Package ROBCBI
Conditionally Unbiased Bounded Influence Estimates for Binomial and Poisson Regression

Alfio Marazzi and Alex Randriamiharisoa

April 20, 2018

Contents

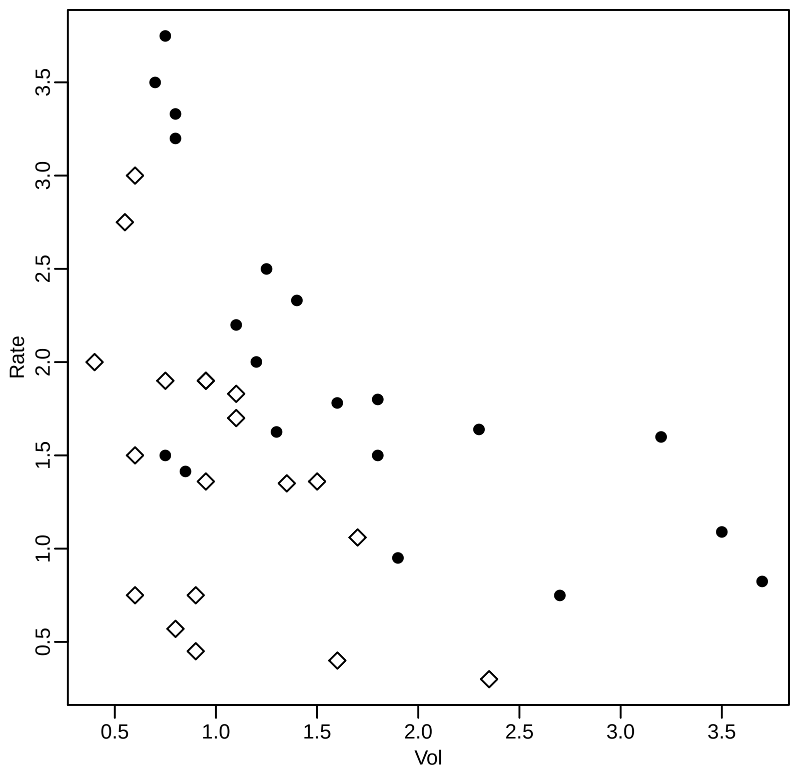
1 Introductory examples

We first consider an example from Finney(1947), which has been used in Pregibon (1981, 1982) and Künsh et al. (1989). There are 39 observations on tree variables: `Resp`, `Vol`, and `Rate`. The data were obtained in a study of the effect of the rate (`Rate`) and volume (`Vol`) of air inspired on a transient vaso-constriction in the skin of the digits. Only the occurrence or nonoccurrence of vaso-constriction (`Resp`) was measured. Type:

```
> library(robcbi)
> data(Finney)
> Vol <- Finney$Vol; Rate <- Finney$Rate; Resp <- Finney$Resp
```

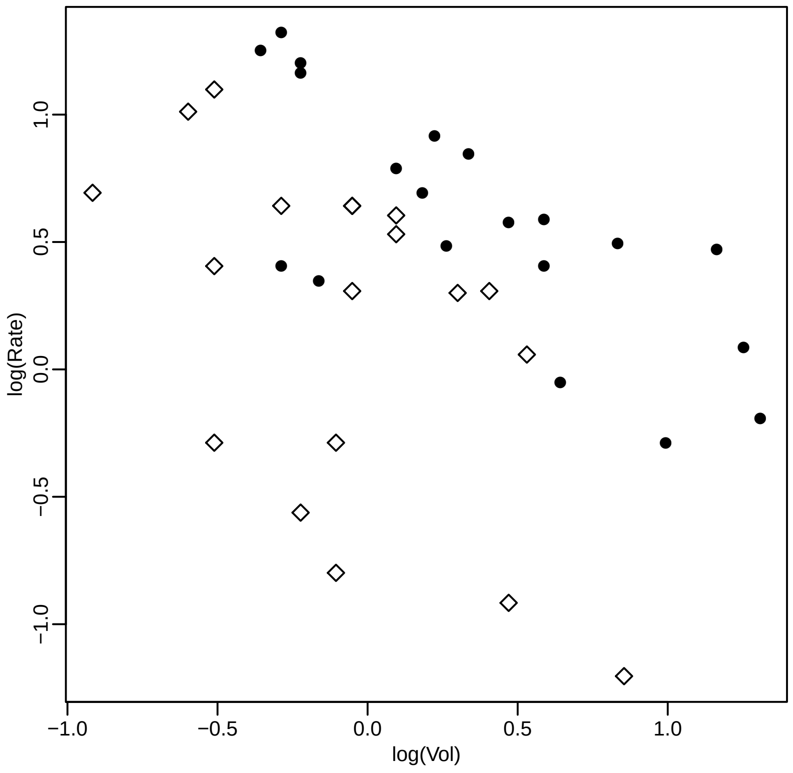
A plot of the data can be obtained by typing:

```
> plot(Vol,Rate,type="n")
> points(Vol[Resp==0],Rate[Resp==0],pch=5, cex=1.2)
> points(Vol[Resp==1],Rate[Resp==1],pch=16,cex=1.2)
```



The shape of the point suggests the use of a logarithmic transformation for both regressor variables, `Vol` and `Rate`. To see how the shape changes, use `plotFdat()` (this function is loaded together with `data(Finney)`).

```
> lVol <- log(Vol); lRate <- log(Rate)
> plotFdat <- Finney$plotFdat
> plotFdat()
```



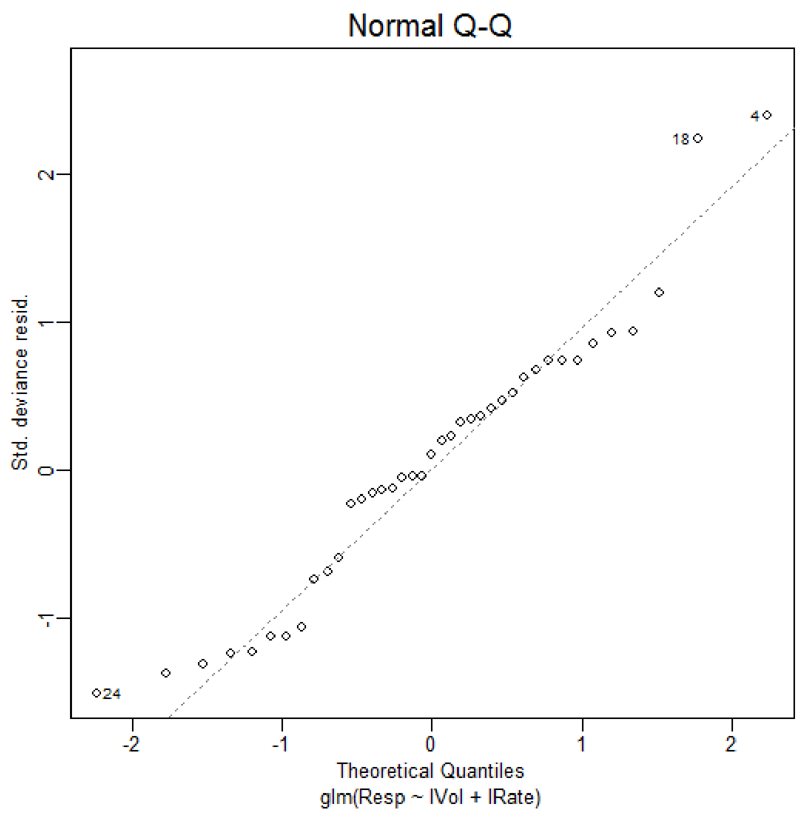
We now use the function `glm()` to fit the model $\text{logit}(\pi) = \theta_1 + \theta_2 \ln(\texttt{Vol}) + \theta_3 \ln(\texttt{Rate})$, where $\text{logit}(\pi) = \ln(\pi/(1-\pi))$. Type

```
> ML <- glm(Resp ~ lVol+lRate,family=binomial)
> summary(ML)

Call:
glm(formula = Resp ~ lVol + lRate, family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.44806  -0.60489   0.09855   0.61009   2.29051
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.924      1.288  -2.270  0.02318 *
lVol           5.220      1.858   2.810  0.00496 **
lRate          4.631      1.789   2.589  0.00964 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 54.040  on 38  degrees of freedom
Residual deviance: 29.264  on 36  degrees of freedom
AIC: 35.264

Number of Fisher Scoring iterations: 6
```

