Title

stata.com

putpdf collect — Add a table from a collection to a PDF file

Description

`putpdf collect` allows you to export a customized table from a collection to a table in the active PDF file. A collection contains a set of results that have been collected from one or more Stata commands using the `collect:` prefix or the `collect get` command. With the suite of `collect` commands, you can specify the layout and style of your table and customize the table.

Unlike `putpdf table`, which allows you to create tables and modify them as needed, `putpdf collect` is designed for exporting a table that you have already finalized using the `collect` suite of commands. To learn more about creating customized tables, see [TABLES] **Intro**.

`putpdf collect` also allows you to include tables created by `table` and `etable` in your report. The `table` command is a powerful command for producing tabulations, tables of summary statistics, tables of regression results, and more. The `etable` command creates tables with the active estimation results, results from `margins`, and results stored with `estimates store`. `table` and `etable` are unique in that they automatically create a collection. Their results can be further styled using the `collect` commands, or they can be included in your report as is with `putpdf collect`. See [R] **table intro** and [R] **etable** for more information on these commands.

Quick start

Create a table in the document using items from the current collection

```
putpdf collect
```

As above, and display the `putpdf` commands used to export to the PDF file

```
putpdf collect, noisily
```

As above, but instead of displaying the commands, save them to the file `myfile.do`

```
putpdf collect, dofile(myfile.do)
```

Syntax

`putpdf collect` [, *options*]

options	Description
`name(`*cname*`)`	use collection *cname*
`memtable`	keep table in memory rather than add it to document
`noisily`	show the `putpdf` commands used to export to the PDF file
`dofile(`*filename*[`, replace`]`)`	save the `putpdf` commands used for exporting to the named do-file
`tablename(`*tablename*`)`	specify a name for the table

Options

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

`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. 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.

`noisily` specifies that `putpdf collect` show the `putpdf` commands used to export to the PDF file.

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

`tablename(`*tablename*`)` specifies a name for the table. By naming the table, you can make further edits with `putpdf table`. The name must be a valid name according to Stata's naming conventions; see **[U] 11.3 Naming conventions**.

If the current collection contains multiple tables, the table names will contain the prefix *tablename* and an integer as the suffix. For example, if you specify `tablename(myreg)` and the collection contains three tables, the table names will be `myreg1`, `myreg2`, and `myreg3`. Also, note that a name is a sequence of 1 to 32 letters, digits, and underscores. If you are exporting multiple tables, consider using a shorter sequence to account for the integer suffix.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction

`putpdf collect` exports a table from a collection to the active PDF file. A collection is a set of results that have been obtained from one or more Stata commands with the `collect` suite of commands, from the `table` command, or from the `etable` command.

To create a table with the `collect` commands, you first collect results from one or more Stata commands. Then you create a table by specifying a layout that determines which results are to be included in the table and how they are to be arranged. For example, you can have the rows correspond to variables and the columns correspond to the statistics, or vice versa. You can also modify the labels in the table, format the results, add borders, change the font style, and make other styling changes to the table. Once you have finalized your table, you can add it to your active PDF file with `putpdf collect`.

It is possible to have many collections in memory. However, there is only one current collection, the one you are working with. The current collection is the one that will be exported with `putpdf collect`, but you can export a table from another collection by specifying the `name()` option.

A table that is conveniently added to a PDF file using `putpdf collect` could equivalently be added using a series of `putpdf table` commands. To see what those commands look like, use the `noisily` option with `putpdf collect`. This long list will quickly fill up your Results window, so you may want to store those commands in a do-file instead by using the `dofile()` option with `putpdf collect`. For reproducibility purposes, you may choose to include your `collect` and `putpdf collect` commands in your do-file, or you may instead incorporate the commands from the do-file created by the `dofile()` option.

While the collect commands provide many generic styling options that will be applied to the table you export to your PDF file, the suite also includes collect style putpdf, which provides styling options specifically for tables exported to the PDF format. You can use collect style putpdf to specify the alignment of the table on the page, specify the table indentation, and more.

In the next section, we will show you how to collect results from a Stata command, customize the table with those results, and export the table to a PDF file. In these examples, we assume some familiarity with the table and collect commands. We recommend that you see [R] **table intro** for information on table and [TABLES] **Intro** for more information on collect to learn more about creating and customizing tables before incorporating them into your PDF file.

