left logo

Test LaTeX file for outreg regressions

right logo

Paul Johson, CRMDA, pauljohn@ku.edu

Guide No: 0

Keywords: R, rockchalk, outreg, LaTeX, regression tables

See https://crmda.ku.edu/guides for updates.

Oct. 7, 2018

Abstract

rockchalk version 1.8.120 introduced decimal-centering of columns in the outreg table-writer for regression models. This has been the most widely requested feature for a while. 10 years ago, I declined to do it because I was afraid I would make tables that would fail in ordinary documents. However, in 2018 I've learned some tricks so I've implemented them and I believe it is adequate. The tables that were OK are still OK, and the ones with dcolumn mostly work as well. I am asking friends and colleagues to explore and test.

This document might become a vignette in rockchalk, but right now it is a plea for insight & advice. Right now, this essay is not in the package, but it is in the repository: https://github.com/pauljohn32/rockchalk, in a folder named "vignette-test".

The problem I see now, where I need your help, is that the dcolumn setup is either giving columns that are too wide, or too narrrow (causing output to overlap). In version 1.8.120 of package, I have the "too wide" problem so some multi-model tables run off the edge of the page. In version 1.8.121, I tighten down columns, but end up with overlapping results. You can see that in the PDF you are reading (see the F rows in Table 19) Because the "special" F markup we decided on is so wide, it breaks the dcolumn spacing.

Contents

The rockchalk package for R (R Core Team, 2018) was created to facilitate authors in stats classes who need to make presentable tables and graphs for regression models. It does not aspire to be a comprehensive table-maker, mainly because R package writers are not consistent in their methodology. Procedures called "regression" within R's base have a certain internal consistency, but user-contributed packages do not. The `outreg` function works for regression models that I use regularly, usually in the base of R, like `lm`, `glm`, and for `lme4` objects. Other regression packages may work. I've tried to be somewhat general in the design.

The new argument `dcolumn` has a default of `FALSE`, meaning there will be no effect, the outreg table will be the same as it always was. However, if the outreg command sets the argument `dcolumn=TRUE`, then the LaTeX package dcolumn will be used. This will succeed only if the user and inserts `\usepackage{dcolumn}` in the preamble. I believe this will "just" work.

In the past, my concern about the dcolumn package was that it turned all of the text in columns into math mode italics, so many tables either had unexpected style changes or failed to compile altogether. The solution is to protect all text from that transformation. I do that in a way that does not alter the behavior of outreg tables for users that don't want decimal-aligned columns. But it also prevents the mathification for users that set `dcolumn=TRUE`.

1 Get the test version

The test versions are uploaded to our local package server, KRAN. Users can obtain like so

```
CRAN <- "http://rweb.crmda.ku.edu/cran"
KRAN <- "http://rweb.crmda.ku.edu/kran"
options(repos = c(KRAN, CRAN))
update("rockchalk")
```

I've already noticed a couple of quirks while testing 1.8.120, so by the time you try this, maybe you'll get newer.

2 Briefly: outreg

First, run a regression model. Second, give that fitted regression object to outreg. When outreg runs, it will have 2 effects. First, it will write out LaTeX code to the screen (unless print.results=FALSE). A user might "copy/paste" that code into a LaTeX document. (Or, in an Sweaved document, the chunk parameter results=tex will cause the table to be displayed in the document.)

Usually, I would suggest instead that authors should create an outreg output file, with a sequence like this

```
lm1 <- lm(... regression commands ...)
lm1.out <- outreg(lm1, print.results=FALSE, ...parameters like
   float=FALSE, tight = FALSE ...)
cat(lm1.out, file = "lm1.out.tex")
```

I'd usually have an output directory where the tex file would be placed, but aside from this detail, that's my workflow. Then, when I want to use that file in my document, I use the LaTeX code "\input{lm1.out.tex}".

One reason for using that 2-step workflow is that the *automatically produced outreg table may not be exactly perfect*. Perhaps a variable label does not look right. Personally speaking, I wish the automatic table were always perfect. Practically, I accept it is not. So I leave open the opportunity that it might need to be revised by hand.

