

Object oriented S-Plus functions for high breakdown point and high efficiency regression with test for bias

Douglas B. Clarkson and Alfio Marazzi
November 1992, Revised March 1997

1 Introductory example

The S-Plus distribution media includes the so called “stack loss” data set first described by Brownlee (1960). This data set has been analyzed by a large number of statisticians. An amusing history of these data is given by Dodge (1996), who reports on 60 published analyses. The “stack loss” in this data is the percent loss (times 10) of ammonia during 21 days of operation. The ammonia is lost during the process of producing nitric acid by dissolving the ammonia in water. Three variables – air flow, x_1 , water temperature, x_2 , and acid concentration, x_3 – may influence the loss of ammonia, y . It is of great interest to model y by means of x_1 , x_2 and x_3 . To display the stack loss response variable, type

```
> stack.loss
```

and to display the stack loss covariates, type

```
> stack.x
```

The function `lm` can be used to fit the linear model

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + e,$$

where e represent an error term. Let

```
> Stack <- data.frame(Loss=stack.loss,stack.x)
```

be a data frame constructed from the stack loss data. Then the least squares fit is given by

```
> Fit <- lm(Loss~Air.Flow+Water.Temp+Acid.Conc.,data=Stack)
```

The fitted model is

$$\hat{y} = -39.91967 + 0.7156402x_1 + 1.295286x_2 - 0.1521225x_3,$$

with a residual standard error of 3.243364. To obtain the standardized residuals type

```
> Std.err <- summary(Fit)$sigma
> Std.res <- residuals(Fit)/Std.err
> Std.res
```

No value fall below -2.5 or above 2.5 ; thus, one could conclude that no point in this data set is armful. However, let us examine an index plot of residuals by typing

```
> win.graph()
> plot(Std.res,ylim=c(-3,3))
> abline(h=-2.5); abline(h=2.5)
```

We remove the 4 observations (1, 3, 4 and 21) with the largest residuals and recompute the least squares fit:

```
> stack <- Stack[-c(1,3,4,21),]
> fit <- update(Fit,data=stack)
```

We obtain

$$\hat{y} = -37.65246 + 0.7976856x_1 + 0.5773405x_2 - 0.06706018x_3,$$

with a residual standard error of 1.252714. There are important differences in the coefficients and in the residual standard error. To examine the standardized residuals of `fit` on the complete data set type

```
> pred <- predict(fit,Stack)
> res <- stack.loss-pred
> std.err <-summary(fit)$sigma
> std.res <- res/std.err
> plot(std.res,ylim=c(-7,7))
> abline(h=-2.5); abline(h=2.5)
```

The points 1, 3, 4 and 21 fall now outside the band and can be recognized as outliers. However, although the analysis required a considerable effort, several questions remain: Which fit should be preferred ? Which standard error ? Is there a quicker way to detect the outliers and provide a good fit to the bulk of the data ?

A robust method proposed by Yohai, Stahel, and Zamar (1991) has been implemented into S-Plus in order to facilitate the analysis. To get the robust fit, you need to load two libraries:

```
> library(robeth,T)
> library(zysreg,T)
```

Then type:

```
> Robust <- lm(Loss~Air.Flow+Water.Temp+Acid.Conc.,
               data=Stack, method="robust", level=0.85)
```

It is very likely that the program gives a warning message at this point. This message will be explained later. The robust fit may be graphically displayed using the generic `plot()` function, but in order to obtain the new index residual plot type

```
> plot(resid(Robust)/Robust$scale)
> abline(h=-2.5);abline(h=2.5)
```

The plot shows that the robust regression technique is able to point out the outliers in one single blow.

