Package `regr` for an Augmented Regression Analysis

Werner A. Stahel, ETH Zurich

December 12, 2020

Abstract

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 `regr` contains the functions that go along with `regr` and some more that help develop regression models. It is available from `R-forge` and is still in development.

```
## Loading required package:  plgraphics
## Registered S3 methods overwritten by 'plgraphics':
##   method       from
##   nobs.survreg survival
##   nobs.coxph   survival
## Registered S3 methods overwritten by 'regr':
##   method         from
##   add1.default   stats
##   add1.mlm       stats
##   drop1.default  stats
##   drop1.mlm      stats
##   drop1.multinom nnet
```

1 Introduction

Regression models are fitted in the statistical programming environment R by diverse functions, depending on the particular type of target variable. 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 arguments. The result contains more informative numerical output than the calls to standard R functions. Residual analysis obtained by `plot`ing the result produces more appropriate and enhanced displays through the functions available from the package `plgraphics`.

`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 `regr` 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("regr", repos="http://r-forge.r-project.org")`.
The reason why it is not on CRAN 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-at-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.

```
data(d.blast, package="plgraphics")
attr(d.blast,"doc")[1] <-
  "Blasting: measurements of tremor in nearby house basements"
regroptions(printstyle="conventional")
r.blast <- regr(logst(tremor) ~ location + log10(distance) + log10(charge),
    data = d.blast)
r.blast


##
## Data:  Blasting for Tunnel Excavation
##    Blasting: measurements of tremor in nearby house basements
##   location: Schaffhausen, Switzerland
##   source: Basler and Hofmann, Zurich
##
## Call:
## regr(formula = logst(tremor) ~ location + log10(distance) + log10(charge),
##     data = d.blast, na.action = nainf.exclude)
## Fitting function: lm  | distribution: gaussian | contrasts: contr.wsum
##
## Terms:
##                   coef df  ciLow   ciUp  R2.x signif0 p.value
## (Intercept)      2.971  1  2.749  3.192     .       .       .
## location             .  .      .      . 0.102    3.48   0 ***
## log10(distance) -1.506  . -1.630 -1.382 0.477  -12.13   0 ***
## log10(charge)    0.623  .  0.547  0.699 0.102    8.17   0 ***
##
## Significance codes:
##
##
## St.dev.error:  0.1409    on 352 degrees of freedom
## Multiple R^2:  0.7981    Adjusted R^2:  0.7929     AIC:  -1409.04
## F-statistic:   154.6   on 9 and 352 d.f.,  p.value: 1.549e-116
##
## Effects of  factor levels :
##
```

```
##  location :
##      loc1          loc2          loc3          loc4          loc5          loc6
## coef -0.0174        0.1334 ***  0.1125 *** -0.1829 *** -0.0511 *     0.0530 *
##      loc7          loc8
## coef -0.0315       -0.0160
```

The package also implements a new alternative set of characteristics for the terms in the model, including "relevance" measures, see Stahel (2020) and below. As shown in the previous and the following R statements, the selection of the desired output is done by setting an option, using the function `regroptions` to be introduced below.

```
regroptions(printstyle="relevance")
r.blast

##
## Data:  Blasting for Tunnel Excavation
##    Blasting: measurements of tremor in nearby house basements
##    location: Schaffhausen, Switzerland
##    source: Basler and Hofmann, Zurich
##
## Call:
## regr(formula = logst(tremor) ~ location + log10(distance) + log10(charge),
##     data = d.blast, na.action = nainf.exclude)
## Fitting function: lm  | distribution: gaussian | contrasts: contr.wsum
##
## Terms:
##                   coef df   Rle   Rls dropRle   dropRls predRle
## (Intercept)      2.971  1  0.00  0.00      .       .          .
## location            .   .     .     .    6.92  5.63 +++    3.79
## log10(distance) -1.506  . 17.37 15.94   12.58 11.54 +++    9.60
## log10(charge)    0.623  .  8.93  7.84    8.47  7.43 +++    5.47
##
## Relevance codes:    -  -1    0  .  1  +  2  ++  5  +++
##
##
## St.dev.error:  0.1409   on 352 degrees of freedom
## Multiple R^2:  0.7981    Adjusted R^2:  0.7929    AIC:  -1409.04
## F-statistic:   154.6   on 9 and 352 d.f.,  p.value: 1.549e-116
##
## Effects of  factor levels :
```

```
##
##  location :
##      loc1        loc2        loc3        loc4        loc5        loc6
## coef -0.0174      0.1334 ++   0.1125 ++  -0.1829 +++ -0.0511 .     0.0530 .
##      loc7        loc8
## coef -0.0315 -   -0.0160 -
```

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, when the "conventional" columns are selected,

`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.

`signif0,` 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; the usual significance symbols are attached.

