Making regression tables from stored estimates

Ben Jann
ETH Zurich, Switzerland

Abstract. Organizing and archiving statistical results and processing a subset of those results for publication are important and often underestimated issues in conducting statistical analyses. Because automation of these tasks is often poor, processing results produced by statistical packages is quite laborious and vulnerable to error. I will therefore present a new package called estout that facilitates and automates some of these tasks. 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 can thus be 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 \texttt{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: st0085, estout, estoutdef, estadd, estimates, regression table, latex, html

1 Introduction

Statistical packages are usually very good at fitting all kinds of regression models, but they are rather poor at keeping the results for those models organized or processing them for publication. This is a real problem because gathering the relevant figures by hand from the large amount of statistical output that is 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, when 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 are 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 \texttt{estimates} command (see [R] \texttt{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 \texttt{estimates table}. Although the \texttt{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 assembling a raw matrix of models and parameters that can be used as a starting point for creating a complex and well-formatted regression table.

In the remainder of this paper, I will present the new \texttt{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 \texttt{outreg} is probably the most widely used package of this kind (Gallup 1998, 1999, 2000, 2001). Among the other packages are \texttt{outtex} by Antoine Terracol, \texttt{est2tex} by Marc Muendler, and \texttt{mktab} by Nicholas Winter. Also see Newson (2003) for a very appealing approach. However, \texttt{estout} represents a good compromise between functionality and usability.

\section{Description and basic examples}

\texttt{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 \texttt{estimates store}. It then writes the table to the Stata log or to a specified text file.

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

\begin{verbatim}
\texttt{estout [namelist] [using filename] [, cells(array) stats(scalarlist) style(style) more_options]}
\end{verbatim}

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

\subsection*{Basic usage}

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

\begin{verbatim}
. sysuse auto
(1978 Automobile Data)
. replace price = price/1000
price was int now float
(74 real changes made)
\end{verbatim}
Making regression tables

. 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

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>1.746559</td>
<td>4.613589</td>
</tr>
<tr>
<td>mpg</td>
<td>-.0495122</td>
<td>.2631875</td>
</tr>
<tr>
<td>forXmpg</td>
<td>-.3072165</td>
<td></td>
</tr>
<tr>
<td>foreign</td>
<td></td>
<td>11.24033</td>
</tr>
<tr>
<td>_cons</td>
<td>1.946068</td>
<td>-14.44958</td>
</tr>
</tbody>
</table>

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 better if it were opened, for example, in a spreadsheet program.

Choosing a style

To align the columns in Stata’s Results window, 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)

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>1.746559</td>
<td>4.613589</td>
</tr>
<tr>
<td>mpg</td>
<td>-.0495122</td>
<td>.2631875</td>
</tr>
<tr>
<td>forXmpg</td>
<td>-.3072165</td>
<td></td>
</tr>
<tr>
<td>foreign</td>
<td></td>
<td>11.24033</td>
</tr>
<tr>
<td>_cons</td>
<td>1.946068</td>
<td>-14.44958</td>
</tr>
</tbody>
</table>

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 _cons)

\begin{tabular}{l l l}
| & m1 & m2 \\
| weight & 1.746559& 4.613589 \\
| mpg & -.0495122 & .2631875 \\
| forXmpg | & -.3072165 \\
| foreign | & 11.24033 \\
| _cons & 1.946068 & -14.44958 \\
\end{tabular}
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)' reports raw coefficients and standard errors:

```
. estout *, cells(b se) style(fixed)
  m1   m2
b/se  b/se
weight     1.746559  4.613589
           .6413538  .7284961
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  7.254961  6.359219  1.89e-08
mpg       .2631875  .1107961  2.375421  .0203122
forXmpg    -.3072165
            .1085307
foreign    11.24033
           2.751681
_cons      14.44958
           4.42572
           3.26491
           0.0017061
```

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. Such a table can be created using the following procedure:
Making regression tables

The `estout` command is used to display regression tables in Stata. The 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 the model)
- `p` (the p-value of the model)
- 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)
```

### 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.
3.1 Using labels

The labels option will cause estout to use variable labels and model labels, if they are 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)
```

