Travaux Pratiques Econométrie des variables qualitatives

Antoine Sétif

1 Exercice

L'idée est de vérifier un certain nombre de propriétés en utilisant une approche expérimentale. Prenons le contexte où on va chercher à modéliser le succès à une année du master MASSS en fonction des notes de contrôles continus.

1. *Créez deux séries de taille n=100 correspondant à deux notes dans deux matières par simulation. Notez les* ***x1*** *et* ***x2****. Faites-en sorte que les moyennes soient disons 13 et 10. Créez également deux variables auxiliaires : l'une correspondant au sexe et l'autre à l'âge des étudiants. L'ensemble de ces variables sont fixées et ne bougeront plus à moins de faire varier n.*

   Pour nos 100 individus, nous allons commencer par créer nos variables x1 et x2 représentant les résultats des notes aux deux examens. Pour cela, nous allons simuler deux lois normales centrées en 13 d'une part et 10 d'autre part.

```
set.seed(1712)  # On fige la racine du generateur aleatoire
x1 <- round(rnorm(100, 13, 2), 1)

c(min(x1), max(x1))
```

```
## [1]  9.4 18.1
```

```
# Verification qu'aucune note est inferieure a 0 et superieure a 20
(mean(x1))  # Moyenne voisine de 13
```

```
## [1] 13.27
```

```
x2 <- round(rnorm(100, 10, 2), 1)
# Simulation d'une loi normale pour avoir les notes au second examen,
# arrondis au 1er decimal

c(min(x2), max(x2))
```

```
## [1]  4.2 15.4
```

```
# Verification qu'aucune note est inferieure a 0 et superieure a 20
(mean(x2))  # Moyenne voisine de 10
```

```
## [1] 10.11
```

   On crée ensuite notre variable sexe en passant par une simulation d'une loi binomiale.

```
sexe <- rbinom(100, 1, p = 0.5)  # Simulation d'une loi binomiale
mean(sexe)  # Proportion d'hommes
```

```
## [1] 0.39
```

   On crée ensuite notre variable âge en passant par une simulation d'une loi uniforme.

```
age <- round(runif(100, 21, 28), 0)
```

2. *Créez une simulation d'un modèle logit noté* ***yl*** *qui ne dépend que de x1 et x2. Fixez* $\beta_1 = 2$ *et* $\beta_2 = 5$ *et ajustez* β_0 *de sorte qu'il y ait à peu près 50 % de 1 dans l'échantillon* ***yp****. Même travail avec un modèle probit, variable notée* ***yp****. Sauvegardez le jeu de données.*

   Dans un premier temps, constatons que :

```
beta1 <- 2
beta2 <- 5
(mean(beta1 * x1 + beta2 * x2))
```

```
## [1] 77.08
```

   C'est à dire que nous devons ajuster β_0 tel qu'il y ait 50% de 0 et 50% de 1. Par l'instruction R suivante, en prenant naturellement β_0=-77 :

```
beta0 <- -77
p <- plogis(beta1 * x1 + beta2 * x2 + beta0, 0, 1)
yl <- rbinom(100, 1, p)
mean(yl)  # 0.48, voisin de 0.5
```

```
## [1] 0.48
```

   De même, réalisons le raisonnement pour le modèle probit.

```
beta0 <- -77
beta1 <- 2
beta2 <- 5
p <- pnorm(beta1 * x1 + beta2 * x2 + beta0, 0, 1)
yp <- rbinom(100, 1, p)
mean(yp)  # On retrouve 0.48
```

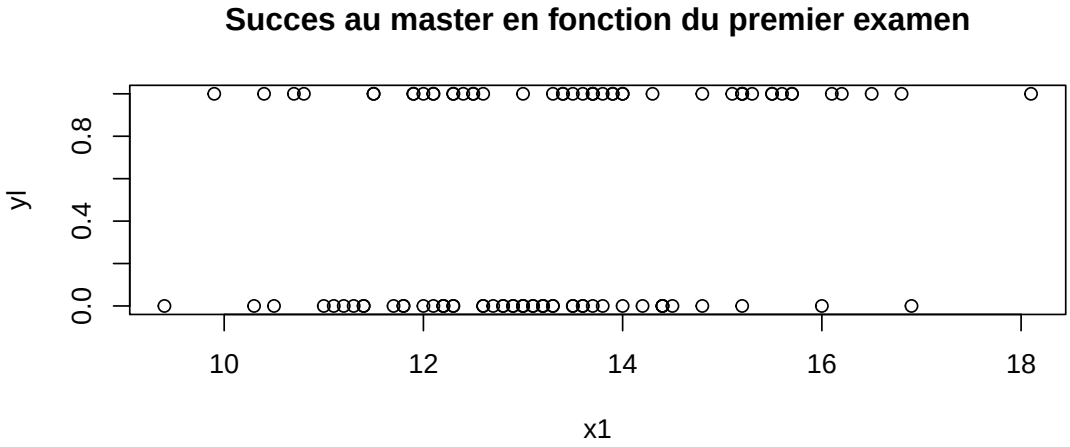
```
## [1] 0.48
```

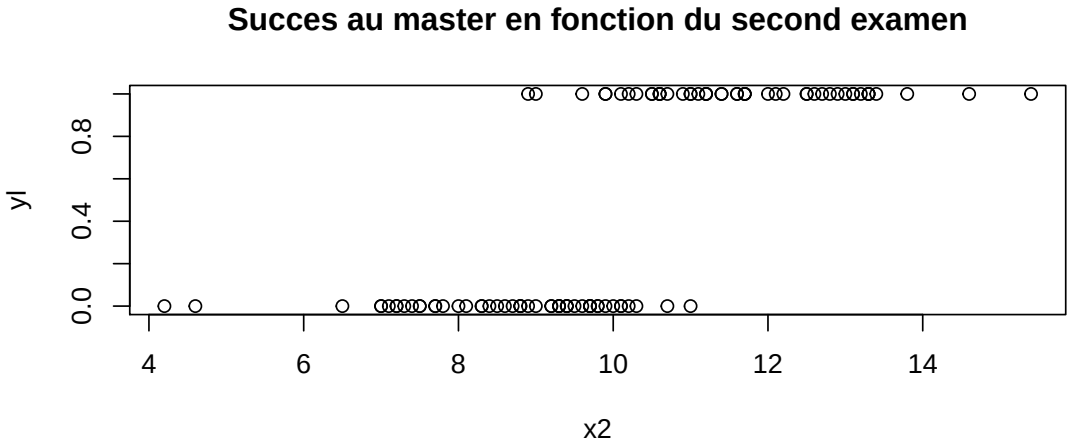
   Enregistrons désormais **yl** et **yp** en tant que dataframe et sauvons nos données.

```
dfl <- data.frame(yl = yl, x1 = x1, x2 = x2, sexe = sexe, age = age)
dfp <- data.frame(yp = yp, x1 = x1, x2 = x2, sexe = sexe, age = age)
save(dfp, dfl, file = "dfProjEcono.Rdata")
```

3. *Tracer* ***yl*** *en fonction de* ***x1*** *et* ***x2*** *et vérifiez que de fortes valeurs des covariables influencent le succès à l'examen final.*

```
par(mfrow = c(2, 1))
plot(x1, yl, main = "Succes au master en fonction du premier examen")
plot(x2, yl, main = "Succes au master en fonction du second examen")
```





Visuellement parlant, d'un rapide coup d'oeil, on peut imaginer que les résultats au second examen influencent plus la réussite au master MASSS que les résultats au premier examen.
Verifions que de fortes valeurs des covariables **x1** et **x2** influencent le succès au master.

```
head(dfl[order(x1, decreasing = T), c(1, 2)])
```

```
##    yl   x1
## 57  1 18.1
## 93  0 16.9
## 23  1 16.8
## 48  1 16.5
## 79  1 16.2
## 39  1 16.1
```

```
head(dfl[order(x2, decreasing = T), c(1, 3)])
```

```
##    yl   x2
## 13  1 15.4
## 23  1 14.6
## 1   1 13.8
## 56  1 13.4
## 41  1 13.3
## 59  1 13.3
```

Analyse du premier modèle (**yl** en fonction de **x1**) :

```
mod1 = (glm(yl ~ x1, family = binomial(logit), data = dfl))
summary(mod1)
```

```
##
## Call:
## glm(formula = yl ~ x1, family = binomial(logit), data = dfl)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.651  -1.108  -0.795   1.154   1.687
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -4.279      1.768   -2.42    0.016 *
## x1             0.316      0.132    2.39    0.017 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.47  on 99  degrees of freedom
## Residual deviance: 132.22  on 98  degrees of freedom
## AIC: 136.2
##
## Number of Fisher Scoring iterations: 4
```

Analyse du second modèle (**yl** en fonction de **x2**) :

```
mod2 = (glm(yl ~ x2, family = binomial(logit), data = dfl))
summary(mod2)
```

```
##
## Call:
## glm(formula = yl ~ x2, family = binomial(logit), data = dfl)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0708  -0.3300  -0.0105   0.2308   2.4479
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -23.984      5.323   -4.51  6.6e-06 ***
## x2             2.364      0.527    4.48  7.4e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.469  on 99  degrees of freedom
## Residual deviance:  51.117  on 98  degrees of freedom
## AIC: 55.12
##
## Number of Fisher Scoring iterations: 7
```

Lez coefficientz β_1 d'une part, et β_2 d'autre part, sont significatifs. Plus les notes aux examens sont élevés, plus les chances de réussite au master MASSS sont importantes (interprétation de β_1 et $\beta_2 > 0$).

4. *Supposons qu'on ait les bonnes covariables. Estimez les paramètres du modèle* ***yl*** *en fonction de* ***x1+x2*** *en supposant un modèle logistique puis probit. Même travail en considérant le modèle* ***yp*** *en fonction de* ***x1+x2****. Rappelez (et vérifiez) comment on peut approximativement "convertir" les estimations d'un modèle logit et probit (et inversement).*

Pour le modèle logit :

```
mod3 = (glm(yl ~ x1 + x2, family = binomial(logit), data = dfl))
mod3$coefficients
```

```
## (Intercept)          x1          x2
##     -85.231       2.275       5.453
```

Constatons que l'on retrouve approximativement β_1=2 et β_2=5. C'est cohérent.

Estimons les paramètres du modèle en considérant qu'il est de nature probit.

```
mod4 = (glm(yl ~ x1 + x2, family = binomial(probit), data = dfl))
tablelogit = cbind(mod3$coefficients, mod4$coefficients * pi/sqrt(3))
colnames(tablelogit) = c("Coeff.mod3", "Coeff.mod4")
tablelogit
```

```
##             Coeff.mod3 Coeff.mod4
## (Intercept)    -85.231    -84.596
## x1               2.275      2.223
## x2               5.453      5.453
```

Pour le modèle probit :

```
mod5 = (glm(yp ~ x1 + x2, family = binomial(probit), data = dfp))
mod5$coefficients
```

```
## (Intercept)          x1          x2
##     -86.011       2.219       5.617
```

Constatons que l'on retrouve une nouvelle fois, de manière approximative, β_1=2 et β_2=5.

Estimons les paramètres du modèle en considérant qu'il est de nature logit.

```
mod6 = (glm(yp ~ x1 + x2, family = binomial(logit), data = dfp))
tableprobit = cbind(mod5$coefficients, mod6$coefficients * sqrt(3)/pi)
colnames(tableprobit) = c("Coeff.mod5", "Coeff.mod6")
tableprobit
```

```
##             Coeff.mod5 Coeff.mod6
## (Intercept)    -86.011    -83.063
## x1               2.219      2.145
## x2               5.617      5.422
```

5. *On se concentre à partir de maintenant uniquement sur le modèle logit. Interprétez les coefficients estimés. En particulier, si la note **x1** augmente d'un point, quelle est l'influence sur la probabilité de succès de l'examen final.*

```
mod3 = (glm(yl ~ x1 + x2, family = binomial(logit), data = dfl))
summary(mod3)
```

```
##
## Call:
## glm(formula = yl ~ x1 + x2, family = binomial(logit), data = dfl)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3898  -0.0409   0.0000   0.0092   2.0346
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -85.231     26.574   -3.21   0.0013 **
## x1             2.275      0.768    2.96   0.0030 **
## x2             5.453      1.692    3.22   0.0013 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.469  on 99  degrees of freedom
## Residual deviance:  20.586  on 97  degrees of freedom
## AIC: 26.59
##
## Number of Fisher Scoring iterations: 9
```

Interprétation des coefficients : Plus les notes augmentent, plus les chances de succès à l'examen sont importantes (car coefficients positifs et significatifs).

Quelle est l'influence sur les probabilités de succès si nous augmentons d'une unité la note **x1** ?

```
exp(mod3$coeff[2])
```

```
##    x1
## 9.731
```

La cote (rapport entre les probabilités de succès et d'échec) va être multipliée par 9.731. Autrement dit, les chances de succès augmentent.

Quelle est l'influence sur les probabilités de succès si nous augmentons d'une unité la note **x2** ?

```
exp(mod3$coeff[3])
```

```
##    x2
## 233.5
```

La cote va être multipliée par plus de 200 ! Les chances de succès augmentent de manière très significative.

Nous pouvons nous rendre compte à ce stade de l'importance de la deuxième note sur les chances de réussite au master.

Voyons comment cela se traduit sur notre deuxième individu :

```
dfl[1:10, ]  # Individu 2 : x1=10.5, x2=9.7
```

```
##    yl   x1   x2 sexe age
## 1   1 12.3 13.8    0  23
## 2   0 10.5  9.7    0  23
## 3   0 13.2  8.8    1  28
## 4   1 15.5 11.7    1  27
## 5   0 11.4  8.1    0  28
## 6   0  9.4 10.2    1  24
## 7   0 12.6  9.7    0  26
## 8   1 13.9 12.8    1  28
## 9   1 12.0 11.1    0  23
## 10  1 13.0 12.1    0  25
```

```
exp(mod3$coeff[1] + mod3$coeff[2] * dfl$x1[2] + mod3$coeff[3] * dfl$x2[2])
```

```
## (Intercept)
##   0.0002154
```

L'individu 2 a 0.0002154 fois plus de chances d'un succès à l'examen plutôt qu'un échec.

```
exp(mod3$coeff[2])
```

```
##    x1
## 9.731
```

```
exp(mod3$coeff[2]) * exp(mod3$coeff[1] + mod3$coeff[2] * dfl$x1[2] + mod3$coeff[3] *
    dfl$x2[2])
```

```
##       x1
## 0.002096
```

Une augmentation de 1 point sur la première note aurait fait passer la cote de 0.0002154 à 0.002096.

Qu'en est-il concernant l'augmentation d'une unité de la seconde note sur notre deuxième individu ?

```
exp(mod3$coeff[3])
```

```
##    x2
## 233.5
```

```
exp(mod3$coeff[3]) * exp(mod3$coeff[1] + mod3$coeff[2] * dfl$x1[2] + mod3$coeff[3] *
    dfl$x2[2])
```

```
##      x2
## 0.05031
```

Une augmentation de 1 point sur la deuxième note de notre deuxième individu aurait fait passer la cote de 0.0002154 à 0.05031.

6. *Quelle est l'erreur standard des estimations des paramètres ? Vérifiez la sortie R avec la formule théorique (indication : calculez la matrice d'information de Fisher).*

```
summary(mod3)
```

```
##
## Call:
## glm(formula = yl ~ x1 + x2, family = binomial(logit), data = dfl)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3898  -0.0409   0.0000   0.0092   2.0346
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -85.231     26.574   -3.21   0.0013 **
## x1             2.275      0.768    2.96   0.0030 **
## x2             5.453      1.692    3.22   0.0013 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.469  on 99  degrees of freedom
## Residual deviance:  20.586  on 97  degrees of freedom
## AIC: 26.59
##
## Number of Fisher Scoring iterations: 9
```

```
fi <- dlogis(mod3$coefficients[1] + mod3$coefficients[2] * x1 + mod3$coefficients[3] *
    x2, 0, 1)
V <- 0
for (i in (1:100)) {
    tmp <- c(1, x1[i], x2[i])
    tmp2 <- t(t(tmp)) %*% tmp
    V <- V + fi[i] * tmp2
}

sqrt(diag(solve(V)))
```

```
## [1] 26.5743  0.7679  1.6918
```

On retrouve bien nos erreurs standards associées à chacun de nos coefficients.

