Projet CS

TANIEL Rémi - GIS2A4

Contents

Traitements préliminaires

Chargement des données

On commence par charger les données contenu dans le fichier `hepatite.Rda`:

```
load("./hepatite.Rda")
```

##Résumer les données

Pour résumer les données, on peut utiliser la fonction `str` qui renseigne le nombre d'observations / variables, et qui pour chaque variable donne son type et quelques exemples de valeurs prises :

```
str(d)
```

```
## 'data.frame':    155 obs. of  20 variables:
##  $ class          : Factor w/ 2 levels "die","live": 2 2 2 2 2 2 1 2 2 2 ...
##  $ age            : int  30 50 78 31 34 34 51 23 39 30 ...
##  $ sex            : Factor w/ 2 levels "female","male": 1 2 2 2 2 2 2 2 2 2 ...
##  $ steriod        : Factor w/ 2 levels "no","yes": 1 1 2 NA 2 2 1 2 2 2 ...
##  $ antivirals     : Factor w/ 2 levels "no","yes": 2 2 2 1 2 2 2 2 2 2 ...
##  $ fatigue        : Factor w/ 2 levels "no","yes": 2 1 1 2 2 2 1 2 1 2 ...
##  $ malaise        : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ anorexia       : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 1 2 2 2 ...
##  $ liver_big      : Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 2 2 2 2 ...
##  $ liver_firm     : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 1 2 ...
##  $ spleen_palpable: Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 1 2 2 2 ...
##  $ spiders        : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 1 2 2 2 ...
```

```
##  $ asites         : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ varices        : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ bilirubin      : num  1 0.9 0.7 0.7 1 0.9 NA 1 0.7 1 ...
##  $ alk_phosphate  : int  85 135 96 46 NA 95 NA NA NA NA ...
##  $ sgot           : int  18 42 32 52 200 28 NA NA 48 120 ...
##  $ albumin        : num  4 3.5 4 4 4 4 NA NA 4.4 3.9 ...
##  $ protime        : int  NA NA NA 80 NA 75 NA NA NA NA ...
##  $ histology      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

On remarque donc que notre dataframe contient donc 155 observations pour 20 variables dont 13 variables qualitatives: `class`, `sex`, `steriod`, `antivirals`, `fatigue`, `malaise`, `anorexia`, `liver_big`, `liver_firm`, `spleen_palpable`, `spiders`, `asites`, `varices` et `histology`

On peut également faire un `summary` afin de savoir si notre jeu de données contient des valeurs manquantes / indéfinis `NA`:

```
summary(d)
```

```
##   class          age           sex      steriod   antivirals fatigue
##  die : 32   Min.   : 7.0   female: 16   no  :76   no : 24    no  :100
##  live:123   1st Qu.:32.0   male  :139   yes :78   yes:131    yes : 54
##             Median :39.0                NA's: 1              NA's:  1
##             Mean   :41.2
##             3rd Qu.:50.0
##             Max.   :78.0
##
##  malaise   anorexia   liver_big  liver_firm spleen_palpable spiders
##  no  :61   no  : 32   no  : 25   no  :60    no  : 30        no  :51
##  yes :93   yes :122   yes :120   yes :84    yes :120        yes :99
##  NA's: 1   NA's:  1   NA's: 10   NA's:11    NA's:  5        NA's: 5
##
##
##
##
##   asites    varices      bilirubin     alk_phosphate         sgot
##  no  : 20   no  : 18   Min.   :0.300   Min.   : 26.00   Min.   : 14.00
##  yes :130   yes :132   1st Qu.:0.700   1st Qu.: 74.25   1st Qu.: 31.50
##  NA's:  5   NA's:  5   Median :1.000   Median : 85.00   Median : 58.00
##                        Mean   :1.428   Mean   :105.33   Mean   : 85.89
##                        3rd Qu.:1.500   3rd Qu.:132.25   3rd Qu.:100.50
##                        Max.   :8.000   Max.   :295.00   Max.   :648.00
##                        NA's   :6       NA's   :29       NA's   :4
##     albumin         protime       histology
##  Min.   :2.100   Min.   :  0.00   no :85
##  1st Qu.:3.400   1st Qu.: 46.00   yes:70
##  Median :4.000   Median : 61.00
##  Mean   :3.817   Mean   : 61.85
##  3rd Qu.:4.200   3rd Qu.: 76.25
##  Max.   :6.400   Max.   :100.00
##  NA's   :16      NA's   :67
```

Données pouvant poser un problème

On remarque que la plupart de nos variables contiennent des valeurs non définis `NA`, de plus nos variables ne sont pas toutes du même type.

Ajustement d'un premier modéle de régression logistique

Ajustement des NA's

On décide de voir ce que fait la fonction `glm` pour les valeurs `NA`, on s'aide donc de la documentation de celle-ci :

