Régression Logistique simple et multiple

S. Jaubert

03 mai 2020

Il est très courant de chercher une relation entre une variable logique Y (0/1, Infecté/Non Infecté, client content/client Pas content, etc...) et une ou plusieurs variables explicatives (catégorielles ou continues).

Y n'ayant que deux états l'idée est de modéliser la probabilité de réalisation de l'un de ses états.

Préliminaire : Notion de cote (ou odd en anglais)

Un exemple : On lance deux dés cubiques, on gagne si on obtient un double 6

$$Y = \begin{cases} 1 & "gagne" \\ 0 & "perd" \end{cases}$$

Y est une variable de Bernouilli

$$\begin{cases} \mathbb{P}(Y=1) & = \pi \\ \mathbb{P}(Y=0) & = 1-\pi \end{cases} \quad ou\ \mathbb{P}(Y=k) = \pi^k(1-\pi)^{1-k}$$

(ici $\pi = \frac{1}{36}$)

La cote du succès est $odd(\pi) = \frac{\pi}{1-\pi} = \frac{\frac{1}{36}}{\frac{35}{36}} = \frac{1}{35}$ on dit aussi 1 contre 35 (1 évènement favorable contre 35 défavorables)

A présent introduisons un autre évènement

$$X = \begin{cases} 1 & "dé\ pipé" \\ 0 & "dé\ non\ pipé" \end{cases}$$

On aura les probabilités conditionnelles suivantes :

$$\begin{cases} \mathbb{P}(Y=1|X=0) & = \pi \\ \mathbb{P}(Y=1|X=1) & = \pi' \end{cases}$$

Désiquilibrons les dés pour augmenter nos chances de gagner ($\pi' = \frac{11}{36}$ par exemple), alors la cote du succès sera $odd(\pi') = \frac{\pi'}{1-\pi'} = \frac{\frac{11}{36}}{\frac{25}{36}} = \frac{11}{25}$ (évidemment supérieur à $odd(\pi)$)

On définit à présent le rapport des chances par l'Odds-Ratio de Y associé à X :

$$OR(\pi, \pi') = \frac{odd(\pi)}{odd(\pi')} = \frac{\frac{\pi}{1-\pi}}{\frac{\pi'}{1-\pi'}}$$

Avec notre exemple, $OR(\pi, \pi') \simeq 0.06$ cela signifie que si on joue avec les dés pipés on aura 16 fois plus de chances de gagner que si on joue avec les dés équilibrés.

On a les relations suivantes :

$$\begin{cases} OR(\pi\,\pi') > 1 \Leftrightarrow & \pi > \pi' \\ OR(\pi, \pi') = 1 \Leftrightarrow & \pi = \pi' \\ OR(\pi, \pi') < 1 \Leftrightarrow & \pi < \pi' \end{cases}$$

Si π et π' sont petits (par rapport à 1) on pourra simplifier $OR(\pi, \pi') \simeq \frac{\pi}{\pi'}$ qui est nettement plus simple à interpréter.

Généralités sur la Regression Logistique

Reprenons notre variable binaire $Y = \begin{cases} 1 \\ 0 \end{cases}$ expliquée par $X = (X_1, X_2, ..., X_p)$

On cherche à modéliser $\mathbb{P}(Y = 1|x) = \pi(x)$ (π dépend des variables explicatives)

$\pi(x)$ est aussi la valeur moyenne de Y quand $x = (x_1, x_2, ..., x_p)$

$$\mathbb{E}(Y = 1|x) = \pi(x)$$

évidemment $0 < \pi(x) < 1$ et nous ne pouvons l'exprimer comme une combinaison linéaire... On va transformer $\pi(x)$ à travers une fonction dite de lien g.

$$\begin{aligned} g \quad &: [0, 1] \to \mathbb{R} \\ &\pi \mapsto g(\pi) \end{aligned}$$

Plusieurs fonctions de lien peuvent être utilisées pour la régression d'une variable binaire ; nous utiliserons le *logit* défini par : $\pi \mapsto g(\pi) = log(\frac{\pi}{1-\pi})$ on reconnait le $log(odd(\pi))$ c'est la raison pour laquelle il est courant d'utiliser le *logit* car son interprétation comme log de odd est commode (log est ici le logarithme de base e).