7. *Les variables **x1** et **x1** sont-elles significatives au seuil de 5% ? Vérifiez le calcul de la p-valeur.*

Les 2 variables sont bien significatives (réponse à la question 5). Vérifions les p-valeurs associées :

```
2 * (1 - pnorm(summary(mod3)$coef[2, 1]/summary(mod3)$coef[2, 2]))
```

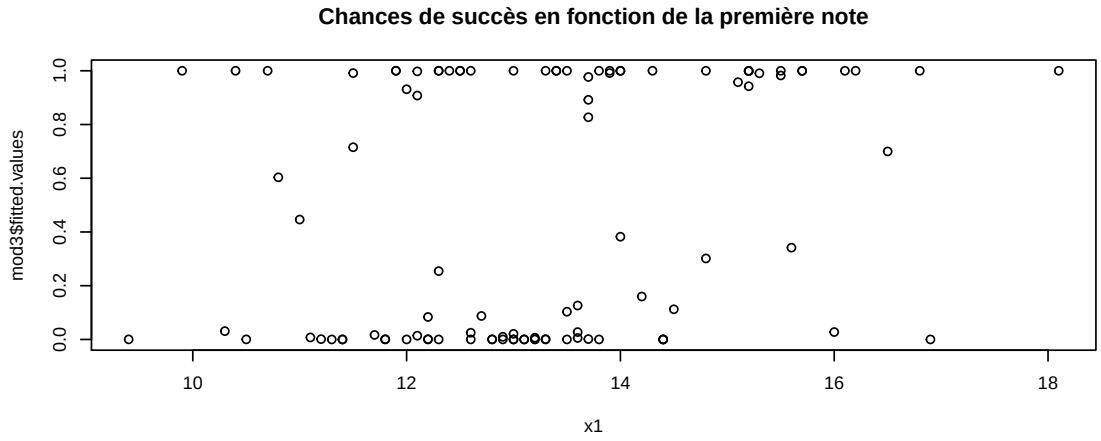
```
## [1] 0.003045
```

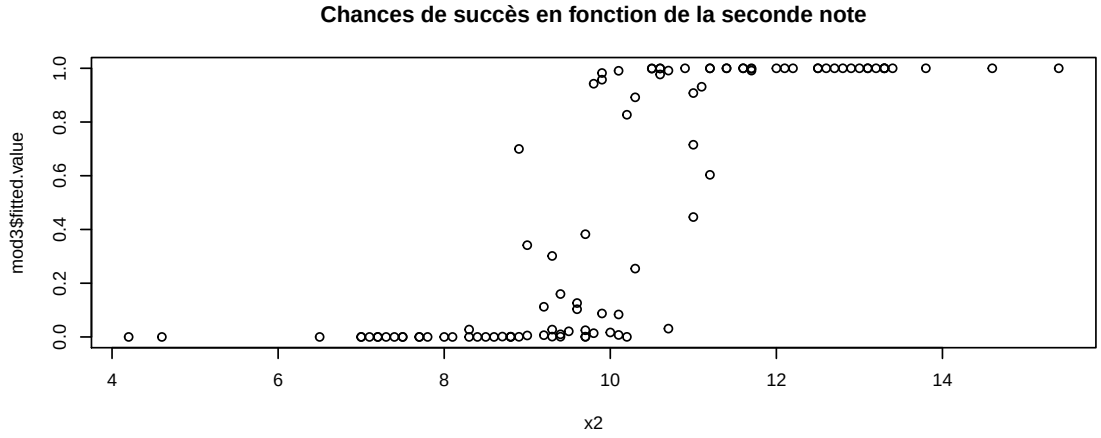
```
2 * (1 - pnorm(summary(mod3)$coef[3, 1]/summary(mod3)$coef[3, 2]))
```

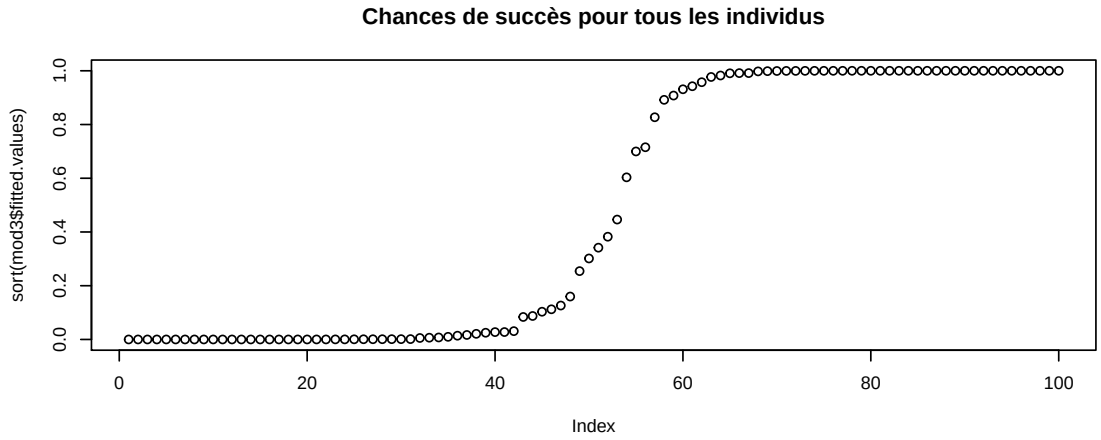
```
## [1] 0.001267
```

Par ailleurs, il est intéressant de noter que :

```
par(mfrow = c(3, 1))
plot(x1, mod3$fitted.values, main = "Chances de succès en fonction de la première note")
plot(x2, mod3$fitted.value, main = "Chances de succès en fonction de la seconde note")
plot(sort(mod3$fitted.values), main = "Chances de succès pour tous les individus")
```







Très peu d'influence de la première note, en revanche, la seconde note détermine beaucoup plus les chances de succès.

8. *Montrer par une approche expérimentale (en simulant un grand nombre de fois la modèle) qu'en moyenne, on retrouve approximativement les valeurs des paramètres. Montrer, en faisant évoluer n que les estimateurs sont consistants (i.e que leur variance tend vers 0). Vérifiez également la normalité asymptotique.*

   Nous allons montrer qu'en moyenne, on retrouve approximativement la valeur des paramètres $\beta_1 = 2$ et $\beta_2 = 5$.

   Premièrement, nous décidons de faire 100 simulations de taille n=100 :

```
coeff <- array(dim = c(1000, 2))
nsim = 1000
for (i in (1:nsim)) {
    set.seed(i)
    x1b <- round(rnorm(100, 13, 0.1), 1)
    x2b <- round(rnorm(100, 10, 0.1), 1)
    pb <- plogis(beta1 * x1b + beta2 * x2b - 76, 0, 1)
    y1b <- rbinom(100, 1, pb)
    mod3b = (glm(y1b ~ x1b + x2b, family = binomial(logit)))
    coeff[i, 1] <- mod3b$coeff[2]
    coeff[i, 2] <- mod3b$coeff[3]
}

mean(coeff[, 1])
```

```
## [1] 2.137
```

```
mean(coeff[, 2])
```

```
## [1] 5.221
```

   Dans un second temps, nous allons chercher à montrer que nos estimateurs sont consistants. Nous allons dans un premier lieu fixer n=1000, puis n=10000 et comparer les sorties observées.

```
#### 1K ####
x11K <- round(rnorm(1000, 13, 1), 1)
x21K <- round(rnorm(1000, 10, 1), 1)
p1K <- plogis(beta1 * x11K + beta2 * x21K - 76, 0, 1)
y11K <- rbinom(1000, 1, p1K)
mod31K = (glm(y11K ~ x11K + x21K, family = binomial(logit)))
#### 10K ####
x110K <- round(rnorm(10000, 13, 1), 1)
x210K <- round(rnorm(10000, 10, 1), 1)
p10K <- plogis(beta1 * x110K + beta2 * x210K - 76, 0, 1)
y110K <- rbinom(10000, 1, p10K)
mod310K = (glm(y110K ~ x110K + x210K, family = binomial(logit)))
```

```
summary(mod3)$coef  # n=100
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -85.231    26.5741  -3.207 0.001340
## x1             2.275     0.7679   2.963 0.003045
## x2             5.453     1.6918   3.223 0.001267
```

```
summary(mod31K)$coef  # n=1000
```

```
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  -82.567     6.1237  -13.48 1.965e-41
## x11K           2.165     0.2022   10.71 9.538e-27
## x21K           5.435     0.3963   13.71 8.323e-43
```

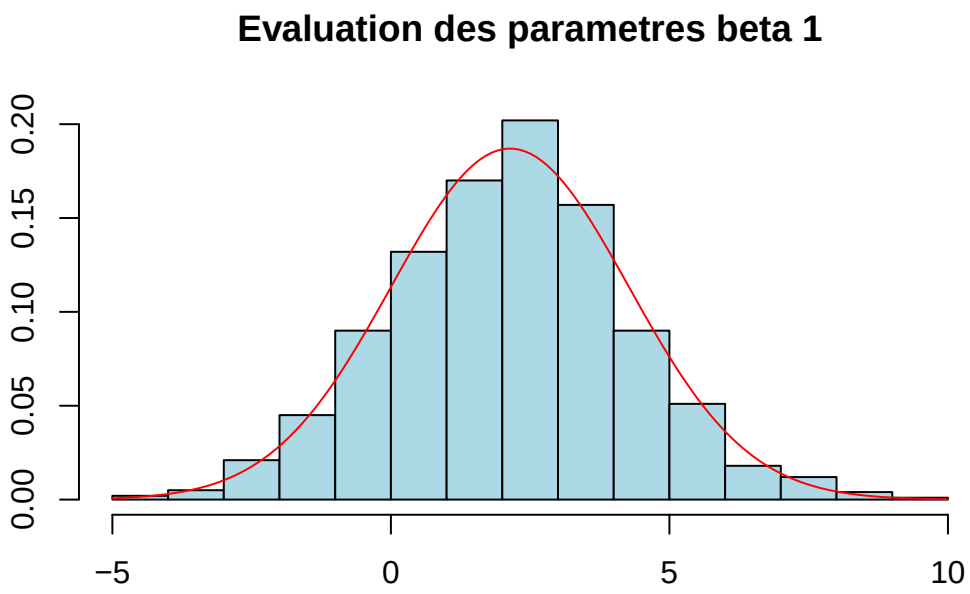
```
summary(mod310K)$coef   # n=10000
```

```
##              Estimate Std. Error z value   Pr(>|z|)
## (Intercept)  -74.078    1.69125  -43.80  0.000e+00
## x110K          1.900    0.05506   34.50 8.425e-261
## x210K          4.945    0.11196   44.17  0.000e+00
```

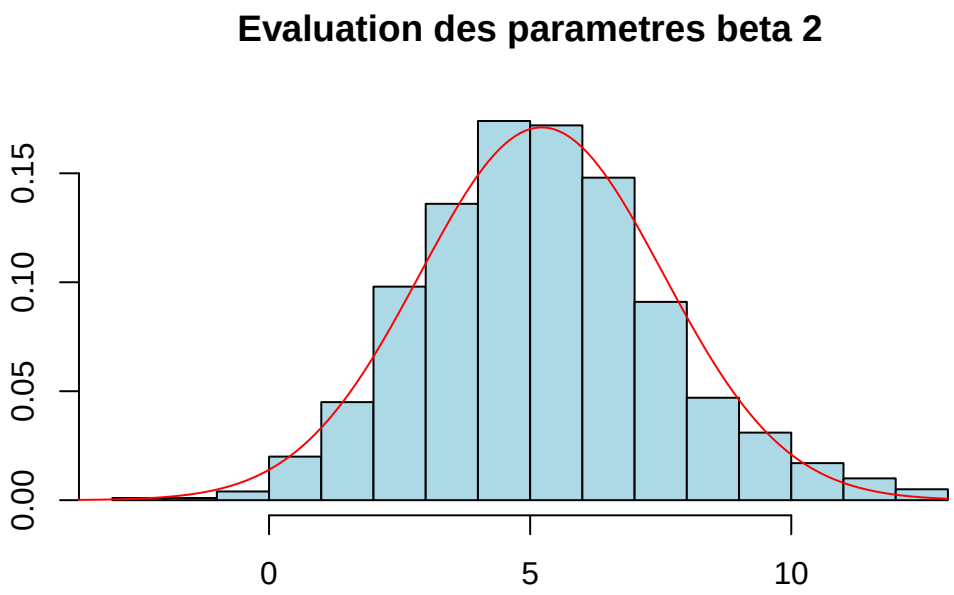
Nous pouvons voir que les coefficients estimés se rapprochent de plus en plus des vraies paramètres et les erreurs standards (donc les variances) tendent vers 0 plus n est grand.

Véfifions désormais la normalité asymptotique (1000 simulations avec n=100) :

```
par(mfrow = c(1, 1))
coeff <- array(dim = c(1000, 2))
nsim = 1000
for (i in (1:nsim)) {
    set.seed(i)
    x1b <- round(rnorm(100, 13, 0.1), 1)
    x2b <- round(rnorm(100, 10, 0.1), 1)
    pb <- plogis(beta1 * x1b + beta2 * x2b - 76, 0, 1)
    y1b <- rbinom(100, 1, pb)
    mod3b = (glm(y1b ~ x1b + x2b, family = binomial(logit)))
    coeff[i, 1] <- mod3b$coeff[2]
    coeff[i, 2] <- mod3b$coeff[3]
}
hist(coeff[, 1], col = "lightblue", main = "Evaluation des parametres beta 1",
    xlab = "", ylab = "", freq = FALSE)
x <- seq(-5, 10, 0.1)
y <- dnorm(x, mean(coeff[, 1]), sd(coeff[, 1]))
lines(x, y, col = "red")
```



```r
hist(coeff[, 2], col = "lightblue", main = "Evaluation des parametres beta 2",
    xlab = "", ylab = "", freq = FALSE)
x <- seq(-5, 13, 0.1)
y <- dnorm(x, mean(coeff[, 2]), sd(coeff[, 2]))
lines(x, y, col = "red")
```



On peut remarquer que dans certains cas, β_1 (et même β_2) peut prendre des valeurs négatives. Il y a donc des valeurs extrêmes pour chacun des coefficients.

A première vue, la normalité est bien respectée. Vérifions cela par un test de Shapiro.
Pour rappel, H_1 *: la série des observations ne suit pas une loi normale.*

Ainsi, si on ne rejette pas l'hypothèse nulle (ou si on ne peut pas accepter H_1), alors on ne pourra pas dire que la série des observations ne suit pas une loi normale.

```r
shapiro.test(coeff[, 1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  coeff[, 1]
## W = 0.9988, p-value = 0.7671
```

```r
shapiro.test(coeff[, 2])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  coeff[, 2]
## W = 0.9922, p-value = 3.948e-05
```

C'est bien le cas ici pour β_1.
En revanche, pour β_2, la sortie R nous indique le contraire du résultat visuellement attendu. Mystère.

9. *Reprenons le jeu de données initial (taille 100), considérez le modèle à 2 variables explicatives* ***x1+x2****. Vérifiez le calcul de la déviance obtenue par la sortie R standard. Quelle est la p-valeur du test de déviance ?*

```
mod3$deviance
```

```
## [1] 20.59
```

1ère méthode :
Pour rappel, $Dviance = -2 * LogVraisemblance$.

```
piEst <- mod3$fitted.values
-2 * sum(yl * log(piEst) + (1 - yl) * log(1 - piEst))
```

```
## [1] 20.59
```

2nde méthode :

```
i0 <- which(yl == 0)
i1 <- which(yl == 1)
-2 * (sum(log(1 - piEst[i0])) + sum(log(piEst[i1])))
```

```
## [1] 20.59
```

Pour rappel, H_1 : Le modèle n'est pas correct.
La statistique de déviance suit approximativement une loi de $khi^2(n-p)$ sous H_0, avec n grand = 100 et p le nombre de régresseurs, 2 variables explicatives + constante = 3).

```
1 - pchisq(mod3$dev, df = 100 - 3)
```

```
## [1] 1
```

Ici, on ne peut pas accepter H_1.
Le modèle n'est pas forcément bon, mais on ne peut pas dire qu'il n'est pas correct.

10. *Déterminons un intervalle de confiance au seuil de 95% des paramètres estimés.*

```
(ICb1 <- summary(mod3)$coeff[2, 1] + c(-1.96, 1.96) * summary(mod3)$coeff[2,
    2])
```

```
## [1] 0.7703 3.7803
```

```
(ICb2 <- summary(mod3)$coeff[3, 1] + c(-1.96, 1.96) * summary(mod3)$coeff[3,
    2])
```

```
## [1] 2.137 8.769
```

11. *Formez le test statistique permettant de montrer H_1 : $\beta_2 > \beta_1$.*
Pour cela, nous avons besoin de la matrice de Fisher :
Sous H_0, la statistique de test (unilatéral) suit une loi normale (0,1).

```
(InfoFis <- sqrt(solve(V)))
```

```
##       [,1]   [,2]  [,3]
## [1,] 26.57    NaN   NaN
## [2,]   NaN 0.7679 1.084
## [3,]   NaN 1.0840 1.692
```

```
(deltaEst.H0 <- (mod3$coeff[3] - mod3$coeff[2])/(sqrt((InfoFis[3, 3] + InfoFis[2,
    2] - 2 * InfoFis[2, 3])/100)))
```

```
##    x2
## 58.84
```

```
1 - pnorm(deltaEst.H0)
```

```
## x2
##  0
```

Ici, on accepte H_1, $\beta_2 > \beta_1$.

12. *Considérez le modèle à 4 variables explicatives* ***x1+x2+age+sexe*** *(modèle complet) et commentez. Testez par un test basé sur la déviance ce nouveau modèle par rapport au modèle précédent.*

```
modcomplet = glm(yl ~ x1 + x2 + sexe + age, family = binomial(logit), data = dfl)
summary(modcomplet)
```

```
##
## Call:
## glm(formula = yl ~ x1 + x2 + sexe + age, family = binomial(logit),
##     data = dfl)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2211  -0.0124   0.0000   0.0031   1.9107
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -104.3110    38.1163   -2.74   0.0062 **
## x1             2.7536     1.0732    2.57   0.0103 *
## x2             6.6462     2.4217    2.74   0.0061 **
## sexe          -1.6510     1.9705   -0.84   0.4021
## age            0.0563     0.5097    0.11   0.9120
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.469  on 99  degrees of freedom
## Residual deviance:  18.848  on 95  degrees of freedom
## AIC: 28.85
##
## Number of Fisher Scoring iterations: 10
```

Visiblement, les variables **sexe** et **age** n'influencent pas le succès au master.
Continuons par un test de déviance sur ce modèle :

Pour rappel, H_1 *: Le modèle n'est pas correct.*

La statistique déviance suit approximativement une loi de $khi^2(n-p)$ sous H_0, avec n grand = 100 et p le nombre de regresseurs, 4 variables explicatives + constante = 5).

```
1 - pchisq(modcomplet$dev, df = 100 - 5)
```

```
## [1] 1
```

On ne peut pas accepter H_1 : on ne peut pas dire que le modèle complet n'est pas correct.

Appliquons désormais un test déviance du nouveau modèle par rapport au précédent (cas : modèles emboités, car mod3 est emboité dans modcomplet).

Pour rappel , le principe du test est le suivant : H_1 : modcomplet est meilleur que mod3.
La statistique de la différence de déviance suit approximativement une loi de $khi^2(p2 - p1)$ sous H_0.

```
1 - pchisq(mod3$deviance - modcomplet$deviance, df = 5 - 3)
```

```
## [1] 0.4193
```

On ne peut pas accepter que le modèle complet soit meilleur que le modèle à 2 variables, ce qui semble cohérent. Autrement dit, le modèle à 2 variables est significativement plus informatif que le modèle complet.

