Making regression tables from stored estimates

Ben Jann
ETH Zurich, Switzerland

Abstract. The organization and archiving of the statistical results and the processing of a subset of those results for publication are important and often underestimated issues in conducting statistical analyses. Because the automation of these tasks is often poor, the processing of results produced by statistical packages is quite laborious, as well as vulnerable to error. I will therefore present a new package that facilitates and automates some of these tasks called `estout`. This new command can be used to produce regression tables for use with spreadsheets, LaTeX, HTML, or word processors. For example, the results for multiple models can be organized in spreadsheets and thus archived in an orderly manner. Alternatively, the results can be directly saved as a publication-ready table for inclusion in, for example, a LaTeX document. `estout` is implemented as a wrapper for `estimates table`, but has many additional features such as support for `mfx`. However, despite its flexibility, `estout` is—I believe—still very straightforward and easy to use. Furthermore, `estout` can be customized via so-called defaults files. A tool to make available supplementary statistics called `estadd` is also provided.

Keywords: st0001, estout, estoutdef, estadd, estimates, regression table, latex, html

1 Introduction

Statistical packages are usually very good at estimating all kinds of regression models, but they are rather poor at keeping the results for those models organized and/or processing them for publication. This is a real problem because gathering the relevant figures by hand from the large amount of statistical output usually produced and arranging the results in clear and presentable tables can be very inefficient and error-prone processes. Furthermore, results must often be processed repeatedly, for example, because operationalizations are modified or mistakes are detected. In order to reduce transcription errors and avoid having to repeat the laborious tasks by hand, it makes sense to automate the processing of results as much as possible.

Fortunately, Stata provides the basis for such an automation. One of the great features in Stata is that, after an estimation command has been carried out, all the relevant results are not only displayed onscreen but returned in places where they can be accessed by the user. This storage of results provides the user with the opportunity to further process the results in a more or less automated manner. Furthermore, Stata 8 saw the introduction of the `estimates` command (see [R] **estimates**), which facilitates the handling of the estimation results for multiple models. More specifically, results from up to 20 models can be stored at a time. Stata also provides a utility for compiling a table of the coefficients for all stored models called `estimates table`. Although the `estimates table` command is rather limited and cannot be used to translate the table to spreadsheet formats or LaTeX code, it does a good job at assembling a raw matrix of models and parameters that can be used as a starting point for the creation of a complex and well formatted regression table.

In the remainder of this paper I will present the new `estout` package, a program that makes use of the possibilities provided by Stata and produces regression tables in what I believe is a very flexible and functional way. Note that there also are other user programs available to produce tables from regression results. John Luke Gallup's `outreg` is probably the most widely used package of this kind (Gallup 1998, 1999, 2000). Among the other packages are `outtex` by Antoine Terracol, `est2tex` by Marc Muendler, and `mktab` by Nicholas Winter. Also see Newson (2003) for a very appealing approach. However, `estout` represents a good compromise between functionality and usability.

2 Description and basic examples

`estout` assembles a table of coefficients, "significance stars", summary statistics, standard errors, *t* or *z* statistics, *p*-values, confidence intervals, and other statistics calculated for up to twenty models previously fitted and stored by `estimates store`. It then writes the table to the Stata log and/or to a specified text file.

The full syntax of `estout` is rather complex and is therefore to be found in the Appendix in Section 4.1 (also see `estout`'s online help). However, consider the following basic syntax, which includes only the most important options:

`estout` [*namelist*] [`using` *filename*] [`, cells(`*array*`) stats(`*scalarlist*`)`
`style(`*style*`)` *options*]

where *namelist* is a list of the names of stored estimates (the name list can be entered as `*` to refer to all stored estimates). The `cells()` and `stats()` options determine the primary contents of the table. The `style()` option determines the basic formatting of the table.

Basic usage

The general procedure for using `estout` is to first store several models using the `estimates store` command and then apply `estout` to save and/or display a table of the estimates. By default, `estout` produces a plain, tab-separated table of the coefficients of the models indicated by the command:

```
. sysuse auto
(1978 Automobile Data)
. replace price = price / 1000
price was int now float
(74 real changes made)
. replace weight = weight / 1000
weight was int now float
(74 real changes made)
```

```
. regress price weight mpg
  (output omitted)
. estimates store m1, title(Model 1)
. generate forXmpg=foreign*mpg
. regress price weight mpg forXmpg foreign
  (output omitted)
. estimates store m2, title(Model 2)
. estout * using example.txt
        m1      m2
        b       b
weight  1.746559        4.613589
mpg     -.0495122        .2631875
forXmpg         -.3072165
foreign         11.24033
_cons   1.946068        -14.44958
```

The table produced by the `estout` command looks messy in the Stata results window or the Stata log because the columns are tab-separated (note that tab characters are not preserved in the results window or the log). However, the stored `example.txt` would look tidy if it were opened, for example, in a spreadsheet program.

