

**collapse** — Make dataset of summary statistics

[Description](#)  
[Options](#)

[Quick start](#)  
[Remarks and examples](#)

[Menu](#)  
[Acknowledgment](#)

[Syntax](#)  
[Also see](#)

## Description

`collapse` converts the dataset in memory into a dataset of means, sums, medians, etc. *clist* must refer to numeric variables exclusively.

Note: See [\[D\]](#) [contract](#) if you want to collapse to a dataset of frequencies.

## Quick start

Replace dataset in memory with means of `v1` and `v2`

```
collapse v1 v2
```

As above, but calculate statistics separately by each level of `catvar`

```
collapse v1 v2, by(catvar)
```

Dataset of mean, standard deviation, and standard error of the mean of `v1`

```
collapse (mean) mean1=v1 (sd) sd1=v1 (semean) sem1=v1
```

Mean and standard error of the mean for binomial `v2`

```
collapse (mean) mean2=v2 (sebinomial) sem2=v2
```

Frequency, median, and interquartile range of `v1`

```
collapse (count) freq=v1 (p50) p50=v1 (iqr) iqr=v1
```

Weighted and unweighted sum of `v2` using frequency weight `wvar`

```
collapse (sum) weighted=v2 (rawsum) unweighted=v2 [fweight=wvar]
```

## Menu

Data > Create or change data > Other variable-transformation commands > Make dataset of means, medians, etc.

## Syntax

```
collapse clist [if] [in] [weight] [, options]
```

where *clist* is either

```
[(stat)] varlist [ [(stat)] ... ]
[(stat)] target_var=varname [target_var=varname ...] [ [(stat)] ... ]
```

or any combination of the *varlist* and *target\_var* forms, and *stat* is one of

<u>mean</u>	means (default)	<u>sum</u>	sums
<u>median</u>	medians	<u>rawsum</u>	sums, ignoring optionally specified weight except observations with a weight of zero are excluded
<u>p1</u>	1st percentile	<u>count</u>	number of nonmissing observations
<u>p2</u>	2nd percentile	<u>percent</u>	percentage of nonmissing observations
<u>...</u>	3rd–49th percentiles	<u>max</u>	maximums
<u>p50</u>	50th percentile (same as <u>median</u> )	<u>min</u>	minimums
<u>...</u>	51st–97th percentiles	<u>iqr</u>	interquartile range
<u>p98</u>	98th percentile	<u>first</u>	first value
<u>p99</u>	99th percentile	<u>last</u>	last value
<u>sd</u>	standard deviations	<u>firstnm</u>	first nonmissing value
<u>semean</u>	standard error of the mean ( $\text{sd}/\sqrt{n}$ )	<u>lastnm</u>	last nonmissing value
<u>sebinomial</u>	standard error of the mean, binomial ( $\sqrt{p(1-p)/n}$ )		
<u>sepoisson</u>	standard error of the mean, Poisson ( $\sqrt{\text{mean}/n}$ )		

If *stat* is not specified, mean is assumed.

<i>options</i>	Description
----------------	-------------

---

Options	
<u>by</u> ( <i>varlist</i> )	groups over which <i>stat</i> is to be calculated
<u>cw</u>	casewise deletion instead of all possible observations
<u>fast</u>	do not restore the original dataset should the user press <i>Break</i> ; programmer's command

---

*varlist* and *varname* in *clist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

aweights, fweights, iweights, and pweights are allowed; see [U] 11.1.6 weight, and see *Weights* below. pweights may not be used with sd, semean, sebinomial, or sepoisson. iweights may not be used with semean, sebinomial, or sepoisson. aweights may not be used with sebinomial or sepoisson.

fast does not appear in the dialog box.

Examples:

```
. collapse age educ income, by(state)
. collapse (mean) age educ (median) income, by(state)
. collapse (mean) age educ income (median) medinc=income, by(state)
. collapse (p25) gpa [fw=number], by(year)
```

## Options

### Options

`by(varlist)` specifies the groups over which the means, etc., are to be calculated. If this option is not specified, the resulting dataset will contain 1 observation. If it is specified, *varlist* may refer to either string or numeric variables.