13. *Comparez les AIC des deux modèles (vérifiez le calcul de la sortie R). Mettez en place une procédure de sélection de variables ascendante, descendante dans les deux directions basée sur un critère AIC.*

```
mod3$aic == mod3$dev + 2 * 3  # modèle à 3 paramètres
```

```
## [1] TRUE
```

```
modcomplet$aic == modcomplet$dev + 2 * 5  # modèle à 5 paramètres
```

```
## [1] TRUE
```

```
mod3$aic < modcomplet$aic
```

```
## [1] TRUE
```

Le modèle à 3 variables possède un plus faible AIC.
On met en place une procédure de sélection de variables suivant le critère de l'AIC par les méthodes connues (ascendante, descendante, dans les deux sens).

```
# Methode ascendante
step(mod3, direction = "forward", scope = list(upper = formula(modcomplet)))
```

```
## Start:  AIC=26.59
## yl ~ x1 + x2
##
##        Df Deviance  AIC
## <none>        20.6 26.6
## + sexe  1     18.9 26.9
## + age   1     19.7 27.7
##
## Call:  glm(formula = yl ~ x1 + x2, family = binomial(logit), data = dfl)
##
## Coefficients:
## (Intercept)           x1           x2
##      -85.23         2.28         5.45
##
## Degrees of Freedom: 99 Total (i.e. Null);  97 Residual
## Null Deviance:     138
## Residual Deviance: 20.6  AIC: 26.6
```

```r
# Methode descendante
step(modcomplet, direction = "backward", scope = list(lower = formula(mod3)))
```

```
## Start:  AIC=28.85
## yl ~ x1 + x2 + sexe + age
##
##        Df Deviance  AIC
## - age   1     18.9 26.9
## - sexe  1     19.7 27.7
## <none>        18.9 28.9
##
## Step:  AIC=26.86
## yl ~ x1 + x2 + sexe
##
##        Df Deviance  AIC
## - sexe  1     20.6 26.6
## <none>        18.9 26.9
##
## Step:  AIC=26.59
## yl ~ x1 + x2
##
## Call:  glm(formula = yl ~ x1 + x2, family = binomial(logit), data = dfl)
##
## Coefficients:
## (Intercept)           x1           x2
##      -85.23         2.28         5.45
##
## Degrees of Freedom: 99 Total (i.e. Null);  97 Residual
## Null Deviance:     138
## Residual Deviance: 20.6  AIC: 26.6
```

```r
# Methode dans les deux sens
step(modcomplet, direction = "both", scope = list(lower = formula(mod3)))
```

Les 3 méthodes nous incitent de conserver le modèle à 3 paramètres.

14. *Par une approche expérimentale, sur une taille d'échantillon fixée (puis une plus grande), calculez la proportion de fois que la procédure sélectionne le bon modèle, un modèle à trois covariables, à quatre covariables...*

    On propose de répondre à cette question en fixant dans un premier temps n=10 :

```r
# n=10
n <- 10
nsim = 1000
aic10 <- array(dim = c(nsim, 4))
for (i in (1:nsim)) {
    x110 <- round(rnorm(n, 13, 0.1), 1)
    x210 <- round(rnorm(n, 10, 0.1), 1)
    sexe10 <- rbinom(n, 1, p = 0.5)
    age10 <- round(runif(n, 21, 28), 0)
    p10 <- plogis(beta1 * x1 + beta2 * x2 - 75, 0, 1)
    yl10 <- rbinom(n, 1, p10)
    mod10 = (glm(yl10 ~ x110 + x210, family = binomial(logit)))
    mod11 = (glm(yl10 ~ x110 + x210 + sexe10, family = binomial(logit)))
    mod12 = (glm(yl10 ~ x110 + x210 + age10, family = binomial(logit)))
```

```
    mod13 = (glm(yl10 ~ x110 + x210 + sexe10 + age10, family = binomial(logit)))
    aic10[i, 1] <- as.numeric((mod10$aic < mod11$aic) & (mod10$aic < mod12$aic) &
        (mod10$aic < mod13$aic))
    aic10[i, 2] <- as.numeric((mod11$aic < mod10$aic) & (mod11$aic < mod12$aic) &
        (mod11$aic < mod13$aic))
    aic10[i, 3] <- as.numeric((mod12$aic < mod10$aic) & (mod12$aic < mod11$aic) &
        (mod12$aic < mod13$aic))
    aic10[i, 4] <- as.numeric((mod13$aic < mod10$aic) & (mod13$aic < mod11$aic) &
        (mod13$aic < mod12$aic))
}

aic10 = rbind(aic10, apply(aic10, 2, mean))  # Calcul les proportions
colnames(aic10) = c("Mod.1", "Mod.2", "Mod.3", "Mod.4")
rownames(aic10) = c(1:nsim, "Proportion")
aic10[c(1:2, (nsim - 2):(nsim + 1)), ]  # Quelques exemples
```

```
##            Mod.1 Mod.2 Mod.3 Mod.4
## 1          0.000 0.000 0.000 1.000
## 2          1.000 0.000 0.000 0.000
## 998        1.000 0.000 0.000 0.000
## 999        1.000 0.000 0.000 0.000
## 1000       1.000 0.000 0.000 0.000
## Proportion 0.446 0.152 0.194 0.206
```

En fixant maintenant n=100 :

```
# n=100
n <- 100
nsim = 1000
aic100 <- array(dim = c(nsim, 4))
for (i in (1:nsim)) {
    x1100 <- round(rnorm(n, 13, 0.1), 1)
    x2100 <- round(rnorm(n, 10, 0.1), 1)
    sexe100 <- rbinom(n, 1, p = 0.5)
    age100 <- round(runif(n, 21, 28), 0)
    p100 <- plogis(beta1 * x1 + beta2 * x2 - 75, 0, 1)
    yl100 <- rbinom(n, 1, p100)
    mod10 = (glm(yl100 ~ x1100 + x2100, family = binomial(logit)))
    mod11 = (glm(yl100 ~ x1100 + x2100 + sexe100, family = binomial(logit)))
    mod12 = (glm(yl100 ~ x1100 + x2100 + age100, family = binomial(logit)))
    mod13 = (glm(yl100 ~ x1100 + x2100 + sexe100 + age100, family = binomial(logit)))
    aic100[i, 1] <- as.numeric((mod10$aic < mod11$aic) & (mod10$aic < mod12$aic) &
        (mod10$aic < mod13$aic))
    aic100[i, 2] <- as.numeric((mod11$aic < mod10$aic) & (mod11$aic < mod12$aic) &
        (mod11$aic < mod13$aic))
    aic100[i, 3] <- as.numeric((mod12$aic < mod10$aic) & (mod12$aic < mod11$aic) &
        (mod12$aic < mod13$aic))
    aic100[i, 4] <- as.numeric((mod13$aic < mod10$aic) & (mod13$aic < mod11$aic) &
        (mod13$aic < mod12$aic))
}

aic100 = rbind(aic100, apply(aic100, 2, mean))
colnames(aic100) = c("Mod.1", "Mod.2", "Mod.3", "Mod.4")
rownames(aic100) = c(1:nsim, "Proportion")
```

```r
aic100[c(1:2, (nsim - 2):(nsim + 1)), ]  # Quelques exemples
```

```
##            Mod.1 Mod.2 Mod.3 Mod.4
## 1          1.000 0.000  0.00 0.000
## 2          0.000 0.000  1.00 0.000
## 998        1.000 0.000  0.00 0.000
## 999        0.000 1.000  0.00 0.000
## 1000       1.000 0.000  0.00 0.000
## Proportion 0.697 0.139  0.13 0.034
```

En fixant désormais n=1000 :

```r
# n=1000
n <- 1000
nsim = 1000
aic1000 <- array(dim = c(nsim, 4))
for (i in (1:nsim)) {
    x11K <- round(rnorm(n, 13, 0.1), 1)
    x21K <- round(rnorm(n, 10, 0.1), 1)
    sexe1K <- rbinom(n, 1, p = 0.5)
    age1K <- round(runif(n, 21, 28), 0)
    p1K <- plogis(beta1 * x1 + beta2 * x2 - 75, 0, 1)
    y1lK <- rbinom(n, 1, p1K)
    mod10 = (glm(y1lK ~ x11K + x21K, family = binomial(logit)))
    mod11 = (glm(y1lK ~ x11K + x21K + sexe1K, family = binomial(logit)))
    mod12 = (glm(y1lK ~ x11K + x21K + age1K, family = binomial(logit)))
    mod13 = (glm(y1lK ~ x11K + x21K + sexe1K + age1K, family = binomial(logit)))
    aic1000[i, 1] <- as.numeric((mod10$aic < mod11$aic) & (mod10$aic < mod12$aic) &
        (mod10$aic < mod13$aic))
    aic1000[i, 2] <- as.numeric((mod11$aic < mod10$aic) & (mod11$aic < mod12$aic) &
        (mod11$aic < mod13$aic))
    aic1000[i, 3] <- as.numeric((mod12$aic < mod10$aic) & (mod12$aic < mod11$aic) &
        (mod12$aic < mod13$aic))
    aic1000[i, 4] <- as.numeric((mod13$aic < mod10$aic) & (mod13$aic < mod11$aic) &
        (mod13$aic < mod12$aic))
}

aic1000 = rbind(aic1000, apply(aic1000, 2, mean))
colnames(aic1000) = c("Mod.1", "Mod.2", "Mod.3", "Mod.4")
rownames(aic1000) = c(1:nsim, "Proportion")
aic1000[c(1:2, (nsim - 2):(nsim + 1)), ]  # Quelques exemples
```

```
##            Mod.1 Mod.2 Mod.3 Mod.4
## 1          0.000 1.000 0.000 0.000
## 2          1.000 0.000 0.000 0.000
## 998        1.000 0.000 0.000 0.000
## 999        1.000 0.000 0.000 0.000
## 1000       1.000 0.000 0.000 0.000
## Proportion 0.696 0.144 0.139 0.021
```

Ainsi, on constate qu'en augmentant n, on se rapproche des proportions suivantes :

(a) 70% pour le modèle à 2 covariables
(b) 14% pour le modèle **x1**, **x2** et **sexe**
(c) 14% pour le modèle **x1**, **x2** et **age**
(d) 2% pour le modèle complet.

15. *Sur le jeu de données initial et le modèle à 2 covariables, calculez la table de prédiction en choisissant un seuil à 50%. Interprétez les valeurs et calculez entre autres les taux de vrais et de faux positifs. Recommencez avec un seuil de 10% et de 90%.*

Analyse discriminante statistique : on créé la matrice de confusion.

```
conf <- table(yl, as.numeric(mod3$fitted.values > 0.5))
tbc <- (conf[1, 1] + conf[2, 2])/sum(conf)  # Taux de bonne classif. = 97%
tvp <- conf[1, 1]/(conf[1, 1] + conf[1, 2])  # Taux de vrais positifs = 98.08%
tfp <- conf[2, 1]/(conf[2, 1] + conf[2, 2])  # Taux de faux positifs = 4.17%
```

Ici, c'est une très bonne classification.

On se propose ensuite de créer une fonction qui permet de calculer automatiquement ces taux à partir de deux arguments : le modèle et le seuil. Ainsi, on a :

```
confusion <- function(mod, seuil) {
    conf <- table(mod$y, as.numeric(mod$fitted.values > seuil))
    tbc <- (conf[1, 1] + conf[2, 2])/sum(conf)
    tvp <- conf[1, 1]/(conf[1, 1] + conf[1, 2])
    tfp <- conf[2, 1]/(conf[2, 1] + conf[2, 2])
    return(list(matconf = conf, tbc = tbc, tvp = tvp, tfp = tfp))
}
confusion(mod3, 0.5)  # Bonne classification
```

```
## $matconf
##
##      0  1
##   0 51  1
##   1  2 46
##
## $tbc
## [1] 0.97
##
## $tvp
## [1] 0.9808
##
## $tfp
## [1] 0.04167
```

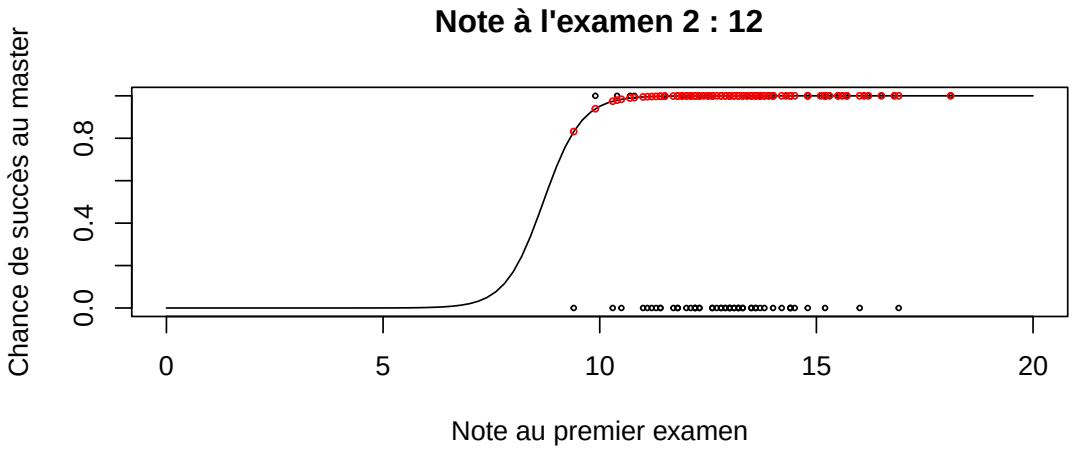
```
confusion(mod3, 0.1)  # Taux de vrais positifs trop faible
```

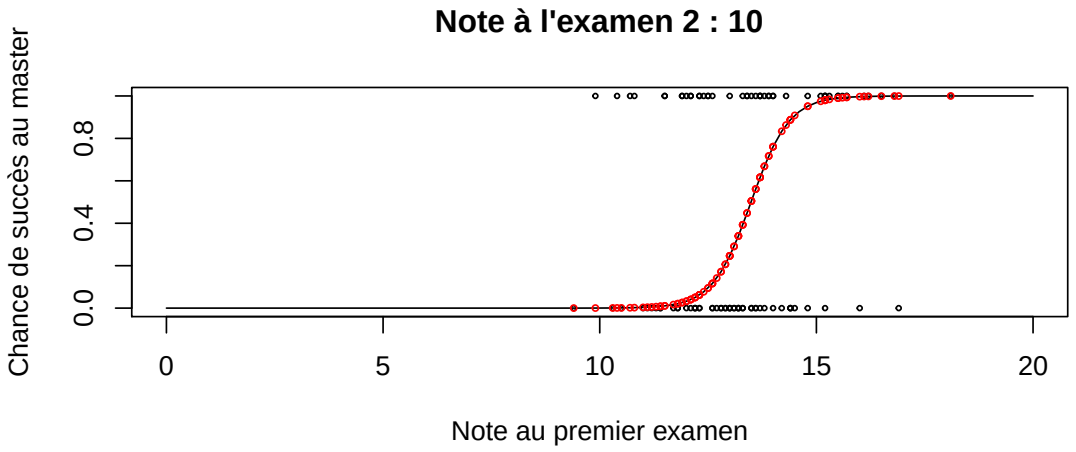
```
## $matconf
##
##      0  1
##   0 44  8
##   1  0 48
##
## $tbc
## [1] 0.92
##
## $tvp
## [1] 0.8462
##
## $tfp
## [1] 0
```

```
confusion(mod3, 0.9)  # Taux de faux positifs trop fort
```

```
## $matconf
##
##      0  1
##   0 51  1
##   1  7 41
##
## $tbc
## [1] 0.92
##
## $tvp
## [1] 0.9808
##
## $tfp
## [1] 0.1458
```

16. *Graphique des prédictions. Commentez et jouez avec les instructions inscrits sur l'énoncé :*

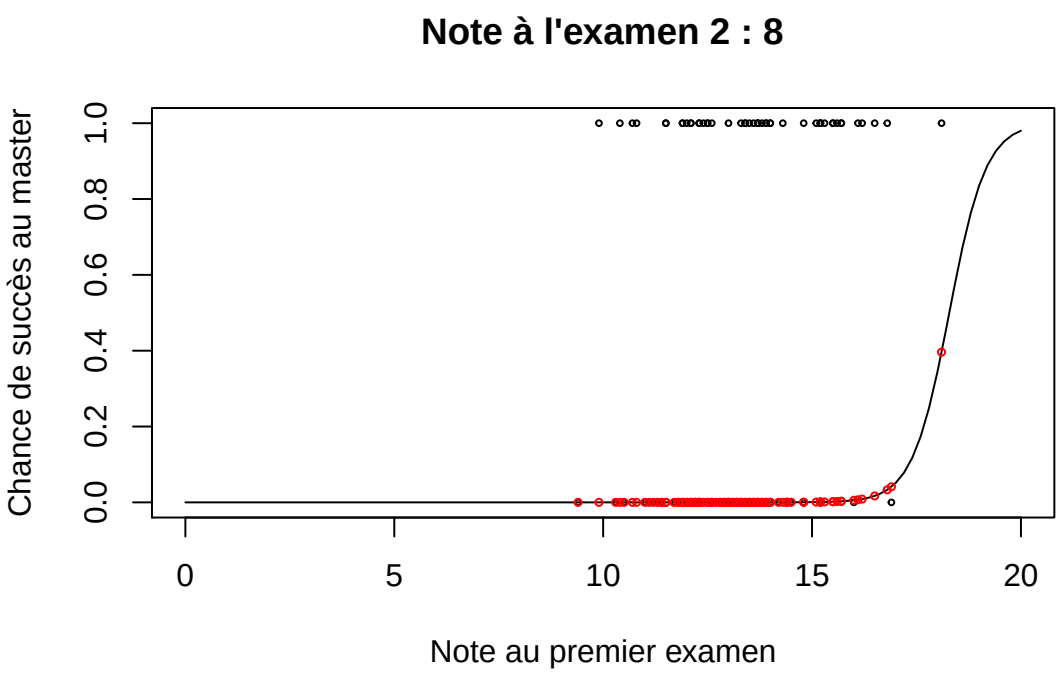




Constatons qu'avec une note de 12 à l'examen 2, nous aurons de très fortes chances d'obtenir le master, même si notre premier examen ne s'est pas bien déroulé. Si nous changeons la valeur de la deuxième note, en la fixant à 10 par exemple, alors une bonne note à l'examen 1 sera décisif.

