texreg: Conversion of Statistical Model Output in R to LaTeX and HTML Tables

Philip Leifeld
University of Konstanz

Abstract

A recurrent task in applied statistics is the (mostly manual) preparation of model output for inclusion in LaTeX, Microsoft Word, or HTML documents – usually with more than one model presented in a single table along with several goodness-of-fit statistics. However, statistical models in R have diverse object structures and summary methods, which makes this process cumbersome. This article first develops a set of guidelines for converting statistical model output to LaTeX and HTML tables, then assesses to what extent existing packages meet these requirements, and finally presents the **texreg** package as a solution that meets all of the criteria set out in the beginning. After providing various usage examples, a blueprint for writing custom model extensions is proposed.

Keywords:˜reporting, table, coefficients, regression, R, LaTeX, Microsoft Word, HTML, Markdown.

This R package vignette is based on an article in the Journal of Statistical Software (Leifeld 2013).

1. Typesetting R model output in LaTeX and HTML

The primary purpose of the statistical programming language R (R Core Team 2013) is the analysis of data with statistical models. One of the strengths of R is that users can implement their own statistical models. While this flexibility leads to an increased availability of even exotic models and shorter cycles between model development and implementation, there are also downsides of this flexibility. In particular, there is no unified data structure of statistical models and no unified way of representing model output, which makes it hard to re-use coefficients and goodness-of-fit (GOF) statistics in other software packages, especially for the purpose of publishing results.

Several generic functions were developed to provide unified accessor methods for coefficients (the `coef()` function), GOF statistics (for example, `AIC()`, `BIC()`, `logLik()`, or `deviance()`), a custom text representation of the fitted model (`summary()`), and other relevant pieces of information (e.g., `nobs()` and `formula()`). Details are provided in Chapter˜11 of Venables, Smith, and R Core Team (2013). Nonetheless, many popular packages have only partially implemented methods for these generics, and in some cases they do not even provide accessor functions at all for their coefficients or GOF statistics. Even worse, the model summary methods are usually structured in idiosyncratic ways and do not lend themselves to easy parsing of coefficients and GOF statistics.

Modern scientific journals, on the other hand, often require nicely formatted and standardized model output, usually in the form of coefficient tables for one or more models. In the majority of applications, these tables show more than one model aligned next to each other with partially overlapping coefficient names, standard errors in parentheses, and superscripted stars indicating the significance of model terms. At the bottom of the table, summary statistics like the number of observations are reported, and GOF measures like AIC or R^2 are shown. Due to the idiosyncratic way model output is currently represented in various classes in R, designing these kinds of tables for a paper submission requires a substantial amount of time and patience, especially if more than one model is involved and if there are many model terms. Copying and pasting coefficients and standard errors one at a time often becomes the default way of handling this task.

An important tool for typesetting academic papers in many academic fields is LaTeX (Lamport 1986). In fact, R and LaTeX are closely linked by the `Sweave()` command in the **utils** package (R Core Team 2013), which allows the integration of R commands in LaTeX documents and their execution and evaluation at runtime (Leisch 2002). In spite of this, common approaches for linking R model output and tables in LaTeX include (1) copying and pasting individual values after every change of the model, (2) custom user-written functions which convert a specific model into a matrix, (3) the use of sophisticated table-management packages (see next section), and (4) the inclusion of single models in the form of the model summary instead of nicely aligned coefficient tables as a second best solution.

Popular alternatives for document preparation include Microsoft Word and the dynamic report generation R package **knitr** (Xie 2013a,b,c). Both **knitr** and Microsoft Word accept HTML input, and **knitr** additionally supports Markdown, a simplified HTML-like markup language. These platforms face similar complications as LaTeX and `Sweave()` regarding the preparation of regression tables for multiple statistical models.

The ideal way to prepare R model output for LaTeX and HTML tables would be a generic function which would directly output LaTeX or HTML tables and for which custom methods for any model type could be written as extensions. While several attempts already exist (see Section˜3), all of them have limitations. This article introduces the **texreg** package (Leifeld 2014), which closes this gap and provides a unified framework for typesetting LaTeX and HTML tables for various statistical models in R. Package **texreg** is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=texreg`.

The remainder of this article is structured as follows: Section˜2 sets out a number of requirements which must be met. In the light of these requirements, Section˜3 compares **texreg** to other R packages and functions which were designed for similar purposes. Section˜4 describes the way how **texreg** works and how its functions and classes are related. After providing several examples and illustrating the options of the `texreg()`, `htmlreg()`, and `screenreg()` functions (Section˜5), Section˜6 describes how new extensions can be implemented.

2. Requirements

The design of the **texreg** package tries to accomplish six goals: it should be capable of dealing with several models in a single table; it should be easily extensible by package writers and users; it should provide options for using the available space in an optimal way; it should take advantage of advanced layout capabilities in LaTeX and HTML; it should take care of

journal- or model-specific habits or best practices; and it should find an optimal balance of customizability and usability. These requirements are elaborated in the following paragraphs.

2.1. Managing multiple models

Quite often, almost-identical models are printed in order to show how an additional model term alters the other coefficients and standard errors. There are, however, also cases where different model types are applied to the same data. This implies that the package must not only be able to merge the names of coefficients to guarantee comparability of coefficient columns; it must also be able to deal with different model classes and accommodate different kinds of GOF statistics in the same table.

Moreover, it must be possible to rename the model terms and GOF statistics. Custom coefficient names not only make the output more easily comprehensibe by the readers; renaming model terms is also mandatory for unifying terms between several models. For example, two models based upon two different datasets may have different variable names for the same kind of theoretical construct. This would result in two separate but complementary rows in the coefficient table. It should be possible to rename coefficients and then conflate any two or more complementary rows with identical labels.

Finally, it should be possible to assign custom names for the different models, instead of the default "Model 1", "Model 2", etc. While it may be easy to rename them manually in many applications, particularly `Sweave()` and `knitr()` require that no manual manipulation is necessary.

2.2. Using generics to make texreg easily extensible

Different model classes have different ways how their coefficients and GOF statistics can be accessed. Hardcoding these extraction rules into the functions of the **texreg** package would inhibit customizability. The best way to make **texreg** extensible is to have a generic `extract()` function which can be invoked on any kind of model, just like `plot()` or `print()` generics in R. Any user—especially model class authors—would then be able to add custom methods to the `extract()` function in order to make **texreg** learn how to cope with new models. For example, an `extract()` method for '`lm`' objects can be written to deal with linear models, or an `extract()` method for '`ergm`' objects can be written to make the generic `extract()` function understand exponential random graph models. All the user has to do is write a custom extract function for a specific model type and then register it as a method for the generic `extract()` function. Section˜6 provides details on how this can be accomplished.

2.3. Use available space efficiently

If a table has many model terms and if standard errors are printed in parentheses below the coefficients, the table may become too large for a single page. For this reason, it should be possible to print standard errors right beside the coefficients instead of aligning them vertically. In **texreg**, this is achieved with the `single.row` argument.

If tables grow too large, other measures might prove useful: removing table margins, setting the table in script size, or setting custom float positions (for LaTeX tables). Very wide tables should be rotated by 90 degrees using the `sidewaystable` environment in the LaTeX package **rotating** (Rahtz and Fairbairns 2008) in order to use the available space in an optimal way.

