# Title

> **putdocx —** Generate Office Open XML (.docx) file

Description | Quick start | Syntax | Options
Remarks and examples | Appendix | References | Also see

## Description

putdocx writes paragraphs, images, and tables to an Office Open XML file (.docx). It may also be used to format each object added. This allows you to automate exporting and formatting of, for example, Stata estimation results and also to generate various reports based on those results. The generated file is compatible with Microsoft Word 2007 and later.

## Quick start

Create a document in memory on which subsequent contents are added
```
putdocx begin
```

Declare a paragraph to be added to the document and center the paragraph
```
putdocx paragraph, halign(center)
```

Append the text "This is paragraph text" to the paragraph declared above and format the text as bold
```
putdocx text ("This is paragraph text"), bold
```

Add a table named tbl1 with three rows and four columns to the document
```
putdocx table tbl1 = (3,4)
```

Set the content of the cell on the first row and second column of the above table as "Cell 2" and align the text to the right
```
putdocx table tbl1(1,2) = ("Cell 2"), halign(right)
```

Add a table named tbl2 with variable names and estimated coefficients after regress, and format coefficients with two decimal places
```
matrix b = e(b)'
putdocx table tbl2 = matrix(b), rownames nformat(%7.2f)
```

Add a PNG image saved as myimg to the document
```
putdocx paragraph
putdocx image myimg.png
```

Save the document in memory to disk as myfile.docx
```
putdocx save myfile.docx
```

Append the contents of filename2.docx and filename3.docx to the end of the contents in filename1.docx
```
putdocx append filename1 filename2 filename3
```

As above, but save the merged contents in a file named filename4.docx
```
putdocx append filename1 filename2 filename3, saving(filename4)
```

## Syntax

*Create document for export*

> putdocx begin [ , pagesize(*psize*) landscape font(*fspec*) ]

*Add paragraph to document*

> putdocx paragraph [ , *paragraph_options* ]

*Add text to paragraph*

> putdocx text (*exp*) [ , *text_options* ]

*Add image to paragraph*

> putdocx image *filename* [ , *image_options* ]

*Add table to document*

> putdocx table *tablename* = (*nrows*, *ncols*) [ , *table_options* ]
>
> putdocx table *tablename* = data(*varlist*) [ *if* ] [ *in* ] [ , varnames obsno
>     *table_options* ]
>
> putdocx table *tablename* = matrix(*matname*) [ , nformat(%*fmt*) rownames colnames
>     *table_options* ]
>
> putdocx table *tablename* = mata(*matname*) [ , nformat(%*fmt*) *table_options* ]
>
> putdocx table *tablename* = *returnset* [ , *table_options* ]

*Customize rows and columns*

> putdocx table *tablename*(*i*, .), *row_col_options*
>
> putdocx table *tablename*(., *j*), *row_col_options*

*Add and format contents of table*

> putdocx table *tablename*(*i*, *j*) = (*exp*) [ , *cell_options* ]
>
> putdocx table *tablename*(*i*, *j*) = image(*filename*) [ , *image_options* *cell_options* ]
>
> putdocx table *tablename*(*i*, *j*) = table(*mem_tablename*) [ , *cell_options* ]
>
> putdocx table *tablename*(*i*, *j*), *cell_options*

*Add page break to document*

> putdocx pagebreak

*Describe current document*

> putdocx describe

*Describe table*

> putdocx describe *tablename*

*Save document*

> putdocx save *filename* $\left[\right.$ , replace | append $\left.\right]$

*Merge documents*

> putdocx append *filename*₁ *filename*₂ $\left[\right.$ *filename*₃ $\left[\right.$ ... $\left.\right]\left.\right]$
>
> > $\left[\right.$ , <u>sav</u>ing(*filename* $\left[\right.$ , replace $\left.\right]$) $\left.\right]$

*tablename* specifies the name of a new table. The name must be a valid name according to Stata's naming conventions; see [U] **11.3 Naming conventions**.

*nrows* and *ncols* specify the number of rows and the number of columns in the new table.

*matname* specifies the name of an existing Stata matrix or Mata matrix.

*returnset* specifies a group of Stata return values and may be one of the following:

| *returnset* | Description |
|---|---|
| <u>esc</u>alars | All ereturned scalars |
| <u>rsc</u>alars | All returned scalars |
| <u>emac</u>ros | All ereturned macros |
| <u>rmac</u>ros | All returned macros |
| <u>emat</u>rices | All ereturned matrices |
| <u>rmat</u>rices | All returned matrices |
| e* | All ereturned scalars, macros, and matrices |
| r* | All returned scalars, macros, and matrices |

*mem_tablename* specifies the name of a previously created table.

| *paragraph_options* | Description |
|---|---|
| style(*pstyle*) | set paragraph text with specific style |
| font(*fspec*) | set font, font size, and font color |
| halign(*hvalue*) | set paragraph alignment |
| valign(*vvalue*) | set vertical alignment of characters on each line |
| indent(*indenttype*, #[ *unit* ]) | set paragraph indentation |
| spacing(*position*, #[ *unit* ]) | set spacing between lines of text |
| <u>shad</u>ing(*sspec*) | set background color, foreground color, and fill pattern |

| *text_options* | Description |
|---|---|
| <u>nf</u>ormat(*%fmt*) | specify numeric format for new text |
| font(*fspec*) | set font, font size, and font color |
| bold | format new text as bold |
| italic | format new text as italic |
| script(sub\|super) | set subscript or superscript formatting of new text |
| <u>underline</u>[ (*upattern*) ] | underline text using specified pattern |
| strikeout | strikeout new text |
| <u>shad</u>ing(*sspec*) | set background color, foreground color, and fill pattern |
| linebreak | add a line break after new text |

| *image_options* | Description |
|---|---|
| <u>w</u>idth(#[ *unit* ]) | set image width |
| <u>h</u>eight(#[ *unit* ]) | set image height |
| link | insert link to image file |
| linebreak | add a line break after new image; not available with putdocx table |