Fixons désormais la deuxième note à 8 :

```
pred.df <- function(x) data.frame(x1 = x, x2 = 8)
plot(df1$yl ~ df1$x1, cex = 0.4, xlim = c(0, 20), xlab = "Note au premier examen",
    ylab = "Chance de succès au master", main = "Note à l'examen 2 : 8")
curve(predict(mod3, pred.df(x), type = "resp"), add = TRUE)
points(df1$x1, predict(mod3, pred.df(df1$x1), type = "resp"), col = "red", cex = 0.5)
```



Ici, même si nous obtenons une excellente note au premier devoir, nous aurons de très faible de chance de succès au master.

Nous pouvons nous rendre compte ici de l'importance de la covariable **x2**.

17. *Implémentez le test de Hosmer-Lemeshow (avec 10 catégories) et vérifier la sortie R avec la fonction hosltem.test (du paquet ResourceSelection). Commentez.*

1ère étape : On ordonne les probabilités estimées.

```
table1 = cbind(mod3$fitted.values, yl)
table2 <- table1[order(table1[, 1]), ]
groupe <- c(rep(1, 10), rep(2, 10), rep(3, 10), rep(4, 10), rep(5, 10), rep(6,
    10), rep(7, 10), rep(8, 10), rep(9, 10), rep(10, 10))
table3 <- cbind(table2, groupe)
colnames(table3) = c("p estim.", "p obs.", "groupe")
head(table3)
```

```
##     p estim. p obs. groupe
## 68 1.449e-13      0      1
## 27 1.725e-10      0      1
## 93 3.792e-10      0      1
## 83 4.749e-09      0      1
## 63 3.220e-08      0      1
## 20 4.043e-08      0      1
```

2nde étape : on sépare les probas estimées en k sous groupes (k=10).

```
require(ResourceSelection)
```

```
## Loading required package: ResourceSelection
## ResourceSelection 0.2-3 2013-06-18
```

```
k <- 10
ck <- 0
for (i in (1:k)) {
    ok <- length(table3[, 1][groupe == i & table3[, 2] == 1])
    mk <- length(table3[, 1][groupe == i])
    muk <- mean(table3[, 1][groupe == i])
    ck[i] <- (ok - mk * muk)^2/(mk * muk * (1 - muk))
}
sum(ck)
```

```
## [1] 4.999
```

```
hoslem.test(yl, mod3$fitted.values, g = 10)  # Vérification
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  yl, mod3$fitted.values
## X-squared = 4.999, df = 8, p-value = 0.7577
```

Pour rappel, H_1 : Le modèle n'est pas correct.
La statistique de test C^2, sous H_0, suit approximativement une loi de $khi^2(k-2)$.
Ici, on ne peut pas dire que le modèle n'est pas correct (il n'est pas nécessairement bon !).

18. *Construire une fonction R permettant de prendre en entrée un vecteur de seuils entre 0 et 1 et le modèle estimé et qui en sortie trace la courbe ROC. Tracez les courbes ROC du modèle à deux covariables et du modèle avec les quatres covariables. Commentez.*

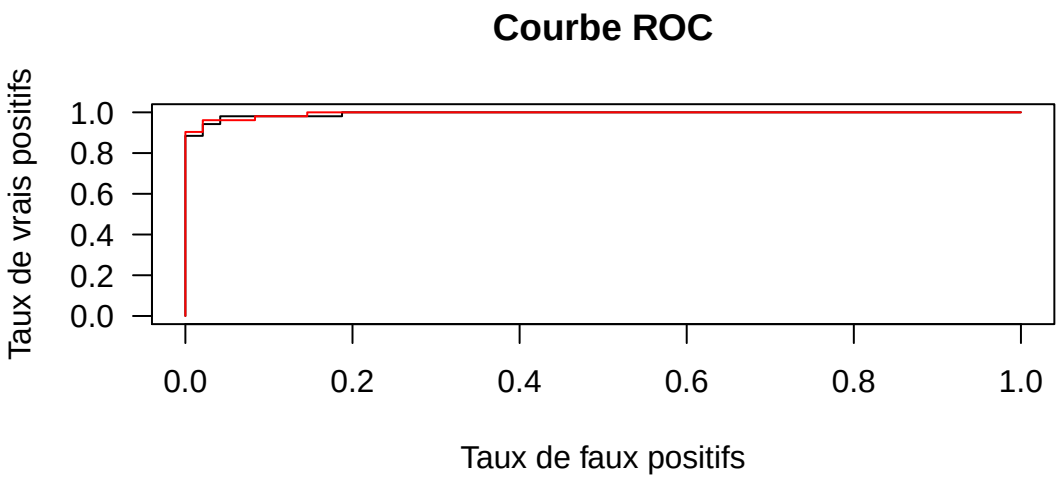
Note : Le critère ROC est un critère tourné vers la prédiction (la qualité du prédicteur) et non sur le sélection de variables.

Nous construisons la fonction suivante :

```
roc <- function(mod, vectseuil, add = FALSE) {
    tvp <- array(dim = c(length(vectseuil), 1))
    tfp <- array(dim = c(length(vectseuil), 1))
    for (i in (1:length(vectseuil))) {
        conf <- table(mod$y, as.numeric(mod$fitted.values > vectseuil[i]))
        tvp[i] <- conf[1, 1]/(conf[1, 1] + conf[1, 2])
        tfp[i] <- conf[2, 1]/(conf[2, 1] + conf[2, 2])
        vtvp <- c(0, tvp, 1)
        vtfp <- c(0, tfp, 1)
    }
    if (!add) {
        plot(vtfp, vtvp, type = "l", xlim = c(0, 1), ylim = c(0,
            1), xlab = "Taux de faux positifs", ylab = "Taux de vrais positifs",
            las = 1, main = "Courbe ROC")
    } else points(vtfp, vtvp, type = "l", col = "red")
}
```

Tracons désormais les 2 courbes ROC désirées :

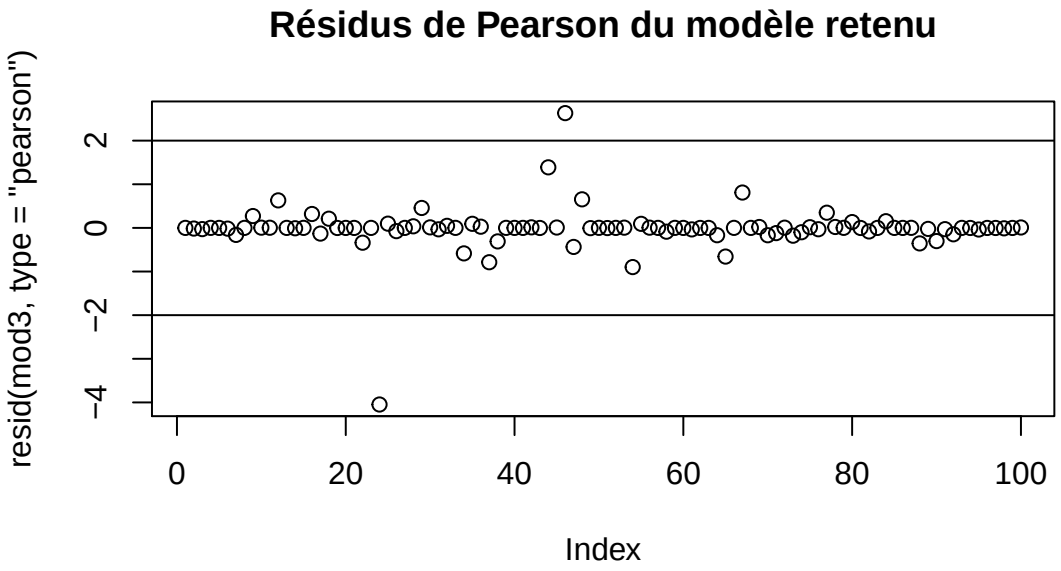
```
roc(mod3, seq(0.01, 0.99, 0.01))
roc(modcomplet, seq(0.01, 0.99, 0.01), add = T)
```



La courbe ROC est souvent utilisé pour comparer plusieurs classificateurs.
Plus l'AUC (Area Under Curve = Aire Sous la Courbe) est forte, meilleur est le classificateur.
Ici, le modèle à 2 covariables et le modèle complet semble être de qualité (prédictive) semblable.

19. *Du modèle à 2 covariables, tirez les résidus de Pearson, tracez-les et commentez.*

```
plot(resid(mod3, type = "pearson"), main = "Résidus de Pearson du modèle retenu")
abline(h = c(-2, 2))
```

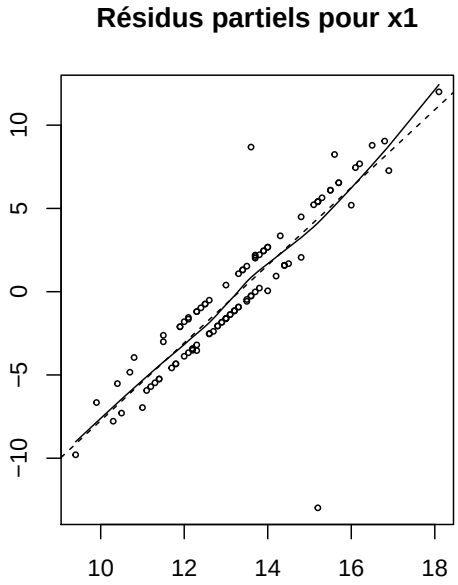


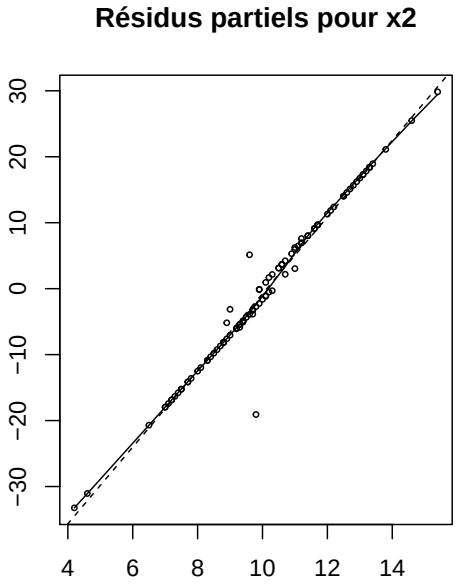
On constate que les résidus ne sont pas corrélés entre eux et que seules 2 valeurs sont extrêmes.
Rien de choquant, seulement 5% de valeurs aberrantes. S'il y avait eu trop de points aberrants, le modèle n'aurait peut-être pas été adéquat.

20. *Que font les instructions suivantes ? Interprétez. Qu'en est-il de la covariable* ***x2*** *?*

```
par(mfrow = c(1, 2))
#### x1 ####
residpartiels <- resid(mod3, type = "partial")
prov <- loess(residpartiels[, "x1"] ~ x1)
ordre <- order(x1)
plot(x1, residpartiels[, "x1"], type = "p", cex = 0.5, xlab = "",
    ylab = "", main = "Résidus partiels pour x1")
matlines(x1[ordre], predict(prov)[ordre])
abline(lsfit(x1, residpartiels[, "x1"]), lty = 2)

#### x2 ####
residpartiels <- resid(mod3, type = "partial")
prov <- loess(residpartiels[, "x2"] ~ x2)
ordre <- order(x2)
plot(x2, residpartiels[, "x2"], type = "p", cex = 0.5, xlab = "",
    ylab = "", main = "Résidus partiels pour x2")
matlines(x2[ordre], predict(prov)[ordre])
abline(lsfit(x2, residpartiels[, "x2"]), lty = 2)
```





Ces instructions tracent les résidus partiels pour la covariable **x1** d'une part et de **x2** d'autre part.

Les tracés sont bels et bien linéaires (matlines et abline se confondent), tout est donc normal, les deux variables sont bien construites.

Si ce n'était pas le cas, il aurait fallu remplacer les variables par une fonction de celles-ci.

x1 et **x2** sont donc de bons prédicteurs linéaires.

21. *Extraire du modèle estimé, les coefficients h_{ii}. Tracez-les. Votre jeu de données contient-ils des points leviers ? Comment s'interprètent-ils ?*

En pratique, on dit que l'individu i est un point levier si $h_{ii} > \frac{4p}{n}$ (ou $\frac{3p}{n}$).

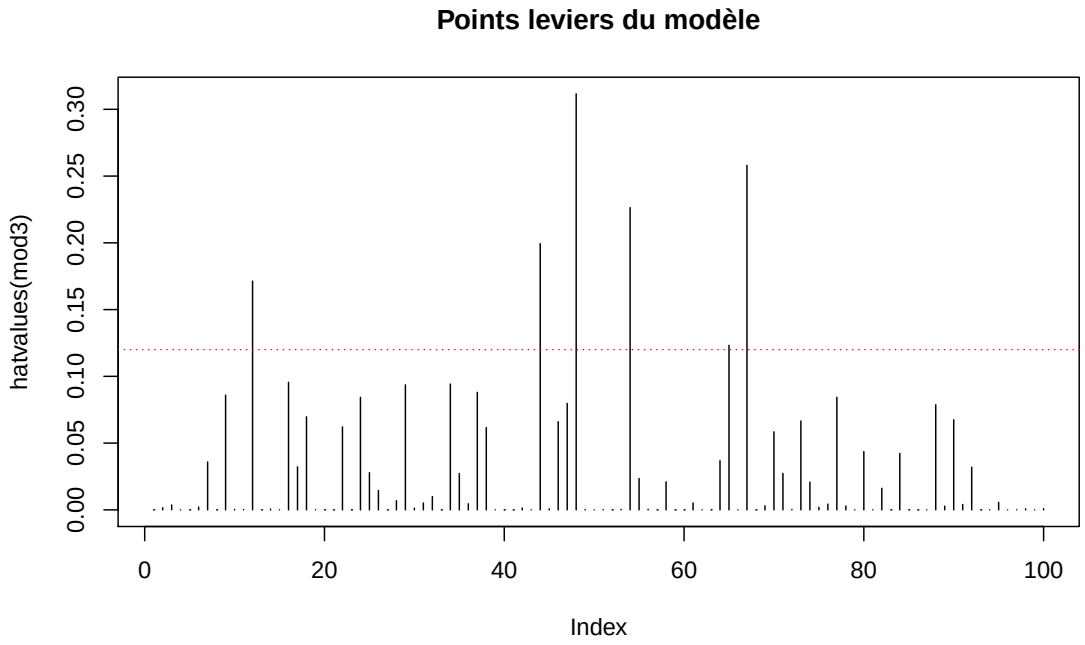
```
head(influence(mod3)$hat)  # Methode 1
```

```
##         1         2         3         4         5         6
## 4.931e-07 1.606e-03 3.662e-03 1.929e-05 5.899e-06 2.116e-03
```

```
head(hatvalues(mod3))  # Methode 2
```

```
##         1         2         3         4         5         6
## 4.931e-07 1.606e-03 3.662e-03 1.929e-05 5.899e-06 2.116e-03
```

```
plot(hatvalues(mod3), type = "h", main = "Points leviers du modèle")
p <- 3  # Nombre de parametres du modele
n <- 100  # Nombre d'individus
abline(h = 4 * p/n, col = "red", lty = "dotted")
```



Pour retrouver la Hat.Matrix :

```
W <- diag(mod3$fitted.values * (1 - mod3$fitted.values))
X <- cbind(rep(1, 100), x1, x2)
H <- sqrt(W) %*% X %*% (solve(t(X) %*% W %*% X)) %*% t(X) %*%
    sqrt(W)

a <- array(dim = c(100, 1))
for (i in (1:100)) {
    a[i] <- H[i, i]
}
a[1:5]  # Vérification
```

```
## [1] 4.931e-07 1.606e-03 3.662e-03 1.928e-05 5.899e-06
```

Un point levier est un point qui participe à une hauteur importante à sa propre prédiction. Ici, on en dénombre 6.

22. *Calculez pour chaque individu la distance de Cook (vérifiez ce calcul pour le premier individu). Commentez.*

Un point influent est un point qui, quand il est supprimé, implique une grosse variation dans les estimations des paramètres. L'indicateur standard est la distance de Cook.
En pratique, on dit qu'un point est influent si $D_{cook_i} > 0.4$.
Pour trouver les distances de Cook pour chaque individu :

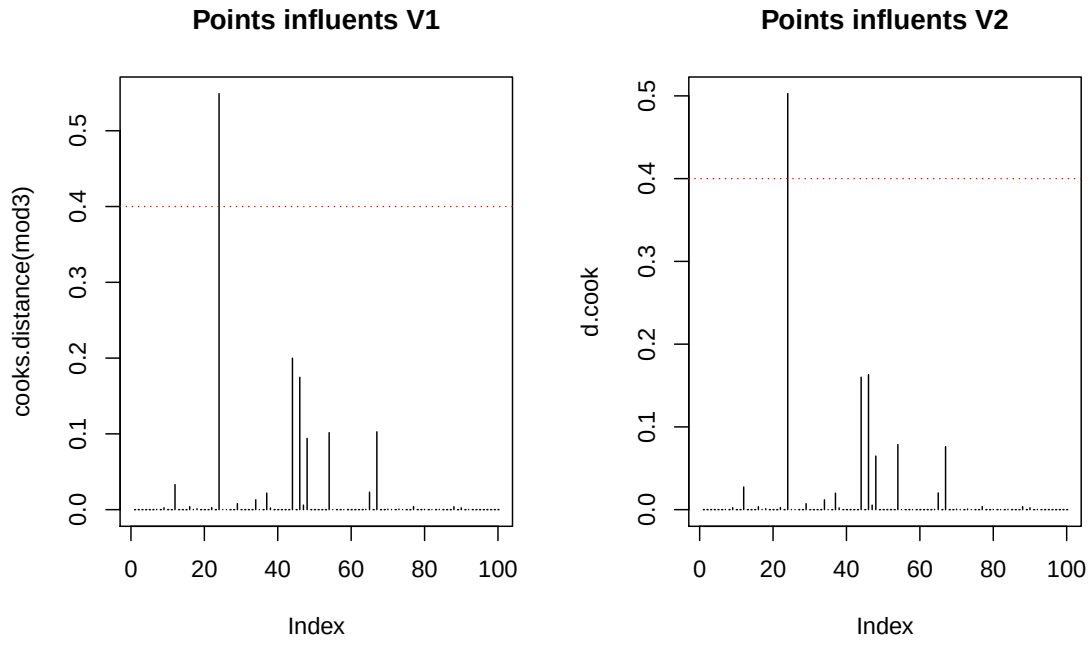
```
res.pearson.st <- resid(mod3, type = "pearson")/(sqrt(1 - hatvalues(mod3)))
d.cook <- (res.pearson.st^2) * (hatvalues(mod3)/3)  # Méthode 2
head(cooks.distance(mod3))  # Distance de Cook des 6 premiers individus avec V1
```

```
##         1         2         3         4         5         6
## 2.475e-15 1.157e-07 9.111e-07 6.272e-12 5.332e-13 1.909e-07
```

```
head(d.cook)  # Distance de Cook des 6 premiers individus avec V2
```

```
##         1         2         3         4         5         6
## 2.475e-15 1.155e-07 9.077e-07 6.272e-12 5.332e-13 1.905e-07
```

```
par(mfrow = c(1, 2))
plot(cooks.distance(mod3), type = "h", main = "Points influents V1")
abline(h = 0.4, col = "red", lty = "dotted")
plot(d.cook, type = "h", main = "Points influents V2")
abline(h = 0.4, col = "red", lty = "dotted")
```




```
c(which(d.cook > 0.4), max(d.cook))
```

```
##      24
## 24.0000  0.5027
```

Ici, nous ne trouvons qu'un seul point influent.
Selon le contexte, il est parfois préférable d'enlever/corriger les points aberrants et refaire l'analyse.

