collect get — Collect results from a Stata command

Description Quick start Menu Syntax
Options Remarks and examples Also see

Description

The `collect get` command and `collect` prefix put results into a collection. Collected results are scalars, macros, and matrices from `e()` and `r()` as well as scalar and matrix expressions. Tables can be constructed from the results in the collection.

The `collect get` command identifies results from a previous Stata command that are to be put into a collection. The `collect` prefix puts results returned by the prefixed command into a collection.

Both the `collect get` command and the `collect` prefix allow you to specify a list of results to add to the automatic `result` levels (automatic levels) for subsequent table layouts. Specifying automatic levels at the time you collect results is an alternative to selecting the results to include at the time you lay out your table; see [TABLES] **collect layout**.

Quick start

Consume results from the regression model, and place them in the current collection

```
collect: regress y x
```

Same as above, and also add coefficients `_r_b` and standard errors `_r_se` to the list of automatic results

```
collect _r_b _r_se: regress y x
```

Fit a linear regression for each level of `catvar`, collect `e()` results from each regression, and add statistics `e(r2)` and `e(r2_a)` to the automatically included results

```
by catvar: collect e(r2) e(r2_a): regress y x
```

Consume results from `r()` results, and add statistics `r(stat1)` and `r(stat2)` to the list of automatic levels

```
collect get r(stat1) r(stat2)
```

Same as above, but place the results in collection `c2`

```
collect get r(stat1) r(stat2), name(c2)
```

Same as above, and attach the tags `dim1[lev1]` and `dim2["lev 2"]` to `r(stat1)` and `r(stat2)`

```
collect get r(stat1) r(stat2), name(c2) tags(dim1[lev1] dim2["lev 2"])
```

Menu

Statistics > Summaries, tables, and tests > Tables and collections > Collect results

Syntax

Basic prefix syntax to consume results from Stata commands

`collect` [`get`]: *command*

Full prefix syntax to consume results from Stata commands

[*prefix* `...:`] `collect` [`get`] [*resultlist*] [*if*] [*in*] [, `tags(`*tags*`)` `commands(`*commands*`)`]: *command*

Consume results after running a Stata command

`collect get` *resultlist* [, `name(`*cname*`)` `tags(`*tags*`)` `commands(`*commands*`)`]

where *prefix* may be `by`, `capture`, `frame`, `noisily`, `quietly`, or `version`.

where *resultlist* is

result [*result* [...]]

result, when specified with the `collect` prefix, identifies individual results to be added to the list of automatic results.

result, when specified with `collect get`, indicates the type of results to be stored and identifies the results to add to the list of automatic results.

result may be one of the following:

result	Description
`r()`	all `r()` results
`e()`	all `e()` results
explist	returned results and named expressions

Specifying any `r()` result will also cause `collect` to consume all `r()` results. Specifying any `e()` result will also cause `collect` to consume all `e()` results.

explist specifies which results to collect. *explist* may include *result identifiers* and *named expressions*.

result identifiers are results stored in `r()` and `e()` by the *command*. For instance, *result identifiers* could be `r(mean)`, `r(C)`, or `e(chi2)`. After estimation commands, *result identifiers* also include the following:

Identifier	Result
`_r_b`	coefficients or transformed coefficients reported by *command*
`_r_se`	standard errors of `_r_b`
`_r_z`	test statistics for `_r_b`
`_r_z_abs`	absolute values of `_r_z`
`_r_df`	degrees of freedom for `_r_b`
`_r_p`	*p*-values for `_r_b`
`_r_lb`	lower bounds of confidence intervals for `_r_b`
`_r_ub`	upper bounds of confidence intervals for `_r_b`
`_r_ci`	confidence intervals for `_r_b`
`_r_cri`	credible interval (CrI) of Bayesian estimates
`_r_crlb`	lower bound of CrI of Bayesian estimates
`_r_crub`	upper bound of CrI of Bayesian estimates

named expressions are specified as *name* = *exp*, where *name* may be any valid Stata name and *exp* is an expression, typically an expression that involves one or more *result identifiers*. Expressions with spaces must be bound in parentheses. Examples of named expressions are `diff = (r(mean_2) - r(mean_1))` and `sd = sqrt(r(variance))`.

tags is a dimension and its corresponding level, or it is a space-separated list of dimensions and their corresponding levels:

dimname[*levvalue*] [*dimname*[*levvalue*] ...]

