita1[pos,]
> plot(ita1,col="light green")
> plot(Poly_Sicily,col="yellow",add=TRUE)
```

Maps and Polygons

```
> plot(ita1,col="light green")
> plot(Poly_Sicily,col="yellow",add=TRUE)
> abline(v=5:20,col="light blue")
> abline(h=35:50,col="light blue")
> axis(1)
> axis(2)
```

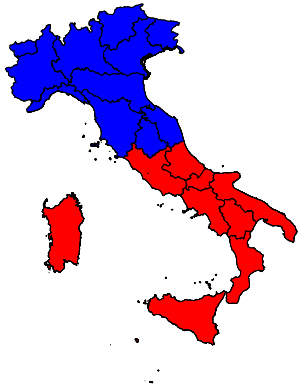


Maps and Polygons

```
> pos<-which(ita1@data[,"NAME_1"] %in% c("
    Abruzzo","Apulia","Basilicata","Calabria
    ","Campania","Lazio","Molise","Sardegna"
    ,"Sicily"))
> ita_south <- ita1[pos,]
> ita_north <- ita1[-pos,]
> plot(ita1)
> plot(ita_south,col="red",add=TRUE)
> plot(ita_north,col="blue",add=TRUE)
```

Maps and Polygons

```
> library(xlsx)
> data_codes<-read.xlsx2(file = "\seminar/
    spatial/Italy_codes.xlsx",sheetName="
    ITALY", startRow=1, header=TRUE)
> names(data_codes)[1]="NAME_1"
> ita2<-merge(ita1, data_codes, by = "NAME_1
    ", all.x=TRUE)
> pos<-which(ita2@data[,"GROUP"] == "SOUTH")
```



Maps and Polygons

Here we have only used colors, but it is also possible to merge the polygons together,

```
> library(rgeos)
> ita_s<-gUnionCascaded(ita_south)
> ita_n<-gUnionCascaded(ita_north)
> plot(ita1)
> plot(ita_s,col="red",add=TRUE)
> plot(ita_n,col="blue",add=TRUE)
```

Maps and Polygons

On polygons, it is also possible to visualize centroids,

```
> plot(ita1,col="light green")
> plot(Poly_Sicily,col="yellow",add=TRUE)
> gCentroid(Poly_Sicily,byid=TRUE)
SpatialPoints:
          x        y
14 14.14668 37.58842
Coordinate Reference System (CRS) arguments:
+proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs +towgs84=0,0,0
> points(gCentroid(Poly_Sicily,byid=TRUE),
    pch=19,col="red")
```

Maps and Polygons

or

```
> G <- as.data.frame(gCentroid(ita1,byid=
    TRUE))
> plot(ita1,col="light green")
> text(G$x,G$y,1:20)
```

Maps and Polygons

Consider two trajectories, caracterized by a series of knots (from the centroïds list)

```
> c1 <-G[c(17,20,6,16,8,12,2),]
> c2 <- G[c(13,11,1,5,3,4),]
```

We can convert those segments into SpatialLines

```
L1<-SpatialLines(list(Lines(list(Line(c1)),"Line1")))
L2<-SpatialLines(list(Lines(list(Line(c2)),"Line2")))
```

To make sure that we can add those lines on the map, use

```
L1@proj4string<-ita1@proj4string
L2@proj4string<-ita1@proj4string
> cross_road <-gIntersection(L1,L2)
> cross_road@coords
SpatialPoints:
          x         y
1 11.06947 44.03287
1 14.20034 41.74879
Coordinate Reference System (CRS) arguments:
+proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs +towgs84=0,0,0
> plot(ita1,col="light green")
> plot(L1,col="red",lwd=2,add=TRUE)
> plot(L2,col="blue",lwd=2,add=TRUE)
> plot(cross_road,pch=19,cex=1.5,add=TRUE)
```

Maps and Polygons

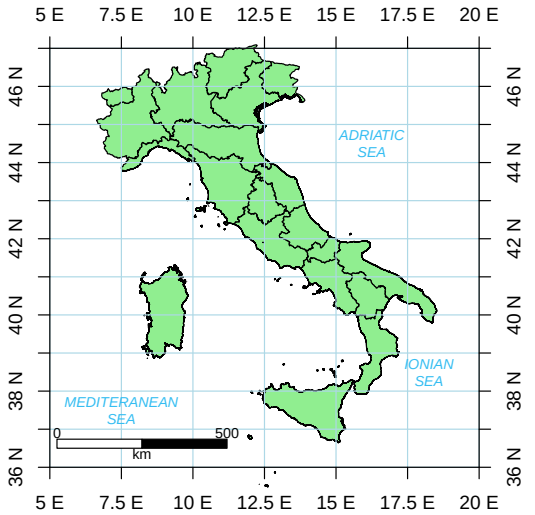
To add elements on maps, consider, e.g.

```
plot(ita1,col="light green")
grat <- border_plot(sp=ita1,WE=seq(5,20,by
    =2.5),NS=seq(36,47))
plot(grat,col="light blue",add=TRUE)
labs<-paste(seq(5,20,by=2.5)," E",sep="")
axis(side=1,pos=36,at=seq(5,20,by=2.5),
labels=labs)
axis(side=3,pos=47,at=seq(5,20,by=2.5),
labels=labs)
labs<-paste(seq(36,47)," N",sep="")
axis(side=2,pos=5,at=seq(36,47),
labels=labs)
axis(side=4,pos=20,at=seq(36,47),
labels=labs)
```

Maps and Polygons

To add elements on maps, consider, e.g.

```
1 sea<-data.frame(Name=c("MEDITERANEAN\nSEA","
    ADRIATIC\nSEA","IONIAN\nSEA"),
2 Longitude=c(7.5,16.25,18.25),Latitude=c
    (37.5,44.5,38.5))
3 text(sea$Longitude,sea$Latitude,sea$Name,
4 cex=0.75,col="#34bdf2",font=3)
5 library(raster)
6 scalebar(d=500,xy=c(5.25,36.5),,type="bar",
    below="km",
7 lwd=4,divs=2,col="black",cex=0.75,lonlat=
    TRUE)
```



Maps, with R

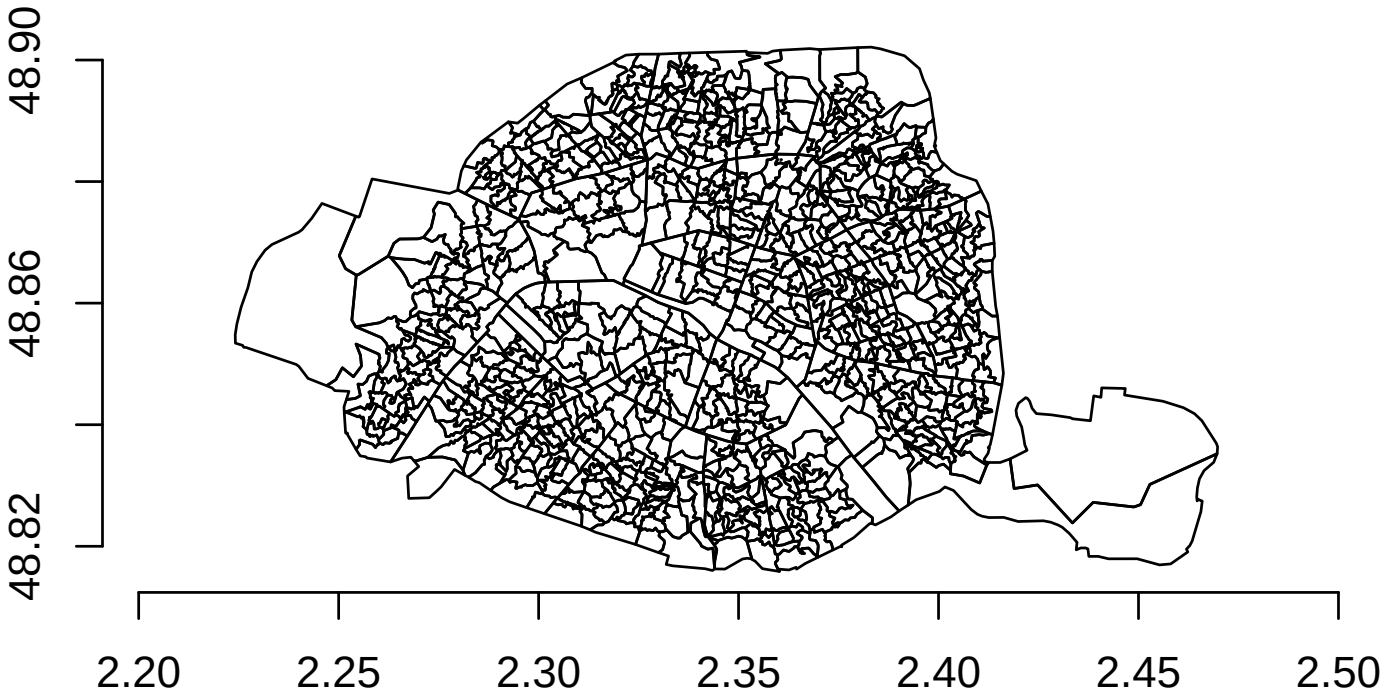
There are $\sim 36,552$ communes in France. Why not illustrate with some smaller datasets? Paris is splited in 850 IRIS areas (*Îlot Regroupé pour des Indicateurs Statistiques* see insee.fr).

```
library(RColorBrewer)
plotclr <- brewer.pal(7,"RdYlBu")[c(7,1)]
library(maptools)
library(rgdal)
library(classInt)
paris <- readShapeSpatial("paris-cartelec.shp")
proj4string(paris) <- CRS("+init=epsg:2154")
paris <- spTransform(paris, CRS("+proj=longlat +ellps=GRS80"))
```

Maps, with R

To visualize all those polygons, use

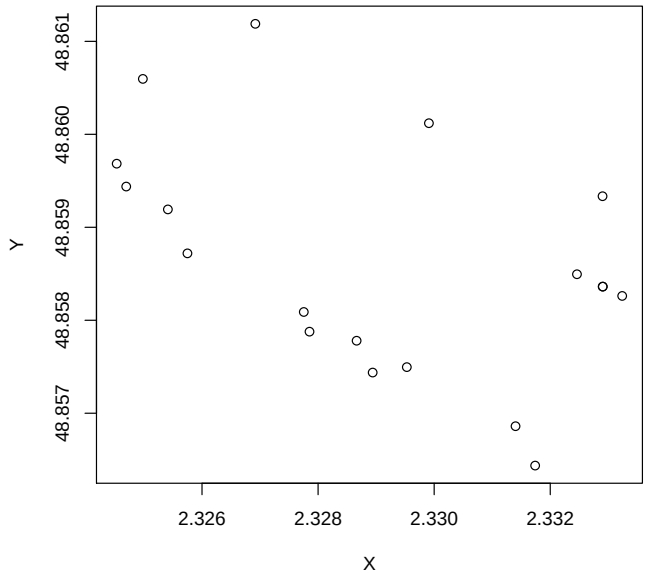
```
1 > plot(paris)
```



Maps, with R

Let us focus on one very specific polygon (the 100th)

```
> length(paris)
[1] 850
> poly_paris <- SpatialPolygons2PolySet(paris)
> sub_poly <- poly_paris[poly_paris$PID==100,]
> sub_poly
   PID SID POS        X        Y
1  100   1   1 2.332904 48.85836
2  100   1   2 2.333240 48.85826
3  100   1   3 2.331740 48.85644

18 100   1  18 2.332460 48.85850
19 100   1  19 2.332904 48.85836
> plot(sub_poly[,c("X","Y")])
> polygon(sub_poly[,c("X","Y")])
```



Maps, with R

Given some GPS coordinates, we want to know in which IRIS it is located

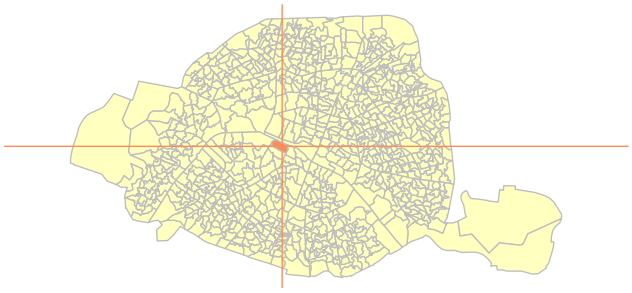
use function (for all possible polygon...it can be long)

```
> library(sp)
> point <- c(2.33,48.859)
> point.in.polygon(point[1],point[2],sub_poly[,"X"],sub_poly[,"Y"])
[1] 1
> point_in_i=function(i,point)
+ point.in.polygon(point[1],point[2],poly_paris[poly_paris$PID==i,"X"
    ],poly_paris[poly_paris$PID==i,"Y"])
> where_is_point=function(point)
+ which(Vectorize(function(i) point_in_i(i,point))(1:length(paris))
    >0)
```

Maps, with R

```
> where_is_point(point)
[1] 100
```

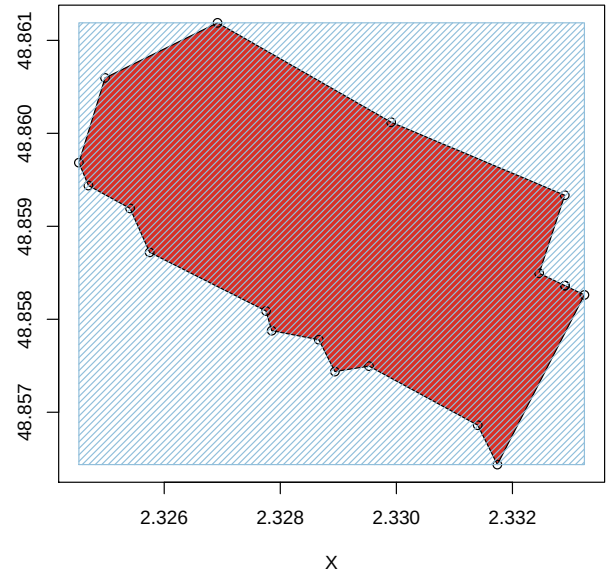
```
> library(RColorBrewer)
> plotclr <- brewer.pal(7,"RdYlBu")
> vizualize_point <- function(point){
+ wp <- where_is_point(point)
+ colcode <- rep(plotclr[4],length(paris))
+ colcode[wp] <- plotclr[1]
+ plot(paris,col=colcode,border="grey")}
> vizualize_point(point)
```



Maps, with R

simplify using some kind of grid, then use the previous function only on a few possible candidates

```
> minmax=function(i){
+ vect_xy=poly_paris[poly_paris$PID==i,c("X"
    ,"Y")]
+ box_xy=c(min(vect_xy[,"X"]),min(vect_xy[,"
    Y"]),
+ max(vect_xy[,"X"]),max(vect_xy[,"Y"]))
+ return(box_xy)}
> rect(minmax(100))
```



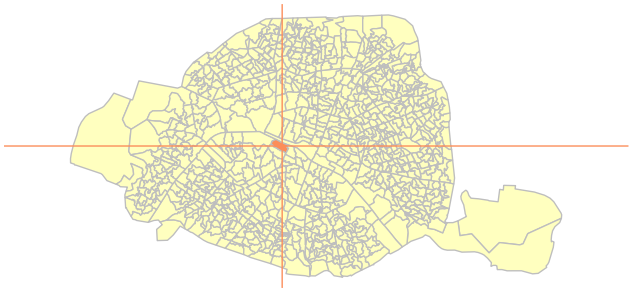
Maps, with R

Then simply use

```
> is.box=function(i,loc){
+ box_i=minmax(i)
+ x <- (loc[1]>=box_i[1])&(loc[1]<=box_i[3])
+ y <- (loc[2]>=box_i[2])&(loc[2]<=box_i[4])
+ return(x&y)}
```

which returns TRUE is some point is in the box around some polygon.

