collect composite — Manage composite results in a collection

Description

`collect composite` manages the creation and removal of composite results in a collection. A composite result is composed from a list of levels in dimension `result`.

Quick start

Define a result composed from a coefficient and its standard error estimate

```
collect composite define bse = _r_b _r_se
```

Drop the above composite result

```
collect composite drop bse
```

Menu

Statistics > Summaries, tables, and tests > Tables and collections > Build and style table

Syntax

Define a composite result

collect composite define *composite* = *elements* [, *options*]

Drop a composite result

collect composite drop *composite* [, name(*cname*)]

composite is the name for a composite result.

elements are levels in the `result` dimension and previously defined composite results.

options	Description
`name(`*cname*`)`	add composite result definition to collection *cname*
`delimiter(`*char*`)`	use character as delimiter between *elements*
[no]`trim`	preserve or trim extra spaces from numeric formats
[no]`override`	preserve or override trim property of elements
`replace`	overwrite *composite* if it already exists

Options

`name(`*cname*`)` specifies the collection to add or remove the composite result. By default, changes are applied to the current collection.

`delimiter(`*char*`)` changes the delimiter between elements. The default is `delimiter(" ")`.

`notrim` and `trim` control the handling of extra spaces caused by numeric formats applied to the elements. `notrim`, the default, preserves the extra spaces; `trim` removes the extra spaces.

Note that the `trim` and `notrim` options do not apply to results `_r_ci` and `_r_cri`.

`nooverride` and `override` control handling of the trim property when an element is a composite result. `nooverride`, the default, does not change the trim property of elements; `override` applies the specified trim property to all elements.

`replace` permits `collect composite define` to overwrite the definition of an existing composite result.

Remarks and examples

`collect composite` is used to create a single result from multiple results. This can be useful when you want to display multiple results in a single cell. For example, you may want to place standard deviations right next to means or confidence intervals right next to coefficients. Any formatting applied to the individual results will persist in the composite result. For example, you may format confidence intervals by enclosing them in brackets. If you then create a composite result with coefficients and confidence intervals, the result will contain coefficients followed by the confidence intervals enclosed in brackets.

You can also build a result from existing composite results. For example, you might first decide to create a composite result from means and standard deviations and call it `meansd`. As you are working, you may decide you want to add on medians to the composite result. You can then create a new composite result with `meansd` and the medians.

Remarks are presented under the following headings:

Example 1: Table of means and standard deviations

We have data from the Second National Health and Nutrition Examination Survey (NHANES II) (McDowell et al. 1981), and we want to create a table reporting the mean and standard deviation of systolic blood pressure, cholesterol, and triglycerides for males and females. Additionally, we want to display the p-value for the mean-comparison test.

```
. use https://www.stata-press.com/data/r19/nhanes2l
(Second National Health and Nutrition Examination Survey)
```

We can use the `ttest` command to perform a mean-comparison test and then create a table with its stored results. The `table` command allows us to issue multiple commands and create a table with their results in a single step. For example, below we create a table with only two variables of interest to prevent the table from wrapping. We specify each `ttest` command in the `command()` option, and we place results from each command on a separate column. While `ttest` reports multiple statistics, we will display only the means (`mu_1` and `mu_2`) and p-values.

```
. table (result[mu_1 mu_2 p]) (command),
> command(ttest bpsystol, by(sex))
> command(ttest tcresult, by(sex))
```

```
--------------------------------------------------------------------------
                        |  ttest bpsystol, by(sex)   ttest tcresult, by(sex)
------------------------+-------------------------------------------------
x1 mean for population 1|                 132.8877                  213.1731
x2 mean for population 2|                 129.0679                  221.7353
Two-sided p-value       |                 8.03e-17                  1.10e-18
--------------------------------------------------------------------------
```

Notice that the commands are used to label the output for each variable. We will now create our table with all three variables, but we will use the `quietly` prefix to suppress our output:

```
. quietly: table () (command),
> command(ttest bpsystol, by(sex))
> command(ttest tcresult, by(sex))
> command(ttest tgresult, by(sex))
```

