

```
putpdf — Create a PDF file
```

[Description](#)[Remarks and examples](#)[Quick start](#)[Appendix](#)[Syntax](#)[References](#)[Options](#)[Also see](#)

## Description

`putpdf` writes paragraphs, images, and tables to a PDF file. 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 generate various reports based on those results.

## Quick start

Create a document in memory on which subsequent contents are added

```
putpdf begin
```

Declare a paragraph to be added to the document and center the paragraph

```
putpdf paragraph, halign(center)
```

Append the text “This is paragraph text” to the paragraph declared above and format the text as bold

```
putpdf text ("This is paragraph text"), bold
```

Add a table named `tbl1` with three rows and four columns to the document

```
putpdf 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

```
putpdf table tbl1(1,2) = ("Cell 2"), halign(right)
```

Add a table named `tbl2` with variable names and estimated coefficients after `regress`

```
putpdf table tbl2 = etable
```

Add a PNG image saved as `myimg` to the document

```
putpdf paragraph  
putpdf image myimg.png
```

Save the document in memory to disk as `myfile.pdf`

```
putpdf save myfile.pdf
```

## Syntax

Create document for export

```
putpdf begin [ , document_options ]
```

Add paragraph to document

```
putpdf paragraph [ , paragraph_options ]
```

Add text to paragraph

```
putpdf text (exp) [ , text_options ]
```

Add image to paragraph

```
putpdf image filename [ , image_options ]
```

Add table to document

```
putpdf table tablename = (nrows, ncols) [ , table_options ]
```

```
putpdf table tablename = data(varlist) [if] [in] [ , varnames obsno  
table_options ]
```

```
putpdf table tablename = matrix(matname) [ , nformat(%fnt) rownames colnames  
table_options ]
```

```
putpdf table tablename = mata(matname) [ , nformat(%fnt) table_options ]
```

```
putpdf table tablename = etable[ (#1 #2 ... #n) ] [ , table_options ]
```

```
putpdf table tablename = returnset [ , table_options ]
```

Customize rows and columns

```
putpdf table tablename(i, .), row_col_options
```

```
putpdf table tablename(., j), row_col_options
```

Add and format contents of table

