table oneway — One-way tabulation

Description

In this entry, we discuss how to use `table` to create a one-way tabulation, including frequencies, percentages, and proportions.

Quick start

One-way table of frequencies, with rows corresponding to the levels of `a1`

```
table a1
```

Same as above, but with columns corresponding to the levels of `a1`

```
table () a1
```

Same as above, but treat missing values like other values of `a1`

```
table () a1, missing
```

One-way table of frequencies, using the collection style `mystyle`

```
table a1, style(mystyle)
```

One-way table of frequencies and percentages

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

Menu

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

Syntax

Basic one-way tabulation

```
table varname [ if ] [ in ] [ weight ] [ , options ]
```

Customized one-way tabulation

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

rowspec and *colspec* may be empty or may include *varname*, `result`, or *varname* and `result`, where `result` refers to the requested statistics.

options	Description
Main	
`nototals`	suppress the marginal totals
Statistics	
`statistic(`*stat*`)`	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	
`missing`	treat numeric missing values of *varname* 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
`markvar(`*newvar*`)`	create *newvar* that identifies observations used in the tabulation

`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

`nototals` prevents `table` from displaying the row or column totals.

Statistics

`statistic(`*stat*`)` 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

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* and *colspec*). 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

`missing` specifies that numeric missing values of *varname* be treated as valid categories. By default, observations with a numeric missing value in *varname* 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:

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

Remarks and examples

Remarks are presented under the following headings:

Tabulation of one variable

To obtain a one-way tabulation that reports the number of observations for each level of a categorical variable, we need specify only the name of the categorical variable following `table`.

To demonstrate, we use data from the Second National Health and Nutrition Examination Survey (NHANES II) (McDowell et al. 1981). We tabulate the `hlthstat` variable, which contains individuals' self-reported health status categories.

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

	Frequency
Health status	
Excellent	2,407
Very good	2,591
Good	2,938
Fair	1,670
Poor	729
Total	10,335

We see that more people self-reported having excellent, very good, or good health status than reported having fair or poor health status.

Above, we see frequencies for those who reported a health status. This information is not available for some individuals in the dataset. We can determine how many missing values we have for this variable by adding the `missing` option

```
. table hlthstat, missing
```

	Frequency
Health status	
Excellent	2,407
Very good	2,591
Good	2,938
Fair	1,670
Poor	729
.	2
Blank but applicable	14
Total	10,351

We find that there is missing health status data for 16 individuals—2 with a generic missing value and 14 whose responses were labeled "Blank but applicable".

Tabulation, including percentages

In addition to frequencies, we can report the proportion or percentage of observations in each health status category. By default, `table` reports frequencies, which is equivalent to including the `statistic(frequency)` option. Here we include that option along with the `statistic(percent)` option to report both frequencies and percentages.

```
. table hlthstat, statistic(frequency) statistic(percent)
```

	Frequency	Percent
Health status		
Excellent	2,407	23.29
Very good	2,591	25.07
Good	2,938	28.43
Fair	1,670	16.16
Poor	729	7.05
Total	10,335	100.00

Now, it is clear that 28.43% of respondents reported having good health.

Customizing results

There are a number of ways that you can customize the results in your table.

In some cases, you may prefer to place frequencies and percentages on the rows and the levels of the variable being tabulated on the columns. To do this, you can include both the row and column specifications in parentheses following `table`. Here we use `result` in the first set of parentheses to request that the statistics be placed on rows and the variable `hlthstat` in the second set of parentheses to request that the levels of this variable be placed on the columns.

```
. table (result) (hlthstat), statistic(frequency) statistic(percent)
```

	Health status					
	Excellent	Very good	Good	Fair	Poor	Total
Frequency	2,407	2,591	2,938	1,670	729	10,335
Percent	23.29	25.07	28.43	16.16	7.05	100.00

Alternatively, we could have omitted `result` and typed

```
. table () (hlthstat), statistic(freq) statistic(percent)
```

Because we requested that `hlthstat` be moved to the columns by specifying it in the second set of parentheses, `table` automatically moves the requested statistics to the rows.

If instead of a short and wide table, you prefer a tall and narrow table, you can specify that both the levels of `hlthstat` and the statistics be used to define the rows by including the variable name and `result` in the first set of parentheses.

```
. table (hlthstat result), statistic(frequency) statistic(percent)
```

Health status	
Excellent	
Frequency	2,407
Percent	23.29
Very good	
Frequency	2,591
Percent	25.07
Good	
Frequency	2,938
Percent	28.43
Fair	
Frequency	1,670
Percent	16.16
Poor	
Frequency	729
Percent	7.05
Total	
Frequency	10,335
Percent	100.00

In addition to modifying the layout of the table, we may want to customize the results reported within the cells of the table. For instance, we can specify that the percentages be reported using only one decimal place by using the `nformat()` option. Here we return to the two-column table layout.

```
. table hlthstat, statistic(frequency) statistic(percent)
> nformat(%5.1f percent)
```

	Frequency	Percent
Health status		
Excellent	2,407	23.3
Very good	2,591	25.1
Good	2,938	28.4
Fair	1,670	16.2
Poor	729	7.1
Total	10,335	100.0

The table command produces its output using a default set of styles, typically those defined in the table style but could be any other style that you have set as the default by using set table_style. When customizing our tables, we can take advantage of one of the styles described in [TABLES] **Predefined styles**. For instance, for tables with only one or two row variables, row labels that are right-aligned may be preferred. Here we use the table-right style.

```
. table hlthstat, statistic(frequency) statistic(percent)
> nformat(%5.1f percent) style(table-right)
```

	Frequency	Percent
Health status		
Excellent	2,407	23.3
Very good	2,591	25.1
Good	2,938	28.4
Fair	1,670	16.2
Poor	729	7.1
Total	10,335	100.0

Advanced customization

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.

Continuing with our example above, if we want to shorten the labels on the column headings, we could use the collect label levels command to define our new labels. After a change using collect, we can use collect preview to see the results.

```
. collect label levels result frequency "Freq" percent "%", modify
. collect preview
```

	Freq	%
Health status		
Excellent	2,407	23.3
Very good	2,591	25.1
Good	2,938	28.4
Fair	1,670	16.2
Poor	729	7.1
Total	10,335	100.0

We could continue making style edits to this table. When we are happy with the result, we can then export it to the format of our choice using collect export.

See [TABLES] **collect label** for details on the collect label command we used here, and for an overview of the collect suite, see [TABLES] **Intro**.

Stored results

`table` stores the following in `r()`:

Scalars
`r(N)` number of observations

References

Bruun, N. H. 2022. Interactively building table reports with basetable. *Stata Journal* 22: 416–429.

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 multiway** — Multiway tables

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

[R] **tabulate oneway** — One-way table of frequencies

[TABLES] **Intro** — Introduction