| *table_options* | Description |
|---|---|
| <u>memtable</u> | keep table in memory rather than add it to document |
| width(#[ *unit*\|% ]) | set table width |
| layout(*layouttype*) | adjust column width |
| halign(*hvalue*) | set table horizontal alignment |
| indent(#[ *unit* ]) | set table indentation |
| <u>headerrow</u>(#) | set number of the top rows that constitute the table header |
| <u>border</u>(*bspec*) | set pattern and color for table border |
| <u>cellmargin</u>(*cmarg*, #[ *unit* ]) | set margins for each table cell |
| <u>cellspacing</u>(#[ *unit* ]) | set spacing between adjacent cells and the edges of the table |

| *row_col_options* | Description |
|---|---|
| <u>nosplit</u> | prevent row from breaking across pages |
| addrows(#[ , before\|after ]) | add # rows in specified location |
| addcols(#[ , before\|after ]) | add # columns in specified location |
| drop | drop specified row or column |

| *cell_options* | Description |
|---|---|
| append | append objects to current content of cell |
| font(*fspec*) | set font, font size, and font color |
| bold | format text as bold in cell |
| italic | format text as italic in cell |
| underline [ (*upattern*) ] | underline cell text using specified pattern |
| strikeout | strikeout cell text |
| halign(*hvalue*) | set horizontal alignment of cell |
| valign(*vvalue*) | set vertical alignment of cell |
| rowspan(#) | merge cells vertically |
| colspan(#) | merge cells horizontally |
| span(#$_1$, #$_2$) | merge cells both horizontally and vertically |
| border(*bspec*) | set pattern and color for cell border |
| shading(*sspec*) | set background color, foreground color, and fill pattern |

*fspec* is

　*fontname* [ , *size* [ , *color* ] ]

　　*fontname* may be any valid font installed on the user's computer. If *fontname* includes spaces, then it must be enclosed in double quotes.

　　*size* is a numeric value that represents font size measured in points. The default is 11.

　　*color* sets the text color.

*sspec* is

　*bgcolor* [ , *fgcolor* [ , *fpattern* ] ]

　　*bgcolor* specifies the background color.

　　*fgcolor* specifies the foreground color. The default foreground color is black.

　　*fpattern* specifies the fill pattern. The most common fill patterns are solid for a solid color (determined by *fgcolor*), pct25 for 25% gray scale, pct50 for 50% gray scale, and pct75 for 75% gray scale. A complete list of fill patterns is shown in the *Appendix*.

*bspec* is

　*bordername* [ , *bpattern* [ , *bcolor* ] ]

　　*bordername* specifies the location of the border.

　　*bpattern* is a keyword specifying the look of the border. The most common patterns are single, dashed, dotted, and double. The default is single. For a complete list of border patterns, see the *Appendix*. To remove an existing border, specify nil as the *bpattern*.

　　*bcolor* specifies the border color.

*unit* may be in (inch), pt (point), cm (centimeter), or twip (twentieth of a point). An inch is equivalent to 72 points, 2.54 centimeters, or 1440 twips. The default is in.

*color*, *bgcolor*, *fgcolor*, and *bcolor* may be one of the colors listed in the table of colors 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.

## Output types

putdocx paragraph adds a new paragraph to the active document. The newly created paragraph becomes the active paragraph. All subsequent text or images will be appended to the active paragraph.

putdocx text (*exp*) adds content to the paragraph created by putdocx paragraph. *exp* may be a valid Stata expression (see [U] **13 Functions and expressions**) or a normal string.

putdocx image *filename* embeds a portable network graphics (.png), JPEG (.jpg), enhanced metafile (.emf), or tagged image file format (.tif) file in the active paragraph. *filename* is the path to the image file. It may be either the full path or the relative path from the current working directory.

Adding an image is not supported on console Stata for Mac.

putdocx table *tablename* creates a new table that can be identified by its assigned name, *tablename*, for future modifications. Tables may be created in several ways:

(*nrows*, *ncols*) creates an empty table with *nrows* rows and *ncols* columns. Microsoft Word 2007/2010 allows a maximum of 63 columns in a table.

data(*varlist*) [ *if* ] [ *in* ] [ , varnames obsno ] adds the current Stata dataset in memory as a table to the active document. *varlist* contains a list of the variable names from the current dataset in memory. If the varnames option is specified, the first row of the table is filled with variable names. If obsno is specified, the first column of the table is filled with observation numbers. if and in may be used to restrict the data to be added to the table.

matrix(*matname*) [ , nformat(*%fmt*) rownames colnames ] adds a matrix called *matname* as a table to the active document. The elements of the matrix are formatted using *%fmt*. If nformat() is not specified, then %12.0g is used. If the rownames option is specified, the first column of the table is filled with matrix rownames. If the colnames option is specified, the first row of the table is filled with matrix colnames.

mata(*matname*) [ , nformat(*%fmt*) ] adds a Mata matrix called *matname* as a table to the active document. The elements of the matrix are formatted using *%fmt*. If nformat() is not specified, then %12.0g is used.

*returnset* exports a group of Stata return values to a table in the active document. It is intended primarily for use by programmers and by those who want to do further processing of their exported results in the active document.

putdocx table *tablename*(*i*, *j*) modifies the content and styles of the cell on the *i*th row and *j*th column in table *tablename*. The cell content may be a valid Stata expression, an image with supported format, a previously created table identified by *mem_tablename*, or combinations of those three elements by using option append.

# Options

Options are presented under the following headings:

## Options for putdocx begin

pagesize(*psize*) sets the page size of the document. *psize* may be letter, legal, A3, A4, or B4JIS. The default is pagesize(letter).

landscape changes the document orientation from portrait to landscape.

font(*fontname* [ , *size* [ , *color* ] ]) sets the font, font size, and font color for the document. Note that 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.

## Options for putdocx paragraph

style(*pstyle*) specifies that the text in the paragraph be formatted with style *pstyle*. Common values for *pstyle* are Title, Subtitle, and Heading1. See the complete list of paragraph styles in the *Appendix*.

font(*fontname* [ , *size* [ , *color* ] ]) sets the font, font size, and font color for the paragraph. Note that 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.