The user should also be able to set the table caption and label, decide whether the table should be in a float environment (for LaTeX tables), align the table horizontally on the page, and set the position of the caption. The **texreg** package provides arguments to control all of these aspects.

2.4. Beautiful and clean table layout

For LaTeX tables, the LaTeX package **dcolumn** (Carlisle 2001) provides facilities for aligning numbers in table cells nicely, for example at their decimal separators. The **booktabs** package (Fear 2005) can draw top, mid and bottom rules for tables and produces a cleaner layout than the default horizontal lines. Both packages are supported by **texreg** and can be switched on or off depending on the availability of the packages in the LaTeX distribution.

For HTML tables, cascading style sheets (CSS) should be used to adjust the layout, and the user should be able to decide whether CSS markup should be included in the file header or inline.

2.5. Journal- or model-specific requirements

Academic journals may have different requirements regarding the number of digits to be printed, the inclusion of superscripted stars indicating significance, or the removal of leading zeroes. Similarly, there are best practices in different academic communities or for specific model types. For example, it is common practice to attach three stars to coefficients with p˜values ≤ 0.001 and small centered dots to coefficients with p˜values between 0.05 and 0.1 in exponential random graph models, while less fine-grained significance levels are adopted in many other communities (for example, three stars for $p \leq 0.01$, or only one star or bold formatting for one single significance level). In yet other communities, journals or models, p˜values or significance stars are not required or even deemed inappropriate (see the **lme4** package by Bates, Maechler, Bolker, and Walker 2013).

2.6. Customizability and usability

Different users have different applications in mind. For this reason, a solution should be as flexible as possible and offer customization via arguments. For example, inclusion of an HTML table in a **knitr** Markdown document requires that only the table is printed without any header or document type information, and that significance stars are escaped using backslashes.

In other situations, it may be important to (1) omit certain coefficients (like random or fixed effects or thresholds), (2) reorder the coefficients in the model (e.g., because some models put interaction effects at the end of the list of coefficients), or (3) replace coefficients, standard errors, or p˜values by custom vectors, for example when heteroskedasticity-consistent ("robust") standard errors are used (Zeileis 2004).

Argument	texreg	htmlreg	screenreg	Short description
`l`	•	•	•	Model or list of models
`file`	•	•	•	Divert output to a file
`single.row`	•	•	•	Print coefs and standard errors in the same row?
`stars`	•	•	•	Threshold levels for significance stars
`custom.model.names`	•	•	•	Set the names of the models
`custom.coef.names`	•	•	•	Replace the names of the model terms
`custom.gof.names`	•	•	•	Replace the names of the GOF statistics
`custom.note`	•	•	•	Replace the default significance legend
`digits`	•	•	•	Number of decimal places
`leading.zero`	•	•	•	Print leading zeroes?
`symbol`	•	•	•	Dot symbol denoting a fourth significance level
`override.coef`	•	•	•	Replace coefficients by custom vectors
`override.se`	•	•	•	Replace standard errors by custom vectors
`override.pval`	•	•	•	Replace p˜values by custom vectors
`omit.coef`	•	•	•	Remove rows using a regular expression
`reorder.coef`	•	•	•	Provide a custom order for the model terms
`reorder.gof`	•	•	•	Provide a custom order for the GOF statistics
`return.string`	•	•	•	Return the table as a character vector?
`ci.force`	•	•	•	Convert standard errors to confidence intervals
`ci.level`	•	•	•	Confidence level for CI conversion
`ci.star`	•	•	•	Print star when 0 is not contained in the CI
`groups`	•	•	•	Partition rows of the table into groups with labels.
`bold`	•	•	○	p˜value below which coefficients are bolded
`center`	•	•	○	Horizontal alignment on the page
`caption`	•	•	○	Set the caption of the table
`caption.above`	•	•	○	Should the caption be placed above the table?
`label`	•	○	○	Set the label of the table
`booktabs`	•	○	○	Use the **booktabs** package (Fear 2005)?
`dcolumn`	•	○	○	Use the **dcolumn** package (Carlisle 2001)?
`sideways`	•	○	○	Use `sidewaystable` (Rahtz and Fairbairns 2008)
`use.packages`	•	○	○	Print the `\usepackage{}` declarations?
`table`	•	○	○	Wrap `tabular` in a table environment?
`no.margin`	•	○	○	Remove margins between columns to save space
`scriptsize`	•	○	○	Use smaller font size to save space
`float.pos`	•	○	○	Specify floating position of the table
`star.symbol`	○	•	○	Change the significance symbol
`inline.css`	○	•	○	Use CSS in the text rather than the header
`doctype`	○	•	○	Include the `DOCTYPE` declaration?
`html.tag`	○	•	○	Include the `<html>` tag?
`head.tag`	○	•	○	Include the `<head>` tag?

Continued on next page

Table 1 – continued from previous page

Argument	texreg	htmlreg	screenreg	Short description
body.tag	○	●	○	Include the <body> tag?
column.spacing	○	○	●	Number of spaces between columns
outer.rule	○	○	●	Line type for the outer rule
inner.rule	○	○	●	Line type for the inner rule
...	●	●	●	Additional arguments for the extract functions

Table 1: Arguments of the `texreg()`, `htmlreg()` and `screenreg()` functions.

On the other hand, users should not be required to learn the meaning of all arguments before they can typeset their first table. The default arguments should serve the needs of occasional users. Moreover, adjusting tables based on a complex set of arguments should be facilitated by printing tables to the R console before actually generating the LaTeX or HTML output. If this screen representation of the table is nicely formatted and aligned using spaces and rules, it can also serve as an occasional replacement for the generic `summary()` method for easy model comparison as part of the statistical modeling workflow.

The **texreg** package tries to balance these needs for customizability and usability by providing many arguments for layout customization (see Table˜1 for a list of arguments), using sensible default values for occasional users, and providing a function for on-screen display of tables for easy model comparison and layout adjustment.

3. Comparison with other packages

Beside **texreg**, several other packages were designed to convert R model output to LaTeX or HTML tables.

The **xtable** package (Dahl 2012) is able to typeset various matrices and data frames from R as LaTeX or HTML tables. It is very flexible and has its strengths particularly when it comes to tables of summary statistics. However, it was not specifically designed for statistical model output. Similarly, the `mat2tex()` command from the **sfsmisc** package (Maechler and others 2012) can export matrices to LaTeX, and the `tex.table()` function in the **cwhmisc** package (Hoffmann 2012) is able to export data frames as LaTeX tables.

For several years, the `outreg()` function in the **rockchalk** package (Johnson 2012) has been available for exporting multiple regression models to LaTeX . However, the function remains fairly basic and does not provide a great deal of layout options, generics, and custom model types (in fact, only '`lm`' and '`glm`' objects).

The **apsrtable** package (Malecki 2012), the `mtable()` function from the **memisc** package (Elff 2012), and the **stargazer** package (Hlavac 2013) are more advanced and can also merge multiple models in a single table. **apsrtable** and **memisc** feature custom functions for the extraction of coefficient and GOF information, and they are based on generics. In this regard, both packages

are somewhat similar to the **texreg** package. **texreg**, however, offers more straightforward ways of custom model implementation. This important feature is notably absent from **stargazer**.

