Title

> **table hypothesis tests —** Table of hypothesis tests

Description	Quick start	Menu	Syntax
Options	Remarks and examples	Stored results	Reference
Also see			

Description

In this entry, we discuss how to use `table` to create tables with results of hypothesis tests.

Quick start

Table with pairwise correlations stored in matrix `r(C)`

```
table (rowname) (colname), command(r(C): pwcorr v1 v2 v3)
```

Table with all the numeric scalars returned by `ttest`; rows correspond to the different results

```
table (result) (command), command(ttest v1, by(catvar))
```

Table with means and two-sided p-values; columns correspond to the different results

```
table (command) (result), ///
    command(r(mu_1) r(mu_2) r(p): ttest v1, by(catvar))
```

Same as above, but with statistics for `v1` and `v2`

```
table (command) (result), ///
    command(r(mu_1) r(mu_2) r(p): ttest v1, by(catvar)) ///
    command(r(mu_1) r(mu_2) r(p): ttest v2, by(catvar))
```

Menu

Statistics > Summaries, tables, and tests > Tables of frequencies, summaries, and command results

Syntax

> table (⌈*rowspec*⌉) (⌈*colspec*⌉) ⌈(*tabspec*)⌉ ⌈*if*⌉ ⌈*in*⌉ ⌈*weight*⌉,
>
> command(*cmdspec*) ⌈command(*cmdspec*) ...⌉ ⌈*options*⌉

rowspec, *colspec*, and *tabspec* may be empty or may include variable names or any of the following
keywords:

keyword	Description
result	requested statistics
stars	stars denoting statistical significance
command	index option command()
colname	column names for matrix statistics
rowname	row names for matrix statistics

options	Description
Commands	
command(*cmdspec*)	collect results from the specified Stata command
Formats	
nformat(%*fmt* ⌈*results*⌉⌈, basestyle⌉)	specify numeric format
sformat(*sfmt* ⌈*results*⌉)	specify string format
Stars	
stars(*starspec*)	add stars to denote statistical significance
Options	
missing	treat numeric missing values like other values
name(*cname*)	collect results into a collection named *cname*
append	append results to an existing collection
replace	replace results of an existing collection
label(*filename*)	specify the collection labels
style(*filename* ⌈, override⌉)	specify the collection style
noisily	display output from each command

fweights, aweights, iweights, and pweights are allowed; see [U] **11.1.6 weight**.

strL variables are not allowed; see [U] **12.4.8 strL**.

noisily does not appear in the dialog box.

Options

> Commands

command(*cmdspec*) specifies the Stata commands from which to collect results. command() may be repeated to collect results from multiple commands.

    cmdspec is $\big[$*explist*:$\big]$ *command* $\big[$*arguments*$\big]$ $\big[$, *cmdoptions*$\big]$

        explist specifies which results to collect and report in the table. *explist* may include *result identifiers* and *named expressions*.

        result identifiers are results stored in r() and e() by the *command*. For instance, *result identifiers* could be r(mean), r(C), or e(chi2). After estimation commands, *result identifiers* also include the following:

Identifier	Result
_r_b	coefficients or transformed coefficients reported by *command*
_r_se	standard errors of _r_b
_r_z	test statistics for _r_b
_r_z_abs	absolute value of _r_z
_r_p	*p*-values for _r_b
_r_lb	lower bounds of confidence intervals for _r_b
_r_ub	upper bounds of confidence intervals for _r_b
_r_ci	confidence intervals for _r_b
_r_crlb	lower bounds of credible intervals for _r_b
_r_crub	upper bounds of credible intervals for _r_b
_r_cri	credible intervals for _r_b
_r_df	degrees of freedom for _r_b

        named expressions are specified as *name* = *exp*, where *name* may be any valid Stata name and *exp* is an expression, typically an expression that involves one or more *result identifiers*. An example of a named expression is sd = sqrt(r(variance)).

