

**putexcel** — Export results to an Excel file
[Description](#)[Remarks and examples](#)[Quick start](#)[Appendix](#)[Menu](#)[Stored results](#)[Syntax](#)[References](#)[Options](#)[Also see](#)

## Description

`putexcel` writes Stata [expressions](#), [matrices](#), tables, images, and [returned results](#) to an Excel file. `putexcel` can also format the cells in the worksheet. This allows you to automate exporting and formatting of, for example, Stata estimation results. 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 a file opened using the command `putexcel set ...`, `open` and saves the file in memory to disk. `putexcel clear` clears the file information set by `putexcel set`. `putexcel describe` displays the file information set by `putexcel set`.

For an advanced syntax to simultaneously write multiple output types, see [\[RPT\] putexcel advanced](#).

## Quick start

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

```
putexcel set myresults
```

As above, but use a new sheet named Estimation Results and replace the existing workbook

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

Write the text “Coefficients” to cell B1

```
putexcel B1 = "Coefficients"
```

Add a table from the current collection to a new sheet named `table2` beginning in cell B2

```
putexcel B2 = collect, sheet(table2)
```

Add variable names and estimated coefficients in the column under “Coefficients” after [regress](#), and format coefficients with two decimal places

```
matrix b = e(b)
putexcel A2 = matrix(b), rownames nformat(number_d2)
```

Format the header row of the table with a bottom border and bold text

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

Add a PNG of a [margins](#) plot saved to disk as `mymargins.png` where the upper-left corner is aligned with the upper-left corner of cell D2

```
marginsplot, name(mymargins)
graph export mymargins.png, name(mymargins)
putexcel D2 = image(mymargins.png)
```

### Menu

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

### Syntax

*Set workbook for export*

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

*Write expression to Excel*

```
putexcel ul_cell = exp [ , expression_options format_options ]
```

*Export Stata matrix to Excel*

```
putexcel ul_cell = matrix(matname) [ , matrix_options format_options ]
```

*Export Stata graph, path diagram, or other image to Excel*

```
putexcel ul_cell = image(filename)
```

*Export returned results to Excel*

```
putexcel ul_cell = returnset [ , colwise overwritefmt ]
```

*Write formula to Excel*

```
putexcel ul_cell = formula(formula) [ , overwritefmt ]
```

*Format cells*

```
putexcel cellrange, overwritefmt format_options
```

*Add the coefficient table from the last estimation command to Excel file*

```
putexcel ul_cell = etable[ (#1 #2 ... #n) ]
```

*Add a table from the current collection to Excel file*

```
putexcel ul_cell = collect[ , collect_options ]
```

*Close and save Excel file*

```
putexcel save
```

*Describe current export settings*

```
putexcel describe
```

Clear current export settings

```
putexcel clear
```

*ul\_cell* is a valid Excel upper-left cell specified using standard Excel notation, for example, A1 or D4. *cellrange* is *ul\_cell* or *ul\_cell:lr\_cell*, where *lr\_cell* is a valid Excel lower-right cell, for example, A1, A1:D1, A1:A4, or A1:D4.

<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 <code>Sheet1</code>

<i>expression_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

<i>matrix_options</i>	Description
Main	
<code>overwritefmt</code>	overwrite existing cell formatting when exporting new content
<code>names</code>	also write row names and column names for matrix <i>matname</i> ; may not be combined with <code>rownames</code> or <code>colnames</code>
<code>rownames</code>	also write matrix row names for matrix <i>matname</i> ; may not be combined with <code>names</code> or <code>colnames</code>
<code>colnames</code>	also write matrix column names for matrix <i>matname</i> ; may not be combined with <code>names</code> or <code>rownames</code>

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

<i>collect_options</i>	Description
<code>name(cname)</code>	use collection <i>cname</i>
<code>noisily</code>	show <code>putexcel</code> commands used for exporting
<code>dofile(filename [, replace])</code>	save <code>putexcel</code> commands used for exporting to the specified do-file

`putexcel ul_cell = collect` and `collect_options` do not appear in the dialog box.