```
putpdf table tablename(i, j) = (exp) [ , cell_options ]
```

```
putpdf table tablename(i, j) = image(filename) [ , cell_options ]
```

```
putpdf table tablename(i, j) = table(mem_tablename) [ , cell_options ]
```

```
putpdf table tablename(i, j), cell_options
```

Add page break to document

```
putpdf pagebreak
```

Describe current document

```
putpdf describe
```

Describe table

```
putpdf describe tablename
```

Close and save document

```
putpdf save filename [, replace]
```

Close without saving

```
putpdf clear
```

*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:

<i>returnset</i>	Description
<u>escalars</u>	All returned scalars
<u>rscalars</u>	All returned scalars
<u>emacros</u>	All returned macros
<u>rmacros</u>	All returned macros
<u>ematrices</u>	All returned matrices
<u>rmatrices</u>	All returned matrices
<u>e*</u>	All returned scalars, macros, and matrices
<u>r*</u>	All returned scalars, macros, and matrices

*mem\_tablename* specifies the name of a previously created table.

*document\_options*

Description

<u>pagesize</u> ( <i>psize</i> )	set page size of document
<u>landscape</u>	set document orientation to landscape
<u>font</u> ( <i>fspec</i> )	set font, font size, and font color
<u>halign</u> ( <i>hvalue</i> )	set horizontal alignment of document
<u>margin</u> ( <i>type</i> , # [ <i>unit</i> ])	set page margins of document
<u>bgcolor</u> ( <i>color</i> )	set background color

<i>paragraph_options</i>	Description
<code>font(<i>fspec</i>)</code>	set font, font size, and font color
<code>halign(<i>hvalue</i>)</code>	set paragraph alignment
<code>valign(<i>vvalue</i>)</code>	set vertical alignment of characters on each line
<code>indent(<i>indenttype</i>, #[<i>unit</i>])</code>	set paragraph indentation
<code>spacing(<i>position</i>, #[<i>unit</i>])</code>	set spacing between lines of text
<code>bgcolor(<i>color</i>)</code>	set background color

<i>text_options</i>	Description
<code>nformat(<i>%fmt</i>)</code>	specify numeric format for text
<code>font(<i>fspec</i>)</code>	set font, font size, and font color
<code>bold</code>	format text as bold
<code>italic</code>	format text as italic
<code>script(sub   super)</code>	set subscript or superscript formatting of text
<code>strikeout</code>	strikeout text
<code>underline</code>	underline text
<code>bgcolor(<i>color</i>)</code>	set background color
<code>linebreak</code>	add a line break after text

<i>image_options</i>	Description
<code>width(#[<i>unit</i>])</code>	set image width
<code>height(#[<i>unit</i>])</code>	set image height
<code>linebreak</code>	add a line break after image

<i>table_options</i>	Description
<code>memtable</code>	keep table in memory rather than add it to document
<code>width(#[<i>unit</i>   %]   <i>matname</i>)</code>	set table width
<code>halign(<i>hvalue</i>)</code>	set table horizontal alignment
<code>indent(#[<i>unit</i>])</code>	set table indentation
<code>spacing(<i>position</i>, #[<i>unit</i>])</code>	set spacing before or after table
<code>border(<i>bspec</i>)</code>	set pattern and color for border

<i>row_col_options</i>	Description
<u>nosplit</u>	prevent row from breaking across pages
<u>addrows</u> (# [ , before   after ])	add # rows in specified location
<u>addcols</u> (# [ , before   after ])	add # columns in specified location
<u>drop</u>	drop specified row or column
<u>margin</u> ( <i>type</i> , # [ <i>unit</i> ])	set margins
<u>halign</u> ( <i>hvalue</i> )	set horizontal alignment
<u>valign</u> ( <i>vvalue</i> )	set vertical alignment
<u>border</u> ( <i>bspec</i> )	set pattern and color for border
<u>bgcolor</u> ( <i>color</i> )	set background color
<u>nformat</u> (% <i>fmt</i> )	specify numeric format for text
<u>font</u> ( <i>fspec</i> )	set font, font size, and font color
<u>bold</u>	format text as bold
<u>italic</u>	format text as italic
<u>strikeout</u>	strikeout text
<u>underline</u>	underline text

<i>cell_options</i>	Description
<u>append</u>	append objects to current content of cell
<u>margin</u> ( <i>type</i> , # [ <i>unit</i> ])	set margins of cell
<u>rowspan</u> (#)	merge cells vertically
<u>colspan</u> (#)	merge cells horizontally
<u>span</u> (# <sub>1</sub> , # <sub>2</sub> )	merge cells both horizontally and vertically
<u>halign</u> ( <i>hvalue</i> )	set horizontal alignment
<u>valign</u> ( <i>vvalue</i> )	set vertical alignment
<u>border</u> ( <i>bspec</i> )	set pattern and color for border
<u>bgcolor</u> ( <i>color</i> )	set background color
<u>nformat</u> (% <i>fmt</i> )	specify numeric format for cell text
<u>font</u> ( <i>fspec</i> )	set font, font size, and font color
<u>bold</u>	format text as bold
<u>italic</u>	format text as italic
<u>script</u> (sub   super)	set subscript or superscript formatting of text
<u>strikeout</u>	strikeout text
<u>underline</u>	underline text

*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. The default font is Helvetica.

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

*color* sets the text color.

*bspec* is

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

*bordername* specifies the location of the border.

*bpattern* is a keyword specifying the look of the border. Possible patterns are `nil` and `single`. The default is `single`. 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* 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

`putpdf 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.

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

`putpdf image filename` embeds a portable network graphics (.png) or JPEG (.jpg) file in the 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.

`putpdf 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. A maximum of 50 columns in a table is allowed.

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

`etable[#1 #2 ... #n]` adds the coefficient table from the last estimation command as a table to the active document. 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 the first and third tables from the estimation output, specify `etable(1 3)`. Note that a few estimation commands do not support the `etable` output type. See [Unsupported estimation commands](#) in [P] **putdocx** for a list of estimation commands that are not supported by `putpdf`.

`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.

`putpdf table tablename(i, j)` modifies the content and style 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, or a previously created table (usually identified by *mem\_tablename*). To modify the style of all cells on the *i*th row, specify `tablename(i, .)`; to modify the style of all cells in the *j*th column, specify `tablename(. , j)`.

## Options

Options are presented under the following headings:

- Options for putpdf begin*
- Options for putpdf paragraph*
- Options for putpdf text*
- Options for putpdf image*
- Options for putpdf table*
  - table\_options*
  - row\_col\_options*
  - cell\_options*
- Option for putpdf save*

### Options for putpdf begin

`pagesize(psize)` sets the page size of the document. *psize* may be `letter`, `legal`, `A3`, `A4`, `A5`, `B4`, or `B5`. 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.

`halign(hvalue)` sets the horizontal alignment of the document within the paragraphs, images, and tables. *hvalue* may be `left`, `right`, or `center`. The default is `halign(left)`.

`margin(type, #[unit])` sets the page margins of the document. *type* may be `top`, `left`, `bottom`, or `right`, which identify the location of the margin inside the document. The margin value *#* is measured in inches unless another *unit* is specified. This option may be specified multiple times in a single command to account for different margin settings.

`bgcolor(color)` sets the background color for the document.

### Options for putpdf paragraph

`font(fontname [, size [, color ]])` sets the font, font size, and font color for the text within 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 `putpdf paragraph` overrides font settings specified with `putpdf begin`.