Export a table with items from a collection

In this example, we will use data from the Second National Health and Nutrition Examination Survey (NHANES II) (McDowell et al. 1981). Suppose we want to model the occurrence of high blood pressure (highbp) and then create a table in our PDF file that lists the estimated probabilities for different groups of people. Below, we begin by loading the data and fitting the logistic regression model:

```
. use https://www.stata-press.com/data/r17/nhanes2, clear
. logistic highbp i.sex i.race i.agegrp
Logistic regression                                     Number of obs =  10,351
                                                        LR chi2(8)    = 1599.71
                                                        Prob > chi2   =  0.0000
Log likelihood = -6250.912                              Pseudo R2     =  0.1134
```

highbp	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
sex						
Female	.6463459	.0279512	-10.09	0.000	.59382	.703518
race						
Black	1.673928	.1179769	7.31	0.000	1.457958	1.921891
Other	1.321562	.2075301	1.78	0.076	.971449	1.797857
agegrp						
30-39	1.957614	.1543556	8.52	0.000	1.677301	2.284775
40-49	3.316726	.2668563	14.90	0.000	2.832851	3.88325
50-59	6.117762	.4864457	22.78	0.000	5.234924	7.149485
60-69	7.239449	.4920962	29.12	0.000	6.336446	8.271139
70+	10.59802	.9377584	26.68	0.000	8.910598	12.605
_cons	.2316432	.013944	-24.30	0.000	.205864	.2606506

Note: **_cons** estimates baseline odds.

Then before collecting any results, we clear out all styles, including the default style. When you collect results from a Stata command, the collection will have a default style. For example, a border will be added between the row headers and the results. For this table, we would like to clear all those specifications and begin with an empty style. The next step is to use margins to obtain the expected probabilities of having high blood pressure. We prefix the margins command with collect: to collect the statistics that are computed.

```
. collect style clear
. collect: margins sex race agegrp
Predictive margins                                      Number of obs = 10,351
Model VCE: OIM
Expression: Pr(highbp), predict()
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
sex						
Male	.4708373	.0065793	71.56	0.000	.457942	.4837326
Female	.3794041	.0061263	61.93	0.000	.3673968	.3914114
race						
White	.4104461	.0047768	85.93	0.000	.4010838	.4198084
Black	.5189607	.0139876	37.10	0.000	.4915455	.546376
Other	.468975	.0328331	14.28	0.000	.4046234	.5333267
agegrp						
20-29	.1668675	.0076677	21.76	0.000	.151839	.1818959
30-39	.2798711	.0110569	25.31	0.000	.2581999	.3015423
40-49	.3949581	.0135814	29.08	0.000	.3683391	.4215771
50-59	.5436194	.0137415	39.56	0.000	.5166866	.5705522
60-69	.5842869	.0091338	63.97	0.000	.5663851	.6021888
70+	.6716042	.0148123	45.34	0.000	.6425726	.7006357

Now these results are stored in a collection, and we can arrange them in a few different ways with `collect layout`. Below, we specify that we want a single table, where the rows correspond to the variable names (`colname`) and the columns correspond to the statistics (`result`). While many statistics are collected from the `margins` command, we only want to include the probabilities and confidence intervals in our table. Each statistic is a level, or component, of the dimension `result`. And we can include levels of a dimension by specifying them within brackets next to the dimension name. `_r_b` represents the probabilities, and `_r_ci` represents the confidence intervals.

```
. collect layout (colname) (result[_r_b _r_ci])
Collection: default
      Rows: colname
   Columns: result[_r_b _r_ci]
   Table 1: 11 x 2
```

			Result Coefficient	Result 95% CI	
Covariate names and column names	Sex	Male	.4708373	.457942	.4837326
Covariate names and column names	Sex	Female	.3794041	.3673968	.3914114
Covariate names and column names	Race	White	.4104461	.4010838	.4198084
Covariate names and column names	Race	Black	.5189607	.4915455	.546376
Covariate names and column names	Race	Other	.468975	.4046234	.5333267
Covariate names and column names	Age group	20-29	.1668675	.151839	.1818959
Covariate names and column names	Age group	30-39	.2798711	.2581999	.3015423
Covariate names and column names	Age group	40-49	.3949581	.3683391	.4215771
Covariate names and column names	Age group	50-59	.5436194	.5166866	.5705522
Covariate names and column names	Age group	60-69	.5842869	.5663851	.6021888
Covariate names and column names	Age group	70+	.6716042	.6425726	.7006357

If we were to export the collection right now, the table above is what we would see in the PDF file. But `collect` has many tools for customizing this table, so let's clean up the headers and format the results before exporting.

Our table reports probabilities, not coefficients, so below we use `collect label levels` to modify the label for the level `_r_b` accordingly. Also, we have two dimensions in this table, `result`

and colname. You may have noticed the repeating titles for both of these dimensions; we can remove them with collect style header:

```
. collect label levels result _r_b "Prob.", modify
. collect style header, title(hide)
```

With collect style cell, we can make changes to a single cell, multiple levels of a dimension, or all cells in a dimension. First, we use the cidelimiter() option to specify that we want to use a comma to separate the upper and lower bounds of each confidence interval. We also use the sformat() option to place square brackets around the confidence intervals. We only need to apply these options to the level _r_ci of the dimension result. Then, for all cells with numeric content, we specify that only three digits should be displayed after the decimal.