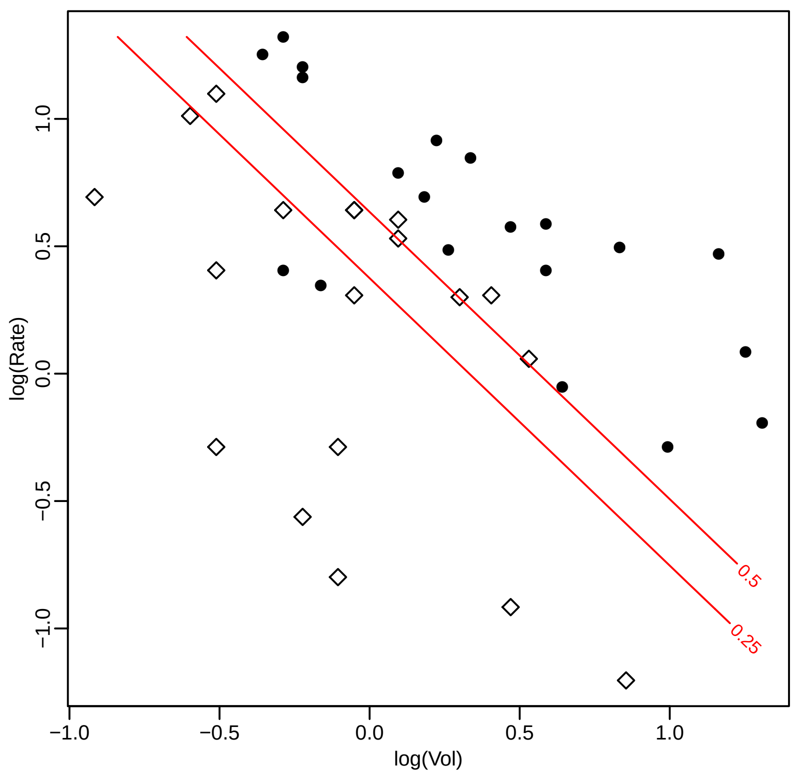
From the summary table we read the estimated coefficients and their standard errors. The deviance for the fit is 29.26 on 36 degrees of freedom and is less than its asymptotic expectation of 36, indicating no gross inadequacy with the model. However, the diagnostic given by the second plot

```
> plot(ML)# Works only in interactive mode
```



suggest that observations 4 and 18 are not well fitted. A contour plot is generated by

```
> plotFdat(zc=ML,cont=T)
```



Two black points (`Resp==1`) are below the 25% contour line shown on the scatter diagram. Using

```
> identify(lVol,lRate)# Works only in interactive mode
```

the two points are identified as observations 4 and 18. (To activate the cursor, click the primary button of the mouse. Click the secondary button to exit the program.) Since these observations are not really associated with extreme values in the design space, their effect on the fit might presumably be small. To confirm this claim, we remove them.

```
> n = length(Resp)
> out <- c(4,18); iii <- (1:n)[-out]
> MLs <- glm(Resp~lVol+lRate, family=binomial, subset=iii)
> summary(MLs)

Call:
glm(formula = Resp ~ lVol+lRate, family=binomial, subset=iii)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.74412  -0.00049   0.00000   0.00425   1.54579

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -24.58      14.02  -1.753   0.0796 .
lVol           39.55      23.25   1.701   0.0889 .
lRate          31.94      17.76   1.798   0.0721 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 51.266  on 36  degrees of freedom
Residual deviance:  7.361  on 34  degrees of freedom
AIC: 13.361

Number of Fisher Scoring iterations: 10
```

A warning message from `glm` indicates numerical unstability and the summary table reveals a dramatic increase in the value of the coefficients and their standard errors. This is an indication that these two data points have a large influence on the fit. A robust fitting procedure called *conditionally unbiased bounded influence regression* due to Künsch et al. (1989) facilitates the analysis. Type:

```
> ctrl <- cubinf.control(ufact=3.2)
> cbi  <- cubinf(cbind(lVol,lRate),Resp, family=binomial,
          control=ctrl)
```

Here, `ufact=3.2` is a tuning parameter. (It is related to the paramter b in Künsch et al. (1989), $b = \texttt{ufact}\sqrt{p}$, where p is the number of regression coefficients; see Section 2.)

As another option, one can use `glm()` with the argument `method=cubinf`:

```
> CBI <- glm(Resp~lVol+lRate, family=binomial, method=cubinf,
         ufact=3.2)
```

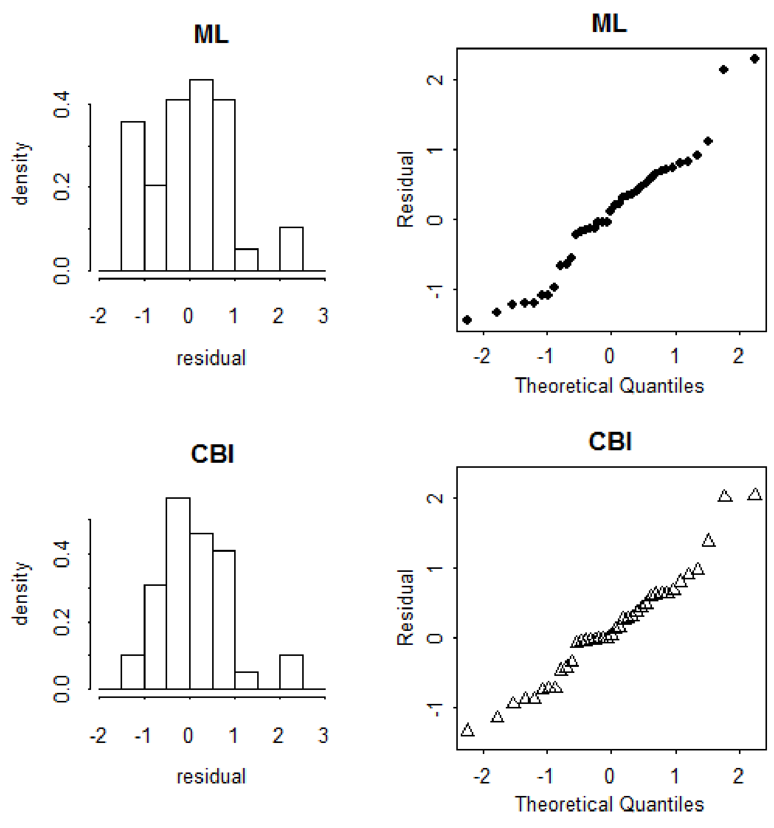
In this case, several access functions are available to retrieve the results, e.g.:

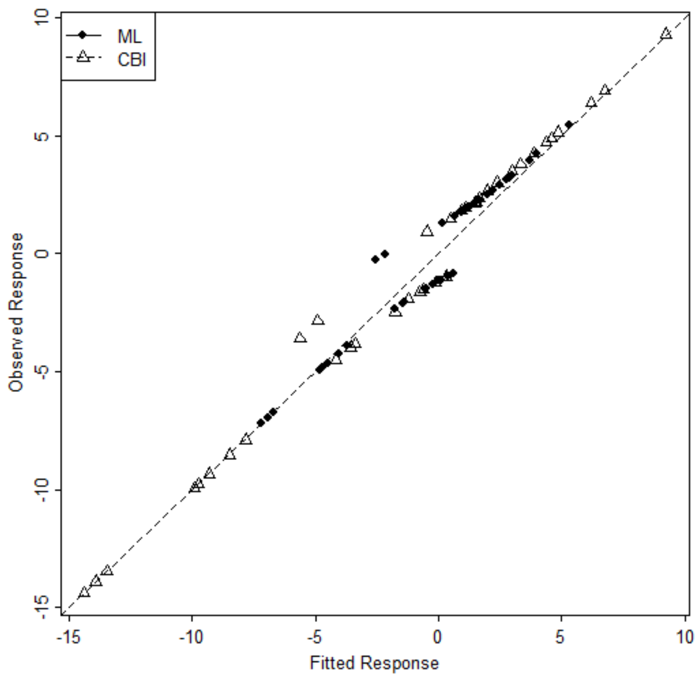
```
> summary(CBI)
> correl(CBI)
> covar(CBI)
> deviance(CBI)
> predict(CBI)
> plot(CBI)
> residuals(CBI,type="pearson")
> rscale(CBI)
```

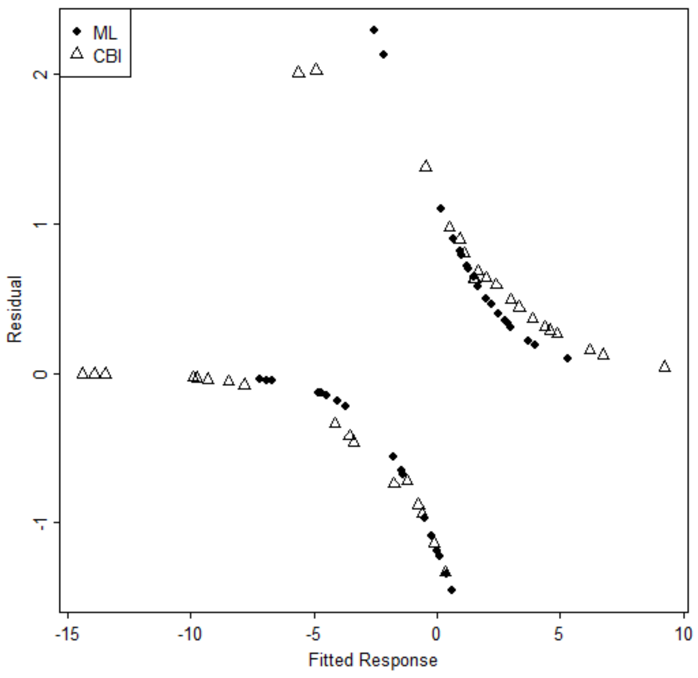
The ML and CBI results can be compared as follows:

```
> comp <- fits.compare(ML,CBI)
> comp

Calls:
Name
ML     glm(formula = Resp ~ lVol + lRate, family = binomial)
CBI    glm(formula = Resp ~ lVol + lRate, family = binomial,
           method = cubinf, ufact = 3.2)

Residual Statistics
        Min       1Q    Median     3Q    Max
ML  -0.6495 -0.16768 4.844e-03 0.1700 0.9274
CBI -0.5873 -0.02171 9.632e-05 0.1006 0.9964

Number of Parameter in each Model
    Nobs Resid df Model Parameters Est. Parameters
ML    39       36                3               3
CBI   39       36                3               3

Coefficients:
                Estimate Std. Error t value
(Intercept):ML  -2.9239   1.2666    -2.3084
(Intercept):CBI -6.3403   2.7902    -2.2723
lVol:ML          5.2205   1.8275     2.8566
lVol:CBI         9.8595   4.2668     2.3107
lRate:ML         4.6312   1.7596     2.6319
lRate:CBI        8.7659   3.7042     2.3665

Residual Scale Estimates:
ML  : 10.01
CBI : 1

Correlation of Coefficients:
```

```
Model =  ML
       (Intercept) lVol
lVol  -0.8089
lRate -0.9286      0.8048

Model =  CBI
       (Intercept) lVol
lVol  -0.9475
lRate -0.9746      0.9363

> plot(comp)  # Works only in interactive mode
```







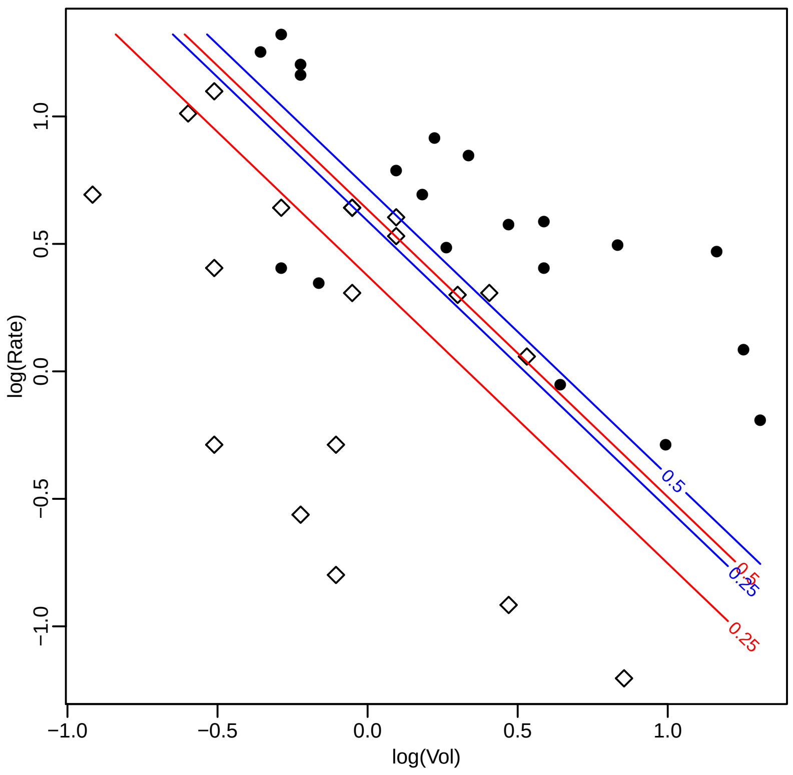
The robust procedure automatically downweighs influential points by means of a weight system. The weights can be obtained by typing:

```
> weights(CBI)
```

```
 [1] 1.0000000 1.0000000 1.0000000 0.2573661 1.0000000 1.0000000
 [7] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[13] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.2970314
[19] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[25] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[31] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[37] 1.0000000 1.0000000 1.0000000
```