Specifying font() with putdocx paragraph overrides font settings specified with putdocx begin.

halign(*hvalue*) sets the horizontal alignment of the characters within the paragraph. *hvalue* may be left, right, center, both, or distribute. distribute and both justify text between the left and right margins equally, but distribute also changes the spacing between words and characters. The default is halign(left).

valign(*vvalue*) sets the vertical alignment of the characters on each line when the paragraph contains characters of varying size. *vvalue* may be auto, baseline, bottom, center, or top. The default is valign(baseline).

indent(*indenttype*, #[ *unit* ]) specifies that the paragraph be indented by # *units*. *indenttype* may be left, right, hanging, or para. left and right indent # *units* from the left or the right, respectively. hanging uses hanging indentation and indents lines after the first line by # inches unless another *unit* is specified. para uses standard paragraph indentation and indents the first line by # inches unless another *unit* is specified. This option may be specified multiple times in a single command to accommodate different indentation settings. If both indent(hanging) and indent(para) are specified, only indent(hanging) is used.

spacing(*position*, #[*unit*]) sets the spacing between lines of text. *position* may be before, after, or line. before specifies the space before the first line of the current paragraph, after specifies the space after the last line of the current paragraph, and line specifies the space between lines within the current paragraph. This option may be specified multiple times in a single command to accommodate different spacing settings.

shading(*bgcolor* [ , *fgcolor* [ , *fpattern* ] ]) sets the background color, foreground color, and fill pattern for the paragraph.

## Options for putdocx text

nformat(%*fmt*) specifies the numeric format of the text when the content of the new text appended to the paragraph is a numeric value. This setting has no effect when the content is a string.

font(*fontname* [ , *size* [ , *color* ] ]) sets the font, font size, and font color for the new text. Note that 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.

Specifying font() with putdocx text overrides all other font settings, including those specified with putdocx begin and putdocx paragraph.

bold specifies that the new text be formatted as bold.

italic specifies that the new text be formatted as italic.

underline[ (*upattern*) ] specifies that the new text be underlinedand optionally specifies the format of the line. By default, a single underline is used. The optional *upattern* may be any of the patterns listed in the *Appendix*. The most common patterns are double, dash, and none.

strikeout specifies that the text have a strikeout mark.

script(sub | super) changes the script style of the text. script(sub) makes the text a subscript. script(super) makes the text a superscript.

shading(*bgcolor* [ , *fgcolor* [ , *fpattern* ] ]) sets the background color, foreground color, and fill pattern for the new text.

Specifying shading() with putdocx text overrides shading specifications from putdocx paragraph.

linebreak specifies that a line break be added after the new text. This option may be specified multiple times to add multiple line breaks after the new text.

## Options for putdocx image

width(#[*unit*]) sets the width of the image. If the width is larger than the body width of the document, then the body width is used. If width() is not specified, then the default size is used; the default is determined by the image information and the body width of the document.

height(#[*unit*]) sets the height of the image. If height() is not specified, then the height of the image is determined by the width and the aspect ratio of the image.

link specifies that a link to the image *filename* be inserted into the document. If the image is linked, then the referenced file must be present so that the document can display the image.

linebreak specifies that a line break be added after the new image. This option may be specified multiple times to add multiple line breaks after the new image.

## Options for putdocx table

### table_options

   memtable specifies that the table be created and held in memory instead of being added to the active
   document. By default, the table is added to the document immediately after it is created. This
   option is useful if the table is intended to be added to a cell of another table or to be used multiple
   times later.

   halign(*hvalue*) sets the horizontal alignment of the table within the page. *hvalue* may be left,
   right, or center. The default is halign(left).

   indent(#[*unit*]) specifies the table indentation from the left margin of the current document.

   layout(*layouttype*) adjusts the column width of the table. *layouttype* may be fixed, autofitwindow,
   or autofitcontents. fixed means the width is the same for all columns in the table. When
   autofitwindow is specified, the column width automatically resizes to fit the window. When aut-
   ofitcontents is specified, the table width is determined by the overall table layout algorithm, which
   automatically resizes the column width to fit the contents. The default is layout(autofitwindow).

   width(#[*unit*|%]) sets the table width. *#* may be an absolute width or a percent of the default table
   width, which is determined by the page width of the document. For example, width(50%) sets
   the table width to 50% of the default table width. The default is width(100%).

   cellmargin(*cmarg*, #[*unit*]) sets the cell margins for table cells. *cmarg* may be top, bottom,
   left, or right. This option may be specified multiple times in a single command to accommodate
   different margin settings.

   cellspacing(#[*unit*]) sets the spacing between adjacent cells and the edges of the table.

   border(*bordername* [, *bpattern* [, *bcolor*]]) sets the pattern and color for the specified border
   of the table or a cell. This option may be specified multiple times in a single command to
   accommodate different border settings.

   For a table or cell, *bordername* may be start, end, top, bottom, or all. For a table, *bordername*
   may also be insideH, the inside horizontal edges border, or insideV, the inside vertical edges
   border.

   Specifying border(start) adds a single border to the leading edge of the table or cell. Similarly,
   specifying border(end), border(top), or border(bottom) adds a single border to the trailing
   edge, top, or bottom of the table or cell. border(all) applies the specified pattern and color to
   all borders, including the inside edges when specified for a table.

   headerrow(#) sets the top *#* rows to be repeated as header rows at the top of each page on which
   the table is displayed. This setting has a visible effect only when the table crosses multiple pages.

   varnames specifies that the variable names be included as the first row in the table when the table
   is created using the dataset in memory. By default, only the data values are added to the table.

   obsno specifies that the observation numbers be included as the first column in the table when the
   table is created using the dataset in memory. By default, only the data values are added to the
   table.

   nformat(%*fmt*) specifies the numeric format to be applied to the source values when creating the
   table from a Stata or Mata matrix. The default is nformat(%12.0g).

   rownames specifies that the row names of the Stata matrix be included as the first column in the
   table. By default, only the matrix values are added to the table.

   colnames specifies that the column names of the Stata matrix be included as the first row in the
   table. By default, only the matrix values are added to the table.

