The Stata Journal (2010)
10, Number 1, pp. 46–60

Tabulating SPost results using estout and esttab

Ben Jann
ETH Zürich
Zürich, Switzerland
jann@soz.gess.ethz.ch

J. Scott Long
Indiana University
Bloomington, IN
jslong@indiana.edu

Abstract. The `SPost` user package (Long and Freese, 2006, *Regression Models for Categorical Dependent Variables Using Stata* [Stata Press]) is a suite of postestimation commands to compute additional tests and effects representations for a variety of regression models. To facilitate and automate the task of tabulating results from `SPost` commands for inclusion in reports, publications, and presentations, we introduce tools to integrate `SPost` with the `estout` user package (Jann, 2005, *Stata Journal* 5: 288–308; 2007, *Stata Journal* 7: 227–244). The `estadd` command can retrieve results computed by the `SPost` commands `brant`, `fitstat`, `listcoef`, `mlogtest`, `prchange`, `prvalue`, and `asprvalue`. These results can then be tabulated by `esttab` or `estout`.

Keywords: st0183, SPost, regression table, estadd, estout, esttab, brant, fitstat, listcoef, mlogtest, prchange, prvalue, asprvalue

1 Introduction

The detailed interpretation of regression models often requires the incorporation of information that goes beyond the standard regression coefficients and reported tests. For example, the interpretation of an ordered logit model might include odds ratios, standardized odds coefficients, predicted probabilities for each outcome, and the results of the Brant test. Similarly, the selection among a series of count models might involve comparison of Bayesian information criterion statistics from competing models. While official Stata and user-written commands provide tables, these tables are generic and are rarely in the specific form an analyst wants for presentations. In this article, we demonstrate extensions of the `estout` and `SPost` packages that allow you to use `estout` and `esttab` to produce professional tables that combine the results from estimation and postestimation commands.

In this article, we assume that the reader is familiar with the `SPost` and `estout` packages. While readers who are unfamiliar with these packages should be able to follow the examples we provide, to fully utilize these new features we suggest that you read Jann (2005, 2007) and Long and Freese (2006). We also suggest consulting the `estout` web site, http://repec.org/bocode/e/estout—which contains dozens of examples that combine `SPost` commands with `estout` and `esttab`—and the `SPost` web site, http://www.indiana.edu/~jslsoc/spost.htm. For a more theoretical discussion of many of the models and tests used, see Long (1997).

Our extensions to the estout and SPost packages add features that allow moving computed results from the SPost commands brant, fitstat, listcoef, mlogtest, prchange, prvalue, and asprvalue to estout or esttab tables. Supported regression models include asmprobit, clogit, cloglog, cnreg, intreg, logit, mlogit, mprobit, nbreg, ologit, oprobit, poisson, probit, regress, rologit, slogit, tobit, zinb, zip, ztnb, and ztp, although not all commands are applicable for all models (see Long and Freese [2006] for details on these models).[1]

To use the new features, you need to install the latest versions of the estout and SPost packages. estout can be obtained from the Statistical Software Components Archive at Boston College. Type

```
. ssc install estout, replace
```

to install the package. The SPost software is available from J. Scott Long's web site. To locate and install the package, type

```
. findit spost9_ado
```

and follow the prompts that you will be given. Alternatively, type

```
. net install spost9_ado, from(http://www.indiana.edu/~jslsoc/stata) replace
```

The new features are only available with SPost for Stata 9 and later (spost9_ado). SPost for Stata 8 (spostado) is not supported.

2 Syntax and examples

The general procedure to tabulate results from an SPost command in estout or esttab is to fit a model with a Stata regression command and then run the SPost command to obtain additional postestimation results. estadd combines the SPost command's results with the model's e() returns. After that, the results are accessible to estout or esttab via the cells() or the stats() option, depending on whether the results are added in a matrix or as scalars. Alternatively, the main(), aux(), and scalars() options can be used in esttab to access the results, although cells() and stats() are more general and flexible, albeit more complex. We begin with two examples that illustrate the potential of combining SPost results with estout. We then provide details on the estadd syntax for each of the supported SPost commands and provide further examples. An extensive set of additional examples can be found at estout's web site (see http://repec.org/bocode/e/estout/spost.html).

