The R-Function `regr` and Package `regr0` for an Augmented Regression Analysis

Werner A. Stahel, ETH Zurich

12. August 2016

Zusammenfassung

The R function `regr` is a wrapper function that allows for fitting several different types of regression models by a unified call, and provides more informative numerical and graphical output than the traditional `summary` and `plot` methods. The package `regr0` contains the functions that go along with `regr` and a number of others. It is written to make data analysis more effective by providing user-oriented, flexible functions. It is available from `R-forge` and is still in development.

1 Introduction

Regression models are fitted in the statistical programming environment R by diverse functions, depending on the particular type of model. Outputs obtained by calling the function `summary` on the object produced by the fitting function look often quite similar. Graphical output for residual analysis is obtained by calling `plot`, but the result is not always informative.

The function `regr` allows for fitting various regression models with the same statement, provides a more informative numerical output and enhanced plots for residual analysis.

`regr` proceeds by checking arguments, then calling the suitable fitting method from standard R or other packages, collecting useful statistics from the resulting object and a call of `summary` on it and adding a few to generate an object of class `regr`.

In particular, the following models can be fitted by `regr`:

- ordinary linear models, using Least Squares or robust estimation, by calling `lm` or `lmrob` from the `robustbase` package,
- generalized linear models, by calling `glm`,
- multinomial response models, by calling `multinom` of package `nnet`,
- ordered response models, by calling `polr` of package `MASS`,
- models for survival data and Tobit regression, by calling `survreg` or `coxph` of package `survival`,
- multivariate ordinary linear models, by calling `lm`
- nonlinear models, by calling `nls`

This document presents the main features of the package `regr0` and explains the ideas behind them. It gives no details about the functions. They can be found in the help files.

The package is available from `R-forge`, e.g. by calling
`install.packages("regr0", repos="http://r-forge.r-project.org")`.

The reason why it is not on CRAN and why it is called `regr0` rather than `regr` is that the author is still developing additional features and does not yet want to guarantee upward compatibility. It also means that comments and suggestions are very welcome: `stahel stat.math.ethz.ch`

2 Numerical Output

The useful numerical output of fitting functions is usually obtained by calling `summary` on the object produced by the fitting method. This results, for most functions, in a table showing the estimated regression coefficients, their standard errors, the value of a test statistic (t or z or deviance) and, for the ordinary linear model, a p value for the tests for zero coefficients. It is followed by an overall summary, usually including a test for the hypothesis that all coefficients are zero, and a standard deviation of the error and coefficient of determination, if applicable.

If there are factors (qualitative explanatory variables) in the model, the coefficients are not always interpreted adequately, and the respective standard errors, t and p values are of little use and often misinterpreted. On the other hand, the information whether a factor has a significant effect is not available from the summary but has to be obtained by calling `drop1` on the fit object. (The function `anova`, which seems to be suited according to its name, usually does not answer this question.)

This situation cannot be called user friendly. The function `regr` is meant to provide the results that are needed without having the user call different functions and select the output that is safe to be interpreted.

Here is a result of printing a `regr` object.

```
> require(regr0)
> data(d.blast)
> r.blast <- regr(logst(tremor) ~ location + log10(distance) + log10(charge),
+     data = d.blast)
> r.blast

  Blasting for Tunnel Excavation

Call:
regr(formula = logst(tremor) ~ location + log10(distance) + log10(charge),
    data = d.blast)
Fitting function:  lm

Terms:
                  coef stcoef df signif p.symb
(Intercept)      2.971     NA  1  13.37     NA
location            NA     NA  7   3.48    ***
log10(distance) -1.506 -0.790  1 -12.13    ***
log10(charge)    0.623  0.406  1   8.17    ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1     1

St.dev.error:  0.141   on 352 degrees of freedom
Multiple R^2:  0.798    Adjusted R-squared: 0.793
```

```
F-statistic:    155   on 9 and 352 d.f.,  p.value: 1.55e-116

Coefficients for factors:
$location
  loc1     loc2     loc3     loc4     loc5     loc6     loc7     loc8
  2.07 *** 2.22 *** 2.20 *** 1.90 *** 2.04 *** 2.14 *** 2.06 *** 2.07 ***
```

2.1 Standard output for continuous explanatory variables

The "Terms:" table characterizes the effects of the individual terms in the model. For continuous explanatory variables (the last 2 lines in the example) it shows:

`coef`, the estimated value of the coefficient;

`df`, degrees of freedom, = 1 for continuous variables;

`ciLow, ciHigh`, the limits of the confidence interval;

`R2.x`, the coefficient of determination for regressing the explanatory variable in question on the other terms in the model. This is one of the wellknown collinearity diagnostics.