**row_col_options**

nosplit specifies that row $i$ not split across pages. When displaying a table row, it is possible that a page break will fall within the contents of a cell on the row, causing the contents of that cell to be displayed across two pages. nosplit prevents this behavior. If the entire row cannot fit on the current page, the row will be moved to the start of the next page.

addrows(# [ , before | after ]) adds # rows to the current table before or after row $i$. If before is specified, the rows are added before the specified row. By default, rows are added after the specified row.

addcols(# [ , before | after ]) adds # columns to the current table to the right or the left of column $j$. If before is specified, the columns are added to the left of the specified column. By default, the columns are added after, or to the right of, the specified column.

drop deletes row $i$ or column $j$ from the table.

**cell_options**

append specifies that the new content for the cell be appended to the current content of the cell. If append is not specified, then the current content of the cell is replaced by the new content.

font(*fontname* [ , *size* [ , *color* ] ]) sets the font, font size, and font color for the text within the cell. Note that 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.

bold specifies that the text be formatted as bold.

italic specifies that the text be formatted as italic.

underline [ (*upattern*) ] specifies that the cell be underlinedand optionally specifies the format of the line. By default, a single underline is used. The optional *upattern* may be any of the patterns listed in the *Appendix*. The most common patterns are double, dash, and none.

strikeout specifies that the text have a strikeout mark.

halign(*hvalue*) sets the horizontal alignment of the cell. *hvalue* may be left, right, or center. The default is halign(left).

valign(*vvalue*) sets the vertical alignment of the cell. *vvalue* may be top, bottom, or center. The default is valign(top).

rowspan(#) sets the specified cell to span vertically # cells downward. If the span exceeds the total number of rows in the table, the span stops at the last row.

colspan(#) sets the specified cell to span horizontally # cells to the right. If the span exceeds the total number of columns in the table, the span stops at the last column.

span(#$_1$, #$_2$) sets the specified cell to span #$_1$ cells downward and span #$_2$ cells to the right.

shading(*bgcolor* [ , *fgcolor* [ , *fpattern* ] ]) sets the background color, foreground color, and fill pattern for the cell.

**image_options**

width(#[ *unit* ]) sets the width of the image. If the width is larger than the width of the cell, then the width is used. If width() is not specified, then the default size is used; the default is determined by the image information and the width of the cell.

height(#[ *unit* ]) sets the height of the image. If height() is not specified, then the height of the image is determined by the width and the aspect ratio of the image.

link specifies that a link to the image *filename* be inserted into the document. If the image is linked, then the referenced file must be present so that the document can display the image.

## Options for putdocx save

replace specifies to overwrite *filename*, if it exists, by the contents of the document in memory.

append specifies to append the contents of the document in memory to the end of *filename*.

## Option for putdocx append

saving(*filename* [ , replace ]) specifies to append the contents of the existing document *filename*$_2$ to the end of *filename*$_1$ and then write the result to the new document *filename*. If *filename* already exists, it can be overwritten by specifying replace. By default, *filename*$_1$ is replaced with the document created by appending content from *filename*$_2$.

If more than two files are specified, the contents are appended in the order in which they are listed. For example, *filename*$_2$ is appended to *filename*$_1$, *filename*$_3$ is appended to the result of the first append, and so forth.

## Remarks and examples

The putdocx command is a suite of commands used to write paragraphs, images, and tables to an Office Open XML file (.docx). It may also be used to format each object added. This allows you to automate the exporting and formatting of, for example, Stata graphs and estimation results, and also to generate various reports based on those elements. Moreover, it can be used to append those elements to an existing .docx file or to merge multiple .docx files. Hence, it provides a convenient way to interact Stata results with a .docx file.

putdocx generates files compatible with Microsoft Word 2007 and later. It is supported on Windows, Mac, and Linux.

Before we can write to a .docx file using putdocx, we need to create a .docx document in memory. We do this using the putdocx begin command.

```
. putdocx begin
```

Once the document is created, other objects such as paragraphs and tables may be added to it. By default, the document created uses the letter pagesize, and its layout is set to portrait. Those properties may be overwritten by specifying corresponding document options. Some other properties may also be customized; see *Options for putdocx begin* in *Options*.

After we are done editing the document, we can save it to disk.

```
. putdocx save example.docx
```

Note that either the `replace` or the `append` option is required if `example.docx` already exists in the saving directory. If `replace` is specified, then all the contents in `example.docx` will be overwritten. If `append` is specified, then all the contents of the active document in memory will be appended to the end of `example.docx`.

Remaining remarks are presented under the following headings:

## Add a paragraph

Before you can add text or an image to the paragraph, you must first begin a new paragraph by using `putdocx paragraph`. You can control the formatting for the whole paragraph, such as font properties and alignment, with options for `putdocx paragraph`. See *Options for putdocx paragraph* for paragraph formatting options. The current paragraph remains active until you add a new paragraph or add a table.

## Add text to paragraph

Once the new paragraph is created, you add text to it by using `putdocx text`. The new text is appended to the text or the image that has already been added to the paragraph. This text can also be formatted individually. See *Options for putdocx text* for text formatting options.

▷ Example 1: Add a paragraph and format the text

Suppose we want to write a description of `auto.dta` to `example.docx`. Our description includes the number of automobiles and the maximum miles per gallon (MPG) among all the automobiles. We can create a paragraph containing that information by using `putdocx paragraph`.