As a simple example, suppose that you want to tabulate information measures computed by fitstat for a linear regression model fit by regress. You could type

1. Stata 11 factor variables are not (yet) supported by the SPost commands.

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/regjob3
(Long´s data on academic jobs of biochemists \ 2009-03-13)
. regress job fem phd ment fel art cit
  (output omitted)
. estadd fitstat, bic
AIC:                            2.580   AIC*n:                       1052.793
BIC:                       -1371.725   BIC´:                         -60.162
BIC used by Stata:          1080.872   AIC used by Stata:           1052.793
added scalars:
              e(aic0) =  2.5803757
             e(aic_n) =  1052.7933
              e(bic0) =  -1371.7248
             e(bic_p) =  -60.162312
          e(statabic) =  1080.8722
          e(stataaic) =  1052.7933
. esttab, cells(none) scalars(aic0 aic_n bic0 bic_p)
──────────────────────────
                      (1)
                      job
──────────────────────────
N                       408
aic0                  2.580
aic_n                1052.8
bic0                -1371.7
bic_p                -60.16
──────────────────────────
```

Notice that we used the `cells(none)` option to suppress the regression coefficients. To customize the labels, you can revise the `esttab` command, for example, as follows:

```
. esttab, cells(none)
>     scalars("aic0 AIC" "aic_n AIC*n" "bic0 BIC" "bic_p BIC´")
──────────────────────────
                      (1)
                      job
──────────────────────────
N                       408
AIC                   2.580
AIC*n                1052.8
BIC                 -1371.7
BIC´                 -60.16
──────────────────────────
```

If you are working with multiple models, you can either add results to each model individually after estimation as above or first fit and store a set of models and then apply `estadd` to all of them in one call by using the colon syntax. Here is an example of the latter, using `eststo` (which is also part of the `estout` package) to store the models:

```
. eststo: quietly regress job fem phd ment
(est1 stored)
. eststo: quietly regress job fem phd ment fel art cit
(est2 stored)
. estadd fitstat: *   // compute fitstat for all models
. esttab, cells(none)
>     scalars("n_rhs # RHS vars" "aic0 AIC" "aic_n AIC*n"
```

```
>             "bic0 BIC" "bic_p BIC´")
>     nonumbers mtitles("Model 1" "Model 2")
------------------------------------------
                   Model 1      Model 2
------------------------------------------
N                      408          408
# RHS vars              3            6
AIC                 2.639        2.580
AIC*n              1076.7       1052.8
BIC               -1359.9      -1371.7
BIC´               -48.30       -60.16
------------------------------------------

. eststo clear  // drop the stored estimates
```

A difference between the two approaches is that with the first method, output from `estadd fitstat` is displayed, whereas execution with the colon syntax is silent. We turned off the model numbers in the table by using the `nonumber` option and added model labels by using `mtitles()`. Furthermore, after using `esttab` to create a table for a given set of results, you may need to clear memory of results that have been stored so that they do not interfere with later tables. This can be done by using the `eststo clear` command.

2.1 Common syntax

Common to all featured commands is the basic syntax

`estadd` *cmd* [*...* `,` `replace` `prefix(`*string*`)` `quietly` *options*] [`:` *namelist*]

where *cmd* is the name of the `SPost` command in question and *namelist* is an optional list of stored estimation sets to which the command will be applied. `replace` permits `estadd` to overwrite existing `e()` returns, `prefix()` specifies a prefix for the names of the added results, and `quietly` suppresses the output of the `SPost` command. *options* are additional options specific to the `SPost` command. `replace`, `prefix()`, `quietly`, and *namelist* are global options and will not be repeated in the syntax diagrams below. Each of the supported `SPost` commands is now considered in alphabetical order.

2.2 estadd brant

The `brant` command tests the parallel regression assumption after an ordered logit or ordered probit model (`ologit` or `oprobit`). The syntax to add results from `brant` is

`estadd brant` [`,` *brant_options*]

where *brant_options* are as described in Long and Freese (2006, 452–454) or in `help brant`. `estadd brant` adds the results of the overall test to the scalars `e(brant_chi2)` (value of test statistic), `e(brant_df)` (degrees of freedom), and `e(brant_p)` (p-value), and adds matrix `e(brant)` containing χ^2 statistics for the individual regressors in the

first row and the corresponding p-values in the second row. The rows of `e(brant)` can be addressed in `estout`'s `cells()` option as `brant[chi2]` and `brant[p>chi2]`. Alternatively, type `brant[#]`, where # is the row number. For example,

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/ordwarm3
(GSS data on attitudes about working women for 1977 & 1989  \ 2009-03-13)
. ologit warm yr89 male white age ed prst
  (output omitted)
. estadd brant, quietly
added scalars:
         e(brant_chi2) =  49.181219
           e(brant_df) =  12
            e(brant_p) =  1.944e-06
added matrix:
              e(brant) :  2 x 6      (chi2, p>chi2)
. esttab, cells("brant[chi2](fmt(1)) brant[p>chi2](fmt(3))" )
>     stats(brant_chi2 brant_p, fmt(1 3) layout("@ @") label("Overall"))
>     nomtitles nonumbers
```

