Title

stata.com

st_subview() — Make view from view

Description | Syntax | Remarks and examples | Conformability
Diagnostics | Also see

Description

`st_subview(`X`, `V`, `i`, `j`)` creates new view matrix X from existing view matrix V. V is to have been created from a previous call to `st_view()` (see [M-5] **st_view()**) or `st_subview()`.

Although `st_subview()` is intended for use with view matrices, it may also be used when V is a regular matrix. Thus code may be written in such a way that it will work without regard to whether a matrix is or is not a view.

i may be specified as a 1×1 scalar, a 1×1 scalar containing missing, as a column vector of row numbers, as a row vector specifying a row-number range, or as a $k \times 2$ matrix specifying both:

a. `st_subview(`X`,`V`, 1,2)` makes X equal to the first row of the second column of V.

b. `st_subview(`X`,`V`, .,2)` makes X equal to all rows of the second column of V.

c. `st_subview(`X`,`V`, (1\2\5),2)` makes X equal to rows 1, 2, and 5 of the second column of V.

d. `st_subview(`X`,`V`, (1,5),2)` makes X equal to rows 1 through 5 of the second column of V.

e. `st_subview(`X`,`V`, (1,5\7,9),2)` makes X equal to rows 1 through 5 and 7 through 9 of the second column of V.

f. When a range is specified, any element of the range (i_1,i_2) may be set to contribute zero observations if $i_2 = i_1 - 1$. For example, `(1,0)` is not an error and neither is `(1,0\5,7)`.

j may be specified in the same way as i, except transposed, to specify the selected columns:

a. `st_subview(`X`,`V`, 2,.)` makes X equal to all columns of the second row of V.

b. `st_subview(`X`,`V`, 2,(1,2,5))` makes X equal to columns 1, 2, and 5 of the second row of V.

c. `st_subview(`X`,`V`, 2,(1\5))` makes X equal to columns 1 through 5 of the second row of V.

d. `st_subview(`X`,`V`, 2,((1\5),(7\9)))` makes X equal to columns 1 through 5 and 7 through 9 of the second row of V.

e. When a range is specified, any element of the range `(`j_1`\`j_2`)` may be set to contribute zero columns if $j_2 = j_1 - 1$. For example, `(1\0)` is not an error and neither is `((1\0),(5\7))`.

Obviously, notations for i and j can be specified simultaneously:

a. `st_subview(`X`,`V`, .,.)` makes X a duplicate of V.

b. `st_subview(`X`,`V`, .,(1\5))` makes X equal to columns 1 through 5 of all rows of X.

c. `st_subview(`X`,`V`, (10,25),(1\5))` makes X equal to columns 1 through 5 of rows 10 through 25 of X.

Also, `st_subview()` may be used to create views with duplicate variables or observations from V.

Syntax

void `st_subview(`X, *transmorphic matrix V*, *real matrix i*, *real matrix j*)

where

1. The type of X does not matter; it is replaced.

2. V is typically a view, but that is not required. V, however, must be real or string.

Remarks and examples

stata.com

Say that you need to make a calculation on matrices X and Y, which might be views. Perhaps the calculation is `invsym(`X`'`X`)*`X`'`Y. Regardless, you start as follows:

```
st_view(X, ., "v2 v3 v4", 0)
st_view(Y, ., "v1 v7"   , 0)
```

You are already in trouble. You smartly coded fourth argument as 0, meaning exclude the missing values, but you do not know that the same observations were excluded in the manufacturing of `X` as in the manufacturing of `Y`.

If you had previously created a `touse` variable in your dataset marking the observations to be used in the calculation, one solution would be

```
st_view(X, ., "v2 v3 v4", "touse")
st_view(Y, ., "v1 v7"   , "touse")
```

That solution is recommended, but let's assume you did not do that. The other solution is

```
st_view(M, ., "v2 v3 v4 v1 v7", 0)
st_subview(X, M, ., (1,2,3))
st_subview(Y, M, ., (4,5))
```

The first call to `st_view()` will eliminate observations with missing values on any of the variables, and the second two `st_subview()` calls will create the matrices you wanted, obtaining them from the correctly formed `M`. Basically, the two `st_subview()` calls amount to the same thing as

```
X = M[., (1,2,3)]
Y = M[., (4,5)]
```

but you do not want to code that because then matrices `X` and `Y` would contain copies of the data, and you are worried that the dataset might be large.

For a second example, let's pretend that you are processing a panel dataset and making calculations from matrix X within panel. Your code looks something like

```
st_view(id, ., "panelid", 0)
for (i=1; i<=rows(id); i=j+1) {
     j = endobs(id, i)
     st_view(X, (i,j), "v1 v2 ...", 0)
     ...
}
```

where you have previously written function endobs() to be

```
scalar endobs(vector id, scalar i)
{
     scalar   j
     for (j=i+1; j<=rows(id); j++) {
          if (id[j]!=id[i]) return(j-1)
     }
     return(rows(id))
}
```

In any case, there could be a problem. Missing values of variable panelid might not align with missing values of variables v1, v2, etc. The result could be that observation and row numbers are not in accordance or that there appears to be a group that, in fact, has all missing data. The right way to handle the problem is

```
st_view(M, ., "panelid v1 v2 ...", 0)
st_subview(id, M, ., 1)
for (i=1; i<=rows(id); i=j+1) {
     j = endobs(id, i)
     st_subview(X, M, (i,j), (2\cols(M)))
     ...
}
```

Conformability

st_subview(X, V, i, j):

input:

V:	$r \times c$		
i:	1×1, $n \times 1$,	or	$n_2 \times 2$
j:	1×1, $1 \times k$,	or	$2 \times k_2$

output:

X:	$n \times k$

Diagnostics

st_subview(X, V, i, j) aborts with error if i or j are out of range. i and j refer to row and column numbers of V, not observation and variable numbers of the underlying Stata dataset.

Also see

[M-5] **select( )** — Select rows, columns, or indices

[M-5] **st_view( )** — Make matrix that is a view onto current Stata dataset

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

For suggested citations, see the FAQ on citing Stata documentation.