```
help(glm)
```

Suivant la documentation, on utilise le paramètre `na.action` pour définir comment sont gérés les NA's, ce paramètre prends différentes valeures : * `na.omit` / `na.exclude`: les observations ayant des valeurs manquantes seronts supprimées * `na.pass`: n'effectue aucun changement sur le jeu de données * `na.fail` (par défaut): lève une exception si le jeu de données contient des valeurs manquantes

On décide de former un nouveau de données en enlevant les observations contenant des valeurs manquantes :

```
d2 <- na.omit(d)
str(d2)
```

```
## 'data.frame':    80 obs. of  20 variables:
##  $ class          : Factor w/ 2 levels "die","live": 2 2 2 2 2 2 2 2 2 2 ...
##  $ age            : int  34 39 32 41 30 38 40 38 38 22 ...
##  $ sex            : Factor w/ 2 levels "female","male": 2 2 2 2 2 2 2 2 2 1 ...
##  $ steriod        : Factor w/ 2 levels "no","yes": 2 1 2 2 2 1 1 2 1 2 ...
##  $ antivirals     : Factor w/ 2 levels "no","yes": 2 1 1 1 2 2 2 2 1 1 ...
##  $ fatigue        : Factor w/ 2 levels "no","yes": 2 2 1 1 1 1 1 2 2 1 ...
##  $ malaise        : Factor w/ 2 levels "no","yes": 2 2 2 2 2 1 2 2 2 2 ...
##  $ anorexia       : Factor w/ 2 levels "no","yes": 2 2 2 2 2 1 2 2 2 2 ...
##  $ liver_big      : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 2 2 1 2 ...
##  $ liver_firm     : Factor w/ 2 levels "no","yes": 2 1 1 1 1 2 1 2 1 2 ...
##  $ spleen_palpable: Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ spiders        : Factor w/ 2 levels "no","yes": 2 2 1 2 2 2 2 2 2 2 ...
##  $ asites         : Factor w/ 2 levels "no","yes": 2 2 2 2 2 1 2 2 2 2 ...
##  $ varices        : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ bilirubin      : num  0.9 1.3 1 0.9 2.2 2 0.6 0.7 0.7 0.9 ...
##  $ alk_phosphate  : int  95 78 59 81 57 72 62 53 70 48 ...
##  $ sgot           : int  28 30 249 60 144 89 166 42 28 20 ...
##  $ albumin        : num  4 4.4 3.7 3.9 4.9 2.9 4 4.1 4.2 4.2 ...
##  $ protime        : int  75 85 54 52 78 46 63 85 62 64 ...
##  $ histology      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:75] 1 2 3 4 5 7 8 9 10 15 ...
##   ..- attr(*, "names")= chr [1:75] "1" "2" "3" "4" ...
```

On obtient alors plus que 80 variables sur nos 155 de base

Avantages / Inconvénients

Le plus gros avantage de cette stratégie est la facilité et la suppression du biais liés aux données manquantes, seulement, on perds énormément de données et donc d'information avec cette stratégie puisque dès qu'une observation possède une donnée manquante on décide de la supprimer, dans notre cas on supprime quasiment 50% des observations avec cette stratégie.

Comme autres stratégies on pourrait utiliser un indicateur statistique ou une régression linéaire sur nos variables quantitatives.

Prédiction de la variable class

Création du modèle

On réalise un premier modèle qui explique la variable `class` en fonction de toutes les autres variables :