Choosing a style

To align the columns, fixed widths can be specified for the columns and tab characters can be removed. This is most easily done via the `style()` option, which provides a style called `fixed`:

```
. estout *, style(fixed)
                       m1           m2
                        b            b
weight           1.746559     4.613589
mpg             -.0495122     .2631875
forXmpg                      -.3072165
foreign                       11.24033
_cons            1.946068    -14.44958
```

Other predefined styles are `tab` (the default), `tex`, and `html`, but it is also possible to define one's own styles (see Appendix 4.3). The `tex` style, for example, modifies the output table for use with LaTeX's tabular environment:

```
. estout *, style(tex) varlabels(_cons \_cons)
            &          m1&          m2\\
            &           b&           b\\
weight      &    1.746559&    4.613589\\
mpg         &   -.0495122&    .2631875\\
forXmpg     &            &   -.3072165\\
foreign     &            &    11.24033\\
\_cons      &    1.946068&   -14.44958\\
```

Note that `_cons` has been replaced by its LaTeX equivalent in the example above using the `varlabels()` option (since the underscore character produces an error in LaTeX unless it is preceded by a backslash). For more information on the `varlabels()` option, consult `estout`'s online help.

The cells option

Use the `cells()` option to specify the parameter statistics to be tabulated and how they are to be arranged. The parameter statistics available are `b` (coefficients; the default), `se` (standard errors), `t` (t/z statistics), `p` (p-values), `ci` (confidence intervals; to display the lower and upper bounds in separate cells use `ci_l` and `ci_u`), as well as any additional parameter statistics included in the `e()`-returns for the models (also see Section 3.7). For example, `cells(b se)` results in the reporting of raw coefficients and standard errors:

```
. estout *, cells(b se) style(fixed)
                       m1           m2
                     b/se         b/se
weight           1.746559     4.613589
                 .6413538     .7254961
mpg             -.0495122     .2631875
                  .086156     .1107961
forXmpg                      -.3072165
                              .1085307
foreign                       11.24033
                              2.751681
_cons            1.946068    -14.44958
                  3.59705      4.42572
```

Multiple statistics are placed in separate rows beneath one another by default as in the example above. However, elements that are listed in quotes are placed beside one another. For example, specifying `cells("b se t p")` produces the following table:

```
. estout m2, cells("b se t p") style(fixed)
                       m2
                        b           se            t            p
weight           4.613589     .7254961     6.359219     1.89e-08
mpg              .2631875     .1107961     2.375421     .0203122
forXmpg         -.3072165     .1085307    -2.830687     .0060799
foreign          11.24033     2.751681     4.084896     .0001171
_cons           -14.44958      4.42572     -3.26491     .0017061
```

The two approaches can be combined. For example, `cells("b p" se)` would produce a table with raw coefficients and standard errors beneath one another in the first column and p-values in the top row of the second column for each model.

Note that for each statistic named in the `cells()` option a set of suboptions may be specified in parentheses. For example, in social sciences it is common to report standard errors or t statistics in parentheses beneath the coefficients and to indicate the significance of individual coefficients with stars. Furthermore, the results are rounded.

Just such a table can be created using the following procedure:

```
. estout *, cells(b(star fmt(%9.3f)) se(par fmt(%9.2f))) style(fixed)
                       m1              m2
                     b/se            b/se
weight              1.747**         4.614***
                   (0.64)          (0.73)
mpg                -0.050           0.263*
                   (0.09)          (0.11)
forXmpg                            -0.307**
                                   (0.11)
foreign                            11.240***
                                   (2.75)
_cons               1.946         -14.450**
                   (3.60)          (4.43)
```

The `estout` default is to display `*` for $p < .05$, `**` for $p < .01$, and `***` for $p < .001$. However, note that the significance thresholds and symbols are fully customizable (see the `starlevels` option in Appendix 4.1).

The stats option

Finally, use the `stats()` option to specify scalar statistics to be displayed in the last rows of each model's table. The available scalar statistics are `aic` (Akaike's information criterion), `bic` (Schwarz's information criterion), `rank` (the rank of `e(V)`, i.e. the number of free parameters in model), `p` (the p-value of the model), as well as any scalar contained in the `e()`-returns for the models (also see Section 3.7). For example, specify `stats(r2 bic N)` to add the R-squared, BIC, and the number of cases to the bottom of the table:

```
. estout *, stats(r2 bic N) style(fixed)
                       m1          m2
                        b            b
weight           1.746559    4.613589
mpg             -.0495122    .2631875
forXmpg                     -.3072165
foreign                      11.24033
_cons            1.946068   -14.44958
r2               .2933891    .5516277
bic              356.2918    331.2406
N                      74          74
```

3 Advanced applications

The `estout` package has many features and it is beyond the scope of this text to provide examples for all of these options. The following presentation is therefore restricted to a few selected examples illustrating the spectrum of `estout`'s capabilities and introducing some of its less obvious applications.