    For r-class commands, the default is to include all numeric scalars posted to r() in the table results. For e-class commands, the default is to include _r_b in the table results.

    command is any command that follows standard Stata syntax.

    arguments may be anything so long as they do not include an if clause, in range, or weight specification.

    Any if or in qualifier and weights should be specified directly with table, not within the command() option. Weights are passed to *command* only if they are specified.

    cmdoptions may be anything supported by *command*.

> Formats

nformat(%*fmt* $\big[$*results*$\big]$ $\big[$, basestyle$\big]$) changes the numeric format, such as the number of decimal places, for specified results. If *results* are not specified, the numeric format is changed for all results.

    results may be any name in the e() or r() results produced by commands specified in option command().

This option is repeatable, and when multiple formats apply to one result, the rightmost specification is applied.

This option does not affect the format of numeric layout variables (*rowspec*, *colspec*, and *tabspec*). The default format of these variables is taken from the dataset.

basestyle indicates that the format be applied to results that do not already have their own format instead of overriding the format for all results.

sformat(*sfmt* [*results*]) changes the string format for specified results. You can, for instance, add symbols or text to the values reported in the table by modifying the string format.

sfmt may contain a mix of text and %s. Here %s refers to the numeric value that is formatted as specified using nformat(). The text will be placed around the numeric values in your table as it is placed around %s in this option. For instance, to place parentheses around the percent statistics, you can specify sformat("(%s)" percent).

results may be any name in the e() or r() results produced by commands specified in option command().

Two text characters must be specified using a special character sequence if you want them to be displayed in your table. To include %, type %%. To include \, type \\. For instance, to place a percent sign following percent statistics, you can specify sformat("%s%%" percent).

This option is repeatable, and when multiple formats apply to one result, the rightmost specification is applied.

> Stars

stars(*starspec*) specifies that stars representing statistical significance be included in the table. *starspec* identifies the result whose values determine significance, which characters should represent each significance level, and where these characters should be displayed in the table. *starspec* is

starres [*#1* "*label1*" [*#2* "*label2*" [*#3* "*label3*" [*#4* "*label4*" [*#5* "*label5*"]]]]]
[, attach(*attachres*) result dimension *starsnoteopts*]

starres is the name of the result whose values determine which characters, typically which number of stars, are to be displayed.

label1 specifies the characters to be displayed when *starres* < *#1*.

label2 specifies the characters to be displayed when *starres* < *#2*.

label3 specifies the characters to be displayed when *starres* < *#3*.

label4 specifies the characters to be displayed when *starres* < *#4*.

label5 specifies the characters to be displayed when *starres* < *#5*.

attach(*attachres*) specifies the name of the result to which the characters defined by *label1*, ..., *label5* are to be attached. If attach() is not specified, a new result named stars is created and is automatically added to the table.

result and dimension control how collect stars adds items when labeling significant results. These options are mutually exclusive.

result specifies the default behavior, and this option is necessary only if the following dimension behavior is in effect and you want to change back to the result behavior.

dimension specifies that dimension stars be added to the collection. Items will be tagged with stars[value], and the labels will be tagged with stars[label]. Use this option for layouts where results are to be stacked within columns, and use new dimension stars in the column specification of the layout.

starsnoteopts control the display and composition of the stars note.

noshownote and shownote control whether to display the stars note.

increasing and decreasing control the order of p-values in the stars note.

pvname(*string*) specifies a name for the p-value in the stars note. The default is pvname(p).

delimiter(*string*) specifies the delimiter between labels in the stars note. The default is delimiter(",").

nformat(% *fmt*) specifies the numeric format for the cutoff values in the stars note. The default is nformat(%9.0g).

prefix(*string*) specifies the prefix for the stars note. The prefix is empty by default.

suffix(*string*) specifies the suffix for the stars note. The suffix is empty by default.