```
m1 <- glm(formula = class~., data=d2, family=binomial(link="logit"))
summary(m1)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial(link = "logit"), data = d2)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.936e-05   2.110e-08   2.110e-08   2.110e-08   2.714e-05
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         8.162e+01  4.556e+05   0.000    1.000
## age                 9.430e-01  2.835e+03   0.000    1.000
## sexmale            -1.566e+02  4.121e+05   0.000    1.000
## steriodyes          8.044e+01  1.243e+05   0.001    0.999
## antiviralsyes       7.408e-01  1.293e+05   0.000    1.000
## fatigueyes          7.624e+00  3.256e+05   0.000    1.000
## malaiseyes         -1.324e+01  2.964e+05   0.000    1.000
## anorexiayes        -6.775e+01  1.752e+05   0.000    1.000
## liver_bigyes       -8.319e+01  1.322e+05  -0.001    0.999
## liver_firmyes       3.665e+01  2.819e+05   0.000    1.000
## spleen_palpableyes  2.501e+01  1.087e+05   0.000    1.000
## spidersyes          1.985e+01  8.394e+04   0.000    1.000
## asitesyes           4.498e+00  2.245e+05   0.000    1.000
## varicesyes          3.119e+01  1.667e+05   0.000    1.000
## bilirubin          -2.760e+01  7.374e+04   0.000    1.000
## alk_phosphate      -3.383e-02  1.146e+03   0.000    1.000
## sgot                8.206e-01  1.354e+03   0.001    1.000
## albumin             1.227e+01  1.603e+05   0.000    1.000
## protime             7.724e-01  1.138e+03   0.001    0.999
## histologyyes       -4.520e+01  1.476e+05   0.000    1.000
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7.1007e+01  on 79  degrees of freedom
## Residual deviance: 6.7558e-09  on 60  degrees of freedom
## AIC: 40
##
## Number of Fisher Scoring iterations: 25
```

Puis nous allons ensuite utiliser une méthode step by step afin d'ajouter/retirer des variables explicative dans le but de minimiser le critère AIC de notre modèle :

```
m2 <- step(m1)
```

Et on affiche le résumé de notre second modèle :

```r
summary(m2)
```

```
##
## Call:
## glm(formula = class ~ sex + steriod + liver_big + liver_firm +
##     bilirubin + sgot + protime + histology, family = binomial(link = "logit"),
##     data = d2)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.551e-04   2.000e-08   2.000e-08   2.000e-08   4.935e-04
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     4098.20  212341.58   0.019    0.985
## sexmale        -3143.98  162480.91  -0.019    0.985
## steriodyes      2805.88  134408.95   0.021    0.983
## liver_bigyes   -2974.72  152717.80  -0.019    0.984
## liver_firmyes   1467.18   70357.00   0.021    0.983
## bilirubin      -1283.83   61732.88  -0.021    0.983
## sgot              34.17    1639.71   0.021    0.983
## protime           14.30     684.76   0.021    0.983
## histologyyes   -1977.67   94857.39  -0.021    0.983
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7.1007e+01  on 79  degrees of freedom
## Residual deviance: 1.3086e-06  on 71  degrees of freedom
## AIC: 18
##
## Number of Fisher Scoring iterations: 25
```

On remarque que la p-value de chacune des variables retenus n'est pas significative, ce qui signifie qu'il n'y a pas de lien entre les variables explicatives et la variable à expliquer, notre modèle n'est pas forcément pertinent

Vérification des performances

Nous allons maintenant vérifier les performances de notre modèle grâce à divers indicateurs (Se, Sp, TBC, Courbe ROC, etc...), on commence donc par créer des fonctions qui vont nous permettre de calculer ces indicateurs :