Exprimons $g(\pi(x))$ comme une combinaison linéaire des $X_1, X_2, ..., X_p$

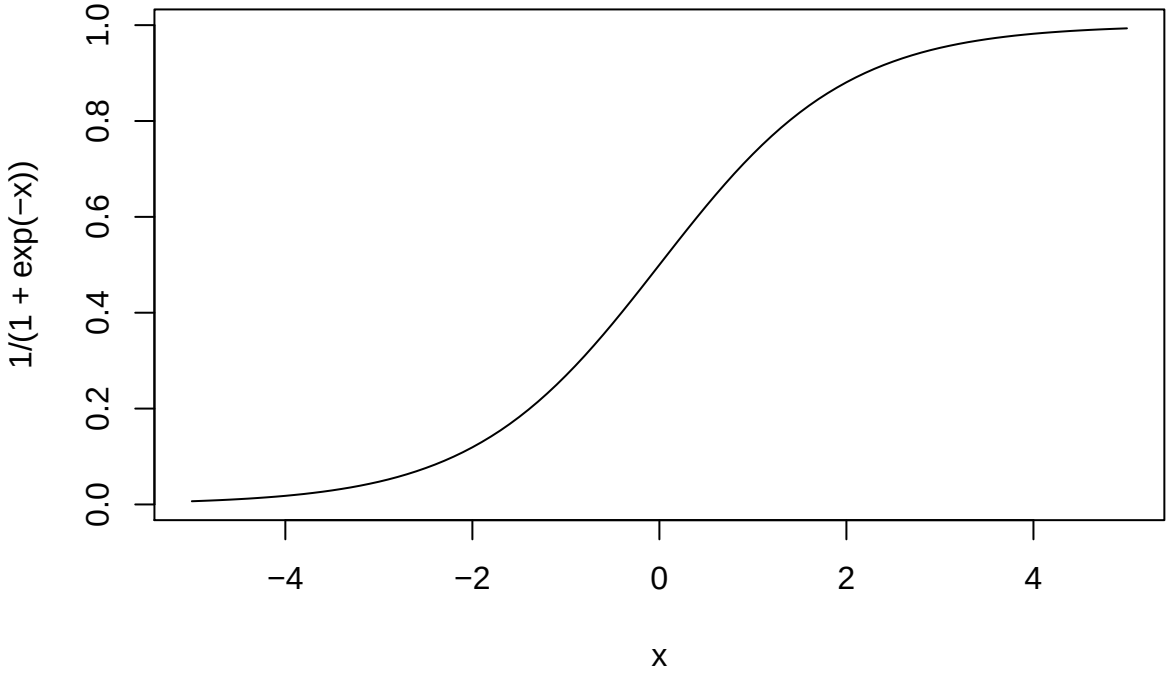
La fonction *logit* nous permet de mettre en relation la probabilité de réalisation (bornée entre 0 et 1), et une combinaison linéaire des variables explicatives.

$$g(\pi(x)) = \eta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p = \beta_0 + \sum_{k=1}^{p} \beta_k x_k$$

$$\pi(x) = \frac{e^{\eta(x)}}{1 + e^{\eta(x)}} = \frac{1}{1 + e^{-\eta(x)}}$$

La probabilité $\mathbb{P}(Y = 1|x) = \pi(x)$, est alors modélisée par une courbe, appelée sigmoïde, bornée par 0 et 1 :

```
curve(expr=1/(1+exp(-x)), from=-5, to= 5)
```



Cette fonction part d'un état des prédicteurs et renvoie une probabilité.

Essayons de comprendre comment interpréter β_1 ; pour simplifer prenons une seule variable explicative.