Table 1: The auto data

	Model 1		Model 2	
	Coef.	p-value	Coef.	p-value
Weight (lbs.)	1.747	.008	4.614	.000
Mileage (mpg)	–.050	.567	.263	.020
Foreign*Mileage			–.307	.006
Foreign car type			11.240	.000
Constant	1.946	.590	–14.450	.002
Adj. R^2	.273		.526	
No. of cases	74		74	

Source: auto.dta

3.1 Using labels

The `labels` option will cause `estout` to use variable labels and model labels, if available. Furthermore, there are options for specifying custom labels for the different table elements, displaying a legend explaining the significance symbols and thresholds, and inserting lines of text at various places in the table. The following example is intended to provide a first impression of these possibilities:

```
. label variable foreign "Foreign car type"

. label variable forXmpg "Foreign*Mileage"

. estout m1 m2, cells("b(star label(Coef.)) se(label(Std. err.))")
> stats(r2 N, labels(R-squared "N. of cases")) label legend
> varlabels(_cons Constant) posthead("") prefoot("") postfoot("")
> varwidth(16) style(fixed)
                      Model 1                       Model 2
                        Coef.       Std. err.         Coef.       Std. err.

Weight (lbs.)        1.746559**      .6413538      4.613589***     .7254961
Mileage (mpg)       -.0495122         .086156      .2631875*       .1107961
Foreign*Mileage                                   -.3072165**      .1085307
Foreign car type                                   11.24033***     2.751681
Constant             1.946068         3.59705     -14.44958**       4.42572

R-squared            .2933891                      .5516277
N. of cases                74                            74

* p<0.05, ** p<0.01, *** p<0.001
```

3.2 LaTeX tables

The highest degree of automation can probably be attained by using `estout` in combination with LaTeX. Table 1 of this document was produced by inserting the line

```
\input{auto.tex}
```

in the LaTeX document for this article after having run the following command:

```
. estout m1 m2 using auto.tex,
> cells("b(label(Coef.) fmt(%9.3f)) p(label(\$p\$-value))")
> stats(r2_a N, fmt(%9.3f %9.0f) labels("Adj. \$R^2\$" "No. of cases"))
> label msign(--) nolz varwidth(16) modelwidth(13) style(tex)
> title(The auto data\label{auto}) varlabels(_cons Constant)
> mlabels(, span prefix(\multicolumn{@span}{c}{) suffix(}))
> prehead("\begin{table}\caption{@title}" "\begin{center}"
>  "\begin{tabular}{l*{@M}{rr}}" "\hline") posthead(\hline)
> prefoot(\hline) postfoot("\hline" "\small\textit{Source:} auto.dta"
>  "\end{tabular}" "\end{center}" "\end{table}")
\begin{table}\caption{The auto data\label{auto}}
\begin{center}
\begin{tabular}{l*{2}{rr}}
\hline
                &\multicolumn{2}{c}{Model 1}&\multicolumn{2}{c}{Model 2}\\
                &        Coef.&    $p$-value&        Coef.&    $p$-value\\
\hline
Weight (lbs.)   &        1.747&         .008&        4.614&         .000\\
Mileage (mpg)   &       --.050&         .567&         .263&         .020\\
Foreign*Mileage &             &             &       --.307&         .006\\
Foreign car type&             &             &       11.240&         .000\\
Constant        &        1.946&         .590&     --14.450&         .002\\
\hline
Adj. $R^2$      &         .273&             &         .526&             \\
No. of cases    &           74&             &           74&             \\
\hline
\small\textit{Source:} auto.dta
\end{tabular}
\end{center}
\end{table}
```

Note that most of the options in the above command could also have been provided via a defaults files (see Appendix 4.3). Working with defaults files can be very efficient if you want to produce a large number of similar tables.