```
> point <- c(2.33,48.859)
> is.box(100,point)
[1] TRUE
```



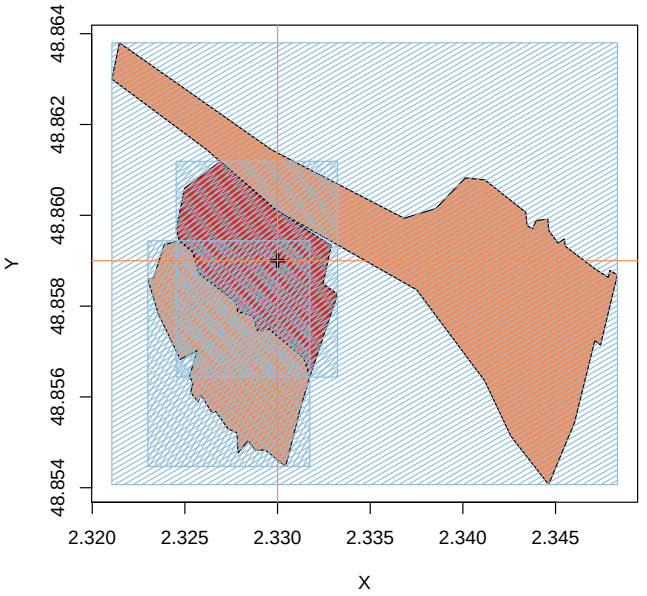
Maps, with R

Then to get all possible candidates use

```
> which.box <- function(loc){
+ which(Vectorize(function(i) is.box(i,loc))
    (1:length(paris)))}
> which.box(point)
[1]   1 100 101
```

see

```
> plot(sub_poly[,c("X","Y")],col="white")
> polygon(poly_paris[poly_paris$PID==1,c("X"
    ,"Y")],col=plotclr[2])
> polygon(poly_paris[poly_paris$PID==100,c("
    X","Y")],col=plotclr[1])
> polygon(poly_paris[poly_paris$PID==101,c("
    X","Y")],col=plotclr[2])
```



Maps, with R

Finally use

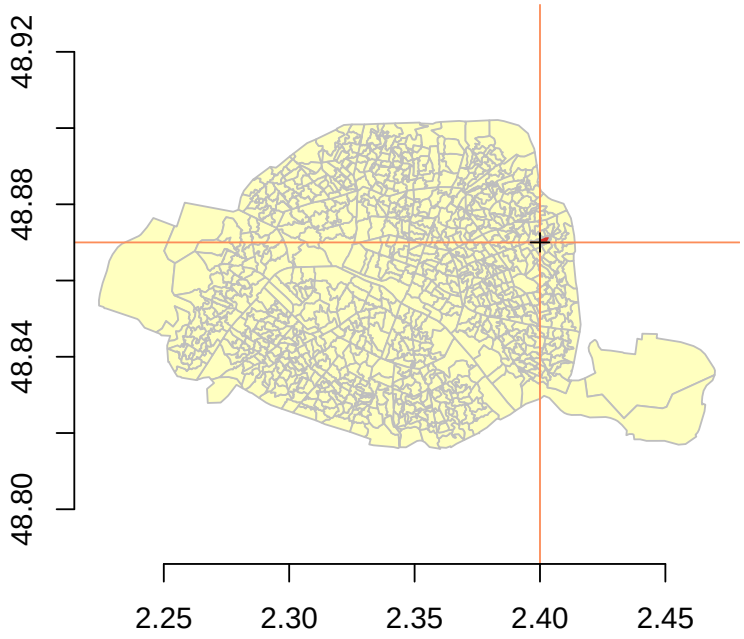
```
> which.poly <- function(point){
+ idx=which.box(point)
+ idx_valid=NULL
+ for(i in idx){
+ pip=point.in.polygon(point[1],point[2],
+ poly_paris[poly_paris$PID==i,"X"],
+ poly_paris[poly_paris$PID==i,"Y"])
+ if(pip>0) idx_valid=c(idx_valid,i)}
+ return(idx_valid)}
```

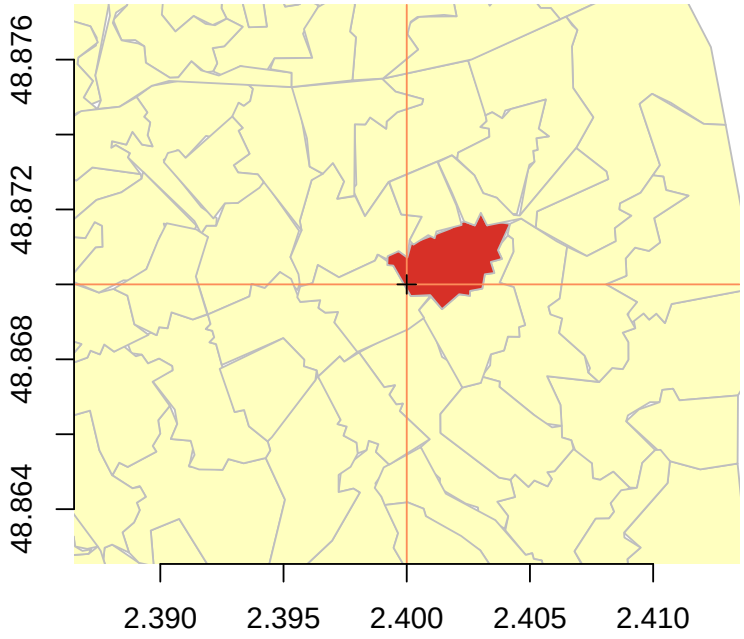
to identify the IRIS polygon

```
> which.poly(point)
[1] 100
```

Maps, with R

```
> vizualize_point <- function(point){
+ wp <- which.poly(point)
+ colcode <- rep(plotclr[4],length(paris))
+ colcode[wp] <- plotclr[1]
+ plot(paris,col=colcode,border="grey")}
> vizualize_point(c(2.4,48.87))
```





Google Maps and Open Street Map

```
> library(ggmap)
>  library(geosphere)
>  (MAIF <- geocode("niort 200 Avenue Allende"))
Information from URL : http://maps.googleapis.com/maps/api/geocode/
    json?address=niort%20200%20Avenue%20Allende&sensor=false
         lon      lat
1 -0.4864266 46.33218
>  (Rennes <- geocode("rennes place hoche"))
Information from URL : http://maps.googleapis.com/maps/api/geocode/
    json?address=rennes%20place%20hoche&sensor=false
       lon      lat
1 -1.67694 48.11526
```

The distance, in km, is obtained using the Haversine formula wikipedia.org

```
> distHaversine(MAIF, Rennes, r=6378.137)
[1] 217.9371
```

(in km.) while the driving distance is

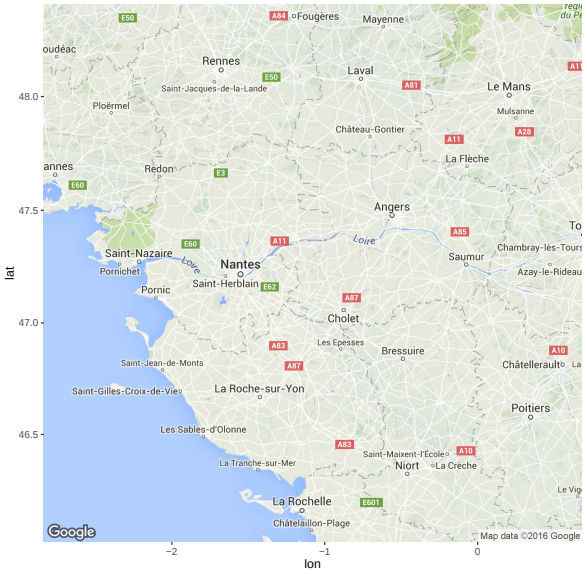
```
> mapdist(as.numeric(MAIF),as.numeric(Rennes), mode = 'driving')
by using this function you are agreeing to the terms at :
http://code.google.com/apis/maps/documentation/distancematrix/

                                                    from
                          to       m       km    miles seconds
1 200 Avenue Salvador Allende, 79000 Niort, France 10-11 Place Hoche,
     35000 Rennes, France 257002 257.002 159.701    9591
  minutes    hours
1  159.85 2.664167
```

Visualizing Maps, via Google Maps

```
1    Loc=data.frame(rbind(MAIF, Rennes))
2  > library(ggmap)
3  > library(RgoogleMaps)
4  > CenterOfMap <- as.list(apply(Loc,2,mean))
5  > W <- get_map(c(lon=CenterOfMap$lon, lat=
       CenterOfMap$lat),zoom = 8, maptype = "
       terrain", source = "google")
6  > WMap <- ggmap(W)
7  > WMap
```

or

```
1  > W <- get_map(c(lon=CenterOfMap$lon, lat=
       CenterOfMap$lat),zoom = 8, maptype = "
       roadmap")
2  > ggmap(W)
```



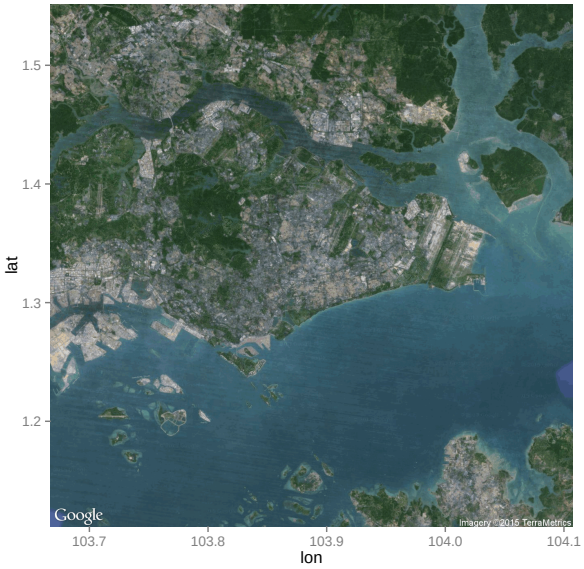
Visualizing Maps, via Google Maps

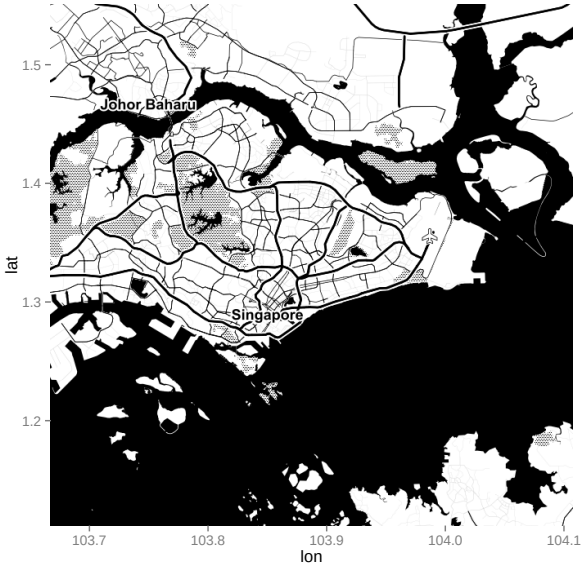
from a technical point of view, those are ggplot2 maps, see ggmapCheatsheet