Nous avons alors $g(\pi(x)) = \eta(x) = \beta_0 + \beta_1 x$

et donc $log(odd(\pi(x)) = \beta_0 + \beta_1 x \Leftrightarrow odd(\pi(x)) = e^{\beta_0+\beta_1 x}$

Augmentons x d'une unité : $odd(\pi(x+1)) = e^{\beta_0+\beta_1(x+1)} = e^{\beta_1} e^{\beta_0+\beta_1 x}$

Donc si x augmente d'une unité les odds seront mulitpliés par e^{β_1} qui est l'odds-Ratio : $OR(\pi(x+1), \pi(x)) = \frac{odd(\pi(x+1))}{odd(\pi(x))} = e^{\beta_1}$

En généralisant, e^{β_j} est l'odds-ration de l'évènement "$Y = 1$" influencé par la variable X_j

Maximum de vraisemblance

Pour les modèles linéaires nous avions utilisé la méthode des moindres carrés pour l'estimation des coefficients $\beta = (\beta_0, \beta_1, ..., \beta_p)^T$, en régression logisitique nous utiliserons la méthode dite du maximum de vraisemblance.

Pour n observations $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,p})$ $(i = 1, 2, ..., n)$, on cherchera à maximiser le produit :

$$\mathrm{L}(\beta) = \prod_{i=1}^{n} \mathbb{P}(Y = y_i | X = x_i)$$

On a vu $\mathbb{P}(Y = y_i | x = x_i) = \pi(x_i)^{y_i}(1 - \pi(x_i))^{1-y_i}$ $(y_i = 1, 0)$, on peut donc écrire le log du maximum de vraisemblance (car plus simple à dériver) :

$$\mathrm{LL}(\beta) = log(\prod_{i=1}^{n} \pi(x_i)^{y_i}(1-\pi(x_i))^{1-y_i})$$

$$\mathrm{LL}(\beta) = \sum_{i=1}^{n} log(\pi(x_i)^{y_i}(1-\pi(x_i))^{1-y_i})$$

et on obtient :

$$\mathrm{LL}(\beta) = \sum_{i=1}^{n} y_i log(\pi(x_i)) + (1-y_i) log(1-\pi(x_i))$$

$$\mathrm{LL}(\beta) = \sum_{i=1}^{n} y_i log(\frac{\pi(x_i)}{1-\pi(x_i)}) + log(1-\pi(x_i))$$

On se rappelle que

$$g(\pi(x_i)) = log(\frac{\pi(x_i)}{1-\pi(x_i)}) = x_i\beta \Leftrightarrow \pi(x_i) = \frac{1}{1+e^{-x_i\beta}}$$

d'où :

$$\mathrm{LL}(\beta) = \sum_{i=1}^{n} y_i x_i \beta - log(1+e^{x_i\beta})$$

Maximiser le log de vraisemblance en fonction de β analytiquement n'est pas possible, il faut passer par des méthodes numériques (descente du gradient, hypothèse de convéxité, algorithme de Newton-Raphson...) est dépasse largment mon propos. Voir à ce sujet quelques références :

- Albert, A. and Anderson, D. (1984). On the existence of maximum likelihood estimates in logistic regression models. Biometrika, 71 :1–10
- Fahrmeir, L. and Kaufmann, Consistency and asymptotic normality of the maximum likelihood estimator in generalized linear models.The Annals of Statistics (1985), 13 :342–368.
- Saporta Gilbert, Probabilités et statistique analyse des données Edition Technip 2006
- Peter K. Dunn and Gordon K. Smyth, Generalized Linear Models With Examples in R Springer 2018

Exemples avec R

Cas simple avec un seul régresseur

Nous avons le résultat d'une étude sur le diabète de type 2. Dans quelle mesure la présence d'un facteur de stress peut influencer la maladie.

$$Y = \begin{cases} 1 & malade \\ 0 & Pas\ malade \end{cases} \quad X = \begin{cases} 1 & Stress \\ 0 & no\ stress \end{cases}$$

Voici la matrice des résultats :

```
diab<-matrix(c(240,115,80,130),byrow = F,nrow = 2)
colnames(diab)<-c("No_diab","Diab")
rownames(diab)<-c("No_stress","Stress")
(Tdiab<-as.table(diab))
```

```
##           No_diab Diab
## No_stress     240   80
## Stress        115  130
```

```
#créons un data frame
(df_diab<-as.data.frame(Tdiab))
```

```
##        Var1    Var2 Freq
## 1 No_stress No_diab  240
## 2    Stress No_diab  115
## 3 No_stress    Diab   80
## 4    Stress    Diab  130
```

(2 états par variable d'où 4 lignes)