`halign(hvalue)` sets the horizontal alignment of the text within the paragraph. *hvalue* may be `left`, `right`, `center`, `justified`, or `distribute`. `distribute` and `justified` 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 `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`, or `para`. `left` and `right` indent *# units* from the left or the right, respectively. `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.



`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.

`bgcolor(color)` sets the background color for the paragraph.

Specifying `bgcolor()` with `putpdf paragraph` overrides background color specifications from `putpdf begin`.

## Options for putpdf 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 within the active 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 `putpdf text` overrides all other font settings, including those specified with `putpdf begin` and `putpdf paragraph`.

`bold` specifies that the new text in the active paragraph be formatted as bold.

`italic` specifies that the new text in the active paragraph be formatted as italic.

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

`strikeout` specifies that the new text in the active paragraph have a strikeout mark.

`underline` specifies that the new text in the active paragraph be underlined.

`bgcolor(color)` sets the background color for the active paragraph.

Specifying `bgcolor()` with `putpdf text` overrides background color specifications from `putpdf begin` and `putpdf 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 putpdf 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.

`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 putpdf 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.

`width(#[unit|%])` and `width(matname)` set the table width. Any two of the types of width specifications can be combined.

`width(#[unit|%])` sets the width based on a specified value. `#` 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%)`.

`width(matname)` sets the table width based on the dimensions specified in the Stata matrix `matname`, which has contents in the form of (`#1, #2, . . . , #n`) to denote the percent of the default table width for each column. `n` is the number of columns of the table, and the sum of `#1` to `#n` must be equal to 100.

`halign(hvalue)` sets the horizontal alignment of the page within the table. `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.

`spacing(position, #[unit])` sets the spacing before or after the table. `position` may be `before` or `after`. `before` specifies the space before the top of the current table, and `after` specifies the space after the bottom of the current table. This option may be specified multiple times in a single command to account for different space settings.

`border(bordername [, bpattern [, bcolor]])` adds a single border in the location specified by `bordername`, which may be `start`, `end`, `top`, `bottom`, `insideH` (inside horizontal borders), `insideV` (inside vertical borders), or `all`. Optionally, you may change the pattern and color for the border by specifying `bpattern` and `bcolor`.

This option may be specified multiple times in a single command to accommodate different border settings. If multiple `border()` options are specified, they are applied in the order specified from left to right.

`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 a table row is displayed, a page break may 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.

`margin(type, #[unit])` sets the margins inside the cells in row  $i$  or column  $j$ . `type` may be `top`, `left`, `bottom`, or `right`, which identify the top margin, left margin, bottom margin, or right margin of the cell, respectively. This option may be specified multiple times in a single command to account for different margin settings.

`halign(hvalue)` sets the horizontal alignment of the cells in row  $i$  or column  $j$ . `hvalue` may be `left`, `right`, or `center`. The default is `halign(left)`.

`valign(vvalue)` sets the vertical alignment of the cells in row  $i$  or column  $j$ . `vvalue` may be `top`, `bottom`, or `center`. The default is `valign(top)`.

`border(bordername [ , bpattern [ , bcolor ]])` adds a single border to all cells in a given row or column in the location specified by `bordername`, which may be `start`, `end`, `top`, `bottom`, or `all`. Optionally, you may change the pattern and color for the border by specifying `bpattern` and `bcolor`.

This option may be specified multiple times in a single command to accommodate different border settings. If multiple `border()` options are specified, they are applied in the order specified from left to right.

`bgcolor(color)` sets the background color for the cells in row  $i$  or column  $j$ .

`nformat(%fmt)` specifies the numeric format of the cell text when the content of the text 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 cell text within the row or column. 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 cell text in the row or column be formatted as bold.

`italic` specifies that the cell text in the row or column be formatted as italic.

`strikeout` specifies that the cell text in the row or column have a strikeout mark.

`underline` specifies that the cell text in the row or column be underlined.

**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.

Unlike with the **putdocx** command, this option with **putpdf** is used only for appending a new string to the cell when the original cell content is also a string.

**margin**(*type*, # [*unit*]) sets the margins inside the cell. *type* may be **top**, **left**, **bottom**, or **right**, which identify the top margin, left margin, bottom margin, or right margin of the cell, respectively.

This option may be specified multiple times in a single command to account for different margin settings.

**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**(#<sub>1</sub>, #<sub>2</sub>) sets the specified cell to span #<sub>1</sub> cells downward and span #<sub>2</sub> cells to the right.

**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)**.

**border**(*bordername* [, *bpattern* [, *bcolor*]]) adds a single border to the cell in the location specified by *bordername*, which may be **start**, **end**, **top**, **bottom**, or **all**. Optionally, you may change the pattern and color for the border by specifying *bpattern* and *bcolor*.

This option may be specified multiple times in a single command to accommodate different border settings. If multiple **border()** options are specified, they are applied in the order specified from left to right.

**bgcolor**(*color*) sets the background color for the specified cell.

**nformat**(*%fmt*) specifies the numeric format of the cell text when the content of the text 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 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 in the cell be formatted as bold.

**italic** specifies that the text in the cell be formatted as italic.

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

**strikeout** specifies that the text in the cell have a strikeout mark.

**underline** specifies that the text in the cell be underlined.

**Option for putpdf save**

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

## Remarks and examples

`putpdf` is a suite of commands used to write paragraphs, images, and tables to a PDF file. It may also be used to format each object added. This allows you to automate exporting and formatting of, for example, Stata graphs and estimation results and also to generate various reports based on those elements. Hence, it provides a convenient way to include Stata results in a PDF file.

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