<table>
<thead>
<tr>
<th></th>
<th>Model 1</th>
<th>Model 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coef.</td>
<td>1.746559**</td>
<td>4.613589***</td>
<td></td>
</tr>
<tr>
<td>Std. err.</td>
<td>.6413538</td>
<td>.7254961</td>
<td></td>
</tr>
<tr>
<td>Weight (lbs.)</td>
<td>-.0495122</td>
<td>-.3072165**</td>
<td></td>
</tr>
<tr>
<td>Mileage (mpg)</td>
<td>.086156</td>
<td>.1107961</td>
<td></td>
</tr>
<tr>
<td>Foreign*Mileage</td>
<td>11.24033***</td>
<td>2.751681</td>
<td></td>
</tr>
<tr>
<td>Foreign car type</td>
<td>1.946068</td>
<td>-14.44958**</td>
<td></td>
</tr>
<tr>
<td>R-squared</td>
<td>.2933891</td>
<td>.5516277</td>
<td></td>
</tr>
<tr>
<td>N. of cases</td>
<td>74</td>
<td>74</td>
<td></td>
</tr>
</tbody>
</table>

* 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.

```
\begin{table}
\centering
\caption{The auto data}
\begin{tabular}{lcc}
\hline
                     & Model 1 & Model 2 \\
\hline
Coef.              & 1.747    & 4.614  \\
p-value           & .008     & .000   \\
Weight (lbs.)      & -.050    & .567   \\
Mileage (mpg)      & .567     & .263   \\
Foreign*Mileage    & -.307    & .006   \\
Foreign car type   & 11.240   & .000   \\
Constant           & 1.946    & -14.450  \\
Adj. $R^2$         & .273     & .526   \\
N. of cases        & 74       & 74     \\
\hline
\end{tabular}
\end{table}
```

Table 1 was produced by inserting the line

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

in the \LaTeX document for this article after having run the following command:
Making regression tables

Note that most of the options in the above command could also have been provided via 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   Coef.& $p$-value& Coef.& $p$-value
weight 1.746559**  4.613589***               
mpg    -.0496122   .2631875*               
       (-.5746806)  (2.376421)            
forXmpg -.3072165***               
foreign 11.24033***               
_cons   1.946068 -14.44958**               
```

Note that most of the options in the above command could also have been provided via 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   Coef.& $p$-value& Coef.& $p$-value
weight 1.746559**  4.613589***               
mpg    -.0496122   .2631875*               
       (-.5746806)  (2.376421)            
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:

```plaintext
. estout *, cells(none) stats(r2_a bic N, star) style(fixed)
```

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>r2_a</td>
<td>.2734846***</td>
<td>.5256351***</td>
</tr>
<tr>
<td>bic</td>
<td>356.2918</td>
<td>331.2406</td>
</tr>
<tr>
<td>N</td>
<td>74</td>
<td>74</td>
</tr>
</tbody>
</table>

Note that in the example the models’ overall significance is denoted by stars attached to values of the adjusted $R^2$-squared (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:

```plaintext
. 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)
```

<table>
<thead>
<tr>
<th></th>
<th>m4</th>
<th>mpg displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>b/t</td>
<td>price</td>
<td>b/t</td>
</tr>
<tr>
<td></td>
<td>foreign</td>
<td>-1.650029</td>
</tr>
<tr>
<td></td>
<td>(5.749981)</td>
<td>(-1.565555)</td>
</tr>
<tr>
<td></td>
<td>weight</td>
<td>-6.587886</td>
</tr>
<tr>
<td></td>
<td>(6.182963)</td>
<td>(-10.55641)</td>
</tr>
<tr>
<td></td>
<td>length</td>
<td>-2.809689</td>
</tr>
<tr>
<td></td>
<td>_cons</td>
<td>41.6797</td>
</tr>
<tr>
<td></td>
<td>(19.46914)</td>
<td>(-3.46855)</td>
</tr>
<tr>
<td></td>
<td>r2</td>
<td>.548808</td>
</tr>
<tr>
<td></td>
<td>(1.255897)</td>
<td>(.8115213)</td>
</tr>
<tr>
<td></td>
<td>chi2</td>
<td>89.73586</td>
</tr>
<tr>
<td></td>
<td>(145.3912)</td>
<td>(318.6174)</td>
</tr>
<tr>
<td></td>
<td>p</td>
<td>2.50e-19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.68e-32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.50e-70</td>
</tr>
</tbody>
</table>

In the case of the multiple-equation models `reg3`, `sureg`, and `mvreg`, summary statistics for all 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 first 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, you must have saved the model after applying `mfx`, as 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      4.271707      .4594595
     (.7128444)     (.1043178)
  _cons    -4.689347
     (1.326547)
```

