# Title

stata.com

**sort** — Sort data



## Description

`sort` arranges the observations of the current data into ascending order based on the values of the variables in *varlist*. There is no limit to the number of variables in the *varlist*. Missing numeric values are interpreted as being larger than any other number, so they are placed last with `.` < `.a` < `.b` < ··· < `.z`. When you sort on a string variable, however, null strings are placed first and uppercase letters come before lowercase letters.

The dataset is marked as being sorted by *varlist* unless `in` *range* is specified. If `in` *range* is specified, only those observations are rearranged. The unspecified observations remain in the same place.

## Quick start

Sort dataset in memory by ascending values of `v1`

```
sort v1
```

As above, and order within `v1` by ascending values of `v2` and within `v2` by `v3`

```
sort v1 v2 v3
```

As above, and keep observations with the same values of `v1`, `v2`, and `v3` in the same presort order

```
sort v1 v2 v3, stable
```

## Menu

Data > Sort

## Syntax

`sort` *varlist* [*in*] [`, stable`]

## Option

`stable` specifies that observations with the same values of the variables in *varlist* keep the same relative order in the sorted data that they had previously. For instance, consider the following data:

| x | b |
|---|---|
| 3 | 1 |
| 1 | 2 |
| 1 | 1 |
| 1 | 3 |
| 2 | 4 |

Typing `sort x` without the `stable` option produces one of the following six orderings:

| x | b |
|---|---|
| 1 | 2 |
| 1 | 1 |
| 1 | 3 |
| 2 | 4 |
| 3 | 1 |

| x | b |
|---|---|
| 1 | 2 |
| 1 | 3 |
| 1 | 1 |
| 2 | 4 |
| 3 | 1 |

| x | b |
|---|---|
| 1 | 1 |
| 1 | 3 |
| 1 | 2 |
| 2 | 4 |
| 3 | 1 |

| x | b |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 2 | 4 |
| 3 | 1 |

| x | b |
|---|---|
| 1 | 3 |
| 1 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 1 |

| x | b |
|---|---|
| 1 | 3 |
| 1 | 2 |
| 1 | 1 |
| 2 | 4 |
| 3 | 1 |

Without the `stable` option, the ordering of observations with equal values of *varlist* is randomized. With `sort x, stable`, you will always get the first ordering and never the other five.

If your intent is to have the observations sorted first on `x` and then on `b` within tied values of `x` (the fourth ordering above), you should type `sort x b` rather than `sort x, stable`.

`stable` is seldom used and, when specified, causes `sort` to execute more slowly.

## Remarks and examples

stata.com

Sorting data is one of the more common tasks involved in processing data. Sometimes, before Stata can perform some task, the data must be in a specific order. For example, if you want to use the `by` *varlist*`:` prefix, the data must be sorted in order of *varlist*. You use the `sort` command to fulfill this requirement.

### ▷ Example 1

Sorting data can also be informative. Suppose that we have data on automobiles, and each car's make and mileage rating (called `make` and `mpg`) are included among the variables in the data. We want to list the five cars with the lowest mileage rating in our data:

```
. use http://www.stata-press.com/data/r15/auto
(1978 Automobile Data)
. keep make mpg weight
. sort mpg, stable
. list make mpg in 1/5
```

| | make | mpg |
|---|---|---|
| 1. | Linc. Continental | 12 |
| 2. | Linc. Mark V | 12 |
| 3. | Cad. Deville | 14 |
| 4. | Cad. Eldorado | 14 |
| 5. | Linc. Versailles | 14 |

◁

### ▷ Example 2: Tracking the sort order

Stata keeps track of the order of your data. For instance, we just sorted the above data on `mpg`. When we ask Stata to `describe` the data in memory, it tells us how the dataset is sorted:

```
. describe
Contains data from http://www.stata-press.com/data/r15/auto.dta
  obs:            74                          1978 Automobile Data
 vars:             3                          13 Apr 2016 17:45
 size:         1,628                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
mpg             int     %8.0g                 Mileage (mpg)
weight          int     %8.0gc                Weight (lbs.)
-------------------------------------------------------------------------------
Sorted by: mpg
     Note: Dataset has changed since last saved.
```

Stata keeps track of changes in sort order. If we were to make a change to the `mpg` variable, Stata would know that the data are no longer sorted. Remember that the first observation in our data has `mpg` equal to 12, as does the second. Let's change the value of the first observation:

```
. replace mpg=13 in 1
(1 real change made)
. describe
Contains data from http://www.stata-press.com/data/r15/auto.dta
  obs:            74                          1978 Automobile Data
 vars:             3                          13 Apr 2016 17:45
 size:         1,628                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
mpg             int     %8.0g                 Mileage (mpg)
weight          int     %8.0gc                Weight (lbs.)
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

After making the change, Stata indicates that our dataset is "Sorted by:" nothing. Let's put the dataset back as it was:

```
. replace mpg=12 in 1
(1 real change made)
. sort mpg
```

◁

## ❑ Technical note

Stata does not track changes in the sort order and will sometimes decide that a dataset is not sorted when, in fact, it is. For instance, if we were to change the first observation of our `auto` dataset from 12 miles per gallon to 10, Stata would decide that the dataset is "Sorted by:" nothing, just as it did above when we changed `mpg` from 12 to 13. Our change in example 2 did change the order of the data, so Stata was correct. Changing `mpg` from 12 to 10, however, does not really affect the sort order.

As far as Stata is concerned, any change to the variables on which the data are sorted means that the data are no longer sorted, even if the change actually leaves the order unchanged. Stata may be dumb, but it is also fast. It sorts already-sorted datasets instantly, so Stata's ignorance costs us little.

❑

## ▷ Example 3: Sorting on multiple variables

Data can be sorted by more than one variable, and in such cases, the sort order is lexicographic. If we `sort` the data by two variables, for instance, the data are placed in ascending order of the first variable, and then observations that share the same value of the first variable are placed in ascending order of the second variable. Let's order our automobile data by `mpg` and within `mpg` by `weight`:

```
. sort mpg weight
. list in 1/8, sep(4)
```

| | make | mpg | weight |
|---|---|---|---|
| 1. | Linc. Mark V | 12 | 4,720 |
| 2. | Linc. Continental | 12 | 4,840 |
| 3. | Peugeot 604 | 14 | 3,420 |
| 4. | Linc. Versailles | 14 | 3,830 |
| 5. | Cad. Eldorado | 14 | 3,900 |
| 6. | Merc. Cougar | 14 | 4,060 |
| 7. | Merc. XR-7 | 14 | 4,130 |
| 8. | Cad. Deville | 14 | 4,330 |

The data are in ascending order of `mpg`, and, within each `mpg` category, the data are in ascending order of `weight`. The lightest car that achieves 14 miles per gallon in our data is the Peugeot 604.

◁

## ❑ Technical note

The sorting technique used by Stata is fast, but the order of variables not included in the *varlist* is not maintained. If you wish to maintain the order of additional variables, include them at the end of the *varlist*. There is no limit to the number of variables by which you may `sort`.

❑

## ▷ Example 4: Descending sorts

Sometimes you may want to order a dataset by descending sequence of something. Perhaps we wish to obtain a list of the five cars achieving the best mileage rating. The `sort` command orders the data only into ascending sequences. Another command, `gsort`, orders the data in ascending or descending sequences; see [D] **gsort**. You can also create the negative of a variable and achieve the desired result:

```
. generate negmpg = -mpg
. sort negmpg
. list in 1/5
```

| | make | mpg | weight | negmpg |
|---|---|---|---|---|
| 1. | VW Diesel | 41 | 2,040 | -41 |
| 2. | Subaru | 35 | 2,050 | -35 |
| 3. | Datsun 210 | 35 | 2,020 | -35 |
| 4. | Plym. Champ | 34 | 1,800 | -34 |
| 5. | Toyota Corolla | 31 | 2,200 | -31 |

We find that the VW Diesel tops our list.

◁

### ▷ Example 5: Sorting on string variables

`sort` may also be used on string variables. The data are sorted alphabetically:

```
. sort make
. list in 1/5
```

| | make | mpg | weight | negmpg |
|---|---|---|---|---|
| 1. | AMC Concord | 22 | 2,930 | -22 |
| 2. | AMC Pacer | 17 | 3,350 | -17 |
| 3. | AMC Spirit | 22 | 2,640 | -22 |
| 4. | Audi 5000 | 17 | 2,830 | -17 |
| 5. | Audi Fox | 23 | 2,070 | -23 |

◁

### ❑ Technical note

Bear in mind that Stata takes "alphabetically" to mean "in order by byte value". This means that all uppercase letters come before lowercase letters; for example, Z < a. As far as Stata is concerned, the following list is sorted alphabetically:

```
. list, sep(0)
```

| | myvar |
|---|---|
| 1. | ALPHA |
| 2. | Alpha |
| 3. | BETA |
| 4. | Beta |
| 5. | alpha |
| 6. | beta |

For most purposes, this method of sorting is sufficient. It is possible to override Stata's sort logic. See [U] **12.4.2.5 Sorting strings containing Unicode characters** for information about ordering strings in a language-sensitive way. We do not recommend that you do this.

❑

## References

Royston, P. 2001. Sort a list of items. *Stata Journal* 1: 105–106.

Schumm, L. P. 2006. Stata tip 28: Precise control of dataset sort order. *Stata Journal* 6: 144–146.

## Also see

[D] **describe** — Describe data in memory or in file

[D] **gsort** — Ascending and descending sort

[U] **11 Language syntax**