To start, we use the summarize command to get descriptive statistics for the `mpg` variable. After that, we can use the returned results from `summarize` in the text that we write. The returned results `r(N)` and `r(max)` store the number of automobiles and the maximum MPG among those automobiles, respectively.

```
. use http://www.stata-press.com/data/r15/auto
(1978 Automobile Data)

. summarize mpg
    Variable │        Obs        Mean    Std. Dev.        Min        Max
─────────────┼─────────────────────────────────────────────────────────
         mpg │         74     21.2973     5.785503         12         41
```

```
. return list
scalars:
                r(N) =  74
            r(sum_w) =  74
             r(mean) =  21.2972972972973
              r(Var) =  33.47204738985561
               r(sd) =  5.785503209735141
              r(min) =  12
              r(max) =  41
              r(sum) =  1576
```

Now, we specify the command to create our document. We then add a new paragraph to the active document and append text to it. The content of each `putdocx text` command may be a valid Stata expression (see [U] **13 Functions and expressions**) or a normal string. `putdocx text` can be used to break long sentences into pieces, and each piece in the paragraph can be customized to have a different style. Here, `r(max)` is formatted as bold.

```
. putdocx begin
. putdocx paragraph
. putdocx text ("In this dataset, there are 'r(N)'")
. putdocx text (" models of automobiles. The maximum MPG among them is ")
. putdocx text (r(max)), bold
. putdocx text (".")
```

This adds text to the document that looks like this:



◁
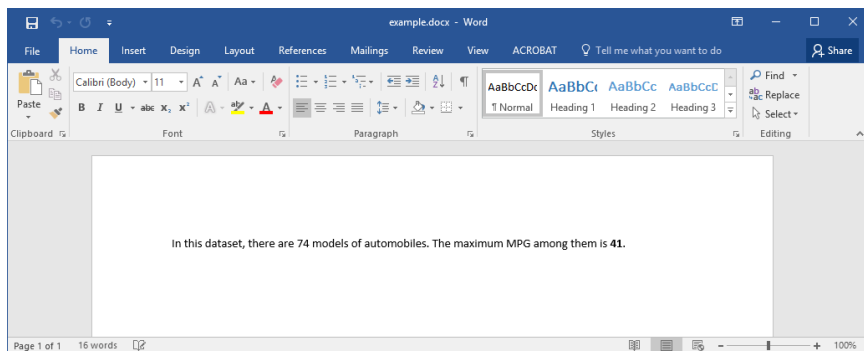
## Add an image to paragraph

You can add any existing `.png`, `.jpg`, `.emf`, and `.tif` image files to a `.docx` file with `putdocx image`. For example, you could include a company logo. You can also add graphs from Stata output. Because Stata graphs use the `.gph` extension, you must use `graph export` to convert the Stata graph to one of the supported image formats; see [G-2] **graph export**.

By default, images are embedded in the file. If the image is embedded, it becomes a part of the document and has no further relationship with the original image on the disk. You can instead link the image by specifying the `link` option. Using linked images means that if the image file on the disk is updated, then the linked image in the document will reflect the change.

If you are adding the image after text and you want the paragraph that contains the image to have the same format as the active paragraph, you can insert the image with no additional step. However, if you want to change the formatting or if there is no active paragraph, you must create one using `putdocx paragraph`. Note that you do not need to declare a new paragraph to insert an image into the cell of a table.

▷ Example 2: Export a Stata graph

We may want to add a scatterplot showing how mileage (`mpg`) correlates with price (`price`) of cars. We can use the scatter command and then graph export to create a `.png` file.

```
. scatter mpg price
```
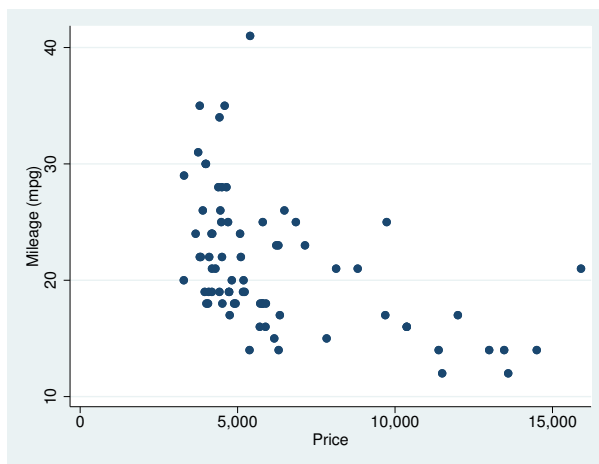


```
. graph export auto.png
(file auto.png written in PNG format)
```

Next, we use `putdocx image` to add the `.png` file to the document. Because our active paragraph is left aligned and we want our image to be centered, we declare a new paragraph and specify the `halign()` option.

```
. putdocx paragraph, halign(center)
. putdocx image auto.png
```

◁

## Add a table

`putdocx table` is used to add a table to the document. A valid table name is required to declare a table. The table name is used later as a reference to customize the table and the cells.

**Export data**

Exporting the data in memory is useful when you want to make a table in the .docx file using content from your dataset. The `if` or `in` qualifier may be applied to export only those observations that meet the specified condition or are in the specified range (or both, if both `if` and `in` are specified).

▷ Example 3: Export table of summary statistics

Suppose we want to export summary statistics, such as the number of automobiles, the maximum and minimum MPG, and the average MPG, that have been calculated separately for foreign and domestic automobiles. To start, we use the statsby command to collect the above statistics for each group. Because `statsby` creates a new dataset that overwrites the dataset in memory, we need to preserve the dataset and then restore it after we have finished exporting the data.

```
. preserve
. statsby Total=r(N) Average=r(mean) Max=r(max) Min=r(min), by(foreign):
> summarize mpg
(running summarize on estimation sample)

      command:  summarize mpg
        Total:  r(N)
      Average:  r(mean)
          Max:  r(max)
          Min:  r(min)
           by:  foreign

Statsby groups
────┼──── 1 ────┼──── 2 ────┼──── 3 ────┼──── 4 ────┼──── 5
..
```

Then, we export the data in memory as a table by specifying in `data()` the variable names `Origin`, `Total`, `Average`, `Max`, and `Min`. In this case, we use the table name tbl1. The order of the variable names in the list determines the column order in the table.

```
. rename foreign Origin
. putdocx table tbl1 = data("Origin Total Average Max Min"), varnames
> border(start, nil) border(insideV, nil) border(end, nil)
```

By default, the exported table includes single borders around all cells. We use the `border()` option to remove all vertical borders from the table. We can apply additional formatting to individual cells or ranges of cells; see example 4.

