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

Philip Leifeld <philip.leifeld@uni-konstanz.de>

April 15, 2013

1 Motivation

The `texreg` package for the statistical computing environment `R` was designed to convert regression model output from multiple models into tables for inclusion in LaTeX or HTML documents. It is an alternative to packages like `xtable`, `apsrtable`, `outreg`, `stargazer` and `memisc`, which can also convert `R` tables to LaTeX tables. Only a subset of these packages is able to merge multiple regression models in a single table. Those packages which can do this do not support important model types such as `lme` or `mer` (linear mixed effects models) and `ergm` objects (exponential random graph models from the statnet suite of packages), or are not customizable and extendable. `texreg`, in contrast, accepts these model types and can also merge multiple models in a single table. Currently supported model types are listed in table 1. New model types can be easily implemented (see section 6). `texreg` can be used within Sweave and knitr. LaTeX packages for creating fancy tables, like `dcolumn` or `booktabs`, are supported.

Beside LaTeX output, `texreg` can also export nicely formatted tables to MS Word files, HTML files (which can be viewed in any web browser), or it can print nicely formatted regression tables directly to the screen (that is, to the `R` console) for easier model comparison.

2 Installation

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

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

The most recent version can always be installed with this command (usually more recent than the CRAN version in the previous command):

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

If this is not possible for some reason, the source files and binaries can be downloaded from http://r-forge.r-project.org/projects/texreg/ (click on "R packages"). To load the package in `R` once it has been installed, enter the following command:

```
> library(texreg)
```

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

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

If the file is not available on the R-Forge repository, you can try to download it from the R-Forge project homepage (http://r-forge.r-project.org/projects/texreg/; click on "R packages") and install it 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 normal command line of your operating system).

*The author would like to thank S. Q. Chang, Skyler Cranmer, Sebastian Daza, Christopher Gandrud, Lena Koerber, Johannes Kutsam, Fabrice Le Lec, Francesco Sarracino, Matthieu Stigler, Sebastian Ugbaje, Gábor Uhrin, Yanghao Wang, and Yihui Xie for valuable input.

Class	Package	Added	Description
aftreg	eha	2013-03-23	Accelerated failure time regression
betareg	betareg	2013-03-13	Beta regression for rates and proportions
clm	ordinal	2012-10-12	Cumulative link models
clogit	survival	2012-09-30	Conditional logistic regression
coxph	survival	2012-10-14	Cox proportional hazard models
coxph.penal	survival	2012-12-04	Cox proportional hazard models with penalty splines
dynml	dynlm	2013-02-14	Time series regression with "ts" data
ergm	ergm	2012-06-18	Exponential random graph models
gam	mgcv	2013-03-13	Generalized additive models
gee	gee	2012-10-14	Generalized estimation equation
glm	stats	2012-06-19	Generalized linear models
glmerMod	lme4 (new)	2012-10-09	Generalized linear mixed models
gls	nlme	2012-06-19	Generalized least squares
gmm	gmm	2013-02-06	Generalized method of moments estimation
ivreg	AER	2013-03-13	Instrumental-variable regression using 2SLS
hurdle	pscl	2013-03-13	Hurdle regression models for count data
lm	stats	2012-06-19	Ordinary least squares
lme	nlme	2012-06-19	Linear mixed-effects models
lmerMod	lme4 (new)	2012-10-08	Linear mixed-effects models
lmrob	robustbase	2012-11-12	MM-type estimators for linear models
lnam	sna	2012-10-07	Linear network autocorrelation models
mer	lme4 (old)	2012-10-08	Linear mixed-effects models
multinom	nnet	2013-03-13	Multinomial log-linear models
negbin	MASS	2012-10-15	Negative binomial generalized linear models
nlmerMod	lme4 (new)	2012-10-09	Nonlinear mixed-effects models
lrm	rms, Design	2012-07-04	Logistic regression models
phreg	eha	2013-03-23	Parametric proportional hazards regression
plm	plm	2012-08-01	Linear models for panel data
pmg	plm	2012-08-01	Linear panel models with heterogeneous coefficients
polr	MASS	2012-10-12	Ordered logistic or probit regression
Relogit	Zelig	2012-10-14	Rare events logistic regression
rem.dyad	relevent	2013-02-28	Relational event model for dyadic data
rlm	MASS	2012-11-12	Robust fitting of linear models
rq	quantreg	2012-08-01	Quantile regression models
sclm	ordinal	2012-10-12	Cumulative link models
simex	simex	2012-10-15	SIMEX algorithm for measurement error models
stergm	tergm	2012-10-23	Separable temporal exponential random graph models
survreg	survival	2013-03-13	Parametric survival regression models
survreg.penal	survival	2013-03-13	Frailty survival models
svyglm	survey	2012-10-14	Survey-weighted generalized linear models
systemfit	systemfit	2012-10-03	Linear structural equations
tobit	AER	2012-10-15	Tobit regression models for censored data
weibreg	eha	2013-03-23	Weibull regression
zeroinfl	pscl	2013-03-13	Zero-inflated regression models

