st_dropvar() — Drop variables or observations

Syntax Description Remarks and examples Conformability Diagnostics Also see

Syntax

void `st_dropvar(`*transmorphic rowvector vars*`)`

void `st_dropobsin(`*real matrix range*`)`

void `st_dropobsif(`*real colvector select*`)`

void `st_keepvar(`*transmorphic rowvector vars*`)`

void `st_keepobsin(`*real matrix range*`)`

void `st_keepobsif(`*real colvector select*`)`

Description

`st_dropvar(`*vars*`)` drops the variables specified. *vars* is a row vector that may contain either variable names or variable indices. `st_dropvar(.)` drops all variables and observations.

`st_dropobsin()` and `st_dropobsif()` have to do with dropping observations.

`st_dropobsin(`*range*`)` specifies the observations to be dropped:

> `st_dropobsin(5)` drops observation 5.
>
> `st_dropobsin((5,9))` drops observations 5 through 9.
>
> `st_dropobsin((5\8\12))` drops observations 5 and 8 and 12.
>
> `st_dropobsin((5,7\8,11\13,13))` drops observations 5 through 7, 8 through 11, and 13.
>
> `st_dropobsin(.)` drops all observations (but not the variables).
>
> `st_dropobsin(J(0,1,.))` drops no observations (or variables).

`st_dropobsif(`*select*`)` specifies a `st_nobs()` × 1 vector. Observations i for which $select_i \neq 0$ are dropped.

`st_keepvar()`, `st_keepobsin()`, and `st_keepobsif()` do the same thing, except that the variables and observations to be kept are specified.

Remarks and examples

To drop all variables and observations, code any of the following:

```
        st_dropvar(.)
        st_keepvar(J(1,0,.))
        st_keepvar(J(1,0,""))
```

All do the same thing. Dropping all the variables clears the dataset.

Dropping all the observations, however, leaves the variables in place.

Conformability

`st_dropvar(`*vars*`)`, `st_keepvar(`*vars*`)`:

vars:	$1 \times k$
result:	*void*

`st_dropobsin(`*range*`)`, `st_keepobsin(`*range*`)`:

range:	$k \times 1$ or $k \times 2$
result:	*void*

`st_dropobsif(`*select*`)`, `st_keepobsif(`*select*`)`:

select:	`st_nobs()` × 1
result:	*void*

Diagnostics

`st_dropvar(`*vars*`)` and `st_keepvar(`*vars*`)` abort with error if any element of *vars* is missing unless *vars* is 1 × 1, in which case they drop or keep all the variables.

`st_dropvar(`*vars*`)` and `st_keepvar(`*vars*`)` abort with error if any element of *vars* is not a valid variable index or name, or if *vars* is a view. If *vars* is specified as names, abbreviations are not allowed.

`st_dropvar()` and `st_keepvar()` set `st_updata()` (see [M-5] **st_updata( )**) unless all variables dropped are temporary; see [M-5] **st_tempname( )**.

`st_dropobsin(`*range*`)` and `st_keepobsin(`*range*`)` abort with error if any element of *range* is missing unless *range* is 1 × 1, in which case they drop or keep all the observations.

`st_dropobsin(`*range*`)` and `st_keepobsin(`*range*`)` abort with error if any element of *range* is not a valid observation number (is not between 1 and `st_nobs()` [see [M-5] **st_nvar( )**] inclusive) or if *range* is a view.

`st_dropobsif(`*select*`)` and `st_keepobsif(`*select*`)` abort with error if *select* is a view.

`st_dropobsin()`, `st_dropobsif()`, `st_keepobsin()`, and `st_keepobsif()` set `st_updata()` if any observations are removed from the data.

Be aware that, after dropping any variables or observations, any previously constructed views (see [M-5] **st_view( )**) are probably invalid because views are internally stored in terms of variable and observation numbers. Subsequent use of an invalid view may lead to unexpected results or an abort with error.

Also see

[M-4] **stata** — Stata interface functions