◁

▷ Example 4: Format table

The cells in the table that we created in example 3 can be further customized. For example, we can reset the contents, set the text alignment, modify the borders, and so forth. Here, we set the horizontal text of the cells on the second through the fifth columns to be right aligned instead of the default left alignment.

```
. forvalues row=1/3 {
  2.          forvalues col=2/5 {
  3.                  putdocx table tbl1(`row',`col'), halign(right)
  4.          }
  5. }
```

Our formatted tbl1 looks like this:
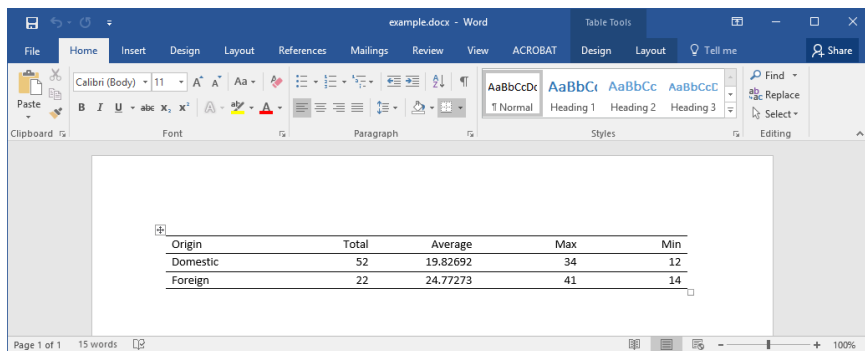


Afterward, we restore the dataset.

```
. restore
```

◁

## Export estimation results

One of the primary uses of putdocx table is to export estimation results. Suppose we fit a linear regression model of mpg as a function of the car's gear ratio (gear_ratio), turning radius (turn), and whether the car is of foreign origin (foreign) using regress. We want to add selected regression results to the document.

```
. regress mpg gear_ratio turn foreign, noheader
```

| mpg | Coef. | Std. Err. | t | P>|t| | [95% Conf. Interval] | |
|---|---|---|---|---|---|---|
| gear_ratio | 4.855506 | 1.522481 | 3.19 | 0.002 | 1.819013 | 7.891999 |
| turn | -.8364698 | .1440204 | -5.81 | 0.000 | -1.123709 | -.5492302 |
| foreign | -3.503218 | 1.433526 | -2.44 | 0.017 | -6.362296 | -.6441404 |
| _cons | 40.865 | 8.692731 | 4.70 | 0.000 | 23.52789 | 58.2021 |

```
. return list

scalars:
            r(level) =  95

matrices:
            r(table) :  9 x 4
. matrix list r(table)

r(table)[9,4]
          gear_ratio         turn      foreign        _cons
     b    4.8555057  -.83646975   -3.5032183    40.864996
    se    1.5224812   .14402036    1.4335262    8.6927313
     t    3.1892057  -5.8079965   -2.4437769    4.7010537
pvalue    .0021348    1.704e-07    .01705791    .00001258
    ll    1.8190127  -1.1237093   -6.3622962    23.527891
    ul    7.8919987   -.5492302   -.64414044    58.202102
    df          70           70           70           70
  crit    1.9944371    1.9944371    1.9944371    1.9944371
 eform           0            0            0            0
. matrix rtable = r(table)'
```

The displayed results returned by `regress` are saved in `r(table)`. We create the matrix `rtable` as the transpose of `r(table)` because we want to see the variable names in rows.

▷ Example 5: Export selected estimation results

Suppose we want to export only the point estimates and confidence intervals. The point estimates and confidence intervals in the regression table can be extracted from the matrix `rtable`. We can extract columns 1, 5, and 6 from `rtable`, combine them, and assign them to another new matrix, `r_table`.

```
. matrix r_table = rtable[1...,1], rtable[1...,5..6]
```

Then, we export `r_table` to the document as a table with the name `tbl2`.

```
. putdocx table tbl2 = matrix(r_table), nformat(%9.3f) rownames colnames
> border(start, nil) border(insideH, nil) border(insideV, nil) border(end, nil)
```

In this table, all values imported from the matrix have been formatted as `%9.3f`. In addition, the row and column names of the matrix `r_table` are included as the first row and first column of the table. We keep only the top and bottom borders of the table by specifying `nil` for the leading edge border (`start`), trailing edge border (`end`), inside horizontal edges border (`insideH`), and inside vertical edges border (`insideV`).

The column names from the matrix may not be exactly what we want; we can modify them by customizing the corresponding cells. We can reset the contents and the horizontal alignment of the cells on the first row to give the columns new titles.

```
. putdocx table tbl2(1,2) = ("Coef."), halign(right)
. putdocx table tbl2(1,3) = ("[95% Conf. Interval]"), halign(right) colspan(2)
```

Afterward, we add back the bottom border of the first row and right-align the cells on the rest of the rows by looping over each cell; see [P] **forvalues**.

```
. forvalues i = 1/5 {
  2.          forvalues j=1/4 {
  3.                  if `i'==1 {
  4.                          putdocx table tbl2(`i',`j'), border(bottom)
  5.                  }
  6.                  else {
  7.                          putdocx table tbl2(`i',`j'), halign(right)
  8.                  }
  9.          }
 10. }
```

Our final table appears in the document as follows:
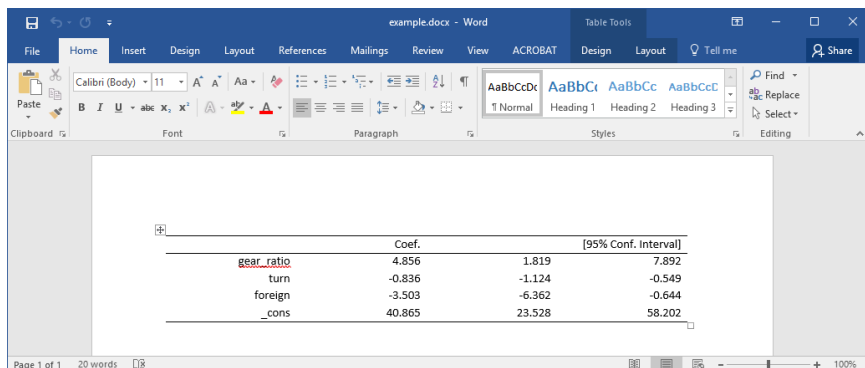


We can now type

```
. putdocx save example.docx
```

to save the document that we created as `example.docx`.