```
> W <- get_map(c(lon=CenterOfMap$lon, lat=
    CenterOfMap$lat),zoom = 8, maptype = "
    satellite")
> ggmap(W)
```

or

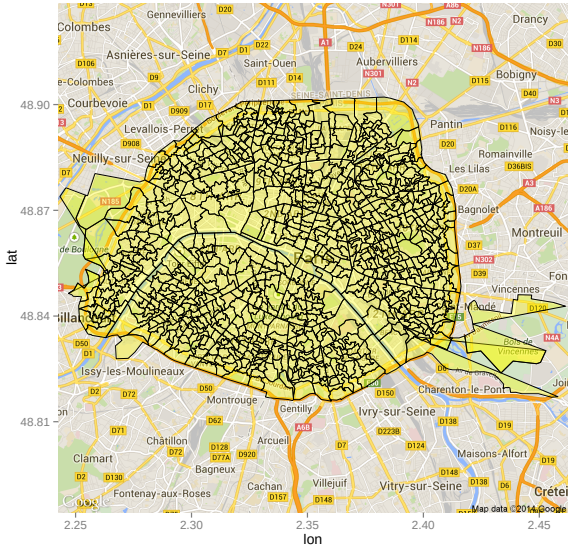
```
> W <- get_map(c(lon=CenterOfMap$lon, lat=
    CenterOfMap$lat),zoom = 8, maptype = "
    toner", source = "stamen")
> ggmap(W)
```

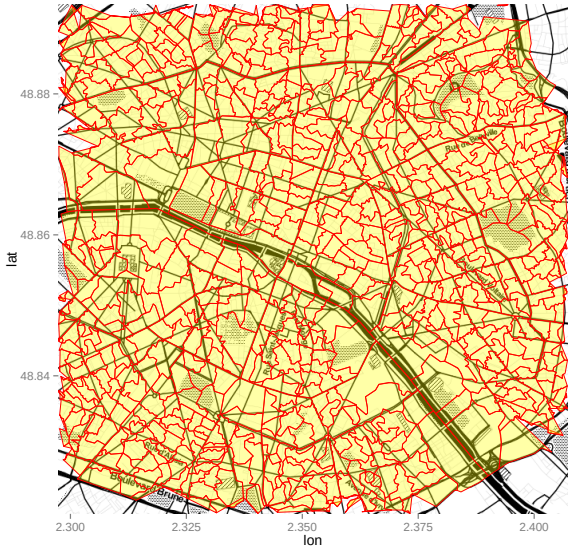




Visualizing Maps, via Google Maps

```
> Paris <- get_map("Paris",zoom = 12,
    maptype = "roadmap")
> ParisMap <- ggmap(Paris)
> library(maptools)
> library(rgdal)
> paris=readShapeSpatial("paris-cartelec.shp
    ")
> proj4string(paris) <- CRS("+init=epsg:2154
    ")
> paris <- spTransform(paris, CRS("+proj=
    longlat +datum=WGS84"))
> ParisMapPlus <- ParisMap + geom_polygon(
    aes(x=long, y=lat, group=group), fill='
    yellow', size=.2,color='black', data=
    paris, alpha=.35)
> ParisMapPlus
```





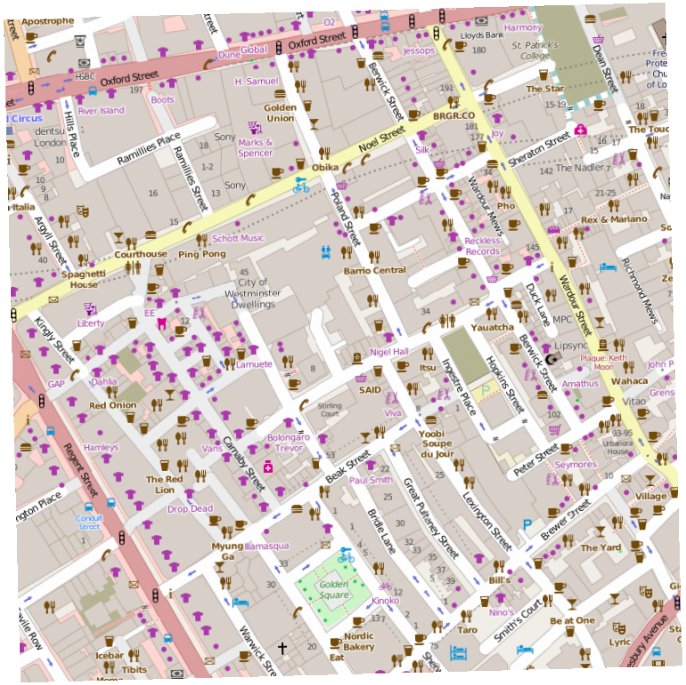
OpenStreet Map

```
> library(OpenStreetMap)
> map <- openmap(c(lat= 51.516,    lon=
    -.141),
+ c(lat= 51.511,    lon= -.133))
> map <- openproj(map, projection = "+init=
    epsg:27700")
> plot(map)
```

OpenStreet Map

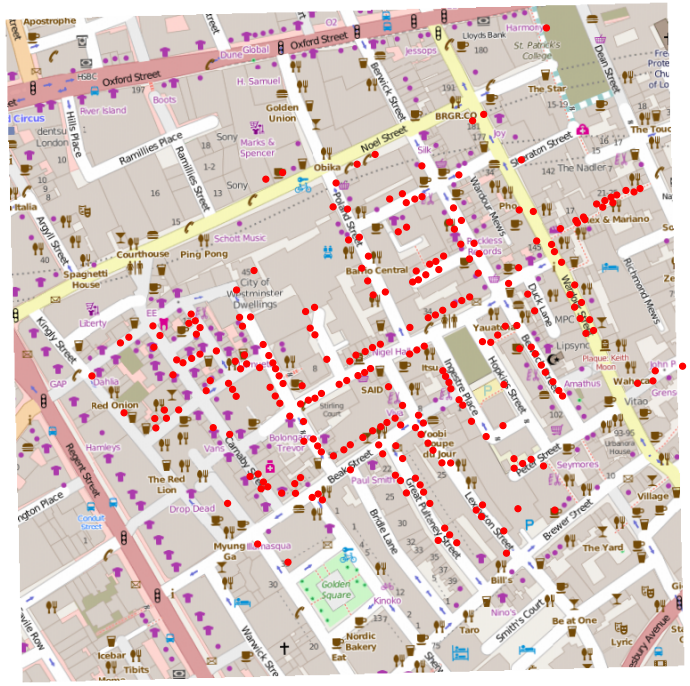
```
> library(OpenStreetMap)
> map <- openmap(c(lat= CenterOfMap$lat+.25,
+         lon= CenterOfMap$lon-.25),
+         c(lat= CenterOfMap$lat-.25,
+         lon= CenterOfMap$lon+.25))
> plot(map)
```

It is a standard R plot, we can add points on that graph.



OpenStreet Map

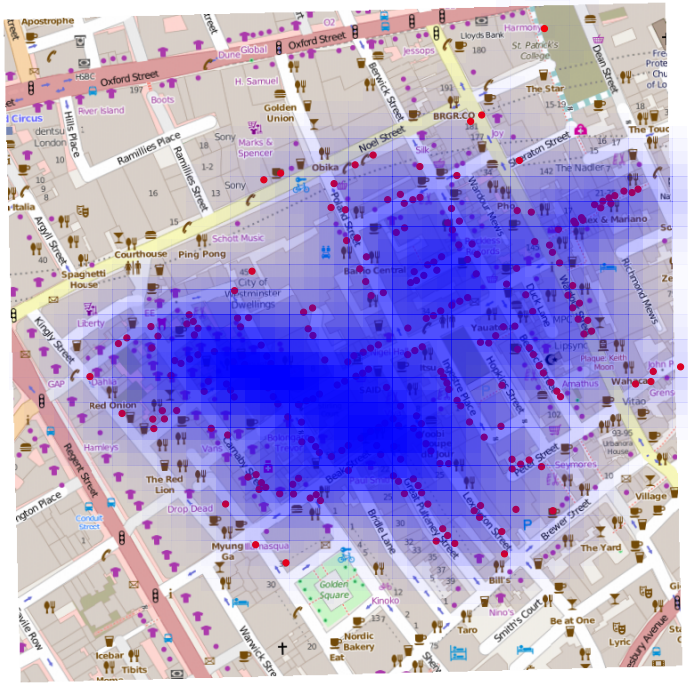
```
> library(maptools)
> deaths<-readShapePoints("Cholera_Deaths")
> head(deaths@coords)
  coords.x1 coords.x2
0  529308.7  181031.4
1  529312.2  181025.2
2  529314.4  181020.3
3  529317.4  181014.3
4  529320.7  181007.9
> points(deaths@coords,col="red", pch=19,
    cex=.7 )
```



Cholera outbreak, in London, 1854,
dataset collected by John (not Jon) Snow

OpenStreet Map

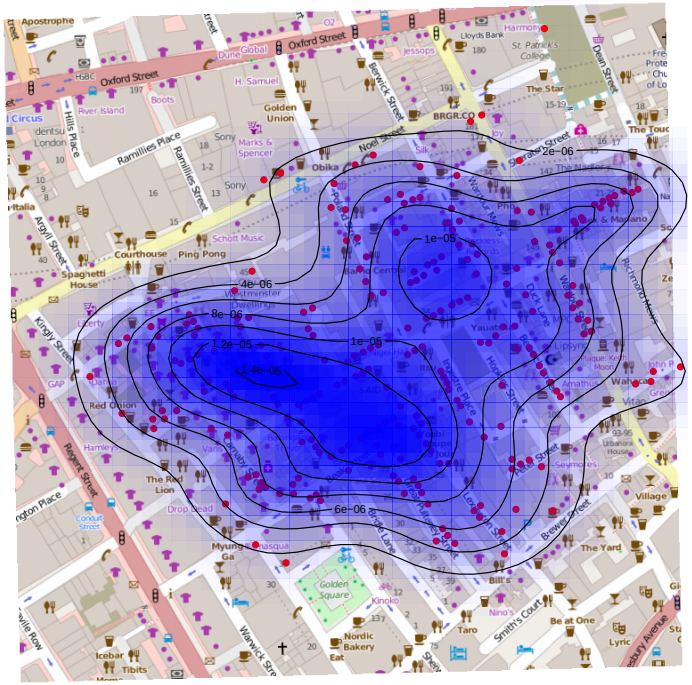
```
> X <- deaths@coords
> library(KernSmooth)
> kde2d <- bkde2D(X, bandwidth=c(bw.ucv(X
    [,1]),bw.ucv(X[,2])))
> library(grDevices)
> clrs <- colorRampPalette(c(rgb(0,0,1,0),
    rgb(0,0,1,1)), alpha = TRUE)(20)
> image(x=kde2d$x1, y=kde2d$x2,z=kde2d$fhat,
     add=TRUE,col=clrs)
```



OpenStreet Map

we can add a lot of things on that map

```
> contour(x=kde2d$x1, y=kde2d$x2,z=kde2d$
    fhat, add=TRUE)
```



OpenStreet Map

There are alternative packages related to OpenStreetMap. See for instance

```
> library(leafletR)
> devtools::install_github("rstudio/leaflet")
```

(see http://rstudio.github.io/leaflet/ for more information).

```
> library(osmar)
> src <- osmsource_api()
> loc.london <- c(-0.137,51.513)
> bb <- center_bbox(loc.london[1],loc.london[2], 800, 800)
> ua <- get_osm(bb, source = src)
```

OpenStreet Map

We can visualize buildings

```
> bg_ids <- find(ua, way(tags(k=="building")
    ))
> bg_ids <- find_down(ua, way(bg_ids))
> bg <- subset(ua, ids = bg_ids)
> bg_poly=as_sp(bg, "polygons")
> plot(bg_poly, col = gray.colors(12)[11],
    border="gray")
```

OpenStreet Map

We can visualize and leisure areas

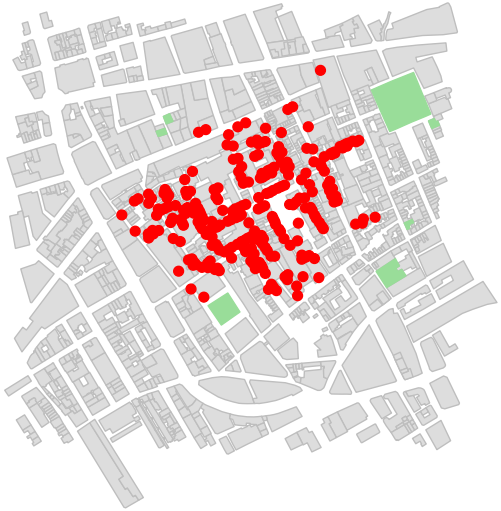
```
> nat_ids=find(ua, way(tags(k=="waterway")))
> nat_ids=find_down(ua, way(nat_ids))
> nat=subset(ua, ids = nat_ids)
> nat_poly=as_sp(nat, "polygons")
> nat_ids=find(ua, way(tags(k=="leisure")))
> nat_ids=find_down(ua, way(nat_ids))
> nat=subset(ua, ids = nat_ids)
> nat_poly=as_sp(nat, "polygons")
> plot(nat_poly, col = "#99dd99",add=TRUE,
    border="#99dd99")
```

(here we consider waterway and leisure tags)

OpenStreet Map

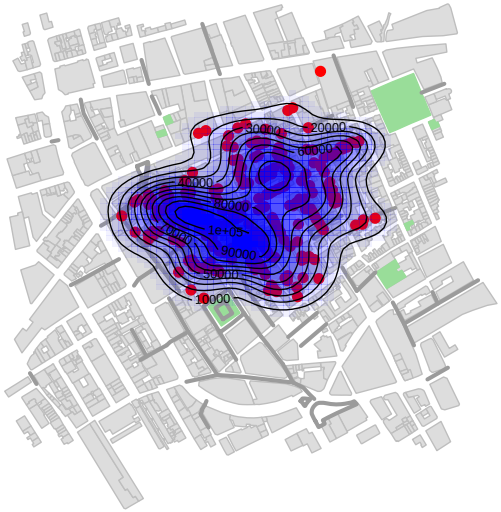
and finally, add the deaths