`signif`, a significance measure that is > 1 for estimated coefficients differing significantly from 0, see below for its definition;

`p.value`, the p value for testing if the coefficient could be zero.

`p.symb`, the usual significance symbols.

In fact, 2 more columns are contained in `rr$termtable`, but they are not printed by default:

`se`, the standard errors of the estimated coefficients;

`stcoef`, the estimated standardized coefficient, defined as `coef` times the standard deviation of the explanatory variable, divided by the standard deviation of the response (if the response is continuous as assumed here), see below for its use;

User options.
The default for the columns to be printed is set by an option stored in the `UserOptions` list and controled by the function `userOption` in the same way that the usual options are controled by the function `options`.

Significance
The usual `summary` output of fitting functions includes the t (or z) values of the coefficients as a column in the coefficients table. They are simply the ratios of the two preceding columns. Nevertheless, they provide a kind of strength of the significance of the coefficients. The p value may also serve as such a measure, but it is less intuitive as it turns tiny for important variables, making comparisons somewhat more difficult than t values. The significance of t values depends on the degrees of freedom, but informed users will know that critical values are around ± 2, and they will therefore informally compare t values to ± 2. Based on these considerations, we introduce a new measure of significance here.

The new significance measure is defined as

$$\texttt{signif} = \texttt{t value} \,/\, \texttt{critical value}$$

where `critical value` is the critical value q_{df} of the t distribution and depends on the degrees of freedom of the residuals.

Confidence Intervals.
The standard errors provided by the usual `summary` tables allow for calculating confidence intervals for continuous explanatory variables, by the formula $\texttt{coef} \pm q_{df} \cdot \texttt{std.error}$. By default, they are shown in the Terms table. If the table gets too wide, the confidence limits may be suppressed. They can then be calculated as

$$\texttt{coef} \cdot (1 \pm 1/\texttt{signif})$$

This is slightly more complicated for a calculation in the mind than `coef`± 2`se`, but the formula shows an additional interpretation of `signif` in terms of the confidence interval: If the input variable were scaled such that the confidence interval had half width 1, then the estimate would be `signif` units away from zero.

Standardized Coefficients
Standardized coefficients are meant to allow for a comparison of the importance of explanatory variables that have different variances. They are also stored in `rr$termtable` as the column `stcoef`.

```
> names(r.blast$termtable)

 [1] "coef"    "se"      "ciLow"   "ciHigh"  "df"      "teststat"  "signif"
 [8] "p.value" "p.symb"  "stcoef"  "R2.x"

> userOptions(termcolumns=c("coef", "stcoef", "df", "signif", "p.symb"))
> r.blast

  Blasting for Tunnel Excavation

Call:
regr(formula = logst(tremor) ~ location + log10(distance) + log10(charge),
    data = d.blast)
Fitting function:  lm

Terms:
                  coef stcoef df signif p.symb
```

```
(Intercept)      2.971     NA  1  13.37     NA
location            NA     NA  7   3.48    ***
log10(distance) -1.506 -0.790  1 -12.13    ***
log10(charge)    0.623  0.406  1   8.17    ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1     1

St.dev.error:  0.141   on 352 degrees of freedom
Multiple R^2:  0.798    Adjusted R-squared: 0.793
F-statistic:    155   on 9 and 352 d.f.,  p.value: 1.55e-116

Coefficients for factors:
$location
  loc1     loc2     loc3     loc4     loc5     loc6     loc7     loc8
  2.07 *** 2.22 *** 2.20 *** 1.90 *** 2.04 *** 2.14 *** 2.06 *** 2.07 ***
```

Each of them shows the effect on the response of increasing "its" carrier $X^{(j)}$ by one standard deviation, as a multiple of the response's standard deviation. This is often a more meaningful comparison of the relevance of the input variables.

Note, however, that increasing one $X^{(j)}$ without also changing others may not be possible in a given application, and therefore, an interpretation of coefficients can always be tricky. Furthermore, for binary input variables, increasing the variable by one standard deviation is impossible, since an increase can only occur from 0 to 1, and therefore, the standardized coeffient is somewhat counter-intuitive in this case.

2.2 Factors

For factors with more than two levels, (`location` in the example), there are several coefficients to be estimated. Their values depend on the scheme for generating the dummy variables characterizing the factor, which is determined by the `contrasts` option (or argument) in use. We come back to this point below ("Contrasts").

Note that for factors with only two levels, the problem does not arise, since the single coefficient can be interpreted in the straightforward manner as for continuous explanatory variables. `regr` therefore treats binary factors in the same way as continuous explanatory variables.