◁

## Advanced uses

In this section, we begin a new document and add some more-advanced tables to it.

```
. putdocx begin
```

Afterward, we can append these tables to the first set of examples by using

```
. putdocx save example.docx, append
```

▷ Example 6: Create table from components

Sometimes, we might need a table with a complicated layout. It may be difficult for us to figure out the dimension of the table and the content of each cell. In this situation, we can consider building the table in pieces and combining them. For illustrative purposes, we use the table created in example 5. The table can be considered to comprise two parts, the header and the bottom. We first create a table for each part.

The header part is specified as a $1 \times 2$ table with borders removed and contents aligned to the right.

```
. putdocx table tbl31 = (1,2), memtable border(all, nil)
. putdocx table tbl31(1,1) = ("Coef."), halign(right)
. putdocx table tbl31(1,2) = ("[95% Conf. Interval]"), halign(right)
```

The bottom part is created as another table by using the same Stata matrix from example 5, `r_table`. Again, we remove all borders and align all contents to the right.

```
. putdocx table tbl32 = matrix(r_table), memtable border(all, nil)
> nformat(%9.3f) rownames

. forvalues i=1/4 {
  2.          forvalues j=1/4 {
  3.                  putdocx table tbl32('i','j'), halign(right)
  4.          }
  5. }
```

Notice that both tables are created with the `memtable` option, which declares that they be created in memory rather than added to the document. This is required for our next steps.

At last, we create a $2 \times 1$ table named `tbl3`, in which the first cell contains `tbl31` and the second cell contains `tbl32`. In addition, we remove its leading edge border and trailing edge border.

```
. putdocx table tbl3 = (2,1), border(start, nil) border(end, nil)

. putdocx table tbl3(1,1) = table(tbl31)

. putdocx table tbl3(2,1) = table(tbl32)
```

This is a simple example of how to create a table using nested tables.

◁

## ▷ Example 7: Create table dynamically

In the previous example, we added a table in two steps. The first step was to create a table in the document. The second step was to fill in the contents for each cell. In this process, the dimensions of the table were often predetermined. Another way to add a table is to create it dynamically: start with a simple table and then add rows or columns to it gradually.

Returning to example 5, suppose we want to add an estimation table to the document containing the point estimates, their standard errors, the total number of observations, and the coefficient of determination. The estimation table looks like the following:



First, we create a $1 \times 2$ table with no borders and fill the single row with the dependent variable name, `mpg`. We also set the table width to be 4 inches and put the table in the center of the document.

```
. putdocx table tbl4 = (1,2), border(all, nil) width(4) halign(center)
. putdocx table tbl4(1,1)=("mpg"), halign(right) colspan(2) border(top)
> border(bottom)
```

Notice that each regressor in the model takes up 2 rows and 2 columns in the table, which means 4 cells. The cell in the first row and first column contains the variable name. The cell in the first row and second column contains the point estimate. The cell in the second row and second column contains the standard errors. Based on this logic, we add two rows for each regressor at the end of the table every time, and we fill in the content and format for each cell one by one.

```
. local row 1
. local vari 1
. foreach x in gear_ratio turn foreign _cons {
  2.        putdocx table tbl4('row',.), addrows(2)
  3.
  4.        local b: display %9.3f rtable['vari',1]
  5.        local se: display %9.3f rtable['vari',2]
  6.        local ++vari
  7.
  8.        local ++row
  9.        putdocx table tbl4('row',1) = ("'x'"), halign(right)
 10.        putdocx table tbl4('row',2) = ("'b'"), halign(right)
 11.        local ++row
 12.        local se = strtrim("'se'")
 13.        putdocx table tbl4('row',2) = ("('se')"), halign(right)
 14. }
```

Afterward, we add three more rows to the end of the table for the number of observations, the coefficient of determination, and a note as a footer.

```
. putdocx table tbl4('row',.), addrows(3)
```

We then add each statistic to the table.

```
. local ++row
. putdocx table tbl4('row',1) = ("N"), border(top) halign(right)
. putdocx table tbl4('row',2) = (e(N)), border(top) halign(right)
. local ++row
. local r2: display %9.3f e(r2)
. putdocx table tbl4('row',1) = ("R2"), halign(right)
. putdocx table tbl4('row',2) = ('r2'), halign(right)
```

Finally, we add the footnote "Note: standard errors in parentheses" that spans two columns with a border on the top to separate the note from the rest of the table.

```
. local ++row
. putdocx table tbl4('row',1) = ("Note: standard errors in parentheses"),
> colspan(2) border(top)
```

◁

▷ Example 8: Nesting images in a table

We might want to add various images to the document and align them side by side, row by row, or both. To be more clear, we use an example to illustrate this purpose. In this example, we use another dataset—the Second National Health and Nutrition Examination Survey (NHANES II) (McDowell et al. 1981).

First, we fit a three-way full factorial model of systolic blood pressure on age group, sex, and body mass index (BMI). Then, we estimate the predictive margins for each combination of `agegrp` and `sex` at levels of BMI from 10 through 40 at intervals of 10 and graph the results.

```
. use http://www.stata-press.com/data/r15/nhanes2, clear

. regress bpsystol agegrp##sex##c.bmi
  (output omitted )

. forvalues v=10(10)40 {
  2.        margins agegrp, over(sex) at(bmi=`v')
  3.        marginsplot
  4.        graph export bmi`v'.png
  5. }
  (output omitted )
```

Now, we want to add those four plots into the document, requiring that the margins plots for `bmi=10` and `bmi=20` lay side by side on top of the other two side-by-side margins plots for `bmi=30` and `bmi=40`. It is also required that each plot have a subtitle indicating the level of BMI and that the final figure have a title. This complicated layout can be easily accomplished using `putdocx table`.