3 Briefly: table and tabular

This is the most confusing thing for beginners. Terminology

tabular In LaTeX, a "tablular" object is a grid, a display like a "spreadsheet".

table In LaTeX, a "table" is not a grid. It is a document "subsection" that "floats" around in the document. It is a container. A tabular is placed inside a table. Table objects are numbered, can be used in cross references (if they have labels)

If we are writing LaTeX code by hand, a tabulr inside a table will look like

```
\begin{table}
    \caption{The title of the floating object is specified in
       caption}
    \label{tab:ex1}% means that for cross referencing, call this
       float "tab:ex1"
\begin{tabular}
  ...code to create tablar defined here
\end{tabular}
\end{table}
```

rockchalk lets you choose, float or no float

The rockchalk outreg function has a parameter, `float=FALSE` or `float=TRUE`, to determine whether the output should include the floating table part along with the tabular object. If outreg users specify the argument title, then they are implicitly setting `float=TRUE` and the value for title is used as the LaTeX caption.

The default for `float` is `FALSE`, because I have usually preferred to control the floating table from my document. However, I recognize that many people who are better at LaTeX than I am disagree, they want to write the table and tabular structures into one file.

Why would a user prefer float=TRUE?

If you are not using LyX, then it may be convenient to let float=TRUE so the floating table can be handled from the outreg function. outreg allows the author to specify the caption and the label. However, I did not allow for any more fine-grained customizations, especially table placement details.

Why I don't generally use float=TRUE

LaTeX has options to control the placement of floating table structures that the rockchalk outreg function does not adjust. I think that if the LaTeX author wants to adjust those things, it is easier for the author to control them in the document itself, rather than the outreg function that writes a tabular object.

In LyX, the user interface has a pull down menu to create floating objects and the GUI includes a feature to set the title (sorry, the 'caption'), and it also includes a way to set a label. If I use this method, then LyX is aware of this thing and the LyX cross-referencing system is available.

Table 1: My One Tightly Printed Regression

	M1
	Estimate
	(S.E.)
(Intercept)	30.245***
	(0.618)
x1	1.546*
	(0.692)
N	100
RMSE	6.121
R^2	0.048
$*p \le 0.05{*}{*}\ p \le 0.01{*}{*}{*}p \le 0.001$	

Another reason to use float=FALSE

In an instructional document using Sweave, the code chunks will print out where they "are" in the input file.

If one creates a table (floating container), and then puts the code chunk inside, then the code will print out with the table. That is demonstrated in Tables 13, 14, 15, 20.

4 Torture test of the "rockchalk" and the outreg function

In most of these examples, I've taken the one-step "float=TRUE" option, but I have some examples where I've manually created the LaTeX float and placed an outreg tabular inside it.

```
set.seed(2134234)
 dat <- data.frame(x1 = rnorm(100), x2 = rnorm(100))
 dat$y1 <- 30 + 5 * rnorm(100) + 3 * dat$x1 + 4 * dat$x2
 dat$y2 <- rnorm(100) + 5 * dat$x2
 m1 <- lm(y1 ~ x1, data = dat)
 m2 <- lm(y1 ~ x2, data = dat)
 m3 <- lm(y1 ~ x1 + x2, data = dat)
 gm1 <- glm(y1 ~ x1, family = Gamma, data = dat)
```

4.1 tight and wide styles for regression tables

In my terminology, a tight table has one "narrow" column for a regression (parameter and standard error stacked on top of each other). A table that is not in the tight format is wide, it has 2 columns for each table (parameter and standard error side by side).

A tight table is the default, it seems to be what many political science, sociology, and economics students will need. See Table 1. A wide format table is, in my opinion, more pleasant for the eyes and seems to be more popular in psychology. See Table 2.

Table 2: My Wide Format ”side-by-side” columns

	M1	
	Estimate	(S.E.)
(Intercept)	30.245***	(0.618)
x1	1.546*	(0.692)
N	100	
RMSE	6.121	
R^2	0.048	
$*p \le 0.05 ** \ p \le 0.01 *** p \le 0.001$		

Table 3: Tight column with dcolumn=TRUE

	M1
	Estimate
	(S.E.)
(Intercept)	30.245$***$
	(0.618)
x1	1.546$*$
	(0.692)
N	100
RMSE	6.121
R^2	0.048
$*p \le 0.05 ** \ p \le 0.01 *** p \le 0.001$	

```
library(rockchalk)
ex1w <- outreg(m1, title = "My Wide Format \"side-by-side\"
   columns",
                 label = "tab:ex1w", tight = FALSE,
                 float = TRUE, print.results = FALSE)
cat(ex1w)
```