```
> points(df_deathscoords,col="red",pch=19)
```



and some heat map

```
> X <- df_deaths@coords
> library(KernSmooth)
> kde2d <- bkde2D(X, bandwidth=c(bw.ucv(X
    [,1]),bw.ucv(X[,2])))
> image(x=kde2d$x1, y=kde2d$x2,z=kde2d$fhat*
    1000, add=TRUE,col=clrs)
> contour(x=kde2d$x1, y=kde2d$x2,z=kde2d$
    fhat, add=TRUE)
```



The analogous Google Map plot

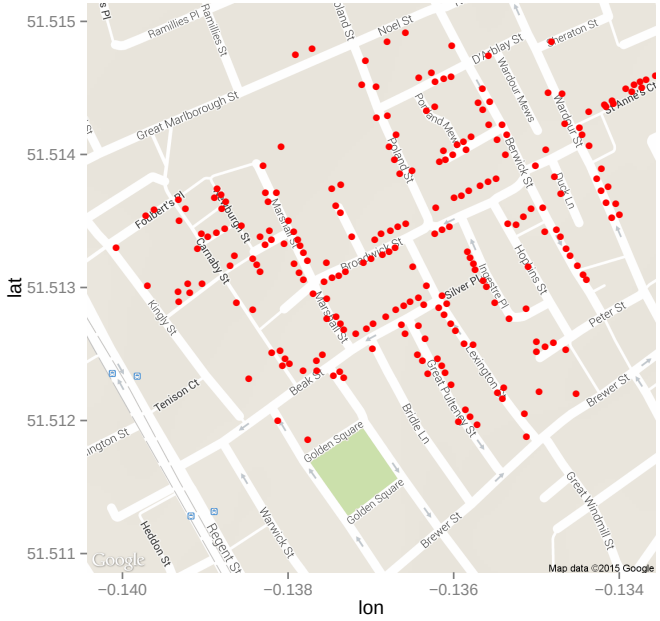
```
> library(ggmap)
> get_london <- get_map(c(-.137,51.513),
    zoom=17)
> london <- ggmap(get_london)
> london
```

Let us add points

```
> df_deaths <- data.frame(X)
```

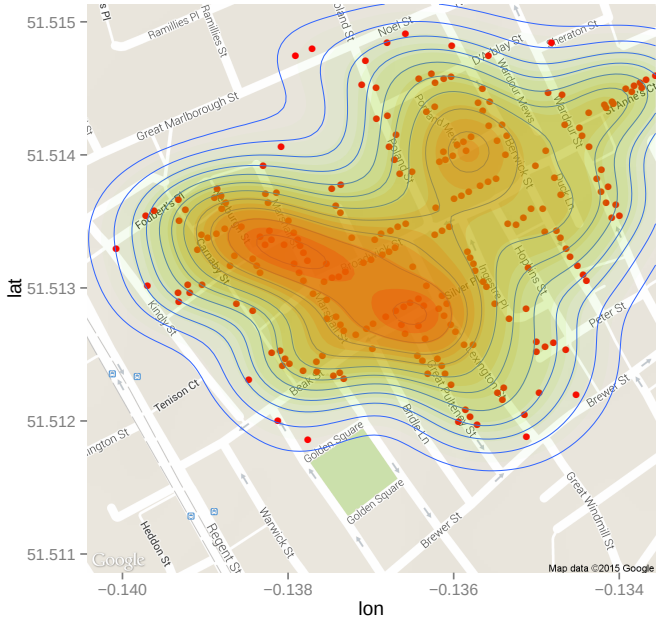
The analogous Google Map plot

```
> library(sp)
> library(rgdal)
> coordinates(df_deaths)=~coords.x1+coords.
    x2
> proj4string(df_deaths)=CRS("+init=epsg
    :27700")
> df_deaths = spTransform(df_deaths,CRS("+
    proj=longlat +datum=WGS84"))
> london + geom_point(aes(x=coords.x1,y=
    coords.x2),data=data.frame(df_deaths
    coords),col="red")
```



and we can add some heat map, too,

```
> london + geom_point(aes(x=coords.x1, y=
    coords.x2),
data=data.frame(df_deaths@coords),col="red")
     +
geom_density2d(data = data.frame(df_
    deaths@coords),
aes(x = coords.x1, y=coords.x2), size = 0.3)
     +
stat_density2d(data = data.frame(df_
    deaths@coords),
aes(x = coords.x1, y=coords.x2,fill = ..
    level.., alpha = ..level..),size = 0.01,
     bins = 16, geom = "polygon") + scale_
    fill_gradient(low = "green", high = "red
    ",guide = FALSE) +
  scale_alpha(range = c(0, 0.3), guide =
    FALSE)
```



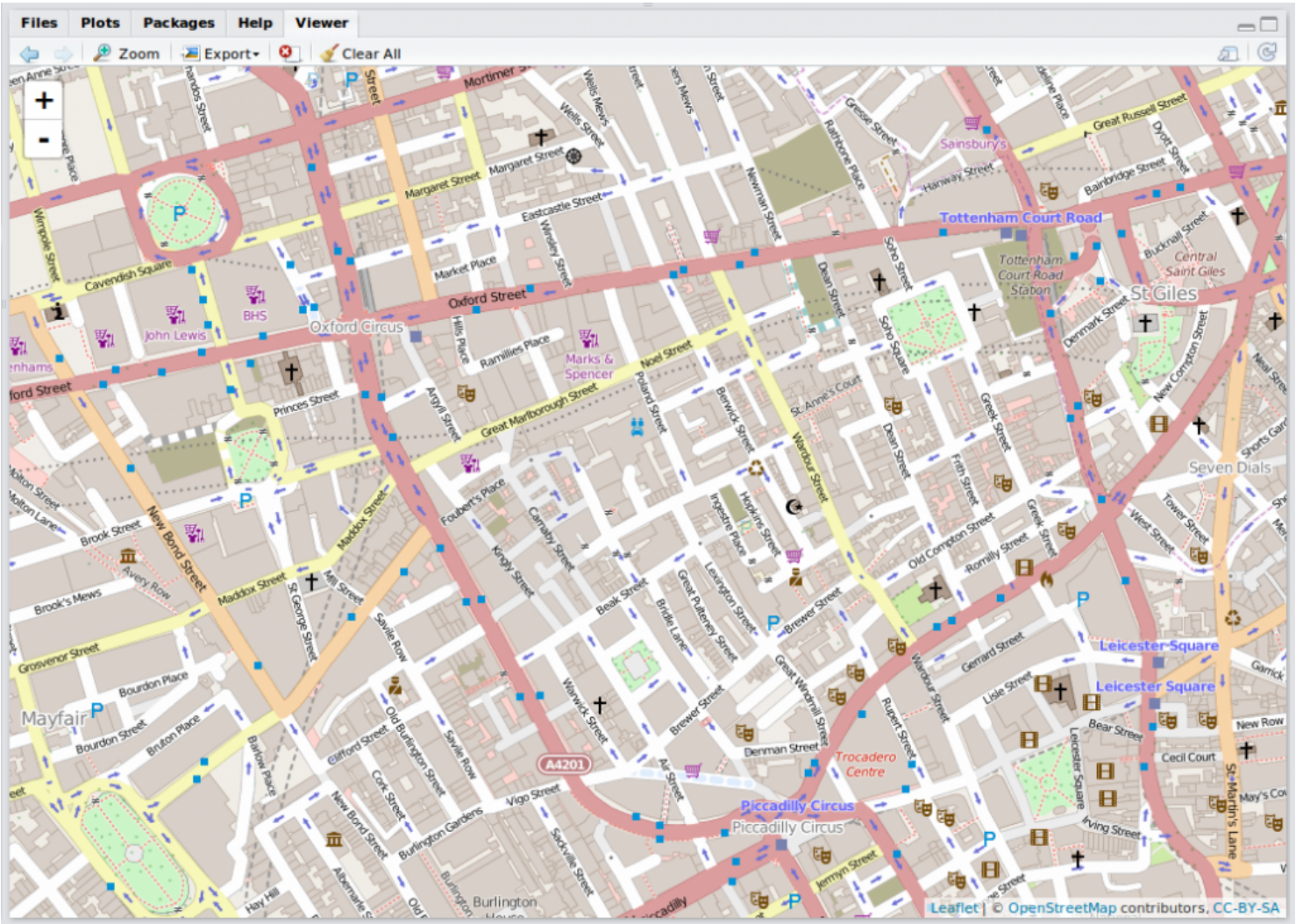
More Interactive Maps

As discussed previously, one can use RStudio and `library(leaflty)` see rpubs.com/freakonometrics/

```
> devtools::install_github("rstudio/leaflet")
> require(leaflet)
> setwd("/cholera/")
> deaths <- readShapePoints("Cholera_Deaths")
> df_deaths <- data.frame(deaths@coords)
> coordinates(df_deaths)=~coords.x1+coords.x2
> proj4string(df_deaths)=CRS("+init=epsg:27700")
> df_deaths=spTransform(df_deaths,CRS("+proj=longlat+datum=WGS84"))
> df=data.frame(df_deaths@coords)
> lng=df$coords.x1
> lat=df$coords.x2
```

```
> m = leaflet()%>% addTiles()
> m %>% fitBounds(-.141,  51.511, -.133, 51.516)
```

More Interactive Maps

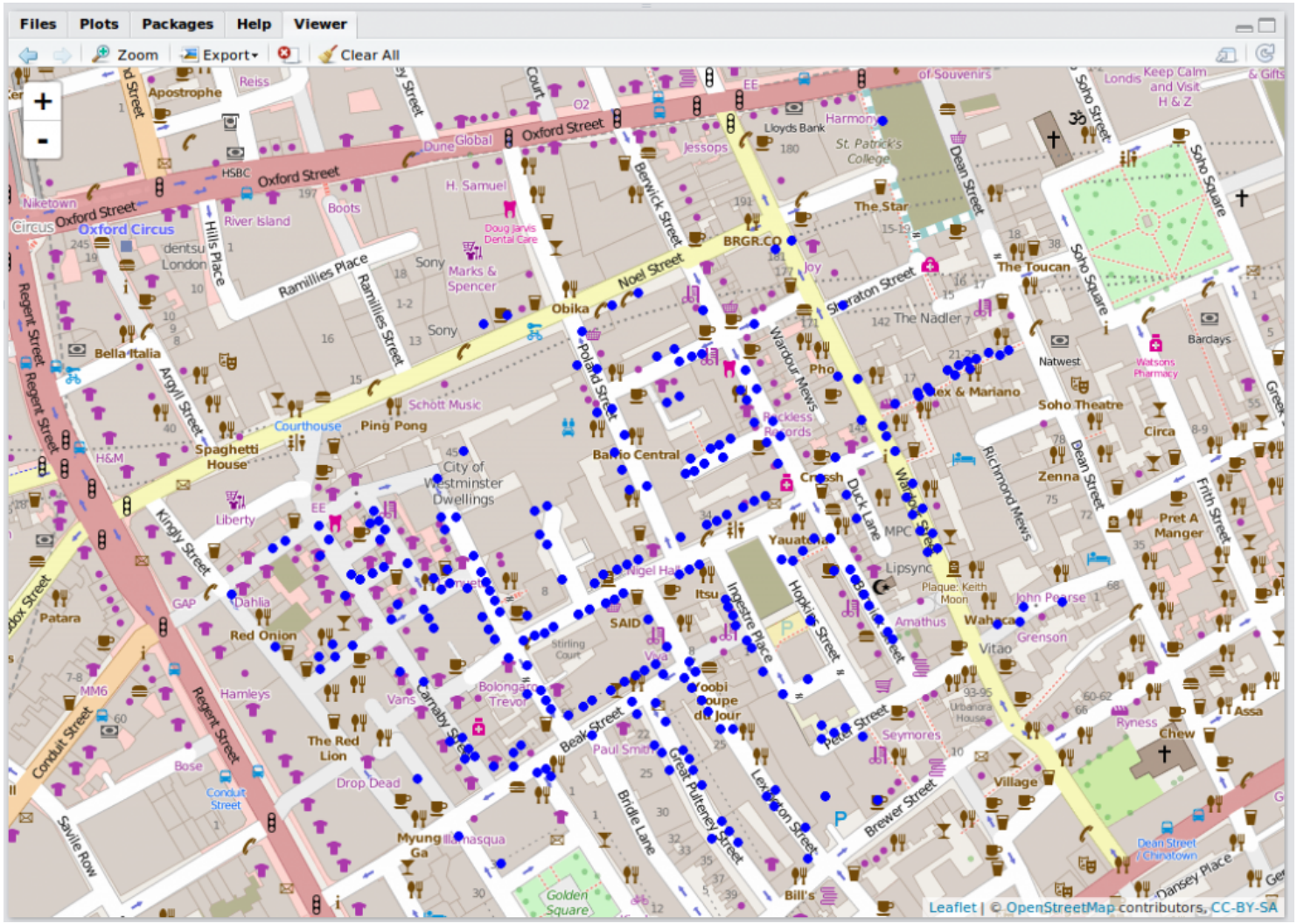


More Interactive Maps

One can add points

```
rd=.5
op=.8
clr="blue"
m = leaflet() %>% addTiles()
m %>% addCircles(lng,lat, radius = rd,opacity=op,col=clr)
```

More Interactive Maps



More Interactive Maps

We can also add some heatmap.

