TEXreg: Conversion of R regression output to LATEX tables

Philip Leifeld <philip.leifeld@eawag.ch>

October 9, 2012

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 documents. It is an alternative to packages like `xtable`, `apsrtable`, `outreg` and `memsic`, 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). TEXreg, in contrast, accepts these model types and can also merge multiple models in a single table. Currently supported model types are listed in section 2. New model types can be easily implemented (see section 6). TEXreg can be used within Sweave. LATEX packages for creating fancy tables, like `dcolumn` or `booktabs`, are supported.

2 Supported model types

Class	Package	Contributed by	Added	Description
`clogit`	`survival`	Sebastian Daza	2012-09-30	Conditional logistic regression
`ergm`	`ergm`	Philip Leifeld	2012-06-18	Exponential random graph models
`glm`	`stats`	Philip Leifeld	2012-06-19	Generalized linear models
`glmerMod`	`lme4` (new)	Philip Leifeld	2012-10-09	Generalized linear mixed models
`gls`	`nlme`	Philip Leifeld	2012-06-19	Generalized least squares
`lm`	`stats`	Philip Leifeld	2012-06-19	Ordinary least squares
`lme`	`nlme`	Philip Leifeld	2012-06-19	Linear mixed-effects models
`lmerMod`	`lme4` (new)	Philip Leifeld	2012-10-08	Linear mixed-effects models
`lnam`	`sna`	Philip Leifeld	2012-10-07	Linear network autocorrelation models
`mer`	`lme4` (old)	Philip Leifeld	2012-10-08	Linear mixed-effects models
`nlmerMod`	`lme4` (new)	Philip Leifeld	2012-10-09	Nonlinear mixed-effects models
`lrm`	`rms`, `Design`	Fabrice Le Lec	2012-07-04	Logistic regression models
`plm`	`plm`	Lena Koerber	2012-08-01	Linear models for panel data
`pmg`	`plm`	Lena Koerber	2012-08-01	Linear panel models with heterogeneous coefficients
`rq`	`quantreg`	Lena Koerber	2012-08-01	Quantile regression models
`systemfit`	`systemfit`	Johannes Kutsam	2012-10-03	Linear structural equations

Table 1: List of currently supported model types

3 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).

4 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/.

5 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)
> table <- texreg(m2)
```

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

\begin{table}
\begin{center}
\begin{tabular}{l D{.}{.}{3.5} @{}}
\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
\vspace{-2mm}\\
\multicolumn{2}{l}{\textsuperscript{***}$p<0.01$, \textsuperscript{**}$p<0.05$, \textsuperscript{*
\end{tabular}
\end{center}
\caption{Statistical models}
\label{table:coefficients}
\end{table}
```

The table is saved in the object `table`. Moreover, it is printed directly to the R console for easy copy & paste. In order to print it to the R console again, the following command can be used:

```
> cat(table)
```

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.

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

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

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

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

Table 3: Statistical models

```
> table <- texreg(list(m1,m2))
```

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:

```
> table <- texreg(list(m1, m2), use.packages=FALSE, label="tab:3",
+   caption="My regression table", scriptsize=TRUE,
+   custom.names=c("(Intercept)", "Treatment", "Control"),
+   model.names=c("First model", "Second model"), float.pos="b")
```

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.

TEXreg employs functions from the `booktabs` and `dcolumn` packages to generate beautiful tables. If these packages should not be used when generating tables, the arguments `booktabs` and `dcolumn`, respectively, can be set to `FALSE`.

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. Switching on `strong.signif` returns the significance levels used by the `ergm` package (three stars for p values smaller than 0.001 etc.) instead of using conventional significance stars:

```
> table <- texreg(list(m4, m5, m6), use.packages=FALSE, label="tab:4",
+   scriptsize=FALSE, strong.signif=TRUE)
```

Table 5 shows the result of this command.

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:

```
> table <- texreg(list(m4, m5, m6), use.packages=FALSE, label="tab:5",
+   single.row=TRUE)
```

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)

> table <- texreg(list(m3, m2, m6), label="tab:6", use.packages=FALSE)
```

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.

	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

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

Table 4: My regression table

	Model 1	Model 2	Model 3
edges	-1.91^{***}	-2.01^{***}	-1.92^{***}
	(0.15)	(0.18)	(0.55)
mutual		0.65	0.66
		(0.55)	(0.56)
twopath			-0.02
			(0.11)
AIC	294.13	294.87	296.83
BIC	298.07	302.75	308.65
Log Likelihood	-146.06	-145.43	-145.41

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

Table 5: Statistical models

	Model 1	Model 2	Model 3
edges	$-1.91\ (0.15)^{***}$	$-2.01\ (0.18)^{***}$	$-1.92\ (0.55)^{***}$
mutual		$0.65\ (0.55)$	$0.66\ (0.56)$
twopath			$-0.02\ (0.11)$
AIC	294.13	294.87	296.83
BIC	298.07	302.75	308.65
Log Likelihood	-146.06	-145.43	-145.41

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

Table 6: Statistical models

	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			-1.92^{***}
			(0.55)
mutual			0.66
			(0.56)
twopath			-0.02
			(0.11)
AIC	447.51		296.83
BIC	460.78		308.65
Log Likelihood	-218.76		-145.41
Num. obs.	108	20	
R^2		0.98	
Adj. R^2		0.98	

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

Table 7: Statistical models

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

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

  gof <- c(rs, adj, n)                     # put the GOFs in a vector
  gof.names <- c("R$^2$", "Adj. R$^2$", "Num. obs.")   #also put names in vector
  decimal.places <- c(TRUE, TRUE, FALSE)   # which GOFs will have decimal places

  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,                 # significance stars shall be printed
      gof=gof,
      gof.decimal=decimal.places           # (optional)
  )
  return(tr)                               # return the 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 4) 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.