```r
se <- function(mat) {
  mat[2,2] / (mat[2,2] + mat[2,1])
}

sp <- function(mat) {
  mat[1,1] / (mat[1,1] + mat[1,2])
}

tbc <- function(mat) {
  (mat[1,1] + mat[2,2]) / sum(mat)
}
```

Appliquées à nos résultats, on obtient :

```
mat_conf <- table(
  factor(ifelse(
    predict(m2, d2[-1]) > 0.5,
    1,
    0)),
  d2$class)

paste('Se (Sensibilité) =', se(mat_conf))
```

```
## [1] "Se (Sensibilité) = 1"
```

```
paste('Sp (Spécificité) =', sp(mat_conf))
```

```
## [1] "Sp (Spécificité) = 1"
```

```
paste('TBC (Taux Bien Classé) =', tbc(mat_conf))
```

```
## [1] "TBC (Taux Bien Classé) = 1"
```

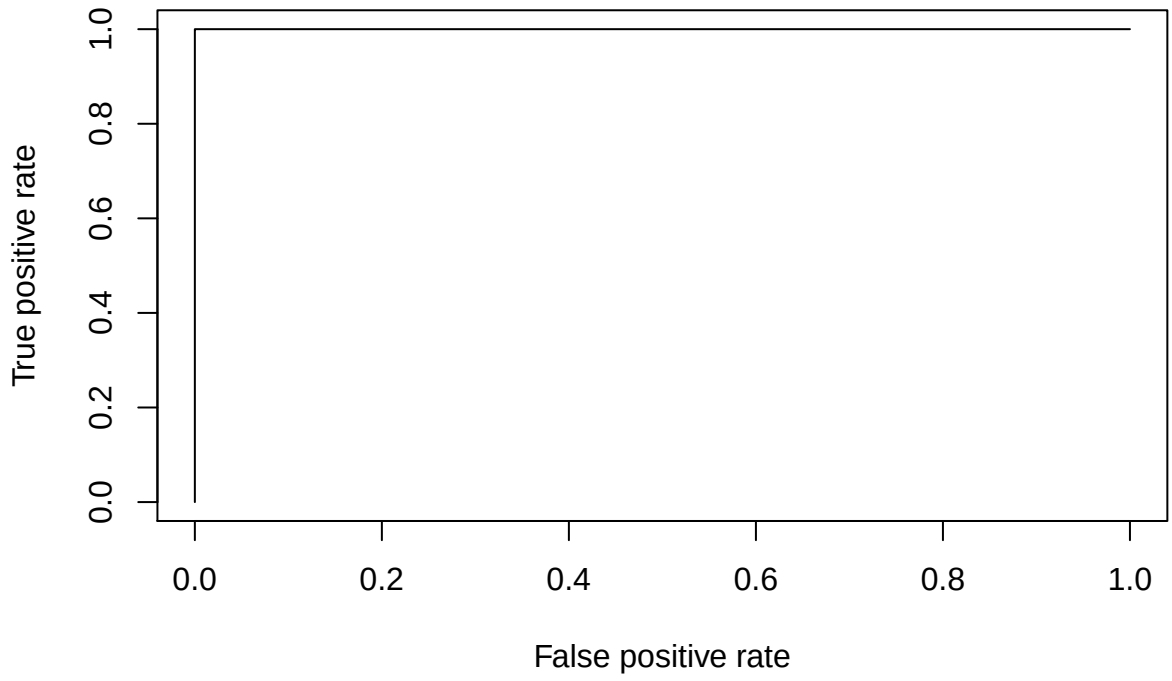
On remarque que tous les indicateurs sont parfaits, on pourrait être dans un cas du surapprentissage des données, nous allons vérifier cela en traçant la courbe ROC, Pour afficher la courbe ROC, on va utiliser la librairie `ROCR` et la fonction suivante :

```
library(ROCR)

roc <- function(model, data) {
  pred <- prediction(model$y, as.integer(data$class))
  perf <- performance(pred, 'tpr', 'fpr')
  return(perf)
}
```

La courbe ROC de notre modèle est la suivante :