Table 1: List of currently supported model types

3 Getting help

This R package vignette is part of the texreg package. It can be displayed in R by entering the command:

```
> vignette("texreg")
```

The help page of the package can be displayed as follows:

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

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

```
> help(texreg)
```

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/.

4 texreg examples

Suppose you fit two simple OLS models. The following example was taken from the lm() help file.

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

The coefficients, standard errors, p values etc. can be displayed as follows:

```
> 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
```

Now it is fairly tedious to copy every single coefficient and standard error to a LaTeX table when you design your academic paper. To improve the situation, the following commands can do this automatically (the LaTeX output code is shown below the R code, and the resulting table is shown in table 2):

```
> library(texreg)
> texreg(m2, booktabs = TRUE, dcolumn = TRUE)
```

```
\usepackage{booktabs}
\usepackage{dcolumn}

\begin{table}
\begin{center}
\begin{tabular}{l D{.}{.}{2.5}@{} }
\toprule
           & \multicolumn{1}{c}{Model 1} \\
\midrule
groupCtl   & 5.03^{***} \\
           & (0.22)     \\
groupTrt   & 4.66^{***} \\
           & (0.22)     \\
\midrule
R$^2$      & 0.98       \\
Adj. R$^2$ & 0.98       \\
Num. obs.  & 20         \\
\bottomrule
\multicolumn{2}{l}{\scriptsize{\textsuperscript{***}$p<0.001$,
  \textsuperscript{**}$p<0.01$,
  \textsuperscript{*}$p<0.05$}}
\end{tabular}
\caption{Statistical models}
\label{table:coefficients}
\end{center}
\end{table}
```

Note that the `booktabs` and `dcolumn` arguments were used to create more fancy tables than the default behavior using the `booktabs` and the `dcolumn` LaTeX packages (for nicer top and bottom rules, and for decimal point alignment of coefficients and standard errors). The resulting table is printed directly to the `R` console for easy copy & paste. It can also be returned as a character string and saved in an object, say `tab`, by adding the `return.string=TRUE` argument. This way, it can be later printed again using the `cat()` function:

```
> tab <- texreg(m2, return.string = TRUE)
> cat(tab)
```

The `texreg` command also accepts multiple models as a `list` and merges them in a table. The output of the following command is shown in table 3. Note the additional `caption` argument to set the caption of the table.

```
> texreg(list(m1, m2), booktabs = TRUE, dcolumn = TRUE,
+     caption = "Multiple models")
```

Tables˜2 and˜3 consume more horizontal space than is actually necessary. This is due to the long note regarding significance thresholds for the p values. To avoid this problem, the note can be replaced by a shorter note using the `custom.note` argument.