The test performed for factors with more than two levels, which is shown in the `Terms` table by the `p.value` entry, is the F test for the whole factor (hypothesis: all coefficients are 0). It is obtained by calling `drop1`. The significance measure is defined as

$$\texttt{signif} = \sqrt{\texttt{F value} \,/\, q_{df1,df2}}$$

where $q_{df1,df2}$ is the critical value of the F distribution. It reduces to the former one for binary factors.

The collinearity measure `R2.x` for factors is a formal generalization of `R2.x` for terms with one degree of freedom, determined by applying the relationship with the "variance inflation factor", `R2.x`$= 1/(1 - \text{vif})$ to the generalized vif. [More explanation planned.]

All coefficients for factors.
The usual contrast option `contrasts="contr.treatment"` gives the coefficients of the dummy variables a clear interpretation: they estimate the difference of the response between level k and level 1 for $k > 1$. Another popular setting is `contrasts="contr.sum"`, for which the kth coefficient estimates the effect of the kth level in such a way that the sum of all coefficients is 0. For this setting, the last of these effects is not given in the vector of coefficients, `rr$coef`.

In order to avoid ambiguities, the `regr` output lists the estimated effects for all levels of the factors after the term table. Even though the interpretation of significance and confidence intervals of the individual effects of the levels is delicate, `regr` objects include a full table, similar to the term table explained above, for each factor. By default, however, only the estimated effects ar shown, together with the "star symbols" for their significance.

Contrasts.
An advantage of `contr.sum` over the usual `contr.treatment` contrasts is that it avoids the (often unconscientious) choice of a reference level – the first level – and allows, for each level, to assess immediately how large its effect is as compared to an overall average effect. An even more important advantage appears when interactions between factors are included in the model: The main effects of one factor, including its significance, may still be interpreted (with caution) as average effects over the levels of the other factor.

The `contr.sum` setting is not well adapted to unbalanced factors, since the unweighted sum of coefficients is forced to be 0. This leads to large standard errors when one of the levels has a low frequency. The `regr0` package provides the option `contr.wsum` for which the sum of coefficients weighted with the frequencies of the levels is zero. This type of contrasts is the default in `regr0`.

2.3 Model summary

The last paragraph of the output gives the summary statistics. For ordinary linear models, the estimated standard deviation or the error term is given first. (It is labelled "Standard error of residual" in the `lm` output, which we would label a misnomer.) The `Multiple R^2` is given next, together with its "adjusted" version, followed by the overall F test for the model.

For generalized linear models, the deviance test for the model is given. If applicable, a test for overdispersion based on residual deviance is also added.

2.4 Model Comparisons

When model development is part of the statistical analysis, it is useful to compare the terms that occur in different models under consideration. There is a function called `modelTable` that collects coefficients, p values, and other useful information, and a `format` and `print` method for showing the information in a useful way.

```
> names(d.blast)

[1] "no"       "datetime" "device"   "charge"   "distance" "tremor"
[7] "location"

> r.blast2 <-
+   regr(logst(tremor) ~ ( location + log10(distance) + log10(charge) )^2,
```

```
+        data=d.blast)
> modelTable(c("r.blast","r.blast2"))

                                r.blast       r.blast2
(Intercept)                      2.9706         2.8048
location                           +++   ***      +++   ***
log10(distance)                 -1.5062  ***   -1.3548  ***
log10(charge)                    0.6226  ***    0.7507  .
location:log10(distance)            -             +++   ***
location:log10(charge)              -             +++   .
log10(distance):log10(charge)       -          -0.1085  .
.sigma.                          0.1409         0.1358
.df.                               10             25
```

2.5 Model Selection

The well-known functions `drop1` and `add1` are adapted by providing additional methods. The function `drop1` has a useful default for the argument (`scope`): It incluses all terms that can be sendsibly dropped from the model. This is not provided for the base version of `add1`. This latter function is very useful to check whether squared continuous variables or interactions between variables should be included into the model. Therefore, our version of `add1` provides this default `scope`, and both `drpp1` and `add1` calculate F tests in addition to AIC.

Since `drop1` and `add1` methods are available, `step` can be used to select a model automatically.

```
> add1(r.blast)

Single term additions

Model:
logst(tremor) ~ location + log10(distance) + log10(charge)
                              Df Sum of Sq  RSS   AIC F value  Pr(>F)
<none>                                     6.99 -1409
I(log10(distance)^2)           1     0.003 6.98 -1407    0.15   0.700
I(log10(charge)^2)             1     0.143 6.84 -1415    7.35   0.007 **
location:log10(distance)       7     0.726 6.26 -1435    5.72 2.8e-06 ***
location:log10(charge)         7     0.112 6.87 -1401    0.80   0.585
log10(distance):log10(charge)  1     0.000 6.99 -1407    0.01   0.929
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> r.step <- step(r.blast, k=5)

Start:  AIC=-1379
logst(tremor) ~ location + log10(distance) + log10(charge)

                 Df Sum of Sq   RSS   AIC F value Pr(>F)
<none>                        6.99 -1379
```

```
- location         7      3.42 10.41 -1270    24.6 <2e-16 ***
- log10(charge)    1      5.12 12.11 -1185   258.1 <2e-16 ***
- log10(distance)  1     11.30 18.29 -1036   569.5 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> formula(r.step)

logst(tremor) ~ location + log10(distance) + log10(charge)
<environment: 0x6f32798>
```

