

**putexcel advanced** — Export results to an Excel file using advanced syntax

[Description](#)  
[Options](#)

[Quick start](#)  
[Remarks and examples](#)

[Menu](#)  
[References](#)

[Syntax](#)  
[Also see](#)

## Description

`putexcel` with the advanced syntax may be used to simultaneously write Stata [expressions](#), [matrices](#), tables, images, and [returned results](#) to an Excel file. It may also be used to format existing contents of cells in a worksheet. This syntax is intended for use by programmers of commands that call `putexcel` in the background and by other advanced users. Excel 1997/2003 (`.xls`) files and Excel 2007/2010 and newer (`.xlsx`) files are supported.

`putexcel set` sets the Excel file to create, modify, or replace in subsequent `putexcel` commands. You must set the destination file before using any other `putexcel` commands. `putexcel save` closes the open file in memory and saves it to disk. `putexcel clear` clears the file information set by `putexcel set`. `putexcel describe` displays the file information set by `putexcel set`.

For a simplified version of the syntax, see [\[RPT\] putexcel](#).

## Quick start

Declare the first sheet of `myresults.xlsx` to be the destination workbook for subsequent `putexcel` commands

```
putexcel set myresults
```

Same as above, but use a new sheet named `Estimation Results` and replace the existing workbook

```
putexcel set myresults, sheet("Estimation Results", replace)
```

Write estimation results in `e(b)` to column B, starting in row 2, with the results formatted to have two decimal places, matrix row names in column A, and the title “Coefs.” in cell B1

```
matrix b=e(b)'  
putexcel B1="Coefs." A2=matrix(b), rownames nformat(number_d2)
```

Add a thin border under cells A1 to B1 and italicize the text

```
putexcel (A1:B1), border(bottom) italic
```

Write “Some text that is too long to fit” to cell D1 and set the text to wrap within the cell

```
putexcel D1="Some text that is too long to fit", txtwrap
```

Merge D1 with D2, E1, and E2, and horizontally and vertically center

```
putexcel (D1:E2), merge hcenter vcenter
```

## Menu

File > Export > Results to Excel spreadsheet (\*.xls;\*.xlsx)

## Syntax

*Set workbook for export*

```
putexcel set filename [ , set_options ]
```

*Specify formatting and output*

```
putexcel spec1 [spec2 [...]] [ , export_options format_options ]
```

*Close and save current Excel file*

```
putexcel save
```

*Describe current export settings*

```
putexcel describe
```

*Clear current export settings*

```
putexcel clear
```

*spec* may be *ul\_cell* or *cellrange* of the form *ul\_cell:lr\_cell* if no output is to be written or may be one of the following **output types**:

*ul\_cell* = *exp*

*ul\_cell:lr\_cell* = *exp*

*ul\_cell* = `matrix(matname)`

*ul\_cell* = `image(filename)`

*ul\_cell* = `returnset`

*ul\_cell* = `formula(formula)`

*ul\_cell* = `hyperlink(link, link_name)`

*ul\_cell* = `etable[#1 #2 ... #n]`

*ul\_cell* = `collect`

*ul\_cell* is a valid Excel upper-left cell specified using standard Excel notation, and *lr\_cell* is a valid Excel lower-right cell. If you specify *ul\_cell* as the output location multiple times, the rightmost specification is the one written to the Excel file.

---

<i>set_options</i>	Description
<code>open</code>	open Excel file in memory
<code>modify</code>	modify Excel file
<code>replace</code>	overwrite Excel file
<code>sheet(sheetname [ , replace ])</code>	specify the worksheet to use; the default sheet name is Sheet1

---

<i>export_options</i>	Description
Main	
<code>overwritefmt</code>	overwrite existing cell formatting when exporting new content
<code>asdate</code>	convert Stata date (%td-formatted) <i>exp</i> to an Excel date
<code>asdatetime</code>	convert Stata datetime (%tc-formatted) <i>exp</i> to an Excel datetime
<code>asdatenum</code>	convert Stata date <i>exp</i> to an Excel date number, preserving the cell's format
<code>asdatetimenum</code>	convert Stata datetime <i>exp</i> to an Excel datetime number, preserving the cell's format
<code>names</code>	also write row names and column names for matrix <i>name</i> ; may not be combined with <code>rownames</code> or <code>colnames</code>
<code>rownames</code>	also write matrix row names for matrix <i>name</i> ; may not be combined with <code>names</code> or <code>colnames</code>
<code>colnames</code>	also write matrix column names for matrix <i>name</i> ; may not be combined with <code>names</code> or <code>rownames</code>
<code>colwise</code>	write results in <i>returnset</i> to consecutive columns instead of rows