Construisons notre modèle logisitique à présent :

```
Logmodel<-glm(data = df_diab,formula = Var2~Var1,family = binomial(logit),weights = Freq)
```

Quelques explications des arguments employés :

- On impose la fonction de lien par **family=binomial(logit)** avec **R** nous en avons bien d'autres :

binomial(link = "logit")

gaussian(link = "identity")

Gamma(link = "inverse")

inverse.gaussian(link = "1/mu^2")

poisson(link = "log")

quasi(link = "identity", variance = "constant")

quasibinomial(link = "logit")

quasipoisson(link = "log")

- **formula** explique que nous voulons créer un modèle qui explique Var2 en fonction de Var1
- **weights** est utilisé pour spécifier les fréquences de chaque cas
- **data** renseigne où chercher les données

Les résultats sont donnés par :

```
summary(Logmodel)
```

```
##
## Call:
## glm(formula = Var2 ~ Var1, family = binomial(logit), data = df_diab,
##     weights = Freq)
##
## Deviance Residuals:
##      1       2       3       4
```

```
## -11.75  -13.19   14.89   12.84
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.0986     0.1291  -8.510  < 2e-16 ***
## Var1Stress    1.2212     0.1818   6.717 1.85e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 745.62  on 3  degrees of freedom
## Residual deviance: 698.62  on 2  degrees of freedom
## AIC: 702.62
##
## Number of Fisher Scoring iterations: 5
```

Analysons ces résultats :

- Nous retrouvons en premier le modèle utilisé et les options
- Le paragraphe *Deviance Residuals* donne une mesure de l'ajustement du modèle.
- La partie suivante des coefficients est essentielle. Nous voyons leurs erreurs standards, la statistique z et les p-value associées (ici statistiquement significatives avec un niveau inféreur à 0.1%). Sur la ligne Var1Stress nous lisons **1.2212** ce n'est rien d'autre que le $log(OR(\pi(x=1), \pi(x=0)))$

Pour voir l'intervallle de confiance des paramètres (au risque de 5%) il suffit de faire :

```
confint(Logmodel)
```

```
##                 2.5 %     97.5 %
## (Intercept) -1.357360 -0.8505665
## Var1Stress   0.867613  1.5808489
```

L'équation de notre modèle s'écrit :

$$\pi(x) = \frac{e^{-1.0986+1.2212x}}{1+e^{-1.0986+1.2212x}}$$

x peut-être égale à 1 ou 0 (Stress/No_Stress)

On remarque que $\pi(x=0) = \frac{e^{-1.0986}}{1+e^{-1.0986}} \simeq 0.25$ (la probabilité d'être atteint de diabète de type 2 en cas d'évènement non stressant serait de 25%) et $\pi(x=1) = \frac{e^{-1.0986+1.2212}}{1+e^{-1.0986+1.2212}} \simeq 0.53$ (probabilité d'être atteint de diabète de type 2 en cas d'évènement stressant)

Nous pouvons calculer l'$OR(\pi(x=1), \pi(x=0)) = \frac{odd(\pi(x=1))}{odd(\pi(x=0))} = \frac{\frac{\pi(x=1)}{1-\pi(x=1)}}{\frac{\pi(x=0)}{1-\pi(x=0)}} \simeq 3.39$ (on retrouve bien $log(3.39) \simeq 1.22$) une personne qui a vécu un évènement stressant a donc 3.39 fois plus "de chances" de développer un diabète de type 2 par rapport à une personne qui n'aurait pas vécu d'évènement stressant.

La p-value associée à Var1Stress est très faible, il y a donc un lien significatif entre la présence de stress et la survenue du diabète de type 2.

