The Stata Journal

The *Stata Journal*, electronic version (ISSN 1536-8734) is a publication of Stata Press, and Stata is a registered trademark of StataCorp LP.

The Stata Journal (2006)
6, Number 1, pp. 144–146

Stata tip 28: Precise control of dataset sort order

L. Philip Schumm
Department of Health Studies
University of Chicago
Chicago, IL
pschumm@uchicago.edu

The observations in a Stata dataset are ordered, so that they may be referred to by their position (e.g., `in 42/48`) and that individual values of a variable may be referred to with subscripts (e.g., `mpg[42]`). This order can be changed by using the `sort` command (see [D] **sort**). Developing a full appreciation of what is possible using `sort` together with the `by:` prefix, the underscore built-ins `_n` and `_N`, and subscripting is a major step toward Stata enlightenment (e.g., see Cox [2002]).

One source of surprise for many users arises when sorting by one or more variables which, when taken together, do not uniquely determine the order of observations. In this case, the resulting order within any group of observations having the same value(s) of those variables is effectively random because `sort` uses an *unstable* sort algorithm. Users who desire a *stable* sort—in which the previous ordering of observations within tied values of the sort variables is maintained—should specify the `stable` option. However, this option will slow `sort` down and, more importantly, can hide problems in your code.

You are likely to discover this issue when coding an operation dependent on the order of the data that gives different results from one run to another. Consider the following dataset consisting of mothers and their children:

```
. list, sepby(family)
```

	family	name	child
1.	2	Harriet	0
2.	2	Lewis	1
3.	1	Sylvia	0
4.	1	Jenny	1
5.	3	Kim	0
6.	3	Peter	1
7.	3	Kim	1

Individuals are grouped by family, the mother always appearing first. Suppose that we want to construct a unique within-family identifier, such that all mothers have the same value. This is a straightforward application of `by:`, but first the data must be sorted by family:

```
. sort family
. by family: generate individual = _n
```



```
. table child individual
```

child	individual 1	2	3
0	2	1	
1	1	2	1

Unfortunately, the result is not as desired: one mother was assigned the value 2. In fact, following the call to `sort`, the order of observations within families—and hence the assignment of identifiers—was random. If we had instead sorted by family *and* child, each mother would have appeared first and would have been assigned a value of 1 (assuming that each family has exactly one mother—a key assumption that should always be checked). Yet even this solution would still be deficient: if a family has multiple children, their identifiers would be random and irreproducible. Only if we sort by family, child, *and* name would we have an adequate solution.

If we had used instead

```
. sort family, stable
```

we would also have obtained the desired result. So why does `sort` by default perform an unstable sort? Apart from better performance, the answer (emphasized by William Gould on Statalist) is that using the `stable` option not only fails to address the problem; it also reduces the chance of discovering it. Our error was to perform a calculation dependent on the sort order of the data without establishing that order beforehand. Using `stable` would have temporarily masked the error. However, had the sort order of the input dataset changed, we would have been in trouble.

How can you avoid such problems? First, train yourself to recognize when a calculation depends on the sort order of the data. Most instances in which you are using `_n` and `_N` or subscripting (either alone or with `by`) are easy to recognize. However, instances in which you are using a function that depends on the order of the data (e.g., `sum()` or `group()`) can be more subtle (Gould 2000).

Second, ensure that the order of the data is fully specified. This check became much easier in Stata 8 with the introduction of the `isid` command ([D] **isid**), which checks whether one or more variables uniquely identify the observations and returns an error if they do not. The command also has a `sort` option, which sorts the dataset in order of the specified variable(s). This option lets us replace our original `sort` command with

```
. isid family child name, sort
```

which, since it runs without error, confirms that we have specified the order fully. Had we used only `family`, or `family` and `child`, `isid` would have returned an error, immediately alerting us to the problem.

References

Cox, N. J. 2002. Speaking Stata: How to move step by: step. *Stata Journal* 2: 86–102.

Gould, W. 2000. FAQ: Sorting on categorical variables.
http://www.stata.com/support/faqs/lang/sort.html.