Example 6 — Table comparing regression results

Description Remarks and examples Reference Also see

Description

In this example, we demonstrate how to collect results from multiple regressions and create a table of coefficients, standard errors, and statistics computed after fitting the model. We also show how to customize the resulting table.

Remarks and examples

Remarks are presented under the following headings:

Collecting regression results and creating a table

Below, we use data from the Second National Health and Nutrition Examination Survey (NHANES II) (McDowell et al. 1981). We would like to create a table comparing the results from two models. If we were including only the estimation results, we could easily create this table with `etable`; this command is used to create and export tables of estimation results in a single step. However, we want to include results from another command, `testparm`, in our table. So, we use the `collect` suite of commands.

We begin by creating a new collection named `ex6`. Then, we fit a model for systolic blood pressure (`bpsystol`) as a function of `weight`, `sex`, and whether an individual has `diabetes`. We use the `collect` prefix to collect the coefficients (`_r_b`) and standard errors (`_r_se`) into the `ex6` collection. We also attach the tag `model[(1)]` to these results. We can later use this tag to refer to these results when we build and customize our table.

```
. use https://www.stata-press.com/data/r19/nhanes2l
(Second National Health and Nutrition Examination Survey)
. collect create ex6
(current collection is ex6)
. collect _r_b _r_se, tag(model[(1)]): regress bpsystol weight i.diabetes i.sex
```

Source	SS	df	MS	Number of obs	=	10,349
				F(3, 10345)	=	382.25
Model	562138.283	3	187379.428	Prob > F	=	0.0000
Residual	5071141.76	10,345	490.2022	R-squared	=	0.0998
				Adj R-squared	=	0.0995
Total	5633280.05	10,348	544.38346	Root MSE	=	22.141

bpsystol	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
weight	.4340474	.0153533	28.27	0.000	.4039519	.4641428
diabetes						
Diabetic	14.34115	1.019611	14.07	0.000	12.34252	16.33979
sex						
Female	1.107633	.4710559	2.35	0.019	.1842724	2.030994
_cons	98.40567	1.235476	79.65	0.000	95.9839	100.8274

In fact, all results that `regress` stores in `e()` are collected when we run the command above. By specifying `_r_b` and `_r_se` following `collect`, we have requested that these results be automatically included in a table when we include the dimension `result`.

Let's also use the `testparm` command to test whether the coefficient on `diabetes` is different from zero. We collect the p-value that `testparm` returns in the scalar `r(p)`. We also tag this result with `model[(1)]`, which will allow us to easily align the result from this command with the regression results when we construct our table.

```
. collect p_d=r(p), tag(model[(1)]): testparm i.diabetes
 ( 1)  1.diabetes = 0
       F(  1, 10345) =  197.83
            Prob > F =    0.0000
```

Now, we add the interaction between `diabetes` and `sex` to the model. We use the `collect` prefix again to collect the results from this model. We add `quietly` prefix to suppress the output. Now that `diabetes` is interacted with `sex`, we perform a joint test for the hypothesis that all coefficients associated with `diabetes`, including those in the interaction, are equal to zero. This time we attach the tag `model[(2)]` to the results from both the `regress` and the `testparm` commands.

```
. quietly: collect _r_b _r_se, tag(model[(2)]): regress bpsystol weight
> diabetes##sex
. collect p_d=r(p), tag(model[(2)]): testparm i.diabetes i.diabetes#i.sex
 ( 1)  1.diabetes = 0
 ( 2)  1.diabetes#2.sex = 0
       F(  2, 10344) =  100.11
            Prob > F =    0.0000
```

All the results are now stored in the current collection. We are ready to arrange the values into a table. These values are organized in the collection by tags, which are made up of dimensions and levels within those dimensions. We need to know the dimension names to lay out and customize our table. Below, we list the dimensions:

```
. collect dims
Collection dimensions
Collection: ex6
──────────────────────────────────────────
                   Dimension   No. levels
──────────────────────────────────────────
Layout, style, header, label
                      cmdset   4
                       coleq   1
                     colname   10
           colname_remainder   1
                    diabetes   2
                       model   2
               program_class   3
                      result   37
                 result_type   3
                     rowname   1
                         sex   2
Style only
                border_block   4
                   cell_type   4
──────────────────────────────────────────
```