```
> X=cbind(lng,lat)
> kde2d <- bkde2D(X, bandwidth=c(bw.ucv(X[,1]),bw.ucv(X[,2])))
```

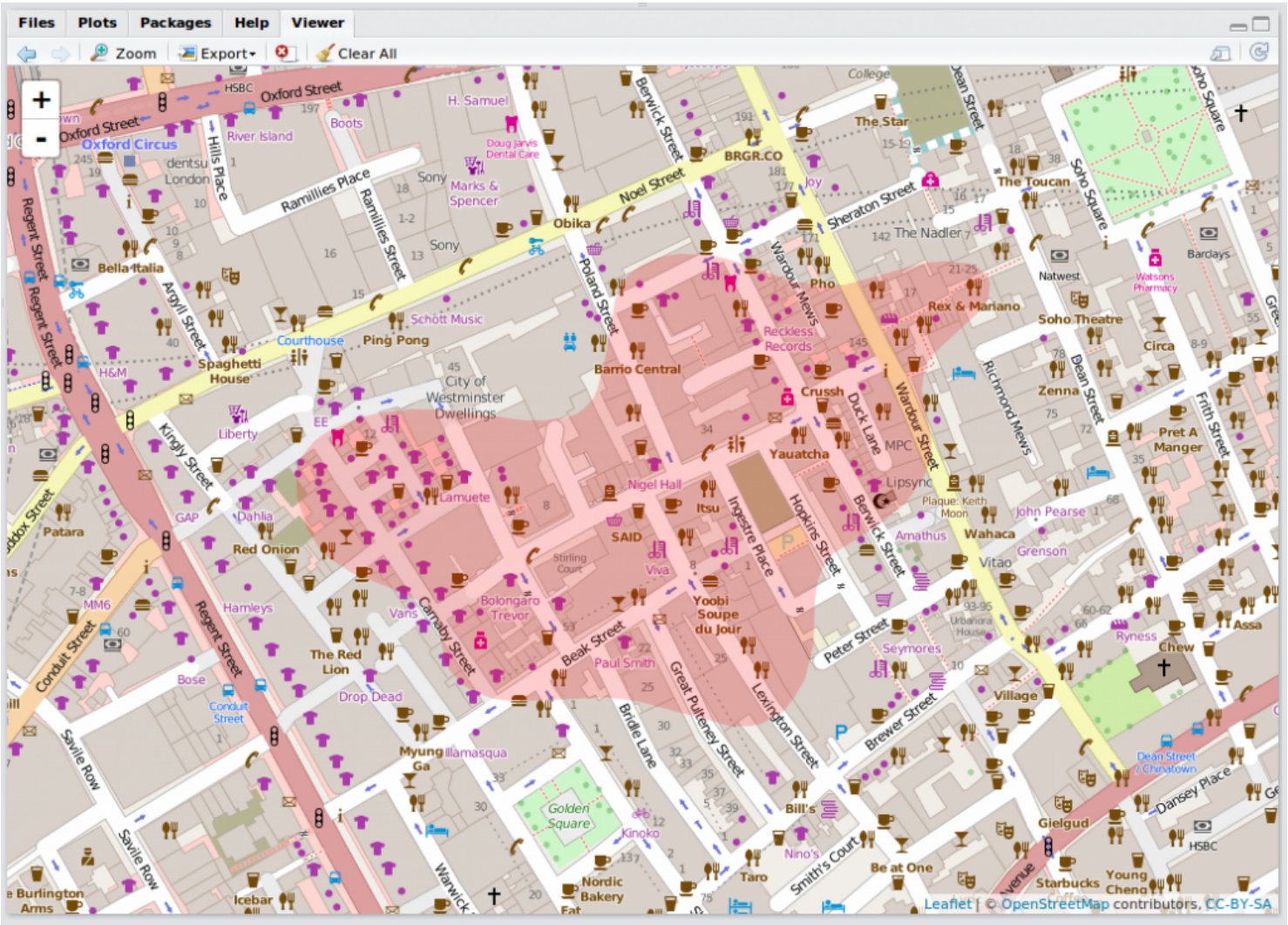
But there is no heatmap function (so far) so we have to do it manually,

```
> x=kde2d$x1
> y=kde2d$x2
> z=kde2d$fhat
> CL=contourLines(x , y , z)
```

We have now a list that contains lists of polygons corresponding to isodensity curves. To visualise of of then, use

```
> m = leaflet() %>% addTiles()
> m %>% addPolygons(CL[[5]]$x,CL[[5]]$y,fillColor = "red", stroke =
    FALSE)
```

More Interactive Maps



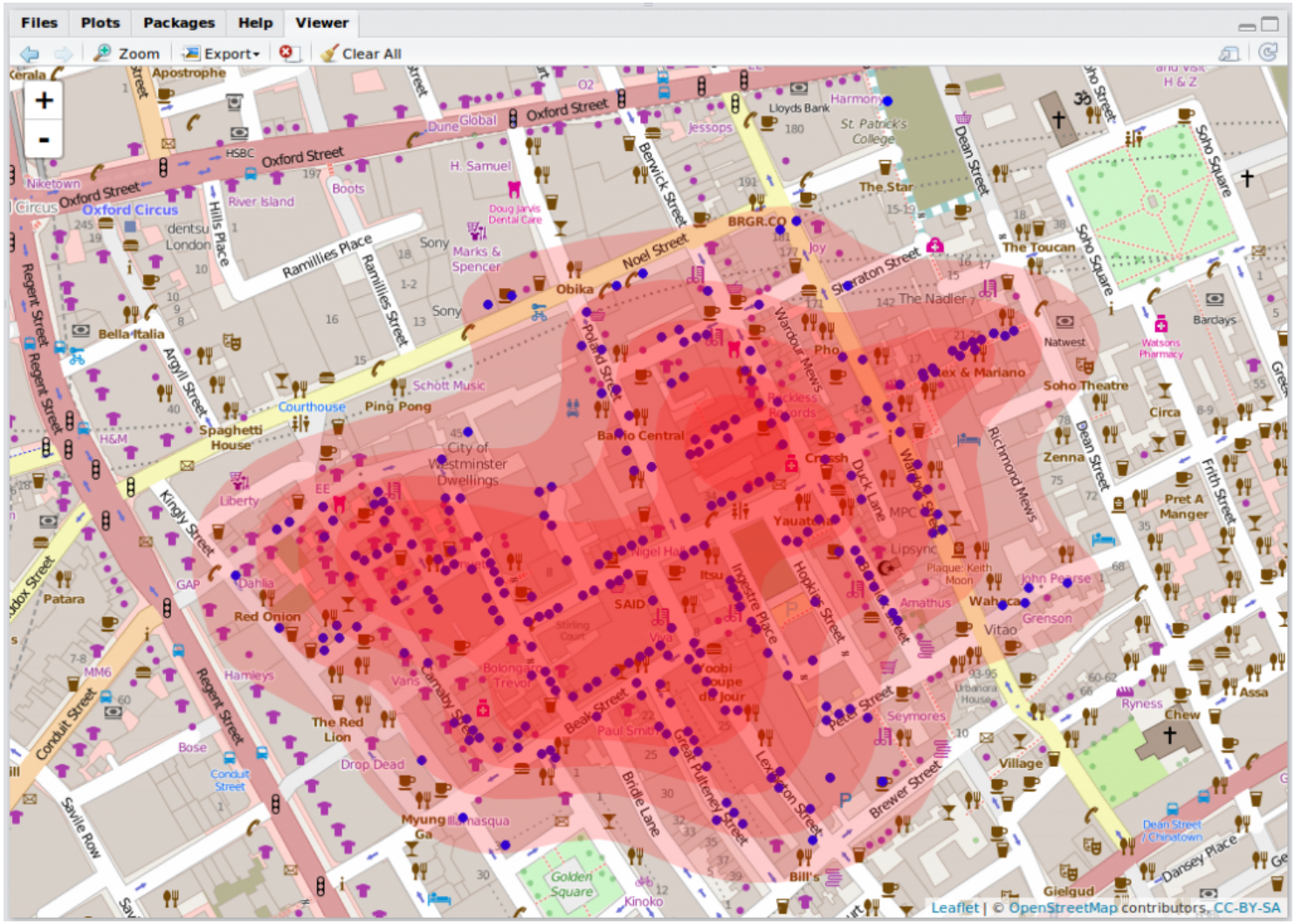
More Interactive Maps

We can get at the same time the points and the polygon

```
> m = leaflet() %>% addTiles()
> m %>% addCircles(lng,lat, radius = rd,opacity=op,col=clr) %>%
  addPolygons(CL[[5]]$x,CL[[5]]$y,fillColor = "red", stroke = FALSE)
```

```
> m = leaflet() %>% addTiles()
> m %>% addCircles(lng,lat, radius = rd,opacity=op,col=clr) %>%
  addPolygons(CL[[1]]$x,CL[[1]]$y,fillColor = "red", stroke = FALSE)
    %>%
  addPolygons(CL[[3]]$x,CL[[3]]$y,fillColor = "red", stroke = FALSE)
    %>%
  addPolygons(CL[[5]]$x,CL[[5]]$y,fillColor = "red", stroke = FALSE)
    %>%
  addPolygons(CL[[7]]$x,CL[[7]]$y,fillColor = "red", stroke = FALSE)
    %>%
  addPolygons(CL[[9]]$x,CL[[9]]$y,fillColor = "red", stroke = FALSE)
```

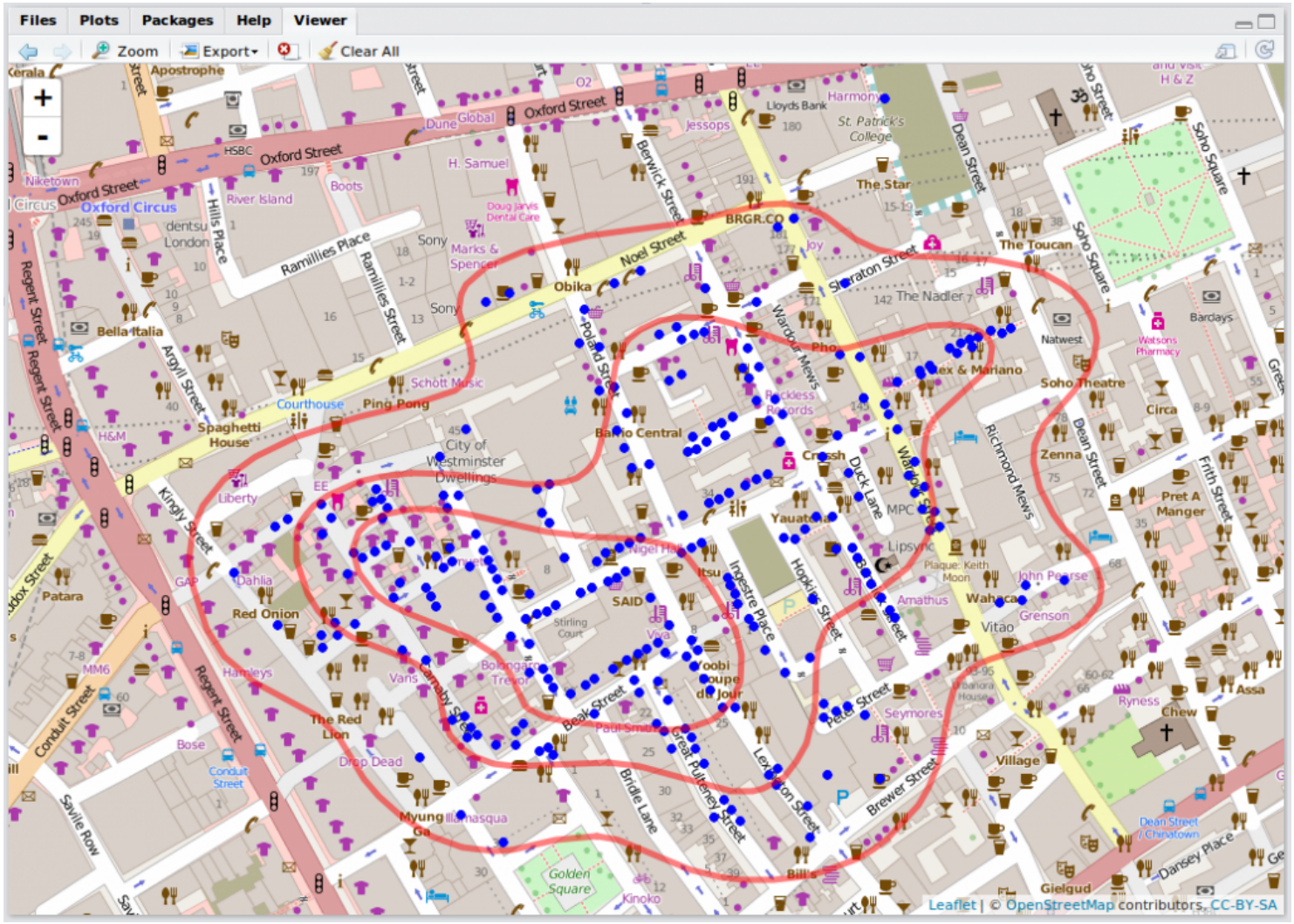
More Interactive Maps



More Interactive Maps

```
> m = leaflet() %>% addTiles()
> m %>% addCircles(lng,lat, radius = rd,opacity=op,col=clr) %>%
  addPolylines(CL[[1]]$x,CL[[1]]$y,color = "red") %>%
  addPolylines(CL[[5]]$x,CL[[5]]$y,color = "red") %>%
  addPolylines(CL[[8]]$x,CL[[8]]$y,color = "red")
```

More Interactive Maps



More Interactive Maps

Another package can be considered

```
> require(rleafmap)
> library(sp)
> library(rgdal)
> library(maptools)
> library(KernSmooth)
> setwd("/home/arthur/Documents/")
> deaths <- readShapePoints("Cholera_Deaths")
> df_deaths <- data.frame(deaths@coords)
> coordinates(df_deaths)=~coords.x1+coords.x2
> proj4string(df_deaths)=CRS("+init=epsg:27700")
> df_deaths = spTransform(df_deaths,CRS("+proj=longlat +datum=WGS84")
    )
> df=data.frame(df_deaths@coords)
> stamen_bm <- basemap("stamen.toner")
```

More Interactive Maps

