# reglogistic.R

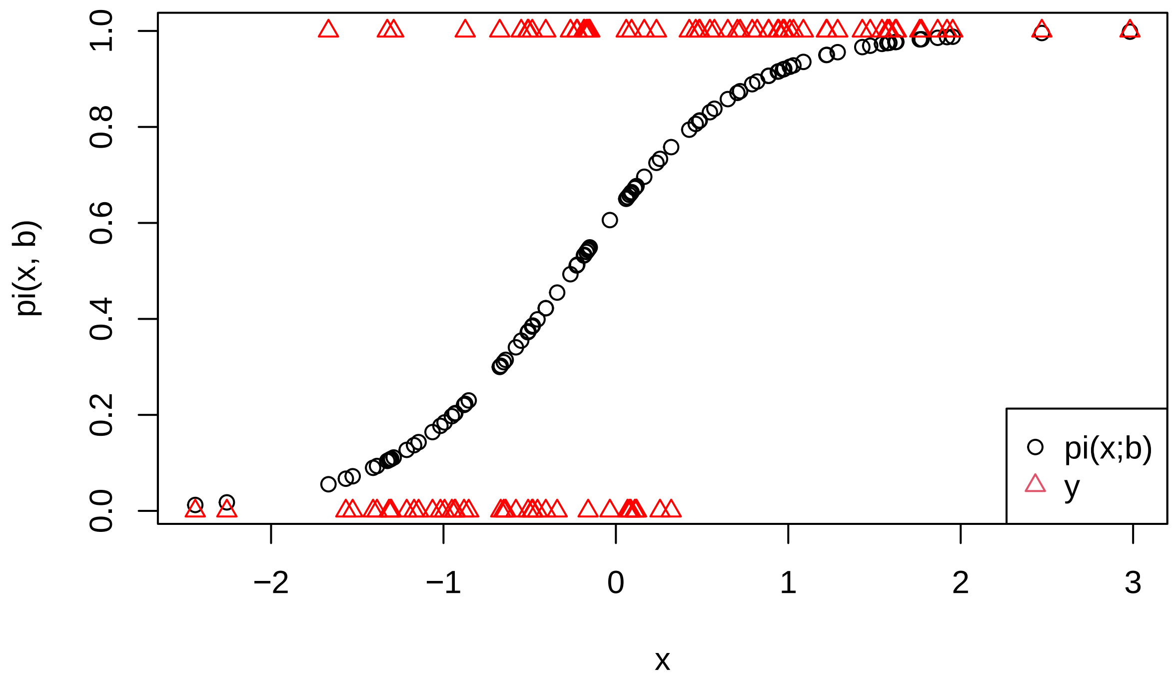
vincentvandewalle

2023-12-12

```r
# Logistic regression
# Newton Raphson algorithm + different test statistics

# Compute pi(x,b)
# x : the design matrix includes a column of 1
pi <- function(x,b){
  p = exp(x %*% b)/(1 + exp(x %*% b))
  p
}

# Basic test on a data point
x = matrix(c(1,2,2),1,3)
b = matrix(c(0.5,2,2),3,1)
pi(x,b)
```

```
##           [,1]
## [1,] 0.9997966
```

```r
# Simple test on simulated data
x = matrix(rnorm(100),100,1)
x = cbind(1,x)
x = as.matrix(x)
b = matrix(c(0.5,2),2,1)
plot(x[,2],pi(x,b), xlab = "x") # Fonctionne en multi-individus
y = rbinom(100,1,pi(x,b))
points(x[,2],y,col = "red", pch=2)
legend("bottomright",legend = c("pi(x;b)","y"), col = 1:2, pch = c(1,2))
```



