table multiway — Multiway tables

Description

In this entry, we discuss how to use the `table` command to create tables displaying frequencies, percentages, and proportions across levels of three or more variables. Additionally, we demonstrate how a single `table` command can create multiple tables corresponding to levels of variables or different statistics.

Quick start

Table of frequencies with rows defined by the levels of `a1` and columns defined by the categories of `a2` and `a3`

```
table (a1) (a2 a3)
```

Same as above, but with rows defined by levels of `a1` and `a2` and columns defined by levels of `a3`

```
table (a1 a2) (a3)
```

Separate tables for each level `a3` with the rows of each table defined by the levels of `a1` and the columns defined by levels of `a2`

```
table (a1) (a2) (a3)
```

Report percentages of observations in each cell rather than frequencies

```
table (a1) (a2 a3), statistic(percent)
```

Report both frequencies and percentages

```
table (a1) (a2 a3), statistic(percent) statistic(frequency)
```

Report the percentages across levels of `a3`

```
table (a1) (a2 a3), statistic(percent, across(a3))
```

Menu

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

Syntax

Table with rows defined by multiple variables

`table` *rowvars colvar* [*if*] [*in*] [*weight*] [, *options*]

Table with columns defined by multiple variables

`table` *rowvar* `(`*colvars*`)` [*if*] [*in*] [*weight*] [, *options*]

Table with rows and columns defined by multiple variables

`table (`*rowvars*`) (`*colvars*`)` [*if*] [*in*] [*weight*] [, *options*]

Multiple multiway tables

`table (`*rowvars*`) (`*colvars*`) (`*tabvars*`)` [*if*] [*in*] [*weight*] [, *options*]

Customized multiway tables

`table (`*rowspec*`) (`*colspec*`)` [`(`*tabspec*`)`] [*if*] [*in*] [*weight*] [, *options*]

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

keyword	Description
`result`	requested statistics
`across`	index `across()` specifications

options	Description
Main	
`totals(`*totals*`)`	report only the specified totals
`nototals`	suppress the marginal totals
Statistics	
`statistic(`*stat*`[ , `*statopts*`])`	statistic to be reported; default is `statistic(frequency)` when no weights are specified and `statistic(sumw)` otherwise
Formats	
`nformat(`*%fmt* `[`*results*`][ , basestyle])`	specify numeric format
`sformat(`*sfmt* `[`*results*`])`	specify string format
Title	
`title(`*string*`)`	add table title
`titlestyles(`*text_styles*`)`	change table title styles
Notes	
`note(`*string*`)`	add table note
`notestyles(`*text_styles*`)`	change table note styles
Export	
`export(`*filename*`.`*suffix*`[ , `*export_opts*`])`	export table
Options	
`listwise`	use listwise deletion to handle missing values
`missing`	treat numeric missing values like other values
`showcounts`	show sample size for all variables in `statistic()` option
`zerocounts`	report 0 for empty cell counts
`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
`markvar(`*newvar*`)`	create *newvar* to identify observations used to compute the statistics

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

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

`markvar()` does not appear in the dialog box.

text_styles	Description
`font(`[*fontfamily*][`,` *font_opts*]`)`	specify font style
`smcl(`*smcl*`)`	specify formatting for SMCL files
`latex(`*latex*`)`	specify LaTeX macro
`shading(`*sspec*`)`	set background color, foreground color, and fill pattern

font_opts	Description
`size(`#[*unit*]`)`	specify font size
`color(`*color*`)`	specify font color
`variant(`*variant*`)`	specify font variant and capitalization
[`no`]`bold`	specify whether to format text as bold
[`no`]`italic`	specify whether to format text as italic
[`no`]`strikeout`	specify whether to strike out text
[`no`]`underline`	specify whether to underline text