Weights smaller than 1 indicate outliers. Note that the weights of observations 4 and 18 are 0.25 and 0.29, respectively, whereas all other observations receive weight equal 1. The weights depend on `ufact`. By setting `ufact` very large (e.g., 100), all weights are equal 1; in this case, the robust procedure gives the classical results based on the complete data set. As `ufact` decreases, these two weights decrease rapidly, clearly indicating the anomalous nature of these two observations. Classical and robust contour plots are generated by:

```
> plotFdat(zc=ML,zr=CBI,rob=T,cont=T)
```

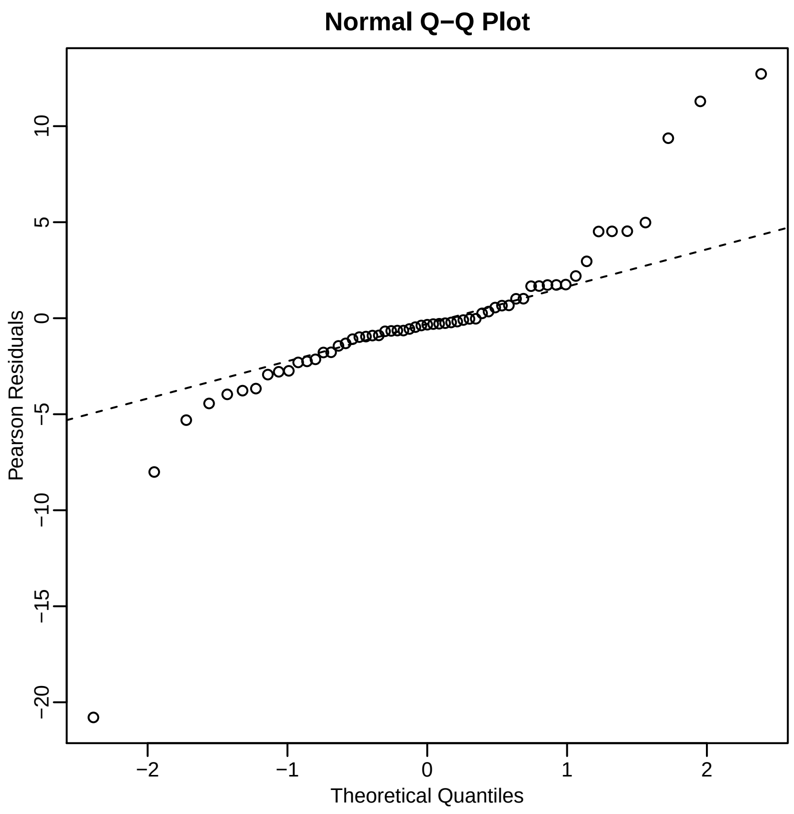


As a second example we now consider the `Breslow` dataset. The data as well as a description are obtained as follows:

```
> library(robcbi)
> data(Breslow)
> help(Breslow)
> y  = Breslow$sumY
> x1 = Breslow$Age10
> x2 = Breslow$Base4
> x3 = rep(0,length(y))
> x3[Breslow$Trt=="progabide"] = 1
> x4 = x2*x3
```

We compute the robust estimate of a Poisson regression model and draw a normal qq-plot,

```
> CBA = glm(y~x1+x2+x3+x4,family=poisson,method=cubinf,ufact=3.2)
> plot(CBA,num=5)


Coefficients:
(Intercept)          x1          x2          x3          x4
     1.8335      0.1939      0.1112     -0.3591      0.0198

Degrees of Freedom: 58 Total (i.e. Null); 54 Residual
Residual Deviance: 379.4517
```



which shows that the distribution of the Pearson residuals has heavy tails and

```
> weights(CBA)
```

```
 [1] 1.000000000 1.000000000 1.000000000 1.000000000 1.000000000
 [6] 1.000000000 1.000000000 0.365586057 1.000000000 0.814194127
[11] 1.000000000 1.000000000 1.000000000 1.000000000 0.189910326
[16] 0.830589863 1.000000000 0.060586568 1.000000000 1.000000000
[21] 1.000000000 1.000000000 1.000000000 1.000000000 0.225039067
[26] 1.000000000 1.000000000 1.000000000 0.311636972 1.000000000
[31] 1.000000000 1.000000000 1.000000000 1.000000000 0.416164619
[36] 1.000000000 1.000000000 0.365693594 1.000000000 1.000000000
[41] 1.000000000 1.000000000 0.590191521 1.000000000 1.000000000
[46] 1.000000000 1.000000000 1.000000000 0.005812668 1.000000000
[51] 1.000000000 1.000000000 0.549705419 1.000000000 1.000000000
[56] 0.484688541 1.000000000 1.000000000 1.000000000
```

points out several influential observations receiving a weight smaller than 1.

We finally compute the R_n^2 statistic (Section 2.5) to compare CBA with a reduced model with three variables:

```
> CBA.red = update(CBA, .~.-x3-x4)
> np = 5       # number of parameters of the full model
> nq = 3       # number of parameters of the reduced model
> CVR = covar(CBA)
> CFF = coef(CBA)
> K22 = CVR[(nq+1):np,(nq+1):np]
> cff = as.matrix(CFF[(nq+1):np])
> Rn2 = t(cff)%*%solve(K22)%*%cff
> Rn2
```

```
         [,1]
[1,] 13.93261
```

Section 2 briefly describes the conditionally unbiased bounded influence estimate and the computational algorithms. Most algorithms have been taken from the ROBETH library described in Marazzi (1993). Section 3 gives a complete description of the object oriented functions.