23. *Artificiellement, rajoutez à votre base de données initiale 5 individus aberrants. Reprenez les dernières étapes (résidus, points leviers et points influents) pour voir si ces indicateurs permettent de détecter ces données.*

Première approche : On commence par ajouter les points aberrants suivants :

(a) individu 101 : (x1=20,x2=0)

(b) individu 102 : (x1=20,x2=0)

(c) individu 103 : (x1=0,x2=20)

(d) individu 104 : (x1=0,x2=20)

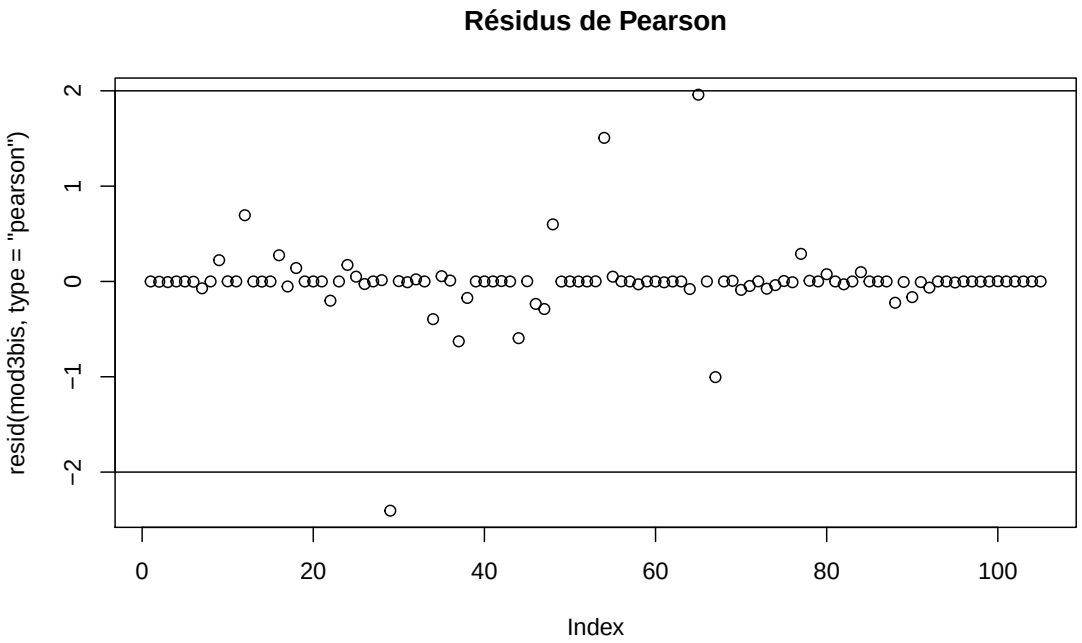
(e) individu 105 : (x1=20,x2=0)

```
x1b <- df1$x1
x2b <- df1$x2
x1b[101] <- 20
x1b[102] <- 20
x1b[103] <- 0
x1b[104] <- 0
x1b[105] <- 20
x2b[101] <- 0
x2b[102] <- 0
x2b[103] <- 20
x2b[104] <- 20
x2b[105] <- 0
beta0 <- -77
beta1 <- 2
beta2 <- 5
pb <- plogis(beta1 * x1b + beta2 * x2b + beta0, 0, 1)
ylb <- rbinom(105, 1, pb)
mod3bis = (glm(ylb ~ x1b + x2b, family = binomial(logit)))
summary(mod3bis)
```

```
##
## Call:
## glm(formula = ylb ~ x1b + x2b, family = binomial(logit))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9569  -0.0100   0.0000   0.0018   1.7758
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -114.38      40.87   -2.80   0.0051 **
## x1b             3.10       1.16    2.68   0.0074 **
## x2b             7.23       2.57    2.81   0.0050 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 145.094  on 104  degrees of freedom
## Residual deviance:  14.895  on 102  degrees of freedom
## AIC: 20.9
##
## Number of Fisher Scoring iterations: 10
```

On obtient les résidus de Pearson suivants :

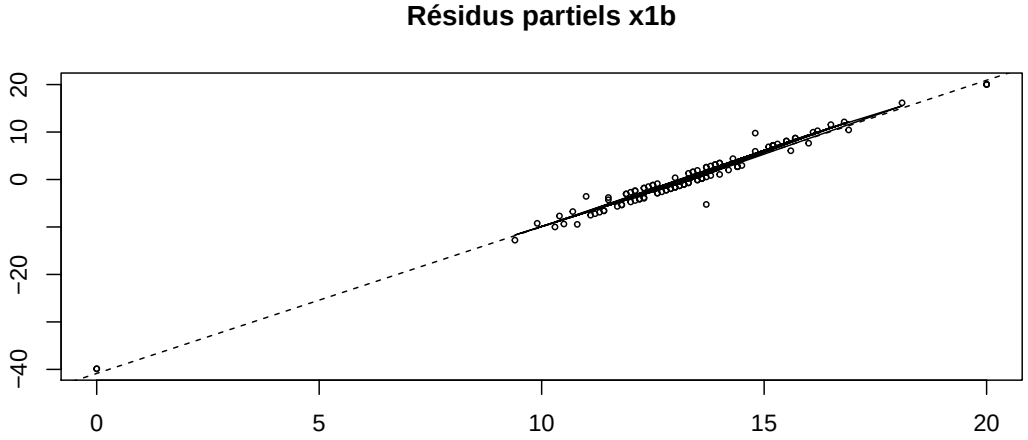
```
plot(resid(mod3bis, type = "pearson"), main = "Résidus de Pearson")
abline(h = c(-2, 2))
```



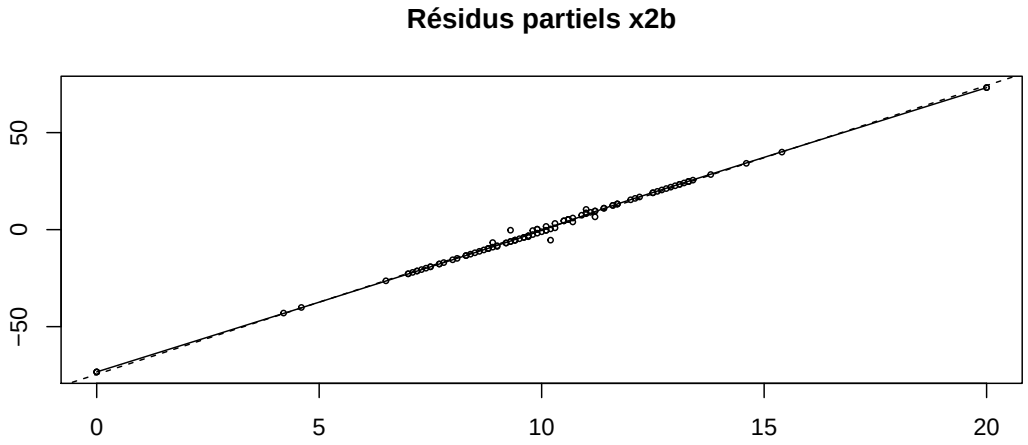
Les résidus de Pearson ne détectent pas nos nouveaux points aberrants.

On obtient les résidus partiels suivants (non propreté du graphique des résidus pour x1b) :

```
# Residus partiels (x1b)
residpartiels <- resid(mod3bis, type = "partial")
prov <- loess(residpartiels[, "x1b"] ~ x1b)
plot(x1b, residpartiels[, "x1b"], type = "p", cex = 0.5, xlab = "",
    ylab = "", main = "Résidus partiels x1b")
matlines(x1b[ordre], predict(prov)[ordre])
abline(lsfit(x1b, residpartiels[, "x1b"]), lty = 2)
```

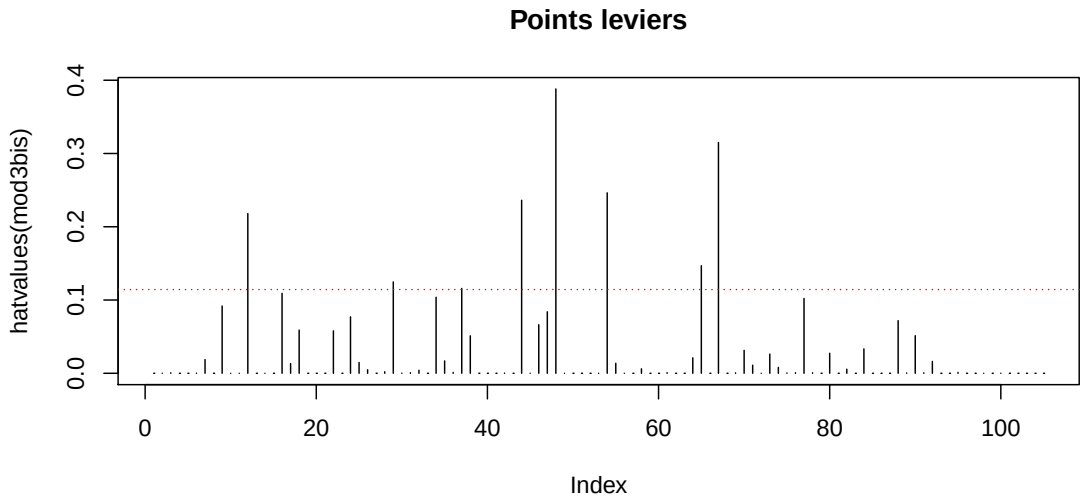


```r
# Residus partiels (x2b)
residpartiels <- resid(mod3bis, type = "partial")
prov <- loess(residpartiels[, "x2b"] ~ x2b)
ordre <- order(x2b)
plot(x2b, residpartiels[, "x2b"], type = "p", cex = 0.5, xlab = "",
    ylab = "", main = "Résidus partiels x2b")
matlines(x2b[ordre], predict(prov)[ordre])
abline(lsfit(x2b, residpartiels[, "x2b"]), lty = 2)
```



On obtient les points leviers suivants :

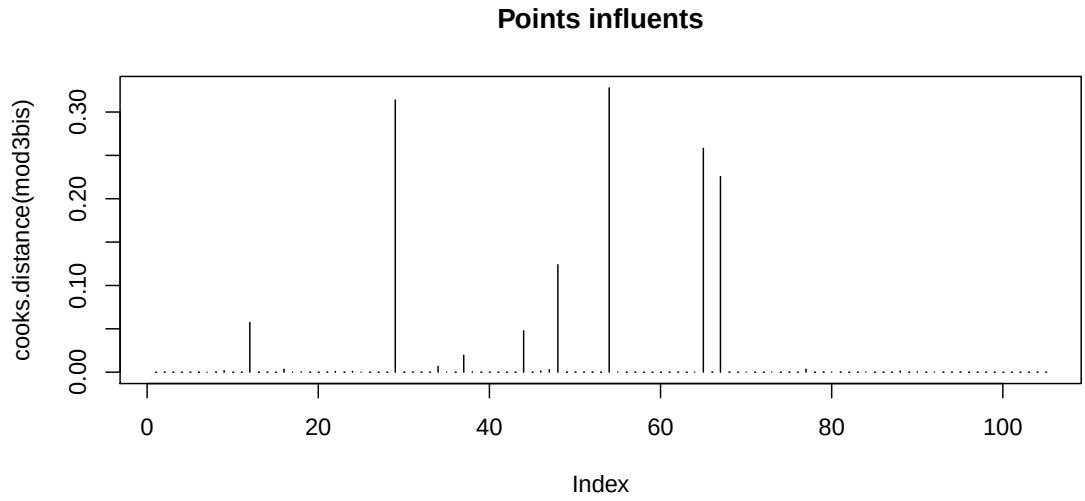
```r
# Points leviers
plot(hatvalues(mod3bis), type = "h", main = "Points leviers")
p <- 3
n <- 105
abline(h = 4 * p/n, col = "red", lty = "dotted")
```



Les nouveaux points ne sont pas des points leviers mais il y a plus de points leviers que précédemment (8 contre 6).

On obtient les points influents suivants :

```
# Points influents
plot(cooks.distance(mod3bis), type = "h", main = "Points influents")
abline(h = 0.4, col = "red", lty = "dotted")
```



Les nouveaux points ne sont pas des points influents.

Seconde approche : On décide de changer les valeurs de nos points aberrants et voir si des différences d'interprétations se produisent :

(a) individu 101 : (x1=0,x2=0)
(b) individu 102 : (x1=0,x2=0)
(c) individu 103 : (x1=20,x2=20)
(d) individu 104 : (x1=0,x2=0)
(e) individu 105 : (x1=20,x2=20)

```
x1b <- dfl$x1
x2b <- dfl$x2
x1b[101] <- 0
x1b[102] <- 0
x1b[103] <- 20
x1b[104] <- 0
x1b[105] <- 20
x2b[101] <- 0
x2b[102] <- 0
x2b[103] <- 20
x2b[104] <- 0
x2b[105] <- 20
c(length(x1b), length(x2b))
```

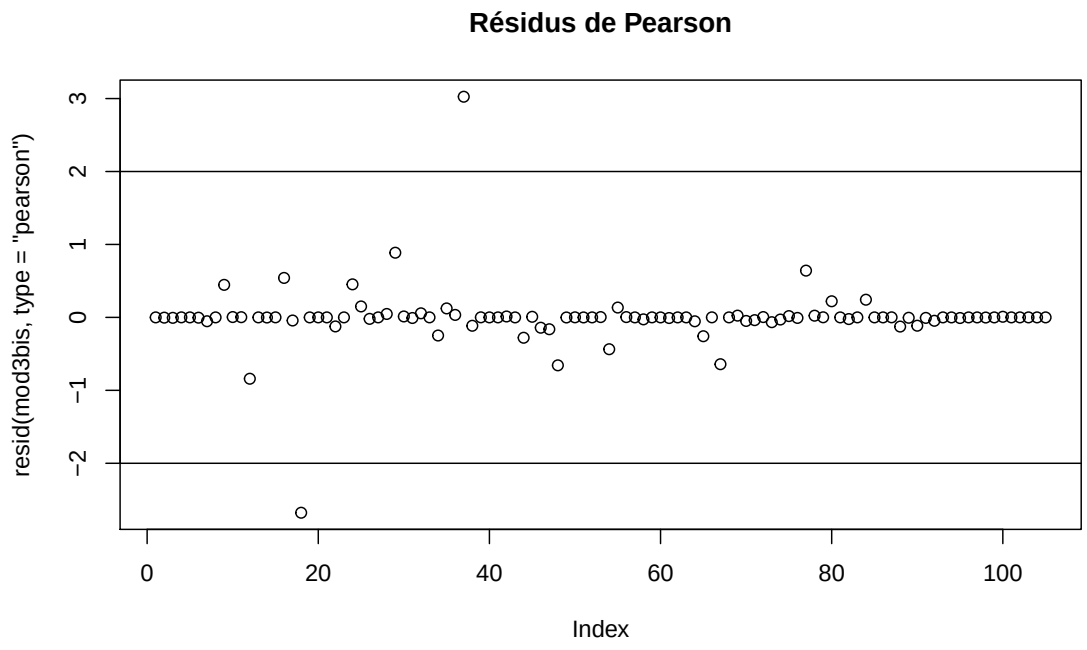
```
## [1] 105 105
```

```
beta0 <- -77
beta1 <- 2
beta2 <- 5
pb <- plogis(beta1 * x1b + beta2 * x2b + beta0, 0, 1)
ylb <- rbinom(105, 1, pb)
mod3bis = (glm(ylb ~ x1b + x2b, family = binomial(logit)))
summary(mod3bis)
```

```
##
## Call:
## glm(formula = ylb ~ x1b + x2b, family = binomial(logit))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0497  -0.0120   0.0000   0.0018   2.1530
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -101.860     35.020   -2.91   0.0036 **
## x1b            2.625      0.944    2.78   0.0054 **
## x2b            6.484      2.240    2.90   0.0038 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 143.947  on 104  degrees of freedom
## Residual deviance:  15.744  on 102  degrees of freedom
## AIC: 21.74
##
## Number of Fisher Scoring iterations: 10
```

On obtient les résidus de Pearson suivants :

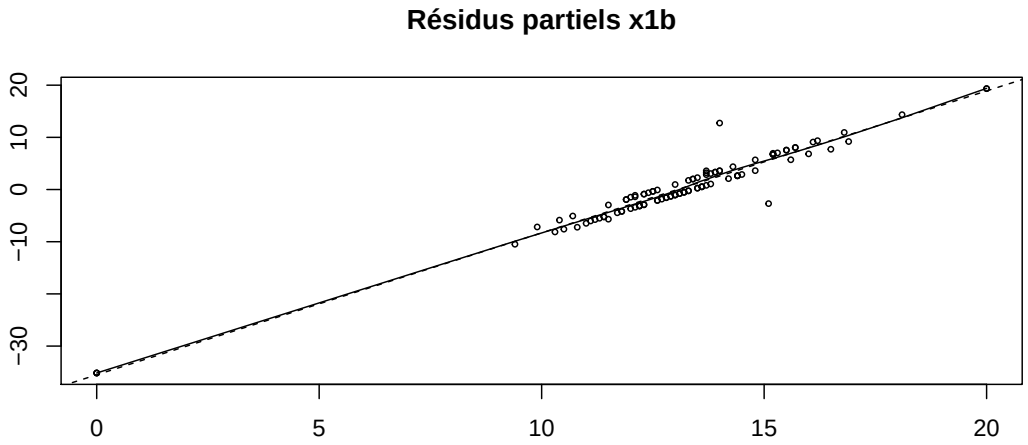
```
plot(resid(mod3bis, type = "pearson"), main = "Résidus de Pearson")
abline(h = c(-2, 2))
```



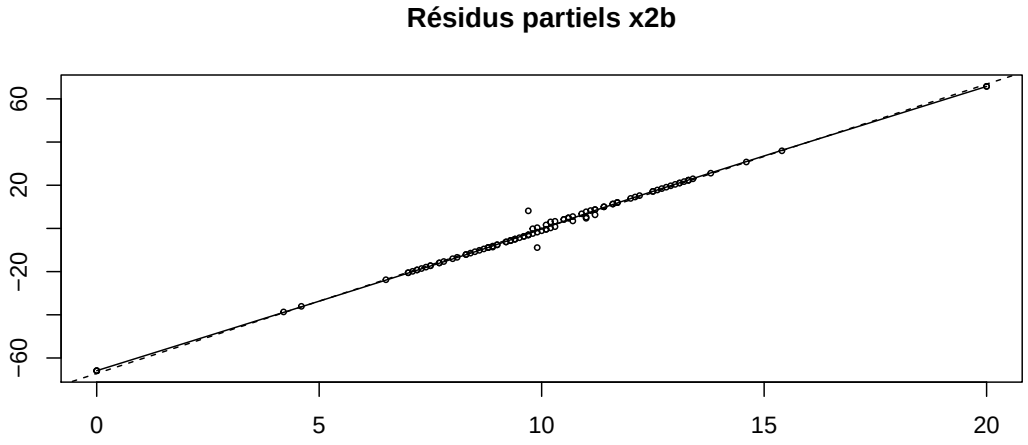
Les résidus de Pearson ne détectent pas nos nouveaux points aberrants.