For example, stars(_r_p 0.01 "***" 0.05 "**" 0.1 "*", attach(_r_b)) could be added to a table of regression results to specify that stars be defined based on the p-values in _r_p and be attached to the reported coefficients (_r_b).

> ┌─── Options ───

missing specifies that numeric missing values of any variables specified in *rowspec*, *colspec*, or *tabspec* be treated as valid categories. By default, observations with a numeric missing value in any of these variables are omitted.

name(*cname*) specifies that a collection named *cname* be associated with the collected statistics and results. The default is name(Table).

append specifies that table append its collection information into the collection named in name().

replace permits table to overwrite an existing collection. This option is implied for name(Table) when append is not specified.

label(*filename*) specifies the *filename* containing the collection labels to use for your table. Labels in *filename* will be loaded for the table, and any labels not specified in *filename* will be taken from the labels defined in c(collect_label). The default is to use only the collection labels set in c(collect_label); see [TABLES] **set collect_label**.

style(*filename* [, override]) specifies the *filename* containing the collection styles to use for your table. The default collection styles will be discarded, and only the collection styles in *filename* will be applied.

If you prefer the default collection styles but also want to apply any styles in *filename*, specify override. If there are conflicts between the default collection styles and those in *filename*, the ones in *filename* will take precedence.

The default is to use only the collection styles set in c(table_style); see [TABLES] **set table_style**.

The following option is available with table but is not shown in the dialog box:

noisily specifies that output from the commands specified in command() options be displayed. By default, output from commands is suppressed.

Remarks and examples

Remarks are presented under the following headings:

Introduction

The `table` command can be used to create tables with results of hypothesis tests. For example, you can create a table with results from a mean-comparison test, a test of proportions, or a test of normality.

`table` does not perform hypothesis tests directly. Rather, `table` will run any Stata command that you include in its `command()` option and place results from that command into the table. You determine which results you would like to see in the table. You can select any of the results stored by the command.

Creating tables from scalars

We have data from the Second National Health and Nutrition Examination Survey (NHANES II) (McDowell et al. 1981). The data contain some demographic information, such as the age, sex, and race of participants. The data also contain some measures of health, including whether the individual has high blood pressure (`highbp`), has `diabetes`, or has had a heart attack previously (`heartatk`).

Suppose we want to examine the proportion of males and females that have high blood pressure, that have diabetes, and that have had a heart attack previously. With `prtest`, we can test whether the proportions are equal between males and females. For example, let's perform a test of proportions for `diabetes`:

```
. use https://www.stata-press.com/data/r18/nhanes2l
(Second National Health and Nutrition Examination Survey)
. prtest diabetes, by(sex)
Two-sample test of proportions                 Male: Number of obs =     4915
                                             Female: Number of obs =     5434
```

Group	Mean	Std. err.	z	P>|z|	[95% conf. interval]	
Male	.0441506	.0029302			.0384074	.0498937
Female	.0518955	.0030091			.0459978	.0577932
diff	-.0077449	.0042001			-.0159769	.0004871
	under H0:	.0042169	-1.84	0.066		

```
        diff = prop(Male) - prop(Female)                          z =  -1.8366
    H0: diff = 0

    Ha: diff < 0              Ha: diff != 0              Ha: diff > 0
 Pr(Z < z) = 0.0331      Pr(|Z| > |z|) = 0.0663      Pr(Z > z) = 0.9669
```

We would like to create a table that includes the proportion of men who have diabetes, the proportion of women who have diabetes, the difference in these proportions, and the p-value for a two-sided test. First, we need to determine how to refer to these statistics.

```
. return list

scalars:
                  r(N1) =  4915
                  r(N2) =  5434
                  r(P1) =  .0441505595116989
                  r(P2) =  .0518954729481045
              r(P_diff) = -.0077449134364056
                 r(se1) =  .0029302258134317
                 r(se2) =  .003009075122777
            r(se_diff0) =  .0042169418903878
             r(se_diff) =  .0042000900481081
                 r(lb1) =  .0384074224508032
                 r(ub1) =  .0498936965725946
                 r(lb2) =  .0459977940806861
                 r(ub2) =  .0577931518155229
             r(lb_diff) = -.0159769386625226
             r(ub_diff) =  .0004871117897114
                   r(z) = -1.836618487454034
                 r(p_l) =  .0331331180748532
                   r(p) =  .0662662361497065
                 r(p_u) =  .9668668819251468
               r(level) =  95
```