suffix	*fileformat*	Output format
`docx`	`as(docx)`	Microsoft Word
`html`	`as(html)`	HTML 5 with CSS
`pdf`	`as(pdf)`	PDF
`xlsx`	`as(xlsx)`	Microsoft Excel 2007/2010 or newer
`xls`	`as(xls)`	Microsoft Excel 1997/2003
`tex`	`as(tex)`	LaTeX
`smcl`	`as(smcl)`	SMCL
`txt`	`as(txt)`	plain text
`markdown`	`as(markdown)`	Markdown
`md`	`as(md)`	Markdown

export_opts	Description
`as(`*fileformat*`)`	specify document type
`replace`	overwrite existing file
docx_options	available when exporting to `.docx` files
html_options	available when exporting to `.html` files
pdf_options	available when exporting to `.pdf` files
excel_options	available when exporting to `.xls` and `.xlsx` files
tex_options	available when exporting to `.tex` files
smcl_option	available when exporting to `.smcl` files
txt_option	available when exporting to `.txt` files
md_option	available when exporting to `.markdown` and `.md` files

docx_options	Description
`noisily`	show the `putdocx` commands used to export to the Microsoft Word file
`dofile(`*filename*[`, replace`]`)`	save the `putdocx` commands used for exporting to the named do-file

html_options	Description
`append`	append to an existing file
`tableonly`	export only the table to the specified file
`cssfile(`*cssfile*`)`	define the styles in *cssfile* instead of *filename*
`prefix(`*prefix*`)`	use *prefix* to identify style classes

pdf_options	Description
`noisily`	show the `putpdf` commands used to export to the PDF file
`dofile(`*filename*`[, replace])`	save the `putpdf` commands used for exporting to the named do-file

excel_options	Description
`noisily`	show the `putexcel` commands used to export to the Excel file
`dofile(`*filename*`[ , replace])`	save the `putexcel` commands used for exporting to the named do-file
`sheet(`*sheetname*`[ , replace])`	specify the worksheet to use; the default sheet name is `Sheet1`
`cell(`*cell*`)`	specify the Excel upper-left cell as the starting position to export the table; the default is `cell(A1)`
`modify`	modify Excel file
`noopen`	do not open Excel file in memory

`noopen` does not appear in the dialog box.

tex_options	Description
`append`	append to an existing file
`tableonly`	export only the table to the specified file

smcl_option	Description
`append`	append to an existing file

txt_option	Description
`append`	append to an existing file

md_option	Description
`append`	append to an existing file

fontfamily specifies a valid font family.

unit may be `in` (inch), `pt` (point), or `cm` (centimeter). An inch is equivalent to 72 points and 2.54 centimeters. The default is `pt`.

variant may be `allcaps`, `smallcaps`, or `normal`.

`variant(allcaps)` changes the text to all uppercase letters; applicable when publishing items from a collection to Microsoft Word, PDF, LaTeX, and HTML files.

`variant(smallcaps)` changes the text to use large capitals for uppercase letters and smaller capitals for lowercase letters; applicable when publishing items from a collection to Microsoft Word, LaTeX, and HTML files.

`variant(normal)` changes the font variant back to normal and leaves the capitalization unchanged from the original text; applicable when publishing items from a collection to Microsoft Word, PDF, LaTeX, and HTML files.

smcl specifies the name of the SMCL directive to render text for SMCL output. The supported SMCL directives are `input`, `error`, `result`, and `text`.

latex specifies the name of a LaTeX macro to render text for LaTeX output. Example LaTeX macro names are `textbf`, `textsf`, `textrm`, and `texttt`. Custom LaTeX macros are also allowed. If *text* is to be rendered in a cell, title, or note, then *latex* is translated to the following when you export to LaTeX:

latex {*text*}

sspec is

[`background(`*bgcolor*`)` `foreground(`*fgcolor*`)` `pattern(`*fpattern*`)`]

bgcolor specifies the background color.

fgcolor specifies the foreground color.

fpattern specifies the fill pattern. A complete list of fill patterns is shown in the *Appendix*.

bgcolor, *fgcolor*, and *color* may be one of the colors listed in the *Appendix*; a valid RGB value in the form ### ### ###, for example, `171 248 103`; or a valid RRGGBB hex value in the form ######, for example, `ABF867`.