```
. putpdf 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 `putpdf begin`](#).

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

```
. putpdf save example.pdf
```

Note that the `replace` option is required if `example.pdf` already exists in the saving directory. If `replace` is specified, then all the contents in `example.pdf` will be overwritten. To close the document in memory and erase all elements in it without saving your work, use `putpdf clear`. `putpdf save` automatically clears the working copy of the document from memory.

Remaining remarks are presented under the following headings:

- [Add a paragraph](#)
  - [Add text to paragraph](#)
  - [Add an image to paragraph](#)
- [Add a table](#)
  - [Export data](#)
  - [Export estimation results](#)
- [Advanced uses](#)

### Add a paragraph

Before you can add text or an image to the paragraph, you must first begin a new paragraph by using `putpdf paragraph`. You can control the formatting for the whole paragraph, such as font properties and alignment, with options for `putpdf paragraph`. See [Options for `putpdf 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 `putpdf text`. The new text is appended to any text or image that has already been added to the paragraph. This text can also be formatted individually. See [Options for `putpdf 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.pdf`. 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 `putpdf 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.

```
. 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

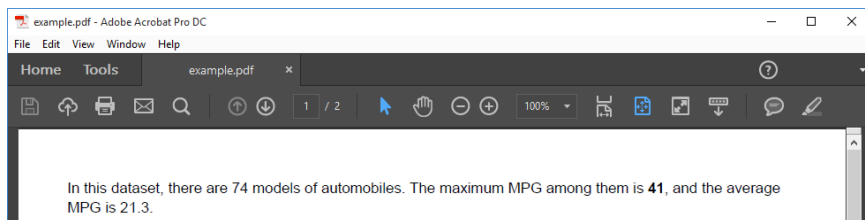
```
. return list
scalars:
      r(N) = 74
r(sum_w) = 74
r(mean)  = 21.2972972972973
r(Var)   = 33.47204738985561
r(sd)    = 5.785503209735141
r(min)   = 12
r(max)   = 41
r(sum)   = 1576
```

The returned results `r(N)`, `r(mean)`, and `r(max)` store the number of automobiles, the average MPG, and the maximum MPG among those automobiles.

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 `putpdf text` command may be a valid Stata expression (see [U] 13 [Functions and expressions](#)) or a normal string. `putpdf 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, and `r(mean)` is formatted to have 1 decimal place.

```
. putpdf begin
. putpdf paragraph
. putpdf text ("In this dataset, there are 'r(N)'"
. putpdf text (" models of automobiles. The maximum MPG among them is ")
. putpdf text (r(max)), bold
. putpdf text (" , and the average MPG is ")
. putpdf text (r(mean)), nformat("%4.1f")
. putpdf text (".")
```

This adds text to the document that looks like this:



## Add an image to paragraph

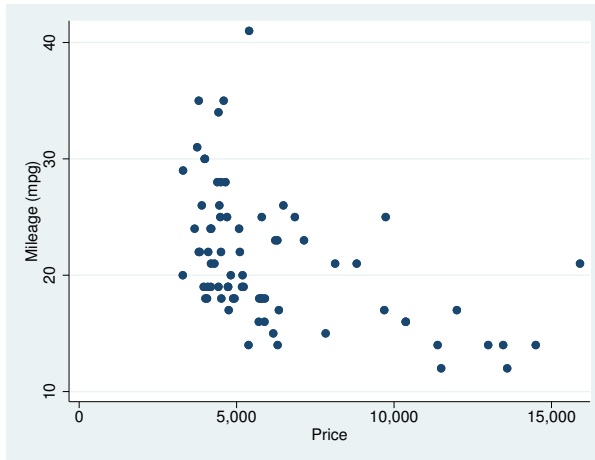
You can add any existing `.png` and `.jpg` image files to a `.pdf` file with `putpdf image`. For example, you could include a company logo. You can also add graphs from Stata output. Because Stata graphs use the `.gph` extension, you must use `graph export` to convert the Stata graph to one of the supported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

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 `putpdf 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 `putpdf 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.