To look at the robust fit type

```
> summary(Robust)
```

and note that the results returned by the “robust” algorithm depend upon an “initial estimate”, a “final estimate” and a “test for bias” explained later. Also, because the initial estimate used in the fitting algorithm is based upon a random algorithm, the estimates you obtain from the S-Plus command above will probably differ slightly from those obtained here. You can make comparisons between the least squares and the robust fits by first using the function `fits.compare` to create an object of the class “fits.compare”, and then using `plot` and `print` on this object. Here’s how you would do it in the case of the stack loss data: type

```
> compar <- fits.compare(Fit, fit, Robust)
```

To obtain a visual comparison of the fits the `plot()` method is used:

```
> plot(compar)
```

Statisticians use the concept of breakdown point – the highest percentage of outliers that will not completely distort the estimate – as one of the properties of a robust estimation procedure. The larger the breakdown point the better, generally, up to the maximal breakdown point of 50 percent. A single outlier can cause least squares estimates to become extremely biased. The breakdown point of least squares estimates is zero. The initial estimate of the Yohai–Stahel–Zamar procedure has a breakdown point close to 50%; it fits the majority of the data and set apart the outliers, even if their percentage is very high. Unfortunately, it performs poorly from the point of view of efficiency. This means that, if the data were distributed according to the usual Gaussian model, the estimated coefficients would have a large variance. For this reason, the Yohai–Stahel–Zamar procedure also computes a final estimate with high efficiency. But, although it can be proved that the breakdown point of the final estimate is still close to 50%, its bias (i.e. the distortion due to the outliers) can be high. In order to reach a conclusion, the Yohai–Stahel–Zamar procedure includes a test for bias. If the test is not significant, the user is advised to choose the final fit and to base inference on the covariance matrix of its coefficients. Otherwise, the bias may be too high and inference based on the final estimate is not safe. In this case, the initial estimate should be used as an exploratory estimate: In other words, outliers should be investigated deeply and the possibility of modifying the model should be considered.

In order to reject the null hypothesis of no bias, the optional parameter `level` (the significance level to reject the null hypothesis) has been set to 0.85 in the stack loss data example. You probably got a warning message from `lm(...,method="robust")`. The test for bias was significant and, thus, `summary(Robust)` provided the initial fit. If you type

```
> robust <- update(Robust,data=stack)
```

then no warning is given. The final estimate is returned, which is quite close to the least square estimate after removal of the outliers.

Section 2 describes the Yohai–Stahel–Zamar procedure in details. Section 3 gives a complete description of the corresponding S-Plus functions and Section 4 completes the stackloss data example.

2 The Robust Regression Estimates

Let $X = (x_{ij})$ denote a real $n \times p$ matrix, $y = (y_1, \dots, y_n)^\top$ denote an observed n -vector of responses, and $\theta = (\theta_1, \dots, \theta_p)^\top$ denote an unknown p -vector of coefficients to be estimated. We consider the usual linear model

$$y = X\theta + e,$$

where $e = (e_1, \dots, e_n)^\top$ is an n -vector of unknown errors. It is assumed that, for given X , the components e_i of e are independent and identically distributed according to a distribution $L(\cdot/\sigma)$, where σ is an unknown scale parameter. Denote the i th row of the matrix X by x_i^\top .

2.1 Estimation and test procedure

The basic idea in Yohai, Stahel, and Zamar (1991) is to compute S-estimates of θ and σ , and then refine them via an MM-estimation procedure. If the MM-estimates are not too different from the S-estimates (a test for this is provided), then the MM-estimates are used; otherwise, the S-estimates are preferred. MM-estimates are desirable because of their high asymptotic efficiency. Because they are computed using the initial S-estimates, they will also have a high breakdown point (but notice that they are accepted only if they are not too different from the MM-estimates).

The Yohai–Stahel–Zamar estimates are computed as follows: For constant k define the bisquare function as

$$\chi_k(s) := \begin{cases} 3(s/k)^2 - 3(s/k)^4 + (s/k)^6 & \text{if } |s| \leq k \\ 1 & \text{if } |s| > k, \end{cases}$$

and perform the following steps.