On obtient les résidus partiels suivants :

```
# Residus partiels (x1b)
residpartiels <- resid(mod3bis, type = "partial")
prov <- loess(residpartiels[, "x1b"] ~ x1b)
ordre <- order(x1b)
plot(x1b, residpartiels[, "x1b"], type = "p", cex = 0.5, xlab = "",
    ylab = "", main = "Résidus partiels x1b")
matlines(x1b[ordre], predict(prov)[ordre])
abline(lsfit(x1b, residpartiels[, "x1b"]), lty = 2)
```

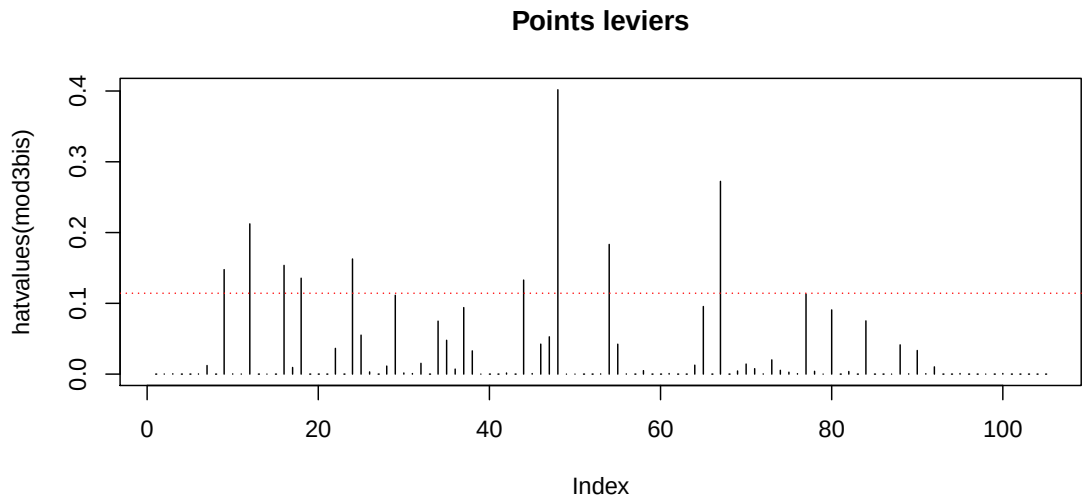



```
# Residus partiels (x2b)
residpartiels <- resid(mod3bis, type = "partial")
prov <- loess(residpartiels[, "x2b"] ~ x2b)
ordre <- order(x2b)
plot(x2b, residpartiels[, "x2b"], type = "p", cex = 0.5, xlab = "",
    ylab = "", main = "Résidus partiels x2b")
matlines(x2b[ordre], predict(prov)[ordre])
abline(lsfit(x2b, residpartiels[, "x2b"]), lty = 2)
```



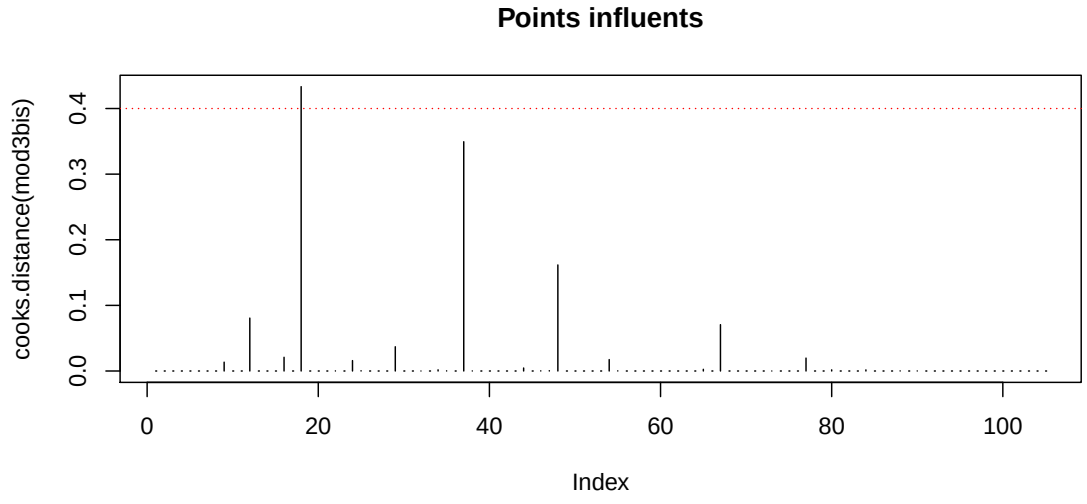
On obtient les points leviers suivants :

```
# Points leviers
plot(hatvalues(mod3bis), type = "h", main = "Points leviers")
p <- 3
n <- 105
abline(h = 4 * p/n, col = "red", lty = "dotted")
```



Les nouveaux points ne sont pas des points leviers, mais on obtient une dizaine de points leviers sur 100.
On obtient les points influents suivants :

```
# Points influents
plot(cooks.distance(mod3bis), type = "h", main = "Points influents")
abline(h = 0.4, col = "red", lty = "dotted")
```



Les nouveaux points ne sont pas des points influents.

En résumé, les nouveaux points que l'on a souhaité aberrants ne sont pas détectés par les méthodes de points leviers et de points influents.

2 Exercice

Cet exercice consiste à analyser le jeu de données icu. Les données ICU (Intensive Care Unit) sont constituées d'un échantillon de 256 patients admis dans un service hospitalier de soins intensifs pour adultes (ICU). L'objectif de l'étude est de mettre en place un modèle de régression logistique pour prédire la probabilité de survie de ces patients et les facteurs de risque associés à la mortalité dans un service de soins intensifs. Les intitutilés des variables sont les suivants :

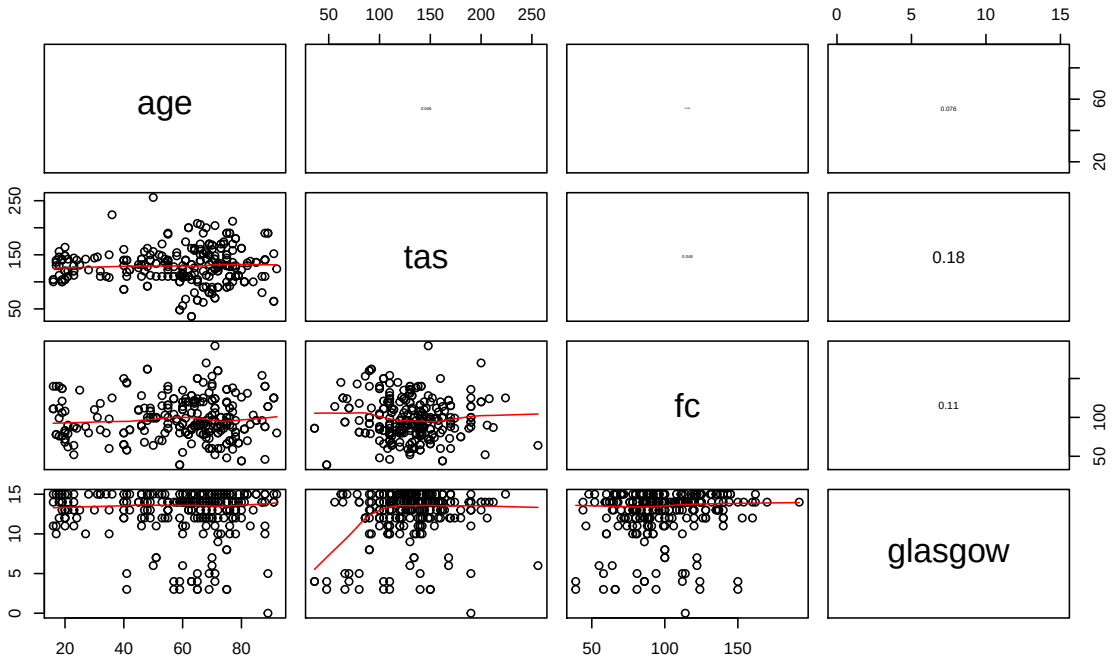
1. *DECEDE : statut vital (0=vivant, 1=mort)*
2. *AGE : age en années*
3. *SEXE : sexe (0=homme, 1=femme)*
4. *SERV : service lors de l'admission en USI (0=Medical, 1=Chirurgical)*
5. *INF_{JO} : infection probable à l'admission en USI (0=Non, 1=Oui)*
6. *TAS : tension artérielle systolique à l'admission en USI (en mm Hg)*
7. *FC : fréquence cardiaque à l'admission en USI (battements/min)*
8. *TYP_{AD} : type d'admission (0=normal, 1=en urgence)*
9. *PO2 : PO2 de la gazométrie artérielle initiale (0>60, 1=60)*
10. *PH : PH de la gazométrie artérielle initiale (0=7.25, 1<7.25)*
11. *BICAR : bicarbonate de la gazométrie artérielle initiale (0=18,1<18)*
12. *CONSC : niveau de conscience à l'admission (0=pas de coma ni stupeur, 1=stupeur profonde, 2=coma)*
13. *GLASGOW : score de Glasgow*

2.1 Phase préliminaire

Cherchons à savoir si les variables quantitatives de ce jeu de données sont corrélées. Pour cela, on va représenter le nuage de points et calculer son coefficient de corrélation entre chaque couple de variables quantitatives.

```
load(file = "icu.RData")
# Analyse des variables quantitatives :
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor,
    ...) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste(prefix, txt, sep = "")
    if (missing(cex.cor))
        cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}
```

```
pairs(icu[, c(2, 6, 7, 13)], lower.panel = panel.smooth, upper.panel = panel.cor)
```



On constate qu'il n'y a pas de forte corrélation entre les variables quantitatives.

2.2 Analyse par régressions logistiques

Nous allons maintenant faire une régression logistique multiple entre notre variable à expliquer (**DECEDE**) et nos variables explicatives.

```
mod.icu = (glm(decede ~ ., family = binomial(logit), data = icu))
summary(mod.icu)$coef
```

```
##                Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) 24.5703433  15.606733  1.57434 0.1154083
## age          0.0367197   0.010282  3.57112 0.0003555
## sexe        -0.2827532   0.363454 -0.77796 0.4365918
## serv        -0.2482798   0.391276 -0.63454 0.5257290
## inf_j0      -0.1279735   0.372743 -0.34333 0.7313507
## tas         -0.0090790   0.005618 -1.61617 0.1060585
## fc           0.0006266   0.006946  0.09021 0.9281195
## typ_ad       1.7157688   0.619848  2.76805 0.0056393
## po2          0.0074107   0.009049  0.81894 0.4128204
## ph          -4.0009453   2.125282 -1.88255 0.0597617
## bicar       -0.0079477   0.028147 -0.28236 0.7776672
## consc        1.9140063   0.768002  2.49219 0.0126959
## glasgow      0.0576718   0.124118  0.46465 0.6421791
```

Les variables les plus significativement liées au décès du patient sont l'âge, le type d'admission, le niveau de conscience à l'admission.

Cherchons désormais à faire autant de régressions logistiques simples qu'il y a de variables explicatives.

```
# En les prenant une à une :
mod.icuF <- array(dim = c(12, 2))
for (i in (2:13)) {
    mod.icuF[i - 1, 1] = summary(glm(icu[, 1] ~ icu[, i],
        family = binomial(logit)))$coef[2, 1]
    mod.icuF[i - 1, 2] = summary(glm(icu[, 1] ~ icu[, i],
        family = binomial(logit)))$coef[2, 4]
}
colnames(mod.icuF) = c("Coeff est.", "p-value")
rownames(mod.icuF) = c("AGE", "SEXE", "SERV", "INF_JO",
    "TAS", "FC", "TYP_AD", "PO2", "PH", "BICAR", "CONSC",
    "GLASGOW")
mod.icuF
```

```
##         Coeff est.   p-value
## AGE       0.030606 4.652e-04
## SEXE      0.088411 7.621e-01
## SERV     -0.962842 1.057e-03
## INF_JO    0.676425 1.811e-02
## TAS      -0.016464 4.630e-04
## FC        0.004202 4.260e-01
## TYP_AD    1.990314 2.209e-04
## PO2      -0.002680 7.233e-01
## PH       -2.448810 1.556e-01
## BICAR    -0.029870 2.115e-01
## CONSC     1.707376 2.071e-07
## GLASGOW  -0.240543 2.679e-07
```

Les variables significatives sont : AGE, SERV, INF_{JO}, TAS, TYP_{AD}, CONSC et GLASGOW.

Attardons-nous sur le signe des coefficients :
Le coefficient de l'**âge** est positif : plus le patient est âgé et plus les risques de décès augmentent.
Le coefficient de l'**infection probable** est positif : si le patient a de forts risques d'infection lors de l'admission en USI, plus il aura de risque de décès.
Le coefficient de la tension artérielle est négatif : plus la tension artérielle est faible, plus les risques de décès augmentent.
Le coefficient du **type d'admission** est positive : si le patient a été admis en urgence, alors il aura plus de risque de décéder.
Pour l'échelle de **Glasgow**, il est normal d'avoir un coefficient négatif pour cette variable, plus le score est bas, plus le risque de décéder est élevé.
Analyse inverse concernent le **niveau de conscience** à l'admission, tel qu'il a été codé.

Remarquons qu'on retrouve ici la variable explicative **GLASGOW** que l'on ne retrouvait pas lors de notre régression multiple. On se doute qu'elle est fortement liée avec la variable **CONSC**, l'état de conscience. En effet, l'échelle de Glasgow est une échelle allant de 3 (coma profond) à 15 (personne parfaitement consciente), et qui s'évalue sur trois critères.
On peut vérifier cette forte corrélation par un test du khi^2 :

```
chisq.test(icu$consc, icu$glasgow)
```

```
##
##  Pearson's Chi-squared test
##
## data:  icu$consc and icu$glasgow
## X-squared = 512, df = 26, p-value < 2.2e-16
```

2.3 Sélection de variables

Pour déterminer les variables que nous allons choisir pour la suite de l'analyse, on fait une sélection par le critère de l'AIC, méthode ascendante.
Lors des régressions simples, nous avons vu que la variable qui était la plus significative avec **DECEDE** était **CONSC**.
Notre modèle de départ sera donc le modèle avec cette seule variable explicative. Le modèle d'arrivée sera le modèle complet.

```
mod.icu.ini = (glm(decede ~ consc, family = binomial(logit), data = icu))
mod.icu.complet = (glm(decede ~ ., family = binomial(logit), data = icu))
```

```
step(mod.icu.ini, direction = "forward", scope = list(upper = formula(mod.icu.complet)))
```

```
## Start:  AIC=257.6
## decede ~ consc
##
##           Df Deviance AIC
## + typ_ad   1      240 246
## + age      1      242 248
## + serv     1      245 251
## + tas      1      248 254
## + ph       1      250 256
## + inf_j0   1      251 257
## + fc       1      252 258
## <none>             254 258
## + glasgow  1      253 259
## + bicar    1      253 259
## + po2      1      254 260
## + sexe     1      254 260
##
## Step:  AIC=245.7
## decede ~ consc + typ_ad
##
##           Df Deviance AIC
## + age      1      224 232
## + ph       1      236 244
## + tas      1      237 245
## <none>             240 246
## + serv     1      238 246
## + inf_j0   1      238 246
## + fc       1      239 247
## + glasgow  1      239 247
## + sexe     1      239 247
## + bicar    1      239 247
## + po2      1      240 248
##
## Step:  AIC=231.5
## decede ~ consc + typ_ad + age
##
##           Df Deviance AIC
## + ph       1      220 230
## + tas      1      221 231
## <none>             224 232
## + sexe     1      222 232
## + serv     1      223 233
```

```
## + bicar    1      223 233
## + fc       1      223 233
## + inf_j0   1      223 233
## + glasgow  1      223 233
## + po2      1      223 233
##
## Step:  AIC=230.2
## decede ~ consc + typ_ad + age + ph
##
##           Df Deviance AIC
## + tas      1      217 229
## <none>             220 230
## + sexe     1      219 231
## + serv     1      220 232
## + po2      1      220 232
## + glasgow  1      220 232
## + bicar    1      220 232
## + fc       1      220 232
## + inf_j0   1      220 232
##
## Step:  AIC=229.1
## decede ~ consc + typ_ad + age + ph + tas
##
##           Df Deviance AIC
## <none>             217 229
## + sexe     1      216 230
## + serv     1      216 230
## + po2      1      217 231
## + glasgow  1      217 231
## + bicar    1      217 231
## + inf_j0   1      217 231
## + fc       1      217 231
##
## Call:  glm(formula = decede ~ consc + typ_ad + age + ph + tas, family = binomial(logit),
##     data = icu)
##
## Coefficients:
## (Intercept)        consc       typ_ad          age           ph
##    24.24064      1.54030      1.84666      0.03548     -3.82029
##         tas
##    -0.00934
##
## Degrees of Freedom: 255 Total (i.e. Null);  250 Residual
## Null Deviance:     296
## Residual Deviance: 217  AIC: 229
```

Lors des différentes étapes de sélection de variables, constatons que **GLASGOW** perd de son sens car très fortement corrélé avec **CONSC**.