The usual stopping rule for stepwise model selection relies on the AIC criterion. Note that there is a funny argument for this choice: The former stopping rule, which aimed at keeping only formally "significant" terms in the model has been criticized because of the multiplicity problem: The retained terms appear more significant than they are, because they are the most significant of an unspecified number of candidates. Thus, the formal significance tests for the terms are liberal to an unknown degree, and the selected models tend to be too large.

Therefore, a new criterion was introduced, the AIC (or the BIC). Its justification is based on optimization of a prediction error sum of squares. It should be noted that it leads to larger models than the significance criterion. Thus, one should expect to see non-significant terms in the selected model, even based on the liberal formal tests provided in regression outputs.

Unfortunately, it is not possible to specify the stopping criterion in `step`. However, it can be shown that setting its argument `k=5`, one usually obtains the model with the formally significant terms.

3 Residual Analysis

The residual plots that are produced by plotting an `lm` or `glm` object are:

- The Tukey-Anscombe plot showing residuals against fitted values, which is certainly the most important single diagnostic tool for assessing model assumptions;
- The Scale plot, which shows square-root transformed absolute values of residuals against fitted values and helps to spot unequal variances (if these depend on the expected value of the response).
- The normal quantile-quantile plot, which makes sense only for ordinary linear models and in some cases for generalized linear models. It is not essential since skewness and outliers can also be seen in the Tukey-Anscombe plot if they are relevant;
- The leverage plot, displaying residuals against leverage values. This plot is useful to detect influential observations.

3.1 What `plot.regr` does

The plotting method for `regr` objects has many addtional features, to be described in more detail in the following subsections.

- The plots are augmented by a smooth fitted to them (which is also available in the classical R functions), and simulated smooths are added in order to assess an informal "significance" of curvature in the smooth of the data. In addition, a reference line is given, which helps finding an appropriate modification of the model if significant curvature is found.

- The set of plots that is shown by default is adjusted to the type of model. A normal QQ-plot of residuals is only shown if appropriate. If weights are used, the residuals divided by the weight are also plotted against the weights, which helps to see if the weighting rule was adequate.

- Most importantly, residuals are plotted against explanatory variables. This can also be achieved by calling `termplot` for other fit objects, but experience shows that this is often neglected by average users.

  Residuals can be plotted easily against the index (sequence number) of the observation, since this may show trends or correlations of errors in time, if the sequence reflects time.

- Finally, plotting methods are defined for models for which no useful methods are available in the basic R packages, notably ordered response regression and censored responses.

The number of plots produced by calling `plot` on a `regr` object may be elevated. The argument `plotselect` allows for requiring or deselecting any of them. The arguments to `plot.regr` are numerous and allow for varying many features, including those for which the default behavior is discussed below.

The plotting pages are marked in the right lower corner by the date and a project title and step label, if available from `options`. (This `stamp` can be suppressed by setting `options(stamp=FALSE)`.)

The default plotting character is a plus sign. Its size is adjusted to the number of points.

The plots can be generated by executing the examples on `help('plot.regr')`. A single plot is shown here for easy reference in the following descriptions.

3.2 Extreme Residuals

If there are one or a few outliers in any plot, the structure of the majority of the data becomes more difficult to grasp. Therefore, the `plot` method first flags outliers in the residuals. If there are any, the range of the vertical axis is split into an "ordinary plotting range", where the non-outliers are shown as usual, and an "outlier margin", where the outliers are shown on a highly nonlinear scale that maps the range from the limit of the ordinary range to infinity to this margin.

In order to ease identification, a suitable number of most extreme residuals are labeled with their `row.names` by default.

3.3 Smooths.

Fitting a smooth (blue dashed line in the Figure) to a residual scatterplot helps to spot a potential nonlinearity of the relationship between the response and the variable shown in the plot – the fitted values or a single explanatory variable. The smooth used by default is `loess` with a span depending on the number of observations (currently $= 5n^{-0.3}$ for ordinary regression, and twice this number for generalized linear regression; do not ask why). Such a smooth is more essential for generalized linear models than for ordinary ones, since artefacts of the residuals may make it impossible to see curved relationships from a display of the residuals.