## 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. It is intended primarily for use by programmers and by those who intend to do further processing of their exported results in Excel. *returnset* may be any one of the following:

<i>returnset</i>	
<code>escalars</code>	<code>escalarmnames</code>
<code>rscalars</code>	<code>rscalarnames</code>
<code>emacros</code>	<code>emacronames</code>
<code>rmacros</code>	<code>rmacronames</code>
<code>ematrices</code>	<code>ematrixnames</code>
<code>rmatrices</code>	<code>rmatrixnames</code>
<code>e*</code>	<code>enames</code>
<code>r*</code>	<code>rnames</code>

`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.

`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*, ..., *tablename<sub>n</sub>*. 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*.

### Main

`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 the output type.

`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 the output type.

`asdatenum` 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 the output type.

`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 the output type.

`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 the output type.

`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 the output type.

`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 the output type.

`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 the output type.

### 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 one of the colors listed in the table of colors in the [Appendix](#) or may be a valid RGB value in the form "### ### ###". 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](#). To remove an existing border, specify `none` as the *style*.

*color* may be one of the colors listed in the table of colors in the [Appendix](#) or may be a valid RGB value in the form "`### ##`". 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](#). To remove an existing border, specify `none` as the *style*.

*color* may be one of the colors listed in the table of colors in the [Appendix](#) or may be a valid RGB value in the form "`### ##`". 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](#). 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 any of the colors listed in the table of colors in the [Appendix](#) or may be a valid RGB value in the form "`### ##`".

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

The following options are available when exporting a collection:

`name(cname)` specifies a collection *cname* from which to export the table. By default, the table is taken from the current collection.

noisily specifies that the `putexcel` commands used to export to the workbook be displayed.

`dofile(filename[, replace])` specifies that `putexcel` save to *filename* the commands used to export to the workbook. If *filename* already exists, it can be overwritten by specifying `replace`. If *filename* is specified without an extension, `.do` is assumed.

## Remarks and examples

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

Remarks are presented under the following headings:

- [Introduction](#)
- [Writing expressions and formatting cells](#)
- [Exporting summary statistics to Excel](#)
- [Export estimation results](#)
- [Export a table from a collection](#)
- [Export graphs and other images](#)

## Introduction

The `putexcel` command is a means of directly controlling the layout of an Excel file. As such, `putexcel` is designed to mimic the options and functionality of Excel, and many options of `putexcel` are simply pass-through arguments to Excel itself. In what follows, we provide documentation of how to use the `putexcel` command. However, for many options, such as the specification of valid numeric formats, font names, and font sizes, users are encouraged to also consult the help for their specific version of Excel.

`putexcel` may be used with Excel 1997/2003 (`.xls`) and Excel 2007/2010 and newer (`.xlsx`). `putexcel` looks at the file extension `.xls` or `.xlsx` to determine which Excel format to write. It is supported on Windows, Mac, and Linux.

`putexcel` also has an advanced syntax that allows you to write multiple types of output to different cells or cell ranges at a time; see [\[RPT\] putexcel advanced](#).

### ▷ Example 1: Setting the workbook and sheet

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 `myresults.xlsx` and a sheet named `Descriptive`.

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

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

◀

## Writing expressions and formatting cells

Although there are many uses for `putexcel`, one of the primary reasons to use the command is to keep track of the results of analyses in a single location for later use. The next several examples show how to export results to a single location for easy sharing and to create formatted tables that can be placed in a paper or poster.

Suppose we are analyzing data about the number of times young adults visited a news website. These data are contained in `website.dta` and described in detail in [example 1](#) of [\[R\] ivpoisson](#).

```
. use https://www.stata-press.com/data/r17/website
(Visits to website)
. describe
Contains data from https://www.stata-press.com/data/r17/website.dta
Observations:      500           Visits to website
Variables:         6             11 Feb 2020 14:45
```

Variable name	Storage type	Display format	Value label	Variable label
visits	byte	%8.0g		Visits to website
female	byte	%8.0g		Female
ad	byte	%8.0g		Advertisements
time	double	%10.0g		Time on internet (hrs.)
phone	double	%10.0g		Time on phone (hrs.)
frfam	double	%10.0g		Time with friends and out of town family (hrs.)