Decimal-centered test case 1

A decimal-centered version of the tight column format can be seen in Table 3 and the decimal-centered version of the wide format is seen in Table 4

```
library(rockchalk)
ex2d <- outreg(m1, title = "Tight column with dcolumn=TRUE",
                 label = "tab:ex2d", dcolumn = TRUE,
                 float = TRUE, print.results=FALSE)
cat(ex2d)
```

```
library(rockchalk)
ex1wd <- outreg(m1, title = "Wide (not tight) format with dcolumn
   = TRUE",
```

Table 4: Wide (not tight) format with dcolumn = TRUE

	M1	
	Estimate	(S.E.)
(Intercept)	30.245∗∗∗	(0.618)
x1	1.546∗	(0.692)
N	100	
RMSE	6.121	
R^2	0.048	

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$

Table 5: Ability to change p values with decimal-centered columns

	Fingers	
	Estimate	(S.E.)
(Intercept)	30.245∗∗∗	(0.618)
x1	1.546 ∗ ∗	(0.692)
N	100	
RMSE	6.121	
R^2	0.048	

$*p \leq 0.1 ** \ p \leq 0.05 *** p \leq 0.01$

```
                    label = "tab:ex2wd", tight = FALSE, dcolumn =
                        TRUE,
                    float = TRUE, print.results = FALSE)
cat(ex1wd)
```

alpha level adjustments

In Table 5, I demonstrate that the alpha parameter can be used to select different values for the critical region. Several alpha adjustments are scattered about in the examples.

```
ex2pd <- outreg(list("Fingers" = m1), tight = FALSE,
             title = "Ability to change p values with
                 decimal-centered columns",
             label = "tab:ex2pd", dcolumn = TRUE,
             float = TRUE,
             alpha = c(0.1, 0.05, 0.01))
```

```
## I didn't put print.results=FALSE, so no need to cat result here
```

In my first tests, I see some trouble here because the stars are not evenly spaced. This is, almost surely, a dcolumn side effect. Compare Table 6, and you see without dcolumn, I don't have star trouble. This is the kind of ripple effect we need to find and fix. Your tests will help on this.

```
ex2p <- outreg(list("Fingers" = m1), tight = FALSE,
             title = "Ability to change p values with
                 decimal-centered columns",
```

Table 6: Ability to change p values with decimal-centered columns

	Fingers	
	Estimate	(S.E.)
(Intercept)	30.245***	(0.618)
x1	1.546**	(0.692)
N	100	
RMSE	6.121	
R^2	0.048	
$*p \leq 0.1{*}{*}\ p \leq 0.05{*}{*}{*}p \leq 0.01$		

```
          label = "tab:ex2p", dcolumn = FALSE,
          float = TRUE,
          alpha = c(0.1, 0.05, 0.01))
```

```
## I didn't put print.results=FALSE, so no need to cat result here
```

Several models in same table

One of the most valuable features of rockchalk is that one can align several models side by side and compare them. The rows are matched by the variable name.

About model names: If a list of regression fits is not named, then the names will be bland, “M1”, “M2”.

It is highly recommended that authors should name the regression models.

Currently, I recommend that we provide the model names **in** the list that provides the fitted regressions, as you see here. That produces Table 7.

```
ex3 <- outreg(list("Model A" = m1, "Model B has a longer heading"
   = m2),
          varLabels = list(x1 = "Billie"),
          title = "My Two Linear Regressions", label = "tab:ex3",
          request = c(fstatistic = "F"),
          print.results = FALSE)
cat(ex3)
```

In the original rockchalk, I had it differently. It is possible to provide model names in a separate argument, `modelLabels`. That is shown in the code below, and in output Table 8. The modelLabels parameter takes precedence, it will replace the labels in the first argument. However, I found it confusing to write tables that way, so I made it work the other way too.

```
ex3b <- outreg(list("Model A" = m1, "Model B" = m2),
          modelLabels = c("Overrides ModelA", "Overrides ModelB"),
          varLabels = list(x1 = "Billie"),
          title = "Note modelLabels Overrides model names",
          label = "tab:ex3b"
 )
```