Régression multiple

Nous allons nous intéresser à un échantillon de vin rouge du nord du Portugal. L'objectif est de modéliser la qualité du vin sur la base de tests physico-chimiques. (les données sont accessibles ici http://archive.ics.uci.edu/ml/datasets/Wine+Quality)

Chargeons les données :

```
qw<-read.csv2(file = "https://sjaubert.github.io/regression/winequality-red.csv",header = T,sep = ";")
```

La structure de nos données est indiquée par :

```
str(qw)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : Factor w/ 96 levels "10","10.1","10.2",..: 71 75 75 13 71 71 76 70 75 72 ..
##  $ volatile.acidity    : Factor w/ 143 levels "0.12","0.16",..: 77 113 89 13 77 69 57 67 53 42 ...
##  $ citric.acid         : Factor w/ 80 levels "0","0.01","0.02",..: 1 1 5 57 1 1 7 1 3 37 ...
##  $ residual.sugar      : Factor w/ 91 levels "0.9","1.2","1.3",..: 11 31 26 11 11 10 6 2 20 73 ...
##  $ chlorides           : Factor w/ 153 levels "0.012","0.034",..: 40 62 56 39 40 39 33 29 37 35 ...
##  $ free.sulfur.dioxide : Factor w/ 60 levels "1","10","11",..: 3 18 7 9 3 5 7 7 60 9 ...
##  $ total.sulfur.dioxide: Factor w/ 144 levels "10","100","101",..: 75 109 95 102 75 81 100 60 57 4 .
##  $ density             : Factor w/ 436 levels "0.99007","0.9902",..: 343 272 288 355 343 343 240 101
##  $ pH                  : Factor w/ 89 levels "2.74","2.86",..: 64 33 39 29 64 64 43 52 49 48 ...
##  $ sulphates           : Factor w/ 96 levels "0.33","0.37",..: 19 31 28 21 19 19 9 10 20 43 ...
##  $ alcohol             : Factor w/ 65 levels "10","10.0333333333333",..: 57 63 63 63 57 57 57 1 58 7
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

La variable **quality** est basée sur des données sensorielles score entre 0 et 10 Il y a beaucoup plus de vins moyens que d'excellents aussi nous recoderons cette variable en une variable binaire 1 si la qualité a eu une note > 6, 0 sinon.

Chargeons quelques packages de **R** bien pratiques pour manipuler les données

```
library(tibble)
library(tidyr)
library(dplyr)
```

On redéfinit **quality** simplement par :

```
new_qw<-qw %>% mutate(quality = ifelse(quality > 6,1,0)) %>%
    mutate(quality =as.integer(quality ))
```

Pour commencer donnons la proportion de vins d'excellentes qualité :

```
table(new_qw$quality)/nrow(new_qw)
```

```
##
##         0         1
## 0.8642902 0.1357098
```

Soit 13.57%

Ce résultat peut-être retrouvé en faisant une régression logistique sur aucune variable en utilisant le code :

```r
glm(data = new_qw,formula = quality~1,family = binomial(logit))->mod0
summary(mod0)
```

```
##
## Call:
## glm(formula = quality ~ 1, family = binomial(logit), data = new_qw)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5401  -0.5401  -0.5401  -0.5401   1.9986
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.85139    0.07302  -25.36   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1269.9  on 1598  degrees of freedom
## Residual deviance: 1269.9  on 1598  degrees of freedom
## AIC: 1271.9
##
## Number of Fisher Scoring iterations: 4
```

Nous obtenons la probabilité $\mathbb{P}(Y=1) = \pi(X=0) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$ avec $\beta_0 = -1.85139$

Soit :

```r
exp(-1.85139)/(1+exp(-1.85139))
```

```
## [1] 0.1357098
```

Ce qui correspond bien à ce que nous avions trouvé.

Maintenant calculant un modèle sur l'ensemble des variables :

```r
mod1<-glm(data = new_qw,formula = quality~.,family = binomial(logit))
```

(le $\sim .$ signifie que l'on considère toutes les variables)

Examinons le modèle :

```r
summary(mod1)
```

```
##
## Call:
## glm(formula = quality ~ ., family = binomial(logit), data = new_qw)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7713  -0.4748  -0.2520  -0.1328   3.0309
```

