Regression Tables

Statistical packages often report regression results in a way that is not how you would want to display them in a paper or on a website. Additionally, they rarely provide an option to display multiple regression results in the same table.

Two (bad) options for including regression results in your paper include copying over each desied number by hand, or taking a screenshot of your regression output. Much better is using a command that outputs regression results in a nice format, in a way you can include in your presentation.

Keep in Mind

- Any good regression table exporting command should include an option to limit the number of significant digits in your result. You should almost always make use of this option. It is very rare that the seventh or eighth decimal place (commonly reported in statistics packages) is actu meaningful, and it makes it difficult to read your table.
- Variable names serve different purposes in statistical coding and in papers. Variable names in papers should be changed to be readable in the language of the paper. So for example, while employment may be recorded as `EMP_STAT` in your statistics package, you should rename it Employment for your paper. Most table exporting commands include options to perform this renaming. But if it doesn't, you can always change it by hand after exporting.
- If you use asterisks to indicate significance, be sure to check the significance levels that different numbers of asterisks indicate in the command you're using, as standards for what significance levels the asterisks mean vary across fields (and so vary across commands as well). Most commands include an option to change the significance levels used. On that note, *always include a table note saying what the different asterisk indicators mean!* These commands should all include one by default - don't take it out!

Also Consider

- If you are a Word user, and the command you are using does not export to Word or RTF, you can get the table into Word by exporting an HTML, CSV, or LaTeX, then opening up the result in your browser, Excel, or TtH, respectively. Excel and HTML tables can generally be copy/pasted

directly into Word (and then formatted within Word). You may at that point want to use Word's "Convert Text to Table" command, especially if you've pasted in HTML.

- By necessity, regression-output commands often have about ten million options, and they can't all be covered on this page. If you want it to do something, it probably can. To reduce errors, it is probably a good idea to do as little formatting and copy/pasting by hand as possible. So if you want to do something it doesn't do by default, like adding additional calculations, check out the **help file** for your command to see how you can do it automatically.

Implementations

Python

The most convenient package for showing multiple regressions together is **stargazer**. In the code below, we'll see an example of using **stargazer** and a couple of its most basic customisation options. Note that in order to run this example yourself, you may need to run 'pip install packagename' on the command line on your computer to install the **pandas**, **stargazer**, and **statsmodels** packages (if you don't already have these packages installed).

```
import pandas as pd
import statsmodels.formula.api as smf
from stargazer.stargazer import Stargazer
```

Get the mtcars data

```
mtcars = (pd.read_csv('https://raw.githubusercontent.com/LOST-STATS/lost-stats.github.io/source/D
                      dtype={'model': str, 'mpg': float, 'hp': float, 'disp': float, 'cyl': "floa
```

Let's run two separate regressions that will be added to our summary table

```
results1 = smf.ols('mpg ~ cyl', data=mtcars).fit()
results2 = smf.ols('mpg ~ cyl + hp', data=mtcars).fit()
```

To see the results table for an individual reg, use

```
print(results1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.726
Model:                            OLS   Adj. R-squared:                  0.717
Method:                 Least Squares   F-statistic:                     79.56
Date:                Sat, 22 May 2021   Prob (F-statistic):           6.11e-10
Time:                        10:09:14   Log-Likelihood:                -81.653
No. Observations:                  32   AIC:                             167.3
Df Residuals:                      30   BIC:                             170.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     37.8846      2.074     18.268      0.000      33.649      42.120
cyl           -2.8758      0.322     -8.920      0.000      -3.534      -2.217
==============================================================================
Omnibus:                        1.007   Durbin-Watson:                   1.670
Prob(Omnibus):                  0.604   Jarque-Bera (JB):                0.874
Skew:                           0.380   Prob(JB):                        0.646
Kurtosis:                       2.720   Cond. No.                         24.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Now let’s create a table with both regressions in

```
stargazer_tab = Stargazer([results1, results2])
# Using a Python interactive window, simply run the
# name of the object to display the table
stargazer_tab
```

	Dependent variable:mpg	
	(1)	(2)

Intercept	37.885***	36.908***
	(2.074)	(2.191)
cyl	-2.876***	-2.265***
	(0.322)	(0.576)
hp		-0.019
		(0.015)
Observations	32	32
R^2	0.726	0.741
Adjusted R^2	0.717	0.723
Residual Std. Error	3.206 (df=30)	3.173 (df=29)
F Statistic	79.561*** (df=1; 30)	41.422*** (df=2; 29)
Note:		*p<0.1; **p<0.05; ***p<0.01

To export to a file, use, for example:

```
open('regression.tex', 'w').write(stargazer_tab.render_latex())  # for latex
open('regression.html', 'w').write(stargazer_tab.render_html())  # for html
```

Finally, it's good practice to use informative names for variables