The output indicates which dimensions can be used with the `collect` subcommands. For example, the first section lists dimensions that can be specified in `collect layout`, which is used to arrange the values in the collection into a table. We place the covariate names (`colname`) and the statistics (`result`) on the rows. We place `model`, the dimension we created with the `tag()` option of `collect` above, on the columns.

```
. collect layout (colname#result) (model)
Collection: ex6
      Rows: colname#result
   Columns: model
   Table 1: 30 x 2
```

	(1)	(2)
Weight (kg)		
Coefficient	.4340474	.4335342
Std. error	.0153533	.0153559
Not diabetic		
Coefficient	0	0
Std. error	0	0
Diabetic		
Coefficient	14.34115	12.57211
Std. error	1.019611	1.538361
Male		
Coefficient	0	0
Std. error	0	0
Female		
Coefficient	1.107633	.9520999
Std. error	.4710559	.4817911
Not diabetic # Male		
Coefficient		0
Std. error		0
Not diabetic # Female		
Coefficient		0
Std. error		0
Diabetic # Male		
Coefficient		0
Std. error		0
Diabetic # Female		
Coefficient		3.146466
Std. error		2.048958
Intercept		
Coefficient	98.40567	98.5238
Std. error	1.235476	1.237787

In the following section, we format the results, remove the base levels, and make some other edits to make this table ready for publication.

Customizing the table

In the table above, the base levels of factor variables were included in the table. Below, we remove the base levels with `collect style showbase`. We also format the statistics to two decimal places. We can do this with `collect style cell`. This command allows us to format all cells in the table at once or to format specific cells. Because we want to apply the numeric formatting to all cells, we do not specify a dimension. We also remove the border on the right side of the row headers by setting the border pattern for this location (`right`) to `nil`. Next, we want to enclose the standard errors in parentheses. The standard errors are stored in the level `_r_se` of the dimension `result`. To apply this format only to this result, we specify the dimension (`result`) and its level; we use brackets (`[]`) to refer to levels of a dimension. Then, we get a preview of our table.

```
. collect style showbase off
. collect style cell, nformat(%5.2f)
. collect style cell border_block, border(right, pattern(nil))
. collect style cell result[_r_se], sformat("(%s)")
. collect preview
```

	(1)	(2)
Weight (kg)		
Coefficient	0.43	0.43
Std. error	(0.02)	(0.02)
Diabetic		
Coefficient	14.34	12.57
Std. error	(1.02)	(1.54)
Female		
Coefficient	1.11	0.95
Std. error	(0.47)	(0.48)
Diabetic # Female		
Coefficient		3.15
Std. error		(2.05)
Intercept		
Coefficient	98.41	98.52
Std. error	(1.24)	(1.24)

Next, we would like to center the results and column headers horizontally. We will need to refer to the levels of the dimension `cell_type`. Below, we list the levels of this dimension. This dimension divides the table into four sections. The sections we want to modify are the values (`items`) in the body of the table and the `column-headers`. We specify the dimension `cell_type` and these levels with `collect style cell` to modify their horizontal alignment.

```
. collect levelsof cell_type

Collection: ex6
 Dimension: cell_type
    Levels: column-header corner item row-header

. collect style cell cell_type[item column-header], halign(center)
```

We also remove the labels `Coefficient` and `Std. error`. These labels are attached to the levels of the dimension `result`. We use `collect style header` to hide the level labels for this dimension. Then, to add an additional space between columns, we use `collect style column` with the `extraspace()` option.