```
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.1750857  0.6717167  -0.261 0.794359
## fixed.acidity        -0.0082524  0.0033263  -2.481 0.013102 *
## volatile.acidity     -0.0162942  0.0049060  -3.321 0.000896 ***
## citric.acid           0.0278723  0.0070039   3.980 6.91e-05 ***
## residual.sugar        0.0291040  0.0053027   5.489 4.05e-08 ***
## chlorides            -0.0136072  0.0042748  -3.183 0.001457 **
## free.sulfur.dioxide  -0.0004143  0.0044910  -0.092 0.926493
## total.sulfur.dioxide -0.0074196  0.0027348  -2.713 0.006667 **
## density              -0.0081932  0.0009787  -8.372  < 2e-16 ***
## pH                   -0.0006516  0.0072259  -0.090 0.928153
## sulphates             0.0414934  0.0054856   7.564 3.91e-14 ***
## alcohol              -0.0168385  0.0049650  -3.391 0.000695 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1269.92  on 1598  degrees of freedom
## Residual deviance:  929.36  on 1587  degrees of freedom
## AIC: 953.36
##
## Number of Fisher Scoring iterations: 6
```

Deux variables *free.sulfur.dioxide* et *pH* ne sont pas significatives, retirons les et recommençons :

```
mod2<-glm(data = new_qw,formula = quality~. -free.sulfur.dioxide -pH,family = binomial(logit))
summary(mod2)
```

```
##
## Call:
## glm(formula = quality ~ . - free.sulfur.dioxide - pH, family = binomial(logit),
##     data = new_qw)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7743  -0.4753  -0.2525  -0.1336   3.0336
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.2313465  0.4771227  -0.485 0.627763
## fixed.acidity        -0.0082393  0.0033036  -2.494 0.012631 *
## volatile.acidity     -0.0162968  0.0049031  -3.324 0.000888 ***
## citric.acid           0.0280581  0.0063668   4.407 1.05e-05 ***
## residual.sugar        0.0290843  0.0052573   5.532 3.16e-08 ***
## chlorides            -0.0135652  0.0042468  -3.194 0.001402 **
## total.sulfur.dioxide -0.0073760  0.0025872  -2.851 0.004358 **
## density              -0.0081824  0.0009656  -8.474  < 2e-16 ***
## sulphates             0.0415458  0.0054697   7.596 3.06e-14 ***
## alcohol              -0.0167697  0.0049317  -3.400 0.000673 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1269.92  on 1598  degrees of freedom
## Residual deviance:  929.38  on 1589  degrees of freedom
## AIC: 949.38
##
## Number of Fisher Scoring iterations: 6
```

Comme nous l'avions vu les coefficients du modèle représentent les log des oods-ratio, en utilisant la librairie **questionR** nous pouvons directement avoir les odds-ratio, leurs intervalles de confiance ainsi que leurs p-value :

```
library(questionr)
odds.ratio(mod2)
```

```
##                          OR   2.5 % 97.5 %         p
## (Intercept)          0.79346 0.31294 2.0364 0.6277626
## fixed.acidity        0.99179 0.98537 0.9982 0.0126311 *
## volatile.acidity     0.98384 0.97407 0.9930 0.0008880 ***
## citric.acid          1.02846 1.01569 1.0414 1.049e-05 ***
## residual.sugar       1.02951 1.01898 1.0402 3.162e-08 ***
## chlorides            0.98653 0.97811 0.9946 0.0014020 **
## total.sulfur.dioxide 0.99265 0.98760 0.9977 0.0043581 **
## density              0.99185 0.98994 0.9937 < 2.2e-16 ***
## sulphates            1.04242 1.03134 1.0537 3.063e-14 ***
## alcohol              0.98337 0.97369 0.9927 0.0006729 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On peut obtenir un tableau plus propre avec :

```
library(gtsummary)
tbl_regression(mod2, exponentiate = TRUE)
```

Characteristic	**OR**	**95% CI**	**p-value**
fixed.acidity	0.99	0.99, 1.00	0.013
volatile.acidity	0.98	0.97, 0.99	<0.001
citric.acid	1.03	1.02, 1.04	<0.001
residual.sugar	1.03	1.02, 1.04	<0.001
chlorides	0.99	0.98, 0.99	0.001
total.sulfur.dioxide	0.99	0.99, 1.00	0.004
density	0.99	0.99, 0.99	<0.001
sulphates	1.04	1.03, 1.05	<0.001
alcohol	0.98	0.97, 0.99	<0.001

