Use of the package `nlstools` to help the fit and assess the quality of fit of a gaussian nonlinear model

Marie Laure Delignette-Muller and Florent Baty

November 30, 2012

The package `nlstools` provides several tools that help to fit of a gaussian nonlinear model [1] using the fonction `nls` and to assess its quality of fit.

$$y = f(\theta, x) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma)$$

The aim of this document is to provide examples showing how to use these tools that help to fit a model to data using the fonction `nls`, to check the validity of the assumptions of the model, to assess its quality of fit, to evaluate the precision of parameters estimates by use of confidence intervals or regions, ... For details, see the documentation of each function, using the R help command (e.g. `?nlsResiduals`). Do not forget to load the library using the function `library` before testing the following examples.

```
> library(nlstools)
```

Contents

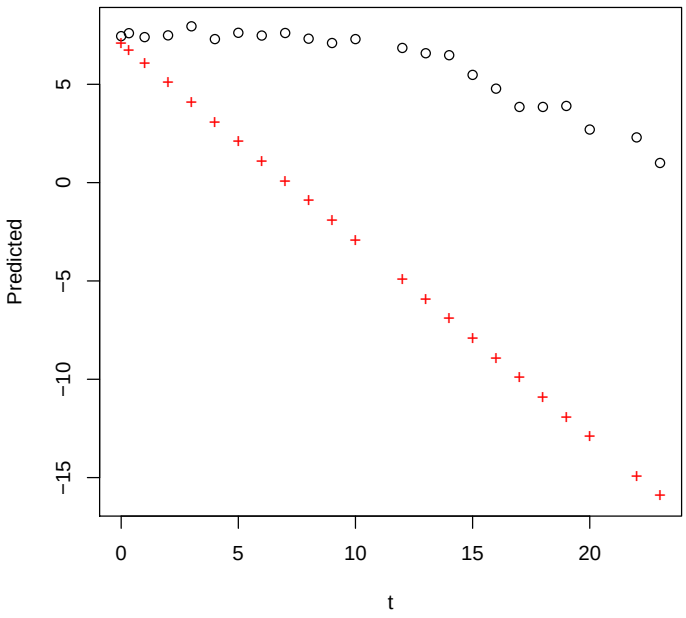
1 Help to fit a model

1.1 Help to define starting values for parameters

To fit a nonlinear model, it is necessary to specify starting estimates for parameters. The function `preview` may be used to facilitate the choice of these values. It provides a superimposed plot of observed (black circles) and predicted (red crosses) values of the dependent variable versus one of the independent variables with the model evaluated at specified values of the parameters. The residual sum-of-squares (RSS) give an indication of the distance between observed and predicted values (the lower, the better). It is then easy to use it repeatedly to reach a good approximation of the starting estimates as in the following example. This example uses a dataset and a model available in the package `nlstools`.

- First iteration with arbitrary values of parameters

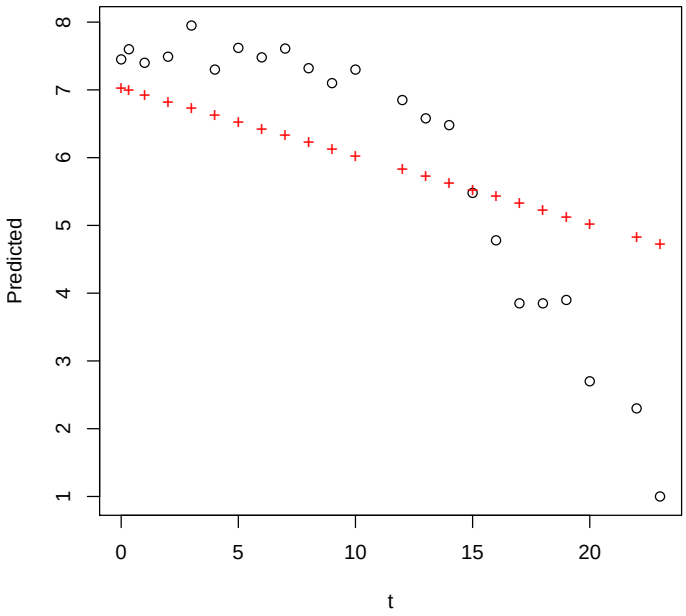
```
> data(survivalcurve2)
> preview(formula=mafart, data=survivalcurve2,
+ start=list(p = 1, delta = 1, LOG10N0 = 7))

RSS:  2780
```