Class	Package	Description
'aftreg'	**eha**	Accelerated failure time regression
'betareg'	**betareg**	Beta regression for rates and proportions
'brglm'	**brglm**	Bias-reduced generalized linear models
'btergm'	**btergm**	Temporal exponential random graph models
'clm'	**ordinal**	Cumulative link models
'clogit'	**survival**	Conditional logistic regression
'coeftest'	**lmtest**	Wald tests of estimated coefficients
'coxph'	**survival**	Cox proportional hazard models
'coxph.penal'	**survival**	Cox proportional hazard with penalty splines
'dynlm'	**dynlm**	Time series regression
'ergm'	**ergm**	Exponential random graph models
'fGARCH'	**fGarch**	GARCH time series models
'gam'	**mgcv**	Generalized additive models
'gamlss'	**gamlss**	GAM for location scale and shape
'gee'	**gee**	Generalized estimation equation
'glm'	**stats**	Generalized linear models
'[gl|l|nl]merMod'	**lme4** (<1.0)	Generalized linear mixed models
'gls'	**nlme**	Generalized least squares
'gmm'	**gmm**	Generalized method of moments estimation
'ivreg'	**AER**	Instrumental-variable regression using 2SLS
'hurdle'	**pscl**	Hurdle regression models for count data
'lm'	**stats**	Ordinary least squares
'lme'	**nlme**	Linear mixed-effects models
'lme4'	**lme4** (≥ 1.0)	Linear mixed-effects models
'lmrob'	**robustbase**	MM-type estimators for linear models
'lnam'	**sna**	Linear network autocorrelation models
'lrm'	**rms**, **Design**	Logistic regression models
'maBina'	**erer**	Marginal effects for binary response models
'mnlogit'	**mnlogit**	Multinomial choice models
'multinom'	**nnet**	Multinomial log-linear models
'negbin'	**MASS**	Negative binomial generalized linear models
'nlme'	**nlme**	Nonlinear mixed-effects models
'pgmm'	**plm**	GMM estimation for panel data
'phreg'	**eha**	Parametric proportional hazards regression
'plm'	**plm**	Linear models for panel data
'pmg'	**plm**	Linear panel models with heterogeneous coefficients
'polr'	**MASS**	Ordered logistic or probit regression
'Relogit'	**Zelig**	Rare events logistic regression
'rem.dyad'	**relevent**	Relational event models for dyadic data
'rlm'	**MASS**	Robust fitting of linear models

Continued on next page

Table 2 – continued from previous page

Class	Package	Description
'rq'	**quantreg**	Quantile regression models
'sclm'	**ordinal**	Cumulative link models
'sienaFit'	**RSiena**	Stochastic actor-oriented models for networks
'simex'	**simex**	SIMEX algorithm for measurement error models
'stergm'	**tergm**	Temporal exponential random graph models
'survreg'	**survival**	Parametric survival regression models
'survreg.penal'	**survival**	Frailty survival models
'svyglm'	**survey**	Survey-weighted generalized linear models
'systemfit'	**systemfit**	Linear structural equations
'texreg'	**texreg**	For easy manipulation of **texreg** tables
'tobit'	**AER**	Tobit regression models for censored data
'weibreg'	**eha**	Weibull regression
'zelig'	**Zelig**	Zelig models (Owen, Imai, King, and Lau 2013)
'zeroinfl'	**pscl**	Zero-inflated regression models

Table 2: List of 56 supported model types (version 1.32).

The **apsrtable** package (version 0.8-8) has custom functions for 'coxph', 'gee', 'glm', 'lm', 'lrm', 'negbxin,' 'svyglm' and 'tobit' objects, but it does not feature any multilevel models or network models. The **memisc** package (version 0.95-38) features 'aftreg', 'betareg', 'clm', 'dynlm', 'glm', 'hurdle', 'ivreg', 'lm', 'lmer', 'mer', 'multinom', 'phreg', 'polr', 'tobit', 'simex', 'survreg', 'weibreg', and 'zeroinfl' models but cannot handle any network models and recent versions of **lme4** multilevel models (Bates *et˜al.* 2013).

The **stargazer** package (version 3.0.1) has built-in functions for 'betareg', 'clm', 'clogit', 'coxph', 'ergm', 'gam', 'gee', 'glm', 'glmerMod, 'gls', 'hurdle', 'ivreg', 'lm', 'lmerMod, 'lmrob', 'multinom', 'nlmerMod', 'plm, 'pmg', 'polr', 'rlm', 'survreg', 'svyglm, 'tobit', and 'zeroinfl' objects as well as several **Zelig** adaptations (Owen *et˜al.* 2013), but it does not support custom user extensions.

texreg (version 1.32), in contrast, can deal with all of the above model types (that is, the union of all three packages, except for some **Zelig** models), is extensible, and offers additional built-in functions for the following model classes: 'brglm', 'btergm','coxph.penal', 'fGARCH', 'gamlss', 'gmm', 'lme', 'lme4', 'lnam', 'maBina', 'nlme', 'rem.dyad', 'rq', 'sclm', 'stergm', 'systemfit', 'texreg' and 'zelig' ('logit' and 'relogit') objects. Table˜2 gives an overview of currently implemented model types.

texreg supports Microsoft Word, HTML, Markdown, and **knitr** whereas the other packages (except for **xtable**) are restricted to LaTeX output. **apsrtable** has an option for Sweave integration, which does not require any argument in **texreg**. In the **memisc** package and in **texreg**, the **booktabs** and **dcolumn** LaTeX packages for table layout (see Section˜2.4) can be used, which is not available in **apsrtable** (and only **dcolumn** is supported in **stargazer**).

While **apsrtable** and **texreg** allow for custom GOF measures, **memisc** and **stargazer** only feature a set of hardcoded statistics. Apart from this, all packages presented here are significantly less flexible than **texreg** regarding the utilization of space (Section˜2.3), layout options

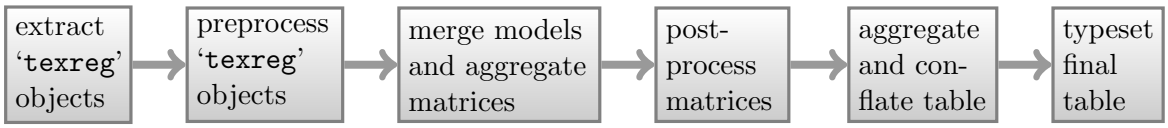

Figure 1: Simplified flow diagram of a `texreg()`, `htmlreg()`, or `screenreg()` call.

(Section˜2.4), outlet- or model-specific requirements (Section˜2.5), and customizability (Section˜2.6).

4. Under the hood: How texreg works

The **texreg** package consists of three main functions:

1. `texreg()` for LaTeX output;
2. `htmlreg()` for HTML, Markdown-compatible and Microsoft Word-compatible output;
3. `screenreg()` for text output to the R console.

There are various internal helper functions, which are called from each of these main functions for purposes of pre- and postprocessing. Moreover, there is a class definition for '`texreg`' objects, and a generic `extract()` function along with its methods for various statistical models. Figure˜1 illustrates the procedure following a call of one of the main functions. Details about each step are provided below.