---

<i>format_options</i>	Description
Number <u>nformat</u> ( <i>excelnfmt</i> )	specify format for numbers
Alignment left hcenter right top vcenter bottom <u>txtindent</u> (#) <u>txtrotate</u> (#) [no] <u>txtwrap</u> [no] <u>shrinkfit</u> merge unmerge	left-align text center text horizontally right-align text vertically align text with the top center text vertically vertically align text with the bottom indent text by # spaces; default is 0 rotate text by # degrees; default is 0 wrap text within each cell shrink text to fit the cell width merge cells in <i>cellrange</i> separate merged cells identified by <i>ul_cell</i>
Font <u>font</u> ([ <i>fontname</i> ] [, <i>size</i> [, <i>color</i> ]]) [no] <u>italic</u> [no] <u>bold</u> [no] <u>underline</u> [no] <u>strikeout</u> <u>script</u> (sub   super   none)	specify font, font size, and font color format text as italic format text as bold underline text in the specified cells strikeout text in the specified cells specify subscript or superscript formatting
Border <u>border</u> ( <i>border</i> [, <i>style</i> [, <i>color</i> ]]) <u>dborder</u> ( <i>direction</i> [, <i>style</i> [, <i>color</i> ]])	specify horizontal and vertical cell border style specify diagonal cell border style
Fill <u>fpattern</u> ( <i>pattern</i> [, <i>fgcolor</i> [, <i>bgcolor</i> ]])	specify fill pattern for cells

## Output types

*exp* writes a valid Stata expression to a cell; see [U] **13 Functions and expressions**. Stata dates and datetimes differ from Excel dates and datetimes. To properly export date and datetime values, use *asdate* and *asdatetime*.

*matrix*(*matname*) writes the values from a Stata matrix to Excel. Stata determines where to place the data in Excel by default from the size of the matrix (the number of rows and columns) and the location you specified in *ul\_cell*. By default, *ul\_cell* contains the first element of *matname*, and matrix row names and column names are not written.

*image*(*filename*) writes a portable network graphics (.png), JPEG (.jpg), Windows metafile (.wmf), device-independent bitmap (.dib), enhanced metafile (.emf), or bitmap (.bmp) file to an Excel worksheet. The upper-left corner of the image is aligned with the upper-left corner of the specified *ul\_cell*. The image is not resized. If *filename* contains spaces, it must be enclosed in double quotes.

*returnset* is a shortcut name that is used to identify a group of **return** values. *returnset* may be any one of the following:

*returnset*

<u>escalars</u>	<u>escalarnames</u>
<u>rscalars</u>	<u>rscalarnames</u>
<u>emacros</u>	<u>emacronames</u>
<u>rmacros</u>	<u>rmacronames</u>
<u>ematrices</u>	<u>ematrixnames</u>
<u>rmatrices</u>	<u>rmatrixnames</u>
<u>e*</u>	<u>enames</u>
<u>r*</u>	<u>rnames</u>

`formula(formula)` writes an Excel formula to the cell specified in `ul_cell`. `formula` may be any valid Excel formula. Stata does not validate formulas; the text is passed literally to Excel.

`hyperlink(link, link_name)` writes a hyperlink to the cell specified in `ul_cell`. `link` may be an external file, a cell, or a webpage. Stata does not validate the links; the text is passed literally to Excel. Examples:

```
putexcel A1 = hyperlink("[auto.xlsx]Sheet1!C2", "Result")
putexcel A1 = hyperlink("https://www.stata.com", "StataCorp")
putexcel A1 = hyperlink(".\auto.xlsx", "1978 automobile data")
```

`etable[#1 #2 ... #n]` adds an automatically generated table to an Excel file starting in `ul_cell`. The table may be derived from the coefficient table of the last estimation command, from the table of margins after the last `margins` command, or from the table of results from one or more models displayed by `estimates table`.

If the estimation command outputs  $n > 1$  coefficient tables, the default is to add all tables and assign the corresponding table names `tablename1`, `tablename2`, ..., `tablenamen`. To specify which tables to add, supply the optional numlist to `etable`. For example, to add only the first and third tables from the estimation output, specify `etable(1 3)`. A few estimation commands do not support the `etable` output type. See [Unsupported estimation commands](#) in [\[RPT\] Appendix for putdocx](#) for a list of estimation commands that are not supported by `putexcel`.