If *levvalue* contains spaces, it must be enclosed in double quotes.

Options

Main

`name(`*cname*`)` specifies the collection into which results will be saved, instead of the current collection.

Options

`tags(`*tags*`)` specifies additional tags to attach to the results being consumed. Each tag takes on the form *dimname* [*levvalue*]; to specify multiple tags separate them with a space. The following dimension names are not allowed in `tags()`: `border_block`, `cell_type`, `program_class`, and `result_type`.

`commands(`*commands*`)` specifies command names that posted the results being consumed but does not change what results are collected. This option exists to aid the search for command-specific result labels.

Unlike the `collect:` prefix, `collect get` cannot determine the command that posted the results being collected, so this option allows you to help it search for command-specific result labels. This option is also allowed by the `collect:` prefix; this is useful, for example, when the prefixed *command* is a custom program that employs official Stata commands to post its results.

Remarks and examples

Remarks are presented under the following headings:

Introduction

The first step in creating a table using `collect` is to collect the results that you wish to display in the table from Stata commands. The `collect get` command and the `collect` prefix consume results from a Stata command and place them in a collection.

To collect results, we can type either

```
.  command
. collect get  results
```

or

```
. collect: command
```

These two methods are almost equivalent. They differ in how they determine which results are to be collected. The `collect` prefix determines which results are stored by the *command* and puts all of their values into the collection. On the other hand, when `collect get` is used after *command*, we must specify results that are to be collected. We can do this in a generic way. For instance, to fit a regression model with *command* and collect results, we can type

```
.  command
. collect get e()
```

and all results stored in `e()` will be collected. However, we can be more specific about the results to be collected. If we wish to include coefficients, referred to as `_r_b`, and standard errors, referred to as `_r_se`, in the tables we are about to create, we can indicate this by typing

```
.  command
. collect get _r_b _r_se
```

All the results in `e()` will still be added to the collection. The difference is that we have now flagged `_r_b` and `_r_se` as results to be automatically included in the tables we create. For instance, if we now specify a table layout by typing

```
. collect layout (colname) (result)
```

the coefficients and standard errors will appear in the table. See [TABLES] **collect layout** for information on specifying the table layout. For our purposes here, the important issue is that we have specified that the statistics, denoted by `result`, are going to be placed on the columns. Because we specified `_r_b` and `_r_se` in our `collect get` command, these are the two statistics that will be reported. We refer to these selected levels as “automatic levels”. The automatic levels simplify the table layout specification, but we are not limited by the results we flag as automatic levels. If we, for instance, decide that we want to include confidence intervals (`_r_ci`) instead of standard errors, we can state this directly in our `collect layout` command.

```
. collect layout (colname) (result[_r_b _r_ci])
```

Above, we selected automatic levels using `collect get` after fitting a model. We can similarly select automatic levels with the `collect` prefix. For instance, we can type

```
. collect _r_b _r_se: command
```

The effects of specifying automatic levels with the prefix are the same as specifying automatic levels with the `collect get` command. Automatic levels can be changed at any time via the command `collect style autolevels`; see [TABLES] **collect style autolevels**.

Support for other prefix commands

Some prefix commands are supported as a prefix to `collect`, others are supported as a prefix to the *command* being executed, and a few are not supported in either form. The following sections provide more details on which prefix commands are supported with the `collect` prefix.

Fully supported

The following prefix commands can be specified as a prefix to the `collect` prefix or as a prefix within *command*.

Command name	Entry
`by`	[D] **by**
`noisily`	[P] **quietly**
`quietly`	[P] **quietly**
`version`	[P] **version**

For example, you can type

```
by var1: collect: regress y x1 x2
```

or

```
collect: by var1, sort: regress y x1 x2
```

Likewise, `noisily`, `quietly`, and `version` may be specified before or after the `collect` prefix.