- Second iteration with adjusted values of parameters

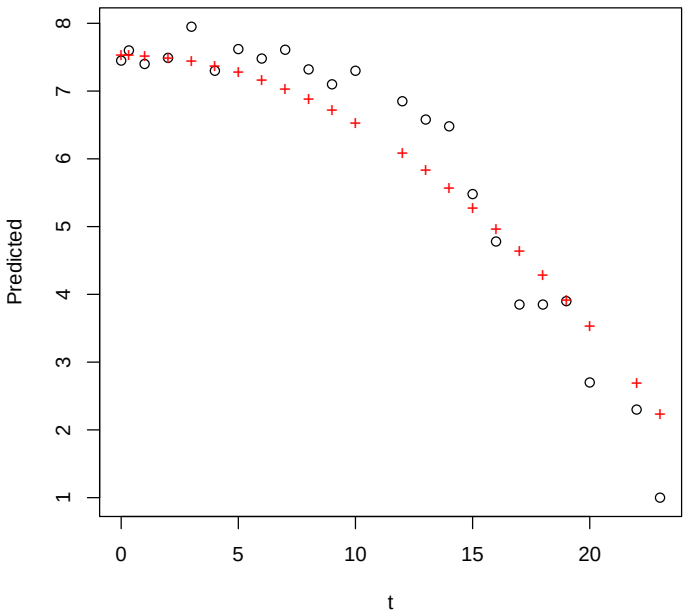
```
> preview(formula=mafart, data=survivalcurve2,
+ start=list(p = 1, delta = 10, LOG10N0 = 7))
```

```
RSS:  45.1
```



- Last iteration with reasonable approximate values of parameters

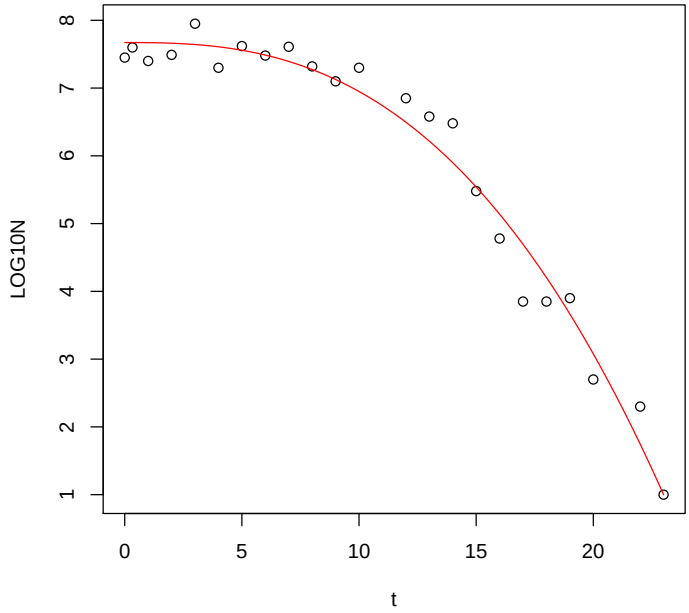
```
> preview(formula=mafart, data=survivalcurve2,
+ start=list(p = 2, delta = 10, LOG10N0 = 7.5))

RSS:  7.11
```



1.2 Fit and plot of fit

When reasonable starting estimates are available for parameters, the model may be fitted using the function `nls`, and the fitted model may be plotted together with the observed data points using the function `plotfit`. Its argument `smooth` enables to draw a smooth line for the fitted model.

```
> nlsmaf <- nls(mafart, survivalcurve2, list(p = 2, delta = 10, LOG10N0 = 7.5))
> plotfit(nlsmaf,smooth=TRUE)
```