```
plot(roc(m2, d2))
```



Notre classification est parfait, c'est à dire que le point (0,1) est atteint, cela s'explique par ce qu'on a vu précédemment, c'est à dire qu'il n'y a pas de corrélation entre les variables explicatives et la variable à expliquer, on peut parler du surapprentissage de notre modèle

Ajustement d'une régression logistique sur le jeu de données augmenté

Le but de cette partie est d'éviter le surapprentissage de notre modèle, pour cela nous devons avoir plus d'individus pour établir notre modèle, on va donc au lieu d'enlever les individus ayant des valeurs manquantes `NA` essayer de prédire ces valeurs manquantes

Rappel du nombre de valeurs manquantes par variables :

```
colSums(is.na(d))
```

```
##           class             age             sex         steriod      antivirals
##               0               0               0               1               0
##         fatigue         malaise        anorexia       liver_big      liver_firm
##               1               1               1              10              11
## spleen_palpable         spiders          asites         varices       bilirubin
##               5               5               5               5               6
##   alk_phosphate            sgot         albumin         protime       histology
##              29               4              16              67               0
```

Modèle de prédiction de la variable `protime`

On va prédire la variable `protime` avec une régression linéaire en utilisant la même méthode que pour la variable `class` :

```
m3 <- lm(formula = protime~., data = d2)
m4 <- step(m3)
```

On obtient donc le modèle suivant :

```
summary(m4)
```

```
##
## Call:
## lm(formula = protime ~ class + antivirals + spiders + bilirubin +
##     albumin, data = d2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -78.393 -10.777  -1.807  11.897  37.767
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.307     18.965   0.069   0.9452
## classlive        9.902      7.158   1.383   0.1707
## antiviralsyes    9.292      5.346   1.738   0.0864 .
## spidersyes       9.655      5.270   1.832   0.0710 .
## bilirubin       -4.851      2.890  -1.678   0.0975 .
## albumin         11.797      4.650   2.537   0.0133 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.97 on 74 degrees of freedom
## Multiple R-squared:  0.3193, Adjusted R-squared:  0.2733
## F-statistic: 6.943 on 5 and 74 DF,  p-value: 2.298e-05
```

Estimation de la variable `protime`

Maintenant que nous avons construit le modèle de prédiction, nous allons estimer les différentes valeurs de cette variable :

```
estimation <- predict(m4, d)
print(estimation)
```

```
##         1         2         3         4         5         6         7
##  72.49431  67.08076  73.94946  64.65795  72.49431  72.97936        NA
##         8         9        10        11        12        13        14
##        NA  78.66834  71.31459  66.46652  50.00872  62.50813  77.29116
##        15        16        17        18        19        20        21
##        NA  54.66688  75.06336  74.43451  75.12918  67.01739  66.04729
##        22        23        24        25        26        27        28
##  63.04900  72.49431  81.23739  69.65227  71.10497  51.94893  45.56283
##        29        30        31        32        33        34        35
##  68.67971  70.28112  44.63532        NA  76.45515  72.56013  76.51852
##        36        37        38        39        40        41        42
##  57.23666  72.21887  55.35400  72.49431  53.25784  64.65795        NA
##        43        44        45        46        47        48        49
##  77.21318  78.66834        NA  78.66834  62.42016  60.35586  75.82385
##        50        51        52        53        54        55        56
##  57.36247  72.97936  72.49431  79.15339  62.23270  76.30890        NA
##        57        58        59        60        61        62        63
##        NA  63.47823  66.74196        NA  79.84806  73.94946  43.63478
##        64        65        66        67        68        69        70
##  63.33199  67.56581  75.82385  62.15227  21.38435  72.49431  66.04484
##        71        72        73        74        75        76        77
##  77.00357        NA        NA  72.49431  85.40540  64.38252  39.82847
##        78        79        80        81        82        83        84
##  65.14301  67.55826  84.56694  82.90216  76.30890  73.94946        NA
##        85        86        87        88        89        90        91
##  60.20451  69.58400        NA  31.50461  37.81028  79.15339  56.24612
##        92        93        94        95        96        97        98
##  27.21712  72.49431  57.09214  45.79288  67.50244  62.62977  47.23976
##        99       100       101       102       103       104       105
##  50.44594        NA  19.16870        NA  53.90639  44.19781  36.78191
##       106       107       108       109       110       111       112
## 102.74778  33.99569        NA  58.48392  52.24990  76.30890  43.71133
##       113       114       115       116       117       118       119
##  72.49431  59.72701  63.75122        NA  73.67403  71.31459        NA
##       120       121       122       123       124       125       126
##  59.46373  43.72104  49.86248        NA  66.86360  62.83939  65.47426
##       127       128       129       130       131       132       133
##  62.83939  34.98550  60.90673  27.21712  78.39290   7.40337  55.62943
##       134       135       136       137       138       139       140
##  56.12448  63.14281  67.28283  69.70809  54.03048  36.42095  64.10708
```