Sorted by:

We want to create a formatted table of summary statistics for men and women.

### ▷ Example 2: Write expression to cell

We start by writing column headers to the first row of our worksheet.

```
. putexcel A1 = "Variable"
file myresults.xlsx saved
. putexcel B1 = "Men"
file myresults.xlsx saved
. putexcel C1 = "Women"
file myresults.xlsx saved
```

Each of these commands opens the Excel workbook, writes the text to the specified cell, and then closes the workbook. We also use expressions to write out specific results (see [example 4](#)).



### ▷ Example 3: Format a range of cells

We may also want to make the column headings bold and add a border underneath. We can format a range of cells by specifying a cell range and the appropriate formatting options.

```
. putexcel A1:C1, bold border(bottom)
file myresults.xlsx saved
```

We also could have specified the `bold` and `border()` options each time we exported the column heading in [example 2](#).

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

Whether you apply formatting at the time you write to Excel or for a range will likely depend on the number of options you have and the number of cells affected by common formatting options.



## □ Technical note

When formatting many cells within a workbook, make sure to write cell formats efficiently using cell ranges. If you do not, you can overload the Excel workbook with too many formats, causing the Excel workbook to become large and, in extreme cases, the cell formatting to stop working. Take the following example:

```
putexcel set test.xlsx
putexcel A1 = "Sex"
putexcel A1, bold
putexcel A1, italic
putexcel A1, font("Arial", 16, "black")
putexcel B1 = "Age"
putexcel B1, bold
putexcel B1, italic
putexcel B1, font("Arial", 16, "black")
putexcel C1 = "Race"
putexcel C1, bold
putexcel C1, italic
putexcel C1, font("Arial", 16, "black")
```

Creating the Excel workbook with the commands above will create nine format entries for the worksheet. A more efficient way to write the cell formats is

```
putexcel set test.xlsx
putexcel A1 = "Sex"
putexcel B1 = "Age"
putexcel C1 = "Race"
putexcel A1:C1, bold italic font("Arial", 16, "black")
```

Now the Excel workbook contains only one format entry for the worksheet. Although the nine format entries created by the previous set of `putexcel` commands are unlikely to cause a problem, your Excel workbook will become large if you format hundreds of cells across multiple worksheets in this manner. Regardless of the number of cells you are formatting, you should always try to write cell formats efficiently to reduce your Excel workbook size.

□

## Exporting summary statistics to Excel

We can use `putexcel` to write summary statistics to an Excel worksheet. Summary statistics can be obtained in two ways. The `table` command computes summary statistics, creates a table of the results, and creates a collection that can be inserted into an Excel file. Alternatively, summary statistics can be computed using another command such as `summarize` or `tabstat`. After running these commands, the available summary statistics are determined by `returned results`. The returned results are documented in the *Stored results* section of the command's manual entry or help. You can also see what is returned by typing `return list` or `ereturn list`.

### ▷ Example 4: Export selected statistics

To export a specific statistic from returned results, we use the expression output type. For example, we might want a table that has the number of observations for men and women followed by means for each variable, which we show in [example 5](#).

We can obtain counts in many ways in Stata. Here we use the `summarize` command and restrict the command to males (`female==0`). We use `visits`, but any continuous variable without missing values would have worked just as well. `r(N)` stores the number of observations.

```
. summarize visits if female==0
```

Variable	Obs	Mean	Std. dev.	Min	Max
visits	257	5.105058	3.893185	1	27

```
. putexcel B2 = `r(N)'  
file myresults.xlsx saved
```

A more direct way to obtain the number of observations is with the `count` command.

```
. count if female==1  
243  
. putexcel C2 = `r(N)'  
file myresults.xlsx saved
```

Notice that we typed `'r(N)'` instead of `"r(N)"`. The `'` tell Stata to fill in the numeric value associated with `r(N)` instead of exporting the text, known as macro substitution; see [P] [macro](#). If we wanted to treat the contents of `r(N)`, or any other return value, like a string, we could have typed `"`r(N)'"`.

