Actuariat de l'Assurance Non-Vie # 2

A. Charpentier (Université de Rennes 1)

credit: Arnold Odermatt

ENSAE 2017/2018

Modélisation économétrique d'une variable dichotomique

Utile en fraude, marketing, crédit, catastrophes, décès ou pour modéliser la fréquence sur des branches spécifiques (max 1 sinistre par an).

Références: Frees (2010), chapitre 11 (p 305-342), Greene (2012), sections 17.2 et 17.3 (p 863-715), de Jong & Heller (2008), chapitre 7.

Remarque: la régression logistique est un cas particulier des modèles GLM, avec une loi binomiale et une fonction de lien logit.

Modèles introduits par Gaddum (1933), Bliss (1935), ou Berkson (1944).

Modélisation économétrique d'une variable dichotomique

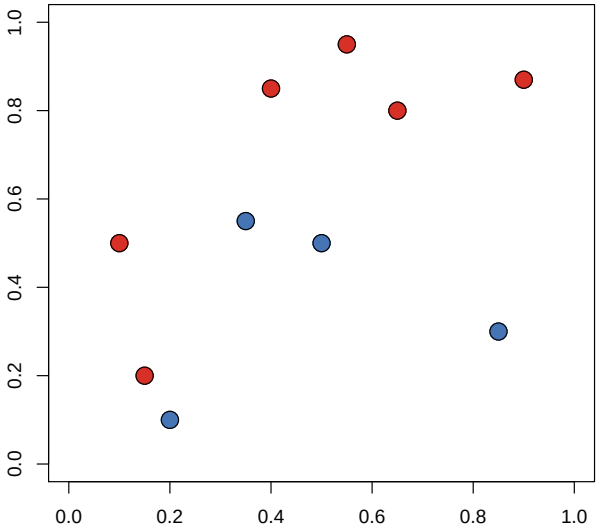
Ici $y_i \in \{0, 1\}$, mais pour les méthodes de machine learning $y_i \in \{-1, +1\}$, ou pour les dessins $y_i \in \{\bullet, \bullet\}$.

La prévision se fera en deux étapes

- estimation d'une fonction de score,

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

- construction d'un classifieur $s(\boldsymbol{x}) \to \widehat{y} \in \{0, 1\}$.



Bases de données

- Repésentation d'un assuré par un avocat (accident corporel)

```
> avocat <- read.table("http://freakonometrics.free.fr/AutoBI.csv",
    header=TRUE,sep=",")
> avocat$CLMSEX <- factor(avocat$CLMSEX, labels=c("M","F"))
> avocat$MARITAL <- factor(avocat$MARITAL, labels=c("M","C","V","D"))
> avocat <- avocat[,-c(1,5)]
```

- Survenance d'accident sur des polices d'assurance de camping car

```
> camping <- read.table("http://freakonometrics.free.fr/baseN.txt",
    header=TRUE,sep=";")
```

- Décès par infarctus suite à une admission aux urgences

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

(cf. code en ligne sur le blog, détails donnés dans les slides)

La loi binomiale $\mathcal{B}(\pi)$

Soient Y_i des variables aléatoires prenant les valeurs $y_i \in \{0, 1\}$, avec probabilité $1 - \pi_i$ et π_i (respectivement), i.e.

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

avec $\pi_i \in [0, 1]$. Aussi, $\mathbb{P}(Y_i = 1) = \pi_i$ et $\mathbb{P}(Y_i = 0) = 1 - \pi_i$.

Alors $\mathbb{E}(Y_i) = \pi_i$ et $\text{Var}(Y_i) = \pi[1 - \pi_i] < \mathbb{E}(Y_i)$.

En statistique, on suppose que $\pi_i = \pi$, $\forall i \in \{1, 2, \cdots, n\}$,

et on cherche à estimer π

Maximum de vraisemblance et méthode des moments

La fonction de vraisemblance, pour un échantillon $y_1, \cdots, y_n$ s'écrit

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

et la log-vraisemblance est

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

La condition du premier ordre est

$$\frac{\partial \log \mathcal{L}(\pi; \boldsymbol{y})}{\partial \pi} = \sum_{i=1}^{n} \frac{y_i}{\pi} - \frac{1-y_i}{1-\pi} = 0$$

Intervalle de confiance pour π

On a un intervalle de confiance asymptotique car on sait que l'estimateur de vraisemblance est asymptotiquement efficace,

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

où $I(\pi)$ est l'information de Fisher, i.e.

$$I(\pi) = \frac{1}{\pi[1-\pi]}$$

d'où un intervalle de confiance approché (à 95%) pour π de la forme

$$\left[\widehat{\pi} \pm \frac{1.96}{\sqrt{n}} \sqrt{\widehat{\pi}[1-\widehat{\pi}]}\right].$$

Intervalle de confiance pour π

On peut aussi construire un intervalle de confiance, par le théorème central limite, car $\widehat{\pi} = \overline{y}$. On sait que

$$\sqrt{n}\frac{\overline{Y} - \mathbb{E}(Y)}{\sqrt{\text{Var}(Y)}} \xrightarrow{\mathcal{L}} \mathcal{N}(0,1),$$

avec ici $\overline{Y} = \widehat{\pi}$, $\mathbb{E}(Y) = \pi$ et $\text{Var}(Y) = \pi(1-\pi)$, i.e. un intervalle de confiance est obtenu par l'approximation

$$\sqrt{n}\frac{\widehat{\pi} - \pi}{\sqrt{\widehat{\pi}[1-\widehat{\pi}]}} \xrightarrow{\mathcal{L}} \mathcal{N}(0,1),$$

d'où un intervalle de confiance (à 95%) pour π de la forme

$$\left[\widehat{\pi} \pm \frac{1.96}{\sqrt{n}}\sqrt{\widehat{\pi}[1-\widehat{\pi}]}\right].$$

Mise en oeuvre pratique

Considérons l'échantillon suivant, $\{y_1, \cdots, y_n\}$

```
> set.seed(1)
> n=20
> (Y=sample(0:1,size=n,replace=TRUE))
 [1] 0 0 1 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 1
```

Comme $Y_i \sim \mathcal{B}(\pi)$, avec $\pi = \mathbb{E}(Y)$. On note $\widehat{\pi} = \overline{y}$, soit ici

```
> mean(X)
[1] 0.55
```

On peut faire un test de $H_0 : \pi = \pi_\star$ contre $H_1 : \pi \neq \pi_\star$ (par exemple 50%)

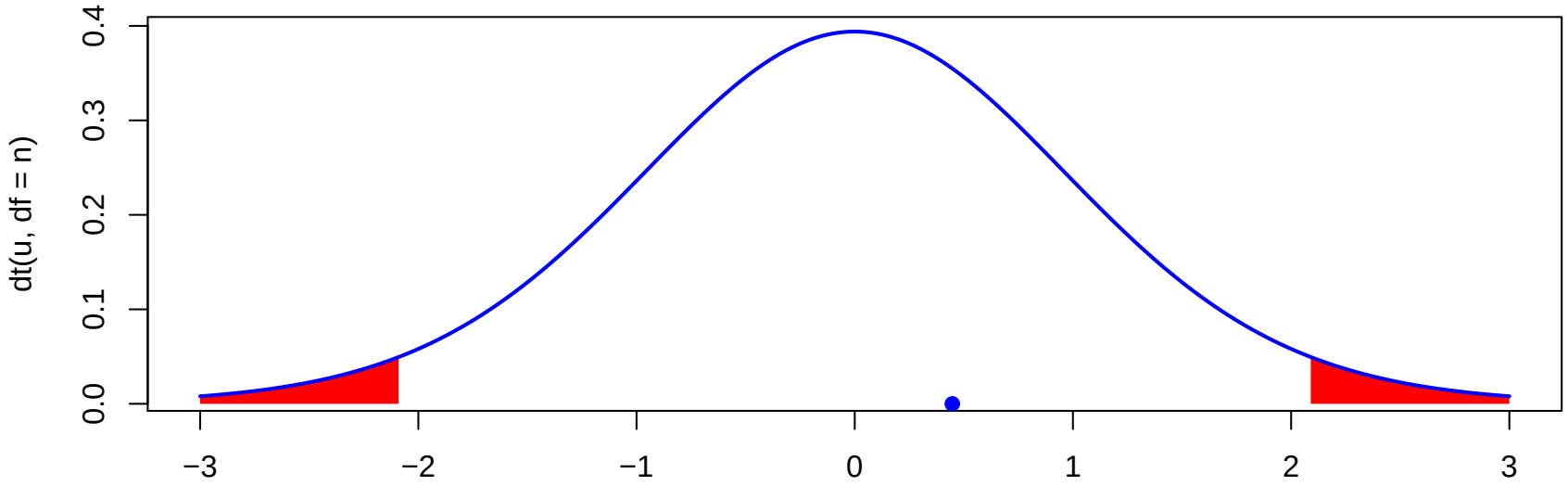
On peut utiliser le test de Student,

$$T = \sqrt{n}\frac{\widehat{\pi} - \pi_\star}{\sqrt{\pi_\star(1-\pi_\star)}}$$

qui suit, sous H_0 une loi de Student à n degrés de liberté.

Mise en oeuvre pratique

```
> (T=sqrt(n)*(pn-p0)/(sqrt(p0*(1-p0))))
[1] 0.4472136
> abs(T)<qt(1-alpha/2,df=n)
[1] TRUE
```

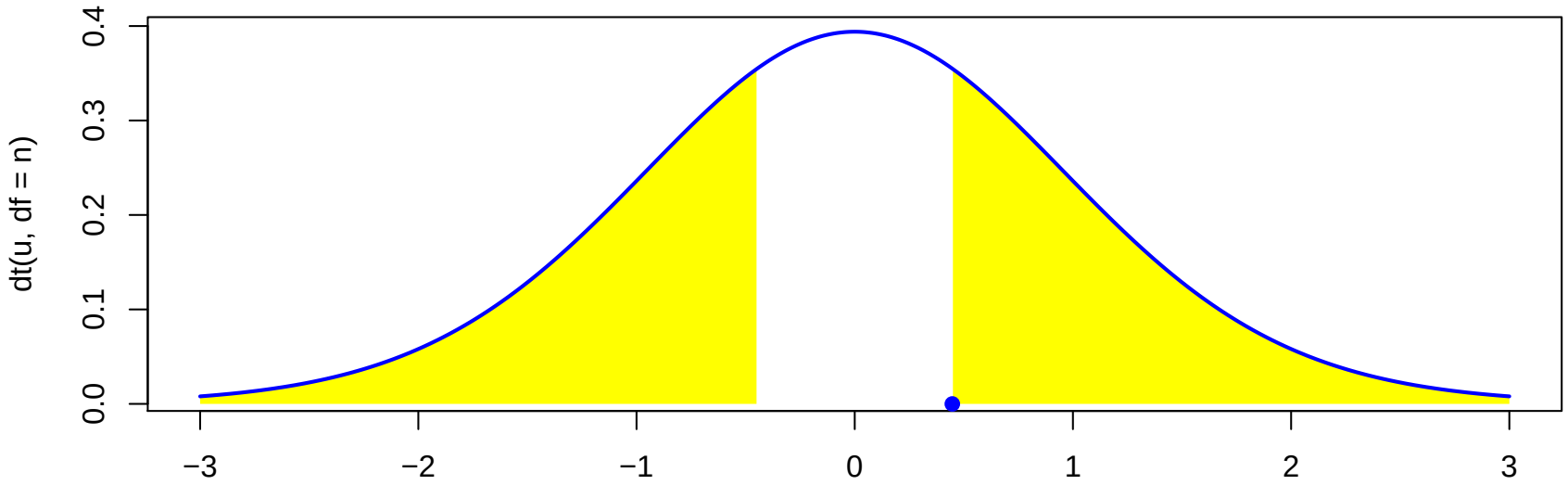


Mise en oeuvre pratique

On est ici dans la région d'acceptation du test.

On peut aussi calculer la p-value, $\mathbb{P}(|T| > |t_{obs}|)$,

```
> 2*(1-pt(abs(T),df=n))
[1] 0.6595265
```



Mise en oeuvre pratique

Test de Wald l'idée est d'étudier la différence entre $\widehat{\pi}$ et $\pi_\star$. Sous H_0,

$$T = n\frac{(\widehat{\pi} - \pi_\star)^2}{I^{-1}(\pi_\star)} \xrightarrow{\mathcal{L}} \chi^2(1)$$

Test du rapport de vraisemblance l'idée est d'étudier la différence entre $\log \mathcal{L}(\widehat{\theta})$ et $\log \mathcal{L}(\theta_\star)$. Sous H_0,

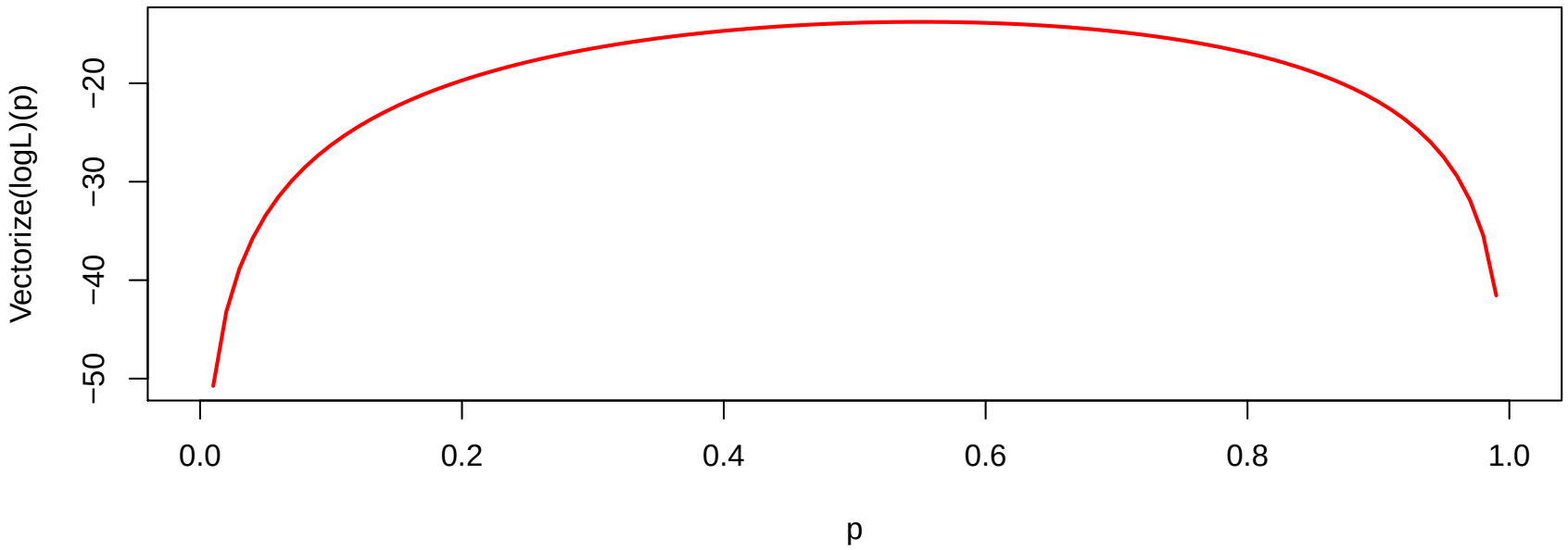
$$T = 2\log\left(\frac{\log \mathcal{L}(\theta_\star)}{\log \mathcal{L}(\widehat{\theta})}\right) \xrightarrow{\mathcal{L}} \chi^2(1)$$

Test du score l'idée est d'étudier la différence entre $\dfrac{\partial \log \mathcal{L}(\pi_\star)}{\partial \pi}$ et 0. Sous H_0,

$$T = \left(\frac{1}{n}\sum_{i=1}^{n}\frac{\partial \log f_{\pi_\star}(x_i)}{\partial \pi}\right)^2 \xrightarrow{\mathcal{L}} \chi^2(1)$$

Mise en oeuvre pratique

```
> p=seq(0,1,by=.01)
> logL=function(p){sum(log(dbinom(X,size=1,prob=p)))}
> plot(p,Vectorize(logL)(p),type="l",col="red",lwd=2)
```



Mise en oeuvre pratique

On peut trouver numériquement le maximum de $\log \mathcal{L}$,

```
> neglogL=function(p){-sum(log(dbinom(X,size=1,prob=p)))}
> pml=optim(fn=neglogL,par=p0,method="BFGS")
> pml
$par
[1] 0.5499996

$value
[1] 13.76278
```

i.e. on retrouve ici $\widehat{\pi} = \overline{y}$.