With single-equation models, the incorporation of results from `mfx` 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, so it is 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 relate 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
```

With single-equation models, the incorporation of results from `mfx` 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, so it is 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 relate 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`:
3.7 Adding supplementary statistics

Results that are included in the \texttt{e()}-returns for the models can be tabulated by \texttt{estout}. Thus one approach for, for example, reporting certain transformations of the coefficients is to add a matrix of the transformed results to the \texttt{e()}-returns and then tabulate the results using \texttt{estout}. The \texttt{estadd} command is designed to support this approach. It
Making regression tables

may be used, for example, 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

\[ \text{estadd } [\text{namelist}], \text{stats(}\text{statslist}) \]

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:

\[ \begin{align*}
. \text{ quietly generate x = uniform()}
. \text{ quietly regress x price weight mpg foreign}
. \text{ estadd, stats(mean sd(nobinary))}
. \text{ estimates store m3}
. \text{ estout m3, cells("mean sd") stats(N) mlabels(,none) drop(_cons) style(fixed)}
\end{align*} \]

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>6.165257</td>
<td>2.949496</td>
</tr>
<tr>
<td>weight</td>
<td>3.019459</td>
<td>.7771936</td>
</tr>
<tr>
<td>mpg</td>
<td>21.2973</td>
<td>5.785503</td>
</tr>
<tr>
<td>foreign</td>
<td>.2972973</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>74</td>
<td></td>
</tr>
</tbody>
</table>

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

---

1More functions are provided by the estadd_plus package (available from the SSC archive).
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 with the `estadd` package, as a starting point for programming new routines.

To report the Cox and Snell pseudo-$R^2$-squared, for example, define the `estadd` subroutine

```stata
program estadd_coxsnell, eclass
    eclass scalar coxsnell = 1 - exp(e(ll_0)-e(ll))^(2/e(N))
end
```

and then type

```stata
    . 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)
```

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 $j$, which are sometimes reported for logistic regression (see Long 1997):

```stata
    2Also see the `eret2` package (available from the SSC Archive). The `eret2` command allows you to add statistics to the `e()`-returns of a model without having to program subroutines. However, `eret2` can be applied only to the currently active estimates.
```
Making regression tables

program estadd_ebsd, eclass
    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 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
            labeling_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.

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)] | noform 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 ...]

row is

[] element [element ...] ["'"]

element is

el[(el_subopts)]

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[(leftstring rightstring)] place el in parentheses
| nopar

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
Making regression tables

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 [...])
labels(stringlist[, label_subopts])
star[(scalarlist) | nostar]
```

set the display formats
label the scalar statistics
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 # ∈ (0, 1] and listed in descending order.

The layout_options are

```
varwidth(#)
modelwidth(#)
[no] abbrev
[no] unstack
begin(string)
delimiter(string)
end(string)
dmarker(string)
msign(string)
[no] lz
substitute(subst_list)
```

set the width of the table’s left stub
set the width of the results columns
abbreviate long names and labels
place individual equations from multiple-equation models in separate columns
specify the beginning of the table rows
specify the column delimiter
specify the ending of the table rows
determine the decimal marker
determine the minus sign
print the leading zero of fixed format numbers in (−1, 1)
apply end-of-pipe substitutions