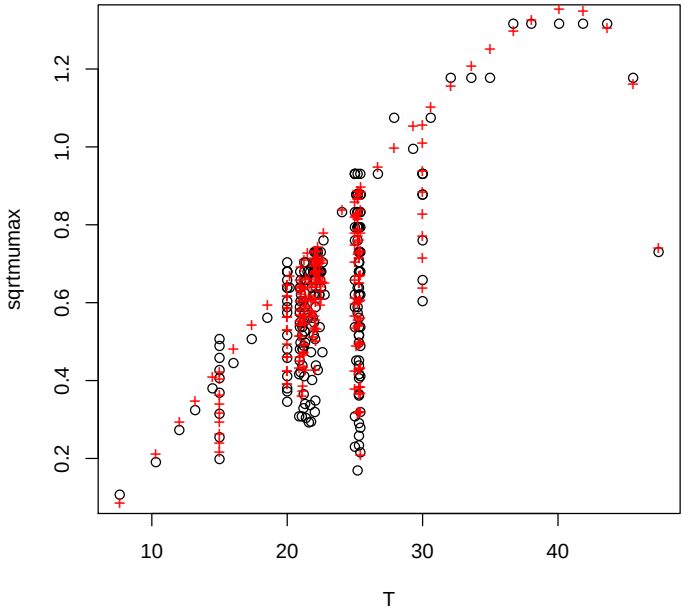


Previous example uses a model predefined in `nlstools`. However, any model may be fitted by explicitly defining its formula. Beware that the names of variables in the model formula must exactly match those in the dataset.

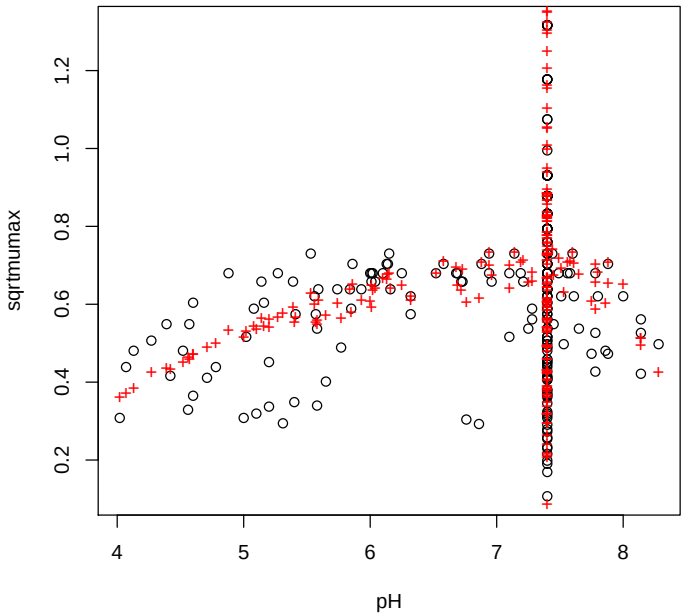
```
> model <- LOG10N ~ LOG10N0 - (t/delta)^p
> nlsmaf <- nls(model, survivalcurve2, list(p = 2, delta = 10, LOG10N0 = 7.5))
```

For the plot of a model with more than one independent variable, it is necessary to specify the argument `variable` of the function `plotfit` to indicate which variable is plotted on the x-axis. In that case, it is not possible to use the argument `smooth` to plot a smooth line for the fitted model. In the following example, a model with 4 independent variables is fitted and the dependent variable is plotted first versus the first independent variable, and then versus the second one.

```
> data(ross)
> d6<-subset(ross, select = c(T, pH, aw, sqrtmumax))
> nls6 <- nls(cpm_T_pH_aw, d6, list(muopt = 2, Tmin = 4,
+ Topt = 40, Tmax = 49,pHmin = 4, pHopt = 6.5, pHmax = 9,
+ awmin = 0.95, awopt = 0.995))
> plotfit(nls6, variable = 1)
```




```
> plotfit(nls6, variable = 2)
```



An extended summary of the fit (giving more information than `summary(nls)`) may be printed using the function `overview`, as below.