Mise en oeuvre pratique

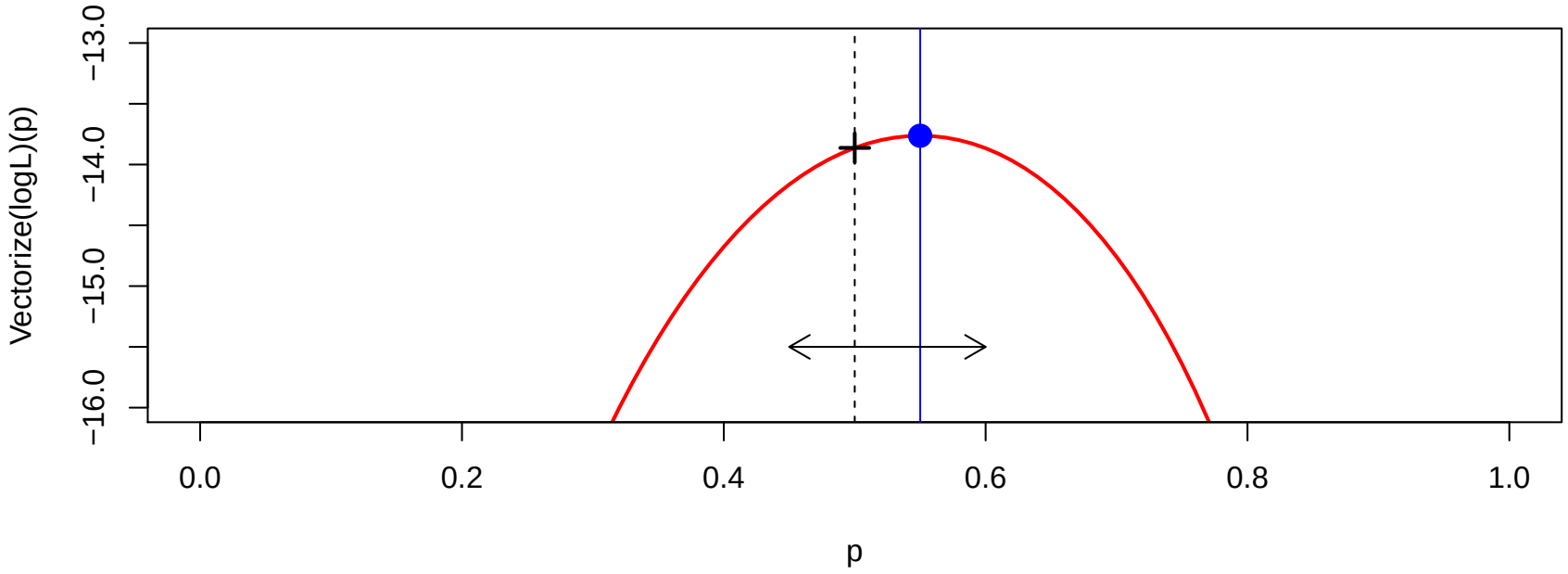
Maintenant, on peut tester $H_0 : \pi = \pi_\star = 50\%$ versus $H_1 : \pi \neq 50\%$. Pour le test de Wald, on peut commencer par calculer $nI(\theta_\star)$ ou ,

```
> nx=sum(X==1)
> f = expression(nx*log(p)+(n-nx)*log(1-p))
> Df = D(f, "p")
> Df2 = D(Df, "p")
> p=p0=0.5
> (IF=-eval(Df2))
[1] 80
```

Mise en oeuvre pratique

On peut d'ailleurs comparer cette valeur avec la valeur théorique, car $I(\pi)^{-1} = \pi(1-\pi)$

```
> 1/(p0*(1-p0)/n)
[1] 80
```



Mise en oeuvre pratique

La statistique du test de Wald est ici

```
>  pml=optim(fn=neglogL,par=p0,method="BFGS")$par
>  (T=(pml-p0)^2*IF)
[1] 0.199997
```

que l'on va chercher à comparer avec le quantile de la loi du χ^2,

```
>  T<qchisq(1-alpha,df=1)
[1] TRUE
```

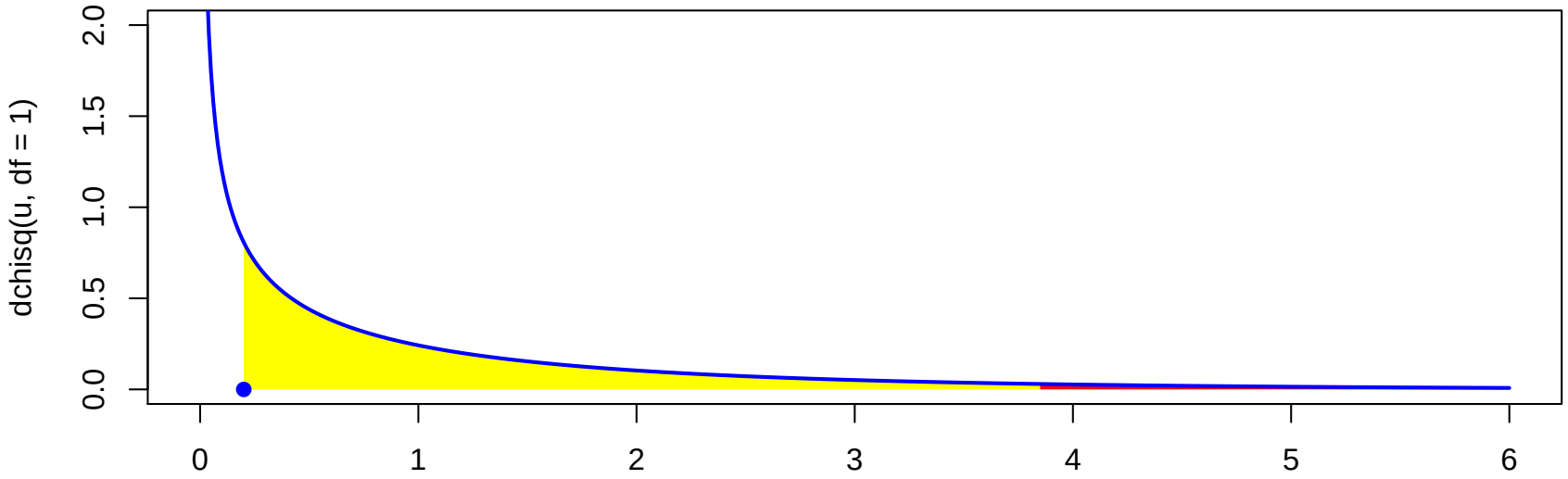
i.e. on est dans la région d'acceptation du test.

Mise en oeuvre pratique

De manière duale, on peut calculer la p-value du test,

```
> 1-pchisq(T,df=1)
[1] 0.6547233
```

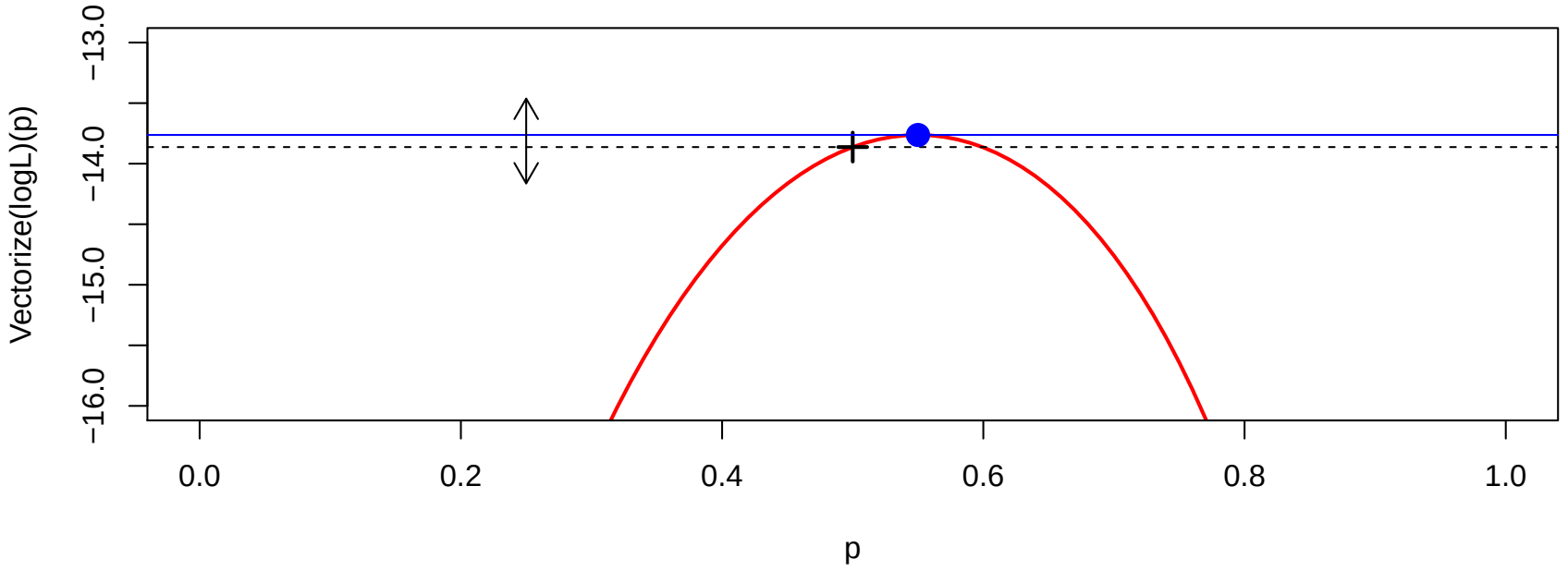
Donc on va accepter H_0.



Mise en oeuvre pratique

Pour le test du rapport de vraisemblance, T est ici

```
> (T=2*(logL(pml)-logL(p0)))
[1] 0.2003347
```



Mise en oeuvre pratique

Là encore, on est dans la région d'acceptation,

```
> T<qchisq(1-alpha,df=1)
[1] TRUE
```

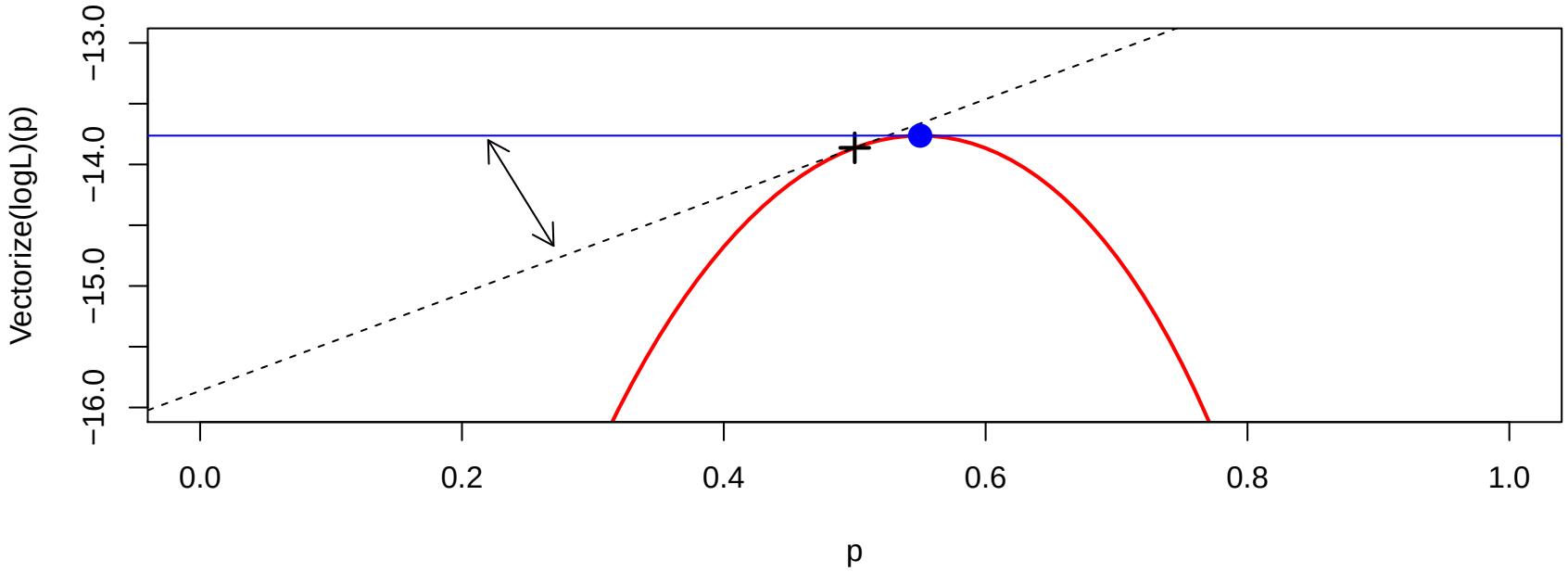
Enfin, on peut faire le test du score. Le score se calcule facilement,

```
> nx=sum(X==1)
> f = expression(nx*log(p)+(n-nx)*log(1-p))
> Df = D(f, "p")
> p=p0
> score=eval(Df)
```

La statistique de test est alors

```
> (T=score^2/IF)
[1] 0.2
```

Mise en oeuvre pratique



Là encore, on est dans la région d'acceptation,

```
> T<qchisq(1-alpha,df=1)
[1] TRUE
```

Prise en compte de l'hétérogénéité

Supposons que π_i soit fonction de variables explicatives $\boldsymbol{X}_i$, e.g.

$$\pi_i = \mathbb{E}(Y_i|\boldsymbol{X}_i).$$

Le modèle classique est le modèle linéaire, $\mathbb{E}(Y_i|\boldsymbol{X}_i) = \boldsymbol{X}_i'\boldsymbol{\beta}$

On peut faire une estimation par moindres carrés pour estimer $\boldsymbol{\beta}$,

$$\widehat{\boldsymbol{\beta}} = \text{argmin}\left\{\sum_{i=1}^{n}[Y_i - \boldsymbol{X}_i'\boldsymbol{\beta}]^2, \boldsymbol{\beta} \in \mathbb{R}^k\right\}$$

Problème: $\pi_i \in [0,1]$ et $\boldsymbol{X}_i'\boldsymbol{\beta} \in \mathbb{R}$.

Problème: modèle non homoscédastique, en effet

$$\varepsilon_i = \begin{cases} 1 - \boldsymbol{X}_i'\boldsymbol{\beta} \text{ avec probabilité } \pi_i \\ -\boldsymbol{X}_i'\boldsymbol{\beta} \text{ avec probabilité } 1-\pi_i \end{cases}$$

i.e. $\text{Var}(\varepsilon_) = \boldsymbol{X}_i'\boldsymbol{\beta} \cdot [1 - x\boldsymbol{X}_i'\boldsymbol{\beta}]$.

Utilisation de la cote (odds)

$\pi_i \in [0,1]$ alors que $\boldsymbol{X}_i'\boldsymbol{\beta} \in \mathbb{R}$, donc on ne peut pas supposer

$$\pi_i = \mathbb{E}(Y_i|\boldsymbol{X}_i) = \boldsymbol{X}_i'\boldsymbol{\beta}$$

$\longrightarrow$ utilisation de la côte

$$\text{odds}_i = \frac{\pi_i}{1-\pi_i} \in [0,\infty].$$

ou, en passant au logarithme

$$\log(\text{odds}_i) = \log\left(\frac{\pi_i}{1-\pi}\right) \in \mathbb{R}.$$

On appelle logit cette transormation,

$$\text{logit}(\pi_i) = \log\left(\frac{\pi_i}{1-\pi}\right)$$

Utilisation de la cote (odds)

On va supposer que

$$\text{logit}(\pi_i) = \boldsymbol{X}_i'\boldsymbol{\beta}$$

ou

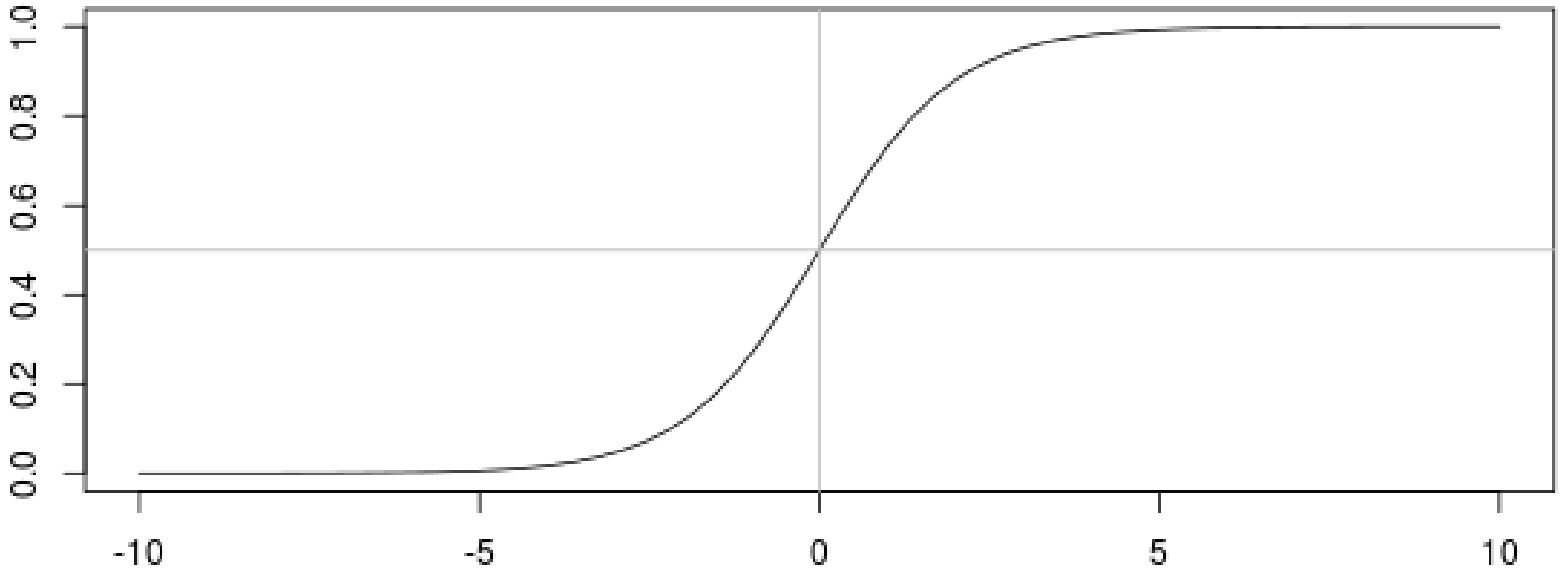
$$\pi_i = \text{logit}^{-1}(\boldsymbol{X}_i'\boldsymbol{\beta}) = \frac{\exp[\boldsymbol{X}_i'\boldsymbol{\beta}]}{1+\exp[\boldsymbol{X}_i'\boldsymbol{\beta}]}.$$

Remarque: si π_i est faible ($\pi_i \sim 0$) alors $\text{logit}^{-1} \sim \exp$, i.e.