The statistics we want to see are stored as `r(P1)`, `r(P2)`, `r(P_diff)`, and `r(p)`. We can specify this in the `command()` option by typing

```
. table ..., command(r(P1) r(P2) r(P3) r(p): prtest diabetes, by(sex))
```

This will get the results we want into our table. Furthermore, because we know what these values represent, we can give them names that will appear in the table headers. We can, for instance, type

```
. table ..., command(Males=r(P1) Females=r(P2) Difference=r(P3) ///
    r(p): prtest diabetes, by(sex))
```

We can specify similar `command()` options for `heartatk` and `highbp` as well.

In addition, we need to specify how our results will be laid out in the table. Below, we type `command` in the first set of parentheses so that the rows correspond to the different commands. We type `result` in the second set of parentheses to specify that statistics appear in the columns.

Finally, we add two options to customize the results. We specify a numeric format so that the statistics be displayed only with three digits after the decimal. We also choose the predefined style `table-right` so that our row headers will be right-aligned. See [TABLES] **Predefined styles** for information on this and other styles.

```
. table (command) (result),
> command(Males=r(P1) Females=r(P2) Difference=r(P_diff) r(p):
> prtest diabetes, by(sex))
> command(Males=r(P1) Females=r(P2) Difference=r(P_diff) r(p):
> prtest heartatk, by(sex))
> command(Males=r(P1) Females=r(P2) Difference=r(P_diff) r(p):
> prtest highbp, by(sex))
> nformat(%5.3f) style(table-right)
```

	Males	Females	Difference	Two-sided p-value
prtest diabetes, by(sex)	0.044	0.052	-0.008	0.066
prtest heartatk, by(sex)	0.065	0.029	0.036	0.000
prtest highbp, by(sex)	0.469	0.381	0.088	0.000

Our table now includes all the statistics we want. Yet we might want to make some modifications. Table customization can go beyond the predefined styles and options available to you in the `table` command. `table` creates a collection of results that can be used in combination with the `collect` suite of commands to produce highly customized tables and to export those tables to presentation-ready formats, such as HTML, Word, LATEX, PDF, Excel, and more.

For this table, we want to modify the labels in our row headers. Instead of showing the full command that was run, row headers will identify the variable we are testing. In addition, we will modify the label for our p-value. We want to use the label `p-value`. Because this is not a valid Stata name, we could not specify it in the `table` command as we did with `Males`. However, we can use `collect label levels` to modify the label on our p-values.

After applying the label updates, we use `collect preview` to see our updated table.

```
. collect label levels command 1 "Diabetes" 2 "Heart attack" 3 "High BP", modify

. collect label levels result p "p-value", modify

. collect preview
```

	Males	Females	Difference	p-value
Diabetes	0.044	0.052	−0.008	0.066
Heart attack	0.065	0.029	0.036	0.000
High BP	0.469	0.381	0.088	0.000

Creating tables from matrices

You may find that the results you want to include in your table are stored in a matrix; these results can also be easily included in a table.

