The Stata Journal (2010)
10, Number 1, pp. 157–159

Stata tip 84: Summing missings

Nicholas J. Cox
Department of Geography
Durham University
Durham City, UK
n.j.cox@durham.ac.uk

Consider this pretend dataset and the result of a mundane `collapse`:

```
. list, sepby(g)

     +-----+
     | g   y |
     |-----|
  1. | 1   1 |
  2. | 1   2 |
  3. | 1   3 |
  4. | 1   4 |
     |-----|
  5. | 2   5 |
  6. | 2   6 |
  7. | 2   7 |
     |-----|
  8. | 3   8 |
  9. | 3   . |
     |-----|
 10. | 4   . |
     +-----+

. collapse (sum) y, by(g)
. list

     +-----+
     | g   y |
     |-----|
  1. | 1  10 |
  2. | 2  18 |
  3. | 3   8 |
  4. | 4   0 |
     +-----+
```

Here we have a grouping variable, `g`, and a response, `y`. When we `collapse, by(g)` to produce a reduced dataset of the sums of `y`, many users are surprised at the results. Clearly $1+2+3+4=10$ and $5+6+7=18$, so no surprises there, but the other results need more comment.

When producing sums, Stata follows two specific rules:

1. Initialize the result at 0 and add pertinent numbers one by one until done. The result is the sum.

2. Ignore missings.

With these two rules, it should not surprise you that the sum $8 + .$ is reported as 8 (the missing is ignored) or that the sum of . is reported as 0 (the missing is ignored, and we just see the result of initializing, namely, 0).

Nevertheless, many users see this reasoning as perverse. A counterargument runs like this: If we know that all values in a group are missing, then we know nothing about their sum, which should also be recorded as missing. Users with this view need to know how to get what they want. They should also want to know more about why Stata treats missings in this way.

Consider a related problem: What is the sum of an empty set? Stata's answer to the problem is best understood by extending the problem. Imagine combining that empty set with any nonempty set, say, 1, 2, 3. What is the sum of the combined set? The Stata answer is that as

$$\text{sum of empty set} + \text{sum of } 1, 2, 3 = \text{sum of combined set } 1, 2, 3$$

so also the sum of an empty set must be regarded as 0. Sums are defined not by what they are but by how they behave. Insisting that the sum of an empty set must be missing prevents you from ever combining that result helpfully with anything else.

This problem, which was dubbed "related", is really the same problem in Stata's mind. Because missings are ignored, it is as if they do not exist. Thus a set of values that is all missing is equivalent to an empty set.

If you look carefully, you will find this behavior elsewhere in Stata, so it should be less of a surprise. `summarize` (see [R] **summarize**) will also report the sum of missings as zero:

```
. sysuse auto, clear
(1978 Automobile Data)
. summmarize rep78 if rep78 == .
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       rep78 |          0
. return list
scalars:
                  r(N) =  0
              r(sum_w) =  0
                r(sum) =  0
```

The sum (and sum of weights) is not explicit in the output from `summarize`, but it is calculated on the side and put in r-class results.

`egen` (see [D] **egen**) functions `total()` and `rowtotal()` by default yield 0 for the total of missings. However, as from Stata 10.1, they now have a new `missing` option. With this option, if all arguments are missing, then the corresponding result will be set to missing.

Mata has an explicit variant on empty sets, e.g.,

```
. mata: sum(J(0,0,.))
  0
. mata: sum(J(0,0,42))
  0
```

Whenever this behavior does not meet your approval, you need a work-around. Returning to the first example, one strategy is to include in the `collapse` enough information to identify groups that were all missing. There are several ways to do this. Here is one:

```
. collapse (sum) y (min) min=y, by(g)
. list
```

	g	y	min
1.	1	10	1
2.	2	18	5
3.	3	8	8
4.	4	0	.

```
. replace y = . if missing(min)
(1 real change made, 1 to missing)
. list
```

	g	y	min
1.	1	10	1
2.	2	18	5
3.	3	8	8
4.	4	.	.

The idea here is that the minimum is missing if and only if all values are missing. Using `missing(min)` as a test also catches any situation in which the smallest missing value is one of `.a`–`.z`.

Missings pose problems for any mathematical or statistical software, and it is difficult to design rules that match all users' intuitions. Indeed, some intuitions are changed by becoming accustomed to certain rules. Stata's philosophy of ignoring missings where possible contrasts with software in which missings are regarded as highly infectious, affecting whatever operations they enter. Here is not the place to debate the merits and demerits of different choices in detail but rather to underline that Stata does attempt to follow rules consistently, with the consequence that you need a work-around whenever you disagree with the outcome.