2 The bounded influence estimate

Let $X = (x_{ij})$ denote a real $n \times p$ design matrix with rows $x_i^{\mathrm{T}} = (x_{i1}, \ldots, x_{ip})$ and $(y_1, \ldots, y_n)^{\mathrm{T}}$ a given n-vector of responses

2.1 Models and estimates

In particular, we consider the following cases:

Bernoulli

$$\text{Probability}(y_i = j|x_i) = \pi_i(1-\pi_i), \quad j = 0, 1, \tag{1}$$

Binomial

$$\text{Probability}(y_i = j|x_i) = \binom{n_i}{j}\pi_i^j(1-\pi_i)^{n_i-j}, \quad j = 0, 1, \ldots, n_i, \tag{2}$$

Poisson

$$\text{Probability}(y_i = j|x_i) = \exp(-\lambda_i)\lambda_i^j/j!\ , \quad j = 0, 1, \ldots, \infty, , \tag{3}$$

where $0 \le \pi_i \le 1$ and $\lambda_i > 0$ are unknown parameters and n_i is the number of trials at the design point x_i in the Binomial case. We define $n_i \equiv 1$ in the Bernoulli and Poisson cases, and assume that

$$\Omega : \quad E_\theta(y_i|x_i) = n_i g(x_i^{\mathrm{T}}\theta), \tag{4}$$

where θ is a p-vector of unknown parameters and

$$g(\vartheta_i) = \pi_i = \frac{\exp(\vartheta_i)}{1+\exp(\vartheta_i)} \quad \text{in the Bernoulli and Binomial cases,} \tag{5}$$

$$g(\vartheta_i) = \lambda_i = \exp(\vartheta_i) \quad \text{in the Poisson case,} \tag{6}$$

where $\vartheta_i = x_i^{\mathrm{T}}\theta$. We are interested in estimating θ.

Let $\psi_a(s) = \max[-a, \min(s, a)]$ denote the Huber function with tuning constant a. Künsch et al. (1989) define the *conditionally unbiased bounded influence estimate* of θ as the solution of the following system of equations for the p-vector θ, the n constants $c_1, \dots, c_n$, and the $p \times p$ lower-triangular matrix A:

$$\sum_{i=1}^{n} \psi_{a_i}(y_i - c_i - n_i g(\vartheta_i))x_i = 0, \tag{7}$$

$$\sum_{j=0}^{M_i} \psi_{a_i}(j - c_i - n_i g(\vartheta_i))\pi_i(j) = 0, \qquad i = 1, \dots, n, \tag{8}$$

$$\frac{1}{n}\sum_{i=1}^{n} u_b(y_i, n_i; \vartheta_i, c_i; |Ax_i|)Ax_i(Ax_i)^{\mathrm{T}} = I_p, \tag{9}$$

where I_p denotes the $p \times p$ identity matrix, $\vartheta_i = x_i^{\mathrm{T}}\theta$, $M_i = n_i$ in the Binomial case, $M_i = \infty$ in the Poisson case, $a_i = b/|Ax_i|$ (and $|\cdot|$ denotes the Euclidean norm), and $b > \sqrt{p}$ is a given tuning constant. The function u_b is defined by

$$u_b(y_i, n_i; \vartheta_i, c_i; |z_i|) = \sum_{j=0}^{M_i} \left[\psi_{b/|z_i|}(j - c_i - n_i g(\vartheta_i))\right]^2 \pi_i(j), \tag{10}$$

where $z_i = Ax_i$. Note that the argument y_i does not appear in the right-hand side of (10).

Remark 1. The classical maximum likelihood estimate is the solution θ of

$$\sum_{i=1}^{n} l_i^0(y_i, x_i^{\mathrm{T}}\theta)x_i = 0. \tag{11}$$

The functions $-l_i^0(y, \vartheta)x$ with $l_i^0(y, \vartheta) = -y + n_i g(\vartheta)$ and $i = 1, \dots, n$ are called *(classical) score functions.* (x and y denote a generic design point and a generic response.) Thus, the bounded influence estimate defined by (7) is based on truncated scores and the constants c_i defined by (8) make this estimate conditionally unbiased, given x_i. Equation (7) is also equivalent to $\sum(y_i - c_i - n_i g(\vartheta_i))w_i x_i = 0$, where $w_i = \min(1, a_i/|y_i - c_i - n_i g(\vartheta_i))|$. The vector $\vartheta = X\theta$ is called the *linear predictor*, and the function g the *inverse link function* in the literature about generalized linear models.

Remark 2. Equation (9) is an empirical version of the condition

$$A\ E_\theta\{xx^{\mathrm{T}}[\psi_{b/|z|}(y - c(x^{\mathrm{T}}\theta, b/|z|) - E_\theta(y|x))]^2\}\ A^{\mathrm{T}} = I_p, \tag{12}$$

where expectation is taken over the joint distribution of x and y and the function $c(\vartheta, a)$ is implicitly defined by

$$E_\theta[\psi_a(y - c(\vartheta, a) - E_\theta(y|x))] = 0, \tag{13}$$

where $z = Ax$ and $\vartheta = x^{\mathrm{T}}\theta$. This condition imposes a bound to the self standardized sensitivity of the estimate (Künsch et al. 1989). In applications, the distribution of x is unknown and it is common to replace it by its empirical version, thus obtaining (9) and (10). Alternatively, we can replace the joint distribution of x and y by its empirical version and obtain (9) with

$$u_b(y_i, n_i; \vartheta_i, c_i; |z_i|) = \left[\psi_{b/|z_i|}(y_i - c_i - n_i g(\vartheta_i))\right]^2. \tag{14}$$

The two choices are asymptotically equivalent.

2.2 Covariance matrix of the coefficient estimates

According to Künsch et al. (1989), under regularity conditions, the coefficient estimate $\hat\theta = (\hat\theta_1, \ldots, \hat\theta_p)$ defined by the solution of (7)–(9) is asymptotically normally distributed with covariance matrix $K_u(\psi, \theta)$ [i.e., $n^{1/2}(\hat\theta - \theta) \sim N(0, K_u(\psi, \theta))$], where

$$\begin{aligned} K_u(\psi, \theta) &= S_1^{-1}(\psi, \theta) S_2(\psi, \theta) S_1^{-1}(\psi, \theta), && (15)\\ S_1(\psi, \theta) &= E_\theta \psi(y, x, \theta, A)(y - E_\theta(y|x)) x^{\mathrm{T}}. && (16)\\ S_2(\psi, \theta) &= E_\theta \psi(y, x, \theta, A) \psi(y, x, \theta, A)^{\mathrm{T}}, && (17)\\ \psi(y, x, \theta, A) &= \psi_{b/|z|}(y - E_\theta(y|x) - c(x^{\mathrm{T}}\theta, b/|z|)) x. && (18) \end{aligned}$$

Here, $z = Ax$ and expectation is taken over the joint distribution of x and y.

In applications, the distribution of x is unknown and it is common to replace it by its empirical version, thus obtaining the following estimate of K_u:

$$\hat K_u = \hat S_1^{-1} \hat S_2 \hat S_1^{-1}, \tag{19}$$

where

$$\hat S_1 = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=0}^{M_i} \psi_{a_i}(j - c_i - n_i g(x_i^{\mathrm{T}}\hat\theta))(j - n_i g(x_i^{\mathrm{T}}\hat\theta)) \pi_i(j) x_i x_i^{\mathrm{T}}, \tag{20}$$

$$\hat S_2 = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=0}^{M_i} [\psi_{a_i}(j - c_i - n_i g(x_i^{\mathrm{T}}\hat\theta))]^2 \pi_i(j) x_i x_i^{\mathrm{T}}. \tag{21}$$

The covariance matrix C of the coefficient estimates can be estimated by multiplying $\hat K_u$ by $1/n$.