```
. putpdf paragraph, halign(center)
. putpdf image auto.png, width(4)
```

Here, the image is put in the center of the document and is set to have a width of 4 inches. Note that the `halign()` option had to be specified with `putpdf paragraph` because alignment is controlled by paragraph settings.

## Add a table

`putpdf 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. We can edit the created table until another object such as a paragraph, another table, or a page break is added to the document.

## Export data

Exporting the data in memory is useful when you want to make a table in the `.pdf` 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
-----|-----|-----|-----|-----|-----|
..
```

Because we want the variable names to serve as column titles, we rename `foreign` to `Origin`. 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
. putpdf table tbl1 = data("Origin Total Average Max Min"), varnames
> border(start, nil) border(end, nil) border(insideV, 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

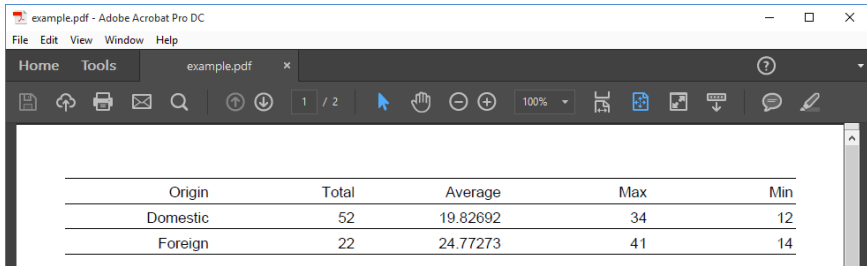
The cells in the table that we created in [example 3](#) can be further customized. For example, we can reset the contents, set the text alignment, modify the borders, and so forth. Here, we set the horizontal text of the cells on the first through the fifth columns to be right aligned instead of the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



To format all the cells in a column, we can simply specify the row index as “.” in the  $(i, j)$  specification for a cell. We loop over the first through fifth column indices using `forvalues`.

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

Our formatted `tbl1` looks like this:



Origin	Total	Average	Max	Min
Domestic	52	19.82692	34	12
Foreign	22	24.77273	41	14

Afterward, we restore the dataset.

```
. restore
```

◀

## Export estimation results

One of the primary uses of `putpdf 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`.

```
. 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

We want to add these regression results to the document. For most estimation commands, we can use the `etable` output type to easily add the elements of the displayed coefficient table. To export all columns of the default `regress` output, we need type only

```
. putpdf table reg = etable
```

But we need not stop there. In [example 5](#), we select a subset of the results and format them.

### ► Example 5: Export selected estimation results

The `etable` output type makes it easy to export the whole estimation table with a simple one-line command. First, we create a new table, `tbl2`, that contains the estimation results from `regress`. We specify the `width` option to ensure that the table occupies the full page width of the document.

Suppose we want to export only the point estimates and confidence intervals from the table above. We can then remove the standard error,  $t$  statistic, and  $p$ -value columns that we do not want by dropping the third through the fifth columns. We work from right to left, dropping column five before four, because in this manner, the column numbers to the left of the newly dropped column do not change. To drop the fifth column, we specify `tbl2(.,5)` followed by the drop option.

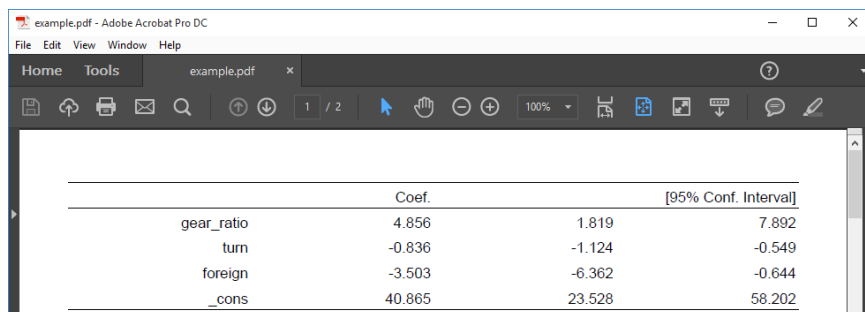
```
. putpdf table tbl2 = etable, width(100%)
. putpdf table tbl2(.,5), drop //drop p-value column
. putpdf table tbl2(.,4), drop //drop t column
. putpdf table tbl2(.,3), drop //drop SE column
```

Equivalently, we could have dropped the third column three times, because each time we drop it, the previous fourth column becomes the third.

Now that we have only the statistics we want, we can format our table by removing the border on the right side of the first column. To do this, we specify `nil` as the border pattern for the right border. We also format all estimates to have three decimal places, looping over what will now be columns two through four to do so. Finally, we erase the text “mpg” from the header for the first column.