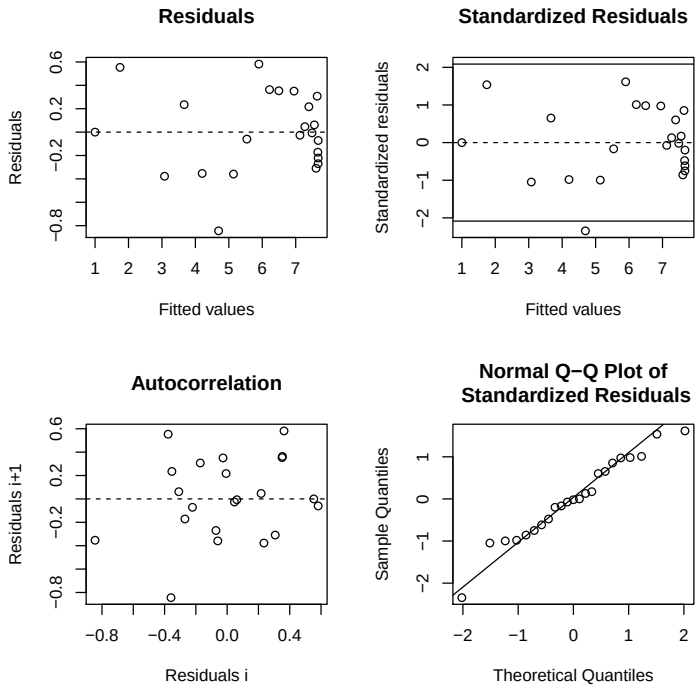
```
> overview(nlsmaf)

------
Formula: LOG10N ~ LOG10N0 - (t/delta)^p

Parameters:
        Estimate Std. Error t value Pr(>|t|)
p         2.6690     0.2256   11.83 1.75e-10 ***
delta    11.2956     0.6477   17.44 1.45e-13 ***
LOG10N0   7.6717     0.1269   60.44  < 2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.3601 on 20 degrees of freedom

Number of iterations to convergence: 7
Achieved convergence tolerance: 6.176e-06

------
Residual sum of squares: 2.59

------
Asymptotic confidence interval:
            2.5%     97.5%
p       2.198435  3.139585
delta   9.944589 12.646691
LOG10N0 7.406874  7.936456

------
Correlation matrix:
                 p      delta    LOG10N0
p        1.0000000  0.9687776 -0.6264922
delta    0.9687776  1.0000000 -0.7327828
LOG10N0 -0.6264922 -0.7327828  1.0000000
```

2 Analysis of residuals

2.1 Graphics of residuals

Several plots may help to check the validity of the assumptions of the error model based on the analysis of residuals. The plot of the result of the function `nlsResiduals` provides, by default, four classic plots of residuals (see following example): non-transformed residuals against fitted values, standardized residuals against fitted values, auto-correlation plot of residuals (i+1th residual against ith residual), and qq-plot of the residuals. See `?nlsResiduals` to have more details or view other possibilities.

```
> resmaf<-nlsResiduals(nlsmaf)
> plot(resmaf)
```



2.2 Residuals tests

The normality of residuals may be tested by the Shapiro-Wilk test and their independence by the runs test using the function `test.nlsResiduals` as below.

```
> test.nlsResiduals(resmaf)

------
        Shapiro-Wilk normality test

data:  stdres
W = 0.968, p-value = 0.6407


------
        Runs Test

data:  as.factor(run)
Standard Normal = -0.2045, p-value = 0.8379
alternative hypothesis: two.sided
```

3 Confidence region

The package `nltools` provides two different methods for the representation of $1-\alpha$ confidence region for model parameters as defined by Beale [1, 2]:

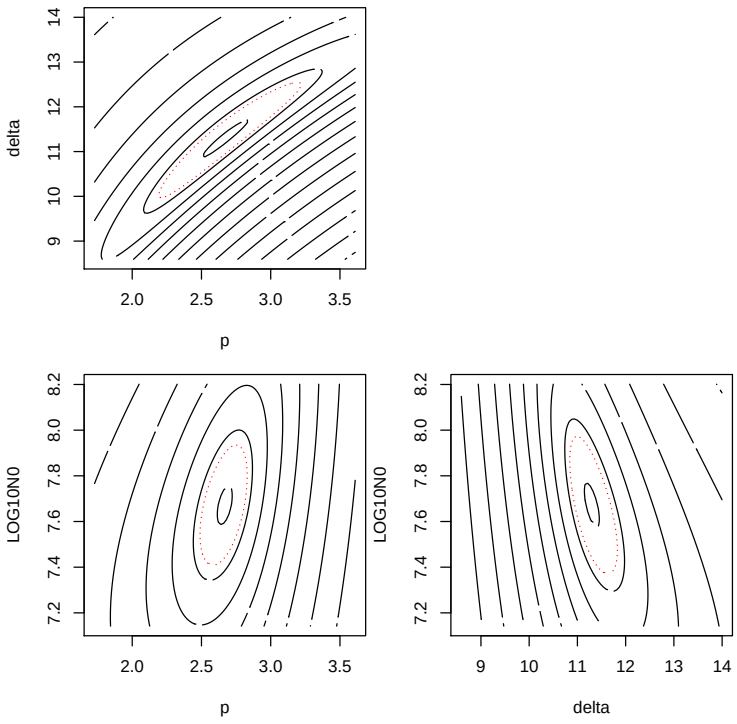
$$SCE(\theta) < SCE_{min}\left[1 + \frac{p}{n-p}F_{1-\alpha}(p, n-p)\right]$$