1. Compute the initial coefficient and scale estimates with high breakdown point and low bias as follows:
 - (a) Calculate a resampling approximation θ^* of the S-estimate $\theta^{(0)}$ defined by

$$\theta^{(0)} = \operatorname{argmin}_\theta S_{k_0}(\theta),$$

where $S_{k_0}(\theta)$ is the solution S of

$$\frac{1}{n-p} \sum_{i=1}^n \chi_{k_0} \left(\frac{y_i - x_i^\top \theta}{S} \right) = \beta.$$

Here $k_0 = 1.548$, $\beta = \int \chi_{k_0}(s) d\Phi(s) = 0.5$, $\Phi(\cdot)$ denote the standard cumulative normal distribution, and $\operatorname{argmin}_x f(x)$ is the value of x minimizing the function $f(x)$. Notice that $S_{k_0}(\theta)$ is implicitly defined.

The resampling approximation θ^* is computed by repeatedly selecting at random p row vectors from X and the corresponding y , computing the estimate θ' that fits exactly these observations, and then finding the value S which satisfies the equation

$$\frac{1}{n-p} \sum_{i=1}^n \chi_{k_0} \left(\frac{y_i - x_i^T \theta'}{S} \right) = \beta.$$

The θ' which minimizes the value of S over the repeated samples of size p is defined as the resampling estimate θ^* . Because the left hand side of this equation is a decreasing function of S , S needs only be computed if

$$\frac{1}{n-p} \sum_{i=1}^n \chi_{k_0} \left(\frac{y_i - x_i^T \theta'}{S^*} \right) < \beta,$$

where S^* is the current minimal value for S .

- (b) Once the resampling approximation θ^* has been obtained, the S-estimate is computed by refining θ^* . An optimization algorithm is used to obtain $\theta^{(0)}$ as a local minimum of S such that $S(\theta^{(0)}) \leq S(\theta^*)$.

Let $\sigma^{(0)} = S(\theta^{(0)})$ be the estimate of σ corresponding to $\theta^{(0)}$, and let $r_i^{(0)} = y_i - x_i^T \theta^{(0)}$ for $i = 1, \dots, n$. The estimates $\theta^{(0)}$ and $\sigma^{(0)}$ are called *initial estimates*.

Note. The resampling approximation is necessary because the function to be minimized, $S_{k_0}(\theta)$, can have many local minima. The resampling approximation is the initial value for the iterative refinement algorithm, which hopefully returns the optimal estimate.

2. Compute the final coefficient estimate with high efficiency.

Compute the MM-estimate $\theta^{(1)}$ as the local minimum of

$$Q_{MM}(\theta) = \sum_{i=1}^n \rho \left(\frac{y_i - x_i^T \theta}{\sigma^{(0)}} \right),$$

such that

$$Q_{MM}(\theta^{(1)}) \leq Q_{MM}(\theta^{(0)}).$$

Here $\rho(\cdot) = \chi_{k_1}(\cdot)$ and $k_1 = 4.687$. With this value of k_1 , the MM-estimate has a relative efficiency of 95% (with respect to the least squares estimate at the Gaussian model). Moreover, because $Q_{MM}(\cdot)$ is decreased, the MM-estimate inherits the high breakdown point of the S-estimate. Let $r_i^{(1)} = y_i - x_i^T \theta^{(1)}$ for $i = 1, \dots, n$.

3. Compute the final scale estimate, $\sigma^{(1)}$, as the solution of

$$\frac{1}{n-p} \sum_{i=1}^n \chi_{k_0} \left(\frac{r_i^{(1)}}{\sigma} \right) = 0.5.$$

The estimates $\theta^{(1)}$ and $\sigma^{(1)}$ are called the *final estimates*.

4. Compute the test statistic for the test for bias.

If the MM-estimates maintain the low bias of the S-estimates, then the scale estimate for the MM-estimates cannot change too much (since, if the scale changes dramatically, it signifies a large shift in the regression plane). The test statistic T used to test for bias in the MM-estimates is a studentized version of a test that the difference between the spread of residuals obtained from $\theta^{(1)}$ and those obtained from the initial estimate $\theta^{(0)}$ is zero.