```
. putpdf table tbl2(., 1), border(right, nil)
. forvalues col=2/4 {
2.     putpdf table tbl2(.,'col'), nformat(%9.3f)
3. }
. putpdf table tbl2(1,1) = ("") // erase the content of first cell "mpg"
```

Our final table appears in the document as follows:



	Coef.	[95% Conf. Interval]	
gear_ratio	4.856	1.819	7.892
turn	-0.836	-1.124	-0.549
foreign	-3.503	-6.362	-0.644
_cons	40.865	23.528	58.202

In this example, because we wanted to use the same number of decimals for all estimates in our table and because we are using the `etable` output type, we could have preemptively set the format with our `regress` command. The modified version of the command is

```
. regress mpg gear_ratio turn foreign, noheader cformat(%9.3f)
```

This avoids the need to apply a numeric format to the text of the columns individually.

## Advanced uses

The [previous](#) example demonstrated the use of the `etable` output type for exporting an estimation table to a `.pdf` file. While this method is efficient for extracting large portions of the estimation results, exporting results to highly customized tables with complicated layouts can at times be done more easily from a matrix of stored results. See [\[U\] 14 Matrix expressions](#) for help with matrix notation.

A small set of estimation commands do not support the `etable` output type; however, matrices of stored results can be exported when using these commands. For a list of estimation commands that do not support the `etable` output type, see [Unsupported estimation commands](#) in [\[P\] putdocx](#).

Finally, the dimension of the table and the content of each cell may not be fixed or may change when commands are executed with different data. In this situation, consider building the table in pieces and combining them.

### ► Example 6: Export selected estimation results from a matrix

To illustrate the basic use of matrix manipulation of stored results to create an estimation table, we re-create the simple estimation table from [example 5](#). The displayed results returned by `regress` are stored in the matrix `r(table)`, which can be viewed by typing `matrix list`.

```
. 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
```

First, we create the matrix `rtable` as the transpose of `r(table)` because we want to see the variable names in rows. 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 rtable = 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 `tbl3`.

```
. putpdf table tbl3 = matrix(r_table), nformat(%9.3f) rownames colnames
> border(start, nil) border(end, nil) border(insideH, nil) border(insideV, 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 column and first row 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 (`b`, `ll`, and `ul`) 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.

```
. putpdf table tbl3(1,2) = ("Coef."), halign(right)
. putpdf table tbl3(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 rows two through five; see [P] [forvalues](#).

```
. putpdf table tbl3(1,.), border(bottom)
. forvalues i = 2/5 {
2.     putpdf table tbl3('i',.), halign(right)
3. }
```

Our final table will be identical to that shown in [example 5](#).

◀

## ▷ Example 7: Create a table from components

We can also build tables in pieces and then combine them. For illustrative purposes, we again use the table created in [example 5](#) and [example 6](#). The table can be considered to comprise two parts, the header and the body. We first create a table for each part. Notice that both tables will be created with the `memtable` option, which declares that they be created in memory rather than added to the document. This is required so we can add the tables as components to the final table.

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

```
. putpdf table tbl41 = (1,2), memtable border(all, nil)
. putpdf table tbl41(1,1) = ("Coef."), halign(right)
. putpdf table tbl41(1,2) = ("[95% Conf. Interval]"), halign(right)
```

The body part contains the content that we want displayed. It is created as another table using the same Stata matrix from [example 6](#), `r_table`. Again, we remove all borders and align all contents to the right.

```
. putpdf table tbl42 = matrix(r_table), memtable border(all, nil)
> nformat(%9.3f) rownames
. forvalues i=1/4 {
2.     putpdf table tbl42('i',.), halign(right)
3. }
```

Finally, we create a  $2 \times 1$  table named `tbl4`, in which the first cell contains `tbl41` and the second cell contains `tbl42`. In addition, we remove its leading edge border and trailing edge border.