For the `by` prefix, the levels of the `by` variables are also collected as part of each results set. `by` is not allowed to be specified simultaneously as a prefix to the `collect` prefix and within *command*.

Partially supported

The following commands are allowed to be specified as a prefix to the `collect` prefix but are not allowed as a prefix in *command*.

Command name	Entry
`capture`	[P] **capture**
`frame`	[D] **frame prefix**
`xi`	[R] **xi**

For example, you can type

```
    frame fr1: collect: regress y x1 x2
```

but not

```
    collect: frame fr1: regress y x1 x2
```

The following commands are not allowed to be specified as a prefix to the `collect` prefix but are allowed as a prefix in *command*.

Command name	Entry
`bayes`	[BAYES] **bayes**
`bootstrap`	[R] **bootstrap**
`fmm`	[FMM] **fmm**
`fp`	[R] **fp**
`jackknife`	[R] **jackknife**
`mfp`	[R] **mfp**
`mi estimate`	[MI] **mi estimate**
`permute`	[R] **permute**
`svy`	[SVY] **svy**

For example, you can type

```
    collect: bayes: regress y x1 x2
```

but not

```
    bayes: collect: regress y x1 x2
```

Not supported

The following commands are not allowed to be specified as a prefix to the `collect` prefix, nor are they allowed as a prefix in *command*.

Command name	Entry
`nestreg`	[R] **nestreg**
`rolling`	[TS] **rolling**
`simulate`	[R] **simulate**
`statsby`	[D] **statsby**
`stepwise`	[R] **stepwise**

Collecting results from margins, contrast, and pwcompare

`margins`, `contrast`, and `pwcompare` return results in `r()` by default. They can also store results in `e()` when the `post` option is specified.

When the `post` option is specified, `collect get` and the `collect` prefix work just as they would with any estimation command.

However, when `margins`, `contrast`, and `pwcompare` are specified without `post`, their interaction with `collect get` and the `collect` prefix is unique. Specifically, if you use `collect get` after one of these commands, you will need to specify `r()` to indicate that stored results should be collected from `r()`, just as you would with any command that stores results in `r()`. However, you can also specify the following identifiers if you wish to collect some of the statistics reported in the table and add them to the list of automatic results.

Identifier	Result
`_r_b`	coefficients or transformed coefficients reported by *command*
`_r_se`	standard errors of `_r_b`
`_r_z`	test statistics for `_r_b`
`_r_z_abs`	absolute values of `_r_z`
`_r_p`	p-values for `_r_b`
`_r_lb`	lower bounds of confidence intervals for `_r_b`
`_r_ub`	upper bounds of confidence intervals for `_r_b`
`_r_ci`	confidence intervals for `_r_b`
`_r_df`	degrees of freedom for `_r_b`

If you use the `collect` prefix with `margins`, `contrast`, and `pwcompare`, it can determine whether `post` was specified and, thus, whether results should be collected from `r()` or `e()`. In either case, the identifiers above can be specified with the `collect` prefix to add selected statistics reported by these commands to the list of automatic results.

Results not collected by default

Above, we told you that if you type

```
. collect: command
```

all results that *command* stores will be added to the collection. This is almost true. However, the following results are not stored by default.

```
e(b)
r(b)
e(V)
r(V)
e(b_fitorder)
e(b_idx)
e(b_keep)
e(b_sd)
e(Cns)
e(gauss_hasbeta)
e(gauss_V)
e(ilog)
e(gradient)
e(Jacobian)
e(table)
r(table)
e(V_modelbased)
e(V_sd)
e(V_vs)
```

In addition, any stored result with a name that begins with `datasignature`, `filename`, or `simtime` will not be collected by default.

Finally, any hidden results, the results you see only if you type

```
. ereturn list, all
```

or

```
. return list, all
```

are not collected by default.

You can collect these results if you specifically request them. You will need to give them a name at the time of collection. For instance, if you want to collect `e(V)` after a regression model, you can type

```
. collect V=e(V): regress y x
```

Also see

[TABLES] **collect create** — Create a new collection

[TABLES] **collect layout** — Specify table layout for the current collection

[TABLES] **collect set** — Set the current (active) collection

[TABLES] **collect unget** — Remove results from a collection

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