```
> j_snow <- spLayer(df_deaths, stroke = FALSE)
> writeMap(stamen_bm, j_snow, width = 1000, height = 750, setView = c
    ( mean(df[,1]),mean(df[,2])), setZoom = 14)
> writeMap(stamen_bm, j_snow, width = 1000, height = 750, setView = c
    ( mean(df[,1]),mean(df[,2])), setZoom = 16)
```

More Interactive Maps

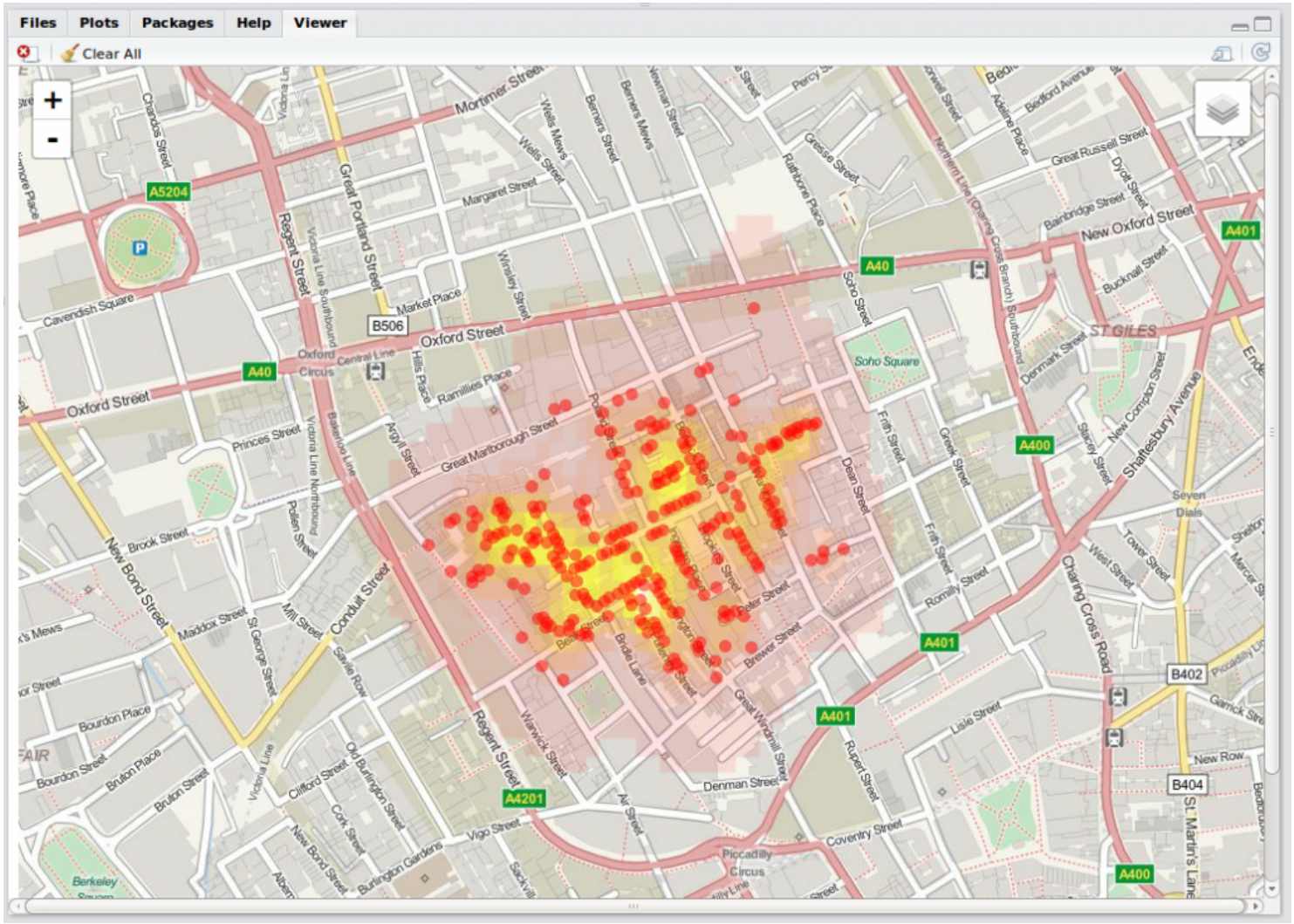
More Interactive Maps

```
> library(spatstat)
> library(maptools)
> win <- owin(xrange = bbox(df_deaths)[1,] + c(-0.01,0.01), yrange =
    bbox(df_deaths)[2,] + c(-0.01,0.01))
> df_deaths_ppp <- ppp(coordinates(df_deaths)[,1], coordinates(df_
    deaths)[,2], window = win)
> df_deaths_ppp_d <- density.ppp(df_deaths_ppp, sigma = min(bw.ucv(
    df[,1]),bw.ucv(df[,2])))
 > df_deaths_d <- as.SpatialGridDataFrame.im(df_deaths_ppp_d)
> df_deaths_d$v[df_deaths_d$v < 10^3] <- NA
> stamen_bm <- basemap("stamen.toner")
> mapquest_bm <- basemap("mapquest.map")
```

More Interactive Maps

```
> j_snow <- spLayer(df_deaths, stroke = FALSE)
> df_deaths_den <- spLayer(df_deaths_d, layer = "v", cells.alpha =
    seq(0.1, 0.8, length.out = 12))
> my_ui <- ui(layers = "topright")
> writeMap(stamen_bm, mapquest_bm, j_snow, df_deaths_den, width =
    1000, height = 750, interface = my_ui, setView = c( mean(df[,1]),
    mean(df[,2])), setZoom = 16)
```

More Interactive Maps



Visualizing a Spatial Process

Consider car/ bike accident in Paris,

see data.gouv.fr for bodily injury car accident in France (2006-2011, BAAC1 dataset) or opendata.paris.fr for accident in Paris, only.

```
> caraccident <- read.csv("http://opendata.paris.fr/explore/dataset/
    accidentologie/download/?format=csv&timezone=Europe/Berlin")
> geo_loc <- function(i) geocode(paste(caraccident$adresse[i], "paris
    ",sep=","))
> mat_geo_loc <- sapply(1:1000,geo_loc)
> save(mat_geo_loc,file="mat_geo_loc.RData")
```

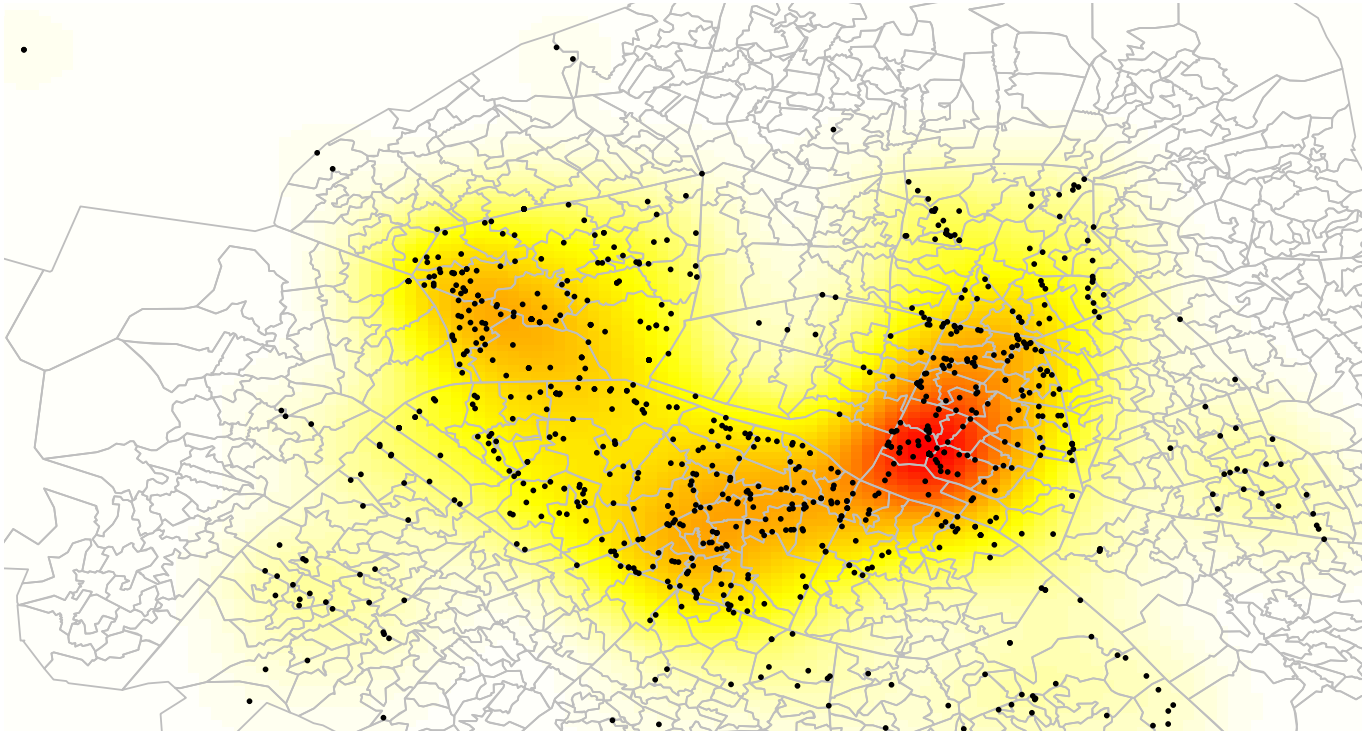
Visualizing a Spatial Process

Keep only accidents located in Paris

```
> mat_loc <- matrix(unlist(mat_geo_loc),nrow=2)
> x <- cbind(mat_loc[1,],mat_loc[2,])
> idx <- (x[,1]>2.25)&(x[,1]<2.4) & (x[,2]>48.83)&(x[,2]<48.9)
> x <- x[idx,]
```

```
> library(ks)
> fhat <- kde(x=x, H=diag(c(1e-6,1e-7))
> image(fhat$eval.points[[1]],fhat$eval.points[[2]],fhat$estimate,col
    = rev(heat.colors(100)),xlab="",ylab="",xlim=c(2.25,2.4), ylim=c
    (48.83,48.9),axes=FALSE)
> plot(paris,add=TRUE,border="grey")
> points(x,pch=19,cex=.3)
```

Visualizing a Spatial Process

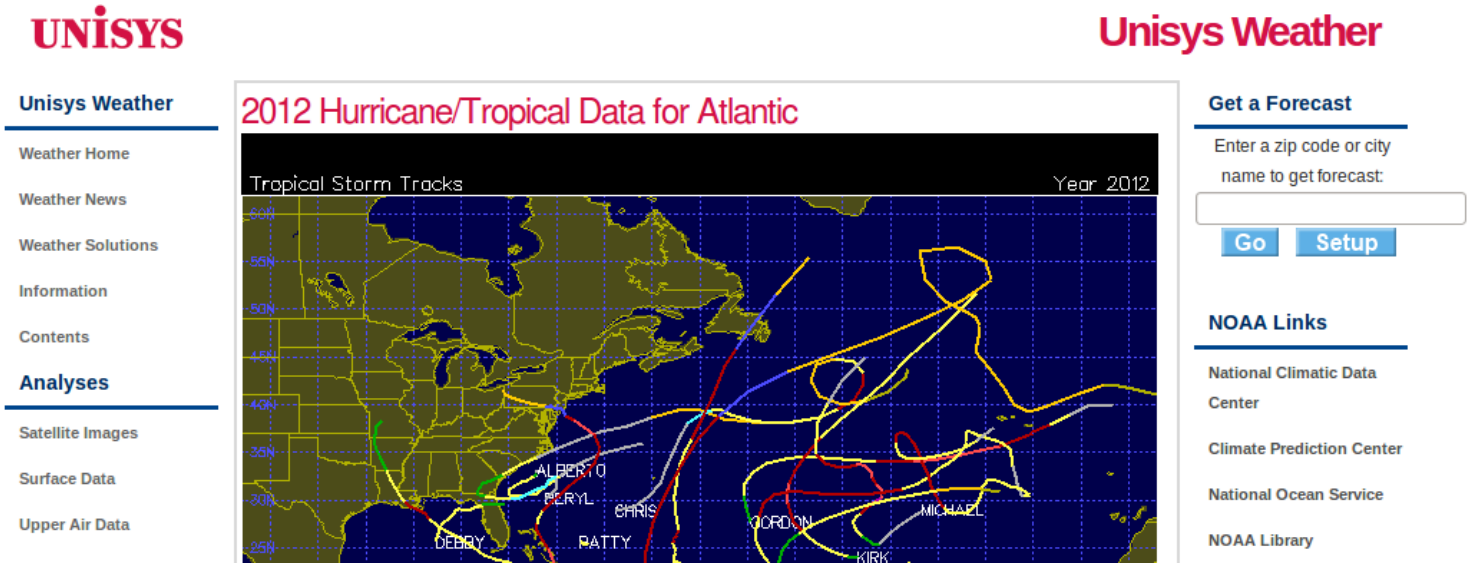


Visualizing Hurricane Paths

National Hurricane Center (NHC) collects datasets with all storms in North Atlantic, the North Atlantic Hurricane Database (HURDAT weather.unisys.com))

For all sorms we have the location of the storm, every six jours (at midnight, six a.m., noon and six p.m.), the maximal wind speed (on a 6 hour window) and the pressure in the eye of the storm. E.g. for 2012,

http://weather.unisys.com/hurricane/atlantic/2012/index.php



http://weather.unisys.com/hurricane/atlantic/2012/SANDY/track.dat

Date: 21-31 OCT 2012

Hurricane-3 SANDY

ADV	LAT	LON	TIME	WIND	PR	STAT
1	14.30	-77.40	10/21/18Z	25	1006	LOW
2	13.90	-77.80	10/22/00Z	25	1005	LOW
3	13.50	-78.20	10/22/06Z	25	1003	LOW
4	13.10	-78.60	10/22/12Z	30	1002	TROPICAL DEPRESSION
5	12.70	-78.70	10/22/18Z	35	1000	TROPICAL STORM
6	12.60	-78.40	10/23/00Z	40	998	TROPICAL STORM
7	12.90	-78.10	10/23/06Z	40	998	TROPICAL STORM
8	13.40	-77.90	10/23/12Z	40	995	TROPICAL STORM
9	14.00	-77.60	10/23/18Z	45	993	TROPICAL STORM
10	14.70	-77.30	10/24/00Z	55	990	TROPICAL STORM
11	15.60	-77.10	10/24/06Z	60	987	TROPICAL STORM
12	16.60	-76.90	10/24/12Z	65	981	HURRICANE-1

Data Scraping: Hurricane Dataset

1. get the name of all hurricane for a given year

