unab — Unabbreviate variable list

Syntax

Expand and unabbreviate standard variable lists

```
unab lmacname : [varlist] [, min(#) max(#) name(string)]
```

Expand and unabbreviate variable lists that may contain time-series operators

```
tsunab lmacname : [varlist] [, min(#) max(#) name(string)]
```

Expand and unabbreviate variable lists that may contain time-series operators or factor variables

```
fvunab lmacname : [varlist] [, min(#) max(#) name(string)]
```

Description

unab expands and unabbreviates a varlist (see [U] 11.4 varlists) of existing variables, placing the result in the local macro lmacname. unab is a low-level parsing command. The syntax command is a high-level parsing command that, among other things, also unabbreviates variable lists; see [P] syntax.

The difference between unab and tsunab is that tsunab allows time-series operators in varlist; see [U] 11.4.4 Time-series varlists.

The difference between tsunab and fvunab is that fvunab allows factor variables in varlist; see [U] 11.4.3 Factor variables.

Options

min(#) specifies the minimum number of variables allowed. The default is min(1).
max(#) specifies the maximum number of variables allowed. The default is max(32000).
name(string) provides a label that is used when printing error messages.

Remarks and examples

Usually, the syntax command will automatically unabbreviate variable lists; see [P] syntax. In a few cases, unab will be needed to obtain unabbreviated variable lists.

If the user has previously set varabbrev off, then variable abbreviations are not allowed. Then typing in a variable abbreviation results in a syntax error. See [R] set.
Example 1

The `separate` command (see [D] `separate`) provides an example of the use of `unab`. Required option `by(byvar | exp)` takes either a variable name or an expression. This is not handled automatically by the `syntax` command.

Here the `syntax` command for `separate` takes the form

```
syntax varname [if] [in], BY(string) [ other options]
```

After `syntax` performs the command-line parsing, the local variable `by` contains what the user entered for the option. We now need to determine if it is an existing variable name or an expression. If it is a variable name, we may need to expand it.

```
capture confirm var `by'
if _rc == 0 {
    unab by: `by', max(1) name(by())
}
else {
    (parse `by' as an expression)
}
```

Example 2

We interactively demonstrate the `unab` command with the `auto` dataset.

```stata
. use http://www.stata-press.com/data/r13/auto
    (1978 Automobile Data)
. unab x : mpg wei for, name(myopt())
. display "'x'
    mpg weight foreign
. unab x : junk
    variable junk not found
    r(111);
. unab x : mpg wei, max(1) name(myopt())
    myopt(): too many variables specified
    1 variable required
    r(103);
. unab x : mpg wei, max(1) name(myopt()) min(0)
    myopt(): too many variables specified
    0 or 1 variables required
    r(103);
. unab x : mpg wei, min(3) name(myopt())
    myopt(): too few variables specified
    3 or more variables required
    r(102);
. unab x : mpg wei, min(3) max(10)
    myopt(): too few variables specified
    3 - 10 variables required
    r(102);
. unab x : mpg wei, min(3) max(10)
    mpg weight:
    too few variables specified
    r(102);
```
Example 3

If we created a time variable and used `tsset` to declare the dataset as a time series, we can also expand time-series variable lists.

```stata
. generate time = _n
. tsset time
. tsunab mylist : l(1/3).mpg
. display "mylist"
L.mpg L2.mpg L3.mpg
. tsunab mylist : l(1/3).(price turn displ)
. di "mylist"
```

Example 4

If `set varabbrev off` has been issued, variable abbreviations are not allowed:

```stata
. unab varn : mp
. display "varn"
mpg
. set varabbrev off
. unab varn : mp
variable mp not found
r(111);
. set varabbrev on
. unab varn : mp
. display "varn"
mpg
```

Reference


Also see

- [P] syntax — Parse Stata syntax
- [P] varabbrev — Control variable abbreviation
- [U] 11 Language syntax
- [U] 18 Programming Stata