We start with a $2 \times 2$ table and remove all of its borders. In each cell, we add a plot and then append center-aligned text to it. Finally, we add another row to the end of the table and fill it with the title for the figure.

```
. putdocx table tbl5 = (2,2), border(all,nil)

. putdocx table tbl5(1,1)=image(bmi10.png)

. putdocx table tbl5(1,1)=("(a) bmi=10"), append halign(center)

. putdocx table tbl5(1,2)=image(bmi20.png)

. putdocx table tbl5(1,2)=("(b) bmi=20"), append halign(center)

. putdocx table tbl5(2,1)=image(bmi30.png)

. putdocx table tbl5(2,1)=("(c) bmi=30"), append halign(center)

. putdocx table tbl5(2,2)=image(bmi40.png)

. putdocx table tbl5(2,2)=("(d) bmi=40"), append halign(center)

. putdocx table tbl5(2,.), addrows(1)

. putdocx table tbl5(3,1)=("Figure 1: Predictive margins of agegrp"),
> halign(center) colspan(2) bold
```

This creates a table that looks like the following:
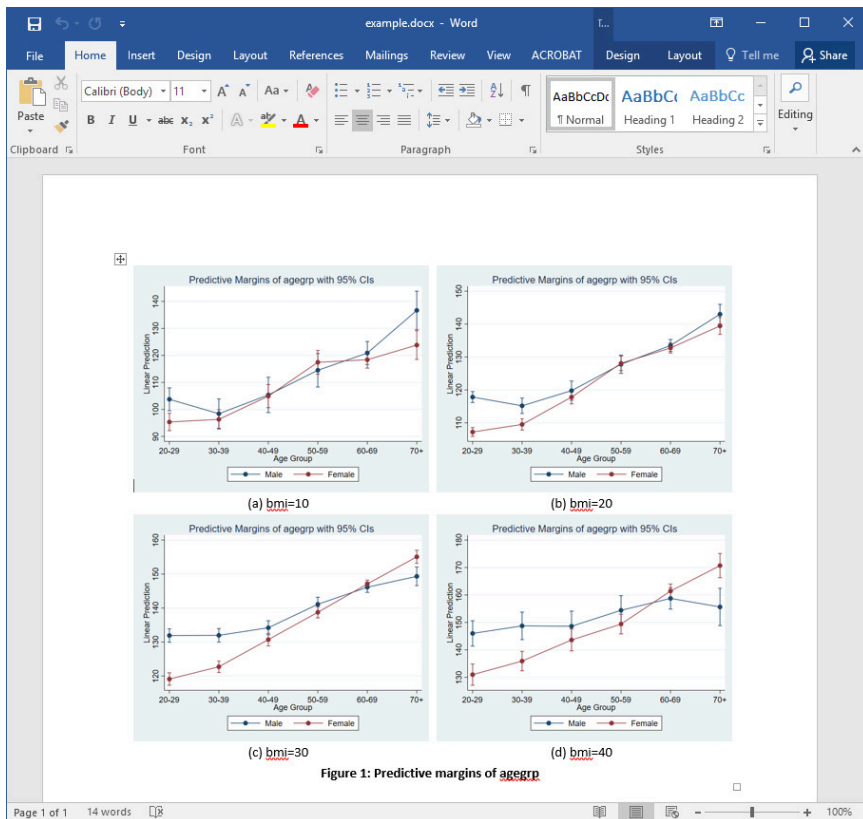
# Appendix

## Paragraph styles

| *pstyle* | |
|---|---|
| Title | Heading5 |
| Subtitle | Heading6 |
| Heading1 | Heading7 |
| Heading2 | Heading8 |
| Heading3 | Heading9 |
| Heading4 | |

## 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 |

*color*, continued

| | |
|---|---|
| darkorange | lightgray |
| darkorchid | lightgreen |
| darkred | lightpink |
| darksalmon | lightsalmon |
| darkseagreen | lightseagreen |
| darkslateblue | lightskyblue |
| darkslategray | lightslategray |
| darkturquoise | lightsteelblue |
| darkviolet | lightyellow |
| 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 | |

## Shading patterns

*fpattern*

| | |
|---|---|
| nil | pct20 |
| clear | pct25 |
| solid | pct30 |
| horzStripe | pct35 |
| vertStripe | pct37 |
| reverseDiagStripe | pct40 |
| diagStripe | pct45 |
| horzCross | pct50 |
| diagCross | pct55 |
| thinHorzStripe | pct60 |
| thinVertStripe | pct62 |
| thinReverseDiagStripe | pct65 |
| thinDiagStripe | pct70 |
| thinHorzCross | pct75 |
| thinDiagCross | pct80 |
| pct5 | pct85 |
| pct10 | pct87 |
| pct12 | pct90 |
| pct15 | pct95 |

## Underline patterns

*upattern*

| | |
|---|---|
| none | dashLong |
| single | dashLongHeavy |
| words | dotDash |
| double | dashDotHeavy |
| thick | dotDotDash |
| dotted | dashDotDotHeavy |
| dottedHeavy | wave |
| dash | wavyHeavy |
| dashedHeavy | wavyDouble |

## Border patterns

*bpattern*

| | |
|---|---|
| nil | thickThinMediumGap |
| single | thinThickThinMediumGap |
| thick | thinThickLargeGap |
| double | thickThinLargeGap |
| dotted | thinThickThinLargeGap |
| dashed | wave |
| dotDash | doubleWave |
| dotDotDash | dashSmallGap |
| triple | dashDotStroked |
| thinThickSmallGap | threeDEmboss |
| thickThinSmallGap | threeDEngrave |
| thinThickThinSmallGap | outset |
| thinThickMediumGap | inset |

## References

Jann, B. 2016. Creating LaTeX documents from within Stata using texdoc. *Stata Journal* 16: 245–263.

McDowell, A., A. Engel, J. T. Massey, and K. Maurer. 1981. Plan and operation of the Second National Health and Nutrition Examination Survey, 1976–1980. *Vital and Health Statistics* 1(15): 1–144.

## Also see

[P] **putexcel** — Export results to an Excel file

[P] **putpdf** — Create a PDF file

[M-5] **_docx*( )** — Generate Office Open XML (.docx) file

[M-5] **Pdf*( )** — Create a PDF file

[M-5] **xl( )** — Excel file I/O class