◀

## ▶ Example 5: Export frequency tables

You can use `putexcel` in Stata to create tables in Excel by using the `matrix()` output type. Suppose we want to create a table of means for each variable for each value of `female`. We can use `tabstat` with the `save` option and then check the return values with `return list` to determine what values to output.

```
. tabstat visits ad time phone frfam, by(female) save
```

Summary statistics: Mean  
Group variable: female (Female)

female	visits	ad	time	phone	frfam
0	5.105058	2.175097	3.222412	3.164086	3.65428
1	4.946502	2.098765	3.161276	3.155391	3.676708
Total	5.028	2.138	3.1927	3.15986	3.66518

```
. return list
```

macros:

```
    r(name2) : "1"  
    r(name1) : "0"
```

matrices:

```
    r(Stat2) : 1 x 5  
    r(Stat1) : 1 x 5  
    r(StatTotal) : 1 x 5
```

First, we transpose the row vectors `r(Stat1)` and `r(Stat2)`, which contain the means, so that the values are written in a column under each heading. We want the variable names to be included, so for males, which is the first matrix we output, we include the `rownames` option. Because we want the values in B3 and the row names of the matrix in A3, we specify A3. We use `nformat(number_d2)` to format the means with two decimal places. For females, we do not need to specify the names again, so we just specify the cell where we want the data to be written.

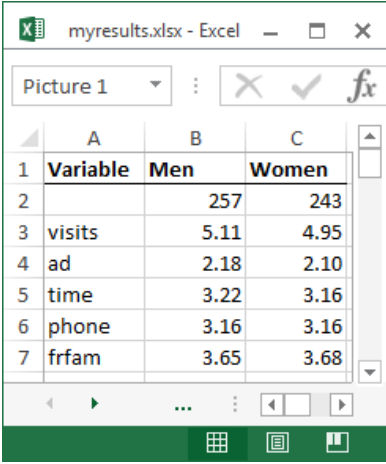
```
. qui tabstat visits ad time phone frfam, by(female) save  
. matrix male = r(Stat1)'  
. matrix female = r(Stat2)'
```

```

. putexcel A3 = matrix(male), rownames nformat(number_d2)
file myresults.xlsx saved
. putexcel C3 = matrix(female), nformat(number_d2)
file myresults.xlsx saved

```

The above commands give a final table of frequencies and means that looks like this:



	A	B	C
1	Variable	Men	Women
2		257	243
3	visits	5.11	4.95
4	ad	2.18	2.10
5	time	3.22	3.16
6	phone	3.16	3.16
7	frfam	3.65	3.68

In building this table, we have demonstrated how to add text, expressions, and matrices to an Excel file individually. Alternatively, we can compute the summary statistics with a single `table` command and format the results using `collect`. We demonstrate this approach in [example 8](#).

◀

## Export estimation results

A similar approach to that used in [example 5](#) can be used to export estimation results. Suppose we want to fit a linear regression model of `visits` as a function of `ad`, `female`, and `time` using `regress`, and we want to export formatted results.

```
. regress visits ad female time
```

Source	SS	df	MS	Number of obs	=	500
Model	5710.6792	3	1903.55973	F(3, 496)	=	346.75
Residual	2722.9288	496	5.4897758	Prob > F	=	0.0000
Total	8433.608	499	16.901018	R-squared	=	0.6771
				Adj R-squared	=	0.6752
				Root MSE	=	2.343

visits	Coefficient	Std. err.	t	P> t	[95% conf. interval]
ad	.7996179	.0516591	15.48	0.000	.6981203 .9011156
female	-.0467997	.2096816	-0.22	0.823	-.4587734 .3651739
time	.82962	.0436601	19.00	0.000	.7438385 .9154014
_cons	.6924339	.2007914	3.45	0.001	.2979273 1.086941

## ► Example 6: Export point estimates and sample size

We start by using `putexcel set` again to create a new worksheet for our regression results. Because we are working with the same workbook that we specified previously, we specify the `modify` option.