4.1. The generic `extract()` function and its methods

First, the user hands over a model object or a list of models to the `texreg()`, `htmlreg()` or `screenreg()` function. This main function calls the generic `extract()` function in order to retrieve all the relevant pieces of information from the models. The `extract()` function knows how to cope with various model types (see Table˜2) because it merely calls the appropriate `extract()` method designed for the specific model type. For example, if the model is of class '`lm`', `extract()` calls the `extract()` method for '`lm`' objects, etc. Custom `extract()` methods can be easily added (see Section˜6).

An `extract()` method aggregates various required pieces of information, like the coefficients, their names, standard errors, p values, and several GOF measures. Which measures are used depends on the specific `extract()` method. It is also possible to let the user decide: beside the `model` argument, each extract method is allowed to have more arguments. For example, the `extract()` method for '`lme4`' objects, which are **lme4** multilevel models (Bates *et˜al.* 2013), has Boolean options like `include.variance`, which turns on the inclusion of random effect variances in the GOF block, and numeric arguments like `conf.level`, which sets the confidence level for computation of profile or bootstrapped confidence intervals. When the main function is called in the first place, the user can include these custom arguments to finetune the behavior of the `extract()` methods.

Once the relevant data have been extracted from a model object, the `extract()` method creates a '`texreg`' object by calling the `createTexreg()` function and handing over the ex-

tracted data. The '`texreg`' object or the list of '`texreg`' objects is finally returned to the main function.

4.2. '`texreg`' objects: An S4 class

There is an S4 class definition for '`texreg`' objects. Such an object contains four vectors for the coefficients—the coefficient values (`numeric`), their names (`character`), standard errors (`numeric`), and p˜values (`numeric`)—, and three vectors for the GOF statistics: the GOF values (`numeric`), their names (`character`), and dummy variables indicating whether it makes sense for the GOF value to have several decimal places (`logical`); for example, one would not want the number of observations to have any decimal places.

As some types of statistical models report confidence intervals rather than standard errors and p˜values, the '`texreg`' class definition can alternatively store lower and upper bounds of confidence intervals instead of standard errors and p˜values. Which slots of the class are used depends on the `extract()` method for the specific model. The **texreg** package checks whether standard errors are present in the '`texreg`' object and use either standard errors or confidence intervals depending on availability.

The class contains validation rules which make sure that the four coefficient vectors all have the same length and that the three GOF vectors also all have the same length. There are several exceptions to this rule: the p˜values, the confidence intervals, and the decimal-place vector are optional and may also have a length of zero.

The '`texreg`' class definition was written to facilitate the handling of the relevant pieces of information. Handing over lists of '`texreg`' objects between functions is more user-friendly than handing over lists of nested lists of vectors. '`texreg`' objects are created by the `extract()` methods and handed over to the `texreg` function (see Section˜4.1).

4.3. Preprocessing the '`texreg`' objects

Once all '`texreg`' objects have been returned to the `texreg()`, `htmlreg()` or `screenreg()` function, they have to be preprocessed. This entails two steps: first, coefficients, standard errors or p˜values must be replaced by user-defined `numeric` vectors (for example if robust standard errors have been manually computed). The arguments `override.coef`, `override.se`, and `override.pvalues` serve to replace the coefficients, standard errors, and p˜values, respectively. Second, LaTeX -specific markup codes are replaced by their HTML or plain-text equivalents if `htmlreg()` or `screenreg()` are called instead of `texreg()`.

4.4. Matching the model terms

After preprocessing the '`texreg`' objects, their contents are arranged in three separate matrices: the *coefficient block matrix* consists of three columns for each model (coefficient, standard error, and p˜value); the *GOF block matrix* consists of one column for each model and one row for each GOF statistic and contains the GOF values; and the *decimal matrix* has the same dimensions as the GOF block matrix and indicates for each GOF value whether it should have decimal places (e.g., R^2, AIC, etc.) or whether it is an integer (e.g., number of observations, number of groups, etc.). All of these matrices are created by matching the names of the coefficients or GOF names of the different models to avoid redundancy. The three matrices are kept separate during the postprocessing stage and are then combined in a single table.

4.5. Postprocessing and rearranging of the matrices

During the postprocessing stage, the coefficient and GOF names are replaced by user-defined names (using the `custom.coef.names` and `custom.gof.names` arguments), coefficient rows are removed by applying regular expressions to the row names (using the `omit.coef` argument), and coefficients/standard errors and GOF statistics are reordered according to the user's wishes (following the `reorder.coef` and `reorder.gof` arguments).

Renaming the coefficients or GOF names may lead to duplicate entries. These duplicate rows must be conflated. For example, there may be one row with the name "duration" (with the `duration` variable only existing in the first model) and another row with the name "time" (with the `time` variable only existing in the second model). After renaming both rows to either of the two names, the two rows must be conflated such that there is only one row left with the `duration` coefficient in the first cell and the `time` coefficient in the second cell.

Rearranging the matrix also entails checking for rows with duplicate names which are in fact *not* complementary and rearranging them only by presenting the fullest rows first. Furthermore, there may be more than two duplicate rows with the same name and other complex configurations which are handled by **texreg**. Finally, rearranged rows are reordered to ensure that models appear as compact as possible in the table.

4.6. Aggregating the table and conflating columns

Before the data are aggregated in the final table, all coefficients, standard errors and GOF values must be formatted according to the specifications of the user: they have to be rounded (following the `digits` argument), leading zeroes must be removed if desired by the user (as set by the `leading.zero` argument), and the `numeric` values are converted into `character` strings.

The p˜value column of the coefficient block matrix is then used to add significance stars or bold formatting depending on the `stars`, `symbol`, `star.symbol`, and `bold` arguments. In the final table, the standard error and p˜value columns are removed, and the standard errors are either inserted between the coefficient and the stars or in separate rows below the coefficients (depending on the `single.row` argument).

4.7. Typesetting the final table

The final table is eventually translated into LaTeX or HTML code and either printed to the R console or diverted to a file (depending on the `file` argument). All three functions, `texreg()`, `htmlreg()` and `screenreg()`, have their own custom arguments for the layout of the table. These specific options are listed and explained at the bottom of Table˜1.

5. Examples

This section gives some practical examples. All data and model formulae were taken from the help files of the respective models and their packages for the sake of replicability.

5.1. The `screenreg()` function

First, consider a simple linear model as created by the `lm()` function:

```
R> ctl <- c(4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14)
R> trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
R> group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
R> weight <- c(ctl, trt)
R> m1 <- lm(weight ~ group)
R> m2 <- lm(weight ~ group - 1)
```

The coefficients, standard errors, p˜values etc. of model˜2 can be displayed as follows:

```
R> summary(m2)

Call:
lm(formula = weight ~ group - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0710 -0.4938  0.0685  0.2462  1.3690

Coefficients:
         Estimate Std. Error t value Pr(>|t|)
groupCtl   5.0320     0.2202   22.85 9.55e-15 ***
groupTrt   4.6610     0.2202   21.16 3.62e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6964 on 18 degrees of freedom
Multiple R-squared:  0.9818,        Adjusted R-squared:  0.9798
F-statistic: 485.1 on 2 and 18 DF,  p-value: < 2.2e-16
```

Next, load the **texreg** package. The output of the two models can be converted into a plain text table using the following command. The text output is shown below the R code.

```
R> library("texreg")
R> screenreg(list(m1, m2))
================================
             Model 1    Model 2
--------------------------------
(Intercept)   5.03 ***
             (0.22)
groupTrt     -0.37       4.66 ***
             (0.31)     (0.22)
groupCtl                 5.03 ***
                        (0.22)
--------------------------------
R^2           0.07       0.98
Adj. R^2      0.02       0.98
Num. obs.    20         20
================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

An arbitrary number of models can be handed over to the `texreg()`, `htmlreg()` or `screenreg()` function by enclosing them in a `list`. If only one model is converted, the `list` wrapper is not needed.

5.2. `texreg()`, table environments, and layout packages

The same table can be typeset in LaTeX by exchanging `screenreg()` for `texreg()`. In the following example, several additional arguments are demonstrated. The LaTeX output code is shown below the R code that generates the table. The resulting table is shown in Table˜3.