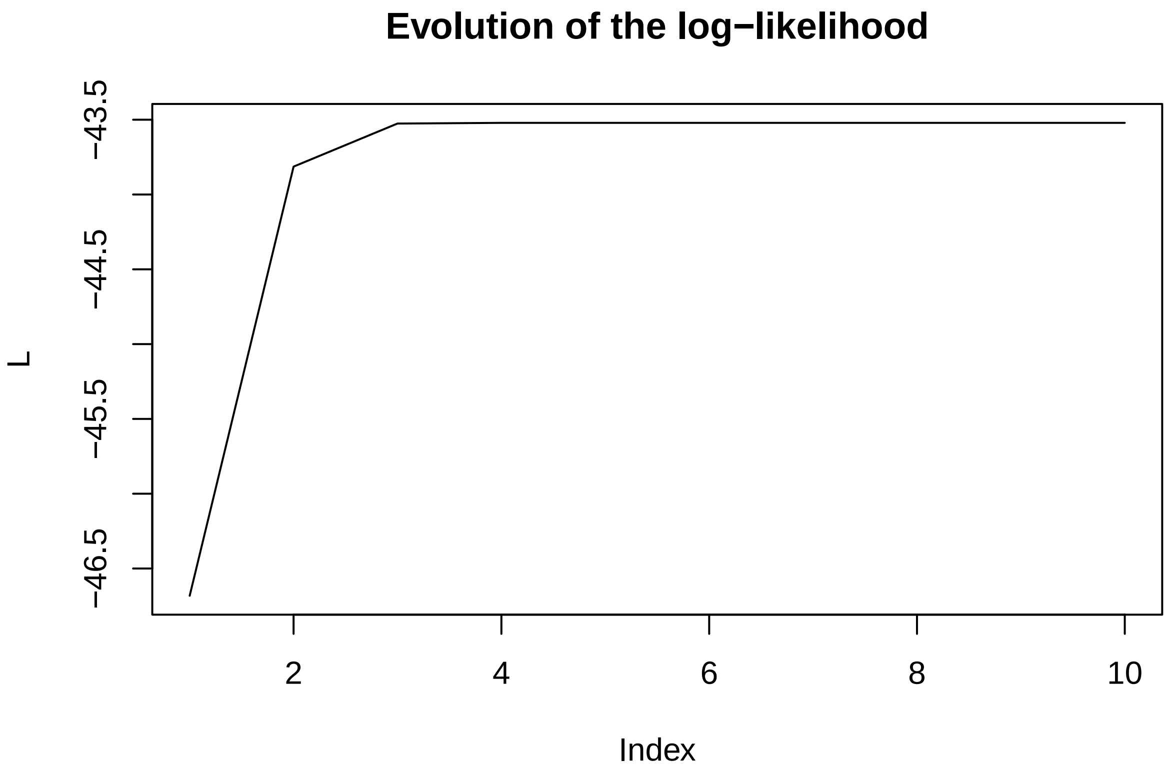

```
# Number of columns :
d = ncol(x)
# Initialisation of b with 0 :
b = matrix(0,d,1)
# L vector of log-likelihood :
L = rep(0,10)

# Newton Raphson algorithm
for (i in 1:10){ # Nb iteration fixed by advance (to improve by setting a given tolerance)
  # Computation of the gradient
  Gradient = t(x) %*% matrix(y - pi(x,b),ncol = 1)
  # Computation of Vtilde : matrix with pi(x)*(1-pi(x)) on the diagonal
  Vtilde = diag(as.vector(pi(x,b)*(1-pi(x,b))))
  # Computation of the Hessian matrix
  Hessian = -t(x) %*% Vtilde %*% x
  # Calcul computation of the variance covairance matrix
  Variance = solve(-Hessian)
  # Updating b
  b = b - solve(Hessian) %*% Gradient
  # Log-likelihood
  L[i] = sum(y*log(pi(x,b)) + (1-y)*log(1-pi(x,b)))
}
L
```

```
##  [1] -46.68219 -43.81362 -43.52578 -43.52114 -43.52114 -43.52114 -43.52114
##  [8] -43.52114 -43.52114 -43.52114
```

```
diff(L)
```

```
## [1] 2.868570e+00 2.878396e-01 4.636563e-03 1.509366e-06 1.634248e-13
## [6] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

```r
plot(L,main = "Evolution of the log-likelihood",type = "l")
```




```r
b # Estimated value
```

```
##           [,1]
## [1,] 0.6805818
## [2,] 1.9685282
```

```r
Variance # Estimated variance
```

```
##            [,1]       [,2]
## [1,] 0.08182871 0.04289406
## [2,] 0.04289406 0.16191173
```

```r
sqrt(diag(Variance)) # Standard value of the coefficient of each variable
```

```
## [1] 0.2860572 0.4023826
```

```r
Wald = b^2/diag(Variance) # Wald statitistic for each variable
Wald
```

```
##           [,1]
## [1,]  5.660502
## [2,] 23.933431
```

```r
# Or equivalently b/(standard deviation)
b/sqrt(diag(Variance))
```

```
##          [,1]
## [1,] 2.379181
## [2,] 4.892181
```

```r
# Computation of p-values for each coefficient with a chi-square distribution
pchisq(Wald,1,lower.tail = FALSE)
```

```
##               [,1]
## [1,] 1.735116e-02
## [2,] 9.972489e-07
```

```
# Computation of AIC criterion
AIC = -2*L[10] + 2*d
AIC
```

```
## [1] 91.04229
```

```
# Comparison of the result with the dedicated R function
don = cbind.data.frame(y=y,x=x[,-1])
res.glm = glm(y ~ x, family = "binomial", data = don)
res.glm
```

```
##
## Call:  glm(formula = y ~ x, family = "binomial", data = don)
##
## Coefficients:
## (Intercept)            x
##      0.6806       1.9685
##
## Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
## Null Deviance:       134.6
## Residual Deviance: 87.04     AIC: 91.04
```