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

We want to give our coefficients the title “Coef.” in the table we create, so we use an expression to write this to cell B1. We create a new matrix for our coefficients as the transpose of the values of e-class matrix `e(b)`. We use column A row 2 as the starting location for the matrix row labels, and we use column B row 2 as the starting location for the coefficients; to do this, we only need to specify the upper-left cell (A2) and the `rownames` option. To make our table more readable, we format the coefficient estimates with two decimal places by using `nformat()` and add a bottom border under the estimates.

```
. putexcel B1 = "Coef."
file myresults.xlsx saved
. matrix b = e(b)'
. putexcel A2 = matrix(b), rownames nformat(number_d2)
file myresults.xlsx saved
. putexcel A5:B5, border(bottom)
file myresults.xlsx saved
```

We then add a right-aligned and italicized “N=” in column A row 6 and the sample size from e-class scalar `e(N)` in column B row 6. We use a medium border under this row instead of the default thin border to indicate that the table is complete.

```
. putexcel A6 = "N=", italic right border(bottom, medium)
file myresults.xlsx saved
. putexcel B6 = matrix(e(N)), nformat(number_sep) border(bottom, medium)
file myresults.xlsx saved
```

The above commands create a table that looks like this:

	A	B	C
1		Coef.	
2	ad	0.80	
3	female	-0.05	
4	time	0.83	
5	_cons	0.69	
6	N=	500	
7			

If you are going to export more-complex tables or many objects, you can use the advanced syntax of `putexcel`; see [\[RPT\] putexcel advanced](#). You might also create a customized table with the `collect` or `table` command, which we demonstrate in the next section.

## Export a table from a collection

The `collect` suite of commands allows you to collect results from Stata commands and create customized tables with these results. You can arrange the results in different ways, modify the labels for the table, format numbers within the table, and make other style changes. With `putexcel`, you can easily export these customized tables to your Excel files, as we demonstrate below. In these examples, we assume some familiarity with the `table` and `collect` commands. We recommend that you see [\[R\] table intro](#) for information on `table` and [\[TABLES\] Intro](#) for more information on `collect` to learn more about creating and customizing tables before incorporating them into your Excel file.

### ► Example 7: Export a customized table with multiple results

Suppose that we now want to compare the results from the linear regression we fit in the previous example with the results from a model with an additional covariate. We will collect the results from each model and then create a customized table with both sets of results; this table will replace the contents on the `Estimation` sheet:

```
. putexcel set myresults.xlsx, sheet(Estimation, replace) modify
```

We begin by clearing out any style specifications for the collection. When you collect results from a Stata command, the collection will have a default style. For example, a border will be added between the row headers and the results. For this table, we would like to clear all of those specifications and begin with an empty style. Then we prefix each estimation command with `collect:`, and we suppress the output with `quietly`. Once we have collected the results, we create our table by specifying the layout with `collect layout`. We set the rows to correspond to the variable names, which are contained in the `colname` dimension. The columns will be determined by the model, accessed through the `cmdset` dimension, and by the statistics, accessed through the `result` dimension. Many results are collected from each regression model, but suppose that we only want to report the coefficients (`_r_b`) and confidence intervals (`_r_ci`).

```
. collect style clear
. quietly: collect: regress visits ad female time
. quietly: collect: regress visits ad female time phone
. quietly: collect layout (colname) (cmdset#result[_r_b _r_ci])
```

If we do not make any style specifications, we will have the title for each dimension displayed, resulting in a very wide table. So we used the `quietly` prefix to suppress the preview that `collect layout` displays by default. Below, we hide the title for all dimensions in the table, format all cells with numeric content to only display three digits after the decimal, and specify that upper and lower bounds of confidence intervals be separated by a comma and enclosed in square brackets. We also request that row headers display the variable names, the values of the dimension `colname`, rather than the variable labels. Then we report the current layout:



```

. collect style header, title(hide)
. collect style cell, nformat(%7.3f)
. collect style cell result[_r_ci], cidelimiter(,) sformat("%s")
. collect style header colname, level(value)
. collect layout
Collection: default
  Rows: colname
  Columns: cmdset#result[_r_b _r_ci]
Table 1: 5 x 4