```
R> texreg(list(m1, m2), dcolumn = TRUE, booktabs = TRUE,
+       use.packages = FALSE, label = "tab:3", caption = "Two linear models.",
+       float.pos = "tb")
```

```
\begin{table}[tb]
\begin{center}
\begin{tabular}{l D{.}{.}{2.5}@{} D{.}{.}{2.5}@{} }
\toprule
            & \multicolumn{1}{c}{Model 1} & \multicolumn{1}{c}{Model 2} \\
\midrule
(Intercept) & 5.03^{***} &            \\
            & (0.22)     &            \\
groupTrt    & -0.37      & 4.66^{***} \\
            & (0.31)     & (0.22)     \\
groupCtl    &            & 5.03^{***} \\
            &            & (0.22)     \\
\midrule
R$^2$       & 0.07       & 0.98       \\
Adj. R$^2$  & 0.02       & 0.98       \\
Num. obs.   & 20         & 20         \\
\bottomrule
\multicolumn{3}{l}{\scriptsize{$^{***}p<0.001$, $^{**}p<0.01$, $^*p<0.05$}}
\end{tabular}
\caption{Two linear models.}
\label{tab:3}
\end{center}
\end{table}
```

The caption, label, and float position of the table are set explicitly. The **dcolumn** package is used to align coefficients at their decimal separators, and the **booktabs** package is employed to create professional horizontal rules. These arguments can be omitted if the two packages are not available (in this case, top, mid and bottom rules are replaced by conventional horizontal rules, and numeric values are horizontally aligned at the center of the column). The `\usepackage{}` declarations for the two packages are suppressed because the code has to be processed by `Sweave()`.

	Model 1	Model 2
(Intercept)	5.03***	
	(0.22)	
groupTrt	−0.37	4.66***
	(0.31)	(0.22)
groupCtl		5.03***
		(0.22)
R^2	0.07	0.98
Adj. R^2	0.02	0.98
Num. obs.	20	20

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

Table 3: Two linear models.

In order to omit the `\begin{table}` and `\end{table}` as well as the `\begin{center}` and `\end{center}` code, the `table` and `center` arguments can be used. If `table = FALSE` and `center = FALSE` are set, only the `tabular` environment is printed, not the `table` and `center` environments. In effect, the resulting table would be printed in-line in the text. Another reason for skipping the table environment could be to finetune the environment manually.

Alternatively, the argument `sideways = TRUE` can be used to rotate the table by 90 degrees using the `sidewaystable` environment in the **rotating** package (Rahtz and Fairbairns 2008) instead of the default `table` environment.

5.3. Custom names, omission of terms, and customization of coefficients

Another example demonstrates how the LaTeX code can be saved in an object using the `return.string` argument. The result is shown in Table˜4.

```
R> mytable <- texreg(list(m1, m2), label = "tab:4",
+      caption = "Bolded coefficients, custom notes, three digits.",
+      float.pos = "h", return.string = TRUE, bold = 0.05, stars = 0,
+      custom.note = "Coefficients with $p < 0.05$ in \\textbf{bold}.",
+      digits = 3, leading.zero = FALSE, omit.coef = "Inter")
```

The table can be printed to the R console later using the `cat()` function.

The example presented above introduced several additional arguments: `bold = 0.05` formats all coefficients with p˜values < 0.05 in bold; `stars = 0` means that only coefficients with p˜values < 0 are decorated with a star, which effectively suppresses all significance stars in the table because negative p˜values are not possible. Note that bold formatting cannot be used in combination with the `dcolumn` argument, so decimal mark alignment is switched off in Table˜4. The `booktabs` argument was also left out to show the difference between conventional horizontal lines in Table˜4 and `booktabs` rules in Table˜3. The `custom.note = "Coefficients with $p < 0.05$ in bold."` argument changes the significance note below the table. The `digits = 3` argument sets three decimal places, `leading.zero = FALSE` suppresses leading zeroes before the decimal separator, and `omit.coef = "Inter"` causes

	Model 1	Model 2
groupTrt	−.371	**4.661**
	(.311)	(.220)
groupCtl		**5.032**
		(.220)
R^2	.073	.982
Adj. R^2	.022	.980
Num. obs.	20	20

Coefficients with $p < 0.05$ in **bold**.

Table 4: Bolded coefficients, custom notes, three digits.

all rows containing the regular expression "Inter" to be skipped from the output (here: the "(Intercept)" term). Note that more complex regular expressions are possible; for example, `omit.coef = "(Trt)|(Ctl)"` would remove all rows matching either "Trt" or "Ctl".

5.4. Multiple model types, `single.row`, and custom names

Another example shows how **texreg** can deal with multiple *kinds* of models in the same table. The following code shows how ordinary least squares (OLS) and generalized least squares (GLS) models are matched in a single output table. The output is shown in Table˜5.

```
R> library("nlme")
R> m3 <- gls(follicles ~ sin(2 * pi * Time) + cos(2 * pi * Time), Ovary,
+      correlation = corAR1(form = ~ 1 | Mare))
R> table <- texreg(
+      list(m1, m3),
+      custom.coef.names = c(
+          "Intercept",
+          "Control",
+          "$\\sin(2 \\cdot \\pi \\cdot \\mbox{time})$",
+          "$\\cos(2 \\cdot \\pi \\cdot \\mbox{time})$"
+      ),
+      custom.model.names = c("OLS model", "GLS model"),
+      reorder.coef = c(1, 3, 4, 2),
+      caption = "Multiple model types, custom names, and single row.",
+      label = "tab:5",
+      stars = c(0.01, 0.001),
+      dcolumn = TRUE,
+      booktabs = TRUE,
+      use.packages = FALSE,
+      single.row = TRUE,
+      include.adjrs = FALSE,
+      include.bic = FALSE
+  )
```

Several interesting things can be noted. First, the `custom.coef.names` argument was used to

	OLS model	GLS model
Intercept	5.03 (0.22)**	12.22 (0.66)**
$\sin(2 \cdot \pi \cdot \text{time})$		-2.77 (0.65)**
$\cos(2 \cdot \pi \cdot \text{time})$		-0.90 (0.70)
Control	-0.37 (0.31)	
R^2	0.07	
Num. obs.	20	308
AIC		1571.45
Log Likelihood		-780.73

$^{**}p < 0.001$, $^{*}p < 0.01$

Table 5: Multiple model types, custom names, and single row.

relabel the coefficient rows. If there were repetitions of coefficient names in the `custom.coef.names` vector, **texreg** would try to conflate rows with identical names. In the case shown here, the two models are only matched on the intercept and the number of observations because all other rows have unique names.

Second, the custom names include LaTeX code. Within the code, in-line math code is allowed. LaTeX commands have to be marked by an additional backslash as an escape character, e.g., `\\pi` instead of `\pi`. Text within math blocks can be included in `\mbox{}` commands.

Third, custom names were also provided for the models. Using the `custom.model.names` argument, the default "Model 1", "Model 2" etc. are replaced by "OLS model" and "GLS model" in this case.

Fourth, the order of the coefficients was changed using the `reorder.coef` argument. The "Control" term was moved to the last position in the table.

Fifth, two significance levels (and, accordingly, a maximum of two stars) are used in the table. The `stars` argument takes at most four values, and when four values are specified, the lowest significance level (usually $0.05 \leq p < 0.1$) is denoted by the character specified in the `symbol` argument (by default a centered dot).

Sixth, the `single.row` argument causes the table to consume less vertical and more horizontal space because the standard errors are inserted right after the coefficients.

And seventh, the `include.adjrs` and `include.bic` arguments suppress the inclusion of the adjusted R^2 and BIC GOF statistics. These are model-specific arguments, which are defined in the `extract()` methods for '`lm`' and '`gls`'. More information about model-specific arguments can be found on the help page of the generic `extract()` function.

5.5. An example with robust standard errors

A common task in econometrics is to report robust—i.e., (Eicker-)Huber-White-corrected, or heteroskedasticity-consistent—standard errors using the **sandwich** (Zeileis 2004, 2006) and **lmtest** (Zeileis and Hothorn 2002) packages. The following code shows how this can be accomplished in combination with the **texreg** package. The resulting table is not reported here.