Let $\psi_{k_0}(\cdot) = \chi'_{k_0}(\cdot)$ and $\psi_{k_1}(\cdot) = \chi'_{k_1}(\cdot)$, where $'$ denotes the derivative with respect to the argument, and calculate

$$\begin{aligned}\tilde{r}_i &:= r_i^{(0)} / \sigma^{(0)} && \text{for } i = 1, \dots, n, \\ v_0 &:= \frac{(1/n) \sum \psi'_{k_0}(\tilde{r}_i)}{(\sigma^{(0)}/n) \sum \psi_{k_0}(\tilde{r}_i) \tilde{r}_i}, \\ \psi_i^{(0)} &:= \frac{\psi_{k_0}(\tilde{r}_i)}{(1/n) \sum \psi'_{k_0}(\tilde{r}_i)} && \text{for } i = 1, \dots, n, \\ \psi_i^{(1)} &:= \frac{\psi_{k_1}(\tilde{r}_i)}{(1/n) \sum \psi'_{k_1}(\tilde{r}_i)} && \text{for } i = 1, \dots, n, \\ d^2 &:= \frac{1}{n} \sum (\psi_i^{(1)} - \psi_i^{(0)})^2, \\ T &:= \frac{2n(\sigma^{(1)} - \sigma^{(0)})}{v_0 d^2 \sigma^{(0)^2}.\end{aligned}$$

Standard asymptotic theory shows that T approximately follows a χ^2 -distribution with p degrees of freedom. If T is larger than the α quantile χ^2_α (a typical α is 0.95) of the χ^2 -distribution with p degrees of freedom, then the estimation procedure gives a warning that the bias may be unacceptably high so that inference based on the final MM-estimates is dangerous. In this case, the initial estimates $\theta^{(0)}$ and $\sigma^{(0)}$ are returned and can be used as exploratory estimates. Otherwise, the bias is not high, and $\theta^{(1)}$ is returned with $\sigma^{(0)}$ as the scale estimate. In this case inference is possible.

Note. The standardized residuals \tilde{r}_i are used because they are computed on the basis of the S-estimates and should thus be less sensitive to outliers. The weights in the covariance matrix estimation are also based upon these residuals.

2.2 Covariance matrix of the estimated coefficients

Estimates of the asymptotic covariance matrices of the estimated initial and final coefficients are computed as follows:

1. To compute the estimated covariance matrix of $\theta^{(0)}$ set $k = k_0$; To compute the estimated covariance matrix of $\theta^{(1)}$ set $k = k_1$.
2. For $i = 1, \dots, n$ set $w_i := \psi_k(\tilde{r}_i) / \tilde{r}_i$ if $\tilde{r}_i \neq 0$ and $w_i := 0$ otherwise.
3. Set $\tilde{X} := (1/\sqrt{\bar{w}}) \text{diag}(\sqrt{w_i}) X$, where $\bar{w} := (1/n) \sum_{i=1}^n w_i$.
4. Set

$$\hat{C} := \sigma^{(0)^2} \hat{f}_H(\tilde{X}^T \tilde{X})^{-1},$$

where \hat{f}_H is the coefficient defined in Huber (1981, p. 173, formulas (6.5) and (6.8)) based on the standardized residuals \tilde{r}_i and on $\psi = \psi_k$. It is computed as

$$\hat{f}_H = \frac{n \sum_i \psi_k^2(\tilde{r}_i)}{(\sum_i \psi_{k_1}'(\tilde{r}_i))^2}.$$

2.3 Proportion of explained variation R^*

This measure corresponds (and reduces) to the R^2 in the classical case. We define it for the S-estimate $\theta^{(0)}$ and for the MM-estimate $\theta^{(1)}$.