Options

Main

`totals(`*totals*`)` and `nototals` control which totals are to be displayed in the table. By default, all totals are reported.

`totals(`*totals*`)` specifies that totals be displayed only for the variables or interactions specified. *totals* can contain *rowvars*, *colvars*, *tabvars*, and interactions between any of these variables. Interactions can be specified by using the `#` operator.

`nototals` prevents `table` from displaying any totals.

Statistics

`statistic(`*stat*[`,` *statopts*]`)` specifies the statistic to be displayed. `statistic()` may be repeated to request multiple statistics.

Available statistics are

stat	Definition
`frequency`	frequency
`sumw`	sum of weights
`proportion`	proportion
`percent`	percentage
`rawproportion`	proportion ignoring optionally specified weights
`rawpercent`	percentage ignoring optionally specified weights

The following options may be specified in combination with statistics `proportion`, `percent`, `rawproportion`, and `rawpercent`:

statopts	Definition
`across(`*cellspec*`)`	percentages or proportions across levels of variables or interactions
`total`	compute overall percentages or proportions

cellspec may contain *rowvars*, *colvars*, *tabvars*, or an interaction between any of these variables. Interactions can be specified by using the `#` operator.

Formats

`nformat(%`*fmt* [*results*][`, basestyle`]`)` 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 statistic named in option `statistic()` (that is, any *stat*).

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 statistic named in option `statistic()` (that is, any *stat*).

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.

Title

`title(`*string*`)` adds the text *string* as a title to the table.

`titlestyles(`*text_styles*`)` changes the style for the table title. *text_styles* are the following:

`font(`[*fontfamily*] [`, size(#` [*unit*]`) color(`*color*`) variant(`*variant*`)` [`no`]`bold` [`no`]`italic` [`no`]`strikeout` [`no`]`underline`]`)` specifies the font style. These font style properties are applicable when exporting the table to Microsoft Word, Microsoft Excel, PDF, HTML, and LaTeX files, unless otherwise specified.

fontfamily specifies a valid font family. This font style property is applicable when publishing items from a collection to Microsoft Word, Microsoft Excel, PDF, and HTML files.

`size(#` [*unit*]`)` specifies the font size as a number optionally followed by units. This font style property is applicable when publishing items from a collection to Microsoft Word, Microsoft Excel, PDF, and HTML files.

`color(`*color*`)` specifies the text color.

`variant(`*variant*`)` specifies the font variant and capitalization.

`bold` and `nobold` specify the font weight. `bold` changes the font weight to bold; `nobold` changes the font weight back to normal.

`italic` and `noitalic` specify the font style. `italic` changes the font style to italic; `noitalic` changes the font style back to normal.

`strikeout` and `nostrikeout` specify whether to add a strikeout mark to the title. `strikeout` adds a strikeout mark to the title; `nostrikeout` changes the title back to normal.

`underline` and `nounderline` specify whether to underline the table title. `underline` adds a single line under the title; `nounderline` removes the underline.

Only one of `strikeout` or `underline` is allowed when publishing to HTML files.

`smcl(`*smcl*`)` specifies how to render the table title for SMCL output. This style property is applicable only when publishing items from a collection to a SMCL file.

`latex(`*latex*`)` specifies how to render the table title for LaTeX output. This style property is applicable only when publishing items from a collection to a LaTeX file.

`shading(`*sspec*`)` sets the background color, foreground color, and fill pattern. The background color is applicable when exporting the table to Microsoft Word, Microsoft Excel, PDF, HTML, and LaTeX files. The foreground color and fill pattern are applicable when exporting the table to Microsoft Word and Microsoft Excel.

Notes

`note(`*string*`)` adds the text *string* as a note to the table. `note()` may be specified multiple times to add multiple notes. Each note is placed on a new line.