It is difficult to decide if a smooth line is curved "significantly", such that searching for a better model should be helpful and not lead to overfitting the data. In order to help judging the "natural curvature", 19 sets of data are generated by re-randomizing the residuals. The smooths obtained for these sets are also shown (in light blue in the Figure – I hope that this can be seen in all versions of this document). If the smooth determined from the data is clearly the most extreme in some aspect, one may conclude that the model does not fit adequately. The number 19 is chosen to correspond to

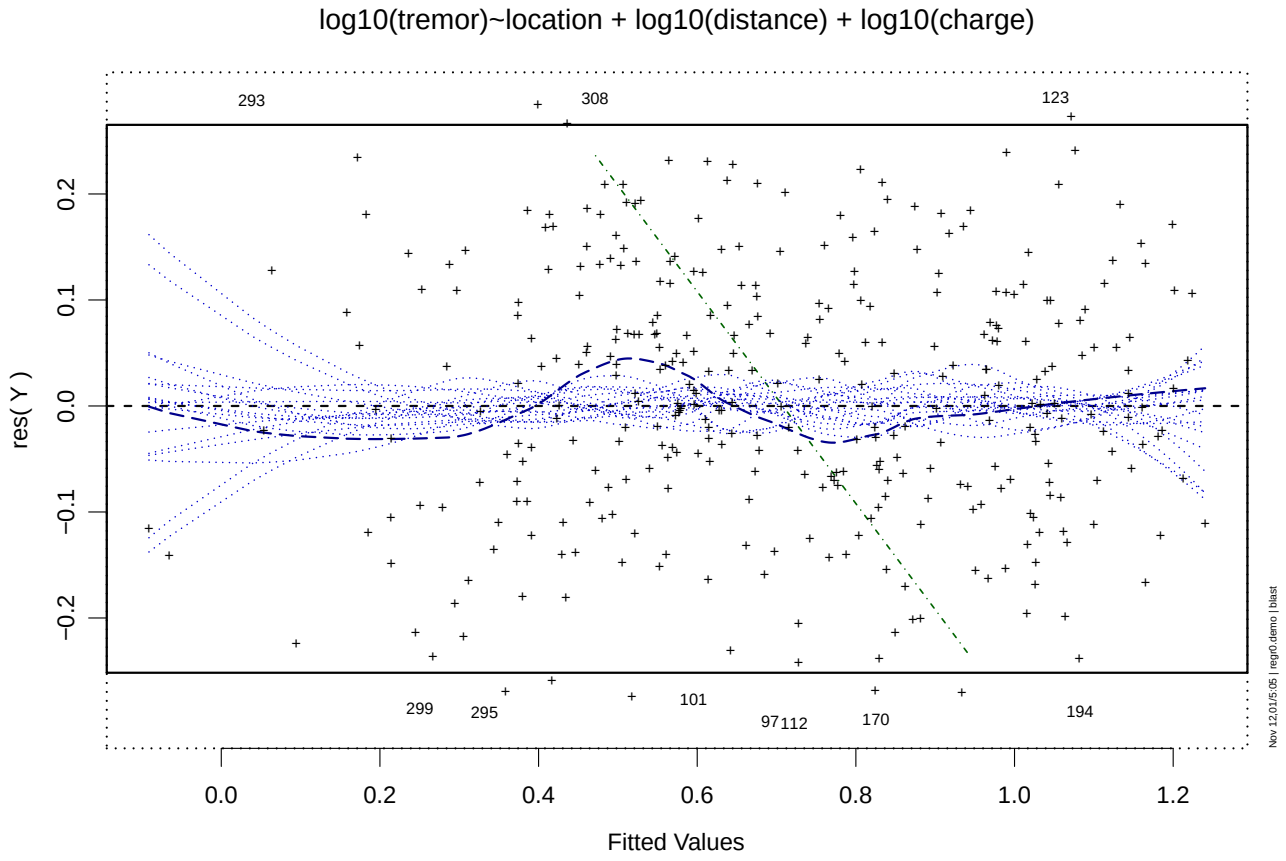

Abbildung 1: A Tukey-Anscombe plot.

a "significance level" of 5%. Up to now, such simulated datasets are only generated for ordinary linear regression.

3.4 Augmented Tukey-Anscombe Plot

The plot of residuals against fitted values is the most important diagnostic tool to assesss the validity of model assumptions. It allows for detecting deviations from linearity, homoscedasticity, symmetry and normal distribution of the random deviations.