Table 7: My Two Linear Regressions

	Model A Estimate (S.E.)	Model B has a longer heading Estimate (S.E.)
(Intercept)	30.245***	29.774***
	(0.618)	(0.522)
Billie	1.546*	.
	(0.692)	
x2	.	3.413***
		(0.512)
N	100	100
RMSE	6.121	5.205
R^2	0.048	0.312
adj R^2	0.039	0.305
F	4.98 df(1,98)*	44.4 df(1,98)***

$*p \le 0.05 ** \; p \le 0.01 *** p \le 0.001$

Table 8: Note modelLabels Overrides model names

	Overrides ModelA Estimate (S.E.)	Overrides ModelB Estimate (S.E.)
(Intercept)	30.245***	29.774***
	(0.618)	(0.522)
Billie	1.546*	.
	(0.692)	
x2	.	3.413***
		(0.512)
N	100	100
RMSE	6.121	5.205
R^2	0.048	0.312
adj R^2	0.039	0.305

$*p \le 0.05 ** \; p \le 0.01 *** p \le 0.001$

Table 9: Still have showAIC argument, as in previous versions

	Whichever Estimate (S.E.)	Whatever Estimate (S.E.)
(Intercept)	30.245***	29.774***
	(0.618)	(0.522)
x1	1.546*	.
	(0.692)	
x2	.	3.413***
		(0.512)
N	100	100
RMSE	6.121	5.205
R^2	0.048	0.312
adj R^2	0.039	0.305
AIC	617.694	617.694
$*p \le 0.05 ** p \le 0.01 *** p \le 0.001$		

Specifying additional summary information

In the first version, I was thinking that everybody would be happy enough if the table included N, the standard error of the regression (which I dubbed RMSE in the old SAS style), and R-square. There were requests for other summaries.

At first, I was thinking I needed to add arguments for each request. I created argument “ showAIC=TRUE ” to request Akaike’s information criterion. For example, see Table 9 produced by the following

```
ex5d <- outreg(list("Whichever" = m1, "Whatever" = m2),
           title = "Still have showAIC argument, as in previous
              versions",
           label = "tab:ex5d", showAIC = TRUE, float = TRUE)
```

However, I anticipated that way might lead me down a bad path of writing a parameter for every possible summary statistic.

My first idea was to create a recipe book of particular summary items and make them available for requests by users. For example, I wrote a customized reporter for F statistics and a parameter to ask for that was called “request”. For example, request = c(fstatistic = ”F”) asks for my special fstatistic and the label to be used for it in the table would be “F”. I ended up not making very many of those fancy object, but in the semTable in the kutils package there is a similar approach for the model Chi-Square and now it seems to me I should come back and do more like that in outreg.

Because I expected that I would never have time to keep up with requests for specialized summary items, I created a “back door” though which users could use functions and include them in the summary.

However, I shot myself in the foot by creating that general purpose ability with a horrible, ungainly name for the function’s argument, runFuns . I admit that is a horrible name, but I had good intentions. “runFuns” is short for “run functions”. It will run a function named as the first argument and then label it with the right hand side argument. So, if one has a lot of request like AIC, BIC, and so forth, the R functions can be used without too much effort.

Table 10: Another way to get AIC output

	Whatever Estimate (S.E.)	Whatever.1 Estimate (S.E.)
(Intercept)	30.245∗∗∗	29.774∗∗∗
	(0.618)	(0.522)
x1	1.546∗	.
	(0.692)	
x2	.	3.413∗∗∗
		(0.512)
N	100	100
RMSE	6.121	5.205
R^2	0.048	0.312
adj R^2	0.039	0.305
Akaike IC	650.11	617.69

$*p \leq 0.05 ** \; p \leq 0.01 *** p \leq 0.001$

```
ex6d <- outreg(list("Whatever" = m1, "Whatever" =m2),
          title = "Another way to get AIC output", label="ex6d",
          runFuns = c("AIC" = "Akaike IC"), dcolumn=TRUE,
             print.results=FALSE)
cat(ex6d)
```