Below, we use `collect label levels` to label the output from each command with the variable label. Then, we use `collect style header` to suppress the labels for the results; we will modify them shortly.

```
. collect label levels command  1 "Systolic BP"
> 2 "Cholesterol (mg/dL)" 3 "Triglycerides (mg/dL)", modify
. collect style header result, level(hide)
```

Next, we will format all results to two decimal places and the p-values to three. Then, we will lay out our table with the results on the columns and the variables on the rows. Each command corresponds to a different variable, so we specify `command` as the row identifier. We are interested only in the means, standard deviations, and p-values, so we list the levels of `result` that we want in our table.

```
. collect style cell result, nformat(%6.2f)
. collect style cell result[p], nformat(%7.3f)
. collect layout (command) (result[mu_1 sd_1 mu_2 sd_2 p])
Collection: Table
      Rows: command
   Columns: result[mu_1 sd_1 mu_2 sd_2 p]
   Table 1: 3 x 5

-----------------------------------------------------------------
Systolic BP            |  132.89     20.99   129.07    25.13   0.000
Cholesterol (mg/dL)    |  213.17     45.98   221.74    51.95   0.000
Triglycerides (mg/dL)  |  154.63    112.30   133.85    77.58   0.000
-----------------------------------------------------------------
```

We would like to display the standard deviations right next to the means, using a plus–minus sign as the delimiter. Below, we combine the means and standard deviations for males (`mu_1` and `sd_1`) into one result and the means and standard deviations for females (`mu_2` and `sd_2`) into another.

The means and standard deviations have a numeric format of `%6.2f`, which means a total output width of 6 with only 2 digits displayed after the decimal. However, you will notice that all the standard deviations for females in the above fourth column have a total width of five: two digits before the decimal, two digits after, and the decimal point. This means we will have an extra space between the plus–minus sign and those standard deviations. Therefore, we use the `trim` option to remove those extra spaces.

Then, we label the results as `Males` and `Females` and modify the label for the p-values. Now that we have customized labels, we use `collect style header` to display the labels for our results.

```
. collect composite define msd1 = mu_1 sd_1, delimiter(±) trim
. collect composite define msd2 = mu_2 sd_2, delimiter(±) trim
. collect label levels result msd1 "Males" msd2 "Females" p "p-value", modify
. collect style header result, level(label)
```

Finally, we can lay out our table using our composite results, `msd1` and `msd2`:

```
. collect layout (command) (result[msd1 msd2 p])
Collection: Table
      Rows: command
   Columns: result[msd1 msd2 p]
   Table 1: 3 x 3

--------------------------------------------------------------
                       |           Males        Females   p-value
--------------------------------------------------------------
Systolic BP            |    132.89±20.99   129.07±25.13     0.000
Cholesterol (mg/dL)    |    213.17±45.98   221.74±51.95     0.000
Triglycerides (mg/dL)  |   154.63±112.30   133.85±77.58     0.000
--------------------------------------------------------------
```

We could customize this table further by, for example, modifying how p-values are displayed and removing some borders, but our focus here is solely on displaying these combined results.

Example 2: Table of means, medians, standard deviations, and confidence intervals

Suppose we want to create a table similar to the one above, but instead of p-values, we now want to include medians and confidence intervals for the means. We can obtain these statistics from the `summarize` and `ci means` commands. First, we clear the collection information in memory:

```
. collect clear
```

To create our table, we loop over our variables of interest and the values of sex (1 and 2). We will use `collect get` to collect the stored results of interest from each command.

First, we obtain our means and standard deviations from `ttest`. Recall that `ttest` stores the means and standard deviations for the groups separately, in `mu_1` and `sd_1` and `mu_2` and `sd_2`. We wish to store all the means in one result, `mymean`, and all the standard deviations in another result, `mysd`. To identify whether they belong to males or females, we will tag each result with the variable and level of sex.