`notestyles(`*text_styles*`)` changes the style for the table notes. *text_styles* are the following:

`font(`[*fontfamily*] [`, size(#` [*unit*]`) color(`*color*`) variant(`*variant*`)` [`no`]`bold` [`no`]`italic` [`no`]`strikeout` [`no`]`underline`]) specifies the font style. These font style properties are applicable when exporting the table to Microsoft Word, Microsoft Excel, PDF, HTML, and LaTeX files, unless otherwise specified.

fontfamily specifies a valid font family. This font style property is applicable when publishing items from a collection to Microsoft Word, Microsoft Excel, PDF, and HTML files.

`size(#` [*unit*]`)` specifies the font size as a number optionally followed by units. This font style property is applicable when publishing items from a collection to Microsoft Word, Microsoft Excel, PDF, and HTML files.

`color(`*color*`)` specifies the text color.

`variant(`*variant*`)` specifies the font variant and capitalization.

`bold` and `nobold` specify the font weight. `bold` changes the font weight to bold; `nobold` changes the font weight back to normal.

`italic` and `noitalic` specify the font style. `italic` changes the font style to italic; `noitalic` changes the font style back to normal.

`strikeout` and `nostrikeout` specify whether to add a strikeout mark to the notes. `strikeout` adds a strikeout mark to the note; `nostrikeout` changes the note back to normal.

`underline` and `nounderline` specify whether to underline the table notes. `underline` adds a single line under the notes; `nounderline` removes the underline.

Only one of `strikeout` or `underline` is allowed when publishing to HTML files.

`smcl(`*smcl*`)` specifies how to render the table notes for SMCL output. This style property is applicable only when publishing items from a collection to a SMCL file.

`latex(`*latex*`)` specifies how to render the table notes for LaTeX output. This style property is applicable only when publishing items from a collection to a LaTeX file.

`shading(`*sspec*`)` sets the background color, foreground color, and fill pattern. The background color is applicable when exporting the table to Microsoft Word, Microsoft Excel, PDF, HTML, and LaTeX files. The foreground color and fill pattern are applicable when exporting the table to Microsoft Word and Microsoft Excel.

Export

`export(`*filename*`.`*suffix*`[ , `*export_opts*`])` exports the table to the specified file. *export_opts* are the following:

`as(`*fileformat*`)` specifies the file format to which the table is to be exported. This option is rarely specified because, by default, `table` determines the format from the suffix of the file being created.

`replace` permits `table` to overwrite an existing file.

`noisily` specifies that `table` show the commands used to export the table to Microsoft Word, Microsoft Excel, and PDF files. The `putdocx`, `putexcel`, or `putpdf` command used to export the table will be displayed.

`dofile(`*filename*`[ , replace])` specifies that `table` save to *filename* the commands used to export the table to Microsoft Word, Microsoft Excel, and PDF files.

If *filename* already exists, it can be overwritten by specifying `replace`. If *filename* is specified without an extension, `.do` is assumed.

`append` specifies that `table` append the table to an existing file.

This option is applicable when you export the table to an HTML, a LaTeX, a SMCL, a `txt`, or a Markdown file. When you export to HTML and LaTeX files, the `append` option implies the `tableonly` option. Furthermore, when you export to HTML files, if the target CSS file already exists, `table` will also append to it.

`tableonly` specifies that only the table be exported to the specified HTML or LaTeX document. By default, `table` produces complete HTML and LaTeX documents.

When you export to an HTML file, if the `cssfile()` option is not specified, a CSS filename is constructed from *filename*, with the extension replaced with `.css`.

`cssfile(`*cssfile*`)` specifies that `table` define the styles in *cssfile* instead of *filename* when you export to HTML.

`prefix(`*prefix*`)` specifies that `table` use *prefix* to identify style classes when you export to HTML.

`sheet(`*sheetname* `[ , replace])` saves to the worksheet named *sheetname*. For more information about this option, see [RPT] **putexcel**.

`cell(`*cell*`)` specifies an Excel upper-left cell as the starting position to publish the table. The default is `cell(A1)`.

`modify` permits `putexcel set` to modify an Excel file. For more information about this option, see [RPT] **putexcel**.

`noopen` prevents `putexcel` from opening the Excel file in memory for modification. It does not appear in the dialog box. For more information about this option, see [RPT] **putexcel**.