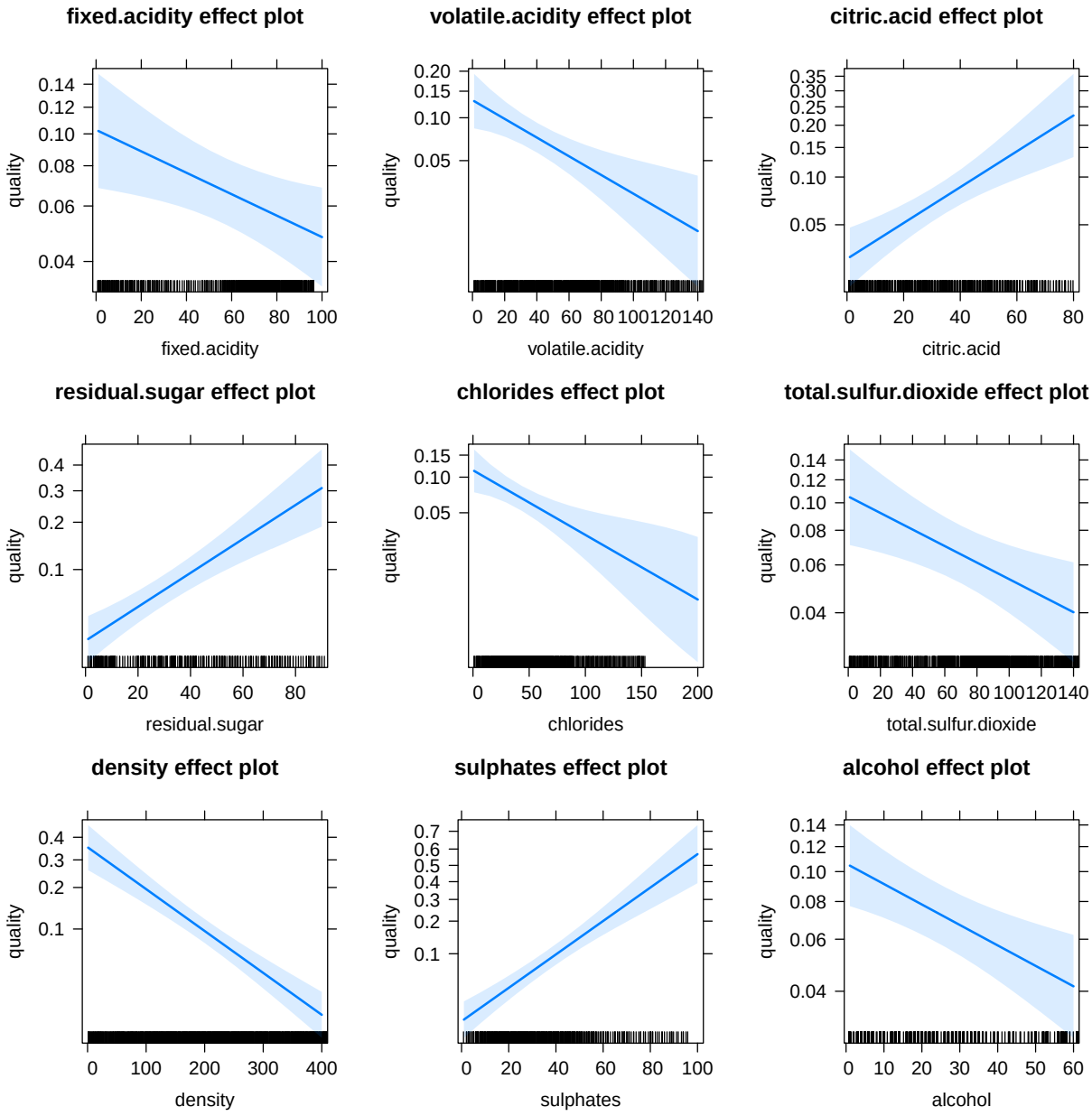
Pour une représentation visuelle résumant les effets de chaque variable du modèle nous utilisons le package **effects** et les instructions suivantes :

```
library(effects)
```

```
## Loading required package: carData
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
par(mar = c(0,0,0,0))
plot(allEffects(mod2))
```



Nous voyons tout de suite les effets sur la “qualité” des variables sulphates, citric.acid et residual.sugar. Je me garderai bien de tout commentaire et je laisse les ardents défenseurs du vin naturel argumenter :-)