```

	1		2	
	Coefficient	95% CI	Coefficient	95% CI
ad	0.800	[0.698, 0.901]	0.795	[0.679, 0.911]
female	-0.047	[-0.459, 0.365]	-0.047	[-0.460, 0.365]
time	0.830	[0.744, 0.915]	0.827	[0.737, 0.917]
phone			0.010	[-0.111, 0.132]
_cons	0.692	[0.298, 1.087]	0.677	[0.246, 1.109]

We could export this table to our Excel file now, but first let's clean up the labels. The values 1 and 2 represent the order of the `collect` commands we issued, and they are levels of the dimension `cmdset`. Below, we change the 1 to `Reduced` and the 2 to `Full`. Also, notice that the values are repeated for each column that corresponds to that model, so we specify that these duplicate column headers be placed in the center of all the cells they span and that they be centered horizontally. Additionally, we change the label for the coefficients to simply `B`:

```

. collect label levels cmdset 1 "Reduced" 2 "Full", modify
. collect style column, dups(center)
. collect label levels result _r_b "B", modify
. collect layout
Collection: default
  Rows: colname
  Columns: cmdset#result[_r_b _r_ci]
Table 1: 5 x 4

```

	Reduced		Full	
	B	95% CI	B	95% CI
ad	0.800	[0.698, 0.901]	0.795	[0.679, 0.911]
female	-0.047	[-0.459, 0.365]	-0.047	[-0.460, 0.365]
time	0.830	[0.744, 0.915]	0.827	[0.737, 0.917]
phone			0.010	[-0.111, 0.132]
_cons	0.692	[0.298, 1.087]	0.677	[0.246, 1.109]

So that all our contents are aligned, below we also center-align our results. The last thing we will do is add some borders to the table. The dimension `border_block` divides the table into four blocks: column-header, item (which contains the results), corner (top left area), and the row-header. We add borders to the top and bottom of the `item` and `row-header` blocks.

```

. collect style cell result, halign(center)
. collect style cell border_block[item], border(top) border(bottom)
. collect style cell border_block[row-header], border(top) border(bottom)

```

Now that we are done customizing our table, we export it to our Excel file with the upper-left cell aligned with the upper-left corner of cell A1:

```

. putexcel A1 = collect
(collection default posted to putexcel)

```

Now the Estimation sheet contains the following:

	A	B	C	D	E	F
1		Reduced		Full		
2		B	95% CI	B	95% CI	
3	ad	0.800	[0.698, 0.901]	0.795	[0.679, 0.911]	
4	female	-0.047	[-0.459, 0.365]	-0.047	[-0.460, 0.365]	
5	time	0.830	[0.744, 0.915]	0.827	[0.737, 0.917]	
6	phone			0.010	[-0.111, 0.132]	
7	_cons	0.692	[0.298, 1.087]	0.677	[0.246, 1.109]	

Exporting the collection only required a single command. However, Stata did a lot of work in the background to export the collection. If you are curious, you can use the `noisily` option to see all the commands that were used to export the collection.

◀

### ► Example 8: Export a customized table of summary statistics

In [example 5](#), we built a table of summary statistics from stored results produced by `summarize`, `count`, and `tabstat`. We could instead compute all of these summary statistics at once using `table`. An advantage of `table` is that it creates a collection with its results. If you are happy with the table, you can simply export the table to Excel using `putexcel`'s `collect` export type. However, you may prefer to further customize the table before export. To do this, you can use the `collect` commands.

To demonstrate, we re-create the table from [example 5](#) using a series of `table` and `collect` commands. To begin, we compute the frequencies and means for males and females. Here we specify that we want statistics (`result`) and variables (`var`) as our row dimensions, while we want gender (`female`) on our columns. We also format our results to include two decimal places for the means.

```
. table (result var) (female), statistic(freq)
> statistic(mean visits ad time phone frfam) nototals
> nformat(%5.2f mean)
```

	Female	
	0	1
Frequency	257	243
Mean		
Visits to website	5.11	4.95
Advertisements	2.18	2.10
Time on internet (hrs.)	3.22	3.16
Time on phone (hrs.)	3.16	3.16
Time with friends and out of town family (hrs.)	3.65	3.68