```
##        141        142        143        144        145        146        147
##  44.107675  42.626841  51.042195  45.098214  29.696043  73.883641  30.337049
##        148        149        150        151        152        153        154
##         NA  72.979358  74.434512  12.665575  76.518516  64.989211  61.593849
##        155
##  41.349440
```

Et on peut maintenant remplacer les valeurs manquantes par nos estimations :

```
d$protime[is.na(d$protime)] = estimation[is.na(d$protime)]
```

Et on créer un nouveau jeu de données avec nos estimations :

```
d3 <- na.omit(d)
summary(d3)
```

```
##   class         age           sex     steriod  antivirals fatigue  malaise
##  die :19   Min.   : 7.0   female:13   no :54   no :22     no :73   no :42
##  live:93   1st Qu.:32.0   male  :99   yes:58   yes:90     yes:39   yes:70
##            Median :39.0
##            Mean   :41.2
##            3rd Qu.:50.0
##            Max.   :78.0
##  anorexia liver_big liver_firm spleen_palpable spiders  asites   varices
##  no :19   no :22    no :54     no :20          no :38   no :14   no :14
##  yes:93   yes:90    yes:58     yes:92          yes:74   yes:98   yes:98
##
##
##
##
##    bilirubin     alk_phosphate        sgot           albumin
##  Min.   :0.300   Min.   : 26.0   Min.   : 14.00   Min.   :2.100
##  1st Qu.:0.700   1st Qu.: 72.0   1st Qu.: 30.00   1st Qu.:3.500
##  Median :1.000   Median : 85.0   Median : 56.50   Median :4.000
##  Mean   :1.272   Mean   :105.5   Mean   : 78.62   Mean   :3.835
##  3rd Qu.:1.400   3rd Qu.:133.5   3rd Qu.: 98.00   3rd Qu.:4.200
##  Max.   :4.800   Max.   :295.0   Max.   :420.00   Max.   :5.300
##     protime       histology
##  Min.   :  0.00   no :65
##  1st Qu.: 46.75   yes:47
##  Median : 62.71
##  Mean   : 62.44
##  3rd Qu.: 75.30
##  Max.   :100.00
```

On décide de regarder si nous avons plus d'individus que notre jeu de données `d2` :

```
dim(d3)
```

```
## [1] 112  20
```

Nouveau modèle de prédiction de la variable `class`

Nous allons construire un nouveau modèle de prédiction de la variable `class` à partir de notre jeu de données `d3`, on va utiliser la même méthode que vu précédemment :

```
m5 <- glm(formula = class~., data = d3, family = binomial(link = 'logit'))
m6 <- step(m5)
```

On obtient le modèle suivant :

```
summary(m6)
```

```
##
## Call:
## glm(formula = class ~ sex + anorexia + spiders + asites + bilirubin +
##     protime, family = binomial(link = "logit"), data = d3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2106   0.0001   0.1757   0.3383   1.7369
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   17.01397 2473.77703   0.007   0.9945
## sexmale      -18.05106 2473.77665  -0.007   0.9942
## anorexiayes   -2.55273    1.09869  -2.323   0.0202 *
## spidersyes     1.97717    0.78093   2.532   0.0113 *
## asitesyes      2.02069    0.84777   2.384   0.0171 *
## bilirubin     -0.64078    0.41220  -1.555   0.1201
## protime        0.05552    0.02076   2.674   0.0075 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 101.992  on 111  degrees of freedom
## Residual deviance:  52.083  on 105  degrees of freedom
## AIC: 66.083
##
## Number of Fisher Scoring iterations: 18
```

On va maintenant vérifier les perfomances de notre nouveau modèle :