With the "relevance" selection, the output shows

`coef, df, R2.x,` as above;

`coefRle,` the estimated Relevance of the coefficient;

`coefRls,` the secured Relevance of the coefficient;

`coefRlp,` the potential Relevance of the coefficient; `Rls` and `Rlp` form a confidence interval for relevance;

`dropRls,` the secured Relevance of the term, see Section 2.2.

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 in the standard output. 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 number of 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 (see Stahel, 2020).

The new significance measure is defined as

$$\texttt{signif0} = \texttt{t value / critical value}$$

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

Standardized Coefficients. Standardized coefficients are meant to allow for a comparison of the importance of explanatory variables that have different variances. Each of them shows the effect on the response of increasing "its" carrier $X^{(j)}$ by one standard deviation, as a multiple of either the response's standard deviation or the error standard deviation `sigma`, called `stcoef` and `estcoef`, respectively. Whereas the first version is popular, the second has a more satisfactory interpretation (see Stahel, 2020).

Note that 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 coeffients are somewhat counter-intuitive in this case.

Relevance. In the standard situation of comparing two groups, a popular standarized effect size is Cohen's d, defined as the ratio of the difference between the groups to the standard deviation within groups. The value of 0.5 has is judged as a "medium effect size". It is used as a threshold for declaring a "relevant effect" by Stahel (2020). If the two groups model is expressed as the simple regression model with just a binary explanatory variable, the error standardized coefficient is $d/2$ in case of equal group sizes, and therefore, the relevance threshold is 0.25. Its use is advocated for the error standardized coefficient in general. The relevance measure is defined as the ratio of the estimated error standardized coefficient to this threshold, `Rle = estcoef/0.25`.

Confidence Intervals. The standard errors provided by the usual `summary` tables allow for calculating confidence intervals for continuous explanatory variables, by the formula `coef` $\pm$ $q_{df} \cdot$`std.error`.

They can be calculated alternatively as

$$\texttt{coef} \cdot \left(1 \pm 1/\texttt{signif0}\right).$$

This is slightly more complicated for a calculation in the mind than `coef` $\pm\,2\,$`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.

Since the standardized coefficients are proportional to the unstandardized one, confidence intervals are easily transfered to them. The two default `regr` outputs show either the interval for `coef` in columns `ciLow` and `ciUp` or for the relevance, as `Rls`, "secured relevance," and `Rlp`, "potential relevance".

Customized output. The result of the `regr` function stores all of these informations in the `termtable` component. The user can select the columns to be shown by setting the `printstyle` or `termcolumns` options. Asking for `regroptions(printstyle="relevance")` (the default) or `regroptions(printstyle="conventional")` selects one of two default sets of columns.

```
names(r.blast$termtable)

##  [1] "coef"           "Df"             "se"             "ciLow"
##  [5] "ciUp"           "signif0"        "stcoef"         "st.Low"
##  [9] "st.Up"          "testst"         "p.value"        "p.symbol"
## [13] "R2.x"           "Rle"            "Rls"            "Rlp"
## [17] "dropRle"        "dropRls"        "dropRlp"        "predRle"
## [21] "predRls"        "predRlp"        "df"             "dropRls.symbol"
## [25] "predRls.symbol"

regroptions(termcolumns=c("coef", "stcoef", "df", "signif0", "p.symbol"))

print(r.blast, call=FALSE, termeffects=FALSE) ## avoid repetition of these parts

##
## Data:  Blasting for Tunnel Excavation
##    Blasting: measurements of tremor in nearby house basements
##   location: Schaffhausen, Switzerland
##   source: Basler and Hofmann, Zurich
##  Terms:
##                  coef stcoef df    signif0
## (Intercept)     2.971  0.000  1        .
## location            .     .   .     3.48 ***
## log10(distance) -1.506 -1.737  . -12.13 ***
## log10(charge)    0.623  0.893  .    8.17 ***
##
## Significance codes:
##
##
## St.dev.error:  0.1409   on 352 degrees of freedom
## Multiple R^2:  0.7981    Adjusted R^2:  0.7929    AIC:  -1409.04
## F-statistic:   154.6   on 9 and 352 d.f.,  p.value: 1.549e-116

regroptions(default="all")  ## reset the options
```

The columns of the `termtable` not yet mentioned are:

se, the standard errors of the estimated coefficients;

testst, the test statistic used for testing if the coefficient could be zero;

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;

estcoef, the "error standardized" coefficient, for which the error standard deviation σ is used instead of the standard deviation of the response;

stciLow, stciUp, estciLow, estciUp, the respective confidence intervals;

dropRle, dropRlp, estimated and potential Relevance for the term;

p.symbol, coefRls.symbol, dropRls.symbol, the symbols for significance and secured relevance that are attached to their numerical values when printed.

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 (or more generally for any term in the regression model), 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

$$\texttt{signif0} = \sqrt{\texttt{F value / critical value}}$$

where `critical value` is the critical value of the F distribution. It reduces to the former one for binary factors, up to the missing sign.