`cw` specifies casewise deletion. If `cw` is not specified, all possible observations are used for each calculated statistic.

The following option is available with `collapse` but is not shown in the dialog box:

`fast` specifies that `collapse` not restore the original dataset should the user press *Break*. `fast` is intended for use by programmers.

## Remarks and examples

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

`collapse` takes the dataset in memory and creates a new dataset containing summary statistics of the original data. `collapse` adds meaningful variable labels to the variables in this new dataset. Because the syntax diagram for `collapse` makes using it appear more complicated than it is, `collapse` is best explained with examples.

Remarks are presented under the following headings:

- [Introductory examples](#)
- [Variablewise or casewise deletion](#)
- [Weights](#)
- [A final example](#)

## Introductory examples

### ▷ Example 1

Consider the following artificial data on the grade-point average (gpa) of college students:

```
. use https://www.stata-press.com/data/r16/college
. describe
Contains data from https://www.stata-press.com/data/r16/college.dta
  obs:                12
  vars:                4                 3 Jan 2018 12:05
```

variable name	storage type	display format	value label	variable label
gpa	float	%9.0g		gpa for this year
hour	int	%9.0g		Total academic hours
year	int	%9.0g		1 = freshman, 2 = sophomore, 3 = junior, 4 = senior
number	int	%9.0g		number of students

Sorted by: year

```
. list, sep(4)
```

	gpa	hour	year	number
1.	3.2	30	1	3
2.	3.5	34	1	2
3.	2.8	28	1	9
4.	2.1	30	1	4
5.	3.8	29	2	3
6.	2.5	30	2	4
7.	2.9	35	2	5
8.	3.7	30	3	4
9.	2.2	35	3	2
10.	3.3	33	3	3
11.	3.4	32	4	5
12.	2.9	31	4	2

To obtain a dataset containing the 25th percentile of gpa's for each year, we type

```
. collapse (p25) gpa [fw=number], by(year)
```

We used frequency weights.

Next we want to create a dataset containing the mean of gpa and hour for each year. We do not have to type (mean) to specify that we want the mean because the mean is reported by default.

```
. use https://www.stata-press.com/data/r16/college, clear
. collapse gpa hour [fw=number], by(year)
. list
```

	year	gpa	hour
1.	1	2.788889	29.44444
2.	2	2.991667	31.83333
3.	3	3.233333	32.11111
4.	4	3.257143	31.71428

Now we want to create a dataset containing the mean and median of gpa and hour, and we want the median of gpa and hour to be stored as variables medgpa and medhour, respectively.

```
. use https://www.stata-press.com/data/r16/college, clear
. collapse (mean) gpa hour (median) medgpa=gpa medhour=hour [fw=num], by(year)
. list
```

	year	gpa	hour	medgpa	medhour
1.	1	2.788889	29.44444	2.8	29
2.	2	2.991667	31.83333	2.9	30
3.	3	3.233333	32.11111	3.3	33
4.	4	3.257143	31.71428	3.4	32

Here we want to create a dataset containing a count of gpa and hour and the minimums of gpa and hour. The minimums of gpa and hour will be stored as variables mingpa and minhour, respectively.

```
. use https://www.stata-press.com/data/r16/college, clear
. collapse (count) gpa hour (min) mingpa=gpa minhour=hour [fw=num], by(year)
. list
```

	year	gpa	hour	mingpa	minhour
1.	1	18	18	2.1	28
2.	2	12	12	2.5	29
3.	3	9	9	2.2	30
4.	4	7	7	2.9	31

Now we replace the values of `gpa` in 3 of the observations with missing values.

```
. use https://www.stata-press.com/data/r16/college, clear
. replace gpa = . in 2/4
(3 real changes made, 3 to missing)
. list, sep(4)
```

	gpa	hour	year	number
1.	3.2	30	1	3
2.	.	34	1	2
3.	.	28	1	9
4.	.	30	1	4
5.	3.8	29	2	3
6.	2.5	30	2	4
7.	2.9	35	2	5
8.	3.7	30	3	4
9.	2.2	35	3	2
10.	3.3	33	3	3
11.	3.4	32	4	5
12.	2.9	31	4	2