The function `nlsContourRSS` provides sections of that confidence region on each plane defined by two parameters, while the function `nlsConfRegions` provides projections of that region on the same planes.

3.1 Residual sum of squares contours or likelihood contours

The function `nlsContourRSS` enables to plot the Residual Sum of Squares (RSS) contours which also correspond to the likelihood contours for a Gaussian model. One of these contours, plotted in red, corresponds to the section of the 95 percent Beale's confidence region in each plane of two parameters. The argument `nlev` corresponds to the number of contours to be plotted, in addition to the red one.

```
> contmaf <- nlsContourRSS(nlsmaf)

> plot(contmaf, col=FALSE, nlev=10)
```



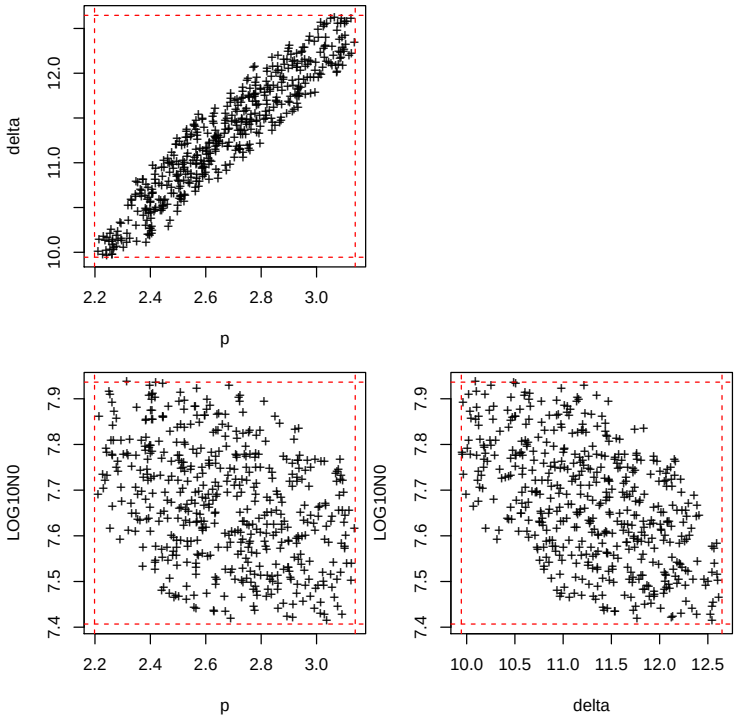
3.2 Projections of the 95 percent Beale's confidence region

A second method is proposed to represent the 95 percent Beale's confidence region, also named the joint parameter likelihood region [1]. This method first requires to sample points belonging to this region. The method consists in randomly sampling the parameter values in a hypercube centered on the least squares estimate and to accept only the sampled values whose residual sum of squares verify the Beale criterion. As soon as the specified number of points to draw in the confidence region is reached (corresponding to the argument `length` of the function `nlsConfRegions`), the iterative sampling is stopped. The algorithm does converge to the confidence region in a reasonable time only if the hypercube defined for sampling is not to small, in order to contain the whole confidence region, but also not too big, so that the probability of a sampling point to be in the confidence region is not too small.

The confidence region is then plotted by projection of the sampled points in each plane defined by a couple of parameters. The bounds of the hypercube in which random values of parameters are drawn may be plotted in order to check if the true confidence region is totally included in the hypercube defined by default. If not, the hypercube should be expanded (by increasing the argument `exp`, fixed by default at 1.5) in order to obtain the full confidence region. It is often necessary to make two or three trials for adjusting the value of the argument `exp` in order to obtain enough points in the whole confidence region in a reasonable time, as in the next example.