Ici, on retient seulement : **AGE** + **CONSC** + TYP_{AD} + **PH** + **TAS** selon le critère de l'AIC par méthode ascendant.

Remarque : Quelle que soit la méthode choisie (ascendante, descendante, dans les deux directions), le modèle retenu est le même.

Cependant, nous avons vu que le **PH** n'était pas une variable significativement explicative, donc le modèle final retenu sera le suivant : **AGE** + **CONSC** + TYP_{AD} + **TAS**.

```
mod.icu.choix <- (glm(decede ~ age + tas + typ_ad + consc, family = binomial(logit),
    data = icu))
summary(mod.icu.choix)
```

```
##
## Call:
## glm(formula = decede ~ age + tas + typ_ad + consc, family = binomial(logit),
##     data = icu)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.301  -0.710  -0.412   0.349   2.339
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.93819    1.13170   -3.48  0.00050 ***
## age          0.03567    0.00976    3.65  0.00026 ***
## tas         -0.00881    0.00532   -1.65  0.09799 .
## typ_ad       1.86034    0.55908    3.33  0.00088 ***
## consc        1.48484    0.36404    4.08  4.5e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 296.38  on 255  degrees of freedom
## Residual deviance: 220.63  on 251  degrees of freedom
## AIC: 230.6
##
## Number of Fisher Scoring iterations: 5
```

2.4 Quelques tests

On effectue un test de Hosmer-Lemeshow pour savoir si le modèle retenu n'est pas mauvais.

```
hoslem.test(icu[, 1], mod.icu.choix$fitted.values, g = 10)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  icu[, 1], mod.icu.choix$fitted.values
## X-squared = 14.67, df = 8, p-value = 0.06587
```

On ne peut pas accepter H1. Autrement dit, on ne peut pas dire que le modèle n'est pas correct (il n'est pas nécessairement bon !).

Appliquons ensuite le test de déviance : il a pour même finalité que le test précédent.

```
1 - pchisq(mod.icu.choix$dev, df = 256 - 5)
```

```
## [1] 0.9169
```

Avec une p-valeur aussi forte, on ne peut accepter H_1 : on ne peut pas dire que le modèle n'est pas correct.

Appliquons désormais un test de différence de déviance dans le cas de modèles emboités.
Par rapport au modèle complet (13 coefficients) dans un premier temps :

```
1 - pchisq(mod.icu.choix$deviance - mod.icu$deviance, df = 13 - 5)
```

```
## [1] 0.6796
```

Ainsi, on ne peut pas accepter que le modèle complet soit significativement plus informatif que le modèle retenu.

Aussi, on peut s'interroger l'intérêt de la variable **PH** par exemple (puisqu'elle n'était pas sélectionnée en faisant le summary individuellement, alors qu'elle était sélectionnée avec la sélection de variables par AIC).

```
mod.icu.choixtest <- (glm(decede ~ age + tas + typ_ad + consc + ph, family = binomial(logit),
    data = icu))
1 - pchisq(mod.icu.choix$deviance - mod.icu.choixtest$deviance, df = 6 -
    5)
```

```
## [1] 0.05894
```

Ainsi, on ne peut pas accepter que le modèle contenant la variable **PH** soit significativement plus informatif que le modèle retenu.

Au final, on peut dire que le modèle retenu est cohérent compte-tenu des derniers tests réalisés.

2.5 Rapport de cote / Augmentation d'une unité

Quelle est l'influence sur les probabilités de décès si nous augmentons d'une unité la variable **AGE** ?

```
exp(mod.icu.choix$coeff[2])
```

```
##   age
## 1.036
```

La cote (rapport entre les probabilités de succès et d'échec) va être multipliée par 1.04.
Autrement dit, les risques de décès augmentent avec l'augmention d'une unité de la variable **AGE**.

Quelle est l'influence sur les probabilités de décès si nous augmentons d'une unité la variable **TAS** ?

```
exp(mod.icu.choix$coeff[3])
```

```
##    tas
## 0.9912
```

La cote va être multipliée par 0.99.
Autrement dit, les risques de décès diminuent avec l'augmention d'une unité de la variable **TAS**.

Quelle est l'influence sur les probabilités de décès si nous augmentons d'une unité la note TYP_{AD} ?

```
exp(mod.icu.choix$coeff[4])
```

```
## typ_ad
##  6.426
```

La cote va être multipliée par 6.42 !
Autrement dit, les risques de décès augmentent clairement avec l'augmention d'une unité de la variable TYP_{AD} (passage d'un type d'admission normal à un type d'admission en urgence).

Quelle est l'influence sur les probabilités de décès si nous augmentons d'une unité la variable **CONSC** ?

```
exp(mod.icu.choix$coeff[5])
```

```
## consc
## 4.414
```

La cote va être multipliée par 4.41 !
Autrement dit, les risques de décès augmentent clairement avec l'augmention d'une unité de la variable **CONSC**.

2.6 Utilité de la matrice d'information de Fischer

Cherchons maintenant à calculer la matrice d'information de Fischer.

```
summary(mod.icu.choix)$coef
```

```
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -3.938193   1.131695  -3.480 5.016e-04
## age          0.035668   0.009762   3.654 2.586e-04
## tas         -0.008806   0.005322  -1.655 9.799e-02
## typ_ad       1.860341   0.559076   3.328 8.762e-04
## consc        1.484837   0.364035   4.079 4.526e-05
```

```
fi <- dlogis(mod.icu.choix$coefficients[1] + mod.icu.choix$coefficients[2] *
    icu$age + mod.icu.choix$coefficients[3] * icu$tas + mod.icu.choix$coefficients[4] *
    icu$typ_ad + mod.icu.choix$coefficients[5] * icu$consc, 0, 1)
V <- 0
for (i in (1:256)) {
    tmp <- c(1, icu$age[i], icu$tas[i], icu$typ_ad[i], icu$consc[i])
    tmp2 <- t(t(tmp)) %*% tmp
    V <- V + fi[i] * tmp2
}

sqrt((diag(solve(V))))  # Verification
```

```
## [1] 1.131701 0.009763 0.005322 0.559084 0.364036
```

La matrice de Fischer nous permet notamment de calculer les intervalles de confiance de nos paramètres :

```
InfoFis <- sqrt(solve(V))
(ICb1 <- mod.icu.choix$coeff[2] + c(-1.96, 1.96) * InfoFis[2, 2])
```

```
## [1] 0.01653 0.05480
```

```
(ICb2 <- mod.icu.choix$coeff[3] + c(-1.96, 1.96) * InfoFis[3, 3])
```

```
## [1] -0.019236  0.001625
```

```
(ICb3 <- mod.icu.choix$coeff[2] + c(-1.96, 1.96) * InfoFis[4, 4])
```

```
## [1] -1.060  1.131
```

```
(ICb4 <- mod.icu.choix$coeff[3] + c(-1.96, 1.96) * InfoFis[5, 5])
```

```
## [1] -0.7223  0.7047
```

Le recours à la matrice de Fischer nous permet également de tester les coefficients, par exemple :

Le coefficient lié à la variable **CONSC** est-il strictement supérieur au coefficient lié à la variable **AGE** ?

Sous H_0, la statistique de test suit une loi normale (0,1).

```
(deltaEst.H0 <- (mod.icu.choix$coeff[5] - mod.icu.choix$coeff[2])/(sqrt((InfoFis[5,
    5] + InfoFis[2, 2] - 2 * InfoFis[5, 2])/256)))
```

```
## consc
##    39
```

```
1 - pnorm(deltaEst.H0)
```

```
## consc
##     0
```

On accepte H_1, c'est à dire que le coefficient lié à la variable **CONSC** soit strictement supérieur au coefficient lié à la variable **AGE** ?

2.7 Matrice de confusion

Poursuivons notre analyse avec la matrice de confusion de notre modèle.

```
confusion(mod.icu.choix, 0.5)
```

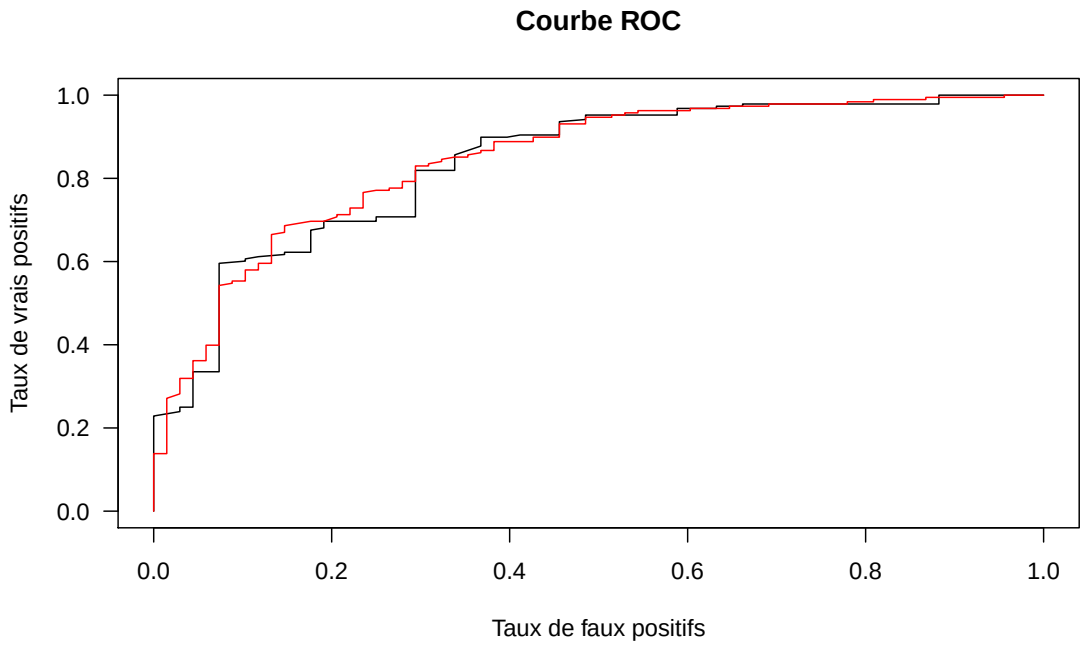
```
## $matconf
##
##       0   1
##   0 182   6
##   1  43  25
##
## $tbc
## [1] 0.8086
##
## $tvp
## [1] 0.9681
##
## $tfp
## [1] 0.6324
```

Le taux de bonne classification est bon.
Le taux de vrais positifs est excellent.
En revanche, le taux de faux positifs est bien trop élevé !.

2.8 Courbe ROC

Traçons désormais la courbe ROC associé à notre modèle et celle associée au modèle complet.

```
roc(mod.icu.choix, seq(0.01, 0.94, 0.001))
roc(mod.icu, seq(0.01, 0.97, 0.001), add = TRUE)
```

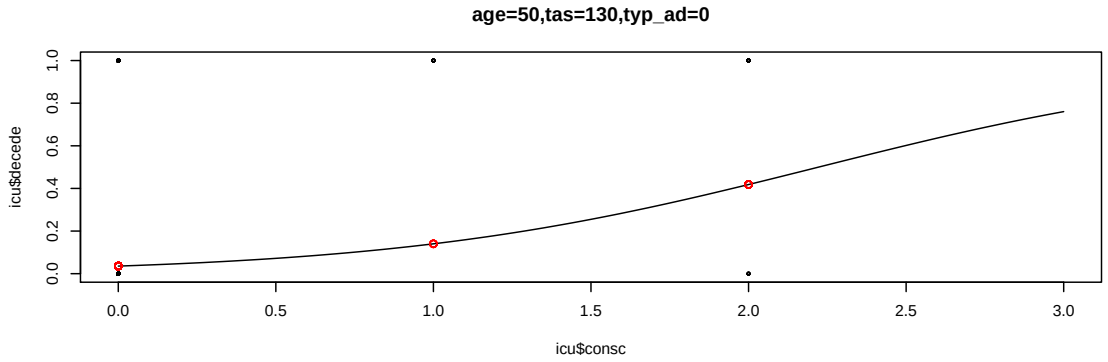


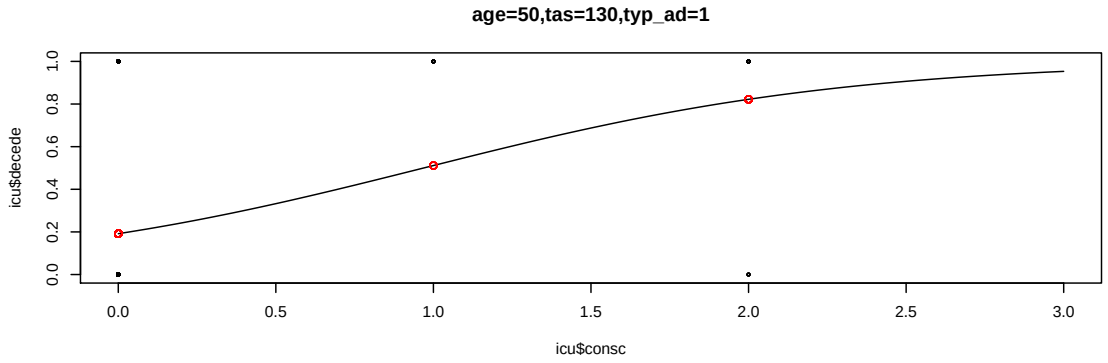
Les graphiques obtenus sont à peu près équivalents, notre modèle est un aussi bon prédicteur que le modèle complet.

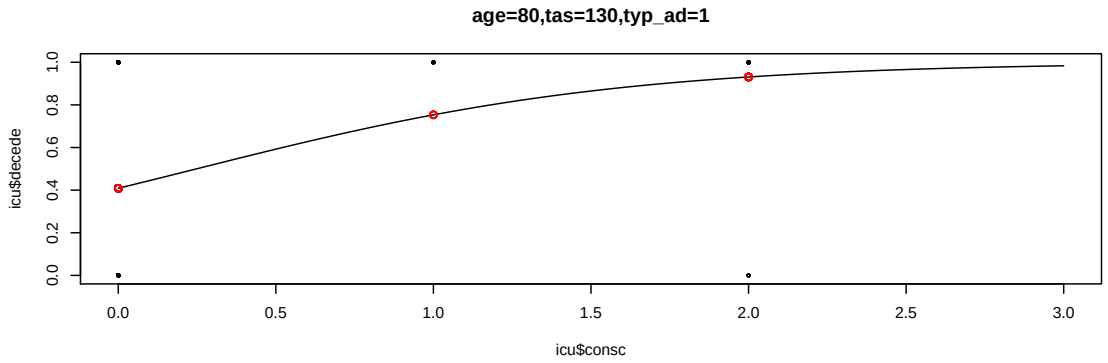
2.9 Graphiques de prédictions

Représentons désormais graphiquement des prédictions :
Commençons par jouer avec la variable **CONSC** :

```
par(mfrow = c(3, 1))
#####
pred.df <- function(x) data.frame(consc = x, age = 50, tas = 130, typ_ad = 0)
plot(icu$decede ~ icu$consc, cex = 0.4, xlim = c(0, 3), main = "age=50,tas=130,typ_ad=0")
curve(predict(mod.icu.choix, pred.df(x), type = "resp"), add = TRUE)
points(icu$consc, predict(mod.icu.choix, pred.df(icu$consc), type = "resp"),
    col = "red", cex = 1)
####
pred.df <- function(x) data.frame(consc = x, age = 50, tas = 130, typ_ad = 1)
plot(icu$decede ~ icu$consc, cex = 0.4, xlim = c(0, 3), main = "age=50,tas=130,typ_ad=1")
curve(predict(mod.icu.choix, pred.df(x), type = "resp"), add = TRUE)
points(icu$consc, predict(mod.icu.choix, pred.df(icu$consc), type = "resp"),
    col = "red", cex = 1)
####
pred.df <- function(x) data.frame(consc = x, age = 80, tas = 130, typ_ad = 1)
plot(icu$decede ~ icu$consc, cex = 0.4, xlim = c(0, 3), main = "age=80,tas=130,typ_ad=1")
curve(predict(mod.icu.choix, pred.df(x), type = "resp"), add = TRUE)
points(icu$consc, predict(mod.icu.choix, pred.df(icu$consc), type = "resp"),
    col = "red", cex = 1)
```





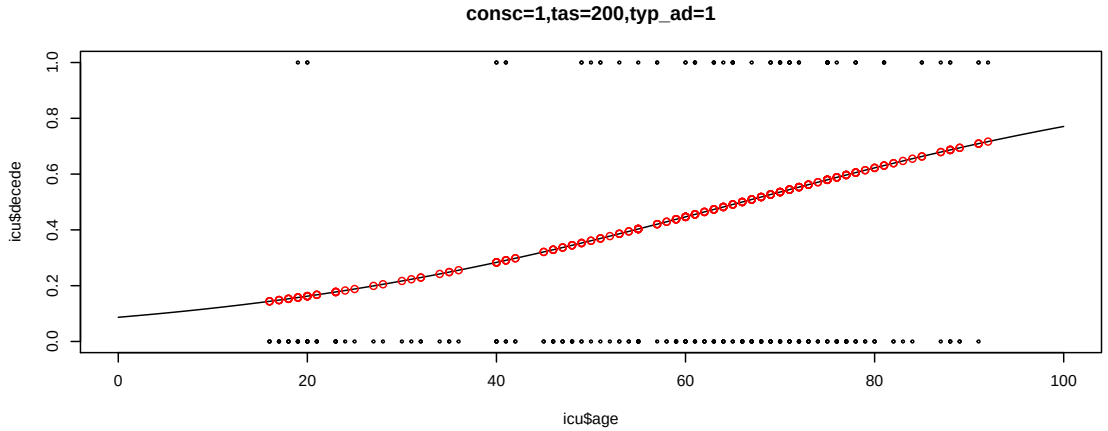


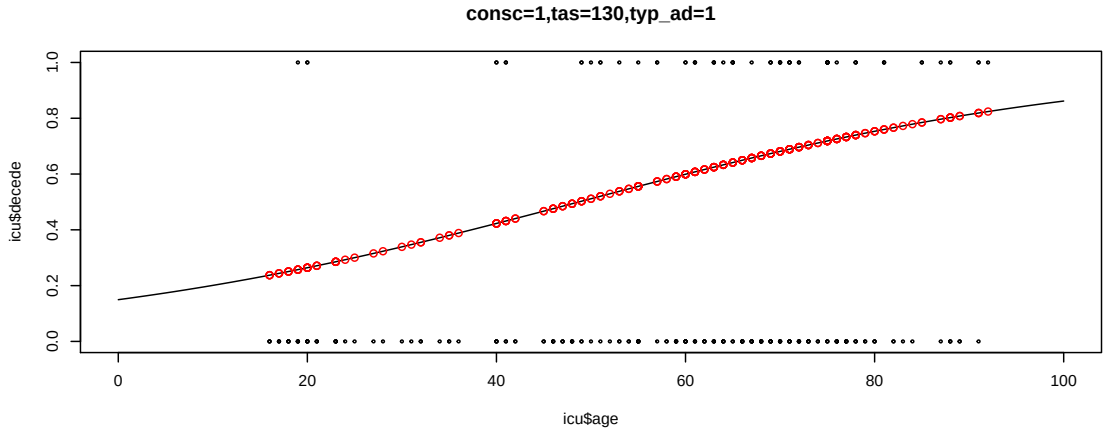
On constate que les probabilités de décès sont supérieures en supposant que le type d'admission du patient est urgent. En augmentant l'âge, les risques de décès augmentent également.