$$\pi_i \sim \exp(\boldsymbol{X}_i'\boldsymbol{\beta})$$

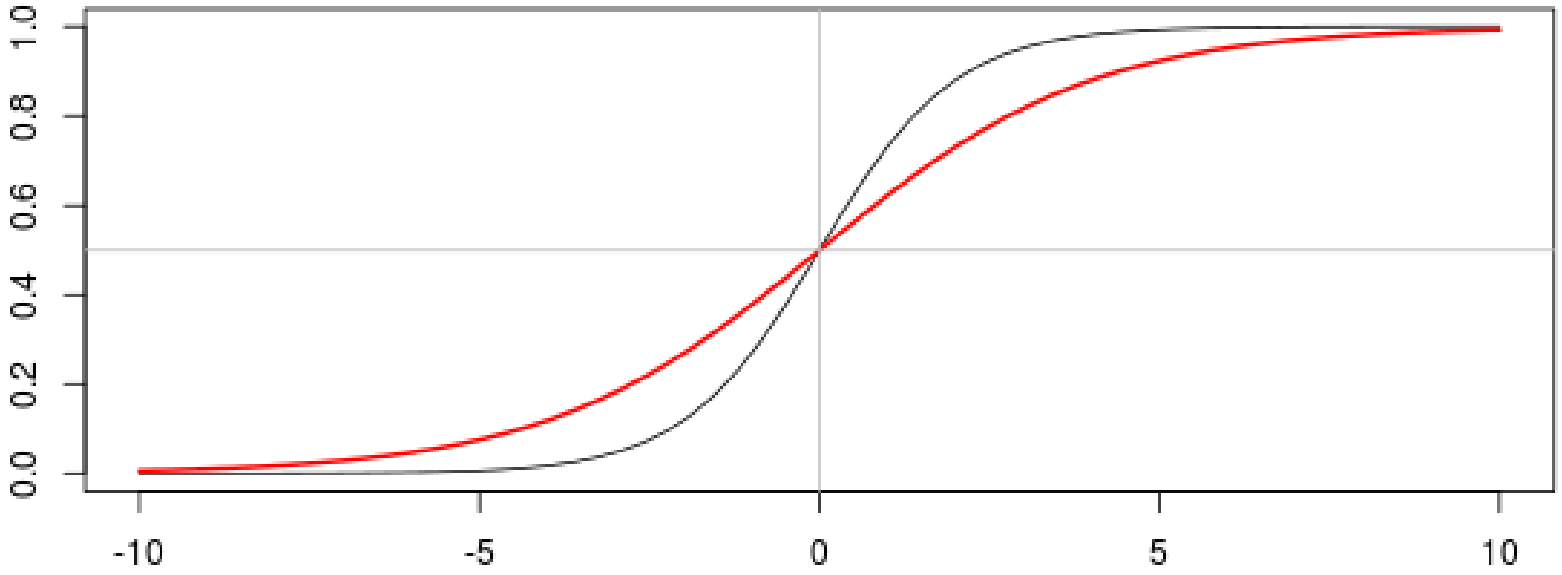
La fonction logistique

$$F(x) = \frac{\exp[x]}{1+\exp[x]}, \forall x \in \mathbb{R}.$$



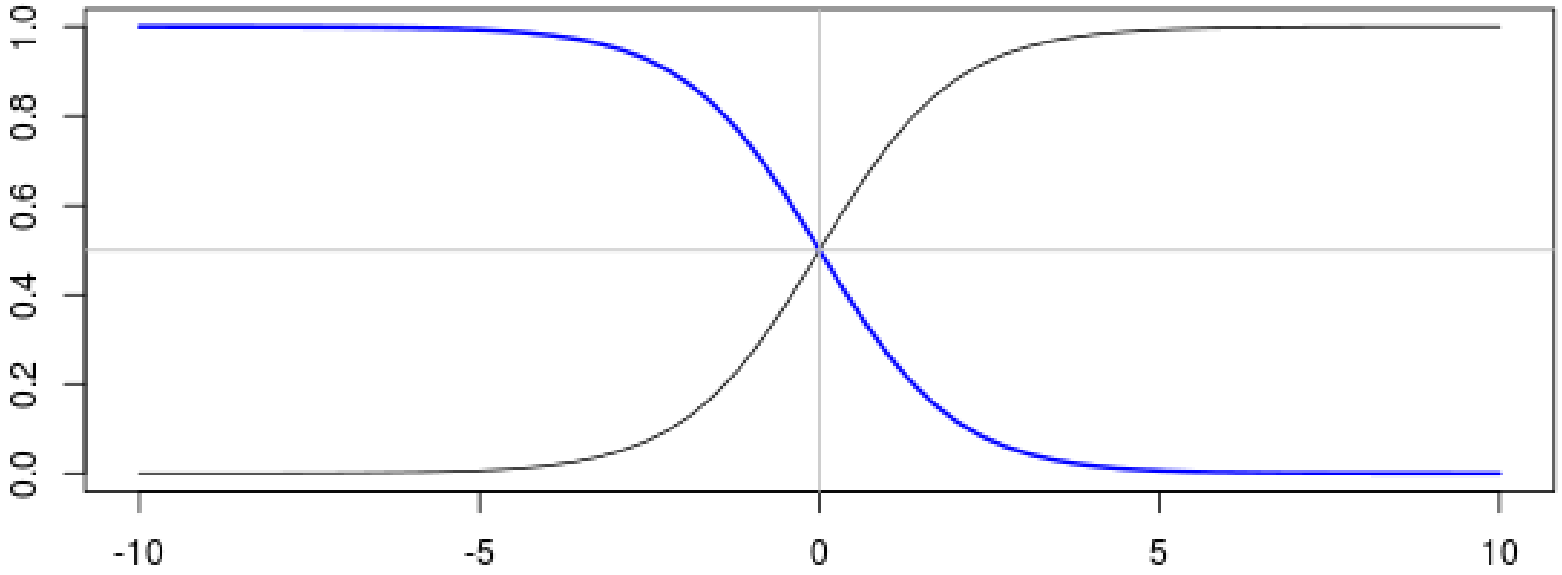
La fonction logistique

$$F(x) = \frac{\exp[2x]}{1+\exp[2x]}, \forall x \in \mathbb{R}.$$



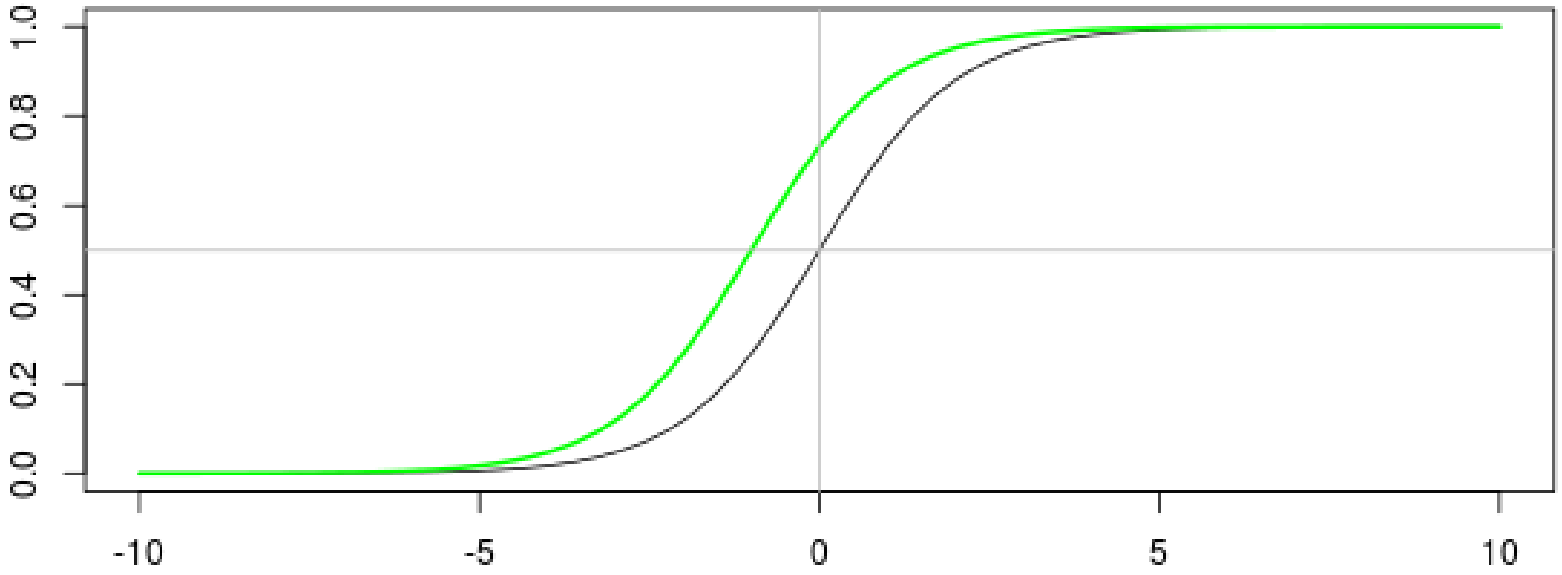
La fonction logistique

$$F(x) = \frac{\exp[-x]}{1+\exp[-x]}, \forall x \in \mathbb{R}.$$



La fonction logistique

$$F(x) = \frac{\exp[1+x]}{1+\exp[1+x]}, \forall x \in \mathbb{R}.$$



La régression logistique

La log-vraisemblance est ici

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

La résolution se fait en écrivant les conditions du premier ordre. Pour cela, on écrit le gradient $\nabla \log \mathcal{L}(\boldsymbol{\beta}) = [\partial \log \mathcal{L}(\boldsymbol{\beta})/\partial\beta_k]$, où

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

or compte tenu de la forme de $\pi_i(\boldsymbol{\beta})$,

$$\frac{\partial \pi_i(\boldsymbol{\beta})}{\partial \beta_k} = \pi_i(\boldsymbol{\beta})[1-\pi_i(\boldsymbol{\beta})]X_{k,i}$$

on obtient

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

Newton Raphson - 'Descente' de gradient

Remarque Pour rechercher le zéro d'une fonction (réelle), i.e. $f(x) = 0$: on se donne x_0, et on pose

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}$$

Remarque Pour rechercher le maximum d'un fonction dérivable, on recherche le zéro de la dérivée, i.e. $h'(x) = 0$: on se donne x_0, et on pose

$$x_k = x_{k-1} - \frac{f'(x_{k-1})}{f''(x_{k-1})}$$

Il faut trouver numériquement le maximum de $\log \mathcal{L}(\boldsymbol{\beta})$.

L'algorithme standard est

1. partir d'une valeur initiale $\boldsymbol{\beta}_0$
2. poser $\boldsymbol{\beta}_k = \boldsymbol{\beta}_{k-1} - H(\boldsymbol{\beta}_{k-1})^{-1} \nabla \log \mathcal{L}(\boldsymbol{\beta}_{k-1})$

où $\nabla \log \mathcal{L}(\boldsymbol{\beta})$ est le gradient, et $H(\boldsymbol{\beta})$ la matrice Hessienne (on parle parfois de Score de Fisher).

Ici, le terme générique de la matrice Hessienne est

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

Posons

$$\boldsymbol{\Omega} = [\omega_{i,j}] = \text{diag}(\widehat{\pi}_i(1 - \widehat{\pi}_i))$$

alors le gradient est

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

alors que la matrice Hessienne estimation

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

Aussi, l'algorithme de Newton Raphson s'écrit

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

On reconnait une régression pondérée itérée,

$$\boldsymbol{\beta}_k = \text{argmin}\,\{(\boldsymbol{Z} - \boldsymbol{X}\boldsymbol{\beta})\mathsf{T}\boldsymbol{\Omega}(\boldsymbol{Z} - \boldsymbol{X}\boldsymbol{\beta}).\}$$

D'un point de vue computationnel, l'estimation se fait de la manière suivante:

```
X=cbind(rep(1,nrow(avocat)),avocat$LOSS)
Y=(avocat$ATTORNEY=2)*1
```

on part d'une valeur initiale (e.g. une estimation classique de modèle linéaire)

```
beta=lm(Y~0+X)$coefficients
```

On fait ensuite une boucle,

```
for(s in 1:20){
pi=exp(X%*%beta)/(1+exp(X%*%beta))
```

```
gradient=t(X)%*%(Y-pi)
omega=matrix(0,nrow(X),nrow(X));diag(omega)=(pi*(1-pi))
hesienne=-t(X)%*%omega%*%X
beta=beta-solve(hesienne)%*%gradient}
```

Ici, on obtient les valeurs suivantes

```
                [,1]         [,2]
 [1,]  0.50034948 0.001821554
 [2,] -0.05689403 0.015434935
 [3,] -0.19941087 0.055570172
 [4,] -0.45629828 0.140490153
 [5,] -0.73627462 0.253154397
 [6,] -0.85830164 0.309712845
 [7,] -0.87192186 0.316615292
 [8,] -0.87207800 0.316697581
```

i.e. on converge (et vite).

Comportement asymptotique (et intervalle de confiance)

On peut montrer que $\widehat{\boldsymbol{\beta}} \xrightarrow{\mathbb{P}} \boldsymbol{\beta}$ et

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

Numériquement, la encore, on peut approcher $I(\boldsymbol{\beta})^{-1}$ qui est la variance (asymptotique) de notre estimateur. Or $I(\boldsymbol{\beta}) = H(\boldsymbol{\beta})$, donc les écart-types de $\widehat{\boldsymbol{\beta}}$ sont

```
> pi=exp(X%*%beta)/(1+exp(X%*%beta))
+ omega=matrix(0,nrow(X),nrow(X));diag(omega)=(pi*(1-pi))
+ hesienne=-t(X)%*%omega%*%X
+ sqrt(-diag(solve(hesienne)))
[1] 0.09128379 0.02902411
```

On retrouve toutes ces valeurs en utilisant

```
> logistique <- glm((ATTORNEY==1)~LOSS, data=avocat,family=binomial(
    link="logit"))

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.87208    0.09128  -9.553   <2e-16 ***
LOSS         0.31670    0.02902  10.912   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1857.6  on 1340  degrees of freedom
Residual deviance: 1609.4  on 1338  degrees of freedom
AIC: 1613.4

Number of Fisher Scoring iterations: 8
```

Test d'hypothèse et significativité

Considérons un test $H_0 : \beta_k = 0$ contre $H_1 : \beta_k \neq 0$ (classique dans le modèle linéaire).

Compte tenu du comportement asymptotiquement Gaussien des estimateurs, on a automatiquement un test de significativité des coefficients: sous H_0,

$$\frac{\widehat{\beta}_k}{\sqrt{\widehat{\text{Var}}(\widehat{\beta}_k)}} \sim \mathcal{N}(0,1).$$

Remarque: ce test correspond à un test de Wald (d'où la notation `z` sous R)

Régression linéaire sur une variable catégorielle

Supposons que l'on régresse sur X, une variable binaire, i.e $X \in \{0, 1\}$, avec

$$\mathbb{P}(Y = 1|X) = \beta_0 + \beta_1 X$$

qui prend les valeurs β_0 si $x = 0$ (modalité de référence) et $\beta_0 + \beta_1$ si $x = 1$.

La log-vraisemblance devient

$$\begin{aligned}\log \mathcal{L}(\beta_0, \beta_1) &= \sum_{i, x_i=0} y_i \log[\beta_0] + \sum_{i, x_i=1} y_i \log[\beta_0 + \beta_1] \\ &\quad + \sum_{i, x_i=0} (1 - y_i) \log[1 - \beta_0] + \sum_{i, x_i=1} (1 - y_i) \log[1 - \beta_0 + \beta_1]\end{aligned}$$

et les conditions du premier ordre donnent

$$\widehat{\beta_0} = \frac{\sum_{x_i=0} y_i}{\sum_{x_i=0} 1} \text{ et } \widehat{\beta_1} = \frac{\sum_{x_i=1} y_i}{\sum_{x_i=0} 1} - \frac{\sum_{x_i=0} y_i}{\sum_{x_i=0} 1}$$

Régression linéaire sur une variable catégorielle

Sur nos données, on peut considérer le sexe comme variable explicative.

```
> xtabs(~ATTORNEY+CLMSEX,data=avocat)
          CLMSEX
ATTORNEY    M    F
       1  325  352
       2  261  390
```

Si on fait la régression linéaire

```
> reglm = lm((ATTORNEY==1) ~ CLMSEX, data=avocat)
> summary(reglm)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.55461    0.02060  26.922  < 2e-16 ***
CLMSEXF     -0.08021    0.02756  -2.911  0.00367 **
```

Régression linéaire sur une variable catégorielle

Ces paramètres correspondent aux fréquences empiriques,

```
> attach(avocat)
> sum((ATTORNEY==1)&(CLMSEX=="M"),na.rm=TRUE)/sum(CLMSEX=="M",na.rm=
    TRUE)
[1] 0.5546075
> sum((ATTORNEY==1)&(CLMSEX=="F"),na.rm=TRUE)/sum(CLMSEX=="F",na.rm=
    TRUE)
[1] 0.4743935
```

Régression logistique sur une variable catégorielle

Avec un modèle logistique, on obtient les mêmes prédictions,

```
> reglogit = glm((ATTORNEY==2) ~ CLMSEX, data=avocat,family=binomial)
> summary(reglogit)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.21930    0.08312  -2.639  0.00833 **
CLMSEXF      0.32182    0.11097   2.900  0.00373 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> exp(reglogit$coefficients[1])/(1+exp(reglogit$coefficients[1]))
[1] 0.4453925
> exp(sum(reglogit$coefficients[1:2]))/(1+exp(sum(reglogit$
    coefficients[1:2])))
[1] 0.5256065
```

Modèle alternatif, le modèle probit

Soit H une fonction de répartition d'une variable rélle, $H : \mathbb{R} \mapsto [0, 1]$. On suppose ici

$$\pi_i = H(\boldsymbol{X}_i'\boldsymbol{\beta}) \text{ ou } \boldsymbol{X}_i'\boldsymbol{\beta} = H^{-1}(\pi_i)$$

- si H est la fonction de répartition de la loi logistique, on retrouve le modèle logit,
- si H est la fonction de répartition de la loi $\mathcal{N}(0, 1)$, on obtient le modèle probit,

Remarque: probit signifie probability unit.

On suppose qu'il existe une variable latente $Y_i^\star$ non observée, telle que

$$\pi_i = \mathbb{P}(Y_i = 1) = \mathbb{P}(Y_i^\star > 0)$$

On suppose que $Y_i^\star = \boldsymbol{X}_i'\boldsymbol{\beta} + \epsilon_i$, où ϵ_i est une erreur de loi H. Alors

$$\pi_i = \mathbb{P}(Y_i = 1) = \mathbb{P}(Y_i^\star > 0) = \mathbb{P}(\epsilon_i > -\boldsymbol{X}_i'\boldsymbol{\beta}) = 1 - H(-\boldsymbol{X}_i'\boldsymbol{\beta})$$

alors que

$$1-\pi_i = \mathbb{P}(Y_i = 0) = \mathbb{P}(Y_i^\star \leq 0) = \mathbb{P}(\epsilon_i \leq -\boldsymbol{X}_i'\boldsymbol{\beta}) = H(-\boldsymbol{X}_i'\boldsymbol{\beta})$$

Si $H = \Phi$, alors H est symmétrique par rapport à 0, et

$$\pi_i = \mathbb{P}(Y_i = 1) = 1 - \Phi(-\boldsymbol{X}_i'\boldsymbol{\beta}) = \Phi(\boldsymbol{X}_i'\boldsymbol{\beta}),$$

ou encore $\eta_i = \boldsymbol{X}_i'\boldsymbol{\beta} = \Phi^{-1}(\pi_i)$.

Si $H(x) = \dfrac{e^x}{1+e^x}$, on a une loi logistique, qui est aussi symmétrique par rapport à 0, et

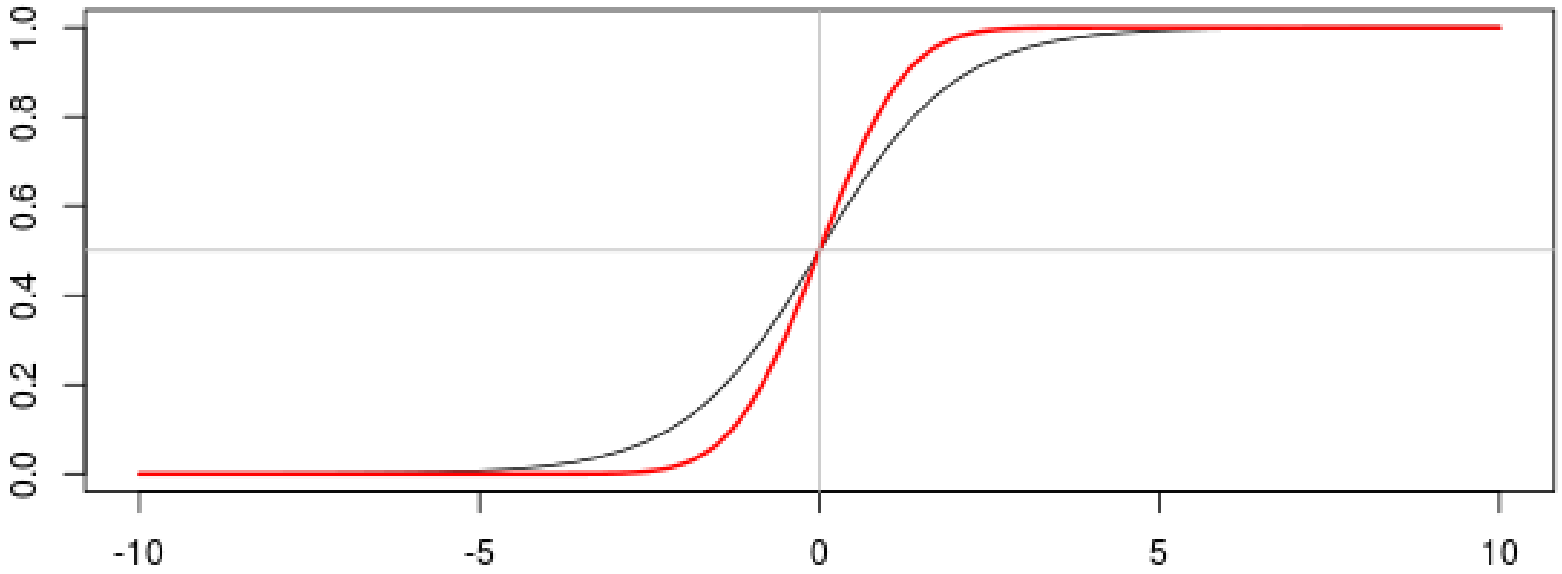
$$\pi_i = \mathbb{P}(Y_i = 1) = 1 - H(-\boldsymbol{X}_i'\boldsymbol{\beta}) = H(\boldsymbol{X}_i'\boldsymbol{\beta}),$$

ou encore $\eta_i = \boldsymbol{X}_i'\boldsymbol{\beta} = H^{-1}(\pi_i)$.