```
> library(XML)
> year <- 2012
> loc <- paste("http://weather.unisys.com/hurricane/atlantic/",year,"
    /index.php",sep="")
> tabs <- readHTMLTable(htmlParse(loc))
> tabs[1]
$`NULL`
    #                      Name           Date Wind Pres Cat <U+00A0>
1   1   Tropical Storm ALBERTO      19-23 MAY   50  995   - <U+00A0>
2   2     Tropical Storm BERYL 25 MAY- 2 JUN   60  992   - <U+00A0>
3   3        Hurricane-1 CHRIS      17-24 JUN   75  974   1 <U+00A0>
4   4     Tropical Storm DEBBY      23-27 JUN   55  990   - <U+00A0>
5   5      Hurricane-2 ERNESTO       1-10 AUG   85  973   2 <U+00A0>
6   6 Tropical Storm FLORENCE       3- 8 AUG   50 1002   - <U+00A0>
7   7    Tropical Storm HELENE       9-19 AUG   40 1004   - <U+00A0>
```

Data Scraping: Hurricane Dataset

We split the Name variable to extract the name

```
> storms <- unlist(strsplit(as.character(tabs[[1]]$Name),split=" "))
> storms
 [1] "Tropical"    "Storm"       "ALBERTO"     "Tropical"    "Storm"
 [6] "BERYL"       "Hurricane-1" "CHRIS"       "Tropical"    "Storm"
```

But we keep only relevant information

```
> index <- storms%in%c("Tropical", "Storm", paste("Hurricane-",1:6,
    sep=""), "Depression", "Subtropical", "Extratropical", "Low",
    paste("Storm-",1:6,sep=""), "Xxx")
> nstorms  <- storms[!index]
> nstorms
 [1] "ALBERTO"   "BERYL"     "CHRIS"    "DEBBY"    "ERNESTO"   "FLORENCE"
 [7] "HELENE"    "GORDON"    "ISAAC"    "JOYCE"    "KIRK"      "LESLIE"
```

Data Scraping: Hurricane Dataset

before scraping the files, check the name of the hurricanes, there are typos

```
> for(i in length(nstorms):1){
if((nstorms[i]=="SIXTEE")&(year==2008)) nstorms[i] <- "SIXTEEN"
if((nstorms[i]=="LAUR")&(year==2008)) nstorms[i] <- "LAURA"
if((nstorms[i]=="FIFTEE")&(year==2007)) nstorms[i] <- "FIFTEEN"
if((nstorms[i]=="CHANTA")&(year==2007)) nstorms[i] <- "CHANTAL"
if((nstorms[i]=="ANDR")&(year==2007)) nstorms[i] <- "ANDREA"
if((nstorms[i]=="NINETE")&(year==2005)) nstorms[i] <- "NINETEEN"
if((nstorms[i]=="JOSEPH")&(year==2002)) nstorms[i] <- "JOSEPHINE"
if((nstorms[i]=="FLOY")&(year==1993)) nstorms[i] <- "FLOYD"
if((nstorms[i]=="KEIT")&(year==1988)) nstorms[i] <- "KEITH"
if((nstorms[i]=="CHARLI")&(year==1972)) nstorms[i] <- "CHARLIE"}
```

Data Scraping: Hurricane Dataset

Finally, loop to scrap files

```
> for(i in length(nstorms):1){
> loc <- paste("http://weather.unisys.com/hurricane/atlantic/",year,"
    /",nstorms[i],"/track.dat",sep="")
> track <- read.fwf(loc,skip=3,widths = c(4,6,8,12,4,6,20))
```

Change format to make sure numeric variables are really numeric

```
names(track)=c("ADV","LAT","LON","TIME","WIND","PR","STAT")
track$LAT=as.numeric(as.character(track$LAT))
track$LON=as.numeric(as.character(track$LON))
track$WIND=as.numeric(as.character(track$WIND))
track$PR=as.numeric(as.character(track$PR))
track$year=year
track$name=nstorms[i]
TRACK=rbind(TRACK,track)}
```

Data Scraping: Hurricane Dataset

We now have our dataset (at least for 2012)

```
> tail(TRACK,10)
     ADV  LAT   LON          TIME WIND   PR             STAT year     name
645   11 30.4 -79.1  05/21/12Z   35 1007 TROPICAL STORM 2012 ALBERTO
646   12 30.5 -78.3  05/21/18Z   35 1006 TROPICAL STORM 2012 ALBERTO
647   13 30.7 -77.1  05/22/00Z   35 1007 TROPICAL STORM 2012 ALBERTO
648   14 31.5 -76.1  05/22/06Z   35 1007 TROPICAL STORM 2012 ALBERTO
649   15 32.5 -74.7  05/22/12Z   30 1008           LOW 2012 ALBERTO
650   16 33.4 -73.4  05/22/18Z   30 1008           LOW 2012 ALBERTO
651   17 34.1 -71.9  05/23/00Z   25 1010           LOW 2012 ALBERTO
652   18 34.9 -70.1  05/23/06Z   25 1011           LOW 2012 ALBERTO
653   19 35.5 -67.9  05/23/12Z   25 1012           LOW 2012 ALBERTO
654   20 35.9 -66.0  05/23/18Z   25 1012           LOW 2012 ALBERTO
```

Then loop on the years, from 2012 to 1851...

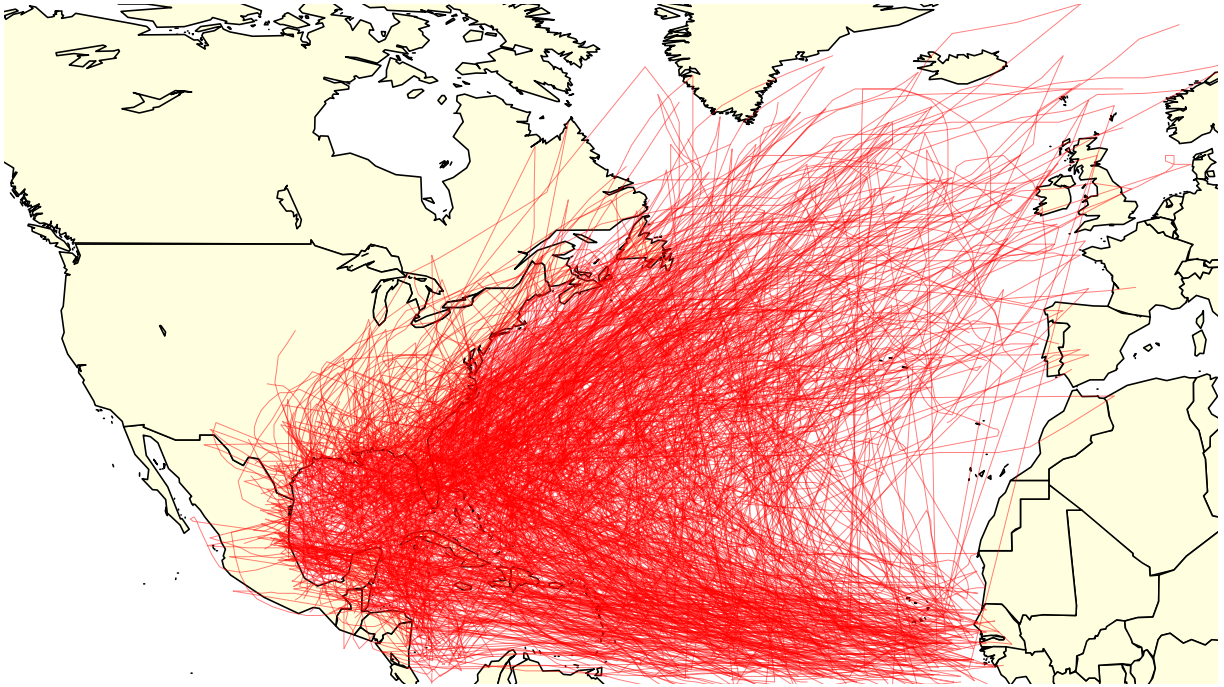
Data Scraping: Hurricane Dataset

```
> TOTTRACK=NULL
> for(y in 2012:1851){
+ TOTTRACK=rbind(TOTTRACK,extract.track(y))
+ }
> save(TOTTRACK,file="TRACK-ATLANTIC.Rdata")
```

We can load that file, and vizualize hurricane tracks

```
> library(grDevices)
> library(maps)
> map("world", col="light yellow", fill=TRUE)
> for(n in unique(TOTTRACK$name)){
+ lines(TOTTRACK$LON[TOTTRACK$name==n],TOTTRACK$LAT[TOTTRACK$name==n
    ],lwd=.5,col=rgb(1, 0, 0, alpha=.5))}
```

Data Scraping: Hurricane Dataset

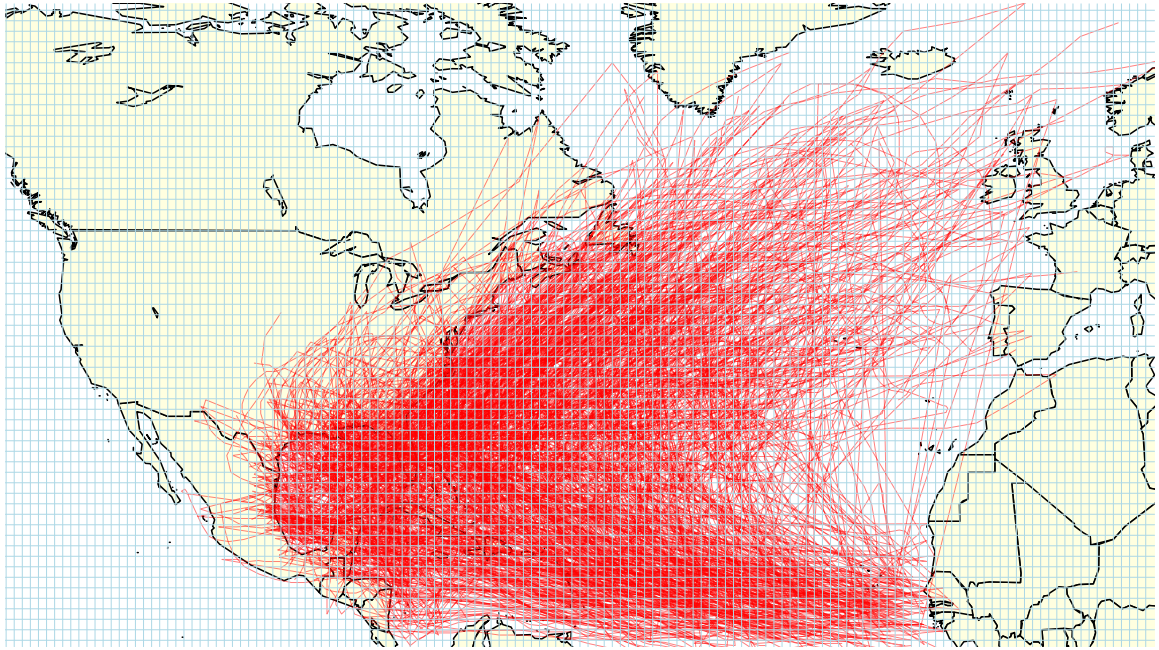


can we use those data to generate our own hurricane trajectories?

A Markov Spatial Model for Hurricanes

Consider a grid for the latitude and the longitude

```
> gridx <- seq(-150,10)
> gridy <- seq(-10,80)
```



A Markov Spatial Model for Hurricanes

Given a location x_t and y_t on the grid, at time t, define the transition probability matrix

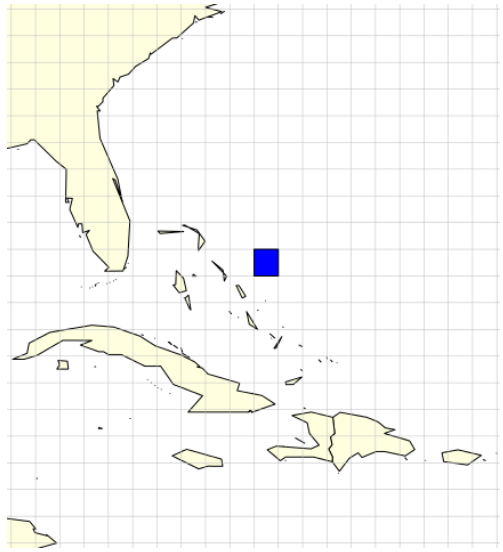
$$\boldsymbol{M}_t^{t+1} = \mathbb{P}[\boldsymbol{z}_{t+1} = (x_{t+1}, y_{t+1}) | \boldsymbol{z}_t = (x_t, y_t)]$$

To compute the empirical version, and the probability distribution of $\boldsymbol{z}_{t+1}$ given $\boldsymbol{z}_t \in [x_i, x_{i+1}] \times [y_j, y_{j+1}]$, use

```
idx <- which((TOTTRACK$LON>=gridx[i])&(TOTTRACK$LON<gridx[i+1]) & (
    TOTTRACK$LAT>=gridy[j])&(TOTTRACK$LAT<gridy[j+1]) )
```

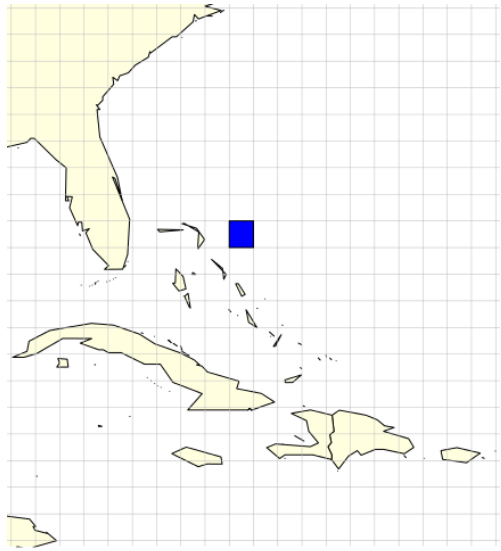
Then we look for all possible next move (i.e. 6 hours later), if any