`collect` adds a table from the current collection to an Excel file starting in `ul_cell`. This table may be created using `collect` or `table`. See [\[TABLES\] Intro](#) for more information on using `collect` to create a customized table from a collection of results from one or more Stata commands. See [\[R\] table intro](#) for information on using `table` to create tabulations, tables of summary statistics, tables of regression results, and more.

## Options

Set

`open` permits `putexcel set` to open the Excel file in memory for modification. The Excel file is written to disk when `putexcel save` is issued.

`modify` permits `putexcel set` to modify an Excel file.

`replace` permits `putexcel set` to overwrite an existing Excel workbook. The workbook is overwritten when the first `putexcel` command is issued unless the `open` option is used.

`sheet(sheetname [, replace])` saves to the worksheet named `sheetname`. If there is no worksheet named `sheetname` in the workbook, then a new sheet named `sheetname` is created. If this option is not specified, `Sheet1` is used.

`replace` permits `putexcel set` to overwrite `sheetname` if it exists in the specified `filename`.

`overwritefmt` causes `putexcel` to remove any existing cell formatting in the cell or cells to which it is writing new output. By default, all existing cell formatting is preserved. `overwritefmt`, when combined with a cell range, writes the cell format more efficiently.

`asdate` tells `putexcel` that the specified *exp* is a Stata `%td`-formatted date that should be converted to an Excel date with `m/d/yyyy` Excel date format.

This option has no effect if an *exp* is not specified as one of the output types.

`asdatetime` tells `putexcel` that the specified *exp* is a Stata `%tc`-formatted datetime that should be converted to an Excel datetime with `m/d/yyyy h:mm` Excel datetime format.

This option has no effect if an *exp* is not specified as one of the output types.

`asdatetimeum` tells `putexcel` that the specified *exp* is a Stata `%td`-formatted date that should be converted to an Excel date number, preserving the cell's format.

This option has no effect if an *exp* is not specified as one of the output types.

`asdatetimeum` tells `putexcel` that the specified *exp* is a Stata `%tc`-formatted datetime that should be converted to an Excel datetime number, preserving the cell's format.

This option has no effect if an *exp* is not specified as one of the output types.

`names` specifies that matrix row names and column names be written into the Excel worksheet along with the matrix values. If you specify `names`, then `ul_cell` will be blank, the cell to the right of it will contain the name of the first column, and the cell below it will contain the name of the first row. `names` may not be specified with `rownames` or `colnames`.

This option has no effect if `matrix()` is not specified as one of the output types.

`rownames` specifies that matrix row names be written into the Excel worksheet along with the matrix values. If you specify `rownames`, then `ul_cell` will contain the name of the first row. `rownames` may not be specified with `names` or `colnames`.

This option has no effect if `matrix()` is not specified as one of the output types.

`colnames` specifies that matrix column names be written into the Excel worksheet along with the matrix values. If you specify `colnames`, then `ul_cell` will contain the name of the first column. `colnames` may not be specified with `names` or `rownames`.

This option has no effect if `matrix()` is not specified as one of the output types.

`colwise` specifies that if a *returnset* is used, the values written to the Excel worksheet be written in consecutive columns. By default, the values are written in consecutive rows.

This option has no effect if a *returnset* is not specified as one of the output types.

## Number

`nformat(excelfmt)` changes the numeric format of a cell range. Codes for commonly used formats are shown in the table of numeric formats in the [Appendix](#). However, any valid Excel format is permitted. Formats are formed from combinations of the following symbols.

Symbol	Description	Cell value	Fmt code	Cell displays
0	Digit placeholder (add zeros)	8.9	#.00	8.90
#	Digit placeholder (no zeros)	8.9	#.##	8.9
?	Digit placeholder (add space)	8.9	0.0?	8.9
.	Decimal point			
%	Percentage	.1	%	10%
,	Thousands separator	10000	#,###	10,000
E- E+ e- e+	Scientific format	12200000	0.00E+00	1.22E+07
\$-+/( ):space	Display the symbol	12	(000)	(012)
\	Escape character	3	0\!	3!
*	Repeat character (fill in cell width)	3	3*	3xxxxx
_	Skip width of next character	-1.2	_0.0	1.2
"text"	Display text in quotes	1.23	0.00 "a"	1.23 a
@	Text placeholder	b	"a"@c"	abc

Formats that contain spaces must be enclosed in double quotes.

## Alignment

`left` sets the specified cells to have contents left-aligned within the cell. `left` may not be combined with `right` or `hcenter`. Right-alignment is the Excel default for numeric values and need not be specified when outputting numbers.