Remarque: pourquoi prendre un seuil nul ? sinon le modèle ne serait pas identificable. De même si on ne fixe pas la variance des résidus. On prend donc (arbitrairement) $s = 0$ et $\sigma^2 = 1$.

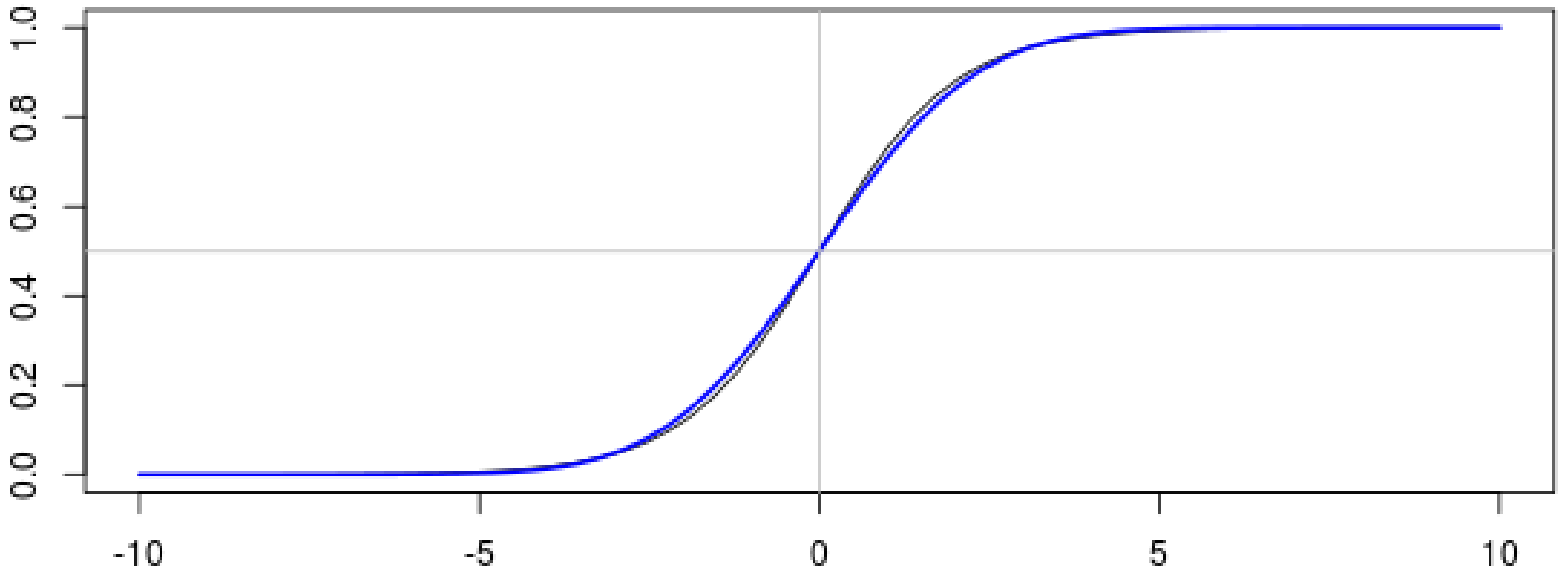
La fonction logistique

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{z^2}{2}\right) dz, \forall x \in \mathbb{R}.$$



La fonction logistique

$$\Phi(x) = \frac{\sqrt{3}}{\pi\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{3z^2}{2\pi^2}\right) dz, \forall x \in \mathbb{R}.$$



Qualité de l'ajustement

Pour juger de la validité du modèle, on peut considérer plusieurs quantités.

On peut définir la déviance, qui compare la log-vraisemblance du modèle et cellle du modèle parfait (cf. section sur les GLMs),

$$D = -2\sum_{i=1}^{n}[Y_i \log[\widehat{\pi}_i] + (1-Y_i)\log[1-\widehat{\pi}_i]]$$

Plus la variance est faible, meilleur est le modèle .

On peut aussi considérer la statistique de Pearson

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i - \widehat{\pi}_i)^2}{\widehat{\pi}_i[1-\widehat{\pi}_i]}$$

Étude des résidus

A l'aide de la relation précédante, on peut définir

$$\widehat{\varepsilon}_i = \frac{Y_i - \widehat{\pi}_i}{\sqrt{\widehat{\pi}_i[1-\widehat{\pi}_i]}}$$

On parlera de résidus de Pearson.

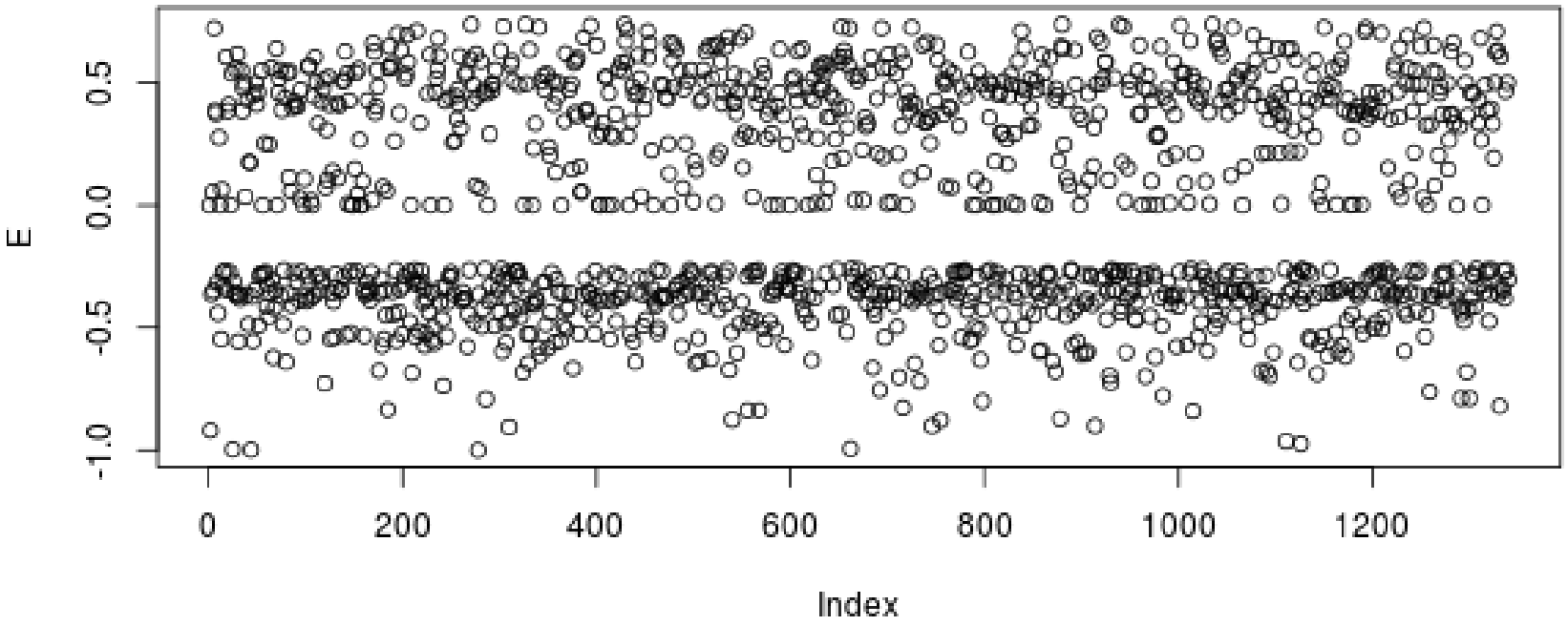
Il existe aussi les résidus de déviance.

$$\widehat{\varepsilon}_i = \pm[Y_i \log[\widehat{\pi}_i] + (1-Y_i)\log[1-\widehat{\pi}_i]]^{\frac{1}{2}}$$

avec un signe positif si $Y_i \geq \widehat{\pi}_i$, négatif sinon.

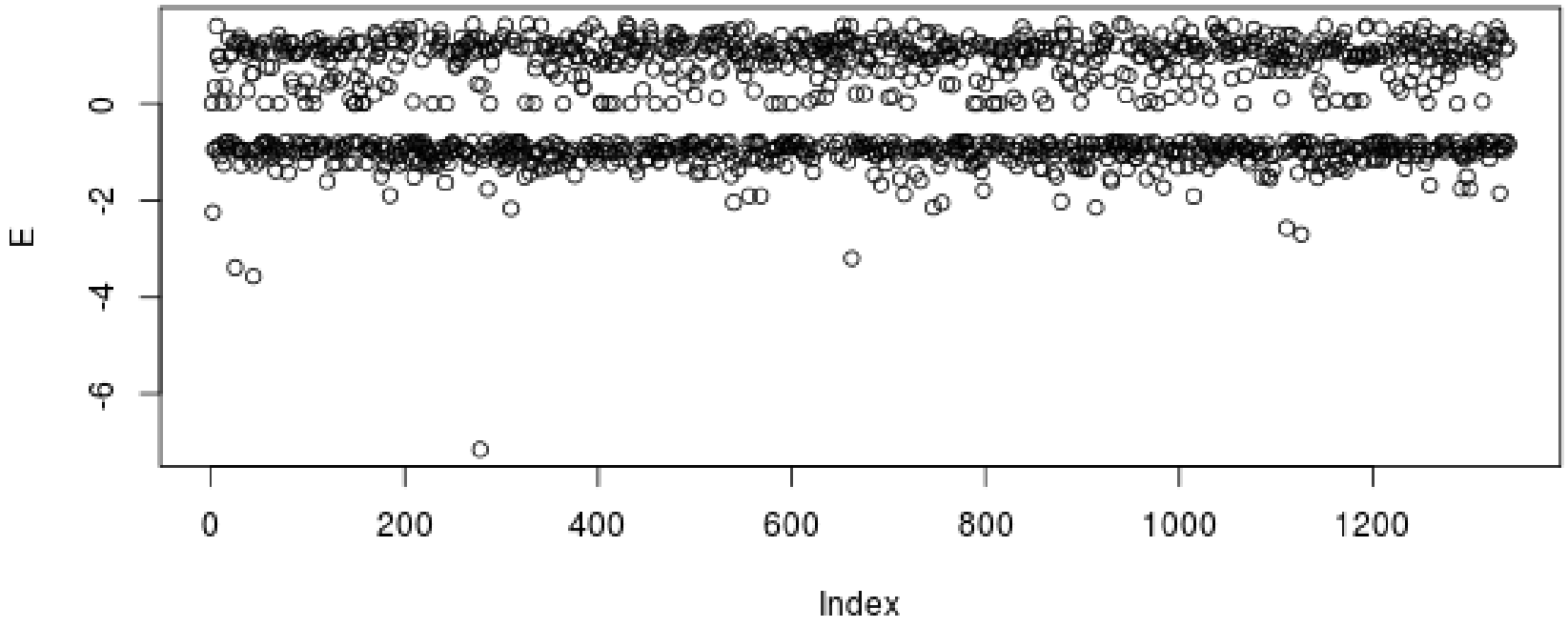
Visualisation des résidus

```
> reglogit = glm((ATTORNEY==1) ~ CLMSEX+LOSS, data=avocat,family=
    binomial)
> E = residuals(reglogit,type="response")
```



Visualisation des résidus

```
> reglogit = glm((ATTORNEY==1) ~ CLMSEX+LOSS, data=avocat,family=
    binomial)
> E = residuals(reglogit,type="deviance")
```



Prédiction avec un modèle logistique

Sous R, la prédiction sera (par défaut) une prédiction du score, i.e. $\boldsymbol{x}^\top\widehat{\boldsymbol{\beta}}$.

```
> predict(reglogit,newdata=data.frame(LOSS=10,CLMSEX="M"))
       1
2.530417
```

Il faut alors utiliser la loi logistique pour prévoir π,

```
> exp(predict(reglogit,newdata=data.frame(LOSS=10,CLMSEX="M")))/
+ (1+exp(predict(reglogit,newdata=data.frame(LOSS=10,CLMSEX="M"))))
         1
0.9262468
```

ou utiliser directement

```
> predict(reglogit,newdata=data.frame(LOSS=10,CLMSEX="M"),
+ type="response")
         1
0.9262468
```

Modélisation de la survenance de sinistre

On peut aussi modéliser la survenance de sinistres

```
> camping=read.table("http://freakonometrics.free.fr/baseN.txt",
+                    header=TRUE,sep=";")
> Y <- camping$nombre
> table(Y)

    0     1
60155  3264
```

i.e. 5.1467% des polices sont touchées par un sinistre.

```
> reg=glm(Y~1,family=binomial(link="logit"))
> predict(reg,type="response")[1]
          1
0.05146723
```

Prise en compte de l'exposition

Parfois, les polices d'assurance ne sont pas observées pendant un an, car les assurés peuvent les résilier. On peut malgré tout utiliser ces informations partielles (censurées, car on cherche une fréquence annuelle).

```
> E=camping$exposition
> table(E)
E
  0.2   0.25   0.33    0.5      1
    5     48    309   1804  61253
```

Si on suppose que la survenance d'un évènement est modélisé par un processus de Poisson (absence de mémoire), alors

$$\mathbb{P}(Y=0)=\mathbb{P}(N=0)^E$$

Si $N\sim\mathcal{B}(\pi)$,

$$\mathbb{P}(Y_i=y_i)=\pi^{e_i y_i}\left[1-\pi^{e_i}\right]^{1-y_i}$$

et la vraisemblance est alors

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

de telle sorte que

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

```
> logL = function(p){
+      -sum(log(dbinom(Y,size=1,prob=1-(1-p)^E)))
+    }
> opt=optimize(f=logL,interval=c(1e-5,1e-1))
> opt$minimum
[1] 0.05225447
```

Plus généralement, on peut alors une régression logistique, en modélisant la non

survenance de sinistre,

$$\mathbb{P}[Y=0|\boldsymbol{X},E]=\left(\frac{\exp(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta})}{1+\exp(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{\beta})}\right)^{E}$$

Si $\mathbb{P}[Y=0]\sim 0$,

$$\mathbb{P}[Y=1]=1-\mathbb{P}[Y=0]=1-(1-\mathbb{P}[N=1])^{E}$$

de telle sorte que

$$\mathbb{P}[Y=1]\sim 1-1+E\cdot\mathbb{P}[N=1]=E\cdot\mathbb{P}[N=1]$$

```
> reg=glm(Y~1+offset(log(E)),family=binomial(link="log"))
> predict(reg,type="response")[1]
          1
0.05212615
```

Prise en compte de l'exposition

Remarque ici, l'exposition est clairement endogène

```
> with(camping,table(nombre, exposition))
       exposition
nombre    0.2  0.25  0.33   0.5     1
     0      2    29   215  1230 58679
     1      3    19    94   574  2574
```

Régression logistique et variables catégorielles

Dans la base, on dispose de variables catégorielles, la situation familiale, le sexe, la marque du véhicule, le type de paiement pour la prime, etc.

Formellement, X_j prend les modalités $\{x_{j,1}, \cdots, x_{j,d_j}\}$, e.g. X_2,

$$\pi_i = \text{logit}^{-1}\left(\beta_0 + \beta_1 X_1 + \sum_{k=2}^{d_2} \beta_{2,k}\mathbf{1}(X_2 = x_{2,k})\right)$$

On perd généralement la première modalité,

```
> levels(camping$situationfamiliale)
[1] "Celiba"    "Marie"     "Veuf/Div"
```

Régression logistique et variables catégorielles

```
> reg=glm(nombre~ageconducteur+situationfamiliale,data=camping,family
    =binomial)
> summary(reg)

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)               -3.8556504  0.1095669 -35.190  < 2e-16 ***
ageconducteur              0.0184951  0.0018124  10.205  < 2e-16 ***
situationfamilialeMarie   -0.2028149  0.0484682  -4.184 2.86e-05 ***
situationfamilialeVeuf/Div -0.0006492  0.0992486  -0.007    0.995
```

Fusion de modalités pour une variable catégorielle

```
> levels(camping$situationfamiliale)=c("Celib/Veuf/Div","Marie","
    Celib/Veuf/Div")
> reg=glm(nombre~ageconducteur+situationfamiliale,data=camping,family
    =binomial)
> summary(reg)

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)             -3.855646   0.109566 -35.190  < 2e-16 ***
ageconducteur            0.018493   0.001779  10.392  < 2e-16 ***
situationfamilialeMarie -0.202681   0.043973  -4.609 4.04e-06 ***
```

Régression et variables continues, *natura non facit saltus*

Les variables continues apparaissent forcément de manière linéaire

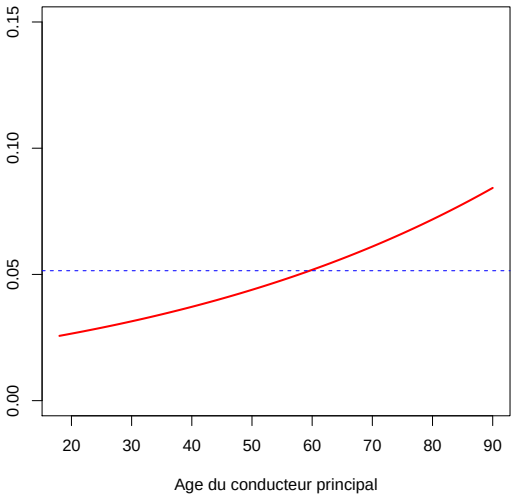
$$\pi_i = \text{logit}^{-1}(\beta_0 + \beta_1 x_1) \sim \exp(\beta_0 + \beta_1 x_1)$$

```
> reg=glm(nombre~ageconducteur,data=camping,family=binomial)
> summary(reg)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.949435   0.108492  -36.40   <2e-16 ***
ageconducteur  0.017375   0.001771    9.81   <2e-16 ***
```

Régression et variables continues, *natura non facit saltus*

```
> reg=glm(nombre~ageconducteur, data=camping,
    family=binomial)
```



Régression et variables continues

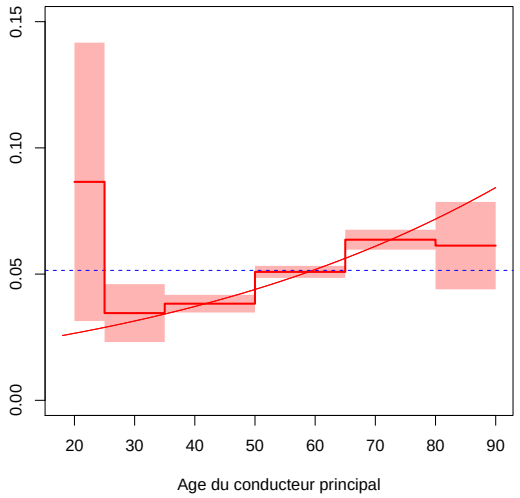
Dans la base, on dispose de variables catégorielles, mais aussi 'continues' comme l'âge du conducteur, l'ancienneté du permis, l'âge du véhicule.