If there is curvature, one remedy may be to transform the response variable. This can help only if the relation between fitted and response values is monotone – otherwise, quadratic terms are the most straightforward modification of the model. Monotonicity is easy to see if the response is plotted on the vertical axis instead of the residuals. This display can also be requested from `plot.regr` (by setting `plotselect=c(yfit=3)`).

In order to avoid the necessity to either call `plot` again or to ask for the plot of response against fitted values routinely, a **reference line** (green dot-dashed line in the Figure) is shown in the Tukey-Anscombe plot. It connects points with equal response values. Since the response can be determined by adding the two coordinates of the plot – fitted plus residual – lines of equal response values have slope -1. The reference line shown is such a line, the one going through the center of the points in the plot. If the smooth never gets steeper than the reference line, then the suggested relation between fitted

and response values is monotone, and a transformation of the response may remove the non-linearity.

One might argue that a plot of the response on the fit is more straightforwardly interpreted than the Tukey-Anscombe plot with the reference line. We think that the Tukey-Anscombe plot can show the deviations from model assumptions more clearly and should therefore be preferred, and that the reference line, after a short learning period, will be easy enough to work with and helps avoiding an additional display. Of course, some users may disagree.

3.5 Scale Plot

The absolute values of the *standardized* residuals are plotted against the fitted values in order to see more clearly than in the Tukey-Anscombe plot whether the scatter of residuals depends on the expected values. The plot shown when plotting `lm` objects uses square roots of absolute residuals instead of the absolute values themselves because they are more symmetrically distributed. This appears as a technical argument to me and it may confuse unexperienced users. While square roots are therefore avoided for plotting, they are still used to calculate the smooth that is added to the plot.

When the smooth line in the Tukey-Anscombe plot indicates a curved relationship, the residuals contain a bias, and deviations from a straight line in the scale plot are not necessarily an indication of heteroscedasticity. On the other hand, clear differences in the scatter around the smooth line in the Tukey-Anscombe plot may not show up in the scale plot as a consequence of the bias. Therefore, `plot.regr` uses modified residuals to generate the scale plot: The difference between the residual and the smooth fit in the TA plot is this modified residual, of which the absolute value is used.

3.6 QQ-Plot

The QQ-normal plot of residuals only makes sense in the ordinary linear or nonlinear regression. It is therefore avoided by default for other models (unlike in standard R). Furthermore, this plot only makes sense if quantities with the same normal distribution are used to generate it. Therefore, standardized residuals should be used. The preceding reasoning for using modified residuals also applies. In fact, these modified residuals are modified again to reflect the potentially different scales: they are divided by the value of the smooth in the scale plot before they are used for the QQ-plot.

3.7 Weights

If weights are given by the user, they should usually reflect different variances of the random errors E_i, and the residuals, multiplied by the square root of the weights, should have constant variance. The standardized absolute residuals are therefore plotted against the weights. If the size of these residuals is not constant but depends on the weight, the rule or weighting function should be revised.

If weights are given, they are also used as the sizes of the plotting symbols (circles) in other plots.

3.8 Leverage Plot

Influential observations can be spotted in the scatterplot of residuals against leverage values (diagonal elements of the projection matrix). The well-know diagnostic called Cook's distance is a simple function of these two quantities, and level curves are drawn in the plot.

3.9 Residual differences and Distance in x-space

If there are groups of observations with identical input variables (or identical x_i vectors), then there is the possibility to perform a kind of overall goodness-of-fit test by comparing the variability of the residuals within these groups to the estimated variance of the error. This is performed by a simple Analysis of Variance of the standardized residuals with the grouping just mentioned. (Note that this method can be applied even if some or many x_i vectors are unique.)

If such groups are not available, an old idea, described by Daniel and Wood (1971/80) is to use "near replicates" for this purpose. They introduced a distance in x-space that they called WSSD and suggested to calculate all distances between observations along with the corresponding differences of residuals. If the differences of residuals are smaller for short distances than for long ones, then this points to a lack of fit of the model, which might be avoided by introducing additional terms or modelling nonlinear dependencies through transformations of variables. The procedure however does not suggest which modifications may help.

As to the distance measure d_{hi}, Daniel & Wood (1971) introduced their WSSD (weighted Squared Standard Distance (?)) with the intention to weigh variables according to their "importance" for the model. Since this feature seems too much dependent on the parametrization of the model, I prefer to use the Mahalanobis distance in the design space,

$$d_{hi}^{(X)} = n(\underline{x}_i - \underline{x}_h)^T (\boldsymbol{X}^T \boldsymbol{X})^{-1} (\underline{x}_i - \underline{x}_h)$$

(which can be calculated efficiently from the QR decomposition of $\boldsymbol{X}$).