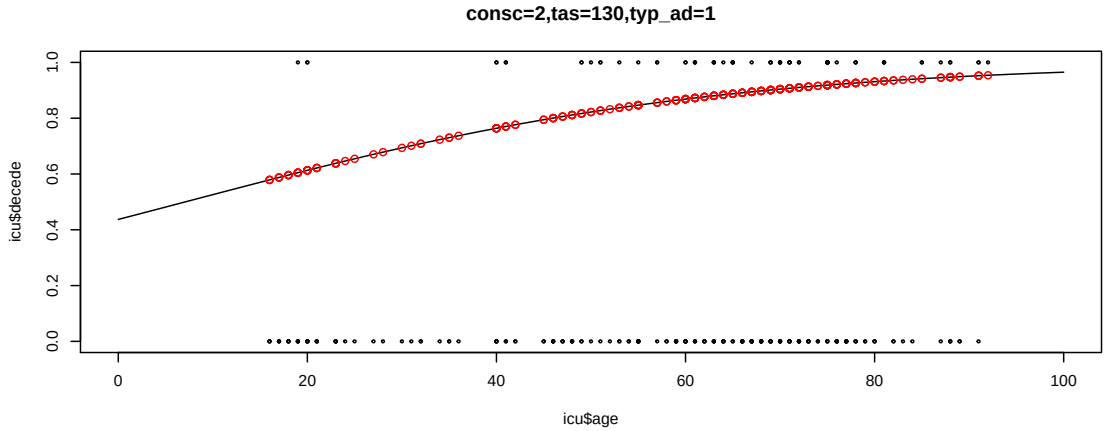
Jouons désormais avec la variable **AGE** :

```
par(mfrow = c(3, 1))
####
pred.df <- function(x) data.frame(age = x, consc = 1, tas = 200, typ_ad = 1)
plot(icu$decede ~ icu$age, cex = 0.4, xlim = c(0, 100), main = "consc=1,tas=200,typ_ad=1")
curve(predict(mod.icu.choix, pred.df(x), type = "resp"), add = TRUE)
points(icu$age, predict(mod.icu.choix, pred.df(icu$age), type = "resp"),
    col = "red", cex = 1)
#####
pred.df <- function(x) data.frame(age = x, consc = 1, tas = 130, typ_ad = 1)
plot(icu$decede ~ icu$age, cex = 0.4, xlim = c(0, 100), main = "consc=1,tas=130,typ_ad=1")
curve(predict(mod.icu.choix, pred.df(x), type = "resp"), add = TRUE)
points(icu$age, predict(mod.icu.choix, pred.df(icu$age), type = "resp"),
    col = "red", cex = 1)
```

```r
#####
pred.df <- function(x) data.frame(age = x, consc = 2, tas = 130, typ_ad = 1)
plot(icu$decede ~ icu$age, cex = 0.4, xlim = c(0, 100), main = "consc=2,tas=130,typ_ad=1")
curve(predict(mod.icu.choix, pred.df(x), type = "resp"), add = TRUE)
points(icu$age, predict(mod.icu.choix, pred.df(icu$age), type = "resp"),
    col = "red", cex = 1)
```





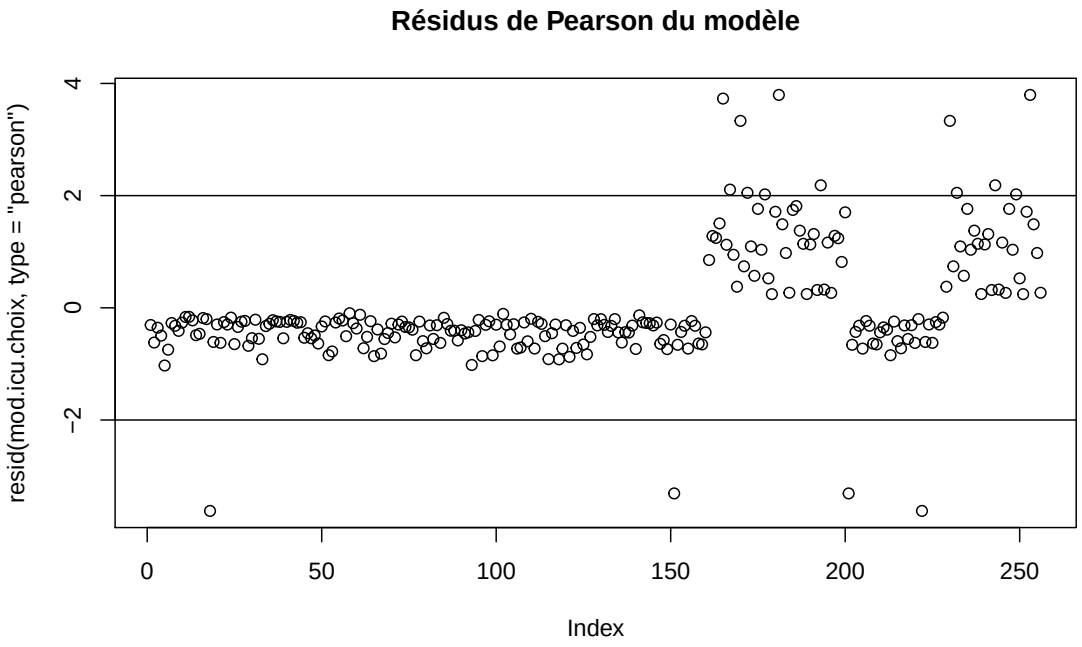


Ici, on constate que les risques de décès augmentent davantage avec une tension artérielle systolique à l'admission faible et avec un niveau de conscience à l'admission plus dramatique...

2.10 Analyse des résidus

Traçons les résidus de Pearson de notre modèle :

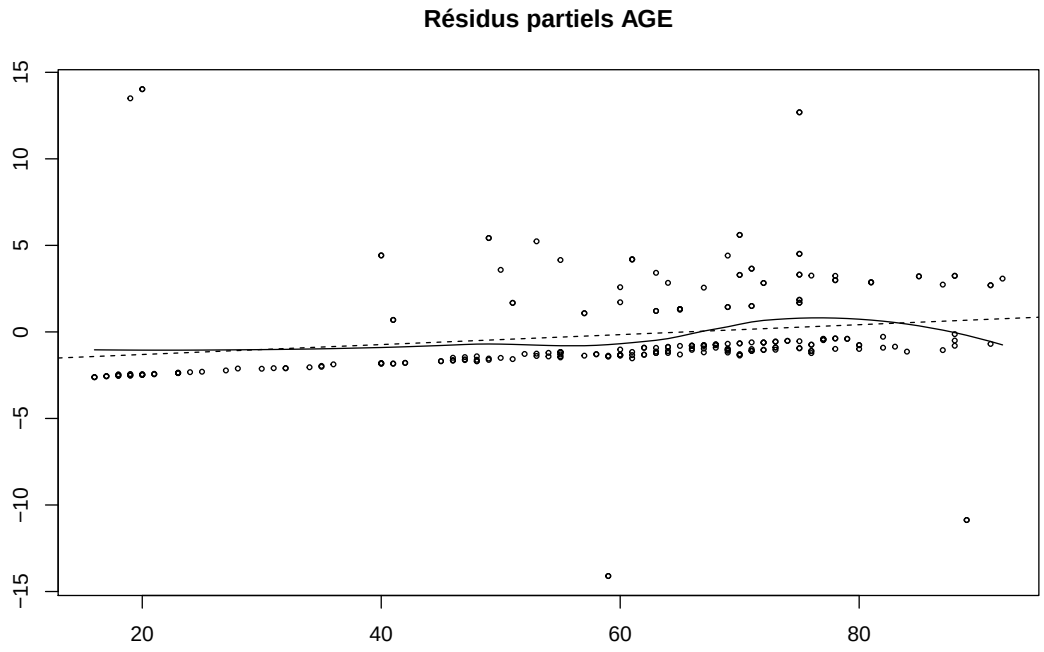
```
plot(resid(mod.icu.choix, type = "pearson"), main = "Résidus de Pearson du modèle")
abline(h = c(-2, 2))
```

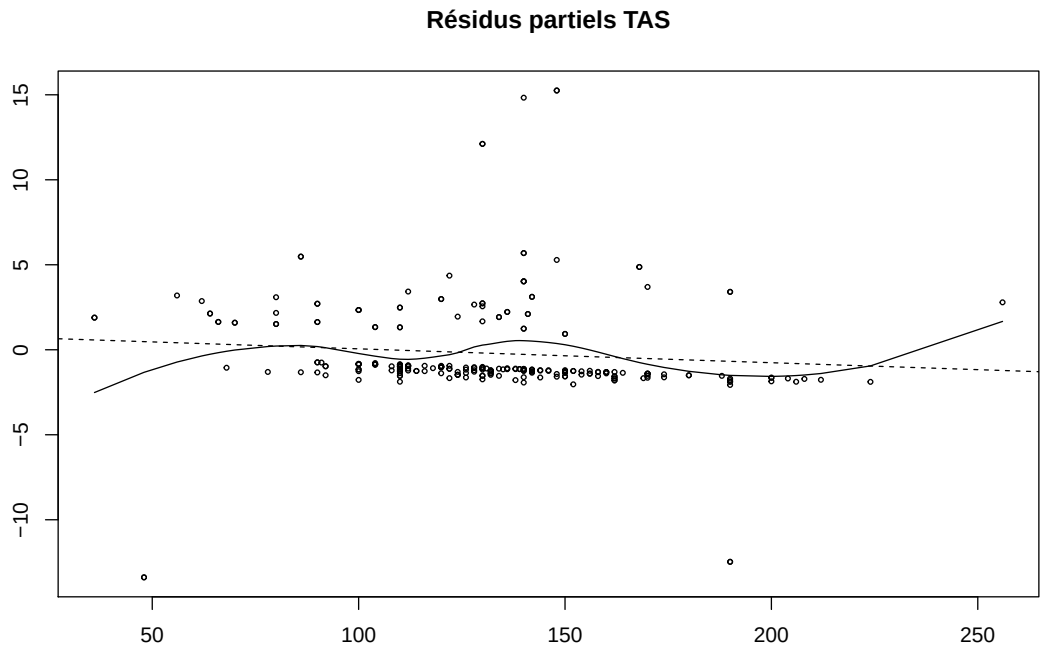


Une quinzaine de valeurs extrêmes, sur les 256, c'est légèrement supérieur à 5%, ce qui est un peu contraignant, sans en être choquant.

Concernant les résidus partiels (sur variables quantitatives) :

```
par(mfrow = c(2, 1))
# Pour la variable AGE :
residpartiels <- resid(mod.icu.choix, type = "partial")
prov <- loess(residpartiels[, "age"] ~ icu$age)
ordre <- order(icu$age)
plot(icu$age, residpartiels[, "age"], type = "p", cex = 0.5, xlab = "",
    ylab = "", main = "Résidus partiels AGE")
matlines(icu$age[ordre], predict(prov)[ordre])
abline(lsfit(icu$age, residpartiels[, "age"]), lty = 2)
# Pour la variable TAS :
residpartiels <- resid(mod.icu.choix, type = "partial")
prov <- loess(residpartiels[, "tas"] ~ icu$tas)
ordre <- order(icu$tas)
plot(icu$tas, residpartiels[, "tas"], type = "p", cex = 0.5, xlab = "",
    ylab = "", main = "Résidus partiels TAS")
matlines(icu$tas[ordre], predict(prov)[ordre])
abline(lsfit(icu$tas, residpartiels[, "tas"]), lty = 2)
```



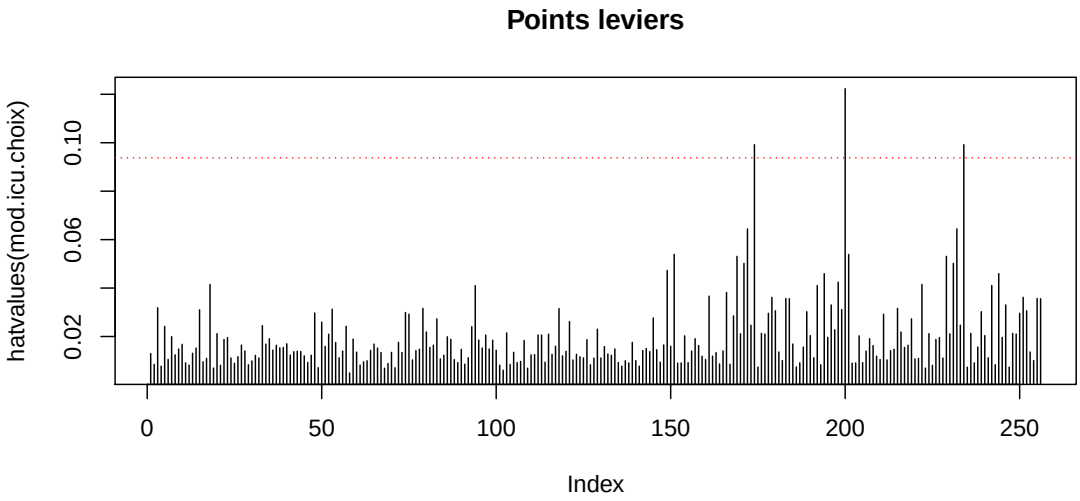


Les tracés pour les 2 variables ne sont pas franchement bien linéaires, mais ils sont acceptables. Nos deux variables sont donc d'assez bons prédicteurs.

2.11 Points leviers et influents

Concernant les points leviers, on trouve :

```
plot(hatvalues(mod.icu.choix), type = "h", main = "Points leviers")
p <- 6
n <- 256
abline(h = 4 * p/n, col = "red", lty = "dotted")
```



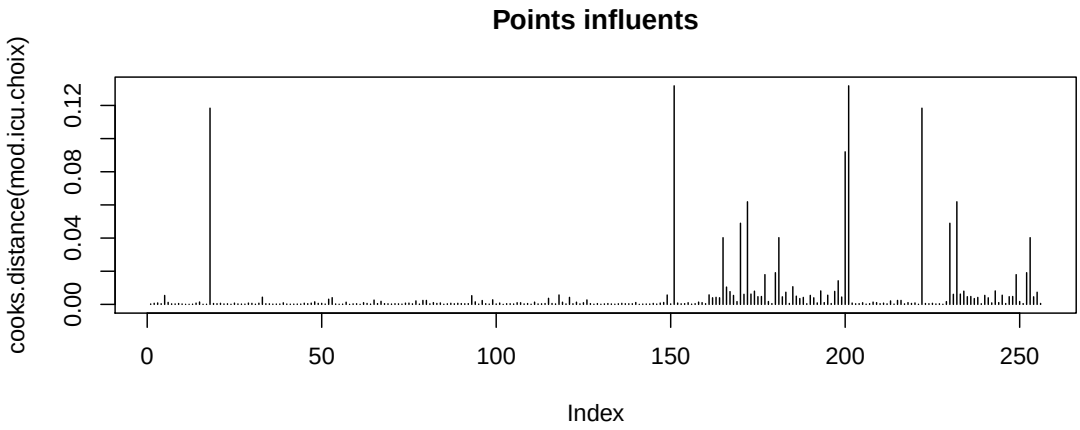

```
which(hatvalues(mod.icu.choix) > 4 * p/n)
```

```
## 174 200 234
## 174 200 234
```

Ici, on trouve 3 points leviers, c'est à dire 3 points qui participent à une hauteur importante à leur propre prédiction.

Concernant les points influents, nous n'en dénotons aucun.

```
plot(cooks.distance(mod.icu.choix), type = "h", main = "Points influents")
abline(h = 0.3, col = "red", lty = "dotted")
```



Note finale : Merci pour ce semestre d'enseignement !