To customize this table, we first suppress the titles of our `result` dimension to remove the `Frequency` and `Mean` headers. For conciseness, we will show the variable names instead of the variable labels. In addition, we can hide the title of `Female`, and instead we can set level labels of “Men” and “Women” to appear in the column headers. Finally, we will remove all borders.

```

. collect style header, title(hide)
. collect style header var, level(value)
. collect style header result, level(hide)
. collect label levels female 0 "Men" 1 "Women"
. collect style cell, border(,pattern(nil))
. collect preview

```

	Men	Women
	257	243
visits	5.11	4.95
ad	2.18	2.10
time	3.22	3.16
phone	3.16	3.16
frfam	3.65	3.68

To fully re-create [example 5](#), we must add `Variable` to the top left cell and then specify that the names across the top are bold with a border underneath. To demonstrate another workflow, we will export our collection from `table` and then use `putexcel`'s formatting options.

```

. putexcel set myresults.xlsx, sheet(Descriptive, replace) modify
. putexcel A1 = collect
(collection Table posted to putexcel)
. putexcel A1 = "Variable"
file myresults.xlsx saved
. putexcel A1:C1, bold border(bottom)
file myresults.xlsx saved

```

We have once again created

	A	B	C
1	<b>Variable</b>	<b>Men</b>	<b>Women</b>
2		257	243
3	visits	5.11	4.95
4	ad	2.18	2.10
5	time	3.22	3.16
6	phone	3.16	3.16
7	frfam	3.65	3.68

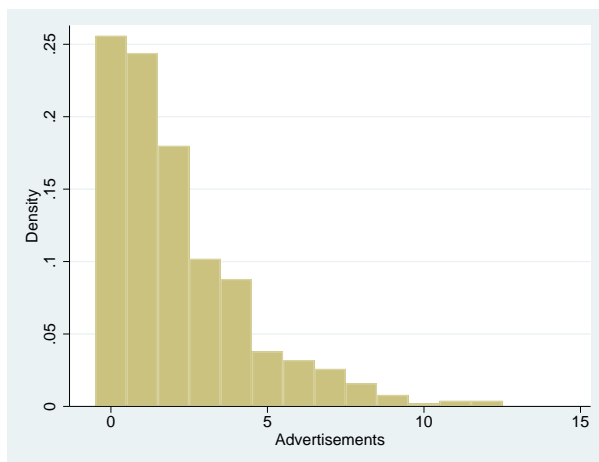
## Export graphs and other images

You can export PNG, JPEG, and other image formats to Excel with `putexcel`. To export a Stata graph, you must first use `graph export` to convert the Stata graph to one of the supported image formats.

### ▷ Example 9: Export Stata graph

We may want to add a histogram about the number of advertisements viewed to our set of descriptive results. We can use the `histogram` command and then `graph export` to create a PNG file of our histogram.

```
. histogram ad, discrete
(start=0, width=1)
. graph export ads.png
file ads.png saved as PNG format
```



Then we output the graph to Excel with the `image()` output type.

```
. putexcel E2 = image(ads.png)
file myresults.xlsx saved
```

This adds the graph with the upper-left corner of the graph aligned in the upper-left corner of cell E2. Graphs are not resized by `putexcel`, but you can change the size with the `graph export` command; see [G-2] [graph export](#).

◀

### □ Technical note

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

□

# Appendix

## Codes for numeric formats

Code	Example
number	1000
number_d2	1000.00
number_sep	100,000
number_sep_d2	100,000.00
number_sep_negbra	(1,000)
number_sep_negbrared	(1,000)
number_d2_sep_negbra	(1,000.00)
number_d2_sep_negbrared	(1,000.00)
currency_negbra	(\$4000)
currency_negbrared	(\$4000)
currency_d2_negbra	(\$4000.00)
currency_d2_negbrared	(\$4000.00)
account	5,000
accountcur	\$ 5,000
account_d2	5,000.00
account_d2_cur	\$ 5,000.00
percent	75%
percent_d2	75.00%
scientific_d2	10.00E+1
fraction_onedig	10 1/2
fraction_twodig	10 23/95
date	3/18/2007
date_d_mon_yy	18-Mar-07
date_d_mon	18-Mar
date_mon_yy	Mar-07
time_hmm_AM	8:30 AM
time_HMMSS_AM	8:30:00 AM
time_HMM	8:30
time_HMMSS	8:30:00
time_MMSS	30:55
time_HOMMSS	20:30:55
time_MMSSO	30:55.0
date_time	3/18/2007 8:30
text	this is text