In this example, the function `nlsConfRegions` is first called with the argument `exp` fixed at 1, which corresponds to a hypercube delimited by the limits of the asymptotic confidence intervals of each parameter. The obtained region is then plotted, fixing the argument `bounds` at `TRUE` in order to visualize the sampling hypercube.

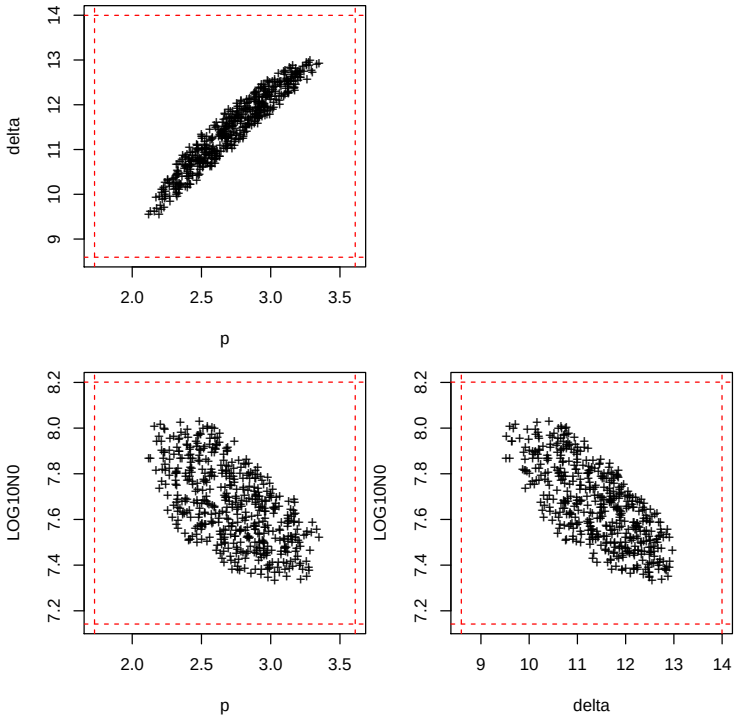
```
> rcmaf <- nlsConfRegions(nlsmaf, length=500, exp=1)

> plot(rcmaf, bounds=T)
```



Since the whole region does not seem to be included in the sampling hypercube, the function `nlsConfRegions` is recalled with the argument `exp` fixed at 2.

```
> rcmaf <- nlsConfRegions(nlsmaf, length=500, exp=2)
```

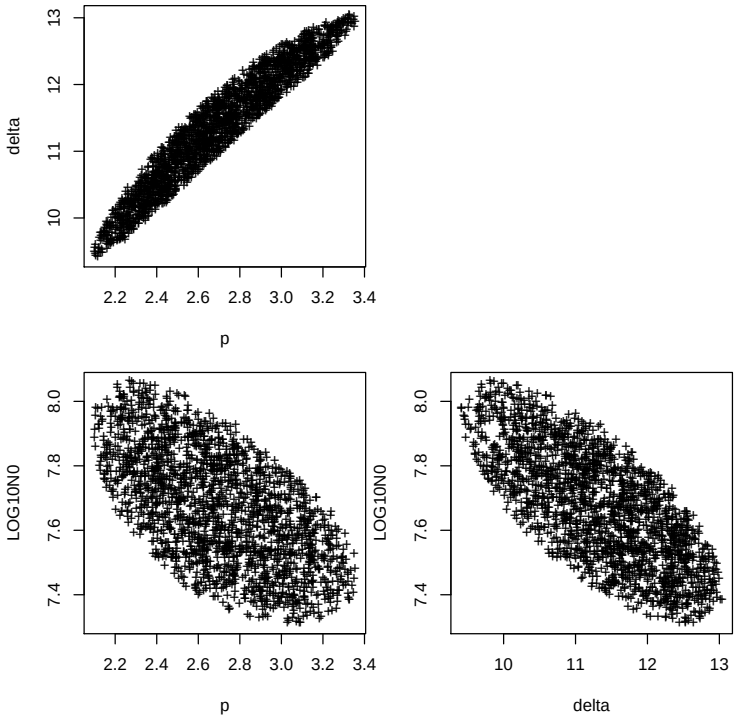
```
> plot(rcmaf,bounds=T)
```



This value of the argument `exp` seems reasonable. The function `nlsConfRegions` may be recalled specifying a greater number of points drawn in the region (argument `length`), and the region may be plotted without showing the bounds of the hypercube.