Insert more regressions in one table

This code produces Table 11, which is NOT decimal aligned

```
ex7 <- outreg(list("Amod" = m1, "Bmod" = m2, "Gmod" = m3),
   dcolumn=FALSE,
               title = "My Three Linear Regressions",
                  label="tab:ex7")
```

The column-aligned version of the same is found in Table 12, produced by the following code.

```
ex7d <- outreg(list("Amod" = m1, "Bmod" = m2, "Gmod" = m3),
   dcolumn=TRUE,
               title = "My Three Linear Regressions (decimal
                  aligned)", label="tab:ex7d")
```

I worried that users who are verbose might break the function. In Table 13, I show that some very long names, even ones with periods, do not seem to cause horrible trouble. They WILL run off the edge of the page if they get much longer. Author will need to change to the tight columns if that is an issue, as seen in Table **??**.

Table 11: My Three Linear Regressions

	Amod Estimate (S.E.)	Bmod Estimate (S.E.)	Gmod Estimate (S.E.)
(Intercept)	30.245***	29.774***	30.013***
	(0.618)	(0.522)	(0.490)
x1	1.546*	.	2.217***
	(0.692)		(0.555)
x2	.	3.413***	3.717***
		(0.512)	(0.483)
N	100	100	100
RMSE	6.121	5.205	4.849
R^2	0.048	0.312	0.409
adj R^2	0.039	0.305	0.397

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$

Table 12: My Three Linear Regressions (decimal aligned)

	Amod Estimate (S.E.)	Bmod Estimate (S.E.)	Gmod Estimate (S.E.)
(Intercept)	30.245***	29.774***	30.013***
	(0.618)	(0.522)	(0.490)
x1	1.546*	.	2.217***
	(0.692)		(0.555)
x2	.	3.413***	3.717***
		(0.512)	(0.483)
N	100	100	100
RMSE	6.121	5.205	4.849
R^2	0.048	0.312	0.409
adj R^2	0.039	0.305	0.397

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$

Table 13: Stress test very long titles (float=FALSE)

```
ex11 <- outreg(list("I Love Long Titles" = m1,
              "Prefer Brevity" = m2,
              "Captain. Kirk. Named. This." = m3), tight =
                 FALSE, float = FALSE,
              dcolumn = TRUE)
```

	I Love Long Titles		Prefer Brevity		Captain. Kirk. Named. This.	
	Estimate	(S.E.)	Estimate	(S.E.)	Estimate	(S.E.)
(Intercept)	30.245∗∗∗	(0.618)	29.774∗∗∗	(0.522)	30.013∗∗∗	(0.490)
x1	1.546∗	(0.692)	.		2.217∗∗∗	(0.555)
x2	.		3.413∗∗∗	(0.512)	3.717∗∗∗	(0.483)
N	100		100		100	
RMSE	6.121		5.205		4.849	
R^2	0.048		0.312		0.409	
adj R^2	0.039		0.305		0.397	

$*p \le 0.05{*}{*}\ p \le 0.01{*}{*}{*}p \le 0.001$

Alternative standard errors

The original rockchalk took the standard errors from the fitted model. A student in Brazil wrote and asked me to make it possible for the author to supply “robust” standard errors. That was a good idea. The code to demonstrate how to create an alternative vector of standard errors (in that case, Huber-White robust standard errors) will appear with the output in Table 15. The code is displayed there because I created the outreg table with float=FALSE, and then I manually created a table container into which I typed the R code chunk.

Aligning different kinds of fits

The output from lm and glm fits may sometimes be usefully compared. The parameter display from rockchalk will align same-named variables. The difference in the available summary statistics is apparent because the rows do not “line up”. See Table 16 for the output from the following.

```
ex13 <- outreg(list("OLS" = m1, "GLM" = gm1), float = TRUE,
              title = "OLS and Logit in same table (dcolumn)",
              label="tab:ex13", alpha = c(0.05, 0.01), dcolumn =
                 TRUE)
```

As seen in Table 17, we check if the supplemental parameter requests hold up with dcolumn

```
ex14 <- outreg(list(OLS = m1, GLM = gm1), float = TRUE,
         title = "OLS and Logit with summary report features
            (dcolumn)",
         label = "tab:ex14",
         request = c(fstatistic = "F"), runFuns = c("BIC" =
            "BIC"),
         dcolumn = TRUE)
```