`hcenter` sets the specified cells to have contents horizontally centered within the cell. `hcenter` may not be combined with `left` or `right`.

`right` sets the specified cells to have contents right-aligned within the cell. `right` may not be combined with `left` or `hcenter`. Left-alignment is the Excel default for text and need not be specified when outputting strings.

`top` sets the specified cells to have contents vertically aligned with the top of the cell. `top` may not be combined with `bottom` or `vcenter`.

`vcenter` sets the specified cells to have contents vertically aligned with the center of the cell. `vcenter` may not be combined with `top` or `bottom`.

`bottom` sets the specified cells to have contents vertically aligned with the bottom of the cell. `bottom` may not be combined with `top` or `vcenter`.

`txtindent(#)` sets the text indentation in each cell in a cell range. `#` must be an integer between 0 and 15.

`txtrotate(#)` sets the text rotation in each cell in a cell range. `#` must be an integer between 0 and 180 or equal to 255. `txtrotate(0)` is equal to no rotation and is the default. `txtrotate(255)` specifies vertical text. Values 1–90 rotate the text counterclockwise 1 to 90 degrees. Values 91–180 rotate the text clockwise 1 to 90 degrees.

`txtwrap` and `notxtwrap` specify whether the text is to be wrapped in a cell or within each cell in a range of cells. The default is no wrapping. `notxtwrap` has an effect only if the cell or cells were previously formatted to wrap. `txtwrap` may not be specified with `shrinkfit`.

`shrinkfit` and `noshrinkfit` specify whether the text is to be shrunk to fit in the cell width of a cell or in each cell of a range of cells. The default is no shrinking. `noshrinkfit` has an effect only if the cell or cells were previously formatted to shrink text to fit. `shrinkfit` may not be specified with `txtwrap`.

`merge` tells Excel to merge cells in the specified cell range. `merge` may be combined with `left`, `right`, `hcenter`, `top`, `bottom`, and `vcenter` to format the merged cell. Merging cells that contain data in each cell will result in the upper-leftmost data being kept.

Once you have merged cells, you can refer to the merged cell by using any single cell from the specified *cellrange*. For example, if you specified a *cellrange* of `A1:B2`, you could refer to the merged cell using `A1`, `B1`, `A2`, or `B2`.

`unmerge` tells Excel to unmerge previously merged cells. When using `unmerge`, you only need to use a single cell from the merged cell in the previously specified *cellrange*.

### Font

`font([fontname] [, size [, color]])` sets the font, font size, and font color for each cell in a cell range. The font size and font color may be specified individually without specifying *fontname*. Use `font("", size)` to specify font size only. Use `font("", "", color)` to specify font color only. For both cases, the default font will be used. If `font()` is not specified, the Excel defaults are preserved.

*fontname* may be any valid Excel font. If *fontname* includes spaces, then it must be enclosed in double quotes. What constitutes a valid Excel font is determined by the version of Excel that is installed on the user's computer.

*size* is a numeric value that represents any valid Excel font size. The default is 12.

*color* may be a valid RGB value in the form `"### ##"` or may be one of the colors listed in the table of colors in the [Appendix in \[RPT\] putexcel](#). If no *color* is specified, then Excel workbook defaults are used.

`italic` and `noitalic` specify whether to italicize or unitalicize the text in a cell or range of cells. The default is for text to be unitalicized. `noitalic` has an effect only if the cell or cells were previously italicized.

`bold` and `nobold` specify whether to bold or unbold the text in a cell or range of cells. The default is for text to be unbold. `nobold` has an effect only if the cell or cells were previously formatted as bold.

`underline` and `nounderline` specify whether to underline the text or remove the underline from the text in a cell or range of cells. The default is for text not to be underlined. `nounderline` has an effect only if the cell or cells previously contained underlined text.

`strikeout` and `nostrikeout` specify whether to strikeout the text or remove the strikeout from the text in a cell or range of cells. The default is for text not to have a strikeout mark. `nostrikeout` has an effect only if the cell or cells previously had a strikeout mark.

`script(sub|super|none)` changes the script style of the cell. `script(sub)` makes all text in a cell or range of cells a subscript. `script(super)` makes all text in a cell or range of cells a superscript. `script(none)` removes all subscript or superscript formatting from a cell or range of cells. Specifying `script(none)` has an effect only if the cell or cells were previously formatted as subscript or superscript.