Historiquement, les assureurs aimaient constituer des classes de risques, et donc cherchaient à découper les variables continues en classes. On peut considérer un découpage arbitraire en classes d'âges, e.g.

$$[18, 25),\ [25, 35),\ [35, 50),\ [50, 65),\ [65, 80),\ [80, 100).$$

Régression et variables continues, *natura non facit saltus*

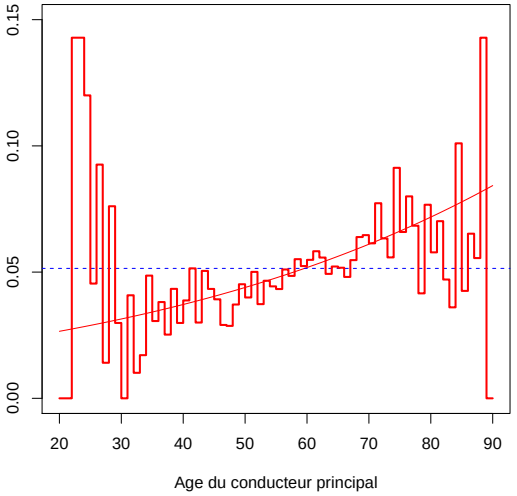
```
> AGE=c(18,25,35,50,65,80,100)
> cut_logistic = glm(nombre~cut(ageconducteur,
    breaks=AGE),data=camping,family=binomial)
```



On a l'impression que le modèle linéaire n'est peut-être pas adapté....

Régression et variables continues, *natura non facit saltus*

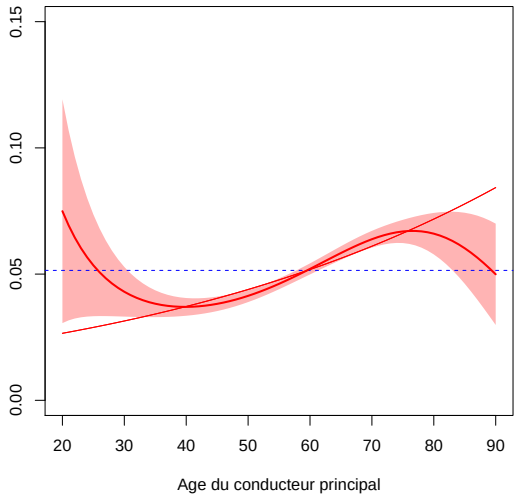
```
> f_logistic = glm(nombre~as.factor(ageconducteur)
    ,data=camping,family=binomial)
```



En prenant l'âge comme variable catégorielle, on retrouve l'effet non linéaire

Régression et variables continues, *natura non facit saltus*

```
> library(splines)
> bs_logistic = glm(nombre~bs(ageconducteur),data=
    camping,family=binomial(link="log"))
```

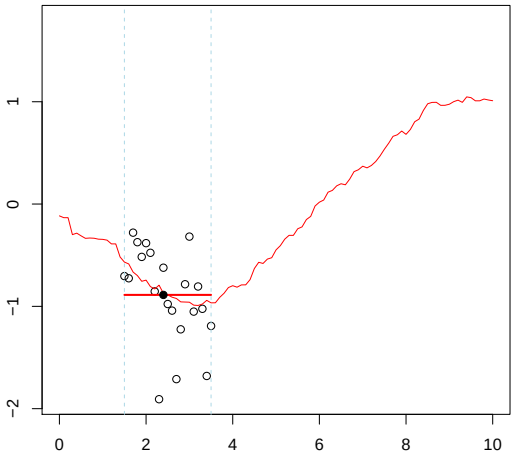


L'utilisation de splines semble confirmer la non-linéarité.

Lissage et régression locale

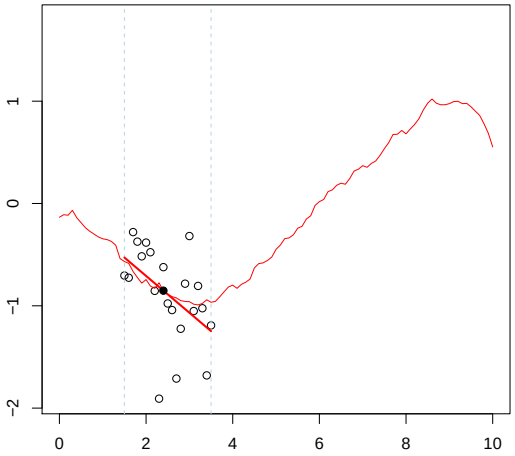
Si on veut prédire $\mathbb{E}[Y|\boldsymbol{X}=\boldsymbol{x}]$, on peut construire un modèle au voisinage de $\boldsymbol{x}$

```
> h=1
> w=(abs(db$x-x)<=h)*1
> regl=lm(y~1,data=db,weights=w)
```



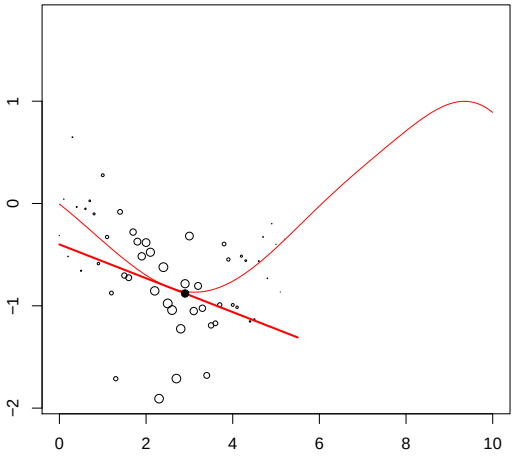
Lissage et régression locale

```
> h=1
> w=(abs(db$x-x)<=h)*1
> regl=lm(y~x,data=db,weights=w)
```



Lissage et régression locale - à noyau

```
> h=1
> w=dnorm(db$x,x,sd=h)
> regl=lm(y~x,data=db,weights=w)
```



Prédicteur linéaire

Si les modèles sont non-linéaires, on construit souvent un prédicteur linéaire.
Dans le modèle linéaire

$$\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}} = \underbrace{\boldsymbol{X}[\boldsymbol{X}^\top\boldsymbol{X}]^{-1}\boldsymbol{X}^\top}_{\boldsymbol{H}}\boldsymbol{y}$$

Plus généralement, on a un prédicteur linéaire si la prévision $\widehat{y}$ au point $\boldsymbol{x}$ s'écrit

$$\widehat{y} = \widehat{m}(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{S}_{\boldsymbol{x},i} y_i = \boldsymbol{S}_{\boldsymbol{x}}^\top \boldsymbol{y}$$

où $\boldsymbol{S}_{\boldsymbol{x}}$ est une vecteur de poids (smoother vector), que l'on peut associer à une matrice $n \times n$ de lissage

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

où les prévisions sont faites aux points $\boldsymbol{x}_i$'s.

Degrés de liberté et complexité de modèle

Pour le modèle linéaire

$$\boldsymbol{S}_{\boldsymbol{x}} = \boldsymbol{H}_{\boldsymbol{x}} = \boldsymbol{X}[\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}]^{-1}\boldsymbol{x}$$

Notons que

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

est souvent utiliser pour comme statistique d'un test de linéarité: sous l'hypothèse de modèle linéaire, T suit une loi de Fisher.

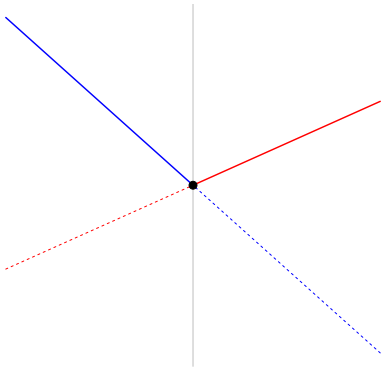
Pour les prédicteurs linéaire, trace($\boldsymbol{S}$) est souvent appelé nombre équivalent de paramètres et peut être relié à $n-$ nombre de degrés de liberté effectifs (cf Ruppert *et al.* (2003)).

Lissage, polynômes et b-splines

De manière générale, on va décomposer $\mathbb{E}[Y|X=x]$ dans une base de fonctions

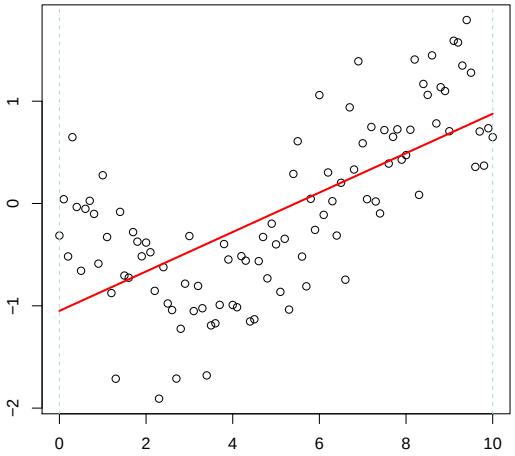
$$m(\boldsymbol{x}) = \sum_{m=0}^{M} \beta_M h_m(\boldsymbol{x}),$$

On va retrouver $h_m(x) = x^m$ pour une régression polynômiale, ou $h_m(x) = (x - s_m)_+$ pour des ensembles de noeuds s_m (avec ici des splines linéaires, mais on peut considérer des splines quadratiques ou cubiques).



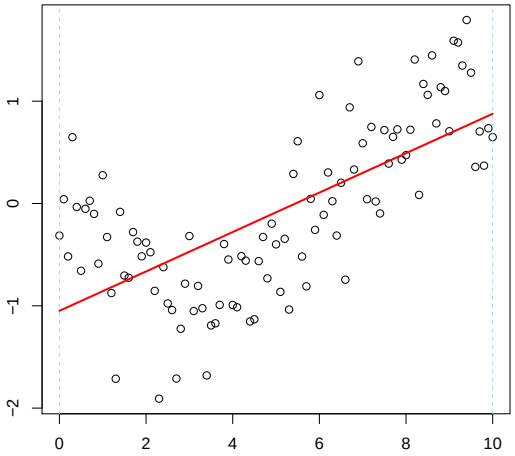
Lissage et régression polynômiale

```
> regp=lm(y~poly(x,M),data=db)
```



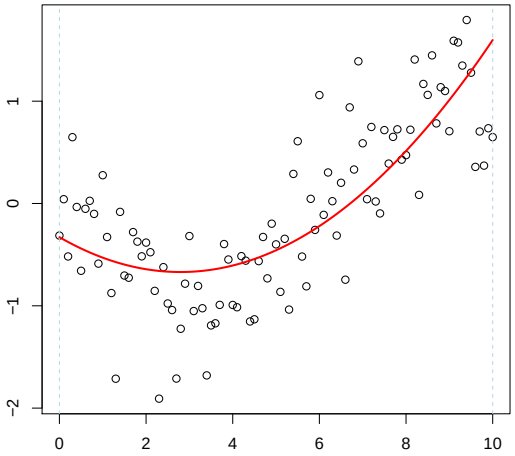
Lissage et régression par splines linéaires

```
> bs_reg_lin=lm(y~bs(x,df=M,degree=1),data=db)
```



Lissage et régression par splines quadratiques

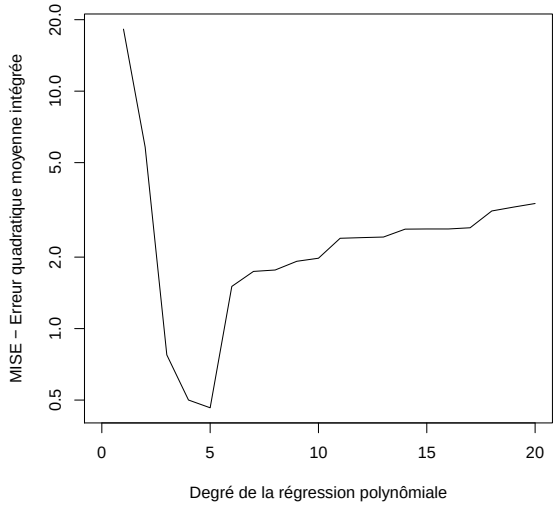
```
> bs_reg_quad=lm(y~bs(x,df=M,degree=2),data=db)
```

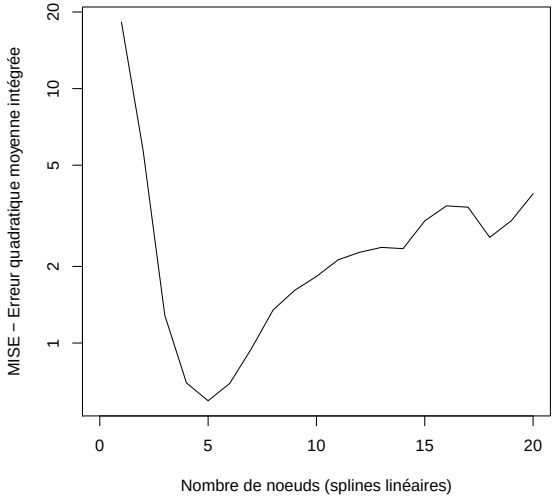


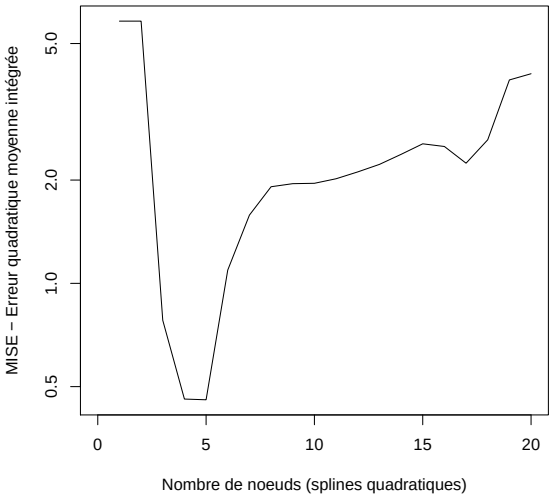
Lissage et tradeoff biais-variance

Problème classique en statistique : choix de M (degré du polynôme ou nombre de noeud), tunning parameter,

$$\text{mse}_M = \frac{1}{n}\sum_{i=1}^{n}[y_i - \widehat{m}_M(x_i)]^2 = \text{variance} + \text{biais}^2$$







Lissage et surapprentissage

- M trop grand: variance importante, biais faible
- M trop faible: biais important, variance faible

On retrouve ici l'idée de surapprentissage avec un modèle trop complexe. Autre soucis,

$$\text{mse}_M = \frac{1}{n}\sum_{i=1}^{n}[y_i - \widehat{m}_M(\boldsymbol{x}_i)]^2 \to ?$$

On ne peut invoquer la loi de grands nombres car $\widehat{m}_M(\cdot)$ dépend de tous les $(\boldsymbol{x}_i, y_i)$.

Surapprentissage et validation croisée

Considérons un échantillon $\boldsymbol{x} = (x_1, \cdots, x_n)$ et une statistique $\widehat{\theta}$. Posons $\widehat{\theta}_n = \widehat{\theta}(\boldsymbol{x})$

Le Jackknife est utilié pour réduire le biais: posons $\widehat{\theta}_{(-i)} = \widehat{\theta}(\boldsymbol{x}_{(-i)})$, et

$$\tilde{\theta} = \frac{1}{n}\sum_{i=1}^{n} \widehat{\theta}_{(-i)}$$

$$\text{Si } \mathbb{E}(\widehat{\theta}_n) = \theta + O(n^{-1}) \text{ alors } \mathbb{E}(\tilde{\theta}_n) = \theta + O(n^{-2}).$$

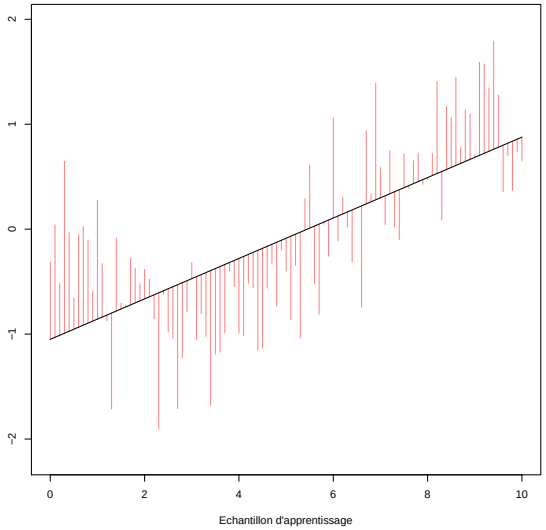
Cf idé de la validation croisée leave-one-out avec $\widehat{m}(\cdot)$

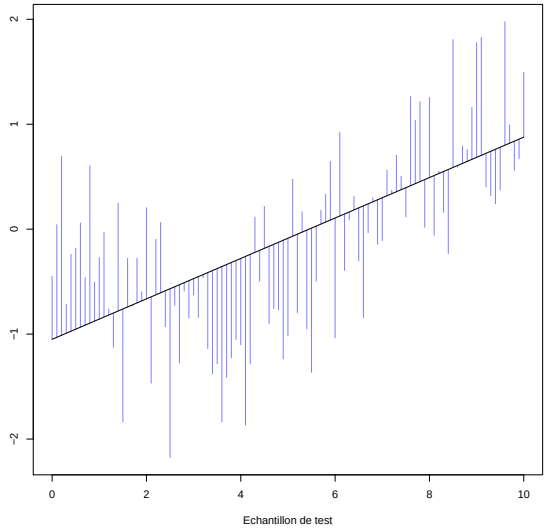
$$\text{mse} = \frac{1}{n}\sum_{i=1}^{n} [y_i - \widehat{m}_{(-i)}(x_i)]^2$$

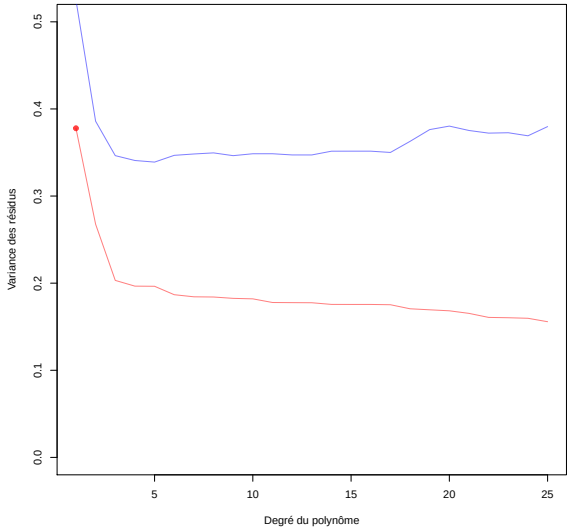
L'estimateur par Boostrap est obtenu par rééchantillonnage: posons $\widehat{\theta}_{(b)} = \widehat{\theta}(\boldsymbol{x}_{(b)})$, et $\tilde{\theta} = \frac{1}{n}\sum_{i=1}^{n} \widehat{\theta}_{(b)}$, où $\boldsymbol{x}_{(b)}$ est un échantillon de taille n, obtenu par tirage avec remise dans $\{x_1, \cdots, x_n\}$. Cf Efron (1979).

Modèle Prédictif et Généralisation

Autre approche classique, utiliser une partie des données pour apprendre, et l'autre pour valider le modèle. Le modèle ne doit pas (seulement) être bon sur la popuation d'apprentissage, il doit surtout l'être sur la population de test.