R^ for the S-estimate $\theta^{(0)}$*

If an intercept is in the model, set

$$R^* = \frac{(n-1)s_y^2 - (n-p)s_e^2}{(n-1)s_y^2},$$

where $s_e = \sigma^{(0)}$ and s_y is the robust scale estimate obtained when the model only includes the intercept term. Use the location S-estimate t^* defined by

$$t^* = \operatorname{argmin}_t s(t),$$

where $s(t)$ is the solution of

$$\frac{1}{n-1} \sum_{i=1}^n \chi_{k_0} \left(\frac{y_i - t}{s} \right) = 0.5,$$

and set $s_y = s(t^*)$. If there is no intercept, set $t^* = 0$ and replace $(n-1)s_y^2$ by ns_y^2 in the definition of R^* .

R^ for the MM-estimate $\theta^{(1)}$*

If there is an intercept term, then compute the location MM-estimate \hat{t} corresponding to the local minimum of

$$Q_y(t) = \sum_{i=1}^n \rho \left(\frac{y_i - t}{\sigma^{(0)}} \right),$$

such that

$$Q_y(\hat{t}) \leq Q_y(t^*),$$

where $\rho(\cdot) = \chi_{k_1}(\cdot)$; otherwise set $\hat{t} = 0$. Set

$$R^* = \frac{\sum \rho((y_i - \hat{t})/\sigma^{(0)}) - \sum \rho((y_i - x_i^T \theta^{(1)})/\sigma^{(0)})}{\sum \rho((y_i - \hat{t})/\sigma^{(0)})}.$$

2.4 Deviance D^*

Define the deviance D^* as the optimal value of the objective function on the σ^2 -scale. More precisely let:

$$\begin{aligned} D^* &= S(\theta^{(0)})^2 = (\sigma^{(0)})^2 && \text{for the S-estimate } \theta^{(0)} \\ D^* &= 2 \cdot (\sigma^{(0)})^2 \sum \rho((y_i - x_i^T \theta^{(1)})/\sigma^{(0)}) && \text{for the MM-estimate } \theta^{(1)} \end{aligned}$$

where $\rho(\cdot) = \chi_{k_1}(\cdot)$. These definitions coincide with the residual sum of squares in the classical case ($\rho(s) = \chi(s) = s^2/2$).

2.5 Test of a linear hypothesis

If the bias is not high, approximate inference based on the final MM-estimate is possible. In order to test a linear hypothesis two options are available: the τ -test and the R_n^2 -test described in Hampel et al. (1986, Chapter 7). Consider the p parameter model

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

let a linear hypothesis be expressed in the canonical form

$$\mathcal{H}_0 : \theta_{q+1} = \theta_{q+2} = \dots = \theta_p, \quad 0 < q < p,$$

and denote by ω the submodel of Ω obtained by imposing the condition \mathcal{H}_0 . Denote by $\hat{\theta}_\Omega$, resp. $\hat{\theta}_\omega$, the final MM-estimate of θ in the model Ω , resp. ω and let $\hat{\sigma}_\Omega$ be the initial estimate of σ under Ω . The τ -test statistic for testing \mathcal{H}_0 is defined as

$$F_\tau = \frac{2}{p-q} \sum_{i=1}^n (\chi_{k_1}(r_{\omega,i}/\hat{\sigma}_\Omega) - \chi_{k_1}(r_{\Omega,i}/\hat{\sigma}_\Omega)),$$

where $r_{\omega,i} = y_i - x_i^T \hat{\theta}_\omega$ and $r_{\Omega,i} = y_i - x_i^T \hat{\theta}_\Omega$. The null distribution of F_τ can be approximately evaluated as the distribution of

$$\left[\int \psi_{k_1}^2(s) d\Phi(s) / \int \psi_{k_1}'(s) d\Phi(s) \right] Z_{p-q},$$

where Z_{p-q} is a χ^2 -distributed random variable with $p-q$ degrees of freedom and Φ denotes the standard cumulative Gaussian distribution function.