## Border

`border`(*border* [ , *style* [ , *color* ] ] ) sets the cell border, style, and color for a cell or range of cells.

*border* may be `all`, `left`, `right`, `top`, or `bottom`.

*style* is a keyword specifying the look of the border. The most common styles are `thin`, `medium`, `thick`, and `double`. The default is `thin`. For a complete list of border styles, see the [Appendix in \[RPT\] putexcel](#). To remove an existing border, specify `none` as the *style*.

*color* may be a valid RGB value in the form "`### ## #`" or may be one of the colors listed in the table of colors in the [Appendix in \[RPT\] putexcel](#). If no *color* is specified, then Excel workbook defaults are used.

`dborder`(*direction* [ , *style* [ , *color* ] ] ) sets the cell diagonal border direction, style, and color for a cell or range of cells.

*direction* may be `down`, `up`, or `both`. `down` draws a line from the upper-left corner of the cell to the lower-right corner of the cell or, for a range of cells, from the upper-left corner of *ul\_cell* to the lower-right corner of *lr\_cell*. `up` draws a line from the lower-left corner of the cell to the upper-right corner of the cell or, for a range of cells, from the lower-left corner of the area defined by *ul\_cell*:*lr\_cell* to the upper-right corner.

*style* is a keyword specifying the look of the border. The most common styles are `thin`, `medium`, `thick`, and `double`. The default is `thin`. For a complete list of border styles, see the [Appendix in \[RPT\] putexcel](#). To remove an existing border, specify `none` as the *style*.

*color* may be a valid RGB value in the form "`### ## #`" or may be one of the colors listed in the table of colors in the [Appendix in \[RPT\] putexcel](#). If no *color* is specified, then Excel workbook defaults are used.

## Fill

`fpattern`(*pattern* [ , *fgcolor* [ , *bgcolor* ] ] ) sets the fill pattern, foreground color, and background color for a cell or range of cells.

*pattern* is a keyword specifying the fill pattern. The most common fill patterns are `solid` for a solid color (determined by *fgcolor*), `gray25` for 25% gray scale, `gray50` for 50% gray scale, and `gray75` for 75% gray scale. A complete list of fill patterns is shown in the [Appendix of \[RPT\] putexcel](#). To remove an existing fill pattern from the cell or cells, specify `none` as the *pattern*.

*fgcolor* specifies the foreground color. The default foreground color is `black`. *fgcolor* may be a valid RGB value in the form "`### ## #`" or may be any of the colors listed in the table of colors in the [Appendix in \[RPT\] putexcel](#).

*bgcolor* specifies the background color. *bgcolor* may be a valid RGB value in the form "`### ## #`" or may be any of the colors listed in the table of colors in the [Appendix in \[RPT\] putexcel](#). If no *bgcolor* is specified, then Excel workbook defaults are used.

## Remarks and examples

[stata.com](http://stata.com)

If you have not already read [Remarks and examples](#) in [\[RPT\] putexcel](#), please do so now. The examples here build on the examples shown there.

Remarks are presented under the following headings:

[Writing expressions and formatting cells](#)

[Using formulas](#)

[Exporting estimation results](#)

## Writing expressions and formatting cells

Before we can write to an Excel workbook using `putexcel`, we need to tell Stata what the destination is. We do this using the `putexcel set` command. For the next several examples, we will use an Excel file named `myresults2.xlsx`. We will begin with a sheet named `Descriptive`.

```
. putexcel set myresults2.xlsx, sheet(Descriptive)
```

If we had not specified the sheet name, `putexcel` would have defaulted to using the first sheet in the workbook.

### ► Example 1: Write multiple expressions and format cells simultaneously

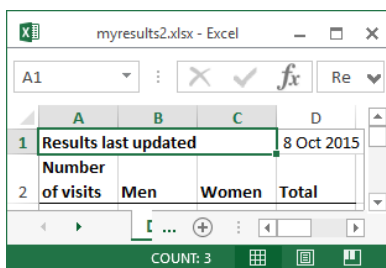
Suppose we want to write the same headers and format them as shown in [example 2](#) and [example 3](#) of [\[RPT\] putexcel](#). Using the advanced syntax of `putexcel`, we can do this with a single `putexcel` command.

```
. putexcel A1="Variable" B1="Men" C1="Women", bold border(bottom)
file myresults2.xlsx saved
```