Similarly, the **relevance** is defined by comparing the full model with the one resulting from dropping the term in question. If σ_f and σ_r are the error standard deviations in the two models, then

$$\texttt{Effect size} = \log(\sigma_r/\sigma_f) = \tfrac{1}{2} \log\big((1 - \texttt{r.squared}_f)/(1 - \texttt{r.squared}_r)\big) \ ,$$

and `Relevance = Effect size / Relevance threshold` with `Relevance threshold = 0.03` by default. Its estimated value and confidence interval is contained in the columns `dropRle` and `dropRls, dropRlp`, respectively. For details, see Stahel (2020).

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, `coefficients(r.blast)`.

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 are 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 `regr` 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 `regr`.

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 call a blunt 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.

3 Model Development

The functions `drop1` and `add1` are basic tools for model development. Their method for `regr` objects provide additional features.

First, they calculate tests by default. Second, `add1` has a useful default value for the argument `scope`: It consists of all squared continuous variables and all interactions between variables in the

9

model. (These terms are calculated by `terms2order`.)

Since `drop1` and `add1` methods are available, `step` can be used to select a model automatically. However, it is preferable to use the version `step.regr` instead. It chooses "both" directions, backward and forward, by default, and extends the `scope` as in `add1`. There are two more features that are explained in the following.

```
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.regr(r.blast, k=4)
formula(r.step)

## logst(tremor) ~ location + log10(distance) + log10(charge)
```

Stopping rule. The usual stopping rule for stepwise model selection relies on the AIC criterion. Note that there is a funny argument for this choice: The rule that was commonly used early on stopped when all terms were formally significant. It 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 larger 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 even larger models than the significance criterion.

Unfortunately, it is not possible to specify the stopping criterion in `step`. However, setting its argument `k=4` – the default for `step.regr` –, one usually obtains a model with formally significant terms, for which every additional regressor would be non-signifcant.
($\text{AIC}^{(k)}$ is $2L + k\text{df}$, where L is the log likelihood. When dropping a single regressor from model 1 to obtain model 2, $2(L_1 - L_2)$ has a chisquared distribution with 1 degree of freedom, for which the

critical value is 3.84. Thus, $\text{AIC}_2^{(3.84)} - \text{AIC}_1^{(3.84)} < 0$, if and only if $2(L_1 - L_2) < 3.84$, i.e., the test is non-significant, and the regressor will be dropped.)

Missing values. The `regr` methods for `drop1` and `step` behave more flexibly than in basic R when there are missing values:

`drop1` drops the rows in `data` that contain a missing value for the regressors in the current model and proceeds, rather than complaining about a variable number of missing observations.

`add1` drops the rows with missings in any variable used by the `scope` before calculations.

`step.regr` uses these two functions for the two directions. It only uses the observations without missing values in any variables in `scope`, but gives as the resulting fit the one based on the complete observations for the variables in the final formula.

3.1 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.

```
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
```

4 Residual Analysis

Various graphical displays involving residuals are of great help to check if assumptions are violated and to find improved models. Plain R provides four displays when the plot function is called for `lm` and similar objects. When `plot` is called on `regr` objects, it produces enhanced displays that

are adapted to the type of fitted model and adds plots of residuals against input variables. This is implemented by making the function `plregr` in the package `plgraphics` the plot method for class `regr`. The `plgraphics` package features easy choices of most graphical elements, like plotting symbols, colors, tickmarks, etc. It implements methods for displaying censored variables, residuals from ordinal regression, plotting residuals against two variables, and more. See the package vignette and help files for details.

Here are the residual plots shown by default.

!!! No, this currently produces an error. Please see the vignette of `plgraphics`

5 Details

5.1 regr options

Regr options shape the output of the `regr` function or even influence the way fitting functions are called.

Here is a list of the options.

digits number of digits used when printing `regr` objects;

regr.contrasts contrasts used as a default in generating designs;

factorNA logical: for explanatory factors, should the missing values form a level of the factor?

testlevel level of tests = 1 - confidence level (two sided);

RlvThres vector of length 2: thresholds for relevance of coefficients and terms (component `drop`);

termtable logical: should a term table be calculated?

vif

show.termeffects logical: should term effects be shown?

termcolumns names of columns of the `termtable` component of the fitting result object which will be printed; alternatively,

printstyle can be set either to `"relevance"` or to `"conventional"`: these then select the first or second of the following components;

termcolumns.r, termcolumns.c columns selected in the case just mentioned;

termeffcolumns same for the term effects tables;

coefcolumns same for the coefficients' table for each term;

na.print symbol by which NA's are printed;

notices logical: should notices be shown?

References

Stahel, Werner A. (2020). *Measuring Significance and Relevance instead of p-values.* In preparation.

This is the end of the story for the time being, 12.12.2020. More material will be added for details about the many more regression models that are supported.

I hope that you will get into using `regr` and have good success with your data analyses. Feedback is highly appreciated.

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