We can arrange our row headers in two ways. One way is to place each item in a separate cell; the other way is to stack the elements in a single column. We choose the latter with `collect style row stack`. Also, notice that by default `collect` uses a `#` as a delimiter for interaction terms. We would instead like to use an `x`, with a space on each side.

```
. collect style header result, level(hide)
. collect style column, extraspace(1)
. collect style row stack, spacer delimiter(" x ")
. collect preview
```

	(1)	(2)
Weight (kg)	0.43	0.43
	(0.02)	(0.02)
Diabetic	14.34	12.57
	(1.02)	(1.54)
Female	1.11	0.95
	(0.47)	(0.48)
Diabetic x Female		3.15
		(2.05)
Intercept	98.41	98.52
	(1.24)	(1.24)

Recall that all `e()` results were collected from our models. In addition to reporting the p-value from `testparm`, we also want to report the R^2 value, which is stored under the level `r2` of the dimension `result`. So, in addition to the results for each covariate (`colname#result`), we also specify the levels `r2` and `p_d` of `result` in the first set of parentheses, which will be used to define the rows of the table. As before, we use `model` for the column identifier.

```
. collect layout (colname#result result[r2 p_d]) (model)
Collection: ex6
      Rows: colname#result result[r2 p_d]
   Columns: model
   Table 1: 18 x 2
```

	(1)	(2)
Weight (kg)	0.43	0.43
	(0.02)	(0.02)
Diabetic	14.34	12.57
	(1.02)	(1.54)
Female	1.11	0.95
	(0.47)	(0.48)
Diabetic x Female		3.15
		(2.05)
Intercept	98.41	98.52
	(1.24)	(1.24)
	0.10	0.10
	0.00	0.00

We hid the labels for the levels of the dimension `result`, but now that we have added the p-values and values of R^2, we want to display their levels. We specify these two levels of the dimension `result` with `collect style header`. Then, we modify the labels for these levels. Additionally, we want to format our p-values to three decimal places and display them as `<0.001` if they are less than 0.001. We can do this by specifying `minimum(0.001)` with `collect style cell`. After making that change, we preview our table once more:

```
. collect style header result[r2 p_d], level(label)
. collect label levels result p_d "Diabetes p-value" r2 "R-squared", modify
. collect style cell result[p_d], nformat(%5.3f) minimum(0.001)
. collect preview
```

	(1)	(2)
Weight (kg)	0.43	0.43
	(0.02)	(0.02)
Diabetic	14.34	12.57
	(1.02)	(1.54)
Female	1.11	0.95
	(0.47)	(0.48)
Diabetic x Female		3.15
		(2.05)
Intercept	98.41	98.52
	(1.24)	(1.24)
R-squared	0.10	0.10
Diabetes p-value	<0.001	<0.001

Last, we want to add stars to the coefficients to indicate which are significant at a 1%, 5%, and 10% level. The significance is determined by the p-values (_r_p), but the actual stars are attached to the coefficients (_r_b). The shownote option adds the note at the bottom of the table, explaining the significance represented by the stars.

```
. collect stars _r_p 0.01 "***"  0.05 "** " 0.1 "*  ", attach(_r_b) shownote
. collect preview
```

	(1)	(2)
Weight (kg)	0.43***	0.43***
	(0.02)	(0.02)
Diabetic	14.34***	12.57***
	(1.02)	(1.54)
Female	1.11**	0.95**
	(0.47)	(0.48)
Diabetic x Female		3.15
		(2.05)
Intercept	98.41***	98.52***
	(1.24)	(1.24)
R-squared	0.10	0.10
Diabetes p-value	<0.001	<0.001

*** p<.01, ** p<.05, * p<.1

Now, we can export our table to our preferred format—Word, PDF, HTML, LaTeX, Excel, or Markdown—using collect export.

Reference

McDowell, A., A. Engel, J. T. Massey, and K. Maurer. 1981. "Plan and operation of the Second National Health and Nutrition Examination Survey, 1976–1980". In *Vital and Health Statistics*, ser. 1, no. 15. Hyattsville, MD: National Center for Health Statistics.

Also see

[TABLES] **collect style cell** — Collection styles for cells

[TABLES] **collect style showbase** — Collection styles for displaying base levels