Let K_{22} be the $(p-q) \times (p-q)$ lower right block of the asymptotic covariance matrix of the final MM-estimate $\hat{\theta}_\Omega$. The R_n^2 -test statistic is defined by

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

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

3 S-Plus Functions

S-Plus provides two types of functions for the Yohai–Stahel–Zamar procedure:

- (a) Estimation functions for the numerical computations of the estimates;
- (b) Interface functions for the use of the estimation functions within the object oriented paradigm of S-Plus, 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 `zysreg()`, which is used by `lm()` to fit the Yohai–Stahel–Zamar estimates. Alternatively, `zysreg()` can be called directly. `zysreg()` returns a list containing the estimates (see `help(zysreg)` for more details). Computations are performed using routines from the ROBETH (Marazzi, 1993) library of FORTRAN routines for robust estimation as follows:

1. The resampling approximation θ^* in step 1 is computed using the subroutine HYSEST of ROBETH (Marazzi 1993, p. 216) with samples of size p .
2. The refinement of θ^* to obtain $\theta^{(0)}$ and $\sigma^{(0)}$ (the S-estimates) is computed by means of the subroutine RYWALG of ROBETH (Marazzi 1993, p. 83).
3. The final coefficient MM-estimate $\theta^{(1)}$ are computed by means of the ROBETH subroutine RYWALG.
4. The final scale estimate $\sigma^{(1)}$ is computed by means of the subroutine RYSIGM of ROBETH (Marazzi 1993, p. 94).
5. The test for bias statistic and the corresponding probabilities are programmed in S using existing S-Plus functions and the ROBETH functions `Psi()` (ψ) and `Psp()` (ψ') (Marazzi 1993, Chapter 11, and p. 404).
6. The covariance matrices are computed using the subroutines RIMTRF, KIASCV, KFASCV and KFFACV of ROBETH (Marazzi 1993, p. 64, and pp. 150-154).

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

- `lm.robust.control()` – There are a few control parameters for the numerical algorithms. Their default values are collected in the auxiliary function `lm.robust.control()` and may be set with this function. The control structure input in `zysreg()` defaults to the list returned by `lm.robust.control()`.

Users may wish to set their own control parameters. This is easily accomplished. In either `zysreg()` or `lm()`, simply set `control=xcontrol` in the calling sequence of either function. Here `xcontrol` is any object produced by `lm.robust.control()` with the desired control parameter values. Use `help(lm.robust.control)` for additional details. In order to set the initial value of the seed parameter of the uniform random number generator, simply set `zys.seed= desired initial value` in the calling sequence of `zysreg()` or `lm()`.

The following functions are service functions:

- `pevdev0()` and `pevdev1()` — These functions are used to compute the R^* and deviance statistics corresponding to the initial and final models.
- `scovcoef()` and `ucovcoef()` — `ucovcoef()` computes the unscaled covariance matrix of the coefficients, while `scovcoef()` scales this unscaled covariance matrix, and converts it from packed symmetric matrix storage to unpacked square matrix storage.
- `comval()` and `dfcomn()` — The ROBETH routines use a few common blocks. Common blocks parameters are retrieved and modified by means of the service functions `comval()` and `dfcomn()` (Marazzi 1993, p. 405).
- The following ROBETH functions are used for numerical computations: `hysest()`, `kiascv()`, `kfascv()`, `kffacv()`, `rywalg()`, `rysigm()`, `rimtrf()`, `Psi()`, `psi()`, `Chi()`, `chi()`, `Psp()`, `psp()`, `Rho()`, and `rho()`.
- ROBETH routines use some auxiliary routines as explained in Marazzi (1993).

3.2 The object oriented interface

The function `zysreg()` computes the robust estimates and returns a list of results. These robust regression estimates are more easily produced using `method="robust"` in the `lm()` function. When `method="robust"` in `lm()`, the current “best” robust regression estimates implemented in S-Plus are computed. At this time these are the Yohai–Stahel–Zamar estimates. The value returned by `lm()` is an object with class “`lm.robust`” inheriting from class “`lm`”.