```
. putpdf table tbl4 = (2,1), border(start, nil) border(end, nil)
. putpdf table tbl4(1,1) = table(tbl41)
. putpdf table tbl4(2,1) = table(tbl42)
```

This is an example of what is called a nested table.

◀

When we create a table from components, the dimensions are 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.

In [example 7](#), we created a table as two parts. The first part was the header and the second was the body that contained the estimation results. Each of the parts was a table created in two steps. The first step created the table, and the second added the content to it. In our third step, we added our two-component tables to a third table, itself first created and formatted before adding the contents.

To create a table dynamically, we again start by creating a table with only basic formatting. But rather than adding content all at once, we add content cell by cell and variable by variable. As we add the content, we apply any specialized formatting for the cells, rows, or columns.

### ► Example 8: Create a table dynamically

Below we use Census data recording the death rate (`drate`) and median age (`medage`) for each state. The data also record four regions of the country in which each state is located, NE, N Cntrl, South, and West. We fit a linear regression model and store the transpose of rows 1, 5, and 6 in column matrices named `b`, `ll`, and `ul`.

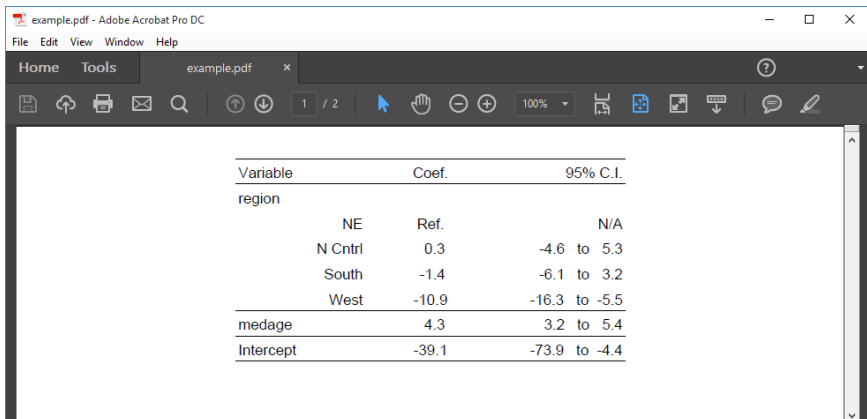
```
. use http://www.stata-press.com/data/r15/census9
(1980 Census data by state)

. regress drate i.region medage [aw=pop], noheader
(sum of wgt is 2.2591e+08)
```

	drate	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
region							
N Cntrl		.3138738	2.456431	0.13	0.899	-4.633632	5.26138
South		-1.438452	2.320244	-0.62	0.538	-6.111663	3.234758
West		-10.90629	2.681349	-4.07	0.000	-16.30681	-5.505777
medage		4.283183	.5393329	7.94	0.000	3.196911	5.369455
_cons		-39.14727	17.23613	-2.27	0.028	-73.86262	-4.431915

```
. matrix rtable = r(table)
. matrix b = rtable[1,1...]'
. matrix ll = rtable[5,1...]'
. matrix ul = rtable[6,1...]'
```

Our goal is to collect the point estimates and confidence intervals and output a table that looks like the following:



The screenshot shows a PDF document titled 'example.pdf' in Adobe Acrobat Pro DC. The document contains a table with the following content:

Variable	Coef.	95% C.I.
region		
NE	Ref.	N/A
N Cntrl	0.3	-4.6 to 5.3
South	-1.4	-6.1 to 3.2
West	-10.9	-16.3 to -5.5
medage	4.3	3.2 to 5.4
Intercept	-39.1	-73.9 to -4.4

We want to start our new table on its own page, so we insert a page break before adding our table.

```
. putpdf pagebreak
```

To create our table of point estimates and confidence intervals, we first create a  $1 \times 3$  table with no borders and fill the single row with Variable, Coef., and 95% C.I.. We also set the table width to be 4 inches, put the table in the center of the document, and add back the top and bottom borders.

```
. putpdf table tbl5 = (1,3), border(all,nil) width(4) halign(center)
. putpdf table tbl5(1,1)="Variable", border(top) border(bottom)
. putpdf table tbl5(1,2)="Coef.", halign(center) border(top) border(bottom)
. putpdf table tbl5(1,3)="95% C.I.", halign(right) border(top) border(bottom)
```

Afterward, we add one row to the end of the table and fill in the content of the first column in this row as region.

```
. putpdf table tbl5(1,:), addrows(1)
. putpdf table tbl5(2,1)="region"
```

In the resulting table, the region variable has four levels, and each level takes up one row. The medage variable and the constant term take up another two rows. For each row, the first column contains the variable label, the second column contains the point estimate that is stored in the matrix `b`, and the third column contains a formatted string of the lower limit and upper limit of the confidence intervals. Those two levels are stored in matrix `ll` and `ul`, respectively.

Based on the above information, we add those rows one by one at the end of the table and fill in the content and format for each cell in the corresponding row. Notice that NE is the base level of region; its point estimate and confidence interval are replaced by Ref. and N/A in the resulting table. In addition, we reset the top and bottom borders for medage and Intercept.