## Colors

*color*

---

aliceblue	deeppink
antiquewhite	deepskyblue
aqua	dimgray
aquamarine	dodgerblue
azure	firebrick
beige	floralwhite
bisque	forestgreen
black	fuchsia
blanchedalmond	gainsboro
blue	ghostwhite
blueviolet	gold
brown	goldenrod
burlywood	gray
cadetblue	green
chartreuse	greenyellow
chocolate	honeydew
coral	hotpink
cornflowerblue	indianred
cornsilk	indigo
crimson	ivory
cyan	khaki
darkblue	lavender
darkcyan	lavenderblush
darkgoldenrod	lawngreen
darkgray	lemonchiffon
darkgreen	lightblue
darkkhaki	lightcoral
darkmagenta	lightcyan
darkolivegreen	lightgoldenrodyellow
darkorange	lightgray
darkorchid	lightgreen
darkred	lightpink
darksalmon	lightsalmon
darkseagreen	lightseagreen
darkslateblue	lightskyblue
darkslategray	lightslategray
darkturquoise	lightsteelblue
darkviolet	lightyellow

---

*color*, continued

lime	peru
limegreen	pink
linen	plum
magenta	powerblue
maroon	purple
mediumaquamarine	red
mediumblue	rosybrown
mediumorchid	royalblue
mediumpurple	saddlebrown
mediumseagreen	salmon
mediumslateblue	sandybrown
mediumspringgreen	seagreen
mediumturquoise	seashell
mediumvioletred	sienna
midnightblue	silver
mintcream	skyblue
mistyrose	slateblue
moccasin	snow
navajowhite	springgreen
navy	steelblue
oldlace	tan
olive	teal
olivedrab	thistle
orange	tomato
orangered	turquoise
orchid	violet
palegoldenrod	wheat
palegreen	white
paleturquoise	whitesmoke
palevioletred	yellow
papayawhip	yellowgreen
peachpuff	

---

## Border styles

*style*

---

none  
thin  
medium  
dashed  
dotted  
thick  
double  
hair  
medium\_dashed  
dash\_dot  
medium\_dash\_dot  
dash\_dot\_dot  
medium\_dash\_dot\_dot  
slant\_dash\_dot

---

## Background patterns

*pattern*

---

none  
solid  
gray50  
gray75  
gray25  
horstripe  
verstripe  
diagstripe  
revdiagstripe  
diagcrosshatch  
thinhorstripe  
thinverstripe  
thindiastride  
thinrevdiagstripe  
thinhorcrosshatch  
thindiastride  
thickdiagcrosshatch  
gray12p5  
gray6p25

---



## Stored results

putexcel collect stores the following in `s()`:

Scalars

<code>s(collection)</code>	name of collection
<code>s(dofile)</code>	name of the new do-file

## 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/>.
- . 2014. Retaining an Excel cell's format when using putexcel. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2014/02/04/retaining-an-excel-cells-format-when-using-putexcel/>.
- . 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. 2017a. Creating Excel tables with putexcel, part 1: Introduction and formatting. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2017/01/10/creating-excel-tables-with-putexcel-part-1-introduction-and-formatting/>.
- . 2017b. Creating Excel tables with putexcel, part 2: Macro, picture, matrix, and formula expressions. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2017/01/24/creating-excel-tables-with-putexcel-part-2-macro-picture-matrix-and-formula-expressions/>.
- Quintó, L. 2012. HTML output in Stata. *Stata Journal* 12: 702–717.

## Also see

- [RPT] [putexcel advanced](#) — Export results to an Excel file using advanced syntax
- [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
- [R] [table intro](#) — Introduction to tables of frequencies, summaries, and command results
- [TABLES] [Intro](#) — Introduction