```
mat_conf <- table(
  factor(ifelse(
    predict(m6, d3[-1]) > 0.5,
    1,
    0)),
  d3$class)

paste('Se (Sensibilité) =', se(mat_conf))
```

```
## [1] "Se (Sensibilité) = 0.945652173913043"
```

```
paste('Sp (Spécificité) =', sp(mat_conf))
```

```
## [1] "Sp (Spécificité) = 0.7"
```

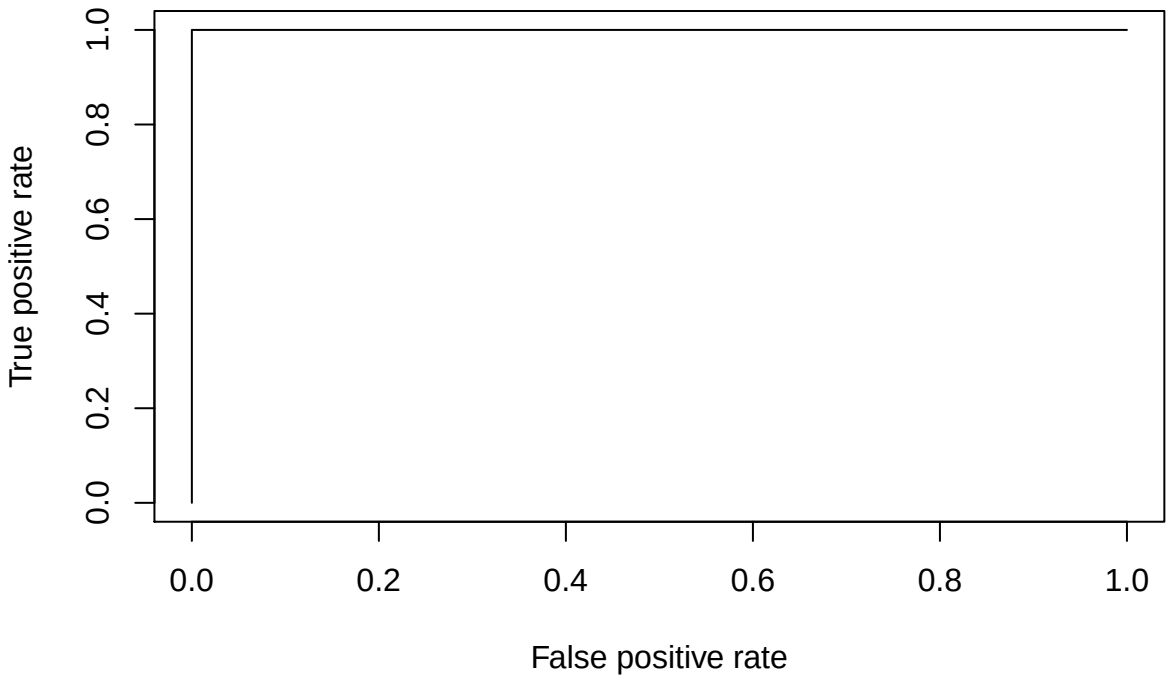
```
paste('TBC (Taux Bien Classé) =', tbc(mat_conf))
```

```
## [1] "TBC (Taux Bien Classé) = 0.901785714285714"
```

On obtient des valeurs qui sont plus cohérentes que précédemment, les indicateurs ne sont plus parfaits mais

restent performants, Et on décided'afficher la courbe ROC associée :

```
plot(roc(m6, d3))
```



On obtient toujours une courbe ROC parfaite, or cela ne semble pas cohérent étant donné que nos indicateurs Se, Sp et TBC ne sont plus parfaits