Table 14: Stress test very long titles (float=FALSE)

```
ex1t1 <- outreg(list("I Love Long Titles" = m1,
                "Prefer Brevity" = m2,
                "Captain. Kirk. Named. This" = m3), float = FALSE,
                dcolumn = TRUE)
```

	I Love Long Titles	Prefer Brevity	Captain. Kirk. Named. This
	Estimate	Estimate	Estimate
	(S.E.)	(S.E.)	(S.E.)
(Intercept)	30.245∗∗∗	29.774∗∗∗	30.013∗∗∗
	(0.618)	(0.522)	(0.490)
x1	1.546∗	.	2.217∗∗∗
	(0.692)		(0.555)
x2	.	3.413∗∗∗	3.717∗∗∗
		(0.512)	(0.483)
N	100	100	100
RMSE	6.121	5.205	4.849
R^2	0.048	0.312	0.409
adj R^2	0.039	0.305	0.397

$*p \le 0.05 ** \ p \le 0.01 *** p \le 0.001$

What if the number of digits is dialed up and alpha is altered? See Table 18.

```
ex15 <- outreg(list(OLS = m1, GLM = gm1), float = TRUE,
          title="OLS and GLM with more digits (digits)",
          label="tab:ex15",
          request = c(fstatistic = "F"), runFuns = c("BIC" =
             "BIC"),
          digits = 5, alpha = c(0.01), dcolumn = TRUE)
```

In Table 19, output shows result when several runFuns are requested. Again, I'm very sorry that parameter name is so aweful. I adjusted the alpha stars as well.

```
ex16 <- outreg(list("OLS 1" = m1, "OLS 2" = m2,  GLM = gm1), float
   = TRUE,
           title = "2 OLS and 1 Logit (dcolumn), additional
              runFuns",
           label="tab:ex16",
           request = c(fstatistic = "F"),
           runFuns = c("BIC" = "BIC", "logLik" = "ll"),
           digits = 5, alpha = c(0.1, 0.05, 0.01), dcolumn = TRUE)
```

After a while, I noticed that the left hand side of runFuns need not be in quotation marks. So my examples are not always consistent. For example, in Table 20 I have some in quotes, some not. And I also show that if authors want to create redundant rows, they are allowed to do so (N, for example).

Table 15: Robust Standard Errors (produced with float=FALSE)

```
if (require(car)){
    newSE <- sqrt(diag(car::hccm(m3)))
    ex8 <- outreg(list("Model A" = m1, "Model B" = m2, "Model C" =
       m3,
              "Model C w Robust SE" = m3),
              SElist= list("Model C w Robust SE" = newSE))
 }
```

	Model A	Model B	Model C	Model C w Robust SE
	Estimate	Estimate	Estimate	Estimate
	(S.E.)	(S.E.)	(S.E.)	(S.E.)
(Intercept)	30.245***	29.774***	30.013***	30.013***
	(0.618)	(0.522)	(0.490)	(0.481)
x1	1.546*	.	2.217***	2.217***
	(0.692)		(0.555)	(0.618)
x2	.	3.413***	3.717***	3.717***
		(0.512)	(0.483)	(0.464)
N	100	100	100	100
RMSE	6.121	5.205	4.849	4.849
R^2	0.048	0.312	0.409	0.409
adj R^2	0.039	0.305	0.397	0.397

$*p \le 0.05{**}\ p \le 0.01{***}p \le 0.001$

References

R Core Team (2018). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Table 16: OLS and Logit in same table (dcolumn)

	OLS Estimate (S.E.)	GLM Estimate (S.E.)
(Intercept)	$30.245**$	$0.033**$
	(0.618)	(0.001)
x1	$1.546*$	$-0.002*$
	(0.692)	(0.001)
N	100	100
RMSE	6.121	
R^2	0.048	
Deviance		4.301
$-2LLR(Model \chi^2)$		0.208

$*p \leq 0.05** \; p \leq 0.01$

Table 17: OLS and Logit with summary report features (dcolumn)