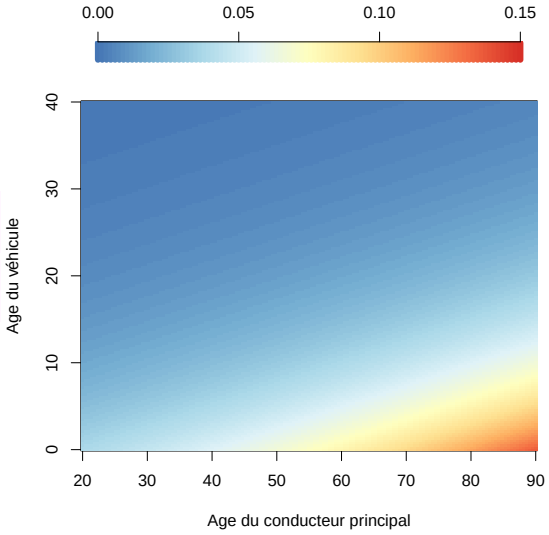


Effets croisés

Dans un modèle linéaire, avec deux variables, on considère une combinaison linéaire

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

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

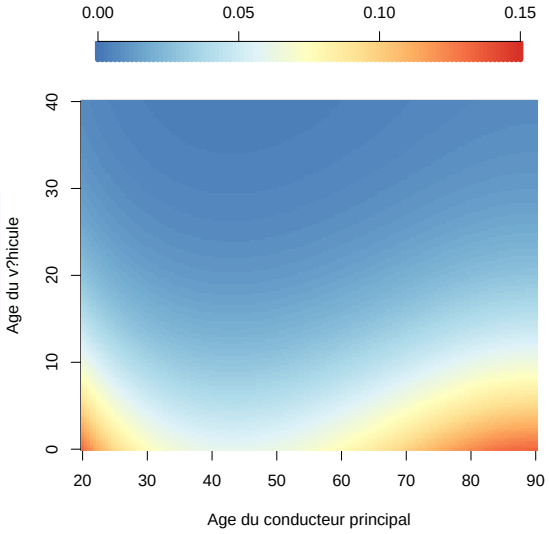


Effets croisés

En lissant les variables continues, le modèle reste additif

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

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

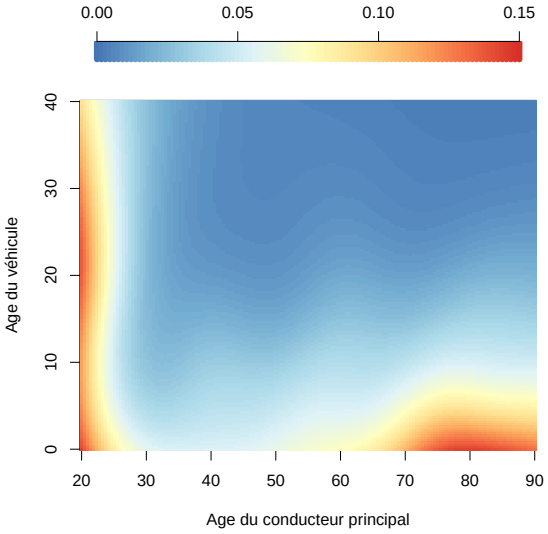


Effets croisés

On peut envisager un lissage bivarié

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

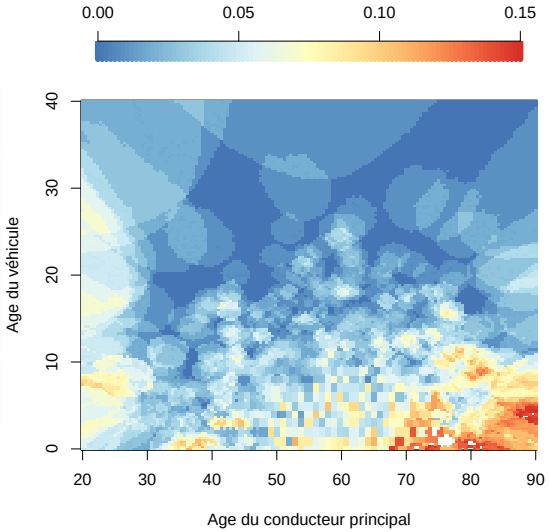
```
> library(mgcv)
> reg_gam=gam(nombre~s(ageconducteur,agevehicule),
    data=camping,family=binomial)
```



Les k-plus proches voisins

Parmi les modèles nonparamétriques classiques, on peut utiliser les k plus proches voisins

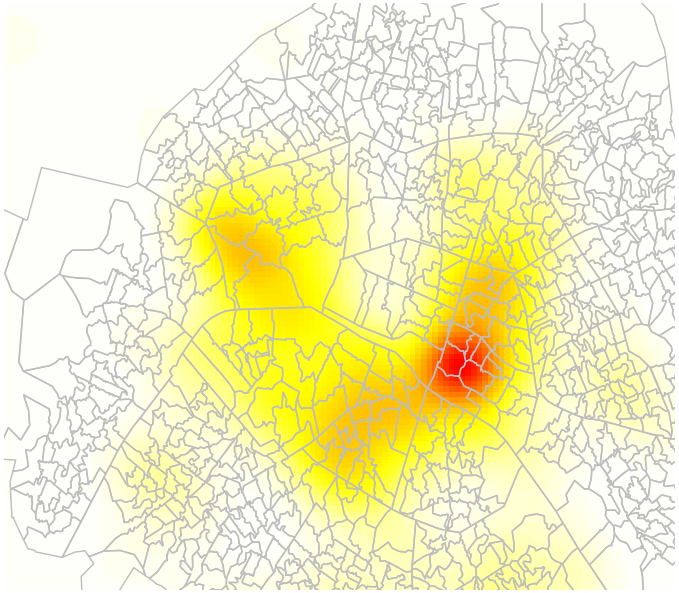
```
> library(caret)
> sc=sd(camping$ageconducteur)
> sv=sd(camping$agevehicule)
> knn=knn3((nombre==1)~I(ageconducteur/sc)+I(
    agevehicule/sv),data=camping,k=100)
```

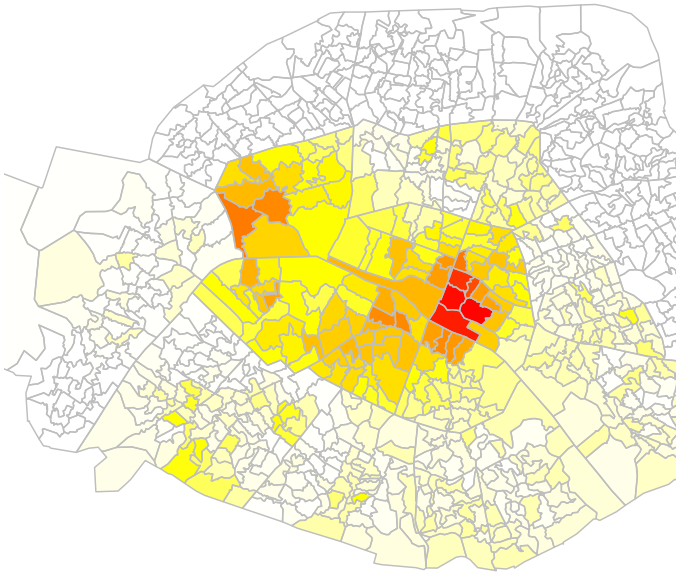


(attention, la distance ne doit pas être sensible aux unités...)

Lissage spatial

On peut aussi utiliser du lissage spatial, quand les données sont géolocalisées (lieu d'habitation, localisation d'un accident, etc), cf `kde()` de `library(ks)`, ou par k-plus proche voisin quand on dispose d'une zone géographique `knn2nb()` de `library(spdep)`





De la loi Binomiale à la loi Multinomiale

Pour une loi de Bernoulli, $y \in \{0, 1\}$,

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

Pour une loi multinomiale, $y \in \{A, B, C\}$

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

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

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

Courbe ROC, receiver operating characteristic

On dispose d'observations Y_i prenant les valeurs 0 (on dira **négatives**) ou 1 (on dira **positives**).

On a construit un modèle qui prédit π_i. En se fixant un seuil s, si $\widehat{\pi}_i \leq s$, on prédit $\widehat{Y}_i = 0$, et si $\widehat{\pi}_i > s$, on prédit $\widehat{Y}_i = 1$.

Le modèle sera bon si les positifs sont prédits positifs, et les négatifs sont prédits négatifs.

Le choix du seuil s permettra de minimiser soit les faux positifs, soit les faux négatifs.

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

où TP sont les True Positive, FP les False Positive.

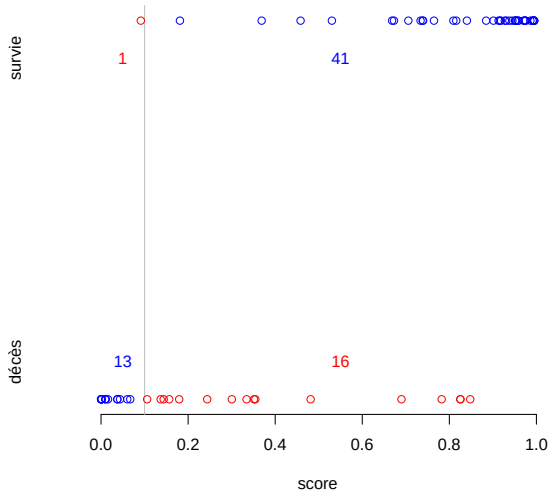
Courbe ROC et Qualité d'un Classifieur

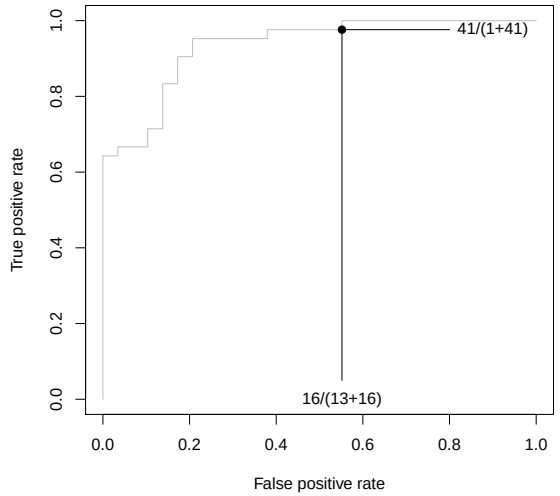
	$Y=0$	$Y=1$		prevalence
$\widehat{Y}=0$	true negative N_{00}	false negative (type II) N_{01}	negative predictive value NPV$=\frac{N_{00}}{N_{0\cdot}}$	false omission rate FOR$=\frac{N_{01}}{N_{0\cdot}}$
$\widehat{Y}=1$	false positive (type I) N_{10}	true positive N_{11}	false discovery rate FDR$=\frac{N_{10}}{N_{1\cdot}}$	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-				

Courbe ROC et Qualité d'un Classifieur

La courbe ROC est construite à partir de

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

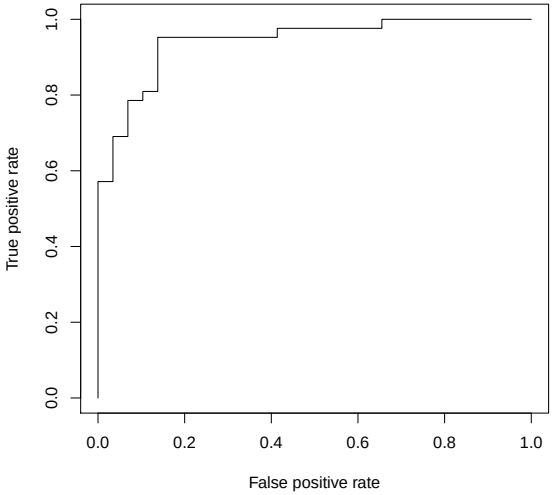


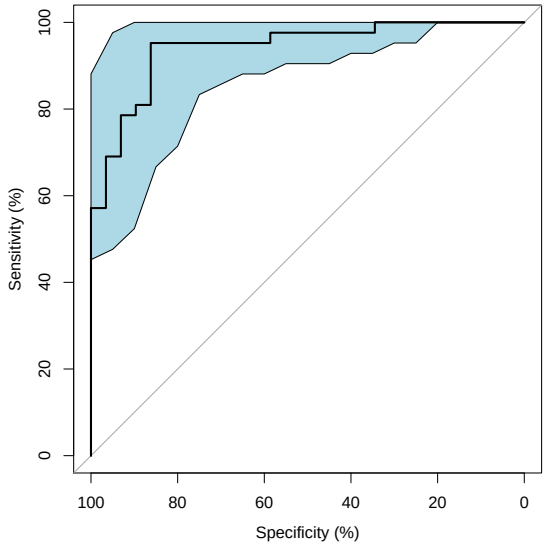


Courbe ROC

avec la régression logistique sur les infarctus

```
> logistic <- step(glm(F,data=camping,family=
    binomial))
> Y=camping$nombre
> S=predict(logistic,type="response")
> library(ROCR)
> pred=prediction(S,Y)
> plot(performance(pred,"tpr","fpr"))
> 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")
```





Mesures construites à partir d'une courbe ROC

L'aire sous la courbe ROC, Area Under the Curve AUC, peut être interprétée comme la probailité qu'une observation soit mieux prédite qu'une prédiction purement aléatoire cf Swets, Dawes & Monahan (2000)

Beaucoup d'autre mesures peuvent être construites

```
> 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
```

avec la H-mesure (cf hmeasure), Gini et AUC, ainsi que l'aire sous l'enveloppe convexe de la courbe (AUCH).

Mesures construites à partir d'une courbe ROC

La statistique κ compare une Observed Accuracy avec une Expected Accuracy (random chance), cf 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

On va comparer les matrices de confusion observée et aléatoire (cf. chi-deux)

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

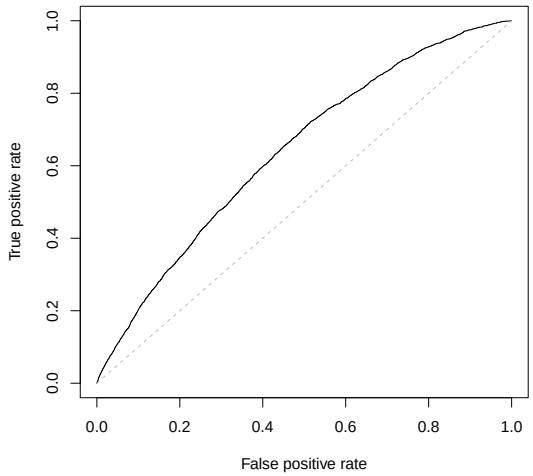
	$Y=0$	$Y=1$	
$\widehat{Y}=0$	11.44	16.56	28
$\widehat{Y}=1$	17.56	25.44	43
	29	42	71

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

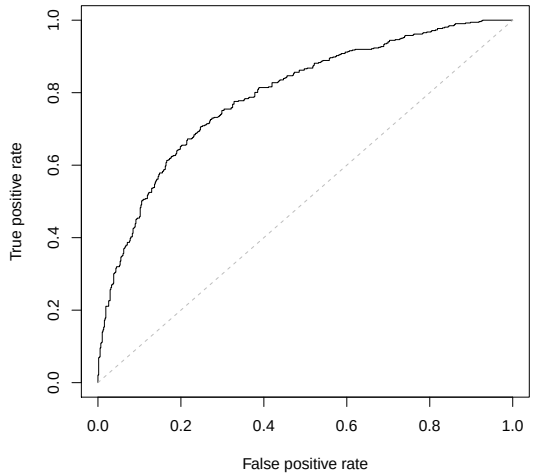
$$\text{random accuracy} = \frac{[TN+FP]\cdot[TP+FN]+[TP+FP]\cdot[TN+FN]}{n^2} \sim 51.93\%$$

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

```
> logistic <- glm(nombre~.,data=
    camping,family=binomial)
> Y=camping$nombre
> S=predict(logistic,type="
    response")
> pred=prediction(S,Y)
> plot(performance(pred,"tpr","
    fpr"))
```

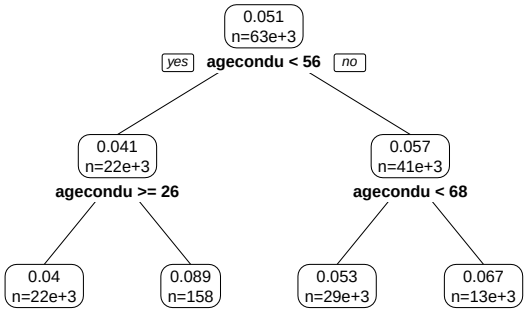



```
> logistic <- glm((ATTORNEY==2)~
    .,data=avocat,family=binomial
    )
> Y=avocat$ATTORNEY
> S=predict(logistic,type="
    response")
> pred=prediction(S,Y)
> plot(performance(pred,"tpr","
    fpr"))
```

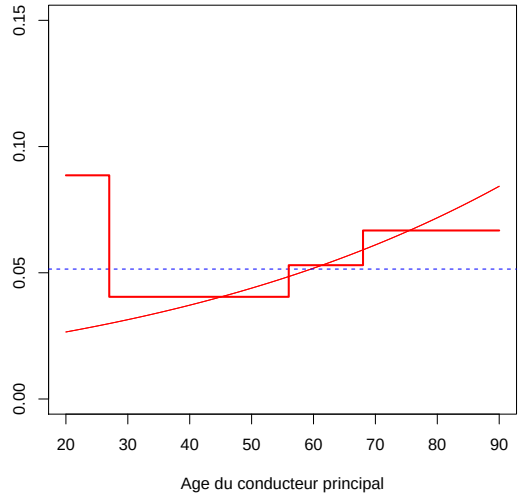


Les arbres de classification

Dans ces structures d'arbre, les feuilles représentent les valeurs de la variable-cible et les embranchements correspondent à des combinaisons de variables d'entrée qui mènent à ces valeurs (Source: wikipedia).




```
> library(rpart)
> tree=rpart((nombre==1)~ageconducteur,data=
    camping,cp=1e-4)
> library(rpart.plot)
> library(rattle)
> prp(tree,type=2,extra=1)
```



Les arbres de classification

Considérons une feuille $\{N\}$, alors l'indice de Gini sur la feuille est

$$\text{gini}(N) = \frac{n_{x,y}}{n_x}\left(1 - \frac{n_{x,y}}{n_x}\right)$$

et si $\{N\}$ et partagée en $\{N_L, N_R\}$,

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

Plus généralement, on dipose d'un indice d'impureté $\mathcal{I}(\cdot)$ et pour partager $\{N\}$ en $\{N_L, N_R\}$, on considère

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

e.g. l'indice de Gini (utilisé dans CART, Breiman *et al.* (1984)), $-p(1-p)$ ou l'indice d'entropie (utilisé dans C4.5 et C5.0) $p\log(p)$.