The functions `print()`, `summary()`, `plot()`, `update()`, `anova()`, as well as the access functions `coef()`, `residual()`, `fitted()`, `formula()`, and `deviance()` have been extended to objects of the class “`lm.robust`”. The following access functions are new.

- `change.robust.estimates()` — This is an access function for the coefficient estimates returned by the `zysreg()` function. It allows the user to change the level in the test for bias and obtain the resulting `zysreg()` estimates without recomputing them (see `help(change.robust.estimates)` for more details).
- `scale.estimate()`, `r.squared()`, `covar()`, `correl()`, `weights()`, and `Rank()` — These access functions extract the scale estimate, the proportion of explained variation R^* , the covariance matrix of the estimated coefficients, the corresponding correlation matrix, the weights (w_i), and the rank of the design matrix.

In addition, a function for comparing the two fits produced by the Yohai–Stahel–Zamar procedure have been added to S-Plus. These functions can also be used to compare more general fits.

- `fits.compare()` — The `fits.compare()` function accepts a sequence of objects of class “`lm`”, “`lm.robust`”, or “`aov`” (with optional names), and creates a class “`fits.compare`” object. The “`fits.compare`” object 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.

Note. The `fits.compare()` and the extractor functions also accept objects of class “`lm.Huber`” (Marazzi, 1997a), “`glm`”, and “`cubinf`” (Marazzi, 1997b). It is not recommended to compare objects with different structure.

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 `lm()` function is not yet available for objects from the class.

- `lm.fit.robust()` — Set up to call `zysreg()`. `lm.fit.robust()` is called by `lm()` when `method="robust"`. It handles some of the record keeping required by the algorithms, calls `zysreg()`, and assigns the class “`lm.robust`” (which inherits from “`lm`”) to the list of parameter estimates returned by `zysreg()`.

Note. Preprocessing before the estimation and test procedure is done by the function `lm()` which remains unchanged. `lm()` calls `lm.fit()` or `lm.wfit()` (both unchanged). `lm.fit()` and `lm.wfit()` call `lm.fits.robust()`. Observation weights, if needed, are handled correctly by the existing function `lm.wfit()`. The function `update.lm()` also remains unchanged.

- `print.lm.robust()`, `summary.lm.robust()`, `print.summary.lm.robust()`, — These functions extend the `print.lm()`, `summary.lm()`, and `print.summary.lm()`, functions to objects of class “`lm.robust`”. The major difference between the class “`lm.robust`” functions and the class “`lm`” functions is that the class “`lm.robust`” functions print the test for bias and indicate whether the initial or final estimates were selected (given the significance level).
- `plot.lm.robust()` — extends `plot()` to objects of class “`lm.robust`”.
- `plot.fits.compare()` — extends `plot()` to objects of class “`fits.compare`”.
- `anova.lm.robust()` — extends `anova()` to objects of class “`lm.robust`”.
- `deviance.lm.robust()`, `scale.estimate.lm.robust()`, `r.squared.lm.robust()`, `covar.lm.robust()`, `correl.lm.robust()`, `weights.lm.robust()`, `Rank.lm.robust()` — These functions are required by the corresponding access functions.
- `add1.lm.robust`, `drop1.lm.robust`, and `step.lm.robust` — These are function stubs used to print error messages that the associated routines are not yet implemented. Stepwise search algorithms have not yet been implemented for objects of class “`lm.robust`”.
- Other function stubs — In addition to the stepwise search functions, the class “`lm`” functions `effects`, `kappa`, `proj`, `alias`, `lm.influence`, and `lm.sensitivity` are not applicable to objects of class “`lm.robust`”. Function stubs `effects.lm.robust`, `kappa.lm.robust`, `proj.lm.robust`, `lm.influence.lm.robust`, `alias.lm.robust`, and `lm.sensitivity.lm.robust` have been added to issue warning messages for these functions.