Rather than beginning with the table headers, we could improve record keeping by including the date we generated the results. To do this, we can use the system parameter, or c-class return value, `c(current_date)`, which will add a string date in the format `dd Mon yyyy`; see [\[P\] creturn](#). We also merge cells A1 and B1 for aesthetic reasons. Notice that we type `"'c(current_date)'"` instead of `"c(current_date)"`. The `'` indicate macro substitution; see [\[P\] macro](#). We specify `replace` as a suboption to `sheet()` in a new `putexcel set` command to overwrite our previous output. Because the workbook `myresults2.xlsx` already exists, we also need to `replace`, or `modify`, the workbook.

```
. putexcel set myresults2.xlsx, replace sheet(Descriptive, replace)
note: file will be replaced when the first putexcel command is issued.
. putexcel A1 = "Results last updated" D1 = "'c(current_date)'"
file myresults2.xlsx saved
. putexcel (A1:C1), merge
file myresults2.xlsx saved
. putexcel A2="Variable" B2="Men" C2="Women", bold border(bottom)
file myresults2.xlsx saved
```

The above commands give an Excel file with a header row that looks like this:



## Using formulas

Writing Excel formulas is useful when you want to perform calculations using the output of Stata commands. Formulas are passed verbatim to Excel, so you must use the correct Excel function (not Stata function) for what you want to accomplish. You may find it helpful to experiment in Excel first and then copy the formula over to your [do-file](#) to keep a record of your work and in case you need to run your analysis again later.

### ► Example 2: Adding a total row and total column

In [example 4](#) of [\[RPT\] putexcel](#), we obtained the number of females and number of males in the `website` dataset and wrote these out to Excel. Suppose instead that we want the number of males and females at each number of visits and the total number of visits.

We can change the heading for the first column from “Variable” to “Number of visits” and add a column for “Total”. We specify the `txtwrap` option with our `putexcel` command so that the long text “Number of visits” wraps within cell A2.

We use the `matcell()` option with `tabulate` to save the cell frequencies to a matrix and the `matrow()` option to save the row values from the table.

```
. use https://www.stata-press.com/data/r18/website
(Visits to website)
. putexcel A2="Number of visits" D2="Total", txtwrap bold border(bottom)
file myresults2.xlsx saved
. tabulate visits female, matrow(nvisits) matcell(freq)
```

Visits to website	Female		Total
	0	1	
1	18	17	35
2	58	42	100
3	32	46	78
4	35	25	60
5	24	40	64
6	21	17	38
7	21	21	42
8	12	7	19
9	8	10	18
10	8	5	13
11	3	5	8
12	4	1	5
13	2	2	4
14	4	0	4
15	1	1	2
16	1	1	2
18	1	0	1
20	2	1	3
22	1	0	1
23	0	1	1
27	1	0	1
51	0	1	1
Total	257	243	500

```
. putexcel A3=matrix(nvisits) B3=matrix(freq)
file myresults2.xlsx saved
```

Totals, however, are not saved. By using formulas in Excel, we can add the row and column totals. Our results will begin on row 3, so we will want to begin calculating row totals here. For example, in Excel the row total for males (column B) and females (column C) in row 3 is  $D3=B3+C3$ . The fastest way to write this formula multiple times for each row is to use `forvalues`; see [\[P\] forvalues](#).

```
. forvalues i=3/24 {
  2.   putexcel D'i'=formula(B'i'+C'i')
  3. }
file myresults2.xlsx saved
      (output omitted)
```

We also want column totals. For this, it is easier to use Excel's `SUM()` function. We supply this as the `formula()` output type and add a border above the total row to set it apart from the rest of the table.

```
. putexcel A25="Total" B25=formula(SUM(B3:B24)) C25=formula(SUM(C3:C24))
> D25=formula(SUM(D3:D24)), bold border(top)
file myresults2.xlsx saved
```

Note that while this example demonstrates how you can easily take advantage of Excel's formulas, you can add results from a tabulation that includes totals in a simpler method. See [\[R\] table twoway](#) to learn about creating tabulations that store results in a collection. With results from `table`, you can use `putexcel`'s `collect` output type to include the full table in an Excel document. See [example 7](#) and [example 8](#) in [\[RPT\] putexcel](#) for examples of `putexcel`'s `collect` output type.