To demonstrate, we create a table with p-values for tests of normality for `height`, `weight`, and diastolic blood pressure (`bpdiast`). The command `sktest` performs tests based on skewness, kurtosis, and a combined test statistic.

```
. sktest height weight bpdiast
```

Skewness and kurtosis tests for normality

Variable	Obs	Pr(skewness)	Pr(kurtosis)	Adj chi2(2)	Prob>chi2
height	10,351	0.0000	0.0000	147.47	0.0000
weight	10,351	0.0000	0.0000	801.40	0.0000
bpdiast	10,351	0.0000	0.0000	362.54	0.0000

(Column group header: ─── Joint test ─── spanning "Adj chi2(2)" and "Prob>chi2")

Let's look at the returned results.

```
. return list

scalars:
                 r(N) =  10351
            r(p_skew) =  1.58706287446e-72
            r(p_kurt) =  6.22330331716e-26
              r(chi2) =  362.5385838320567
           r(p_chi2) =  1.88689086684e-79

matrices:
             r(table) :  3 x 5
```

The statistics we want plus a few others are stored in `r(table)`.

Now, let's place these values in a table. We specify that our table be arranged with the row names (rowname) of the matrix defining the rows of the table. Similarly, the column names (colname) of the matrix define the columns of the table. Then, we specify that we want to collect the results from the matrix r(table) from the sktest command.

```
. table (rowname) (colname),
> command(r(table): sktest height weight bpdiast)
```

	N	p_skew	p_kurt	chi2	p_chi2
Height (cm)	10351	.0000179	1.87e-35	147.4712	9.48e-33
Weight (kg)	10351	1.6e-166	1.63e-49	801.3958	9.5e-175
Diastolic blood pressure	10351	1.59e-72	6.22e-26	362.5386	1.89e-79

Because the row names in r(table) corresponded to variables, our table automatically put the variable labels in the row headers. However, the column headers are not nicely labeled.

We can create better labels and modify our table in many other ways. table creates a collection of results that can be used in combination with the [collect](#) suite of commands to further customize tables.

To clean up our table, let's use collect label levels to modify the labels for the p-values for the skewness, kurtosis, and joint tests; these are the statistics we will include in our table below. To use collect label levels, we need to know just a little about the collect system. In collections, values are organized according to dimensions and levels within those dimensions. In fact, we use these dimensions in table. The keywords that we can use to define our rows and columns are dimensions. Here colname is our dimension that defines the columns, and its levels are N, p_skew, ... To modify labels, we need to tell collect label levels which dimension (colname) we would like to change and then specify labels for levels of that dimension.

We specify the dimension and then the label for each level:

```
. collect label levels colname p_skew "Skewness p-value"
> p_kurt "Kurtosis p-value" p_chi2 "Joint p-value", modify
```

To learn more about modifying labels, see [TABLES] **collect label**.

Let's also change the numeric format of our p-values. With [collect style cell](#), we can modify all cells in the table, all cells in a particular dimension, or particular cells of a particular dimension. Below, we specify the numeric formatting for only three levels of colname.

```
. collect style cell colname[p_skew p_kurt p_chi2], nformat(%7.3f)
```

Finally, we want to show only the three p-values in our table. We can use [collect layout](#) to specify the statistics we want to include in our final table.

```
. collect layout (rowname) (colname[p_skew p_kurt p_chi2])
Collection: Table
      Rows: rowname
   Columns: colname[p_skew p_kurt p_chi2]
   Table 1: 3 x 3
```

	Skewness p-value	Kurtosis p-value	Joint p-value
Height (cm)	0.000	0.000	0.000
Weight (kg)	0.000	0.000	0.000
Diastolic blood pressure	0.000	0.000	0.000

Notably, all *p*-values for all tests are very small, so this is not a particularly exciting table. However, our table customizations made it easy to quickly see the results of tests of normality for all our variables.

Stored results

table stores the following in `r()`:

Scalars
    `r(N)`   number of observations

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. *Vital and Health Statistics* 1(15): 1–144.

Also see

[R] **table** — Table of frequencies, summaries, and command results

[R] **table intro** — Introduction to tables of frequencies, summaries, and command results

[R] **table regression** — Table of regression results

[TABLES] **Intro** — Introduction