```
R> library("sandwich")
R> library("lmtest")
R> hc <- vcovHC(m2)
R> ct <- coeftest(m2, vcov = hc)
R> se <- ct[, 2]
R> pval <- ct[, 4]
R> texreg(m2, override.se = se, override.pvalues = pval)
```

The standard errors and p˜values are first extracted from the `hc` matrix and then handed over to the `texreg()` function using the `override.se` and `override.pvalues` arguments.

5.6. `htmlreg()`, Microsoft Word, knitr, and Markdown

The following examples show how the `htmlreg()` function can be used. The output code for these examples is not reported here.

The output of any `texreg()`, `htmlreg()` or `screenreg()` call can be written directly to a file by adding the `file` argument. This is especially handy because HTML files can be read by Microsoft Word if a ".doc" file extension is added.

If the table is exported to a file, it is advisable to include the full header information of the HTML file to make sure that Microsoft Word or other programs can parse the file. An example:

```
R> htmlreg(list(m1, m2, m3), file = "mytable.doc", inline.css = FALSE,
+      doctype = TRUE, html.tag = TRUE, head.tag = TRUE, body.tag = TRUE)
```

The `doctype` argument adds the document type declaration to the first line of the HTML document. The `inline.css = FALSE` argument causes the function to write cascading style sheets (the table formatting code) into the `<head>` tag rather than into the table code. The `head.tag` argument actually adds such a `<head>` tag to the document. Similarly, the `body.tag` argument wraps the table in a `<body>` tag, and the `html.tag` argument encloses both—the `<head>` and the `<body>` tag—in an `<html>` tag. In other words, these arguments create a whole HTML document rather than merely the table code. The resulting file can be read by Microsoft Word because the HTML file has a ".doc" extension.

The `htmlreg()` function also works well with the **knitr** package for dynamic report generation (Xie 2013c). The default arguments are compatible with **knitr** and HTML. In addition to HTML, **knitr** is also compatible with Markdown, a simplified markup language. **texreg** can work with Markdown as well, but an additional argument should be provided to make it work:

```
R> htmlreg(list(m1, m2, m3), star.symbol = "\\*", center = TRUE,
+      doctype = FALSE)
```

The `star.symbol = "\\*"` argument makes sure that Markdown does not interpret the significance stars as special Markdown syntax. The additional (and optional) `center = TRUE` argument centers the table horizontally on the page.

5.7. Confidence intervals instead of standard errors

Most model types implemented in **texreg** report standard errors and p˜values. However, some model types report confidence intervals by default. The **btergm** package (Leifeld, Cranmer,

	Model 1	Model 2	Model 3
(Intercept)	**5.03** (0.22)***	**5.03** [4.60; 5.46]*	
groupTrt	−0.37 (0.31)	−0.37 [−0.98; 0.24]	**4.66** [4.23; 5.09]*
groupCtl			**5.03** [4.60; 5.46]*
R^2	0.07	0.07	0.98
Adj. R^2	0.02	0.02	0.98
Num. obs.	20	20	20

***$p < 0.001$, **$p < 0.01$, *$p < 0.05$ (or 0 outside the confidence interval).

Table 6: Enforcing confidence intervals.

and Desmarais 2014) and the **lme4** package (Bates *et˜al.* 2013) are two examples. If confidence intervals are preferred to standard errors but they are not available by default, the `ci.force` argument allows conversion of standard errors to confidence intervals. The `ci.force.level` argument determines at which confidence level the interval should be computed.

A star is added to estimates where the confidence interval does not contain the value given by `ci.test` (to remove significance stars, `ci.test = NULL` can be set). When the `bold` argument is used in conjunction with confidence intervals, `bold` values greater than 0 cause **texreg** to print estimates in bold where the `ci.test` value is outside the confidence interval, regardless of the actual value of the `bold` argument (see Table˜6):

```
R> texreg(list(m1, m1, m2), ci.force = c(FALSE, TRUE, TRUE), ci.test = 0,
+       ci.force.level = 0.95, bold = 0.05, float.pos = "tb",
+       caption = "Enforcing confidence intervals.",
+       booktabs = TRUE, use.packages = FALSE, single.row = TRUE)
```

5.8. Coefficient plots

Finally, it is possible to display the results of statistical models using a coefficient plot (a forest plot applied to the estimates and confidence intervals or standard errors of a regression model; for another implementation, see Lander 2013). Using the `plotreg` function, one or multiple statistical model objects can be plotted. Most of the arguments of the `screenreg`, `texreg`, and `htmlreg` functions are also supported by the `plotreg` function. Where confidence intervals are not natively available, the `plotreg` function converts standard errors into confidence intervals. Alternatively, the `use.se` argument can be used to plot error bars with one or two standard errors from the point estimate. The diagram produced by the following code is shown in figure˜2:

```
R> plotreg(m1, custom.coef.names = c("Intercept", "Group Trt"))
```

6. Writing extensions for new models

The previous examples have demonstrated how the **texreg** package can be used to convert statistical model output into plain-text, LaTeX, HTML, and Markdown tables. Yet, this only

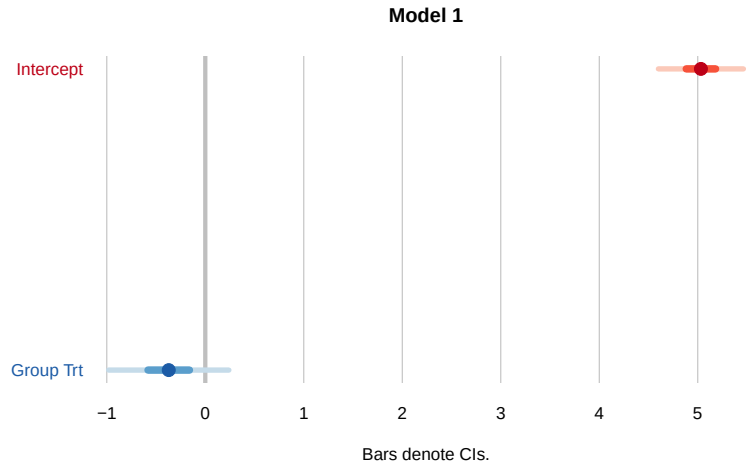

Figure 2: Result of the `plotreg` function. Significant estimates are colored in red.

works with model types known to **texreg**. Accordingly, this section shows how methods for new model types can be devised and registered.

6.1. Simple extensions

A custom extract function can be easily implemented. For any model type, there exists a function which extracts the relevant information from a model. For example, the `extract()` method for '`lm`' objects provides coefficients and GOF statistics for '`lm`' objects, the `extract()` method for '`ergm`' objects provides this information for '`ergm`' objects, etc.

To get an overview of the model type one is interested in, it is recommended to fit a model and examine the resulting object using the `str(model)` command, the `summary(model)` command, the `summary(model)$coef` command, and related approaches.

Any new extract function should retrieve the data shown in Table˜7 from a statistical model. Note that `pvalues` and `gof.decimal` are optional and can be omitted. Either the `se` slot or the `ci.low` and `ci.up` slots must contain values.

Once these data have been located and extracted, a '`texreg`' object can be created and returned to the `texreg()` function. The following code provides a simple example for '`lm`' objects:

```
R> extract.lm <- function(model) {
+     s <- summary(model)
+     names <- rownames(s$coef)
+     co <- s$coef[, 1]
+     se <- s$coef[, 2]
+     pval <- s$coef[, 4]
+
+     rs <- s$r.squared
+     adj <- s$adj.r.squared
```

Arguments	Description
`coef.names`	The names of the independent variables or coefficients.
`coef`	The actual coefficients. These values must be in the same order as the `coef.names`.
`se`	(*optional*) The standard errors, which will later be put in parentheses. These values must be in the same order as the `coef.names`.
`pvalues`	(*optional*) The p˜values. They are used to add significance stars. These values must be in the same order as the `coef.names`.
`ci.low`	(*optional*) Lower bounds of the confidence intervals. An alternative to the `se` slot.
`ci.up`	(*optional*) Upper bounds of the confidence intervals. An alternative to the `se` slot.
`gof.names`	The names of some GOF statistics to be added to the table. For example, the `extract()` method for '`lm`' objects extracts R^2, Adj. R^2 and Num. obs.
`gof`	A vector of GOF statistics to be added to the table. These values must be in the same order as the `gof.names`.
`gof.decimal`	(*optional*) A vector of logical (Boolean) values indicating for every GOF value whether the value should have decimal places in the output table. This is useful to avoid decimal places for the number of observations and similar count variables.

Table 7: Arguments of the `createTexreg()` function.

```
+    n <- nobs(model)
+
+    gof <- c(rs, adj, n)
+    gof.names <- c("R$^2$", "Adj.\\ R$^2$", "Num.\\ obs.")
+
+    tr <- createTexreg(
+        coef.names = names,
+        coef = co,
+        se = se,
+        pvalues = pval,
+        gof.names = gof.names,
+        gof = gof
+    )
+    return(tr)
+  }
```

First, the names of the model terms, the coefficient values, the standard errors, and the p˜values are extracted from the model or its summary (they can be computed if not available).

Second, various summary statistics and GOF measures are extracted from the model object (in this case: R^2, Adj. R^2 and Num. obs.) and saved in a `numeric` vector.

Third, the names of these statistics should be defined in a `character` vector. All vectors so far should have the same length.

Fourth, a new '`texreg`' object should be created, with the information extracted before included as arguments.

Fifth, the '`texreg`' object must be returned. This is necessary for the `texreg()` function to continue processing the model.

After writing a custom function, the function has to be registered as a method for the generic `extract()` function. In the above example, this can be achieved with the following code:

```
R> setMethod("extract", signature = className("lm", "stats"),
+      definition = extract.lm)
```

Assume, for instance, that an extension for '`clogit`' objects called `extract.clogit()` is written. The `clogit()` function (and the corresponding class definition) can be found in the **survival** package (Therneau and Grambsch 2000; Therneau 2012). Then the code above should be changed as follows:

```
R> setMethod("extract", signature = className("clogit", "survival"),
+      definition = extract.clogit)
```

After executing the definition of the function and the adjusted `setMethod()` command, **texreg** can be used with '`clogit`' models.

6.2. A complete example

The following code shows the complete `extract.lm()` function as included in the **texreg** package.

```
R> extract.lm <- function(model, include.rsquared = TRUE,
+      include.adjrs = TRUE, include.nobs = TRUE, ...) {
+
+    s <- summary(model, ...)
+    names <- rownames(s$coef)
+    co <- s$coef[, 1]
+    se <- s$coef[, 2]
+    pval <- s$coef[, 4]
+
+    gof <- numeric()
+    gof.names <- character()
+    gof.decimal <- logical()
+    if (include.rsquared == TRUE) {
+      rs <- s$r.squared
+      gof <- c(gof, rs)
+      gof.names <- c(gof.names, "R$^2$")
+      gof.decimal <- c(gof.decimal, TRUE)
+    }
+    if (include.adjrs == TRUE) {
+      adj <- s$adj.r.squared
+      gof <- c(gof, adj)
```

```
+      gof.names <- c(gof.names, "Adj.\\ R$^2$")
+      gof.decimal <- c(gof.decimal, TRUE)
+    }
+    if (include.nobs == TRUE) {
+      n <- nobs(model)
+      gof <- c(gof, n)
+      gof.names <- c(gof.names, "Num.\\ obs.")
+      gof.decimal <- c(gof.decimal, FALSE)
+    }
+
+    tr <- createTexreg(
+        coef.names = names,
+        coef = co,
+        se = se,
+        pvalues = pval,
+        gof.names = gof.names,
+        gof = gof,
+        gof.decimal = gof.decimal
+    )
+    return(tr)
+  }
R> setMethod("extract", signature = className("lm", "stats"),
+      definition = extract.lm)
```

In addition to the simple example code shown above, this function has several arguments, which can be used to include or exclude various GOF or summary statistics. Additional arguments can also be used in other contexts. For example, the user can decide whether random effect variances should be included in **texreg** tables of '`mer`' objects (from the **lme4** package, see Bates *et˜al.* 2013) by setting the `include.variance` argument. Similarly, the output of '`stergm`' models (Hunter, Handcock, Butts, Goodreau, and Morris 2008; Handcock, Hunter, Butts, Goodreau, Krivitsky, and Morris 2012) or '`hurdle`' or '`zeroinfl`' models (Zeileis, Kleiber, and Jackman 2008) can be typeset in two columns using the `beside` argument.

New extract functions and methods can be easily used locally. Once they work well, submission of new extract functions to the online forum of **texreg** is encouraged.

Existing functions can also be manipulated and overwritten locally in order to change the GOF statistics block.

7. Installation and help

It should be possible to install **texreg** using a simple command:

```
R> install.packages("texreg")
```

texreg is hosted on the R-Forge repository, which means that the most recent version can be installed with this command (often more recent than the CRAN version in the previous command):

```
R> install.packages("texreg", repos = "http://R-Forge.R-project.org")
```

The package can be updated to the most recent version by typing:

```
R> update.packages("texreg", repos = "http://R-Forge.R-project.org")
```

Alternatively, the source files and binaries can be downloaded from the **texreg** homepage (http://r-forge.r-project.org/projects/texreg/) and installed manually by entering something like

```
> R CMD INSTALL texreg_1.xx.tar.gz
```

(replace `xx` by the current version number) on the terminal (not the R terminal, but the command line of the operating system).

After loading the package, its help page can be displayed as follows:

```
R> help(package = "texreg")
```

More specific help on the `texreg()` command can be obtained by entering the following command once the package has been loaded:

```
R> help("texreg")
```

To get an overview of currently implemented extract functions and methods, one of these two commands can be used.