4 Example

Continuing the introductory example, type:

```
> MM <- robust
> S <- change.robust.estimate(robust)
```

Now, `S` and `MM` contains the initial and the final estimate. You can compare them using the `fits.compare()` function:

```
> fits.compare(S, MM)
```

As the final estimate obtained with `stack` is not biased with respect to the initial estimate (the test for bias is not significant), you can make inference. For example, you can decide that `Acid.Conc.` is not necessary in the model because the corresponding t -value, -0.9881 , is not significant. In addition you can test the hypothesis that the pair of regressors `Water.Temp` and `Acid.Conc.` does not substantially contribute to explain the variation of `Loss`:

```
> robust.reduced <- update(robust, .~.-Water.Temp-Acid.Conc.)
> anova(robust.reduced, robust)
```

Remark. The final coefficient estimate $\theta^{(1)}$ can be computed as a weighted least squares estimate:

$$X^T \hat{W} X \theta^{(1)} \approx X^T \hat{W} y,$$

where $\hat{W} = \text{diag}(\hat{w}_i)$ and

$$\hat{w}_i = \frac{\psi_k \left(r_i^{(1)} / \sigma^{(0)} \right)}{r_i^{(1)} / \sigma^{(0)}},$$

with $k = 4.687$. You can verify this approximation by typing

```
> MM.res <- resid(MM)
> scale0 <- scale.estimate(S)
> dfcomn(ipsi=4,xk=4.687)
> MM.w <- Psi(MM.res/scale0)/(MM.res/scale0)
> MM.wls <- lm(Loss~Air.Flow+Water.Temp+Acid.Conc.,
               data=stack, weights=MM.w)
```

and comparing `MM.wls` to `MM`. A similar approximation holds for the initial estimate $\theta^{(0)}$:

$$X^T \check{W} X \theta^{(0)} \approx X^T \check{W} y,$$

where $\check{W} = \text{diag}(\check{w}_i)$ and

$$\check{w}_i = \frac{\psi_k \left(r_i^{(0)} / \sigma^{(0)} \right)}{r_i^{(0)} / \sigma^{(0)}},$$

with $k = 1.548$. The weights \check{w}_i are simply given by `S$M.weights`.

Acknowledgment. This work was supported by grant No. 21-36521.92. from the Swiss National Science Foundation and by a grant from the American National Science Foundation Small Business Innovation Research.

References

- Chambers J.M., Hastie T.J., Eds. (1992).
Statistical Models in S, Wadsworth & Brooks/Cole Computer Science Series, Pacific Grove.
- Dodge Y. (1996). The guinea pig of multiple regression. In Rieder H (ed.), *Robust statistics, data analysis, and computer intensive methods*, In honor of Peter Huber's 60th birthday. Springer, New York.
- Hampel F.R., Ronchetti E.M., Rousseeuw P.J., Stahel W.A. (1986).
Robust Statistics: The Approach Based on Influence Functions, Wiley, New York.
- Huber P. (1981). *Robust Statistics*, Wiley, New York.
- Marazzi A. (1993). *Algorithms, Routines and S Functions for Robust Statistics, The FORTRAN library ROBETH with an interface to S-Plus*, Chapman and Hall, New York.
- Marazzi A. (1997a). Object oriented S-plus functions for Huber type robust regression. Technical report. Inst Univ Med Soc Prev, Bugnon 17, CH-1005 Lausanne.
- Marazzi A. (1997b). Object oriented S-plus functions for discrete generalized linear models. Technical report. Inst Univ Med Soc Prev, Bugnon 17, CH-1005 Lausanne.
- Yohai V.J., Stahel W.A., Zamar R.H. (1991). A procedure for robust estimation and inference in linear regression, in Stahel W.A. and Weisberg S.W., Eds., *Directions in robust statistics and diagnostics*, Part II, Springer-Verlag, New York.