If we now want to list the data containing the mean of `gpa` and `hour` for each year, `collapse` uses all observations on `hour` for `year = 1`, even though `gpa` is missing for observations 1–3.

```
. collapse gpa hour [fw=num], by(year)
. list
```

	year	gpa	hour
1.	1	3.2	29.44444
2.	2	2.991667	31.83333
3.	3	3.233333	32.11111
4.	4	3.257143	31.71428

If we repeat this process but specify the `cw` option, `collapse` ignores all observations that have missing values.

```
. use https://www.stata-press.com/data/r16/college, clear
. replace gpa = . in 2/4
(3 real changes made, 3 to missing)
. collapse (mean) gpa hour [fw=num], by(year) cw
. list
```

	year	gpa	hour
1.	1	3.2	30
2.	2	2.991667	31.83333
3.	3	3.233333	32.11111
4.	4	3.257143	31.71428

◀

## ▷ Example 2

We have individual-level data from a census in which each observation is a person. Among other variables, the dataset contains the numeric variables `age`, `educ`, and `income` and the string variable `state`. We want to create a 50-observation dataset containing the means of age, education, and income for each state.

```
. collapse age educ income, by(state)
```

The resulting dataset contains means because `collapse` assumes that we want means if we do not specify otherwise. To make this explicit, we could have typed

```
. collapse (mean) age educ income, by(state)
```

Had we wanted the mean for `age` and `educ` and the median for `income`, we could have typed

```
. collapse (mean) age educ (median) income, by(state)
```

or if we had wanted the mean for `age` and `educ` and both the mean and the median for `income`, we could have typed

```
. collapse (mean) age educ income (median) medinc=income, by(state)
```

This last dataset will contain three variables containing means—`age`, `educ`, and `income`—and one variable containing the median of income—`medinc`. Because we typed `(median) medinc=income`, Stata knew to find the median for income and to store those in a variable named `medinc`. This renaming convention is necessary in this example because a variable named `income` containing the mean is also being created.

◀

## Variablewise or casewise deletion

### ▷ Example 3

Let's assume that in our census data, we have 25,000 persons for whom age is recorded but only 15,000 for whom income is recorded; that is, `income` is missing for 10,000 observations. If we want summary statistics for `age` and `income`, `collapse` will, by default, use all 25,000 observations when calculating the summary statistics for `age`. If we prefer that `collapse` use only the 15,000 observations for which `income` is not missing, we can specify the `cw` (casewise) option:

```
. collapse (mean) age income (median) medinc=income, by(state) cw
```

◀

## Weights

`collapse` allows all four weight types; the default is `aweight`s. Weight normalization affects only the `sum`, `count`, `sd`, `semean`, and `sebinomial` statistics.

Let  $j$  index observations and  $i$  index by-groups. Here are the definitions for `count` and `sum` with weights:

**count:**

`unweighted:`  $N_i$ , the number of observations in group  $i$

`aweight:`  $N_i$ , the number of observations in group  $i$

`fweight, iweight, pweight:`  $\sum w_j$ , the sum of the weights over observations in group  $i$

**sum:**

`unweighted:`  $\sum x_j$ , the sum of  $x_j$  over observations in group  $i$

`aweight:`  $\sum v_j x_j$  over observations in group  $i$ ;  $v_j =$  weights normalized to sum to  $N_i$

`fweight, iweight, pweight:`  $\sum w_j x_j$  over observations in group  $i$

When the `by()` option is not specified, the entire dataset is treated as one group.

The `sd` statistic with weights returns the square root of the bias-corrected variance, which is based on the factor  $\sqrt{N_i/(N_i - 1)}$ , where  $N_i$  is the number of observations. Statistics `sd`, `semean`, `sebinomial`, and `sepoisson` are not allowed with `pweighted` data. Otherwise, the statistic is changed by the weights through the computation of the weighted count, as outlined above.