3.3 Selective information

estout has a keep() and a drop() option to select the parameters (or equations) to be tabulated (an example can be found in Section 3.6), as does estimates table (see [R] **estimates**). However, a useful additional feature of estout is that the information displayed can be varied by regressors. Sometimes certain statistics are of interest only for some parameters and not for others. Those statistics can therefore be suppressed for individual parameters, using the keep() or the drop() suboption within the cells() option to save space:

```
. estout *, cells(b(star) t(par keep(mpg))) style(fixed)
                       m1              m2
                      b/t             b/t
weight           1.746559**      4.613589***
mpg             -.0495122        .2631875*
              (-.5746806)      (2.375421)
forXmpg                         -.3072165**
```

```
foreign                          11.24033***
_cons            1.946068       -14.44958**
```

Furthermore, the parameter statistics reported for the various models can be specified using the `pattern()` suboption within the `cells()` option (for example, it is possible to print the *t* statistics for, say, the second model only; an example can be found in Section 3.6).

3.4 Summary statistics only

`estout` can also be used to produce a table displaying only summary statistics:

```
. estout *, cells(none) stats(r2_a bic N, star) style(fixed)
                       m1              m2
r2_a             .2734846***     .5256351***
bic              356.2918        331.2406
N                      74              74
```

Note that in the example the models' overall significance is denoted by stars (both models are significant at the 0.001 level).

3.5 Multiple-equation models

The default in `estout` is to arrange the different equations of multiple-equation models in vertical order. However, for models like `mlogit` or `sureg` it is sometimes convenient to arrange the equations horizontally, which can be achieved through the use of the `unstack` option:

```
. sureg (price foreign weight length) (mpg displ = foreign weight)
  (output omitted)
. estimates store m4
. estout m4, cells(b t(par)) unstack stats(r2 chi2 p) style(fixed)
                       m4
                    price          mpg displacement
                      b/t          b/t          b/t
foreign           3.57526    -1.650029     -25.6127
               (5.749891)  (-1.565555)  (-2.047999)
weight           5.691462    -6.587886     96.75485
               (6.182983)  (-10.55641)   (13.06594)
length          -.0882711
              (-2.809689)
_cons            4.506212      41.6797    -87.23547
               (1.255897)   (19.64914)   (-3.46585)
r2                .548808     .6627029     .8115213
chi2             89.73586     145.3912     318.6174
p                2.50e-19     2.68e-32     6.50e-70
```

In the case of the multiple-equation models reg3, sureg, and mvreg, summary statistics for all of the model's equations will be printed in separate columns in the same row. For all other models, the summary statistics will be placed in the fist column.

3.6 Marginal effects

estout supports Stata's mfx command for calculating marginal effects or elasticities (see [R] **mfx**). In order to report the mfx results in estout, use the margin option. However, it is important that the model was saved *after* the application of mfx, as is illustrated by the following example. Note that the last column of the table in the example below displays the points around which the marginal effects were estimated (mfx returns these values in e(Xmfx_X)).

```
. generate record = 0
. replace record = 1 if rep > 3
(34 real changes made)
. logit foreign mpg record
  (output omitted)
. estimates store raw
. mfx
  (output omitted)
. estimates store mfx
. estout raw mfx, cells("b Xmfx_X(pattern(0 1))" se(par)) margin legend
> style(fixed)
                      raw         mfx
                     b/se        b/se       Xmfx_X
mpg              .1079219     .0184528      21.2973
               (.0565077)   (.0101674)
record (d)       2.435068     .4271707     .4594595
               (.7128444)   (.1043178)
_cons           -4.689347
               (1.326547)
(d) marginals for discrete change of dummy variable from 0 to 1
```

With single-equation models, the incorporation of mfx's results in the table is straightforward. However, matters become more complicated for multiple-equation models. Marginal effects have nothing to do with the equations per se and it is therefore not clear where to report the mfx results if some variables appear in several different equations. The default in estout is to print the mfx coefficients in each row that relates to the variable in question. This default can be changed with the meqs() option, which specifies that the mfx results be printed only in select equations. For example, proceed as follows to report the marginal effects for the probability of only the main outcome in heckprob:

```
. set seed 6630
. generate u = uniform() > 0.5
. heckprob u headroom, select(foreign = turn headroom) nolog
  (output omitted)
```

```
. estimates store raw
. mfx
  (output omitted)
. estimates store mfx
. estout raw mfx, cells(b se(par)) margin meqs(u) keep(u: foreign:)
> style(fixed)
                      raw          mfx
                     b/se         b/se
u
headroom        -1.003445    -.2843565
               (.6077779)   (.2326952)
_cons            2.176479
               (1.923797)
foreign
turn            -.2954961
               (.0675027)
headroom        -.1261772
               (.2919013)
_cons            11.05306
               (2.479492)
```