where subst_list is

```
from to [from to ...]
```
The **labeling_options** are

- `title(string)` specify a title for the table
- `[no] legend` add a legend explaining the significance symbols
- `prehead(stringlist)` add text lines before the table heading
- `posthead(stringlist)` add text lines after the table heading
- `prefoot(stringlist)` add text lines before the table footer
- `postfoot(stringlist)` add text lines after the table footer
- `[no] label` use variable labels
- `varlabels(matchlist[, varl_subopts])` relabel the parameters
- `mlabels(stringlist[, mlabels_subopts])` label the models
- `collabels(stringlist[, label_subopts])` label the columns within models
- `eqlabels(stringlist[, label_subopts])` label the equations
- `mgroups(stringlist[, 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`

the `mlabels_subopts` are

- `[no] numbers` number the models
- `[no] depvars` use dependent variables as models’ labels
- `label_subopts`

the `mgroups_subopts` are

- `pattern(pattern)` establish the grouping of the models
- `label_subopts`

and 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
Making regression tables

```plaintext
end(string)  
  add an overall suffix

[no] last  
  print the last occurrence of end

[no] span  
  span columns if appropriate

erpeat(string)  
  add a “span” suffix

lhs(string)  
  insert string into the left stub of the table
```

The output options are

```plaintext
[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

```plaintext
style(style)  
  specify a style for the output table
```

where style is one of the following

```plaintext
  tab  
  fixed  
  tex  
  html  
  mystyle
```

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):

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

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

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

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 specify 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:

<table>
<thead>
<tr>
<th>settings</th>
<th>styles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tab</td>
</tr>
<tr>
<td>begin</td>
<td></td>
</tr>
<tr>
<td>delimiter</td>
<td>_tab</td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
<tr>
<td>varwidth</td>
<td>0</td>
</tr>
<tr>
<td>modelwidth</td>
<td>0</td>
</tr>
<tr>
<td>abbrev</td>
<td>off</td>
</tr>
</tbody>
</table>

It is very easy to generate your 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 option settings, 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, such as `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

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

To temporarily turn off an option that has been activated in a defaults file, specify \texttt{nooption} in the command line (do not, however, use \texttt{nooption} 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 \texttt{estout}, type

\begin{verbatim}
. estout ..., nolegend
\end{verbatim}

On the other hand, options that take arguments, such as \texttt{prehead(args)}, \texttt{delimiter(args)}, or \texttt{stats(args, ...)} are specified as

\begin{verbatim}
option args
\end{verbatim}

in the defaults file (where \texttt{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 \texttt{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: \texttt{eform[(args)]} and \texttt{margin[(args)]}. These options can be specified as either

\begin{verbatim}
option option
\end{verbatim}

or

\begin{verbatim}
option args
\end{verbatim}

in the defaults file; the \texttt{no} form is allowed.

Many \texttt{estout} options have suboptions; i.e., an option might take the form \texttt{option(..., suboption)} or \texttt{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

\begin{verbatim}
optionsuboption suboption
\end{verbatim}

or

\begin{verbatim}
optionsuboption args
\end{verbatim}

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., \texttt{"optionsuboption"="option"+"suboption"}. For example, the \texttt{labels()} suboption of the \texttt{stats()} option would be set by the term \texttt{statslabels}. Analogously, the three level nesting in the \texttt{stats()} option yields suboption names composed of three names. For instance, the suboption called by the command

\begin{verbatim}
. estout ..., stats(..., labels(..., prefix( args)))
\end{verbatim}
would be referred to as

\begin{verbatim}
statslabelsprefix args
\end{verbatim}
in the defaults file. The \texttt{cells()} option represents an exception to this rule. It may be defined in the defaults file using only a simple array of cells elements without suboptions, e.g.

\begin{verbatim}
cells "b se" p
\end{verbatim}

However, the suboptions of the cells elements may be referred to as \texttt{el\_suboption}, for example

\begin{verbatim}
b_star star
\end{verbatim}
or

\begin{verbatim}
se_par [ ]
\end{verbatim}

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.

\section{Acknowledgments}

Some of the \texttt{estout} code has been adapted from the official \texttt{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.

\section{References}


Making regression tables


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.