The `texreg` package contains many customizations. Among other options, the `use.packages` argument can be used to switch off package loading at the beginning of the table code. Using the `label` argument, the label of the table can be set. In a similar way, the `caption` argument takes care of the caption. Activating the `scriptsize` option prints the table in a smaller font size. The `sideways` argument rotates the table by 90 degrees and uses the `rotating` package and the `sidewaystable` environment. The position of the table on the page or in the document can be specified using the `float.pos` argument. The `custom.names` and `model.names` arguments can be used to specify the names of the model terms and the models, respectively. An example:

```
> texreg(list(m1, m2), booktabs = TRUE, dcolumn = TRUE,
+     use.packages = FALSE, label = "tab:3",
```

	Model 1
groupCtl	5.03***
	(0.22)
groupTrt	4.66***
	(0.22)
R^2	0.98
Adj. R^2	0.98
Num. obs.	20

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

Table 2: Statistical models

	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: Multiple models

```
+     caption = "Custom coefficient and model names",
+     custom.coef.names = c("(Intercept)",  "Treatment", "Control"),
+     custom.model.names = c("First model", "Second model"),
+     float.pos = "p")
```

The output of this command is shown as table 4. Another argument is `table`. By deactivating it, the plain `tabular` environment is printed, and the whole table environment and header is omitted from the output. This may be useful for integrating tables in Sweave, or for tweaking the floating environment of the table. The `no.margin` argument can be used to control the cell spacing of the table. If set to `TRUE`, regular margins are used. By default, no margins are used in order not to waste any horizontal space on the page.

The `texreg` package can also handle `ergm` objects (that is, exponential random graph models, which are used in social network analysis). Here is an example: the following code creates a network matrix.

```
> mat <- rbinom(400, 1, 0.16)  #create a matrix
> mat <- matrix(mat, nrow = 20)
```

Using the `network` package, the matrix can be converted into a network object. The `ergm()` command from the `ergm` package can be used to fit some models:

```
> library(network)
> library(ergm)
> nw <- network(mat)
> m4 <- ergm(nw ~ edges)
> m5 <- ergm(nw ~ edges + mutual)
> m6 <- ergm(nw ~ edges + mutual + twopath)
```

The `texreg` command can then be used to create a table with the coefficients. The `summary` function for `ergm` objects attaches centered dots to weakly significant model terms by default (that is, a $\cdot$ is used to denote p values between 0.05 and 0.1). This behavior can be imitated by modifying the `stars` argument and adding a fourth significance level as follows:

```
> texreg(list(m4, m5, m6), use.packages = FALSE, label = "tab:4",
+     booktabs = TRUE, dcolumn = TRUE, float.pos = "p",
+     caption = "Centered dots for weak significance",
+     stars = c(0.001, 0.01, 0.05, 0.1))
```

Table 5 shows the result of this command. Note that the legend under the table is now extended, but the model terms do not feature any p values between 0.05 and 0.1 in this case.

Most academic journals require tables where the coefficient and the standard error are stored in two separate rows of the table, as shown in tables 2 to 5. In some situations, however, it makes sense to accommodate them in a single row. The `single.row` argument can take care of this:

```
> texreg(list(m4, m5, m6), use.packages = FALSE, label = "tab:5",
+     booktabs = TRUE, dcolumn = TRUE, single.row = TRUE,
+     float.pos = "p", caption = "In a single row")
```

The result is shown in table 6. Note the difference between tables 5 and 6.

The `texreg` command can also combine the output of different model types in a single table. Consider the following example of an `lm` object, an `lme` (linear mixed-effects) model and an `ergm` object:

```
> library(nlme)
> m3 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)

> texreg(list(m3, m2, m6), label = "tab:6", use.packages = FALSE,
+     booktabs = TRUE, dcolumn = TRUE,
+     caption = "Mixing different kinds of models")
```

	First model	Second model
(Intercept)	5.03***	
	(0.22)	
Treatment	−0.37	4.66***
	(0.31)	(0.22)
Control		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 4: Custom coefficient and model names

	Model 1	Model 2	Model 3
edges	−1.60***	−1.49***	−1.28*
	(0.14)	(0.16)	(0.55)
mutual		−0.78	−0.78
		(0.64)	(0.64)
twopath			−0.04
			(0.09)
AIC	346.56	346.86	348.62
BIC	350.50	354.74	360.44
Log Likelihood	−172.28	−171.43	−171.31