	OLS Estimate (S.E.)	GLM Estimate (S.E.)
(Intercept)	$30.245***$	$0.033***$
	(0.618)	(0.001)
x1	$1.546*$	$-0.002*$
	(0.692)	(0.001)
N	100	100
RMSE	6.121	
R^2	0.048	
F	$4.98 df(1,98)*$	
Deviance		4.301
$-2LLR(Model \chi^2)$		0.208
BIC	657.92	659.82

$*p \leq 0.05** \; p \leq 0.01***p \leq 0.001$

Table 18: OLS and GLM with more digits (digits)

	OLS Estimate (S.E.)	GLM Estimate (S.E.)
(Intercept)	30.24550$*$	0.03313$*$
	(0.61763)	(0.00068)
x1	1.54553	-0.00173
	(0.69242)	(0.00078)
N	100	100
RMSE	6.12090	
R^2	0.04838	
F	$4.9821 df(1, 98)$	
Deviance		4.30066
$-2LLR(Model \chi^2)$		0.20827
BIC	657.92	659.82

$*p \le 0.01$

Table 19: 2 OLS and 1 Logit (dcolumn), additional runFuns

	OLS 1 Estimate (S.E.)	OLS 2 Estimate (S.E.)	GLM Estimate (S.E.)
(Intercept)	30.24550$***$	29.77420$***$	0.03313$***$
	(0.61763)	(0.52229)	(0.00068)
x1	$1.54553 * *$	.	$-0.00173 * *$
	(0.69242)		(0.00078)
x2	.	3.41342$***$	.
		(0.51222)	
N	100	100	100
RMSE	6.12090	5.20508	
R^2	0.04838	0.31184	
adj R^2	0.03867	0.30482	
F	$4.9821 df(1, 98)**$	$44.409 df(1, 98)***$	
Deviance			4.30066
$-2LLR(Model \chi^2)$			0.20827
BIC	657.92	625.51	659.82
ll	$-322.05(df = 3)$	$-305.85(df = 3)$	$-323(df = 3)$

$*p \le 0.1 ** p \le 0.05 *** p \le 0.01$

Table 20: Additional test on summary stats (dcolumn)

```
ex17 <- outreg(list("Model A" = gm1, "Model B label with Spaces" =
    m2),
          request = c(fstatistic = "F"),
          runFuns = c("BIC" = "Schwarz IC", "AIC" = "Akaike IC",
             "logLik" = "ll",
          "nobs" = "N Again?"), dcolumn=TRUE)
```

	Model A Estimate (S.E.)	Model B label with Spaces Estimate (S.E.)
(Intercept)	0.033∗∗∗	29.774∗∗∗
	(0.001)	(0.522)
x1	−0.002∗	.
	(0.001)	
x2	.	3.413∗∗∗
		(0.512)
N	100	100
RMSE		5.205
R^2		0.312
adj R^2		0.305
F		$44.4 df(1, 98)$∗∗∗
Deviance	4.301	
$-2LLR(Model \chi^2)$	0.208	
Schwarz IC	659.82	625.51
Akaike IC	652.00	617.69
ll	$-323(df = 3)$	$-306(df = 3)$
N Again?	100	100

∗$p \leq 0.05$∗∗ $p \leq 0.01$∗∗∗$p \leq 0.001$

Replication Information

```
R version 3.5.1 (2018-07-02)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.1 LTS

Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] car_3.0-0         carData_3.0-1     rockchalk_1.8.121

```

```
loaded via a namespace (and not attached):
 [1] Rcpp_0.12.17      magrittr_1.5      kutils_1.49       splines_3.5.1     MASS_7.3-50
 [6] mnormt_1.5-5      pbivnorm_0.6.0    xtable_1.8-2      lattice_0.20-35   rlang_0.2.1
[11] minqa_1.2.4       plyr_1.8.4        tools_3.5.1       grid_3.5.1
    data.table_1.11.4
[16] nlme_3.1-137      rio_0.5.10        abind_1.4-5       readxl_1.1.0      lme4_1.1-17
[21] tibble_1.4.2      lavaan_0.6-1      Matrix_1.2-14     zip_1.0.0         nloptr_1.0.4
[26] curl_3.2          haven_1.1.1       openxlsx_4.1.0    cellranger_1.1.0  pillar_1.2.3
[31] compiler_3.5.1    forcats_0.3.0     stats4_3.5.1      foreign_0.8-70
```