```
R> help("extract")
R> help("extract-methods")
```

If all else fails, more help can be obtained from the homepage of the **texreg** package. Questions can be posted to a public forum at http://r-forge.r-project.org/projects/texreg/.

Acknowledgments

The author would like to thank Oleg Badunenko, Tom Carsey, S. Q. Chang, Skyler Cranmer, Sebastian Daza, Christopher Gandrud, Lena Koerber, Johannes Kutsam, Fabrice Le Lec, Florian Oswald, Markus Riester, Francesco Sarracino, Giorgio Spedicato, Matthieu Stigler, Sebastian Ugbaje, Gábor Uhrin, Antoine Vernet, Yanghao Wang, and Yihui Xie for their valuable input and ideas.

References

Bates D, Maechler M, Bolker B, Walker S (2013). ***lme4****: Linear mixed-effects models using Eigen and S4*. R package version 1.0-4, URL http://CRAN.R-project.org/package=lme4.

Carlisle D (2001). *The* ***dcolumn*** *Package*. LaTeX package version 1.06, URL http://www.CTAN.org/pkg/dcolumn/.

Dahl DB (2012). ***xtable***: *Export Tables to LaTeX or HTML*. R package version 1.7-0, URL http://CRAN.R-project.org/package=xtable.

Elff M (2012). ***memisc***: *Tools for Management of Survey Data, Graphics, Programming, Statistics, and Simulation*. R package version 0.95-38, URL http://CRAN.R-project.org/package=memisc.

Fear S (2005). ***booktabs***: *Publication Quality Tables in LaTeX*. LaTeX package version 1.61803, URL http://www.CTAN.org/pkg/booktabs/.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Krivitsky PN, Morris M (2012). ***ergm***: *Fit, Simulate and Diagnose Exponential-Family Models for Networks*. R package version 3.0-999, URL http://CRAN.R-project.org/package=ergm.

Hlavac M (2013). ***stargazer***: *LaTeX Code for Well-Formatted Regression and Summary Statistics Tables*. R package version 3.0.1, URL http://CRAN.R-project.org/package=stargazer.

Hoffmann CW (2012). ***cwhmisc***: *Miscellaneous Functions for Maths, Plotting, Printing, Statistics, Strings, and Tools*. R package version 3.0, URL http://CRAN.R-project.org/package=cwhmisc.

Hunter DR, Handcock MS, Butts CT, Goodreau SM, Morris M (2008). "**ergm**: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks." *Journal of Statistical Software*, **24**(3), 1–29. URL http://www.jstatsoft.org/v24/i03/.

Johnson PE (2012). ***rockchalk***: *Regression Estimation and Presentation*. R package version 1.6.2, URL http://CRAN.R-project.org/package=rockchalk.

Lamport L (1986). *LaTeX User's Guide and Document Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Lander JP (2013). ***coefplot***: *Plots Coefficients from Fitted Models*. R package version 1.2.0, URL http://CRAN.R-project.org/package=coefplot.

Leifeld P (2013). "**texreg**: Conversion of Statistical Model Output in R to LaTeX and HTML Tables." *Journal of Statistical Software*, **55**(8), 1–24. URL http://www.jstatsoft.org/v55/i08/.

Leifeld P (2014). ***texreg***: *Conversion of Statistical Model Output in R to LaTeX and HTML Tables*. R package version 1.32, URL http://CRAN.R-project.org/package=texreg.

Leifeld P, Cranmer S, Desmarais B (2014). ***btergm***: *Temporal Exponential Random Graph Models by Bootstrapped Pseudolikelihood*. R package version 0.35, URL http://r-forge.r-project.org/projects/btergm/.

Leisch F (2002). "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W˜Härdle, B˜Rönz (eds.), *COMPSTAT 2002 – Proceedings in Computational Statistics*, pp. 575–580. Physica-Verlag, Heidelberg.

Maechler M, others (2012). ***sfsmisc***: *Utilities from Seminar für Statistik ETH Zurich*. R package version 1.0-21, URL http://CRAN.R-project.org/package=sfsmisc.

Malecki M (2012). ***apsrtable**: Model-Output Formatter for Social Science*. R package version 0.8-8, URL http://CRAN.R-project.org/package=apsrtable.

Owen M, Imai K, King G, Lau O (2013). ***Zelig**: Everyone's Statistical Software*. R package version 4.1-3, URL http://CRAN.R-project.org/package=Zelig.

Rahtz S, Fairbairns R (2008). ***rotating**: A Package for Rotated Objects in LaTeX*. LaTeX package version 2.16b, URL http://www.CTAN.org/pkg/rotating.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Therneau T (2012). ***survival**: A Package for Survival Analysis in S*. R package version 2.36-14, URL http://CRAN.R-project.org/package=survival.

Therneau TM, Grambsch PM (2000). *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag, New York.

Venables WN, Smith DM, R Core Team (2013). *An Introduction to R. Notes on R: A Programming Environment for Data Analysis and Graphics*. URL http://CRAN.R-project.org/doc/manuals/R-intro.html.

Xie Y (2013a). *Dynamic Documents with R and **knitr***. Chapman and Hall/CRC.

Xie Y (2013b). "**knitr**: A Comprehensive Tool for Reproducible Research in R." In V˜Stodden, F˜Leisch, RD˜Peng (eds.), *Implementing Reproducible Computational Research*. Chapman and Hall/CRC.

Xie Y (2013c). ***knitr**: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.1, URL http://CRAN.R-project.org/package=knitr.

Zeileis A (2004). "Econometric Computing with HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. URL http://www.jstatsoft.org/v11/i10/.

Zeileis A (2006). "Object-Oriented Computation of Sandwich Estimators." *Journal of Statistical Software*, **16**(9), 1–16. URL http://www.jstatsoft.org/v16/i09/.

Zeileis A, Hothorn T (2002). "Diagnostic Checking in Regression Relationships." ***R** News*, **2**(3), 7–10. URL http://CRAN.R-project.org/doc/Rnews/.

Zeileis A, Kleiber C, Jackman S (2008). "Regression Models for Count Data in R." *Journal of Statistical Software*, **27**(8), 1–25. URL http://www.jstatsoft.org/v27/i08/.

Affiliation:

Philip Leifeld
University of Konstanz
Box 216
78457 Konstanz, Germany
E-mail: philip.leifeld@uni-konstanz.de
URL: http://www.philipleifeld.de