2.3 Numerical procedures

A detailed description of the numerical procedures is given in Marazzi (1993, Chapter 10). The following summary is taken from this book.

The main algorithm for computing the bounded influence estimate is based on three modules with the following purposes:

θ-step: For given values of a_i and c_i $(i = 1, \ldots, n)$, find the solution θ of (7).

c-step: For given values of ϑ_i and a_i $(i = 1, \ldots, n)$, find the solutions $c_1, \ldots, c_n$ of the equations (8).

A-step: For given values of ϑ_i and c_i $(i = 1, \ldots, n)$, find the solution A of (9) and compute $a_1, \ldots, a_n$.

The modules are based on iterative algorithms that compute improved values $\theta^{(1)}$, $c_i^{(1)}$, and $A^{(1)}$ of θ, c_i, and A given the current values $\theta^{(0)}$, $c_i^{(0)}$, and $A^{(0)}$.

Main algorithm. Let TOL be a given tolerance and $MAXIT$ a given positive integer (maximum number of iterations). The main algorithm has the following structure:

```
0. Set $NIT := 1$.
   Initialize $\theta^{(0)}$, $c_i^{(0)}$ $(i = 1, \ldots, n)$, and $A^{(0)}$; compute $a_1, \ldots, a_n$.
1. Compute $\theta^{(1)}$ using the $\theta$-step; compute $\vartheta_1, \ldots, \vartheta_n$.
2. Check convergence:
   If a measure of deviation between $\theta^{(1)}$ and $\theta^{(0)}$ (see Remark 2) is smaller than $TOL$ , go to step 6.
3. Compute $A^{(1)}$ and the corresponding values of $a_i$ $(i = 1, \ldots, n)$ using the $A$-step.
4. Compute $c_i^{(1)}$ $(i = 1, \ldots, n)$ using the $c$-step.
5. Set $\theta^{(0)} := \theta^{(1)}$, $A^{(0)} := A^{(1)}$, $c_i^{(0)} := c_i^{(1)}$ $(i = 1, \ldots, n)$.
   Set $NIT := NIT + 1$; if $NIT \leq MAXIT$, go to step 1.
6. Exit.
```

We call a *cycle* a generic iteration of the main algorithm consisting of steps 1, 2, 3, 4 and 5.

Remark 1. The reader is referred to Marazzi (1993) for the definition of the *initial values* $\theta^{(0)}$, $c_i^{(0)}$ $(i = 1, \ldots, n)$, and $A^{(0)}$.

Remark 2. For the *convergence criterion* of the main algorithm we assume that a preliminary estimate $\hat{C}$ of the covariance matrix of the coefficient estimates, or its inverse $\hat{C}^{-1}$, is available before starting iterations. According to the value of an option parameter $ICNV$, the user sets $S = \hat{C}$ if $ICNV = 1$ or $S = \hat{C}^{-1}$ if $ICNV = 2$ or 3 (for $ICNV = 1$ or 3 only the diagonal elements are required). Let $\delta = \theta^{(1)} - \theta^{(0)}$.

The main algorithm evaluates the function

$$\begin{aligned} \mathcal{I}_\delta(\delta, S, TOL, ICNV) = {} & \text{pos?}(TOL\sqrt{s_{jj}} - \delta_j) \quad \text{if} \quad ICNV = 1 \\ & \text{pos?}(TOL - \sqrt{\delta^{\mathrm{T}} S \delta}) \quad \text{if} \quad ICNV = 2 \\ & \text{pos?}(TOL - \delta_j\sqrt{s_{jj}}) \quad \text{if} \quad ICNV = 3, \end{aligned}$$

where TOL is a given tolerance for the relative error and $\text{pos?}(s) = 1$ if $s > 0$ and 0 otherwise. Iterations are stopped if $\mathcal{I}_\delta = 1$. The most common option is $ICNV = 1$.

Remark 3. Although the convergence of the algorithms used for the θ-, the c-, and the A-step can be ensured on the grounds of known general results, no proof of their joint convergence in the main algorithm is known.

The θ-step. Let s denote a given real number. We use the following notations:
Binomial case

$$g(\vartheta) = \frac{\exp(\vartheta)}{1 + \exp(\vartheta)}, \quad G(\vartheta) = \ln(1 + \exp(\vartheta)), \tag{22}$$

$$t_1 = \ln\left(\frac{s-a}{n_i - s + a}\right), \quad t_2 = \ln\left(\frac{s+a}{n_i - s - a}\right), \tag{23}$$

$$\xi_1 = -(s-a)t_1 + n_i \ln\left(\frac{n_i}{n_i - s + a}\right), \quad \xi_2 = -(s+a)t_2 + n_i \ln\left(\frac{n_i}{n_i - s - a}\right). \tag{24}$$

Poisson case

$$g(\vartheta) = \exp(\vartheta), \quad G(\vartheta) = \exp(\vartheta), \tag{25}$$

$$t_1 = \ln(s-a), \quad t_2 = \ln(s+a), \tag{26}$$

$$\xi_1 = -(s-a)t_1 + (s-a), \quad \xi_2 = -(s+a)t_2 + (s+a). \tag{27}$$

For a given i let $L_i(s,\vartheta,a)$ and $l_i(s,\vartheta,a)$ be the functions defined in Table 1 and Table 2; note that $l_i(s,\vartheta,a) = (\partial/\partial\vartheta)L_i(s,\vartheta,a)$. The values t_1 and t_2 defined above are the solutions of $l^0(s,\vartheta) = -a$ and $l^0(s,\vartheta) = a$ (l^0 is defined in Section 2, Remark 1) and the values ξ_1 and ξ_2 are chosen so that the function $L_i(s,\vartheta)$ is continuous in ϑ. Clearly, for some values of s, a, and n_i, one or both solutions may not exist.

The solution of (7) for given values of $a_1,\ldots,a_n$, $c_1,\ldots,c_n$ characterizes the minimum of

$$Q(\theta) = \sum_{i=1}^{n} L_i(s_i, x_i^{\mathrm{T}}\theta, a_i), \tag{28}$$

where $s_i = y_i - c_i$. As $Q(\theta)$ is a convex and differentiable function, we use the Newton algorithm for solving this optimality problem.

The gradient and the Hessian matrix of Q are

$$\nabla Q(\theta) = \frac{\partial Q}{\partial\theta} = X^{\mathrm{T}}\ell \qquad \text{and} \qquad H(\theta) = \frac{\partial^2 Q}{\partial\theta\partial\theta^{\mathrm{T}}} = X^{\mathrm{T}}\mathcal{D}X, \tag{29}$$

where $\ell = (\ell_1,\ldots,\ell_n)^{\mathrm{T}}$ and

$$\ell_i = l_i\left(s_i,\vartheta_i,a_i\right) \quad \text{for} \quad i = 1,\ldots,n, \tag{30}$$

$$\mathcal{D} = \mathrm{diag}\left({}'_i\left(s_i,\vartheta_i,a_i\right)\right). \tag{31}$$

The Newton algorithm computes an improvement δ by solving $-X^{\mathrm{T}}\mathcal{D}X\delta = X^{\mathrm{T}}\ell$. In other words, if $\theta^{(0)}$ denotes the current value of θ, we approximate $Q(\theta^{(0)}+\delta)$ by

$$V(\delta) = Q(\theta^{(0)}) + \delta^{\mathrm{T}}\nabla Q(\theta^{(0)}) + \frac{1}{2}\delta^{\mathrm{T}}H(\theta^{(0)})\delta \tag{32}$$

and minimize $V(\delta)$. There are two well-known difficulties: (1) H can become singular; (2) $|\delta|$ can be too large (and thus the objective function Q is not reduced). A remedy for (1) is the use of a positive definite generalized inverse as described in Marazzi (1993, Chapter 2, Section 2.1.2, Remark 3, with $\breve{X} = \mathcal{D}^{1/2}X$), whereas difficulty (2) is usually overcome by means of an adaptive step length algorithm.

