Description

`drop` eliminates variables or observations from the data in memory.

`keep` works the same way as `drop`, except that you specify the variables or observations to be kept rather than the variables or observations to be deleted.

Warning: `drop` and `keep` are not reversible. Once you have eliminated observations, you cannot read them back in again. You would need to go back to the original dataset and read it in again. Instead of applying `drop` or `keep` for a subset analysis, consider using `if` or `in` to select subsets temporarily. This is usually the best strategy. Alternatively, applying `preserve` followed in due course by `restore` may be a good approach.

Quick start

Remove v1, v2, and v3 from memory

```
    drop v1 v2 v3
```

Remove all variables whose name begins with `code` from memory

```
    drop code*
```

Remove observations where `v1` is equal to 99

```
    drop if v1==99
```

Also drop observations where `v1` equals 88 or `v2` is missing

```
    drop if inlist(v1,88,99) | missing(v2)
```

Keep observations where `v3` is not missing

```
    keep if !missing(v3)
```

Keep the first observation from each cluster identified by `cvar`

```
    by cvar: keep if _n==1
```

Menu

**Drop or keep variables**

Data > Variables Manager

**Drop or keep observations**

Data > Create or change data > Drop or keep observations
Syntax

Drop variables

    drop varlist

Drop observations

    drop if exp

Drop a range of observations

    drop in range [if exp]

Keep variables

    keep varlist

Keep observations that satisfy specified condition

    keep if exp

Keep a range of observations

    keep in range [if exp]

by is allowed with the second syntax of drop and the second syntax of keep; see [D] by.

Remarks and examples

You can clear the entire dataset by typing drop _all without affecting value labels, macros, and programs. (Also see [U] 12.6 Dataset, variable, and value labels, [U] 18.3 Macros, and [P] program.)
We will systematically eliminate data until, at the end, no data are left in memory. We begin by describing the data:

```
. use http://www.stata-press.com/data/r14/census11
(1980 Census data by state)
. describe
    obs:     50  1980 Census data by state
    vars:   15  2 Dec 2014 14:31
   size: 3,200

    variable name   type    display format value label
    #1. state       str13    %-13s        State
    #2. state2      str2     %-2s         Two-letter state abbreviation
    #3. region      byte     %-8.0g cenreg Census region
    #4. pop         long     %12.0gc      Population
    #5. pop15       long     %12.0gc      Pop, < 5 year
    #6. pop5_17     long     %12.0gc      Pop, 5 to 17 years
    #7. pop18p      long     %12.0gc      Pop, 18 and older
    #8. pop65p      long     %12.0gc      Pop, 65 and older
    #9. popurban    long     %12.0gc      Urban population
   #10. medage      float    %9.2f        Median age
   #11. death       long     %12.0gc      Number of deaths
   #12. marriage    long     %12.0gc      Number of marriages
   #13. divorce     long     %12.0gc      Number of divorces
   #14. mrgrate     float    %9.0g        Marriage rate
   #15. dvcrate     float    %9.0g        Divorce rate

Sorted by: region
```

We can eliminate all the variables with names that begin with pop by typing `drop pop*`:
Let's eliminate more variables and then eliminate observations:

```stata
. drop marriage divorce mrgrate dvcrate
. describe
obs: 50 1980 Census data by state
vars: 5 2 Dec 2014 14:31
size: 1,200
```

Sorted by: region  
Note: Dataset has changed since last saved.

Next we will drop any observation for which `medage` is greater than 32.

```stata
. drop if medage > 32
(3 observations deleted)
```

Let's drop the first observation in each region:

```stata
. by region: drop if _n==1
(4 observations deleted)
```

Now we drop all but the last observation in each region:

```stata
. by region: drop if _n!=_N
(39 observations deleted)
```

Let's now drop the first 2 observations in our dataset:

```stata
. drop in 1/2
(2 observations deleted)
```
Finally, let’s get rid of everything:

```
. drop _all
. describe
Contains data
   obs:    0
   vars:   0
   size:   0
Sorted by:
```

Typing `keep in 10/1` is the same as typing `drop in 1/9`.

Typing `keep if x==3` is the same as typing `drop if x !=3`.

`keep` is especially useful for keeping a few variables from a large dataset. Typing `keep myvar1 myvar2` is the same as typing `drop` followed by all the variables in the dataset except `myvar1` and `myvar2`.

--- Technical note ---

In addition to dropping variables and observations, `drop _all` removes any business calendars; see [D] `datetime business calendars`.

--- References ---


--- Also see ---

[D] `clear` — Clear memory

[D] `varmanage` — Manage variable labels, formats, and other properties

[U] 11 Language syntax

[U] 13 Functions and expressions