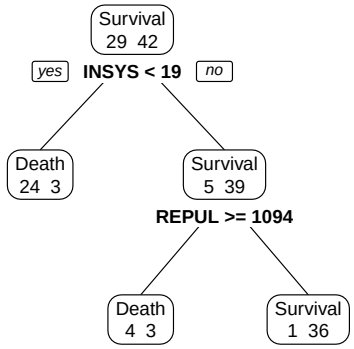
On considère des feuilles de la forme $N_L = \{x_{i,j} \leq s\}$ et $N_R = \{x_{i,j} > s\}$

Les arbres de classification

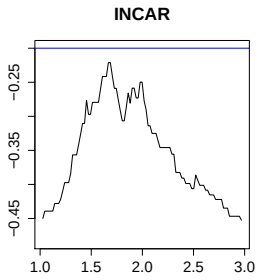
```
> library(rpart)
> cart<-rpart(PRONO~., data = myocarde)
> library(rpart.plot)
> library(rattle)
> prp(cart, type=2, extra=1)
```

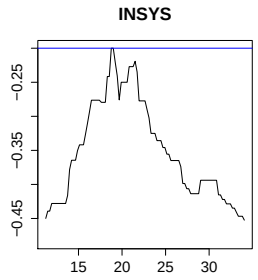
ou

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



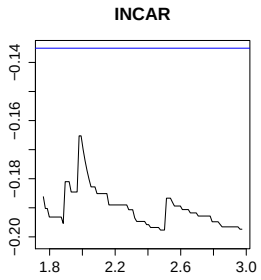
Les arbres de classification

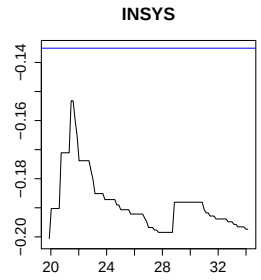


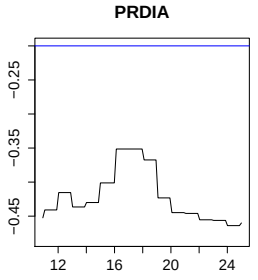


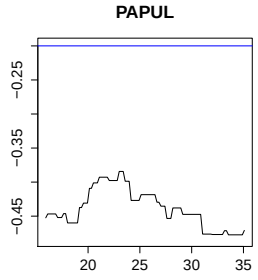
$$N_L\text{: } \{x_{i,j} \leq s\} \quad N_R\text{: } \{x_{i,j} > s\}$$

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

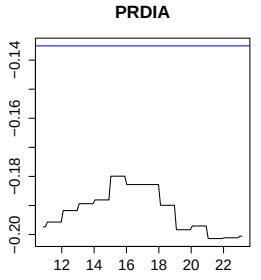


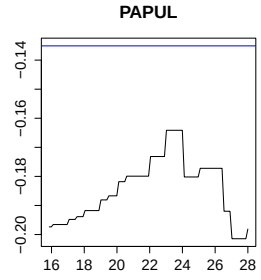


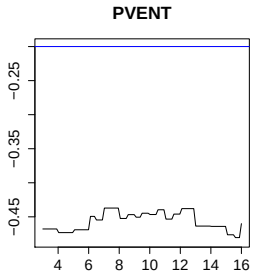


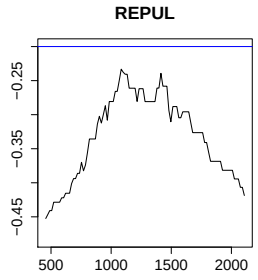


$\longleftarrow$ premier split

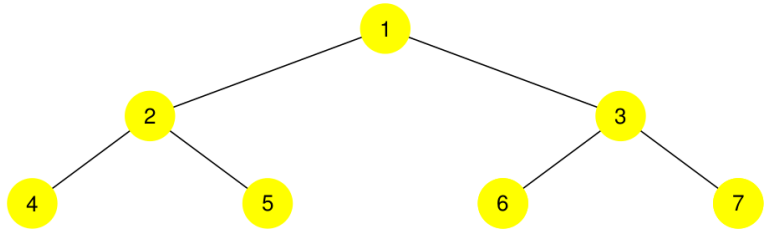


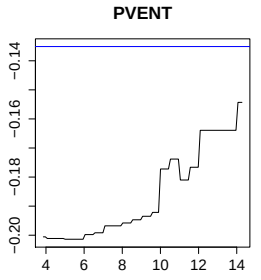


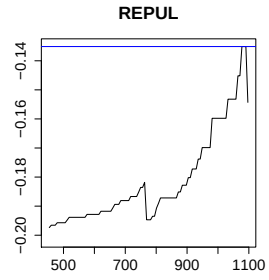




second split $\longrightarrow$







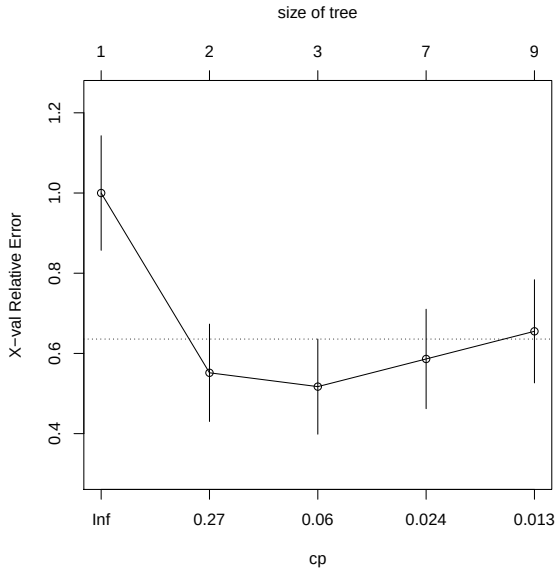
Taille des arbres (élagage, *pruning*)

On ne partage pas à un noeud si $\Delta\mathcal{I}$ est trop faible,

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

Remarque on note `cp` (complexity parameter) le seuil sur $\Delta\mathcal{I}(N_L, N_R)/\mathcal{I}(N)$ (gain relatif d'indice), cf Breiman *et al.* (1984).

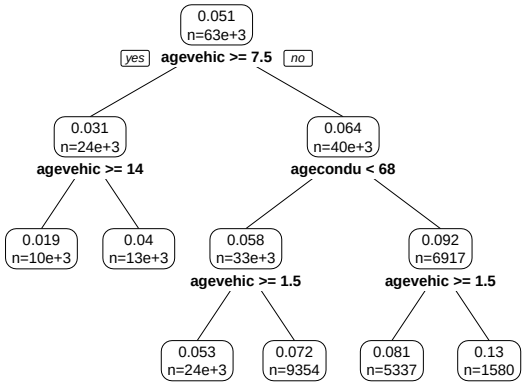
```
> cart <- rpart(PRONO ~ .,data = myocarde,
+ minsplit=3)
> plotcp(cart)
```



Les arbres de classification

En diminuant la valeur de `cp`, on peut construire un arbre plus profond (on peut aussi diminuer `mincut`)

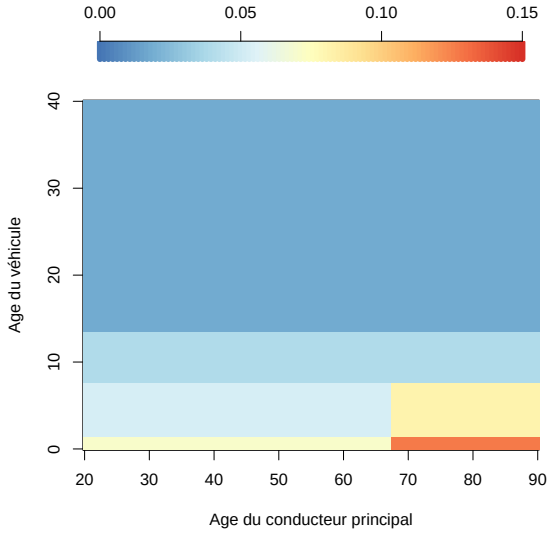
```
> tree=rpart((nombre==1)~.,data=camping,cp=7e-4)
> prp(tree,type=2,extra=1)
```

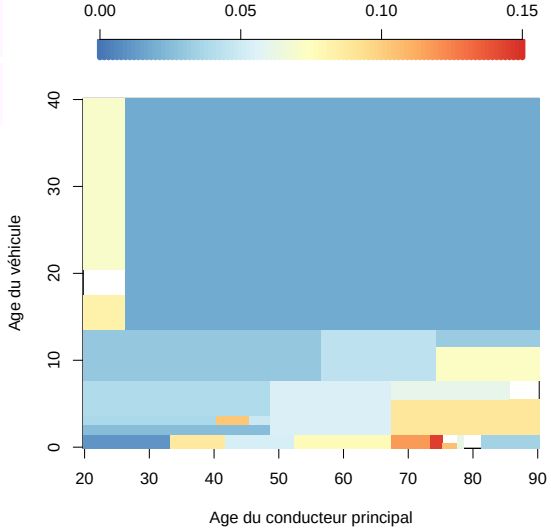


Effets croisés

On peut utiliser les arbres pour visualiser des possibles effets croisés entre deux variables continues

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





Méthodes de *Bagging* (ou ensachage)

Les arbres sont (relativement) peu robustes...

On peut construire plusieurs arbres de décision en ré-échantillonnant l'ensemble d'apprentissage [boostrap], puis en construisant les arbres par une procédure de consensus[aggregation] ou en moyennant les prédictions.

→ marche aussi pour des GLM (cf dernière section du cours) mais est très populaire sur les arbbres Breiman (1996)

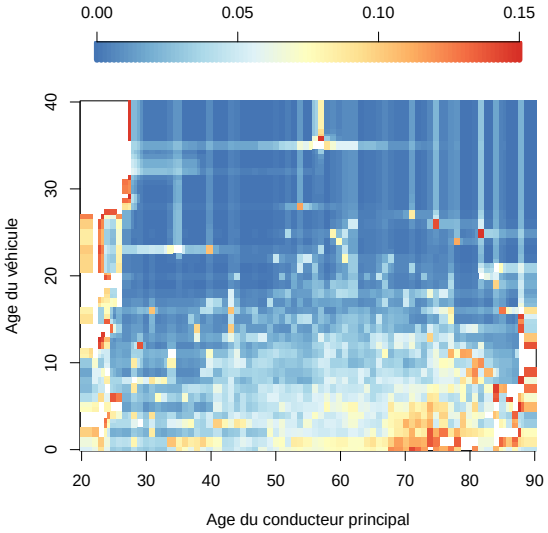
Le bagging permet de définir la notion de marge,

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

Remarque La probabilité que la ligne i ne soit pas dans un échantillon est $(1-n^{-1})^n \to e^{-1} \sim 36.8\%$, cf ratio training / validation (2/3-1/3)

Bagging

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



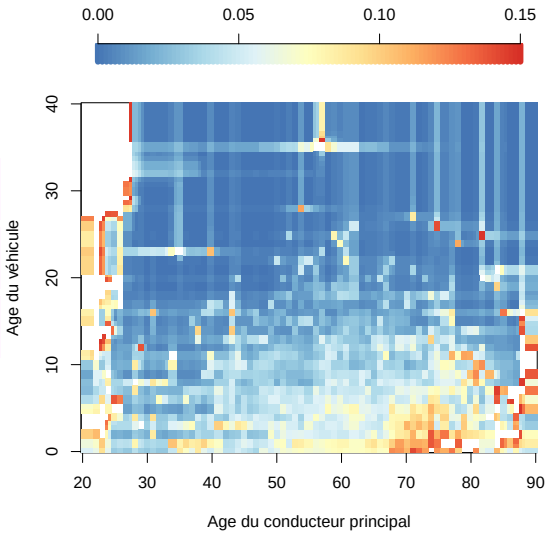
Les forêts aléatoires (*Random Forest*)

Générer des forêts en boostrappant puis en agrégeant pour obtenir une prévision glboale est du *bagging*.

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))

Par défaut, on sélection $\sqrt{k}$ variables explicatives.

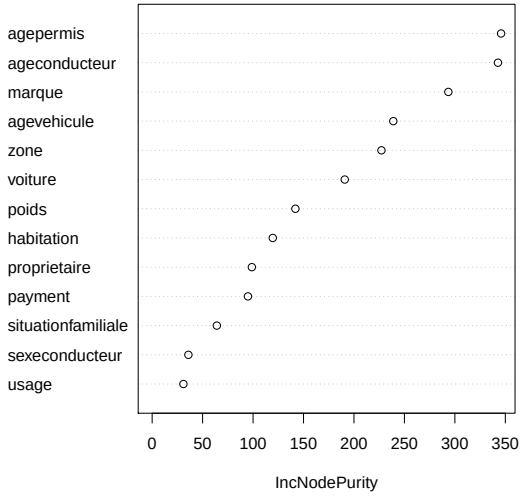
```
> library(randomForest)
> rf=randomForest((nombre==1)~ageconducteur+
    agevehicule,data=camping)
```



Variable Importance

Étant donnée une forêt composée de M arbres, on pose $I(X_k) = \frac{1}{M}\sum_m \sum_{t\in X_k} \frac{N_t}{N}\Delta\mathcal{I}(t)$ où la première somme est sur tous les arbres, et la seconde sur tous les noeuds splités suivant la variable X_k.

```
> RF=randomForest((nombre==1) ~ .,data = camping)
> varImpPlot(RF,main="")
> importance (RF)
                     IncNodePurity
ageconducteur            342.68711
agepermis                345.79078
sexeconducteur            35.90909
situationfamiliale        64.09047
habitation               119.52306
zone                     227.22830
agevehicule              238.98409
proprietaire              98.81689
payment                   94.92349
marque                   293.54250
poids                    141.96059
usage                     31.00314
```

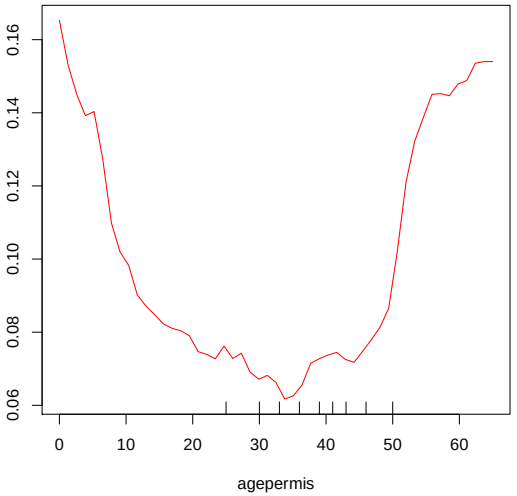


Partial Response Plots

On peut aussi construire, pour toutes les variables continues des 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, camping, eval(name), col="red",
    main="", xlab=name)
```



Gradient Boosting en Classification

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)

Avec des notations très générales, on souhaite résoudre

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

pour une fonction de perte ℓ (loss function). Heuristiquement, on construit itérativement des modèles sur les erreurs résiduelles

On ajuste un modèle sur $\boldsymbol{y}$, $m_1(\cdot)$ à partir de $\boldsymbol{y}$ et $\boldsymbol{X}$, et on définit l'erreur $\boldsymbol{\varepsilon}_1 = \boldsymbol{y} - m_1(\boldsymbol{X})$. On ajuste un modèle sur $\boldsymbol{\varepsilon}_1$, $m_2(\cdot)$ à partir de $\boldsymbol{\varepsilon}_1$ et $\boldsymbol{X}$, et on définit l'erreur $\boldsymbol{\varepsilon}_2 = \boldsymbol{\varepsilon}_1 - m_2(\boldsymbol{X})$, etc. Puis on pose

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

'Gradient' Boosting ?

Dans un modèle linéaire $\boldsymbol{y} - m_k(\boldsymbol{X})$ est un résidu, qui correspond au gradient de $\frac{1}{2}[y - m(\boldsymbol{x})]^2$

La descente de Gradient vient de l'expansion de Taylor

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

Mais là, ce que l'on considère, c'est

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

où ? pourrait être interprété comme un 'gradient'.

Boosting avec rétrécissement (*shrinkage*)

L'algorithme est ici

- on considère un modèle (simple) sur y, $\boldsymbol{y} = h_1(\boldsymbol{x})$
- on construit les résidus (avec un paramètre ν de shrinkage), $\boldsymbol{\varepsilon}_1 = \boldsymbol{y} - \nu h_1(\boldsymbol{x})$

à l'étape j,

- on considère un modèle (simple) sur ε_j, $\boldsymbol{\varepsilon}_j = h_j(\boldsymbol{x})$
- on construit les résidus $\boldsymbol{\varepsilon}_{j+1} = \boldsymbol{\varepsilon}_j - \nu h_j(\boldsymbol{x})$

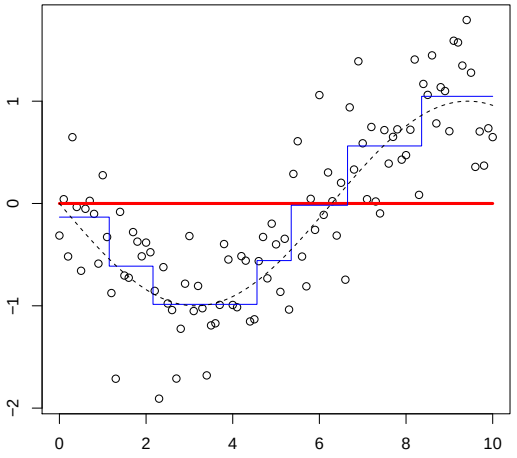
et on boucle. Finalement

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

Boosting

Apprentissage avec des stumps (arbres)

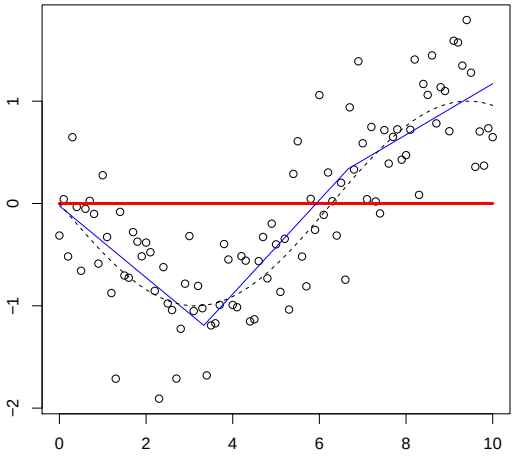
```
> df$e <- df$y - mean(df$y)
> for(k in 1:1000){
+ fit <- rpart(e ~ x,data=df)
+ df$e <- df$y - v*predict(fit)
+ YP <- cbind(YP,v*predict(fit))
+ }
> apply(YP,1,sum)
```



Boosting

Apprentissage avec des splines (linéaires)