Options

`listwise` handles missing values through listwise deletion, meaning that the entire observation is omitted from the sample if any variable specified in a `statistic()` option is missing for that observation. By default, `table` will omit an observation only if all variables specified in all `statistic()` options are missing for that observation.

`missing` specifies that numeric missing values of any *rowvars*, *colvars*, or *tabvars* be treated as valid categories. By default, observations with a numeric missing value in *rowvars*, *colvars*, or *tabvars* are omitted.

`showcounts` specifies that `table` report the sample size for each variable specified in option `statistic()`.

`zerocounts` specifies that `table` report a 0 in empty cells for the `frequency` statistic.

`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:

`markvar(`*newvar*`)` generates an indicator variable that identifies the observations used in the tabulation.

Remarks and examples

Remarks are presented under the following headings:

Introduction

The table command allows you to create complex tables beyond one- and two-way tabulations. In multiway tabulations, you can display frequencies across levels of two or more variables. You can have levels of one variable nested within levels of another variable in columns, in rows, or in both dimensions. And with a single command, you can create separate tables for levels of one or more variables or for different results.

Tables with columns defined by multiple variables

We use 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 dataset also contains some measures of health, including whether the individual has high blood pressure (highbp).

Before we create any tables, we will modify a few labels in our dataset so that they will appear as we wish in our tables.

```
. use https://www.stata-press.com/data/r19/nhanes2l
(Second National Health and Nutrition Examination Survey)
. label define yesno 0 "No" 1 "Yes"
. label values highbp diabetes heartatk yesno
. label variable diabetes "Diabetes"
```

Suppose we want to examine how many males and females in each age group have high blood pressure. Let's place the levels of age group on the rows and the levels of high blood pressure and sex on the columns.

```
. table (agegrp) (sex highbp), nototals
```

	Sex			
	Male		Female	
	High blood pressure		High blood pressure	
	No	Yes	No	Yes
Age group				
20–29	825	291	1,103	101
30–39	480	290	687	165
40–49	336	274	434	228
50–59	255	347	335	354
60–69	568	801	625	866
70+	147	301	180	358

By default, table includes the totals for each category; we added the nototals option to suppress them here.

To better compare the occurrence of high blood pressure, let's now compute percentages of highbp. Below, we create the same table, but within each sex and age group combination, we report the percentage of individuals with and without high blood pressure.

```
. table (agegrp) (sex highbp), nototals statistic(percent, across(highbp))
```

	Sex			
	Male		Female	
	High blood pressure		High blood pressure	
	No	Yes	No	Yes
Age group				
20–29	73.92	26.08	91.61	8.39
30–39	62.34	37.66	80.63	19.37
40–49	55.08	44.92	65.56	34.44
50–59	42.36	57.64	48.62	51.38
60–69	41.49	58.51	41.92	58.08
70+	32.81	67.19	33.46	66.54

Here we see that 26.08% of males in their 20s have high blood pressure and only 8.39% of females in their 20s have high blood pressure.

If we had simply typed `statistic(percent)`, then we would see the percentage of observations in each cell. With the suboption `across()`, we can compute the percentage within each sex and age group combination (across levels of highbp).

Next, let's request that percentages be calculated across the categories of high blood pressure and sex. We can alternatively think of this as being percentages within age group.

```
. table (agegrp) (sex highbp), nototals statistic(percent, across(highbp#sex))
```

	Sex			
	Male		Female	
	High blood pressure		High blood pressure	
	No	Yes	No	Yes
Age group				
20–29	35.56	12.54	47.54	4.35
30–39	29.59	17.88	42.36	10.17
40–49	26.42	21.54	34.12	17.92
50–59	19.75	26.88	25.95	27.42
60–69	19.86	28.01	21.85	30.28
70+	14.91	30.53	18.26	36.31

Here we see that 12.54% of individuals in their 20s are males with high blood pressure and 4.35% are females with high blood pressure.