```
. local row 2
. local i 1
. foreach name in "NE" "N Cntrl" "South" "West" "medage" "Intercept" {
2.   putpdf table tbl5('row',.), addrows(1)
3.   local ++row
4.   if "'name'"=="NE" {
5.     local coef "Ref."
6.     local ci "N/A"
7.   }
8.   else {
9.     local coef : display %5.1f b['i',1]
10.    local low : display %5.1f ll['i',1]
11.    local upp : display %5.1f ul['i',1]
12.    local ci "'low' to 'upp'"
13.  }
14.  putpdf table tbl5('row', 1) = ("'name'"), halign(right)
15.  putpdf table tbl5('row', 2) = ("'coef'"), halign(center)
16.  putpdf table tbl5('row', 3) = ("'ci'"), halign(right)
17.  if "'name'"=="medage" | "'name'"=="Intercept" {
18.    putpdf table tbl5('row', 1), halign(left) border(top)
> border(bottom)
19.    putpdf table tbl5('row', 2), border(top) border(bottom)
20.    putpdf table tbl5('row', 3), border(top) border(bottom)
21.  }
22.  local ++i
23. }
```

## ► Example 9: 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.

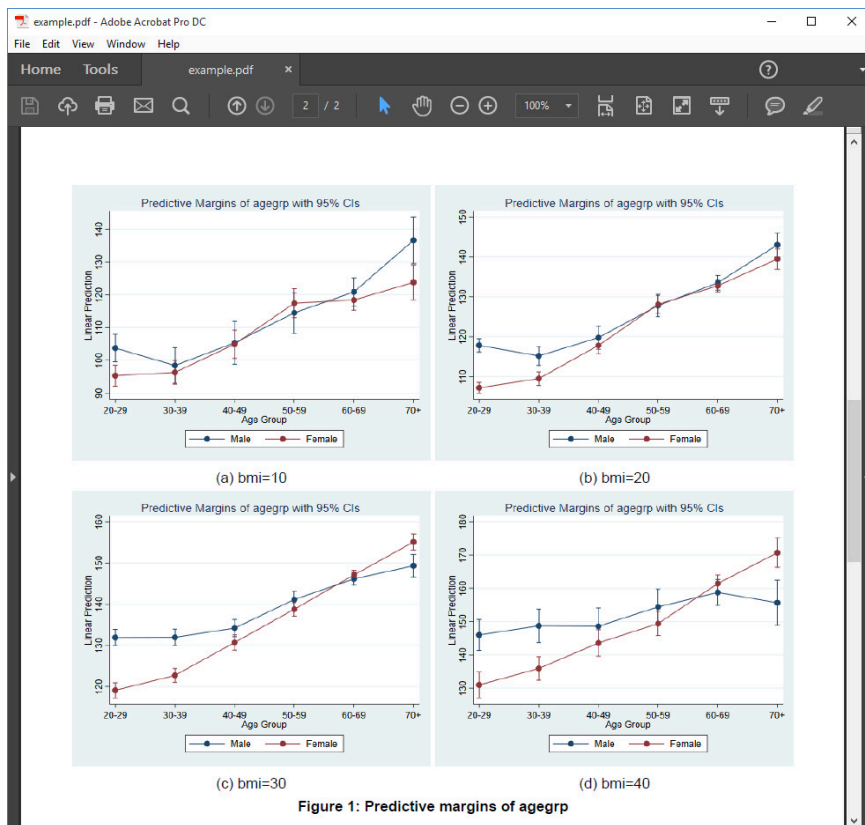
```
. use http://www.stata-press.com/data/r15/nhanes2
. regress bpsystol agegrp##sex##c.bmi
  (output omitted)
. forvalues v=10(10)40 {
  2.     margins agegrp, over(sex) at(bmi='v')
  3.     marginsplot
  4.     graph export bmi'v'.png
  5. }
  (output omitted)
```

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 `putpdf table`.

We start with a  $4 \times 2$  table and remove all of its borders. In each cell on the odd rows, we add a plot, and we fill each cell on the even rows with the title corresponding to the plot above it and center-align the text in the cell. Finally, we add another row to the end of the table and fill it with the title for the figure.

```
. putpdf table tbl6 = (4,2), border(all,nil)
. putpdf table tbl6(1,1)=image(bmi10.png)
. putpdf table tbl6(2,1)="(a) bmi=10", halign(center)
. putpdf table tbl6(1,2)=image(bmi20.png)
. putpdf table tbl6(2,2)="(b) bmi=20", halign(center)
. putpdf table tbl6(3,1)=image(bmi30.png)
. putpdf table tbl6(4,1)="(c) bmi=30", halign(center)
. putpdf table tbl6(3,2)=image(bmi40.png)
. putpdf table tbl6(4,2)="(d) bmi=40", halign(center)
. putpdf table tbl6(4,.), addrows(1)
. putpdf table tbl6(5,1)="Figure 1: Predictive margins of agegrp",
> halign(center) colspan(2) bold
```

This creates a table that looks like the following:



We can now type

```
. putpdf save example.pdf
```

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



# Appendix

## 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	

---

## 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] **putdocx** — Generate Office Open XML (.docx) 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