Taking the additional step of inserting the marginal effects for the selection probability in the example above is rather involved because the marginal effects for the two functions must be saved in different models. The solution is to print only the main equation in a first `estout` call and then append the rest of the table in a second call:

```
. mfx, predict(psel)
  (output omitted)
. estimates store mfx2
. tempfile foo
. estout raw mfx using "`foo'", cells(b se(par)) margin keep(u:)
> style(fixed) notype
. estout raw mfx2 using "`foo'", cells(b se(par)) margin
> keep(foreign:) mlabels(, none) collabels(, none)
> style(fixed) notype append
. type "`foo'"
                      raw          mfx
                     b/se         b/se
u
headroom        -1.003445    -.2843565
               (.6077779)   (.2326952)
_cons            2.176479
               (1.923797)
foreign
turn            -.2954961     -.068597
               (.0675027)   (.0158482)
headroom        -.1261772     -.029291
               (.2919013)   (.0665186)
_cons            11.05306
               (2.479492)
```

3.7 Adding supplementary statistics

Results that are included in the `e()`-returns for the models can be tabulated by `estout`. Thus, one approach for, for example, reporting certain transformations of the coefficients is to add a matrix of the transformed results to the `e()`-returns and then tabulate the results using `estout`. The `estadd` command, which is part of the `estout` package, is designed to support this approach. It may, for example, be used to add standardized coefficients or the means and standard deviations of the regressors to the `e()`-returns for the stored models. However, `estadd`'s basic capabilities can be extended by writing subroutines to allow for additional statistics.

The basic syntax of `estadd` is

`estadd` [*namelist*] `, stats(`*statslist*`)` [`prefix(`*string*`)`]

where *namelist* is again a list of stored estimates (if *namelist* is empty, `estadd` will be applied to the current estimates). Use `stats()` to specify the statistics to be added to the `e()`-returns of the indicated models. For more details, see `estadd`'s online help.

Table of descriptives

`estadd` is equipped with a few predefined statistics such as `beta` (standardized coefficients), `mean` (means of regressors), and `sd` (standard deviations of regressors). The latter can be used, for example, to produce a table of descriptives for the variables in the models in our examples:

```
. quietly generate x = uniform()
. quietly regress x price weight mpg foreign
. estadd, stats(mean sd(nobinary))
. estimates store m3
. estout m3, cells("mean sd") stats(N) mlabels(,none) drop(_cons) style(fixed)
                     mean          sd
price            6.165257    2.949496
weight           3.019459    .7771936
mpg               21.2973    5.785503
foreign          .2972973
N                      74
```

Adding user-defined statistics

Writing new `estadd` subroutines to add user-defined statistics is not overly complicated, as we will illustrate below. In general, a new subroutine should be called `_estadd_`*mystat*. *mystat* will be available to the `stats()` option of the `estadd` command after the program code has been executed or the subroutine file has been saved as `_estadd_`*mystat*`.ado` in either the current directory or somewhere else in the ado path ([P] **sysdir**). The subroutine will be called once for each model with the model's estimates restored. The `e()`-returns for the model in question may be therefore used to calculate new statistics.

Within a subroutine, use the `ereturn` command ([P] **ereturn**) to append new statistics to the existing `e()`-returns. New summary statistics should be returned as scalars using the `ereturn scalar` command, whereas new parameter statistics (e.g. transformations of the regression coefficients) should be returned as matrices (row vectors, to be precise) using the `ereturn matrix` command. Note that the columns of the added matrices should be named according to the row names of the coefficients matrix `e(b)` in order to ensure `estout`'s ability to tabulate the new parameter statistics. Use the examples below or the `_estadd_beta`, `_estadd_mean` and `_estadd_sd` subroutines, which are supplied within the file `estadd.ado` of the `estout` package, as a starting point for programming new routines.

To report the Cox and Snell pseudo R-squared, for example, define the `estadd`-subroutine

```
program _estadd_coxsnell, eclass
        syntax [ , prefix(name) * ]
        local coxsnell = 1 - exp(e(ll_0)-e(ll))^(2/e(N))
        ereturn scalar ‘prefix’coxsnell = ‘coxsnell’
end
```

and then type[1]

```
. logit foreign price weight
  (output omitted)
. estimates store m5
. logit foreign price weight mpg
  (output omitted)
. estimates store m6
. estadd m5 m6, stats(coxsnell)
. estout m5 m6, stats(coxsnell) style(fixed)
                       m5           m6
                        b            b
price           .9295969     .9263907
weight         -5.878539    -6.849737
mpg                         -.1210918
_cons           9.000472     14.42237
coxsnell         .518701     .5291797
```

New parameter statistics can be added in a similar manner. For example, the following lines of code comprise a subroutine to insert the standardized factor change coefficients, or $\exp(\beta_j S_j)$, where S_j is the standard deviation of regressor j, that are sometimes reported for logistic regression (see Long 1997):

```
program _estadd_ebsd, eclass
```