The distance pairs and the respective differences of residuals are calculated by the function `xdistResdiff`. They are grouped and the corresponding group means of absolute residual differences are then produced by `xdistResScale`. The distances are then classified, and the residual differences are averaged over these classes. This yields a pair of mean distance $\bar{d}_k^{(X)}$ and mean residual differences $\bar{d}_k^{(R)}$ for each class k. (Trimmed means ($\alpha = 1/6$) are used by default to obtain some robustness against outliers.) These pairs are then plotted.

The class limits are chosen by fixing the percentages of distance values that they should contain. By default, the first class is quite small – because plausible deviations from the null hypothesis will lead to lower mean residual differences for short distances –, followed by a somewhat larger class, then a really big class for an intermediate range and finally a class (again somewhat smaller) for the upper end. The current default limits are therefore at 3%,10%, and 90%.

In order to assess the variability of the mean residual differences, we resort to simulation. The residuals are permuted among the observations, and then the calculation of differences is repeated. (In fact, the assignment of residual differences to the observation pairs is re-determined.) This leads to simulated mean differences for each class, and their standard deviations can be used as standard errors of the observed mean differences. Corresponding error bars are shown in the plot.

This idea can also be used for a test of goodness of fit: The test statistic

$$T = \sum_{k=1}^{K} \left((\bar{d}_k^{(R)} - \bar{d}) / \mathrm{se}_k \right)^2$$

should have a Chisquared distribution with $K - 1$ degrees of freedom. The test is calculated by the function `xdistResscale` along with the mean distances and mean residual differences.

3.10 Sequence Plot

Residuals are plotted against the sequence number in the dataset in order to show any trends and correlation that may be related to this sequence.

3.11 Residuals against explanatory variables

Plotting residuals against explanatory variables serves to detect nonlinearities of the relation between the response and the variable in question. For ordinary regression and some other models, they can show dependencies of the scale of errors on the explanatory variables.

The plots are enhanced by smooths as discussed for the Tukey-Anscombe plot. Also, a **reference line** is added that helps to decide whether a transformation of the explanatory variable may help to remove any non-linearity shown by the smooth. Again, the reference line intends to connect points of equal response values. Note, however, that they do not really align on a straight line because of the contributions of the other terms in the model to the fit. Therefore, the reference line connects points for which the sum of "component effect" $\widehat{\beta}_j x^{(j)}$ plus residual is equal. Note that the slope of the reference line is negative if the regressor has a positive effect on the response. The line is again useful for finding an adequate modification if a curved smooth appears: If all parallels to the reference line cut the smooth only once, a monotone transformation of the regressor can help. Otherwise, adding its square to the model may help.

Alternatively, the sums "component effect $\widehat{\beta}_j x^{(j)}$ plus residual" may be used as the vertical axis instead of the residuals for plotting. This display is usually called the "partial residual plot" and can be obtained from `plot.regr`, too. It is related to the plots shown by default in the same way as the response versus fitted plot is to the Tukey-Anscombe plot.

When **transformed explanatory variables** appear in the model, the residuals are plotted against the original values. The reason is as follows: If any curvature is detected, an improvement of the model may be possible by transforming the variable shown. It is then more natural to search for an enhancement of the original transformation rather than a transformation of transformed values, and this should be easier if the original scale is used. If this appears to be unsuitable in a given application, the user may store the transformed variable under a new name and then use this new variable in the model formula, or specify the transformed variable in the call to `plot`.

Residuals may be plotted against any variables using an additional function `plresx`. The function `plres2x` displays residuals and two (explanatory) variables in a special type of 3d plot. This serves to **check for interactions.**

3.12 Residuals for an ordinal or binary response

Ordinal regression models are best specified by introducing the idea of a "latent response variable" Z, which is continuous and follows a linear relationship with the explanatory variables. The response Y is considered to be a classification of Z into k classes of possibly unequal width. The most well-known model specifies a logistic distribution for the error term in the linear model for Z. Then, the coefficients and the thresholds or class limits are estimated by maximum likelihood. This is implemented in the function `polr` (proportional odds linear regression) of the `MASS` package, which is the function invoked by `regr` for ordinal regression.

Residuals for this model may be defined by considering the conditional distribution of the latent variable Z, given the observation Y and the explanatory variables. This is a logistic distribution with parameters given by the model, restricted to the class of Z given by the observed Y. Residuals are

defined as the median of this distribution, and it may help to characterize the uncertainty about the latent residuals by the quartiles of the conditional distribution. The plots show the interquartile range by a vertical line and the median, by a horizontal tick on them. (The vertical line is not shown if the probability observed Y is either too low or too high, see argument `condprobrange`.)

Note that this definition may not have appeared in the literature yet.