	chi2	p>chi2
yr89	13.0	0.001
male	22.2	0.000
white	1.3	0.531
age	7.4	0.025
ed	4.3	0.116
prst	4.3	0.115
Overall	49.2	0.000

We use the `fmt(#)` option to specify the number of decimal digits to report and the `label("Overall")` option to set the name for the omnibus Brant test results. The `layout("@ @")` option specifies that the `brant_chi2` and `brant_p` statistics be placed in the same row.

2.3 estadd fitstat

The `fitstat` command computes numerous measures of fit for many kinds of regression models. The syntax to add results from `fitstat` is

`estadd fitstat` [, *fitstat_options*]

where *fitstat_options* are as described in Long and Freese (2006, 452–454) or in `help fitstat`. The list of added statistics depends on model and options. For example, researchers frequently want to include at the base of a table the sample size along with fit measures and statistical tests. This can be done as follows, where we also illustrate how you can add information on the provenance of the table to the footer.

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/binlfp3
(Mroz´s 1976 data on labor force participation of women \ 2009-03-13)
```

```
. logit lfp k5 k618 age wc hc lwg inc, nolog
  (output omitted)
. estadd fitstat
  (output omitted)
. eststo logit
. probit lfp k5 k618 age wc hc lwg inc, nolog
  (output omitted)
. estadd fitstat
  (output omitted)
. eststo probit
. local lrlbl "LRX2(`e(lrx2_df)´)"
. local date : di %dCY-N-D d(`c(current_date)´) // get date
. local notes addnotes("t statistics in parentheses"
>     "Two-tailed tests: * p<0.05, ** p<0.01, *** p<0.001"
>     "Source: bjsl04-fitstat.do `date´ Scott Long.")
. esttab,
>     scalars("r2_mf R2_McFadden" "bic BIC" "lrx2 `lrlbl´" "lrx2_p Prob")
>     wide mtitles title(Comparing logit and probit on lfp) nonotes `notes´
Comparing logit and probit on lfp
```

	(1) logit		(2) probit	
k5	-1.463***	(-7.43)	-0.875***	(-7.70)
k618	-0.0646	(-0.95)	-0.0386	(-0.95)
age	-0.0629***	(-4.92)	-0.0378***	(-4.97)
wc	0.807***	(3.51)	0.488***	(3.60)
hc	0.112	(0.54)	0.0572	(0.46)
lwg	0.605***	(4.01)	0.366***	(4.17)
inc	-0.0344***	(-4.20)	-0.0205***	(-4.30)
_cons	3.182***	(4.94)	1.918***	(5.04)
N	753		753	
R2_McFadden	0.121		0.121	
BIC	958.3		958.4	
LRX2(7)	124.5		124.4	
Prob	8.92e-24		9.47e-24	

```
t statistics in parentheses
Two-tailed tests: * p<0.05, ** p<0.01, *** p<0.001
Source: bjsl04-fitstat.do 2009-10-14 Scott Long.
. eststo clear  // drop the stored estimates
```

The commands

```
logit lfp k5 k618 age wc hc lwg inc, nolog
estadd fitstat
eststo logit
```

fit the logit model, compute fit statistics, and store the model estimates along with the postestimation statistics using the reference name `logit`. Similarly, the following commands do the same things for the probit model:

```
probit lfp k5 k618 age wc hc lwg inc, nolog
estadd fitstat
eststo probit
```

To label the likelihood-ratio χ^2 statistic `LRX2`, we want to incorporate the degrees of freedom into the label. We do this by constructing a local with the label in which the degrees of freedom are retrieved from the stored `e(lrx2_df)`:

```
local lrlbl "LRX2(`e(lrx2_df)´)"
```

Within the `scalars()` option of `esttab`, `"lrx2 ‘lrlbl’"` indicates that the statistic named `lrx2` be given the label saved in the local `lrlbl`. We also want to customize the footer of the table. To do this, we create a local `notes` as follows, where each line of the footer is included in double quotes:

```
local date : di %dCY-N-D d(`c(current_date)´)
local notes addnotes("t statistics in parentheses"       ///
    "Two-tailed tests: * p<0.05, ** p<0.01, *** p<0.001" ///
    "Source: bjsl04-fitstat.do `date´ Scott Long.")
```

Finally, in the `esttab` command, the `nonotes` option suppresses the footer that is created by default, and our customized footer is created with the `addnotes()` option contained in the local `notes`.

2.4 estadd listcoef

The `listcoef` command lists transformations of the estimated coefficients for a variety of regression models. The syntax to add results from `listcoef` is

`estadd listcoef` [*varlist*] [`, nosd` *listcoef_options*]

where *listcoef_options* are as described in Long and Freese (2006, 464–467) or in `help listcoef`. Furthermore, the `nosd` option suppresses adding the standard deviations of the variables in `e(b_sdx)`.

Depending on the estimation command and options, `estadd listcoef` adds several vectors containing statistics such as standardized coefficients or factor change coefficients. See `estadd`'s online documentation for details. As a simple example, consider tabulating standardized coefficients for a `logit` model:

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/binlfp3
(Mroz´s 1976 data on labor force participation of women \ 2009-03-13)
. logit lfp k5 k618 age wc hc lwg inc, nolog
  (output omitted)
. estadd listcoef, std quietly
added matrices:
               e(b_xs) :  1 x 7      (bStdX)
               e(b_ys) :  1 x 7      (bStdY)
              e(b_std) :  1 x 7      (bStdXY)
              e(b_sdx) :  1 x 7      (SDofX)
```

```
. estadd fitstat
  (output omitted)
. local lrlbl "LRX2(`e(lrx2_df)´)"
. esttab, cells((b(lab(b-unstd) f(3))     t(lab(t-value) f(2))
>                b_xs(lab(b-xstd) f(3))   b_ys(lab(b-ystd) f(3))
>                b_std(lab(b-std) f(3))   b_sdx(lab(sd(x)) f(3))
>               ))
>     scalars("bic0 BIC" "lrx2 `lrlbl´" "lrx2_p Prob")
>     nomtitles nonumbers compress
>     title(Logit on lfp)
Logit on lfp
```

	b-unstd	t-value	b-xstd	b-ystd	b-std	sd(x)
k5	-1.463	-7.43	-0.767	-0.714	-0.374	0.524
k618	-0.065	-0.95	-0.085	-0.031	-0.042	1.320
age	-0.063	-4.92	-0.508	-0.031	-0.248	8.073
wc	0.807	3.51	0.363	0.394	0.177	0.450
hc	0.112	0.54	0.055	0.055	0.027	0.488
lwg	0.605	4.01	0.355	0.295	0.173	0.588
inc	-0.034	-4.20	-0.401	-0.017	-0.195	11.635
_cons	3.182	4.94				
N	753					
BIC	-4029.7					
LRX2(7)	124.5					
Prob	8.92e-24					

The `f(#)` option within `cells()` controls the number of decimal digits that are reported.