[1] Also see the `eret2` package (available from the SSC Archive). The `eret2` command provides the possibility of adding statistics to the `e()`-returns of a model without having to program subroutines. However, `eret2` can be applied only to the currently active estimates.

```
        syntax [ , prefix(name) * ]
        if "`e(cmd)'" != "logit" | "`e(wexp)'" != "" exit
        tempname results
        matrix `results' = e(b)
        local vars: colnames `results'
        local j 0
        foreach var of local vars {
                local ++j
                capture confirm variable `var'
                if _rc matrix `results'[1,`j'] = .z
                else {
                        quietly summarize `var' if e(sample)
                        matrix `results'[1,`j'] = exp( `results'[1,`j'] * r(sd) )
                }
        }
        ereturn matrix `prefix'ebsd = `results'
end
```

If the program is saved in the ado path as `_estadd_ebsd.ado`, it can, for example, be called as follows:

```
. estadd m5, stats(ebsd sd)

. estout m5, eform drop(_cons)
> cells("b(label(e^b)) ebsd(label(e^(b*sdx))) sd(label(sdx))") style(fixed)

                       m5
                      e^b    e^(b*sdx)          sdx
price            2.533488     15.51554     2.949496
weight           .0027989     .0103708     .7771936
```

4 Appendix

4.1 Full syntax of estout

`estout` [*namelist*] [`using` *filename*] [`,` *parameter_statistics_options*
summary_statistics_option *significance_stars_options* *layout_options*
labelling_options *output_options* *defaults_option*]

where *namelist* is either `_all` or `*` or *name* [*name* ...], and *name* is the name of stored estimates. The results estimated last may be indicated by a period (`.`) even if they have not yet been stored. For a detailed discussion of `estout`'s options, see the online help. A brief list of the options is provided below. Note that ⟨...⟩ stands for [[`]"]...["[']] and *str_list* denotes ⟨*string*⟩ [⟨*string*⟩ ...].

The *parameter_statistics_options* are

`cells(` { *array* | `none` } `)` specify the contents of the table cells (coefficients, standard errors, etc.)

`drop(`*droplist*`)` drop individual parameters or equations

keep(*keeplist*)	keep individual parameters or equations
equations(*eqmatchlist*)	match the models' equations
{ eform[(*pattern*)] | noeform }	display the results in exponentiated form
{ margin[({ u | c | p })] | nomargin }	report marginal effects or elasticities
{ discrete(*string*) | nodiscrete }	identify dummy variables when reporting marginal effects
meqs(*eq_list*)	select equations for marginal effects
level(#)	set the level for confidence intervals

where *array* is

⟨*row*⟩ [⟨*row*⟩ ...]

and *row* is

el[(*el_subopts*)] [*el*[(*el_subopts*)] ...]

and *el* is one of the following statistics

b	raw coefficients
se	standard errors
t	*t* statistics
p	*p*-values
ci	confidence intervals
ci_l	lower bounds of confidence intervals
ci_u	upper bounds of confidence intervals
myel	additional statistics included in e()

and the *el_subopts* are

[no]star	attach "significance stars"
fmt(%*fmt* [%*fmt* ...])	set the display formats
label(⟨*string*⟩)	define a label for *el*
{ par[(⟨*left*⟩ ⟨*right*⟩)] | nopar }	place *el* in parentheses
drop(*droplist*)	drop certain individual statistics
keep(*keeplist*)	keep certain individual statistics
pattern(*pattern*)	report *el* for selected models only
[no]abs	use absolute *t* statistics

The *summary_statistics_option* is

`stats(`*scalarlist*[`,` *stats_subopts*]`)`	specify scalar statistics to be displayed at the bottom of the table

where the *stats_subopts* are

`fmt(%`*fmt* [`%`*fmt* ...]`)`	set the display formats
`labels(`*str_list*[`,` *label_subopts*]`)`	label the scalar statistics
{ `star`[`(`*scalarlist*`)`] | `nostar` }	denote overall model significance

The *significance_stars_options* are

`starlevels(`*levelslist*`)`	define thresholds and symbols for "significance stars"
[`no`]`stardetach`	display the stars in their own column

where *levelslist* is

⟨*symbol*⟩ # [⟨*symbol*⟩ # ...]

with $\# \in (0,1]$ and listed in descending order.

The *layout_options* are

`varwidth(`#`)`	set the width of the table's left stub
`modelwidth(`#`)`	set the width of the results columns
[`no`]`abbrev`	abbreviate long names and labels
[`no`]`unstack`	place individual equations from multiple-equation models in separate columns
`begin(`⟨*string*⟩`)`	specify the beginning of the table rows
`delimiter(`⟨*string*⟩`)`	specify the column delimiter
`end(`⟨*string*⟩`)`	specify the ending of the table rows
`dmarker(`⟨*string*⟩`)`	determine the decimal marker
`msign(`⟨*string*⟩`)`	determine the minus sign
[`no`]`lz`	print the leading zero of fixed format numbers in $(-1,1)$
`substitute(`*subst_list*`)`	apply end-of-pipe substitutions

where *subst_list* is

⟨*from*⟩ ⟨*to*⟩ [⟨*from*⟩ ⟨*to*⟩ ...]