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$, $^{\cdot}p < 0.1$

Table 5: Centered dots for weak significance

	Model 1	Model 2	Model 3
edges	−1.60 (0.14)***	−1.49 (0.16)***	−1.28 (0.55)*
mutual		−0.78 (0.64)	−0.78 (0.64)
twopath			−0.04 (0.09)
AIC	346.56	346.86	348.62
BIC	350.50	354.74	360.44
Log Likelihood	−172.28	−171.43	−171.31

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

Table 6: In a single row

	Model 1	Model 2	Model 3
(Intercept)	17.71***		
	(0.83)		
age	0.66***		
	(0.06)		
SexFemale	−2.32**		
	(0.76)		
groupCtl		5.03***	
		(0.22)	
groupTrt		4.66***	
		(0.22)	
edges			−1.28*
			(0.55)
mutual			−0.78
			(0.64)
twopath			−0.04
			(0.09)
AIC	447.51		348.62
BIC	460.78		360.44
Log Likelihood	−218.76		−171.31
Num. obs.	108	20	
R^2		0.98	
Adj. R^2		0.98	

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

Table 7: Mixing different kinds of models

The output is shown in table 7. Note that different model types may report different kinds of goodness-of-fit statistics at the bottom of the table. These can be switched on or off (see `?extract` for details).

Many people use robust standard errors. To include include them in a `texreg` table, the original standard errors can be replaced and new, custom values can be handed over. To do this, the argument `override.se` can be used. The argument expects a list of vectors, with one vector of standard errors for each model (which means that there should be as many elements in the list as there are models). Beside standard errors, there are similar arguments for p values (`override.pval`) and coefficients (`override.coef`).

5 htmlreg and screenreg

Tables can also be converted into HTML code instead of LaTeX code using the following command:

```
> htmlreg(list(m3, m2, m6))
```

The output of either of the two commands can be written directly to a file by adding the `file` argument. This is especially handy because HTML files can be read by MS Word. So it is possible to use the `texreg` package not only with LaTeX, but also with MS Office. If the table is exported to a file, it is advisable to include the full header information of the HTML file. An example:

```
> htmlreg(list(m3, m2, m6), file = "mytable.doc", inline.css = FALSE,
+     doctype = TRUE, html.tag = TRUE, head.tag = TRUE,
+     body.tag = TRUE)
```

The `htmlreg()` function works well with the `knitr` package for dynamic HTML report generation. The default arguments should work well 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: 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. Here is an example:

```
> htmlreg(list(m3, m2, m6), star.symbol = "\\*", center = TRUE)
```

Finally, there is another function, which can print tables to the R console. The command will nicely arrange the spaces etc. of your tables and will greatly facilitate model comparison (as a substitute of the default `summary` function). To show the three models side-by-side in your R console, type the following code:

```
> screenreg(list(m3, m2, m6))
```

```
================================================
                Model 1      Model 2    Model 3
------------------------------------------------
(Intercept)       17.71 ***
                  (0.83)
age                0.66 ***
                  (0.06)
SexFemale         -2.32 **
                  (0.76)
groupCtl                      5.03 ***
                             (0.22)
groupTrt                      4.66 ***
                             (0.22)
edges                                    -1.28 *
                                         (0.55)
mutual                                   -0.78
                                         (0.64)
twopath                                  -0.04
                                         (0.09)
------------------------------------------------
AIC              447.51                  348.62
BIC              460.78                  360.44
Log Likelihood  -218.76                 -171.31
Num. obs.        108         20
R^2                           0.98
Adj. R^2                      0.98
================================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

This is especially useful to prepare a table on screen and eventually export it to LaTeX or HTML.

6 Creating templates for new model types

Implementing new kinds of statistical models is fairly easy (if you know how to modify R functions). For any model type, there exists a function which extracts the relevant information from a model. For example, `extract.lm()` provides coefficients and goodness-of-fit statistics for `lm` objects, `extract.ergm()` provides this information for `ergm` objects, etc.

You can get an overview of the model type you are interested in by fitting a model and examining the resulting object using the `str(model)` command, the `summary(model)` command, the `summarymodel$coef` command, and related commands. Any new extract function must retrieve the following data from a statistical model:

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 The standard errors, which will later be put in parentheses. These values must be in the same order as the `coef.names`.

pvalues The p values (*optional*). They are used to add significance stars. These values must be in the same order as the `coef.names`.

gof.names The names of some goodness-of-fit statistics to be added to the table. For example, the `extract.lm()` function extracts R^2, Adj. R^2 and Num. obs.

gof A vector of goodness-of-fit statistics to be added to the table. These values must be in the same order as the `gof.names`.

gof.decimal A vector of logical (boolean) values indicating for every GOF value whether the value should have decimal places in the output table (*optional*). This is useful to avoid decimal places for the number of observations and similar count variables.

Once you have located all these data, you can create a `texreg` object and return it to the `texreg()` function. The following code provides an example. It shows the `extract.lm()` function:

```
extract.lm <- function(model, include.rsquared = TRUE, include.adjrs = TRUE,
    include.nobs = TRUE, ...) {

  s <- summary(model, ...)                    # save the summary statistics

  names <- rownames(s$coef)                   # extract coefficient names
  co <- s$coef[, 1]                           # extract the coefficient values
  se <- s$coef[, 2]                           # extract the standard errors
  pval <- s$coef[, 4]                         # extract the p-values

  rs <- s$r.squared                           # extract R-squared
  adj <- s$adj.r.squared                      # extract adjusted R-squared
  n <- nobs(model)                            # extract number of observations

  gof <- numeric()                            # create a vector for the GOFs
  gof.names <- character()                    # create a vector for the GOF names
  gof.decimal <- logical()                    # should the GOFs have dec. places?
  if (include.rsquared == TRUE) {             # if the user wants r-squared...
    gof <- c(gof, rs)                         # add it to the GOF list
    gof.names <- c(gof.names, "R$^2$")        # add its name to the list
    gof.decimal <- c(gof.decimal, TRUE)       # and make sure it has dec. places
  }
  if (include.adjrs == TRUE) {                # same for adjusted r-squared
    gof <- c(gof, adj)
    gof.names <- c(gof.names, "Adj.\ R$^2$")
    gof.decimal <- c(gof.decimal, TRUE)
  }
  if (include.nobs == TRUE) {                 # same for number of observations
    gof <- c(gof, n)
    gof.names <- c(gof.names, "Num.\ obs.")
    gof.decimal <- c(gof.decimal, FALSE)      # but these are integer numbers
  }

  tr <- createTexreg(                         # create a texreg object
      coef.names = names,
      coef = co,
      se = se,
      pvalues = pval,                         # p-values are only needed when
```

```
      gof.names = gof.names,                # signif. stars shall be printed
      gof = gof,
      gof.decimal = gof.decimal             # (optional)
  )
  return(tr)                                # return texreg object to texreg
}
```

After writing a custom function, the function has to be registered. In other words, you have to tell the more general `extract` function that objects of the new class should be handled by using your custom function. In the above example, this is achieved with the following code:

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

Let's say you have written an extension for `clogit` objects called `extract.clogit()`. The `clogit` command (and the corresponding class definition) can be found in the `survival` package. Then you would have to adjust the code above as follows:

```
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 your models.

If you write a new extract function and a `setMethod` configuration, it would be very helpful to post them in the forum (see section 3) in order to let other users profit from it. If it works and if you can provide a self-contained example, the code can be implemented in a future version of `texreg`. Please make sure that you do not modify anything else in the code, and that you stick to the formatting rules used in the remaining file; otherwise comparison with the original may be difficult. Please send an inquiry if you are interested in joining the `texreg` project and working directly on the code.

7 How to obtain the source code

If you would like to inspect the `texreg` source code in order to develop your own extensions, you can download the `.tar.gz` file from the repository homepage. To do this, you can either search the list of R-Forge contributions (http://download.r-forge.r-project.org/src/contrib/) for `texreg`, or click on the "R packages" link on the `texreg` package homepage at R-Forge (http://r-forge.r-project.org/projects/texreg/). Make sure you download the `texreg` file with the `.tar.gz` extension, open this compressed file (e.g., using 7Zip if you are on Windows), and open the `texreg.R` file in the `R/` directory.