```
temp = summary(res.glm)
temp
```

```
##
## Call:
## glm(formula = y ~ x, family = "binomial", data = don)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7609  -0.7084   0.2112   0.6006   2.3120
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.6806     0.2861   2.379   0.0173 *
## x             1.9685     0.4024   4.892 9.96e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 134.602  on 99  degrees of freedom
## Residual deviance:  87.042  on 98  degrees of freedom
## AIC: 91.042
##
## Number of Fisher Scoring iterations: 5
```

```
temp$cov.unscaled
```

```
##             (Intercept)          x
## (Intercept)  0.08182562 0.04288983
## x            0.04288983 0.16190099
```

```r
# Conclusion: It's ok!

### Computation of likelihood ratio test and score test

# Newton-Raphson algorithm
Newton <- function(x,y){
  d = ncol(x)
  b = matrix(0,d,1)
  L = rep(0,10) # Value of 10 arbitrarly fixed
  for (i in 1:10){
    # Gradient
    Gradient = t(x) %*% matrix(y - pi(x,b),ncol = 1)
    # Vtilde
    Vtilde = diag(as.vector(pi(x,b)*(1-pi(x,b))))
    # Hessian matrix
    Hessian = -t(x) %*% Vtilde %*% x
    # Covariance matrix
    Variance = solve(- Hessian)
    # Calcul de l'intéré suivant
    b = b - solve(Hessian) %*% Gradient
    L[i] = sum(y*log(pi(x,b)) + (1-y)*log(1-pi(x,b)))
  }
  return(list(L = L, b = b, Variance = Variance))
}

#### Computation of LRT + score statistic:

## LRT : Maximum likelihood under H0
# We remove the related columns in the design matrix x
mod.H0 = Newton(as.matrix(x[,-2]),y)
LH0 = mod.H0$L[10] # log-likelihood under H0
LRT = -2*(LH0 - L[10]) #
pchisq(LRT,1,lower.tail = FALSE) # Reject of null hypothesis for alpha = 0.05
```

```
## [1] 5.334463e-12
```

```r
## Score test
# Parameters bH0
bH0 = matrix(c(mod.H0$b,0),ncol = 1)
bH0
```

```
##           [,1]
## [1,] 0.4054651
## [2,] 0.0000000
```

```r
# Gradient in bH0
Gradient = t(x) %*% matrix(y - pi(x,bH0),ncol = 1)
Gradient
```

```
##               [,1]
## [1,] -8.881784e-16
## [2,]  3.205048e+01
```

```r
# Variance in bH0
Vtilde = diag(as.vector(pi(x,bH0)*(1-pi(x,bH0))))
Hessian = -t(x) %*% Vtilde %*% x
```

```r
Variance = solve(- Hessian)
Variance
```

```
##              [,1]         [,2]
## [1,]  0.041769464 -0.001932616
## [2,] -0.001932616  0.036333633
```

```r
# Score statistic
score = t(Gradient) %*% Variance %*% Gradient
# p-value
pchisq(score,1,lower.tail = FALSE)
```

```
##              [,1]
## [1,] 1.000909e-09
```

```r
### Confidence intervals on the parameter


alpha = 0.05
modele = Newton(x,y)
u = qnorm(1 - alpha/2)
b0 = modele$b[1]
b1 = modele$b[2]
s0 = sqrt(modele$Variance[1,1])
s1 = sqrt(modele$Variance[2,2])
ICb0 = c(b0 - u*s0, b0 + u*s0) # IC on b0
ICb1 = c(b1 - u*s1, b1 + u*s1) # IC on b1

## IC sur the odds-ratio
# Let focus on the odds-ration of  x = 3 vs x = 2 : OR(3/2)
Odds3 = pi(matrix(c(1,3),1,2),modele$b)/(1-pi(matrix(c(1,3),1,2),modele$b))
# or
exp(matrix(c(1,3),1,2) %*% modele$b)
```

```
##          [,1]
## [1,] 724.9961
```

```r
Odds2 = pi(matrix(c(1,2),1,2),modele$b)/(1-pi(matrix(c(1,2),1,2),modele$b))
# or
exp(matrix(c(1,2),1,2) %*% modele$b)
```

```
##          [,1]
## [1,] 101.2546
```

```r
# OR(3/2)
OR3_2 = Odds3/Odds2
OR3_2
```

```
##         [,1]
## [1,] 7.16013
```

```r
# Ce qu'on aurait pu calculer directement par exp(b1*(60 - 50))
exp(modele$b[2]*(3 - 2))
```

```
## [1] 7.16013
```

```r
# Then we deduce the confidence interval for the odds-ration
exp(ICb1*(3-2))
```

```
## [1]  3.25396 15.75541
```