```
. collect style cell result[_r_ci], cidelimiter(", ") sformat("[%s]")
. collect style cell, nformat(%5.3f)
. collect layout
Collection: default
      Rows: colname
   Columns: result[_r_b _r_ci]
   Table 1: 11 x 2
                Prob.         95% CI
     Sex   Male 0.471 [0.458, 0.484]
     Sex Female 0.379 [0.367, 0.391]
    Race  White 0.410 [0.401, 0.420]
    Race  Black 0.519 [0.492, 0.546]
    Race  Other 0.469 [0.405, 0.533]
Age group 20-29 0.167 [0.152, 0.182]
Age group 30-39 0.280 [0.258, 0.302]
Age group 40-49 0.395 [0.368, 0.422]
Age group 50-59 0.544 [0.517, 0.571]
Age group 60-69 0.584 [0.566, 0.602]
Age group   70+ 0.672 [0.643, 0.701]
```

When we type collect layout without any arguments, we get a report of the current layout. This collection looks nice, but there are just a few minor things we can do to make it complete. First, notice that the row headers Sex, Race, and Age group are repeated for all the levels of the factor variables. We can use collect style row to hide all but the first of the duplicate row headers. Then, we add some borders. border_block is another dimension of the table and it divides the table into four blocks: row header, column header, corner (top left area), and item (which contains the results).

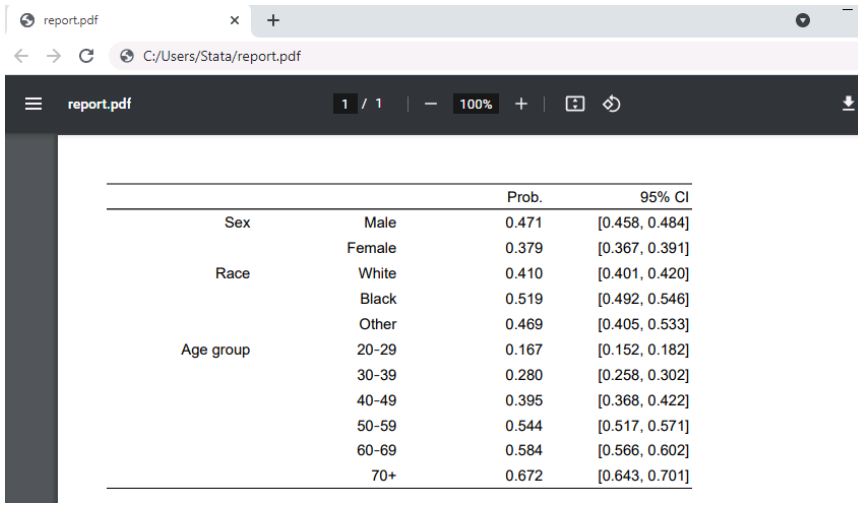
```
. collect style row split, dups(first)
. collect style cell border_block[row-header], border(top) border(bottom)
. collect style cell border_block[column-header], border(top) border(bottom)
. collect style cell border_block[corner], border(top)
. collect style cell border_block[item], border(bottom)
```

If you find yourself modifying borders or making other style changes in the same way often enough, you do not need to type these commands every time you create a table. Instead, you can save the style specifications with collect style save. Then you can apply that style to your tables with collect style use.

We are almost ready to export our polished collection to a PDF file. The last thing we will do is set the width of the table to 80% of the default table width; we can make this change with collect style putpdf. Then we will create a document in memory and export the collection.

```
. collect style putpdf, width(80%)
. putpdf begin
. putpdf collect
(collection default posted to putpdf)
. putpdf save report, replace
successfully created "C:/mypath/report.pdf"
```

After saving our work, `report.pdf` contains the following:



		Prob.	95% CI
Sex	Male	0.471	[0.458, 0.484]
	Female	0.379	[0.367, 0.391]
Race	White	0.410	[0.401, 0.420]
	Black	0.519	[0.492, 0.546]
	Other	0.469	[0.405, 0.533]
Age group	20-29	0.167	[0.152, 0.182]
	30-39	0.280	[0.258, 0.302]
	40-49	0.395	[0.368, 0.422]
	50-59	0.544	[0.517, 0.571]
	60-69	0.584	[0.566, 0.602]
	70+	0.672	[0.643, 0.701]

Stored results

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

Macros

`s(collection)`	name of collection
`s(dofile)`	name of the new do-file

References

Huber, C. 2021. Customizable tables in Stata 17, part 6: Tables for multiple regression models. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2021/09/02/customizable-tables-in-stata-17-part-6-tables-for-multiple-regression-models/.

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

[RPT] **putpdf intro** — Introduction to generating PDF files

[RPT] **putpdf begin** — Create a PDF file

[RPT] **putpdf pagebreak** — Add breaks to a PDF file

[RPT] **putpdf paragraph** — Add text or images to a PDF file

[RPT] **putpdf table** — Add tables to a PDF file

[R] **etable** — Create a table of estimation results

[R] **table intro** — Introduction to tables of frequencies, summaries, and command results

[TABLES] **Intro** — Introduction

[TABLES] **collect style putpdf** — Collection styles for putpdf