The *labelling_options* are

title(⟨*string*⟩)	specify a title for the table
[no]legend	add a legend explaining the significance symbols
prehead(*str_list*)	add text lines before the table heading
posthead(*str_list*)	add text lines after the table heading
prefoot(*str_list*)	add text lines before the table footer
postfoot(*str_list*)	add text lines after the table footer
[no]label	use variable labels
varlabels(*matchlist*[, *varl_subopts*])	relabel the parameters
mlabels(*str_list*[, *mlabels_subopts*])	label the models
collabels(*str_list*[, *label_subopts*])	label the columns within models
eqlabels(*str_list*[, *label_subopts*])	label the equations
mgroups(*str_list*[, *mgroups_subopts*])	define and label groups of models

where the *varl_subopts* are

blist(*matchlist*)	assign specific prefixes to certain rows
elist(*matchlist*)	assign specific suffixes to certain rows
label_subopts	

and the *mlabels_subopts* are

[no]numbers	number the models
[no]depvars	use dependent variables as models' labels
label_subopts	

and the *mgroups_subopts* are

pattern(*pattern*)	establish the grouping of the models
label_subopts	

and where the *label_subopts* are

[no]none	suppress the labels
prefix(⟨*string*⟩)	add a common prefix
suffix(⟨*string*⟩)	add a common suffix
begin(⟨*string*⟩)	add an overall prefix

`end(`⟨*string*⟩`)`	add an overall suffix
[`no`]`last`	print the last occurrence of `end`
[`no`]`span`	span columns if appropriate
`erepeat(`⟨*string*⟩`)`	add a “span” suffix
`lhs(`⟨*string*⟩`)`	insert *string* into the left stub of the table

The *output_options* are

[`no`]`replace`	overwrite an existing file
[`no`]`append`	append the output to an existing file
[`no`]`type`	print the table in the results window
[`no`]`showtabs`	display tabs as `<T>`s

The *defaults_option* is

`style(`*style*`)`	specify a “style” for the output table

where *style* is one of the following

`tab`	tab delimited table (the default)
`fixed`	fixed format table
`tex`	table for use with LaTeX
`html`	table for use with HTML
mystyle	user defined addition

4.2 Using @-variables

`estout` features several variables that can be used within string specifications. The following list provides an overview of these variables (also see the example in Section 3.2):

`@span` Returns the value of a count variable for the total number of physical columns of the table if used in the labels in the `blist()` and `elist()` suboptions of `varlabels()`, or in the text specified in `prehead()`, `posthead()`, `prefoot()`, or `postfoot()`.

`@span` Returns the number of spanned columns if used in the text specified in the `prefix()` and `suffix()` suboptions of `mgroups()`, `mlabels()`, `eqlabels()`, or `collabels()`, or in the labels specified in these options.

`@span` Returns the range of spanned columns (e.g. `2-4` if columns 2, 3 and 4 are spanned) if used in the text specified in the `erepeat()` suboption of `mgroups()`, `mlabels()`, `eqlabels()`, or `collabels()`.

`@M` Returns the number of models in the table if used in the text specified in `prehead()`, `posthead()`, `prefoot()`, or `postfoot()`.

`@title` Returns the title specified with the `title()` option if used in the text specified in `prehead()`, `posthead()`, `prefoot()`, or `postfoot()`.

`@discrete` Returns the explanations provided by the `discrete()` option (provided that the `margin` option is activated) if used in the text specified in `prehead()`, `posthead()`, `prefoot()`, or `postfoot()`.

`@starlegend` Returns a legend explaining the significance symbols if used in the text specified in `prehead()`, `posthead()`, `prefoot()`, or `postfoot()`.