We can reverse the order of our column variables so that we have the levels of sex nested within levels of high blood pressure. We will again request percentages across the categories of highbp.

```
. table (agegrp) (highbp sex), nototals statistic(percent, across(highbp))
```

	High blood pressure			
	No		Yes	
	Sex		Sex	
	Male	Female	Male	Female
Age group				
20–29	73.92	91.61	26.08	8.39
30–39	62.34	80.63	37.66	19.37
40–49	55.08	65.56	44.92	34.44
50–59	42.36	48.62	57.64	51.38
60–69	41.49	41.92	58.51	58.08
70+	32.81	33.46	67.19	66.54

The last two columns represent the percentage of males with high blood pressure in the age group and the percentage of females with high blood pressure in the age group. We can clearly see that, across all age groups, the percent of males who have high blood pressure is greater than the percentage of females with high blood pressure.

Perhaps we want a table that includes only the two columns on the right. To create a table with the percentages of individuals with high blood pressure, we can take advantage of a unique feature of `table`—its results are automatically stored in a "collection" and can be easily customized. Specifically, when we create a table using the `table` command, the results are stored in a collection named `Table`, and these results replace the results from any previous `table` command. The `collect` suite of commands can be used to change the layout, style, and formatting of tables created from results in collection; see [TABLES] **Intro** to learn about collections of results and creating customized tables. For our table, we will use the `collect layout` command, which specifies how items from a collection should be arranged. Below, we arrange the percentages in a table with rows defined by the categories of `agegrp` and columns defined by the categories of sex and category 1 of `highbp`.

```
. collect layout (agegrp) (highbp[1]#sex)
Collection: Table
      Rows: agegrp
   Columns: highbp[1]#sex
   Table 1: 7 x 2
```

	High blood pressure	
	Yes	
	Sex	
	Male	Female
Age group		
20–29	26.07527	8.388704
30–39	37.66234	19.3662
40–49	44.91803	34.44109
50–59	57.6412	51.37881
60–69	58.50986	58.08182
70+	67.1875	66.54275

Appending tables

When we specify multiple variables for the row or column specification, the levels of one variable are nested within the levels of another. When you simply wish to join rows or columns from multiple tables, this can be easily done with the append option.

For example, we first create a table with the percentage of males and females in each age group with diabetes.

```
. table (sex agegrp) (diabetes), nototals statistic(percent, across(diabetes))
```

	Diabetes	
	No	Yes
Sex		
Male		
Age group		
20–29	99.64	0.36
30–39	99.61	0.39
40–49	97.38	2.62
50–59	94.68	5.32
60–69	91.96	8.04
70+	88.39	11.61
Female		
Age group		
20–29	99.09	0.91
30–39	97.88	2.12
40–49	96.07	3.93
50–59	94.19	5.81
60–69	91.42	8.58
70+	89.03	10.97

We want to include the same information for highbp and heartatk in the same table, which indicates whether someone has had a heart attack. To do this, we will run the table command three times, once specifying each of these variables as the column variable. We will use the name(table1) option to specify that the results be stored in a collection named table1. To the second and third table commands, we will add the append option so that all the results are stored in the same collection rather than overriding the results from the previous command.

```
. quietly: table (sex agegrp) (diabetes), nototals
> statistic(percent, across(diabetes)) name(table1)

. quietly: table (sex agegrp) (highbp), nototals
> statistic(percent, across(highbp)) name(table1) append

. quietly: table (sex agegrp) (heartatk), nototals
> statistic(percent, across(heartatk)) name(table1) append
```

With the results from all our `table` commands stored in one collection, we can again take advantage of the `collect layout` command to arrange results into a table. We request that `sex` and `agegroup` define the rows. By including the `#` between variable names, we specify that we want the levels of these variables to be interacted to form the rows. We request that levels of `diabetes`, `highbp`, and `heartatk` form the columns. Because we did not include `#` between the variable names, their levels will not be interacted. Instead, they will be listed one after the other.

```
. collect layout (sex#agegrp) (diabetes highbp heartatk)
Collection: table1
      Rows: sex#agegrp
   Columns: diabetes highbp heartatk
   Table 1: 17 x 6
```