As another example, consider the tabulation of results for specific contrasts in a multinomial logit model. Use `listcoef`'s `lt`, `gt`, and `adjacent` options to determine the contrasts for which results are to be computed. For example,

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/nomocc3
(GSS data on career outcomes for 1982 \ 2009-03-13)
. mlogit occ white ed exper, nolog
  (output omitted)
. estadd listcoef, gt adjacent quietly
added matrices:
              e(b_raw) :  1 x 12     (b)
               e(b_se) :  1 x 12     (se)
                e(b_z) :  1 x 12     (z)
                e(b_p) :  1 x 12     (P>|z|)
             e(b_fact) :  1 x 12     (e^b)
            e(b_facts) :  1 x 12     (e^bStdX)
              e(b_sdx) :  1 x 12     (SDofX)
```

(*Continued on next page*)

```
. esttab, main(b_facts) unstack not nostar nonote modelwidth(14)
```

	(1) occ			
	BlueCol-Menial	Craft-BlueCol	WhiteCol-Craft	Prof-WhiteCol
white	1.407	0.810	1.355	1.058
ed	0.746	1.767	2.147	3.505
exper	1.068	1.378	1.101	1.015
N	337			

The raw coefficients for the requested contrasts are added in `e(b_raw)` (along with additional vectors containing standard errors, z statistics, and p-values).

2.5 estadd mlogtest

The `mlogtest` command computes various tests for multinomial logit models (`mlogit`). The syntax to add results from `mlogtest` is

`estadd mlogtest` [*varlist*] [, *mlogtest_options*]

where *mlogtest_options* are as described in Long and Freese (2006, 473–476) or in `help mlogtest`. `estadd mlogtest` adds a variety of results depending on the specified options (see the online documentation). For example, to compute the likelihood-ratio test that all the coefficients associated with a given independent variable are simultaneously equal to zero, you can use the command `mlogtest, lr`. To place these results in a table, type

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/nomocc3
(GSS data on career outcomes for 1982 \ 2009-03-13)
. mlogit occ white ed exper, nolog
  (output omitted)
. estadd mlogtest, lr
**** Likelihood-ratio tests for independent variables (N=337)
 Ho: All coefficients associated with given variable(s) are 0.
              |       chi2   df   P>chi2
--------------+---------------------------
        white |      8.095    4    0.088
           ed |    156.937    4    0.000
        exper |      8.561    4    0.073
--------------+---------------------------

added matrices:
            e(lrtest) :  3 x 3      (chi2, df, p)
```

```
. esttab, cell(( lrtest[chi2](label(LRX2))
>                lrtest[df]
>                lrtest[p](label(p-value))
>              ))
>     mlabel(none) nonumber noobs
```

	LRX2	df	p-value
white	8.095408	4	.0881451
ed	156.9372	4	6.63e-33
exper	8.560953	4	.073061

2.6 estadd prchange

The `prchange` command computes discrete and marginal changes in predictions for regression models. The syntax to add results from `prchange` is

`estadd prchange` [*varlist*] [*if*] [*in*] [`, pattern(`*typepattern*`) binary(`*type*`) continuous(`*type*`)` [`no`]`avg split` [`(`*prefix*`)`] *prchange_options*]

where *prchange_options* are as described in Long and Freese (2006, 478–485) or in `help prchange`. For example, the `outcome()` option may be used with models for count, ordered, or nominal variables to request results for a specific outcome. Further options are the following:

`pattern(`*typepattern*`)`, `binary(`*type*`)`, and `continuous(`*type*`)` determine which types of discrete change effects are added as the main results. The default is to add the 0 to 1 change effect for binary variables and the standard deviation change effect for continuous variables. Use `binary(`*type*`)` and `continuous(`*type*`)` to change these defaults. Available types are

type	Description
`minmax`	minimum to maximum change effect
`01`	0 to 1 change effect
`delta`	`delta()` change effect
`sd`	standard deviation change effect
`margefct`	marginal effect (only some models)

Use `pattern(`*typepattern*`)` if you want to determine the type of the added effects individually for each regressor. For example, `pattern(minmax sd delta)` would add `minmax` for the first regressor, `sd` for the second, and `delta` for the third, and then proceed using the defaults for the remaining variables.

`avg` requests that only the average results over all outcomes are added if applied to ordered or nominal models (`ologit`, `oprobit`, `slogit`, `mlogit`, and `mprobit`). The default is to add the average results as well as the individual results for the different

outcomes (unless `prchange`'s `outcome()` option is specified, in which case only results for the indicated outcome are added). Furthermore, specify `noavg` to suppress the average results and add only the outcome-specific results. `avg` may not be combined with `split` or `outcome()`.