Next, we will collect the upper (`r(ub)`) and lower (`r(lb)`) bounds of the confidence intervals from `ci means` and the median from `summarize`. Note that we also tag these results with the variable and level of sex.

```
. foreach x of varlist bpsystol tcresult tgresult {
  2.     ttest ‘x’, by(sex)
  3.     collect get mymean=r(mu_1) mysd=r(sd_1), tags(var[‘x’] sex[1])
  4.     collect get mymean=r(mu_2) mysd=r(sd_2), tags(var[‘x’] sex[2])
  5.
.     forvalues i = 1/2 {
  6.         ci means ‘x’ if sex==‘i’
  7.         collect get r(lb) r(ub), tags(var[‘x’] sex[‘i’])
  8.
.         summarize ‘x’ if sex==‘i’, detail
  9.         collect get r(p50), tags(var[‘x’] sex[‘i’])
 10.     }
 11. }
  (output omitted)
```

We have now collected all of our results, and we can begin building our table. First, let’s create a composite result from the upper and lower bounds of the confidence intervals. This result, `meanci`, will use a dash as the delimiter:

```
. collect composite define meanci = lb ub, delimiter(-) trim
```

We will format all results to two decimal places and center them. Then, we will place the confidence intervals in brackets:

```
. collect style cell result, nformat(%6.2f) halign(center)
. collect style cell result[meanci], sformat("[%s]")
```

Next, we will join the means and standard deviations, as we did with the previous example. Before we combine the medians with the composite result, `meansd`, we enclose them in parentheses. We then create a new composite result called `msdmed`.

```
. collect composite define meansd = mymean mysd, delimiter(±) trim
. collect style cell result[p50], sformat("(%s)")
. collect composite define msdmed = meansd p50
```

Before we lay out our table, we will hide the labels for the results with `collect style header`. Then, we will use `collect layout` to lay out our table with the mean, standard deviation, and median for each variable on one row and the confidence interval on the following row. The columns will be defined by the levels of `sex`.

```
. collect style header result, level(hide)

. collect layout (var#result[msdmed meanci]) (sex)

Collection: default
      Rows: var#result[msdmed meanci]
   Columns: sex
   Table 1: 6 x 2

-------------------------------------------------------------------------
                             |                 Male               Female
-----------------------------+-------------------------------------------
Systolic blood pressure      |   132.89±20.99 (130.00) 129.07±25.13 (124.00)
                             |      [132.30-133.47]       [128.40-129.74]
Serum cholesterol (mg/dL)    |   213.17±45.98 (209.00) 221.74±51.95 (217.00)
                             |      [211.89-214.46]       [220.35-223.12]
Serum triglycerides (mg/dL)  |  154.63±112.30 (128.00) 133.85±77.58 (115.00)
                             |      [150.17-159.09]       [130.87-136.83]
-------------------------------------------------------------------------
```

Our table looks great; we just need to modify the headers. Below, we center the column headers, add a descriptive title, and then preview our table:

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

. collect title "Mean, SD, median, and confidence interval values for health me
> asures"

. collect preview

Mean, SD, median, and confidence interval values for health measures
-------------------------------------------------------------------------
                             |           Male                  Female
-----------------------------+-------------------------------------------
Systolic blood pressure      |   132.89±20.99 (130.00) 129.07±25.13 (124.00)
                             |      [132.30-133.47]       [128.40-129.74]
Serum cholesterol (mg/dL)    |   213.17±45.98 (209.00) 221.74±51.95 (217.00)
                             |      [211.89-214.46]       [220.35-223.12]
Serum triglycerides (mg/dL)  |  154.63±112.30 (128.00) 133.85±77.58 (115.00)
                             |      [150.17-159.09]       [130.87-136.83]
-------------------------------------------------------------------------
```

Example 3: Table of regression results

Creating composite results can also be useful when creating a table of regression results. For example, we wish to create a table with the coefficients, confidence intervals, and *p*-values from a linear regression model. Below, we create a new collection and collect the results from our model:

```
. collect create regress
(current collection is regress)

. quietly: collect: regress bpsystol i.agegrp bmi i.sex
```

Before laying out our table, we format the results to two decimal places and center the coefficients and p-values. We also place the confidence intervals in brackets and use a comma to separate the upper and lower bounds.