```
> for(s in 1:length(idx)){
> locx <- floor(TOTTRACK$LON[idx[s]+1])
> locy <- floor(TOTTRACK$LAT[idx[s]+1])}
```



We also compute the probability to end at this locate, or to move. Given that the hurricane is still moving, we get a list for the next location.

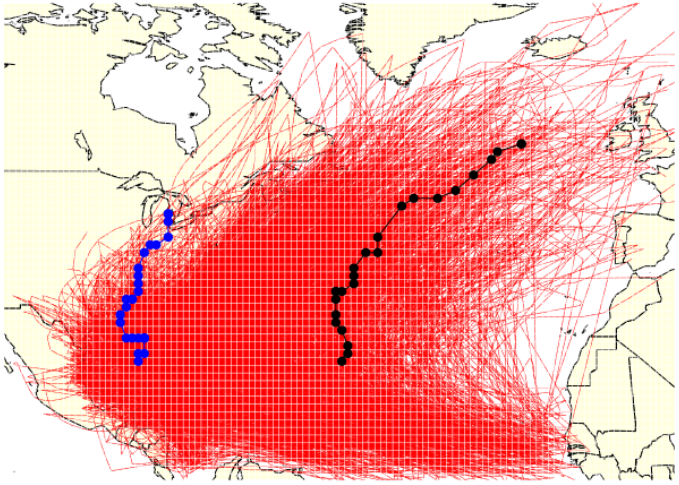
Then we loop



Generating Hurricanes Trajectories

The algorithm to generate a trajectory is

- select a possible starting point (or cell in the grid) (x_0, y_0)
- given (x_t, y_t) draw a Bernoulli varible to see if the hurricane continues, or ends
- if it continues, sample from an observed trajectories a possible(x_{t+1}, y_{t+1})



Gas Price, in France

```
> rm(list=ls())
> year=2014
> loc=paste("http://donnees.roulez-eco.fr/opendata/annee/",year,sep="
    ")
> download.file(loc,destfile="oil.zip")

Content type 'application/zip' length 15248088 bytes (14.5 MB)

> unzip("oil.zip", exdir="./")
> fichier=paste("PrixCarburants_annuel_",year,
".xml",sep="")
> library(plyr)
> library(XML)
> library(lubridate)
> l=xmlToList(fichier)
> length(l)
[1] 11064
```

Gas Price, in France

To extract information for gas station `no=2` and `Gasole` use

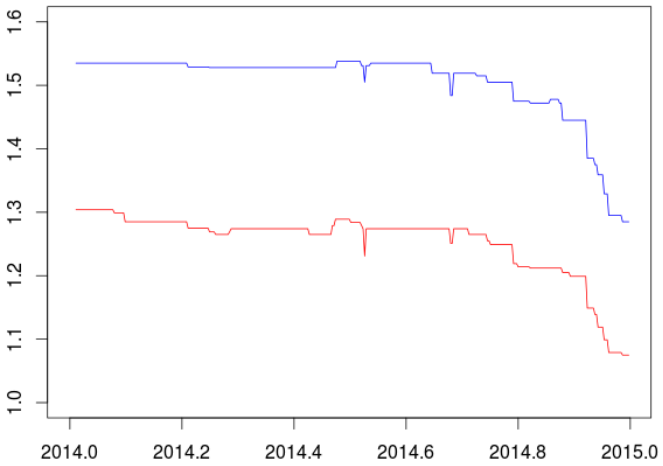
```
>    prix=list()
>    date=list()
>    nom=list()
>    j=0; no=2
>    for(i in 1:length(l[[no]])){
+      v=names(l[[no]])
+      if(!is.null(v[i])){
+        if(v[i]=="prix"){
+          j=j+1
+   date[[j]]=as.character(l[[no]][[i]]["maj"])
+   prix[[j]]=as.character(l[[no]][[i]]["valeur"])
+   nom[[j]]=as.character(l[[no]][[i]]["nom"])
+       }}  }
>    id=which(unlist(nom)==type_gas)
```

Gas Price, in France

```
> ext_y=function(j) substr(date[[id[j]]],1,4)
> ext_m=function(j) substr(date[[id[j]]],6,7)
> ext_d=function(j) substr(date[[id[j]]],9,10)
> ext_h=function(j) substr(date[[id[j]]],12,13)
> ext_mn=function(j) substr(date[[id[j]]],15,16)
> prix_essence=function(i) as.numeric(prix[[id[i]]])/1000
>     Y=unlist(lapply(1:n,ext_y))
>     M=unlist(lapply(1:n,ext_m))
>     D=unlist(lapply(1:n,ext_d))
>     H=unlist(lapply(1:n,ext_h))
>     MN=unlist(lapply(1:n,ext_mn))
> date=paste(base1$Y,"-",base1$M,"-",base1$D,
+ " ",base1$H,":",base1$MN,":00",sep="")
> date_base=as.POSIXct(date, format =
+                 "%Y-%m-%d %H:%M:%S", tz = "UTC")
```

Gas Price, in France

```
> d=paste(year,"-01-01 12:00:00",sep="")
> f=paste(year,"-12-31 12:00:00",sep="")
> vecteur_date=seq(as.POSIXct(d, format =
+ "%Y-%m-%d %H:%M:%S"),
+ as.POSIXct(f, format =
+ "%Y-%m-%d %H:%M:%S"),by="days")
> vect_idx=Vectorize(function(t) sum(vecteur
    _date[t]>=date_base))(1:length(vecteur_
    date))
> prix_essence=function(i) as.numeric(prix[[
    id[i]]])/1000
> P=c(NA,unlist(lapply(1:n,prix_essence)))
> Z=ts(P[1+vect_idx],start=year,frequency
    =365)
```



Gas Price, in France

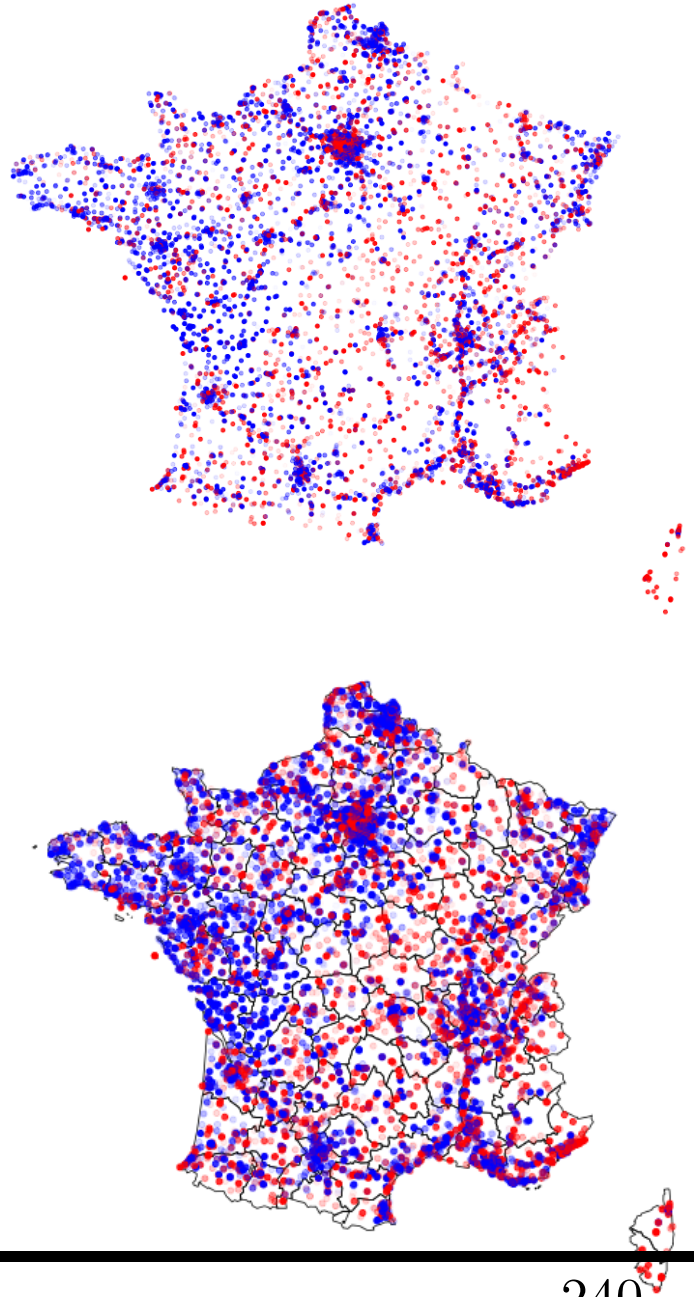
```
> dt = as.Date("2014-05-05")
>   base=NULL
>   for(no in 1:length(l)){
+     prix=list()
+     date=list()
+     j=0
+     for(i in 1:length(l[[no]])){
+     v=names(l[[no]])
+     if(!is.null(v[i])){
+       if(v[i]=="prix"){
+   j=j+1
+   date[[j]]=as.character(l[[no]][[i]]["maj"])
+       }}}
+   n=j
+   D=as.Date(substr(unlist(date),1,10),"%Y-%m-%d")
+   k=which(D==D[which.max(D[D<=dt])])
```

Gas Price, in France

```
+ if(length(k)>0){
+   B=Vectorize(function(i) l[[no]][[k[i]]])(1:length(k))
+ if("nom" %in%  rownames(B)){
+   k=which(B["nom",]=="Gazole")
+   prix=as.numeric(B["valeur",k])/1000
+   if(length(prix)==0) prix=NA
+   base1=data.frame(indice=no,
+   lat=as.numeric(l[[no]]$.attrs["latitude"])/100000,
+   lon=as.numeric(l[[no]]$.attrs["longitude"])/100000,
+   gaz=prix)
+   base=rbind(base,base1)
+ }}
```

Gas Price, in France

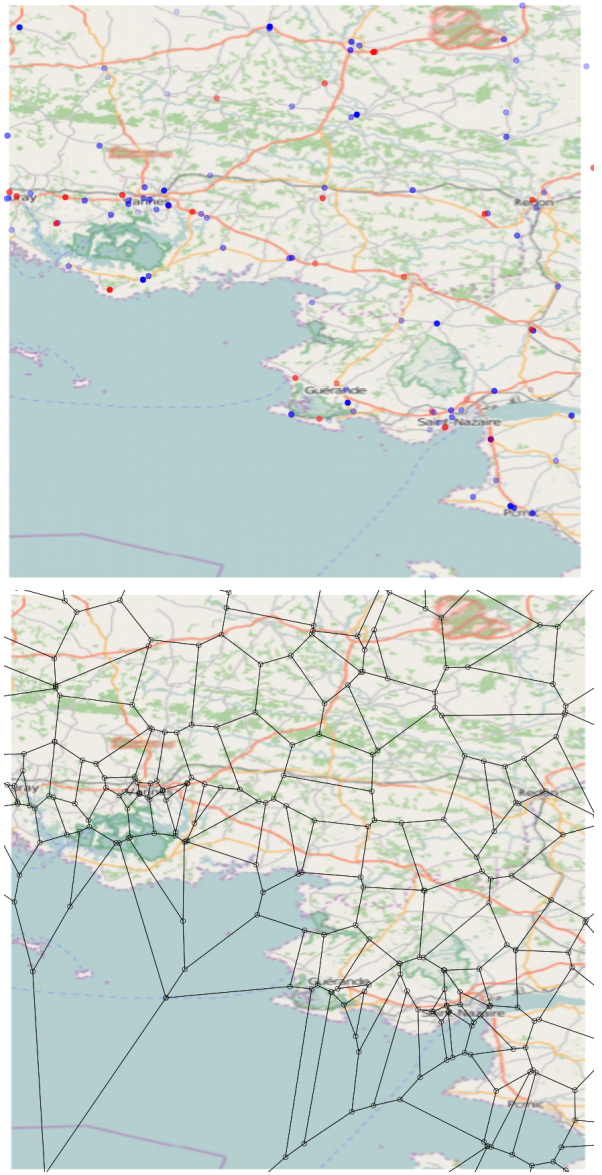
```
> idx=which((base$lon>(-10))&(base$lon<20)&
+ (base$lat>35)&(base$lat<55))
> B=base[idx,]
> Q=quantile(B$gaz,seq(0,1,by=.01),na.rm=
    TRUE)
> Q[1]=0
> x=as.numeric(cut(B$gaz,breaks=unique(Q)))
> CL=c(rgb(0,0,1,seq(1,0,by=-.025)),
+ rgb(1,0,0,seq(0,1,by=.025)))
> plot(B$lon,B$lat,pch=19,col=CL[x])
> library(maps)
> map("france")
> points(B$lon,B$lat,pch=19,col=CL[x])
```



Gas Price, in France

```
> library(OpenStreetMap)
> map <- openmap(c(lat= 48,    lon= -3),
+                c(lat= 47,    lon= -2))
> map <- openproj(map)
> plot(map)
> points(B$lon,B$lat,pch=19,col=CL[x])
```

```
> library(tripack)
> V <- voronoi.mosaic(dB$lon[id],dB$lat[id])
> plot(V,add=TRUE)
```



Gas Price, in France

```
> plot(map)
> P <- voronoi.polygons(V)
> library(sp)
> point_in_i=function(i,point) point.in.
    polygon(point[1],point[2],P[[i]][,1],P[[
    i]][,2])
> which_point=function(i) which(Vectorize(
    function(j) point_in_i(i,c(dB$lon[id[j
    ]],dB$lat[id[j]])))(1:length(id))>0)
> for(i in 1:length(P)) polygon(P[[i]],col=
    CL[x[id[which_point(i)]]],border=NA)
```