An alternative procedure for reducing $|\delta|$ is to minimize a quadratic comparison function $\bar{V}$ that majorizes Q. Let $\bar{V}$ be defined by

$$\bar{V}(\delta) = \sum \bar{L}_i\left(x_i^{\mathrm{T}}\delta\right) \tag{33}$$

with

$$\bar{L}_i(\vartheta) = \bar{a}_i + \bar{b}_i\vartheta + \bar{c}_i\vartheta^2, \tag{34}$$

where $2n$ of the $3n$ coefficients are determined so that $\bar{L}_i(\vartheta)$ and $L_i(s_i, \vartheta, a_i)$ have the same value and the same tangent for $\vartheta = x_i^{\mathrm{T}}\theta^{(0)}$. Writing $\bar{H} = X^{\mathrm{T}}\bar{\mathcal{D}}X$, the comparison function becomes

$$\bar{V}(\delta) = Q(\theta^{(0)}) + \delta^{\mathrm{T}}\nabla Q(\theta^{(0)}) + \frac{1}{2}\delta^{\mathrm{T}}\bar{H}\delta. \tag{35}$$

Finally, the n diagonal elements $\bar{c}_i/2$ of $\bar{\mathcal{D}}$ are chosen so that $\bar{V}(\delta) \geq Q(\theta^{(0)} + \delta)$ for all δ. This procedure clearly reduces the objective function and thus no further adaptation of the step length is needed. See Marazzi (1993) for details.

Table 1 Functions L_i and l_i for the Binomial case

Condition	Condition	$L_i(s,\vartheta,a)$	$l_i(s,\vartheta,a)$
1. $\begin{cases} -s < -a \\ -s+n_i \le -a \end{cases}$	—	$-a\vartheta$	$-a$
2. $\begin{cases} -s < -a \\ -s+n_i \in (-a,a] \end{cases}$	$\vartheta < t_1$	$-a\vartheta+\xi_1$	$-a$
	$\vartheta \ge t_1$	$-s\vartheta+n_iG(\vartheta)$	$-s+n_ig(\vartheta)$
3. $\begin{cases} -s < -a \\ -s+n_i > a \end{cases}$	$\vartheta < t_1$	$-a\vartheta+\xi_1$	$-a$
	$\vartheta \in [t_1,t_2]$	$-s\vartheta+n_iG(\vartheta)$	$-s+n_ig(\vartheta)$
	$\vartheta > t_2$	$a\vartheta+\xi_2$	a
4. $\begin{cases} -s \in [-a,a) \\ -s+n_i \le a \end{cases}$	—	$-s\vartheta+n_iG(\vartheta)$	$-s+n_ig(\vartheta)$
5. $\begin{cases} -s \in [-a,a) \\ -s+n_i > a \end{cases}$	$\vartheta \le t_2$	$-s\vartheta+n_iG(\vartheta)$	$-s+n_ig(\vartheta)$
	$\vartheta > t_2$	$a\vartheta+\xi_2$	a
6. $\begin{cases} -s \ge a \\ -s+n_i \ge a \end{cases}$	—	$a\vartheta$	a

Table 2 Functions L_i and l_i for the Poisson case

Condition	Condition	$L_i(s,\vartheta,a)$	$l_i(s,\vartheta,a)$
1. $-s < -a$	$\vartheta < t_1$	$-a\vartheta+\xi_1$	$-a$
	$\vartheta \in [t_1,t_2]$	$-s\vartheta+G(\vartheta)$	$-s+g(\vartheta)$
	$\vartheta > t_2$	$a\vartheta+\xi_2$	a
2. $-s \in [-a,a)$	$\vartheta \le t_2$	$-s\vartheta+G(\vartheta)$	$-s+g(\vartheta)$
	$\vartheta > t_2$	$a\vartheta+\xi_2$	a
3. $-s \ge a$	—	$a\vartheta$	a

2.4 Deviance D^*

Consider the p parameter model $\Omega : y = X\theta + e$, and denote by $\hat{\theta}_\Omega$ the robust estimate of θ in the model Ω. This estimate depends on the values $a_1,\ldots,a_n$ and $c_1,\ldots,c_n$ defined by (7)–(9). Let $Q(\theta) = \sum_{i=1}^n L_i(y_i - c_i, x_i^{\mathrm{T}}\theta, a_i)$ be the function defined in (28) and let θ^* be the minimum of this function when $X = I_n$. θ^* is usually called the *unconstrained minimum* of $Q(\theta)$. The *deviance* D^* of Ω is defined as

$$D^* = 2Q(\hat{\theta}_\Omega) - 2Q(\theta^*).$$

2.5 Residuals

Residuals for GLMs are defined in several different ways. For the conditionally unbiased bounded influence estimates, the `residual()` method for object has a `type=` argument, with three choices:

- `"deviance"` (the default): Deviance residuals are defined as

  $$r_i^D = \text{sign}(y_i - \hat{\mu}_i)\sqrt{d_i},$$

  where d_i is the contribution of the ith observation to the deviance, i.e.,

  $$d_i = 2L_i(y_i - c_i, x_i^{\mathrm{T}}\hat{\theta}_\Omega, a_i) - 2L_i(y_i - c_i, \theta_i^*, a_i),$$

  and $\hat{\mu}_i = n_i g(\hat{\theta}_{\Omega,i})$.

- `"pearson"`: In analogy with the classical case, we define these residuals as

  $$r_i^P = (y_i - \hat{\mu}_i)/\sqrt{v(\hat{\mu}_i)},$$

  where $v(\hat{\mu}_i) = \hat{\mu}_i$ in the Poisson case, $v(\hat{\mu}_i) = \hat{\mu}_i(1 - \hat{\mu}_i)$ in the Bernoulli case, $v(\hat{\mu}_i) = \hat{\mu}_i(1 - \hat{\mu}_i)/n_i$ in the Binomial case, and $\hat{\mu}_i = n_i g(\hat{\theta}_i)$.

- `"response"`: These are simply $y_i - \hat{\mu}_i$, where $\hat{\mu}_i = n_i g(\hat{\theta}_i)$.