	Diabetes		High blood pressure		Prior heart attack	
	No	Yes	No	Yes	No	Yes
Sex						
Male						
Age group						
20–29	99.64	0.36	73.92	26.08	100.00	
30–39	99.61	0.39	62.34	37.66	99.74	0.26
40–49	97.38	2.62	55.08	44.92	98.03	1.97
50–59	94.68	5.32	42.36	57.64	92.36	7.64
60–69	91.96	8.04	41.49	58.51	86.56	13.44
70+	88.39	11.61	32.81	67.19	83.48	16.52
Female						
Age group						
20–29	99.09	0.91	91.61	8.39	99.92	0.08
30–39	97.88	2.12	80.63	19.37	99.76	0.24
40–49	96.07	3.93	65.56	34.44	98.79	1.21
50–59	94.19	5.81	48.62	51.38	96.66	3.34
60–69	91.42	8.58	41.92	58.08	94.57	5.43
70+	89.03	10.97	33.46	66.54	92.01	7.99

Multiple tables with specified totals

There may be times when instead of creating one large table for multiple variables, you would prefer to create separate tables for each level of one or more variables or for different statistics. For example, we previously had levels of age group nested within categories of sex. Now, we would like to create tables that show how males and females in each age group rate their own health. The variable `hlthstat` records how individuals self-rate their health. Let's create a table that shows what percent of males in each age group selected each of the health categories and a separate table to list the percent of females.

```
. table (agegrp) (hlthstat) (sex), statistic(percent, across(hlthstat))
```

Sex = Male

	Health status					
	Excellent	Very good	Good	Fair	Poor	Total
Age group						
20–29	39.61	32.71	20.97	5.82	0.90	100.00
30–39	36.88	29.61	26.10	5.84	1.56	100.00
40–49	28.03	25.90	29.51	12.30	4.26	100.00
50–59	17.80	21.30	35.11	15.97	9.82	100.00
60–69	13.33	19.56	28.94	23.00	15.16	100.00
70+	14.77	14.99	26.62	28.41	15.21	100.00
Total	25.50	24.71	27.30	14.71	7.78	100.00

Sex = Female

	Health status					
	Excellent	Very good	Good	Fair	Poor	Total
Age group						
20–29	32.39	34.55	24.92	6.98	1.16	100.00
30–39	29.76	30.71	28.12	9.29	2.12	100.00
40–49	26.71	23.07	29.29	15.63	5.31	100.00
50–59	17.15	20.93	33.43	20.20	8.28	100.00
60–69	10.76	20.31	33.22	25.49	10.22	100.00
70+	10.78	19.14	26.39	30.48	13.20	100.00
Total	21.29	25.40	29.45	17.47	6.40	100.00

Sex = Total

	Health status					
	Excellent	Very good	Good	Fair	Poor	Total
Age group						
20–29	35.86	33.66	23.02	6.42	1.03	100.00
30–39	33.15	30.19	27.16	7.65	1.85	100.00
40–49	27.34	24.43	29.39	14.03	4.81	100.00
50–59	17.46	21.10	34.21	18.23	9.00	100.00
60–69	11.99	19.95	31.17	24.30	12.59	100.00
70+	12.59	17.26	26.50	29.54	14.11	100.00
Total	23.29	25.07	28.43	16.16	7.05	100.00

We see that 14.77% of males in their 70s and beyond rated their health as excellent and 10.78% of females in their 70s and beyond rated their health as excellent.