```
stargazer_tab.rename_covariates({'cyl': 'Cylinders', 'hp': 'Horsepower'})
stargazer_tab
```

	Dependent variable:mpg	
	(1)	(2)

Intercept	37.885***	36.908***
	(2.074)	(2.191)
Cylinders	-2.876***	-2.265***
	(0.322)	(0.576)
Horsepower		-0.019
		(0.015)
Observations	32	32
R^2	0.726	0.741
Adjusted R^2	0.717	0.723
Residual Std. Error	3.206 (df=30)	3.173 (df=29)
F Statistic	79.561*** (df=1; 30)	41.422*** (df=2; 29)
Note:		*p<0.1; **p<0.05; ***p<0.01

R

There are many, many packages for exporting regression results in R, including RStudio's **gt**, **texreg**, and **xtable**. Here we will focus on two: **stargazer**, which is probably the easiest to use, and **huxtable**, which is slightly more up-to-date and offers advanced formatting options, outlined on its website.

```
# Install stargazer if necessary
# install.packages('stargazer')
library(stargazer)

# Get mtcars data
data(mtcars)

# Let's give it two regressions to output
```

```
lm1 <- lm(mpg ~ cyl, data = mtcars)
lm2 <- lm(mpg ~ cyl + hp, data = mtcars)

# Let's output an HTML table, perhaps for pasting into Word
# We could instead set type = 'latex' for LaTeX or type = 'text' for a text-only table.
stargazer(lm1, lm2, type = 'html', out = 'my_reg_table.html')

# In line with good practices, we should use readable names for our variables
stargazer(lm1, lm2, type = 'html', out = 'my_reg_table.html',
          covariate.labels = c('Cylinders','Horsepower'),
          dep.var.labels = 'Miles per Gallon')
```

This produces:

	(1)	(2)
Cylinders	-2.876 ***	-2.265 ***
	(0.322)	(0.576)
Horsepower		-0.019
		(0.015)
N	32	32
R2	0.726	0.741
logLik	-81.653	-80.781
AIC	169.306	169.562

*** p < 0.001; ** p < 0.01; * p < 0.05.

Now we will do the same thing with `huxtable`, using mostly defaults.

```
# Install huxtable and magrittr if necessary
# install.packages('huxtable', 'magrittr')
# huxtable works more easily with the pipe %>%
# which can come from magrittr or dplyr or tidyverse, etc.
library(huxtable)
```

```
library(magrittr)

# First we build a huxreg object, using readable names
huxreg(lm1, lm2,
       coefs=c('Cylinders' = 'cyl',
               'Horsepower' = 'hp')) %>%
  # We can send it to the screen to view it instantly
  print_screen()

# Or we can send it to a file with the quick_ functions, which can
# output to pdf, docx, html, xlsx, pptx, rtf, or latex.
huxreg(lm1, lm2,
       coefs=c('Cylinders' = 'cyl',
               'Horsepower' = 'hp')) %>%
  # Let's make an HTML file
  quick_html(file = 'my_reg_output.html')
```

Which produces (note the different asterisks behavior, which can be changed with `huxreg`'s `stars` option):

	Dependent variable:	
	Miles per Gallon	
	(1)	(2)
Cylinders	-2.876***	-2.265***
	(0.322)	(0.576)
Horsepower		-0.019
		(0.015)
Constant	37.885***	36.908***
	(2.074)	(2.191)

Observations	32	32
R^2	0.726	0.741
Adjusted R^2	0.717	0.723
Residual Std. Error	3.206 (df = 30)	3.173 (df = 29)
F Statistic	79.561^{***} (df = 1; 30)	41.422^{***} (df = 2; 29)
Note:		*p<0.1; **p<0.05; ***p<0.01

Stata

There are two main ways of outputting regression results in Stata, both of which must be installed from `ssc install`: **outreg2** and **estout**. We will use **estout** here, as it is more flexible. More detail is available on the **estout** website.

Also note that, in a pinch, if you're using a strange command that does not play nicely with **estout**, you can often select *any* Stata regression output, select the output, right-click, do "Cop Table", and paste the result into Excel. This is only if all else fails.

```
* Install estout if necessary
* ssc install estout

* Load auto data
sysuse auto.dta, clear

* Let's provide it two regressions
* Making sure to store the results each time
reg mpg weight
estimates store weightonly
reg mpg weight foreign
estimates store weightandforeign

* Now let's export the table using estout
```

```
* while renaming the variables for readability using the variable labels already in Stata
* replacing any table we've already made
* and making an HTML table with style(html)
* style(tex) also works, and the default is tab-delimited data for use in Excel.
* Note also the default is to display t-statistics in parentheses. If we want
* standard errors instead, we say so with se
esttab weightonly weightandforeign using my_reg_output.html, label replace style(html) se
```

Which produces:

	(1)	(2)
	Mileage (mpg)	Mileage (mpg)
Weight (lbs.)	-0.00601***	-0.00659***
	(0.000518)	(0.000637)
Car type		-1.650
		(1.076)
Constant	39.44***	41.68***
	(1.614)	(2.166)
Observations	74	74

Standard errors in parentheses
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$