`split`[`(`*prefix*`)`] saves each outcome's results in a separate estimation set if applied to ordered or nominal models (`ologit`, `oprobit`, `slogit`, `mlogit`, and `mprobit`). This can be useful if you want to tabulate the results for the different outcomes in separate columns beside one another. The estimation sets are named *prefix*#, where # is the value of the outcome at hand. If no *prefix* is provided, the name of the estimation set followed by an underscore is used as the prefix. If the estimation set has no name (because it has not yet been stored), then the name of the estimation command followed by an underscore is used as the prefix (e.g., `ologit_`). The estimation sets stored by the `split` option are intended for tabulation only and should not be used with other postestimation commands.

`estadd prchange` returns the discrete change effects in matrix `e(dc)`. The first row of the matrix contains the main results as determined by `pattern()`, `binary()`, and `continuous()`. The second and following rows contain the separate results for each type of effect, using the labels provided by `prchange` as row names. Type `dc[#]` or `dc[`*label*`]` to address the rows in `estout`'s `cells()` option, where # is the row number or *label* is the row name. For example, type `dc[-+sd/2]` to address the centered standard-deviation change effects. To tabulate the main results (first row), simply type `dc`. See the online documentation for further details on added results.

Space constraints do not permit giving detailed examples for all the variants. We illustrate only the application of the `split` option with a stereotype logistic regression:

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/ordwarm3
(GSS data on attitudes about working women for 1977 & 1989  \ 2009-03-13)
. slogit warm yr89 male white age ed prst, nolog
  (output omitted)
. estadd prchange, split quietly
added scalars:
            e(predval) =  .11714775
            e(outcome) =  1
              e(delta) =  1
           e(centered) =  1
added matrices:
                 e(dc) :  5 x 6      (main, Min->Max, 0->1, -+1/2, -+sd/2)
            e(pattern) :  1 x 6
                  e(X) :  4 x 6      (X, SD, Min, Max)
first row in e(dc) contains:
  01 change for binary variables
  sd change for continuous variables
results for outcome 1 stored as slogit_1
results for outcome 2 stored as slogit_2
results for outcome 3 stored as slogit_3
results for outcome 4 stored as slogit_4
```

```
. esttab, main(dc 3) not nostar scalars("predval Pr(y|x)") noobs
>     mtitles nonote title(Discrete changes in outcome probabilities.)
>     addnotes("Note: Change from 0 to 1 for binary variable,"
>              "      else a standard deviation change.")
Discrete changes in outcome probabilities.
```

	(1) 1SD	(2) 2D	(3) 3A	(4) 4SA
yr89	-0.055	-0.081	0.058	0.078
male	0.077	0.105	-0.082	-0.100
white	0.036	0.056	-0.036	-0.055
age	0.039	0.055	-0.042	-0.052
ed	-0.021	-0.030	0.022	0.028
prst	-0.010	-0.014	0.011	0.013
Pr(y|x)	0.117	0.323	0.392	0.167

```
Note: Change from 0 to 1 for binary variable,
      else a standard deviation change.
. eststo clear
```

2.7 estadd prvalue

The `prvalue` command computes model predictions at specified values of the independent variables for regression models for categorical and count variables. The procedure to add results from `prvalue` slightly differs from that for the other commands. First, results have to be collected by repeated calls to `estadd prvalue` by using the syntax

`estadd prvalue` [*if*] [*in*] [`,` `label(`*string*`)` *prvalue_options*]

where *prvalue_options* are as described in Long and Freese (2006, 493–497) or in `help prvalue`. For example, use `x()` and `rest()` to set the values of the independent variables. Furthermore, use `label()` to label the single calls. `pred`# is used as the label if `label()` is omitted, where # is the number of the call. Labels may contain spaces, but they will be trimmed to a maximum length of 30 characters, and some special characters (`:` and `.` and `"`) will be replaced by an underscore. The results from the single calls are collected in some intermediary matrices. Specify `replace` to drop results from previous calls.

Second, after collecting results, `estadd prvalue post` is used to post the predictions and their standard errors (if available) in `e(b)` and `e(se)` so that they can be tabulated (see the online help for information on additional returns). The syntax for posting the predictions is

`estadd prvalue post` [*name*] [`,` `title(`*string*`)` `swap`]

The default for `estadd prvalue post` is to replace the current model with the posted results. However, if *name* is specified, the posted results are stored and the

current model remains active. Use `title()` to specify a title for the stored results. Use `swap` to determine how the results are arranged in `e(b)`. The default is to group predictions by outcomes (i.e., outcome labels are used as equations). However, if `swap` is specified, predictions are grouped by calls (i.e., prediction labels are used as equations).

As an example, consider tabulating the predicted probabilities of labor force participation depending on whether a woman attended college:

```
. use http://www.indiana.edu/~jslsoc/stata/spex_data/binlfp3
(Mroz´s 1976 data on labor force participation of women \ 2009-03-13)
. logit lfp k5 k618 age wc hc lwg inc, nolog
  (output omitted)
. estadd prvalue, x(wc=1) label(Wife attended college)
  (output omitted)
. estadd prvalue, x(wc=0) label(Wife did not go to college) save
  (output omitted)
. estadd prvalue, x(wc=1) label(Difference) dif
  (output omitted)
. estadd prvalue post
scalars:
                  e(N) =  753
macros:
             e(depvar) : "lfp"
                e(cmd) : "estadd_prvalue"
              e(model) : "logit"
         e(properties) : "b"
matrices:
                  e(b) :  1 x 6      (predictions)
                 e(se) :  1 x 6      (standard errors)
                 e(LB) :  1 x 6      (lower CI bounds)
                 e(UB) :  1 x 6      (upper CI bounds)
           e(Category) :  1 x 6      (outcome values)
                  e(X) :  7 x 3      (k5, k618, age, wc, hc, lwg, inc)
. esttab, ci(3) wide nostar noobs nonumber nomtitle nonote
>     varwidth(30) modelwidth(11 16) collabels("Pr(in LF)" "95% CI")
>     eqlabels(none) keep(1InLF:)
>     title(Change in probability if wife attends college)
>     note(Note: All other variables held at their mean)
Change in probability if wife attends college
------------------------------------------------------------------------------
                                  Pr(in LF)           95% CI
------------------------------------------------------------------------------
Wife attended college                 0.710    [0.633,0.786]
Wife did not go to college            0.522    [0.473,0.570]
Difference                            0.188    [0.090,0.286]
------------------------------------------------------------------------------
Note: All other variables held at their mean
```

The three `prvalue` commands compute predictions that are used in three rows of the created table. Notice that the first `prvalue` computes the predictions with `wc=1` and the second, with `wc=0`, while the third `prvalue` computes the difference between the two prior predictions. `estadd prvalue post` then saves the predictions. The `esttab`

command uses options already introduced, although there are a few things to note. `varwidth()` specifies the width of the left stub of the table containing the labels. `modelwidth()` specifies the widths of the models' columns.

2.8 estadd asprvalue

The `asprvalue` command computes predicted values for models with alternative-specific variables. Adding results from `asprvalue` is analogous to adding results from `prvalue`. That is, first, a series of predictions is collected by repeated calls by using the syntax

`estadd asprvalue` [*if*] [*in*] [`,` `label(`*string*`)` *asprvalue_options*]

where *asprvalue_options* are as described in Long and Freese (2006, 450–452) or in `help asprvalue`. Second, the collected results are posted by using the syntax

`estadd asprvalue post` [*name*] [`,` `title(`*string*`)` `swap`]

See section 2.7 for details. Examples can be found on `estout`'s web site.

3 Conclusions

Tables are critical to effectively convey substantive results. But creating tables can be an extremely tedious and error-prone process. The `estout` package makes it very simple to construct basic tables and export them to your word processor. The enhancements to the `estout` and `SPost` packages make it simple to create complex tables that contain both coefficients and postestimation results. We hope that this not only makes life simpler for users but also encourages researchers to create tables containing more detailed information.

4 References

Jann, B. 2005. Making regression tables from stored estimates. *Stata Journal* 5: 288–308.

———. 2007. Making regression tables simplified. *Stata Journal* 7: 227–244.

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

Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.

(*Continued on next page*)

About the authors

Ben Jann is a sociologist at ETH Zürich in Zürich, Switzerland.

J. Scott Long is Chancellor's Professor of Sociology and Statistics at Indiana University. He is the author of *Regression Models for Categorical and Limited Dependent Variables*, coauthor (with Jeremy Freese) of *Regression Models for Categorical Dependent Variables Using Stata*, and author of the recently published *The Workflow of Data Analysis Using Stata*.