```
> library(freeknotsplines)
> df$e <- df$y - mean(df$y)
> for(k in 1:1000){
+ noeuds = freelsgen(df$x,df$e,degree=1,numknot=2)
+ fit <- lm(e ~ bs(x,degree=1,knots=noeuds@optknot
    ,data=df)
+ df$e <- df$y - v*predict(fit)
+ YP <- cbind(YP,v*predict(fit))
+ }
> apply(YP,1,sum)
```



la localisation des noeuds est ici optimisée

Boosting for Classification

On cherche toujours $m^\star(\cdot) = \text{argmin}\{\mathbb{E}[\ell(Y, m(\boldsymbol{X})]\}$. Classiquement, en machine learning, $y \in \{-1, +1\}$, et on considère comme fonction de perte

$$\ell(y, m(\boldsymbol{x})) = e^{-y \cdot m(\boldsymbol{x})}$$

on obtient l'algorithme dit AdaBoot, cf Freund & Schapire (1996).

Alors

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

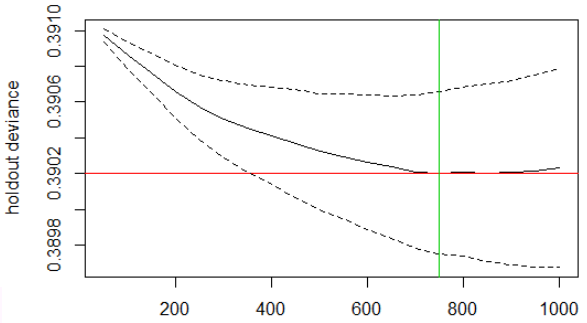
qui rappelle la loi logistique... A l'étape k on résout

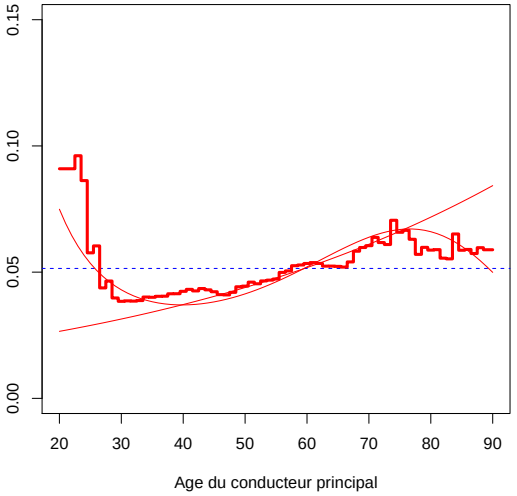
$$\text{argmin} \left\{ \sum_{i=1}^{n} \underbrace{e^{y_i \cdot m_k(\boldsymbol{x}_i)}}_{\omega_{i,k}} \cdot e^{y_i \cdot h(\boldsymbol{x}_i)} \right\}$$

Boosting

Apprentissage avec des stumps (arbres)

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

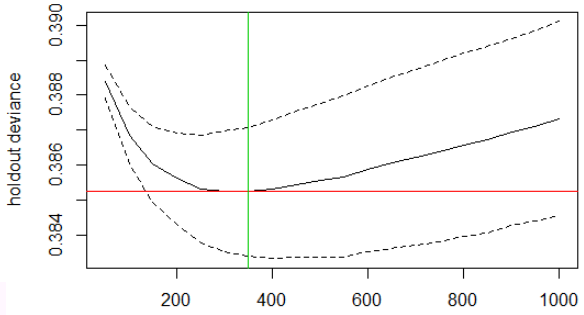


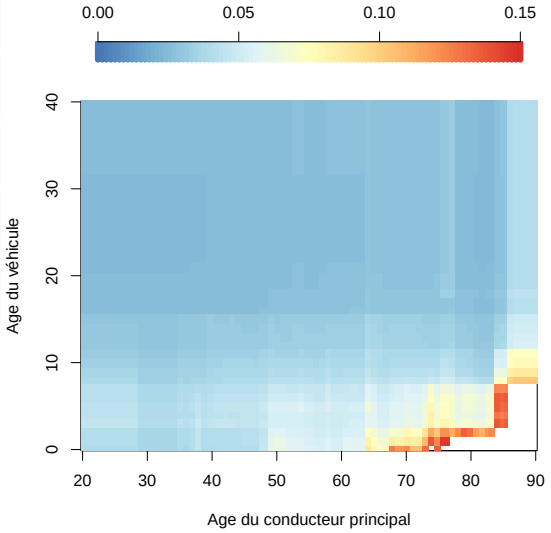


Boosting

Apprentissage avec des stumps (arbres)

```
> 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)
```

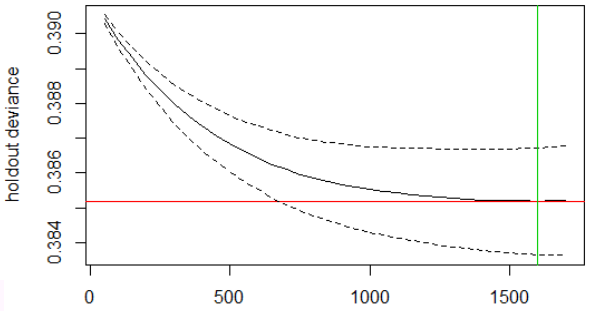


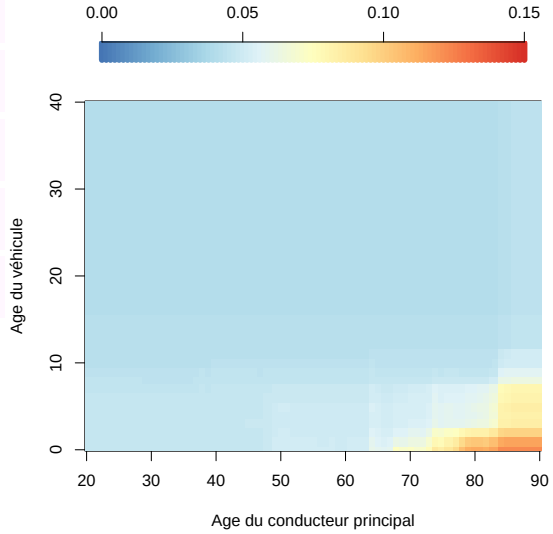


Boosting

Apprentissage avec des stumps (arbres)

```
> library(dismo)
> library(gbm)
> fit <- gbm.step(data=camping, gbm.x=c(1,7), gbm.
    y=13, family="bernoulli", tree.complexity=5,
    learning.rate=0.001, bag.fraction=0.5)
> predict(fit, type="response", n.trees=1500)
```



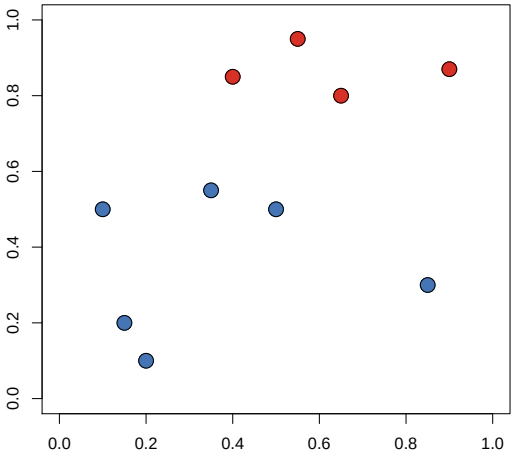


Support Vector Machine

Les SVMs ont été développés dans les années 90 à partir de Vapnik & Lerner (1963), cf Vailant (1984).

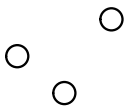
On suppose que les points sont linérement séparables, i.e. il existe $\boldsymbol{\omega}$ et b tels que

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



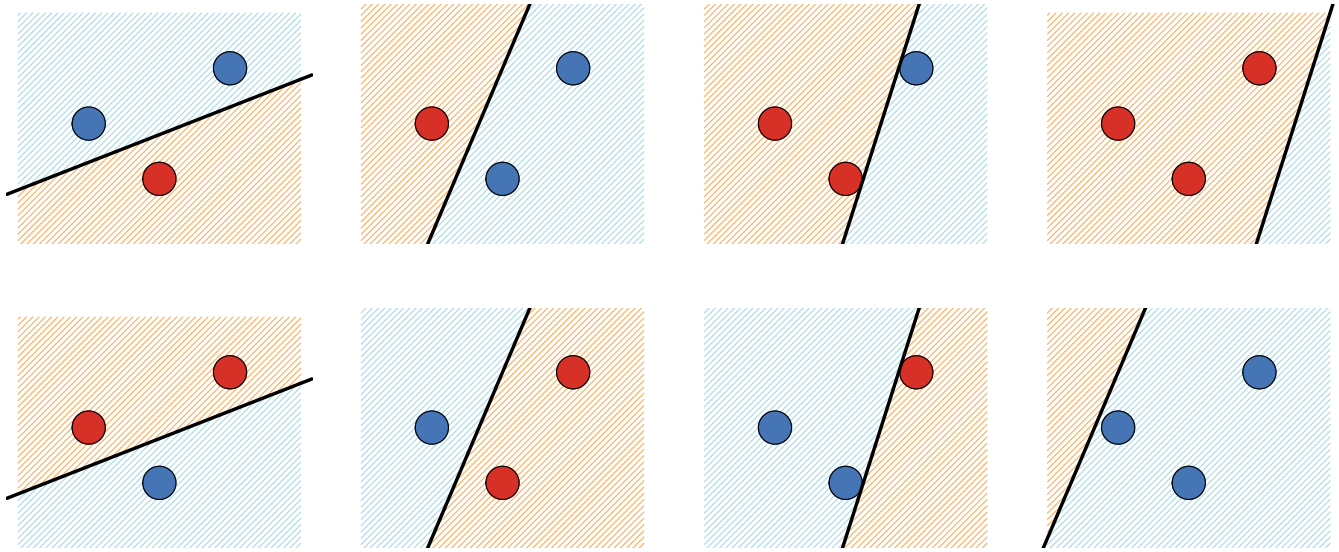
Problème: nombre infini de solutions, il faut en choisir une bonne qui sépare les deux nuages autant que possible.

Remarque derrière l'idée de séparabilité se cache la dimension VC. Soit $\mathcal{H} : \{h : \mathbb{R}^d \mapsto \{-1, +1\}\}$. Alors $\mathcal{H}$ est un shatter des points $\boldsymbol{X}$ si toutes les dichotomies peuvent être obtenues.

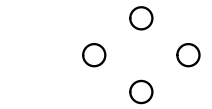


E.g. avec trois point, toutes les configurations sont atteignables

Support Vector Machine



Mais avec les quatre point ci-contre, non: plusieurs configuration ne peuvent être obtenues (avec un séparateur linéaire)



Support Vector Machine

La dimension de Vapnik (VC) est la taille du plus grand ensemble séparé (shattered) de $\boldsymbol{X}$.

En pratique, où est le plan optimal ?

La distance de $\boldsymbol{x}_0$ au plan $\boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}+b$ est

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

et le plan optimal (dans le cas séparable) est

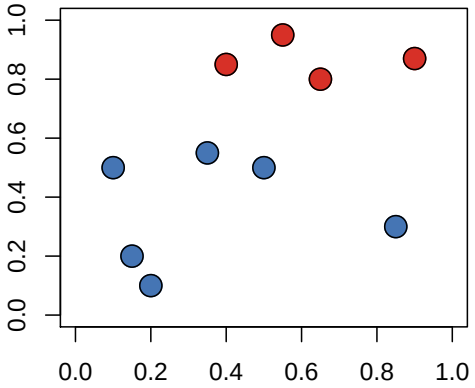
$$\text{argmin}\left\{\min_{i=1,\cdots,n} d(\boldsymbol{x}_i, H_{\boldsymbol{\omega},b})\right\}$$

Support Vector Machine

On définit vectors supports comme les observations telles que

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

La marge est la distance des supports vecteurs au plan.



La distance des vecteurs supports à $H_{\boldsymbol{\omega},b}$ est $\|\boldsymbol{\omega}\|^{-1}$, et la marge est $2\|\boldsymbol{\omega}\|^{-1}$.

$\longrightarrow$ l'aglorithme va minimiser l'inverse des marges, sous contrainte que le plan $H_{\boldsymbol{\omega},b}$ sépare les points (ici ± 1) 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

Problème difficile à résoudre (trop de contraintes: n)

$\longrightarrow$ on va résoudre ce problème d'optimisation dans l'espace dual

Dans l'espace initial, on aurait

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

Dans l'espace dual, le programme s'écrit

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

qui peut s'écrire

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

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

Support Vector Machine

Quid du cas non-séparable?

Ici on ne peut pas avoir $y_i(\boldsymbol{\omega}^\mathsf{T}\boldsymbol{x}_i + b) \geq 1 \ \forall i$.

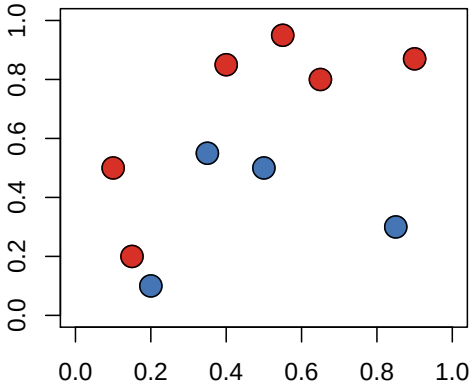
On introduit la notion de slack variables,

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

où $\xi_i \geq 0 \ \forall i$.

On commet une erreur de classification lorsque $\xi_i > 1$.
L'idée est de résoudre

$$\min\left\{\frac{1}{2}\boldsymbol{\omega}^\mathsf{T}\boldsymbol{\omega} + C\mathbf{1}^\mathsf{T}\mathbf{1}_{\boldsymbol{\xi}>1}\right\}, \text{ au lieu de } \min\left\{\frac{1}{2}\boldsymbol{\omega}^\mathsf{T}\boldsymbol{\omega}\right\}$$



Support Vector Machines, with a Linear Kernel

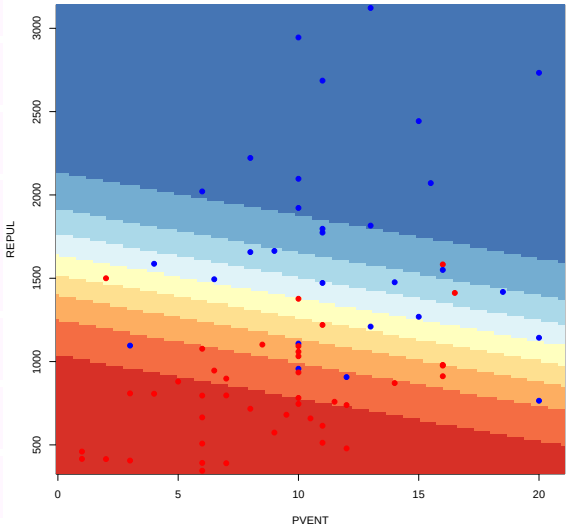
Pour l'instant, on avait considéré

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

où $\|\cdot\|_{\ell_2}$ est la norme Euclidienne (ℓ_2),

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

```
> library(kernlab)
> SVM2 <- ksvm(PRONO ~ PVENT + REPUL, data =
    myocarde, prob.model = TRUE, kernel = "
    vanilladot", type="C-svc")
> 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

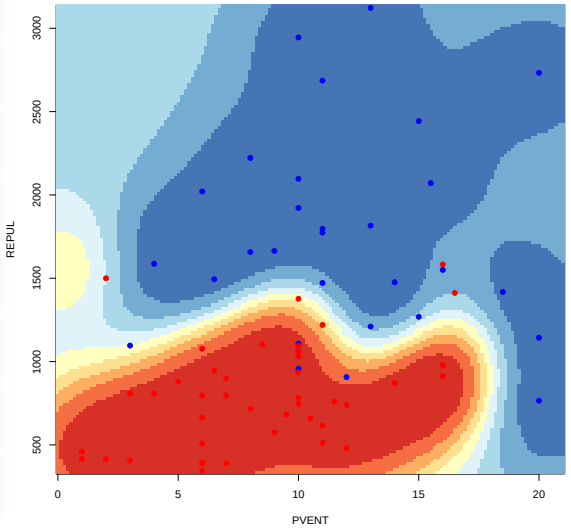
Mais plus généralement, on peut considérer

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

où $\|\cdot\|_k$ est une norme construite avec un noyau

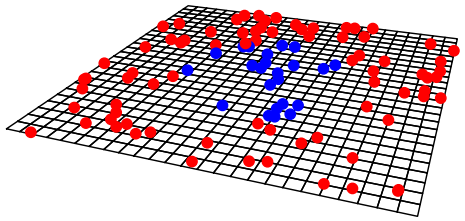
$$\|\boldsymbol{x}_0 - \boldsymbol{x}\|_k = \sqrt{k(\boldsymbol{x}_0,\boldsymbol{x}_0) - 2k(\boldsymbol{x}_0,\boldsymbol{x}) + k(\boldsymbol{x}\cdot\boldsymbol{x})}$$

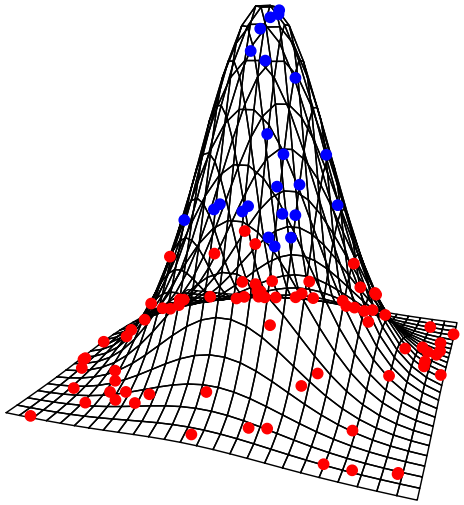
```
1 > library(kernlab)
2 > SVM2 <- ksvm(PRONO ~ PVENT + REPUL, data =
      myocarde, prob.model = TRUE, kernel = "rbfdot"
      , type="C-svc")
3 > pred_SVM2 = function(p,r){
4 + return(predict(SVM2,newdata=
5 + data.frame(PVENT=p,REPUL=r), type="probabilities
      ")[,2])}
```



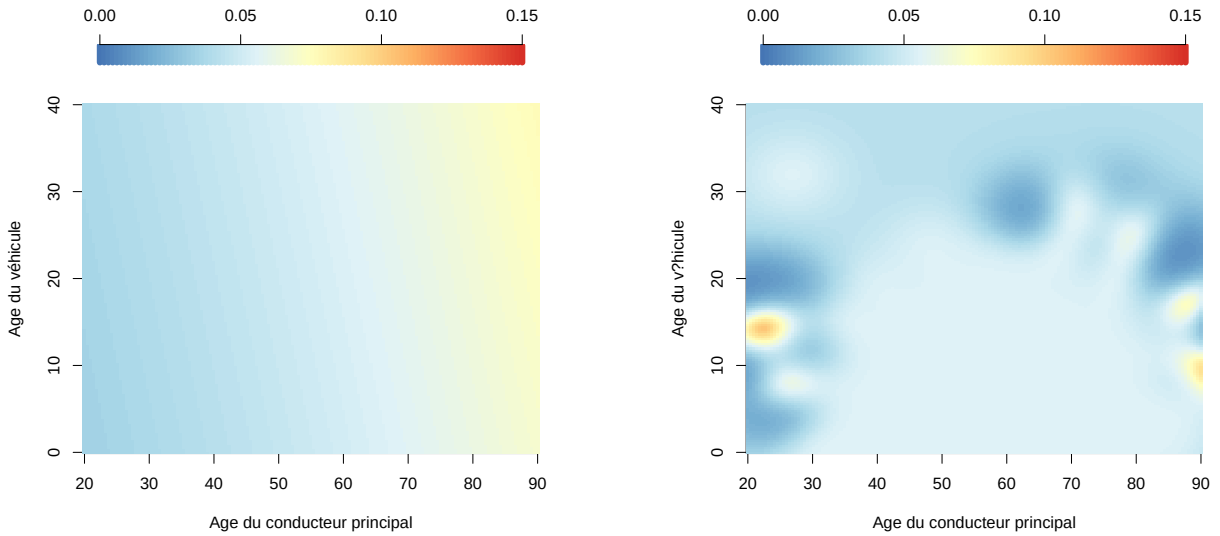
Heuristique sur les SVM à noyau

Une interprétation est que les données ne sont pas linéarement séparables dans l'espace de départ, mais on peut changer d'espace (e.g. noyau Gaussien ici) pour les rendre séparables,





Support Vector Machines



Remarque toutes ces méthodes de type apprentissage statistique sont très sensibles aux tunning parameters.