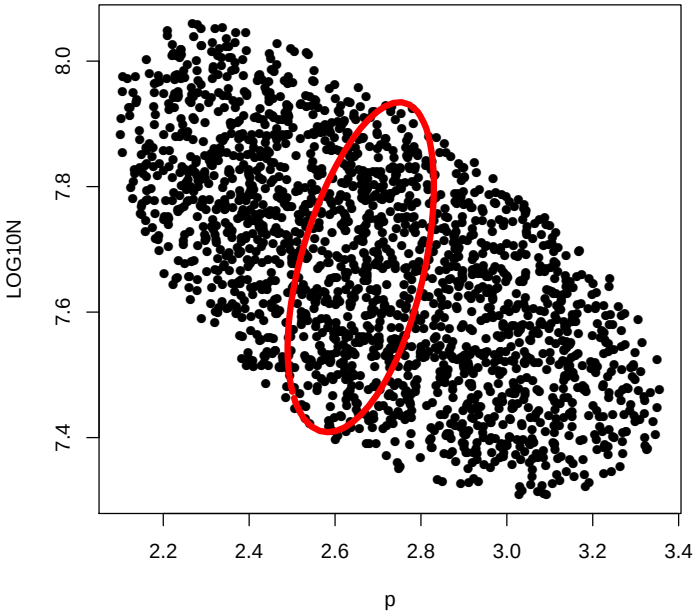
```
> rcmaf <- nlsConfRegions(nlsmaf, length=2000, exp=2)
```

```
> plot(rcmaf, bounds=F)
```



3.3 Comparison of both representations of the confidence region

It must be noticed that the representation of the confidence region in sections using `nlsContourRSS` does not give the same information as its representation by projections using `nlsConfRegions` when the number of parameters is greater than two. Sections of multidimensional (at least 3D) objects are smaller than projections and the representation of a confidence region in sections tends to underestimate its size in comparison to its representation by projections. The perception of structural correlations between parameters from graphical representation of the confidence region may also be very different between both representations. Let us see below such a difference with the previous example, for which both the section and the projection of the confidence region was plotted on the same plane.



We recommend to use the representation of RSS contours to detect non linearities and/or the presence of more than one minimum in the neighbourhood of the estimate, and the representation of projections of the confidence region to evaluate the uncertainty on the estimated parameters and to judge of their structural correlations.

4 Resampling

4.1 Jackknife

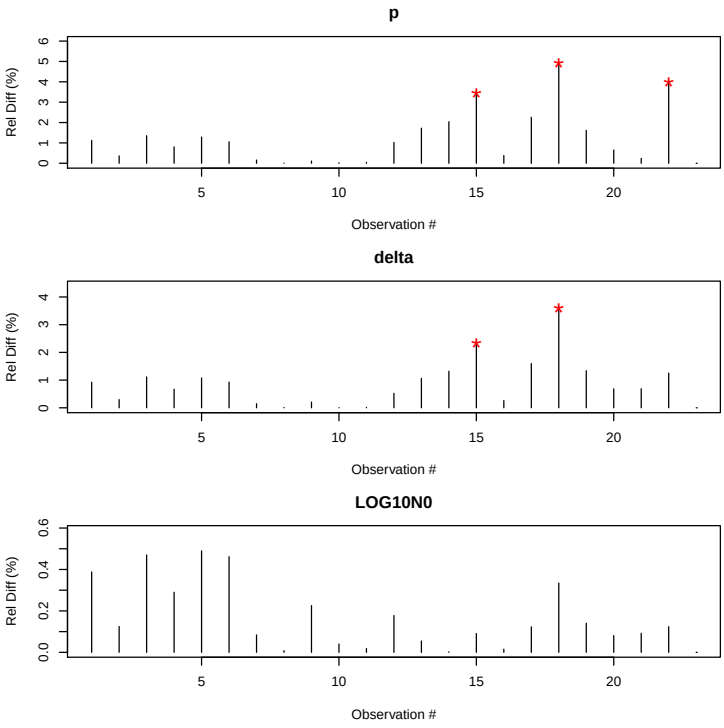
The function `nlsJack` may be used to obtain, for a data set with n observations, n resampled data sets of $n-1$ observations and the n corresponding new estimations for each parameter of the model. The jackknife estimates with confidence intervals are then calculated as described by Seber and Wild [3].

```
> jackmaf <- nlsJack(nlsmaf)

> summary(jackmaf)

------
Jackknife estimates
        p     delta   LOG10N0
 2.632607 11.260269  7.669289

------
Jackknife confidence intervals
             Low        Up
```

```
p       2.155147  3.110067
delta   9.916903 12.603636
LOG10N0 7.497482  7.841096

------
Influential values
* Observation 15 is influential on p
* Observation 18 is influential on p
* Observation 22 is influential on p
* Observation 15 is influential on delta
* Observation 18 is influential on delta
```