3.13 Residuals for censored responses

If response values are censored, so are the residuals. In the same spirit as for the case of ordered responses, it is straightforward to calculate a conditional distribution, given the fitted value and the censoring value, for each residual. This can again be plotted by showing quartiles.

3.14 Residual plots for multivariate regression.

For multivariate regression, the plots corresponding to each target variable should be examined. They are arranged such that the same type of plot for the different response variables are grouped together, and a matrix of plots is produced for residuals against explanatory variables. In addition, the qq-plot of Mahalanobis norms of residuals is shown as well as the scatterplot matrix of residuals.

4 Further Plotting Functions

4.1 Scatterplot Matrices

Scatterplot matrices are produced by the `pairs` function. In `regr0`, there is a function with more flexibility, called `plmatrix`.

- The set of variables shown horizontally and vertically need not be the same. `plmatrix` can be used to show the dependence of several response variables (vertical axes) on several explanatory ones (horizontal axes).
- A traditional square scatterplot matrix can be split to avoid tiny panels – this is even done by default. For example, if a scatterplot matrix of 15 variables is needed, `plmatrix` produces by default 6 pages of graphical output, the first one showing the upper corner of the lower triangle of the scatterplot matrix, the second, variables 7 to 11 against 1 to 6, the third, 7 to 11 against 7 to 12, and so on, until the whole lower triangle of the full scatterplot matrix is presented.
- The first argument can be a formula. If it contains a left hand side, then the respective variables will be shown last in a scatterplot matrix. This is convenient since then, the target variable will appear as the vertical axis in the last row of plots, such that its ("marginal") dependence on the explanatory variables can be seen in the usual way.
- Plotting characters and colors may be specified directly (instead of writing a panel function).
- If only one x and one y variable is given, then a regular plot appears. Therefore, `plmatrix` can be used to generate a plot with specified plotting symbols (possibly labels of more than one character each), colors and symbol sizes.

4.2 Multiboxes

5 Miscellaneous and Utility Functions

5.1 Transformation: Started Logarithm

The logarithmic transformation is very useful for quantitative variables. In fact, it should appear more often than not, since it is the "first aid transformation" for amounts and concentrations and should therefore be applied to most quantitative variables by default.

Now, amounts and concentrations may be zero (exactly or recorded as such), and the logarithm of 0 is `-Inf`. A pragamtic way out of this problem often consists of adding a constant to the variable before taking logs. Generally, this and other ideas to avoid the `-Inf` as a transformed value are called "started logs" according to John Tukey.

`regr0` includes a function `logst` that is based on the following ideas:

- The modification should only affect the values for "small" arguments.
- What is "small" should be determined in connection with the non-zero values of the original variable, since it should behave well (be equivariant) with respect to a change in the "unit of measurement".
- The function must remain monotone, and it should remain (weakly) convex.

The function `logst` implements these criteria in the following way: The shape is determined by a threshold c at which – coming from above – the log function switches to a linear function with the same slope at this point. This is obtained by

$$g(x) = \begin{cases} \log_{10}(x) & \text{if} x \geq c \\ \log_{10}(c) - (c-x)/(c\log(10)) & \text{if} x < c \end{cases}$$

The threshold c is set to $c = q_1^{1+r}/q_3^r$, where q_1 and q_3 are the quartiles of the positive data and r is a tuning constant. The default of r is 1 and leads to an expected percentage of about 2% of the data below c for log-normal data.

It is certainly useful to inspect a graph of the function, as drawn in `example(logst)`.

5.2 Documentation

For a data analysis, it is often useful to save graphical displays in documents or on paper. In an extended data analysis, one can easily lose control over the precise content of the different plots. `regr0` provides some help for keeping track.

- Every graphical display generated by a graphical function from the package gets a "stamp" in the lower right corner that indicates date and time of its creation and, if specified by the user before that time point, a project title and a step name (by writing `userOptions(project=projecttitle, step=stepname)`).
  (This stamp can of course be suppressed for producing publication graphics.)
- Data sets may be documented by attaching two attributes, `tit` and `doc` – title and description –, which will be printed with numerical output if desired.

5.3 Subsetting Data

If observations (rows) of a data.frame are to be deleted, the usual R-function `subset` is useful. The package slightly modifies it to make sure that the attributes – most importantly the documentation by `tit` and `doc` (see the preceding subsection) are passed along.

There is an additional function `dropdata` which allows for dropping observations or variables based on their names.

5.4 Multiple Frames

Function `mframe` splits the screen by calling `par(mfrow=...)`. It adds flexibility and sets other defaults for margin widths and the like.

This is the end
of the story for the time being. I hope that you will get into using `regr0` and have good success with your data analyses. Feedback is highly appreciated.

Werner Stahel, `stahel at stat.math.ethz.ch`