4.3 Working with defaults files

`estout`'s `style()` option may be used to specifies a "style" for the output table. A "style" is a named combination of options that is saved in an auxiliary file called `estout_`*style*`.def`. `estout` is already equipped with four such files. The four styles and their particulars are:

settings	styles			
	`tab`	`fixed`	`tex`	`html`
`begin`				`<tr><td>`
`delimiter`	`_tab`	`" "`	`&`	`</td><td>`
`end`			`\\`	`</td></tr>`
`varwidth`	`0`	`12`	`12`	`12`
`modelwidth`	`0`	`12`	`12`	`12`
`abbrev`	off	on	off	off

It is very easy to generate one's own set of default options. Type

```
. estoutdef style, edit
```

to open one of the existing defaults files (where *style* is the name of the defaults set, e.g., `tab`; the `estoutdef` command is provided with the `estout` package), make the desired modifications and save the file as `estout_`*newstyle*`.def` in the current directory or elsewhere in the ado path (see [P] **sysdir**). To use the new options set, type:

```
. estout ..., style(newstyle)
```

`estout` has two main types of options, which are treated differentially in defaults files. On the one hand, there are simple on/off options without arguments, like `legend` or `showtabs`. To turn such an option on, enter the option followed by the options name as an argument, i.e. add the line

option option

to the defaults file. For example,

```
legend legend
```

specifies that a legend be printed in the table footer. Otherwise, if you want to turn the option of, just delete or comment out the line that contains it (or specify *option* without an argument).

To temporarily turn off an option that has been activated in a defaults file, specify `no`*option* in the command line (do not, however, use `no`*option* in defaults files). For example, if the legend has been turned on in the defaults file, but you want to suppress it in a specific call of `estout`, type

```
. estout ..., nolegend
```

On the other hand, there are options that take arguments, such as `prehead(`*args*`)`, `delimiter(`*args*`)`, or `stats(`*args*`, ...)`. Such options are specified as

option args

in the defaults file (where *args* must not include suboptions; see below). Specifying an option in the command line overwrites the settings from the defaults file. However, note that a `no` form, which exists for the first options type, is not available here.

Last but not least, there are two options that reflect a combination of the first and second types: `eform`$\big[$`(`*args*`)`$\big]$ and `margin`$\big[$`(`*args*`)`$\big]$. These options can be specified as either

option option

or

option args

in the defaults file; the `no` form is allowed.

Many `estout` options have suboptions, i.e., an option might take the form *option*`(...,` *suboption*`)` or *option*`(...,` *suboption*`(`*args*`))`. In the defaults file, the suboptions cannot be included in the definition of a higher-level option. Instead, they must be specified in their own lines, as either

optionsuboption suboption

or

optionsuboption args

In the case of a two-level nesting of options, the name used to refer to the suboption is a concatenation of the option's name and the suboption's name, i.e. `"`*optionsuboption*`"="`*option*`"+"`*suboption*`"`. For example, the `labels()` suboption of the `stats()` option would be set by the term `statslabels`. Analogously, the three level nesting in the `stats()` option yields suboption names composed of three names. For instance, the suboption called by the command

```
. estout ..., stats(..., labels(..., prefix(args)))
```

would be referred to as

```
statslabelsprefix args
```

in the defaults file. The `cells()` option represents an exception to this rule. It may be defined in the defaults file using only a plain array of cells elements without suboptions, e.g.

```
cells "b se" p
```

However, the suboptions of the cells elements may be referred to as *el_suboption*, for example

```
b_star star
```

or

```
se_par [ ]
```

Be aware that the support for comments in defaults files is limited. In particular, the `/*` and `*/` comment indicators cannot be used. The other comment indicators work (more or less) as usual, that is:

- Empty lines and lines beginning with `*` (with or without preceding blanks) will be ignored.
- `//` preceded by one or more blanks indicates that the rest of the line should be ignored. Lines beginning with `//` (with or without preceding blanks) will be ignored.
- `///` preceded by one or more blanks indicates that the rest of the line should be ignored and the part of the line preceding it should be added to the next line. In other words, `///` can be used to split commands into two or more lines of code.

5 Acknowledgements

Some of the code of `estout` has been adapted from the official `est_table.ado`. I would like to thank Kit Baum, Elisabeth Coutts, Henriette Engelhardt, Jonathan Gardnerand, Friedrich Huebler, Maren Kandulla, Clive Nicholas, Fredrik Wallenberg, Ian Watson, and Vince Wiggins for their comments and suggestions.

6 References

Gallup, J. L. 1998. sg97: Formatting regression output for published tables. *Stata Technical Bulletin* 46: 28–30.

—. 1999. sg97.1: Revision of outreg. *Stata Technical Bulletin* 49: 23.

—. 2000. sg97.2: Update to formatting regression output. *Stata Technical Bulletin* 58: 9–13.

Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables.* Thousand Oaks (Calif.): Sage.

Newson, R. 2003. Confidence intervals and p-values for delivery to the end user. *The Stata Journal* 3(3): 245–269.

About the Author

Ben Jann (jann@soz.gess.ethz.ch) is research assistant at the Department of Sociology of the Swiss Federal Institute of Technology Zurich (ETH) and a Ph.D. candidate at the University of Bern in Switzerland.