2.6 Test of a linear hypothesis

In order to test a linear hypothesis the R_n^2-test described in Hampel et al. (1986, Chapter 7) can be used. Consider the p parameter model

$$\Omega : \qquad y = X\theta + e,$$

and let a linear hypothesis be expressed in the canonical form

$$\mathcal{H}_0 : \theta_{q+1} = \theta_{q+2} = \ldots = \theta_p, \qquad 0 < q < p.$$

Denote by $\hat{\theta}_\Omega$ the robust estimate of θ in the model Ω and let K_{22} be the $(p - q) \times (p - q)$ lower right block of the asymptotic covariance matrix of $\hat{\theta}_\Omega$. The R_n^2-test statistic is defined by

$$R_n^2 = n(\hat{\theta}_{\Omega,q+1}, \ldots, \hat{\theta}_{\Omega,p})K_{22}^{-1}(\hat{\theta}_{\Omega,q+1}, \ldots, \hat{\theta}_{\Omega,p})^{\mathrm{T}}.$$

Under $\mathcal{H}_0$, the R_n^2-test statistic follows approximatively a χ^2-distribution with $(p - q)$ degrees of freedom.

3 Functions

Two types of functions are provided:

(a) Functions for the numerical computations of the estimates;

(b) Interface functions for the use of the estimation functions within the object oriented paradigm of R, as described in Chambers and Hastie (1992).

The functions of type (b) are likely to be appropriate for most users and applications. In order to control the computational algorithms in details some users may want to use functions of type (a). In addition there are service functions which are normally intended for programmers and developers. These functions are not necessarily documented elsewhere and are listed here for completeness.

3.1 The estimation functions

The main estimation function is `cubinf()`, which is is used by `glm()` to fit the conditionally unbiased bounded influence estimates. Alternatively, `cubinf()` can be called directly. `cubinf()` returns a list containing the estimate of coefficients and other values returned by the function `gymain` of the package `robeth`. (See `help(cubinf)` for more details). Computations are performed using routines from the `robeth` package as follows:

1. The initial values of $\hat{\theta}$, A and c_i, $i = 1, \ldots, n$ are computed using the function `gintac` (Marazzi 1993, p.292).
2. The M-estimates are computed by means of the function `gymain` (Marazzi 1993, p.304).
3. Covariance matrices of estimated coefficients are obtained using the functions `ktaskw` and `gfedca` (Marazzi 1993, p.148, and p.309).

The following function can be used in order to change defaults:

- `cubinf.control()` – There are a few control parameters for the numerical algorithms. Their default values are collected in the auxiliary function `cubinf.control()` and may be set with this function. The control structure input in `cubinf()` defaults to the list returned by `cubinf.control()`. Users may wish to set their own control parameters. This is easily accomplished. In either `cubinf()` or `glm()`, simply set `control=`*xcontrol* in the calling sequence of either function. Here *xcontrol* is any object produced by `cubinf.control()` with the desired control parameter values. Use `help(cubinf.control)` for additional details.

The following functions are service functions:

- The functions `gintac()`, `gymain()`, `gfedca()`, `ktaskw()`, `mchl()`, `minv()`. `mtt1()`, `glmdev()`, are wrapper functions for the `robeth` functions used by `cubinf()`.

- The `robeth` functions mentioned above call other routines of the same library as explained in Marazzi (1993).

3.2 The object oriented interface

The function `cubinf()` computes the robust estimates and returns a list of results. These are more easily produced using `method="cubinf"` in the R function `glm()`. The value returned by `glm()` is an object with class "cubinf" inheriting from class "glm". The function `glm()` was modified to call `cubinf()` when `method="cubinf"`.

The functions `print()`, `summary()` and `plot()` as well as the access functions `coef()`, `residual()`, `fitted()`, `formula()` and `deviance()` have been extended to objects of the class "cubinf".

The following access functions are new.

- `scale.estimate()`, `covar()`, `correl()`, `weights()`, `Rank()` — These functions extract the scale estimate, the covariance matrix of the estimated coefficients, the corresponding correlation matrix, the weights w_i, and the rank of the design matrix.

Note. The weights w_i are defined as $w_i = \min(1, a_i/|y_i - c_i - n_i g(\vartheta_i))|$ (see Section 2.1, Remark 1).

In addition, a function for comparing the two fits has been added.

- `fits.compare()` — This function accepts a sequence of objects of class "glm" or "cubinf" (with optional names), and creates a class "fits.compare" object. The "fits.compare" objects is nothing more than a list of the input objects with names. However, when the “fits.compare” object is printed, summaries of each of the input objects are computed and printed in a manner suitable for comparing the input models. Plotting the “fits.compare” object results in a sequence of graphical displays. These displays are designed to be of use in comparing two sets of parameter estimates in linear models.

Finally, a number of new auxiliary functions are required by the interface. Many of these routines are simple program stubs used to print error messages that the associated `glm()` function is not yet available for objects from the class.

- `print.cubinf()`, `summary.cubinf()`, and `print.summary.cubinf()` — These functions extend the `print.lm()`, `summary.lm()`, and `print.summary.lm()` functions to objects of class “cubinf”.
- `plot.cubinf()` — extends `plot()` to objects of class “cubinf”.
- `plot.fits.compare()` — extends `plot()` to objects of class “fits.compare”.
- `deviance.cubinf()`, `scale.estimate.cubinf()`, `covar.cubinf()`, `correl.cubinf()`, `weights.cubinf()`. `Rank.cubinf()` — These functions are required by the corresponding access functions.
- `anova()`, `add1.cubinf()`, `drop1.cubinf()`, and `step.cubinf()` — These are function stubs used to print error messages that the associated routines are not implemented.
- Other function stubs of class “glm”, `effects()`, `kappa()`, `proj()`, `alias()`, `lm.influence()`, and `lm.sensitivity()` are not applicable to objects of class “cubinf”. Therefore, function stubs `effects.cubinf()`, `kappa.cubinf()`, `proj.cubinf()`, `lm.influence.cubinf()`, `alias.cubinf()`, and `lm.sensitivity.cubinf()` have been added to issue warning messages for these functions.
- The function `update()` has been extended in order to compare two nested models when `method="cubinf"`.

Acknowledgment. This work was supported by grant No. 21–36521.92. from the Swiss National Science Foundation

4 References

Chambers J.M., Hastie T.J., Eds. (1992).
Statistical Models in S, Wadsworth & Brooks/Cole Computer Science Series, Pacific Grove.

Finney, D.J. (1947). The estimation from individual records of the relationship between dose and quantal response. *Biometrika*, 34, 320-334.

Hampel F.R., Ronchetti E.M., Rousseeuw P.J., Stahel W.A. (1986).
Robust Statistics: The Approach Based on Influence Functions, Wiley, New York.

Künsch, H.R., Stefanski L.A., Carrol, R.J. (1989). Conditionally unbiased bounded-influence estimation in general regression models, with application to generalized linear models. *Journal of the American Statistical Association*, 84, 460-466.

Marazzi A. (1993). Algorithms, Routines and S Functions for Robust Statistics, The FORTRAN library ROBETH with an interface to S-Plus, Wadsworth & Brooks/Cole Statistics/Probability Series, Pacific Grove.

Pregibon D. (1981). Logistic regression diagnostic. *The Annals of Statistics*, 9, 705-724.