```
. collect style cell result[_r_b _r_p], nformat(%5.2f) halign(center)

. collect style cell result[_r_ci], nformat(%9.2f) sformat("[%s]")
> cidelimiter(,)
```

Now that we have formatted our confidence intervals and coefficients, we can combine them into a single result. We create a composite result called `coefci` and trim the extra spaces resulting from the numeric format. Then, we lay out our table with our composite result and the p-values.

```
. collect composite define coefci = _r_b _r_ci, trim

. collect layout (colname) (result[coefci _r_p])

Collection: regress
      Rows: colname
   Columns: result[coefci _r_p]
   Table 1: 10 x 2

----------------------------------------------------------
                      |   Coefficient [95% CI] p-value
----------------------+-----------------------------------
20-29                 |                0.00
30-39                 |    0.87 [-0.37,     2.10]   0.17
40-49                 |     6.69 [5.36,     8.03]   0.00
50-59                 |   14.75 [13.42,    16.09]   0.00
60-69                 |   20.99 [19.91,    22.06]   0.00
70+                   |   28.00 [26.55,    29.45]   0.00
Body mass index (BMI) |     1.36 [1.28,     1.44]   0.00
Male                  |                0.00
Female                |   -4.07 [-4.82,    -3.32]   0.00
Intercept             |   87.08 [85.03,    89.12]   0.00
----------------------------------------------------------
```

Notice that we have a lot of extra space preceding the upper bounds of the confidence intervals. We set the output width to 9 (`%9.2f`), but our bounds require at most a width of 5. We chose this format for `_r_ci` to illustrate that the `trim` option does not work with this result or with `_r_cri` if you are working with credible intervals. The reason is that the `collect` system has already composed these from upper and lower bounds; in a sense, these are composite results. To prevent having this extra space, we have two options. One option is to specify a smaller width, perhaps `%5.2f`, for our intervals. When creating your own tables, you may find that several results have a numeric format that is too large. You can use `collect query cell` to query the appearance styles for your results. The other option is to build your own composite result with the confidence intervals to which you can then apply the `trim` option. This is the method we will use. Below, we create a new composite result with the lower (`_r_lb`) and upper (`_r_ub`) bounds and then enclose these results in brackets.

```
. collect composite define myci = _r_lb _r_ub, delimiter(,)

. collect style cell result[myci], nformat(%9.2f) sformat("[%s]")
```

Now, we can create our composite result with the coefficients and composite result; we call it `coefci2`. We `trim` these results and specify the `override` option; this allows us to apply this trim property to the composite result, `myci`. Then, we label `coefci2` by referring to `__LEVEL__`, which is the confidence level that is recorded when results are consumed; in our case, the level is 95, but our notation avoids us having to check whether that is the case. If you issue `collect label list result`, you will see that this notation is used for `_r_lb`, `_r_ub`, and `_r_ci`. After labeling our new result, we lay out our table with the new composite result:

```
. collect composite define coefci2 = _r_b myci, trim override
. collect label levels result coefci2 "Coef. [__LEVEL__% CI]"
. collect layout (colname) (result[coefci2 _r_p])
Collection: regress
      Rows: colname
   Columns: result[coefci2 _r_p]
   Table 1: 10 x 2

---------------------------------------------------
                      |     Coef. [95% CI] p-value
----------------------+----------------------------
20-29                 |               0.00
30-39                 |   0.87 [-0.37,2.10]   0.17
40-49                 |    6.69 [5.36,8.03]   0.00
50-59                 | 14.75 [13.42,16.09]   0.00
60-69                 | 20.99 [19.91,22.06]   0.00
70+                   | 28.00 [26.55,29.45]   0.00
Body mass index (BMI) |    1.36 [1.28,1.44]   0.00
Male                  |               0.00
Female                | -4.07 [-4.82,-3.32]   0.00
Intercept             | 87.08 [85.03,89.12]   0.00
---------------------------------------------------
```

Here we can see that the intervals were in fact trimmed of the extra spaces.

Combining our coefficients and confidence intervals into a single result is particularly useful when displaying results from multiple models. For example, below we refit our model but add indicator variables for `race`. Then, we lay out our table once more:

```
. quietly: collect: regress bpsystol i.agegrp bmi i.sex i.race
. collect layout (colname) (cmdset#result[coefci2 _r_p])
Collection: regress
      Rows: colname
   Columns: cmdset#result[coefci2 _r_p]
   Table 1: 13 x 4

---------------------------------------------------------------------------------
                      |                  1       1                   2       2
                      |     Coef. [95% CI] p-value      Coef. [95% CI] p-value
----------------------+----------------------------------------------------------
20-29                 |               0.00                        0.00
30-39                 |   0.87 [-0.37,2.10]   0.17   0.93 [-0.30,2.16]   0.14
40-49                 |    6.69 [5.36,8.03]   0.00    6.80 [5.46,8.13]   0.00
50-59                 | 14.75 [13.42,16.09]   0.00 14.84 [13.51,16.18]   0.00
60-69                 | 20.99 [19.91,22.06]   0.00 21.12 [20.04,22.20]   0.00
70+                   | 28.00 [26.55,29.45]   0.00 28.12 [26.67,29.58]   0.00
Body mass index (BMI) |    1.36 [1.28,1.44]   0.00    1.35 [1.27,1.42]   0.00
Male                  |               0.00                        0.00
Female                | -4.07 [-4.82,-3.32]   0.00 -4.08 [-4.82,-3.33]   0.00
White                 |                                           0.00
Black                 |                                 2.61 [1.39,3.84]   0.00
Other                 |                                2.07 [-0.65,4.78]   0.14
Intercept             | 87.08 [85.03,89.12]   0.00 87.01 [84.96,89.06]   0.00
---------------------------------------------------------------------------------
```

Note that we did not have to create another composite result for the coefficients and confidence intervals for the second model we fit. The composite result (`coefci2`) will be built from all existing values of `_r_b`, `_r_lb`, and `_r_ub`.

We can give the table a neater look with just a couple of modifications. First, we suppress the base levels from the output. Then, we specify that duplicate headers should be displayed only once and centered. Finally, we label the group of results from each set of commands as `Model 1` and `Model 2` and preview our table:

```
. collect style showbase off
. collect style column, dups(center)
. collect label levels cmdset 1 "Model 1" 2 "Model 2"
. collect preview
```

```
-------------------------------------------------------------------------------
                      |            Model 1                      Model 2
                      |  Coef.  [95% CI]  p-value       Coef.  [95% CI]  p-value
----------------------+--------------------------------------------------------
30-39                 |   0.87 [-0.37,2.10]   0.17     0.93 [-0.30,2.16]    0.14
40-49                 |    6.69 [5.36,8.03]   0.00      6.80 [5.46,8.13]    0.00
50-59                 | 14.75 [13.42,16.09]   0.00  14.84 [13.51,16.18]    0.00
60-69                 | 20.99 [19.91,22.06]   0.00  21.12 [20.04,22.20]    0.00
70+                   | 28.00 [26.55,29.45]   0.00  28.12 [26.67,29.58]    0.00
Body mass index (BMI) |    1.36 [1.28,1.44]   0.00      1.35 [1.27,1.42]    0.00
Female                | -4.07 [-4.82,-3.32]   0.00  -4.08 [-4.82,-3.33]    0.00
Black                 |                                 2.61 [1.39,3.84]    0.00
Other                 |                                2.07 [-0.65,4.78]    0.14
Intercept             | 87.08 [85.03,89.12]   0.00  87.01 [84.96,89.06]    0.00
-------------------------------------------------------------------------------
```

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 label** — Manage custom labels in a collection

[TABLES] **collect levelsof** — List levels of a dimension

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

[TABLES] **collect query** — Query collection style properties