For instance, consider a case in which there are 25 observations in the dataset and a weighting variable that sums to 57. In the unweighted case, the weight is not specified, and the count is 25. In the analytically weighted case, the count is still 25; the scale of the weight is irrelevant. In the frequency-weighted case, however, the count is 57, the sum of the weights.

The `rawsum` statistic with `aweight`s ignores the weight, with one exception: observations with zero weight will not be included in the sum.

### ▷ Example 4

Using our same census data, suppose that instead of starting with individual-level data and aggregating to the state level, we started with state-level data and wanted to aggregate to the region level. Also assume that our dataset contains `pop`, the population of each state.

To obtain unweighted means and medians of age and income, by region, along with the total population, we could type

```
. collapse (mean) age income (median) medage=age medinc=income (sum) pop,
> by(region)
```

To obtain weighted means and medians of age and income, by region, along with the total population and using frequency weights, we could type

```
. collapse (mean) age income (median) medage=age medinc=income (count) pop
> [fweight=pop], by(region)
```

Note: Specifying `(sum) pop` would not have worked because that would have yielded the pop-weighted sum of `pop`. Specifying `(count) age` would have worked as well as `(count) pop` because `count` merely counts the number of nonmissing observations. The counts here, however, are frequency-weighted and equal the sum of `pop`.

To obtain the same mean and medians as above, but using analytic weights, we could type

```
. collapse (mean) age income (median) medage=age medinc=income (rawsum) pop
> [aweight=pop], by(region)
```

Note: Specifying `(count) pop` would not have worked because, with analytic weights, `count` would count numbers of physical observations. Specifying `(sum) pop` would not have worked because `sum` would calculate weighted sums (with a normalized weight). The `rawsum` function, however, ignores the weights and sums only the specified variable, with one exception: observations with zero weight will not be included in the sum. `rawsum` would have worked as the solution to all three cases. ◀

## A final example

### ▷ Example 5

We have census data containing information on each state's median age, marriage rate, and divorce rate. We want to form a new dataset containing various summary statistics, by region, of the variables:



```
. use https://www.stata-press.com/data/r16/census5, clear
(1980 Census data by state)

. describe

Contains data from https://www.stata-press.com/data/r16/census5.dta
  obs:                50                1980 Census data by state
  vars:                7                 6 Apr 2018 15:43
```

variable name	storage type	display format	value label	variable label
state	str14	%14s		State
state2	str2	%-2s		Two-letter state abbreviation
region	int	%8.0g	cenreg	Census region
pop	long	%10.0g		Population
median_age	float	%9.2f		Median age
marriage_rate	float	%9.0g		
divorce_rate	float	%9.0g		

Sorted by: region

```
. collapse (median) median_age marriage divorce (mean) avgmrate=marriage
> avgdrate=divorce [aw=pop], by(region)

. list
```

	region	median_age	marriage_rate	divorce_rate	avgmrate	avgdrate
1.	NE	31.90	.0080657	.0035295	.0081472	.0035359
2.	N Cntrl	29.90	.0093821	.0048636	.0096701	.004961
3.	South	29.60	.0112609	.0065792	.0117082	.0059439
4.	West	29.90	.0089093	.0056423	.0125199	.0063464

```
. describe
```

```
Contains data
  obs:                4                1980 Census data by state
  vars:                6
```

variable name	storage type	display format	value label	variable label
region	int	%8.0g	cenreg	Census region
median_age	float	%9.2f		(p 50) median_age
marriage_rate	float	%9.0g		(p 50) marriage_rate
divorce_rate	float	%9.0g		(p 50) divorce_rate
avgmrate	float	%9.0g		(mean) marriage_rate
avgdrate	float	%9.0g		(mean) divorce_rate

Sorted by: region

Note: Dataset has changed since last saved.

4

## Acknowledgment

We thank David Roodman of the Open Philanthropy Project for writing `collapse2`, which inspired several features in `collapse`.

## Also see

- [D] [contract](#) — Make dataset of frequencies and percentages
- [D] [egen](#) — Extensions to generate
- [D] [statsby](#) — Collect statistics for a command across a by list
- [R] [summarize](#) — Summary statistics