The leave-one-out procedure may also be employed to assess the influence of each observation on each parameter estimate, as in the end of the previous summary or in the following representation.

```
> plot(jackmaf)
```



4.2 Bootstrap

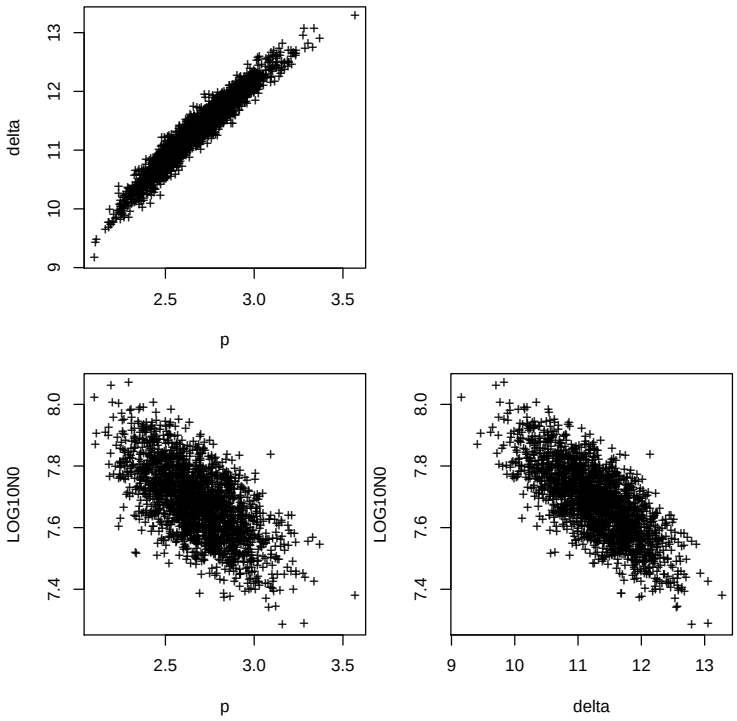
The function `nlsBoot` uses non-parametric bootstrapping of mean centered residuals [3] to obtain a chosen number (argument `niter`) of bootstrap estimates. Ponctual estimations (resp. confidence intervals) of parameters are then provided using medians (resp. the 2.5 and 97.5 percentiles) of the bootstrap sample of estimates.

```
> boomaf <- nlsBoot(nlsmaf, niter=2000)
```

```
> summary(boomaf)

------
Bootstrap estimates
        p     delta   LOG10N0
 2.673325 11.298143  7.676025

------
Bootstrap confidence intervals
             2.5%     97.5%
p        2.285030  3.103516
delta   10.084606 12.421245
LOG10N0  7.446597  7.910091
```

The boostrap sample of estimates may be visualized by projection on each plane defined by a couple of parameters, as below. This representation is generally close to the representation of the 95 percent Beale's confidence region provided by `nlsConfRegions`.

```
> plot(boomaf, type="pairs")
```



References

[1] Bates˜D.M. and Watts˜D.G. (1988) Nonlinear regression analysis and its applications. Wiley, Chichester, UK.

[2] Beale˜E.M.L. (1960) Confidence regions in non-linear estimations. *Journal of the Royal Statistical Society*, 22B, 41-88.

[3] Seber˜G.A.F., Wild˜C.J. (1989) Nonlinear regression. Wiley, New York.

[4] Huet˜S., Bouvier˜A., Poursat˜M.A., Jolivet˜E. (2003) Statistical tools for nonlinear regression: a practical guide with S-PLUS and R examples. Springer, Berlin, Heidelberg, New York.