## Exporting estimation results

We start by using `putexcel set` again to create a new worksheet for our regression results, modifying our existing workbook.

```
. putexcel set myresults2.xlsx, sheet(Estimation) modify
```

### ► Example 3: Export point estimates and formatted confidence intervals

We continue from [example 6](#) of [\[RPT\] putexcel](#), where we gave the coefficients the title "Coef.". Here we add a column named "C.I." for the confidence interval, which we center in cells C1 and D1.

```
. putexcel B1="Coef." C1="C.I."
file myresults2.xlsx saved

. putexcel (C1:D1), merge hcenter
file myresults2.xlsx saved
```

We fit the same model that we did in [example 6](#) of [\[RPT\] putexcel](#). We copy the `r-class` return `r(table)`, which contains the values that were returned by the command in the estimation results table, into a new matrix named `table`.

```

. quietly regress visits ad female time
. matrix table = r(table)
. matrix list table
table[9,4]
      ad      female      time      _cons
b     .79961794  -.04679974  .82961995  .69243389
se    .05165911  .2096816   .04366007  .20079144
t     15.47874   -.22319432  19.001801  3.448523
pvalue 2.235e-44  .82347616  6.794e-61  .00061166
ll    .69812028  -.45877341  .74383847  .29792725
ul    .90111561  .36517393  .91540143  1.0869405
df           496           496           496           496
crit  1.9647583  1.9647583  1.9647583  1.9647583
eform      0           0           0           0

```

We then select row 1 for the coefficients (b), row 5 for the lower limit of the confidence interval (ll), and row 6 for the upper limit of the confidence interval (ul) into separate vectors. We take the transpose to ensure that our results for each variable are in a row rather than in a column when we write them out. We specify `nformat(number_d2)` for the coefficients, but we will specify a custom format for the confidence interval. We also add the `rownames` option so that the variable names are written out along with the coefficient estimates.

```

. matrix b = table[1, 1...]'
. matrix ll = table[5, 1...]'
. matrix ul = table[6, 1...]'
. putexcel A2=matrix(b), rownames nformat(number_d2)
file myresults2.xlsx saved

```

We want our confidence interval to be displayed as (0.74 to 0.92), taking `time`, for example. That is, the numbers should have a 0 before the decimal and be rounded to two decimal places. The lower and upper limits should be separated by the word “to”, and the confidence interval should be enclosed in parentheses.

We specify our format in two steps. For the lower limit, we ensure that the negative sign will display inside the parentheses by specifying separate formats for positive and negative numbers. The code for positive numbers is specified before the semicolon. The code for negative numbers is specified after the semicolon. We add the word “to” so that it is printed between the lower and upper limit values. For the upper limit, the negative sign will display in front of the number by default, so we only have to specify one format.

```

. putexcel C2=matrix(ll), nformat("(0.00 to;(-0.00 to)") right
file myresults2.xlsx saved
. putexcel D2=matrix(ul), nformat("0.00)") left
file myresults2.xlsx saved

```

The above commands give a table that looks like this:

	Coef.	C.I.
ad	0.80	(0.70 to 0.90)
female	-0.05	(-0.46 to 0.37)
time	0.83	(0.74 to 0.92)
_cons	0.69	(0.30 to 1.09)

This example demonstrates how to use `putexcel`'s formatting options in combination with specifying the output that is to appear in the cells. However, if your goal is to create a regression table and customize the format of the confidence interval, the `collect` and `table` commands provide another, sometimes more direct, way to do this. See [example 7](#) in [\[RPT\] putexcel](#) for an example of creating a regression table with customized confidence intervals using `collect` and exporting it using `putexcel`'s `collect` output type.

◀

#### ► Example 4: Export SEM estimates and path diagram

Continuing [example 3](#), suppose we also fit a corresponding linear regression model using `sem` and want to export the coefficient estimates, confidence intervals, and the path diagram. We use the same general approach as in [example 3](#) but with some modification. First, we specify the range of columns from `table` to be 1 through 4 when we create our `b`, `ll`, and `ul` vectors to exclude the variance.

```
. quietly sem (visits <- ad female time)
. matrix table = r(table)
. matrix list table
table[9,5]
      visits:      visits:      visits:      visits:      var(e.vis~):
      ad          female        time         _cons       _cons
b      .79961794  -.04679974  .82961995   .69243389   5.4458576
se      .05145206  .20884119  .04348508   .19998666   .34442628
z      15.541029  -.22409249  19.078268   3.4624004   .b
pvalue  1.830e-54   .82268533   3.827e-81   .00053538   .b
ll      .69877376  -.45612096  .74439077   .30046724   4.810958
ul      .90046212  .36252147  .91484914   1.0844005   6.1645445
df      .          .          .          .          .
crit    1.959964   1.959964   1.959964   1.959964   1.959964
eform   0          0          0          0          0
. matrix b = table[1, 1..4]'
. matrix ll = table[5, 1..4]'
. matrix ul = table[6, 1..4]'
```