Note that we actually created three tables, one for males, one for females, and one for everybody. We are mainly interested in the first two tables and would like to drop the third table. But if we use the `nototals` option, we will not get the row totals we see above. Instead, we can specify which totals we do want with the `totals()` option:

```
. table (agegrp) (hlthstat) (sex), statistic(percent, across(hlthstat))
> totals(sex#agegrp sex#hlthstat sex)
```

Sex = Male

	Health status					
	Excellent	Very good	Good	Fair	Poor	Total
Age group						
20–29	39.61	32.71	20.97	5.82	0.90	100.00
30–39	36.88	29.61	26.10	5.84	1.56	100.00
40–49	28.03	25.90	29.51	12.30	4.26	100.00
50–59	17.80	21.30	35.11	15.97	9.82	100.00
60–69	13.33	19.56	28.94	23.00	15.16	100.00
70+	14.77	14.99	26.62	28.41	15.21	100.00
Total	25.50	24.71	27.30	14.71	7.78	100.00

Sex = Female

	Health status					
	Excellent	Very good	Good	Fair	Poor	Total
Age group						
20–29	32.39	34.55	24.92	6.98	1.16	100.00
30–39	29.76	30.71	28.12	9.29	2.12	100.00
40–49	26.71	23.07	29.29	15.63	5.31	100.00
50–59	17.15	20.93	33.43	20.20	8.28	100.00
60–69	10.76	20.31	33.22	25.49	10.22	100.00
70+	10.78	19.14	26.39	30.48	13.20	100.00
Total	21.29	25.40	29.45	17.47	6.40	100.00

`sex#agegrp` gives us the row totals, the total for each age group within each category of `sex`. `sex#hlthstat` provides us with the column totals, one of which would be the total percent of females that rated their health as excellent. Finally, `sex` gives us the total in the rightmost cell in the bottom of each table.

In all our previous tables, we used variables to define the rows and columns, but you can also use results in the row, column, and table identifiers. For example, suppose that in addition to reporting percentages for males and females in each age group, we wanted to report frequencies. We can create separate tables for the percentages and frequencies by specifying `result` in the third set of parentheses. To include the totals for each age group and for each category of sex, we specify the interaction of `sex` and `agegrp` in the `totals()` option:

```
. table (sex agegrp) (hlthstat) (result), statistic(percent, across(hlthstat))
> statistic(frequency) totals(sex#agegrp)
```

Percent

	Health status					
	Excellent	Very good	Good	Fair	Poor	Total
Sex						
Male						
Age group						
20–29	39.61	32.71	20.97	5.82	0.90	100.00
30–39	36.88	29.61	26.10	5.84	1.56	100.00
40–49	28.03	25.90	29.51	12.30	4.26	100.00
50–59	17.80	21.30	35.11	15.97	9.82	100.00
60–69	13.33	19.56	28.94	23.00	15.16	100.00
70+	14.77	14.99	26.62	28.41	15.21	100.00
Female						
Age group						
20–29	32.39	34.55	24.92	6.98	1.16	100.00
30–39	29.76	30.71	28.12	9.29	2.12	100.00
40–49	26.71	23.07	29.29	15.63	5.31	100.00
50–59	17.15	20.93	33.43	20.20	8.28	100.00
60–69	10.76	20.31	33.22	25.49	10.22	100.00
70+	10.78	19.14	26.39	30.48	13.20	100.00

Frequency

	Health status					
	Excellent	Very good	Good	Fair	Poor	Total
Sex						
Male						
Age group						
20–29	442	365	234	65	10	1,116
30–39	284	228	201	45	12	770
40–49	171	158	180	75	26	610
50–59	107	128	211	96	59	601
60–69	182	267	395	314	207	1,365
70+	66	67	119	127	68	447
Female						
Age group						
20–29	390	416	300	84	14	1,204
30–39	253	261	239	79	18	850
40–49	176	152	193	103	35	659
50–59	118	144	230	139	57	688
60–69	160	302	494	379	152	1,487
70+	58	103	142	164	71	538

In the table of frequencies, we see that of the 1,204 females in their 20s, only 390 rated their health as excellent. In the table of percentages, we see that this is equal to 32.39% of females in their 20s.

If we wish to include one of these tables, for example, in a paper or on a webpage, we can easily export it in LaTeX, Word, Excel, HTML, and a variety of other formats by using the `export()` option.

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". In *Vital and Health Statistics*, ser. 1, no. 15. Hyattsville, MD: National Center for Health Statistics.

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 oneway** — One-way tabulation

[R] **table twoway** — Two-way tabulation

[TABLES] **Intro** — Introduction