If we wanted that term as well, we could select it separately. Because we do not need to specify row names again, we also specify the `ul_cell` for `b` in reference to its values, not the labels this time. We use the same numeric format for `ll` and `ul`.

```
. putexcel E2=matrix(b), nformat(number_d2)
file myresults2.xlsx saved

. putexcel F2=matrix(l1), nformat("(0.00 to;(-0.00 to)") right
file myresults2.xlsx saved

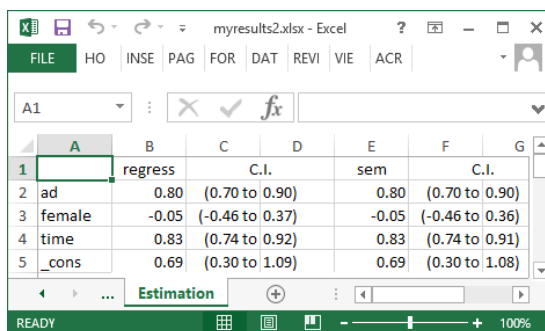
. putexcel G2=matrix(u1), nformat("0.00)") left
file myresults2.xlsx saved
```

Because we are adding estimation results, we change “Coef.” to “regress” and add a column heading for “sem”. As before, we include a merged column heading for the confidence interval. We center all column headings by using the `hcenter` option.

```
. putexcel B1="regress" E1="sem" F1="C.I.", hcenter
file myresults2.xlsx saved

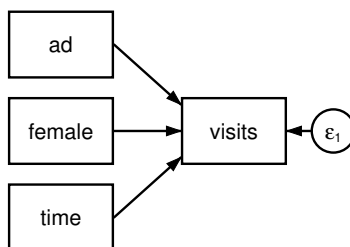
. putexcel (F1:G1), merge
file myresults2.xlsx saved
```

Our table from [example 3](#) now includes additional columns for our `sem` results:



	regress	C.I.	sem	C.I.
ad	0.80	(0.70 to 0.90)	0.80	(0.70 to 0.90)
female	-0.05	(-0.46 to 0.37)	-0.05	(-0.46 to 0.36)
time	0.83	(0.74 to 0.92)	0.83	(0.74 to 0.91)
_cons	0.69	(0.30 to 1.09)	0.69	(0.30 to 1.08)

We might also want to export the path diagram for our model so that we can view it while looking at our results.



The SEM Builder will allow you to save your path diagram as a PNG file, in addition to other file types; PNGs can be exported to Excel. We name our path diagram `visits_sem` and save it as a PNG, and we then output the file to Excel with the `image()` output type.

```
. putexcel B6 = image(visits_sem.png)
file myresults2.xlsx saved
```

This adds the path diagram with the upper-left corner aligned in the upper-left corner of cell B6.

## □ Technical note

See the technical notes [Excel data size limits](#) and [Dates and times](#) in [D] **import excel**.

**References**

- Crow, K. 2013. Export tables to Excel. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2013/09/25/export-tables-to-excel/>.
- . 2018. Export tabulation results to Excel—Update. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2018/06/07/export-tabulation-results-to-excel-update/>.
- Gallup, J. L. 2012. A new system for formatting estimation tables. *Stata Journal* 12: 3–28.
- Huber, C. 2017. Creating Excel tables with putexcel, part 3: Writing custom reports for arbitrary variables. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2017/04/06/creating-excel-tables-with-putexcel-part-3-writing-custom-reports-for-arbitrary-variables/>.
- Quintó, L. 2012. HTML output in Stata. *Stata Journal* 12: 702–717.

**Also see**

- [RPT] **putexcel** — Export results to an Excel file
- [RPT] **putdocx intro** — Introduction to generating Office Open XML (.docx) files
- [RPT] **putpdf intro** — Introduction to generating PDF files
- [D] **import excel** — Import and export Excel files
- [M-5] **\_docx\*()** — Generate Office Open XML (.docx) file
- [M-5] **Pdf\*()** — Create a PDF file
- [M-5] **xl()** — Excel file I/O class
- [P] **postfile** — Post results in Stata dataset
- [P] **return** — Return stored results
- [R] **table intro** — Introduction to tables of frequencies, summaries, and command results
- [TABLES] **Intro** — Introduction

Stata, Stata Press, and Mata are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow and NetCourseNow are trademarks of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2023 StataCorp LLC, College Station, TX, USA. All rights reserved.



For suggested citations, see the FAQ on [citing Stata documentation](#).