numlist — Parse numeric lists

Description Syntax Options Remarks and examples Stored results Also see

Description

The `numlist` command expands the numeric list supplied as a string argument and performs error checking based on the options specified. Any numeric sequence operators in the *numlist* string are evaluated, and the expanded list of numbers is returned in `r(numlist)`. See **[U] 11.1.8 numlist** for a discussion of numeric lists.

Syntax

`numlist "`*numlist*`" [ , ascending descending integer missingokay min(#) max(#)`
`range(`*operator* # [*operator* #]`) sort ]`

numlist consists of one or more *numlist_elements* shown below.

operator is as follows: < | <= | > | >=.

There is no space between *operator* and #; for example,

```
range(>=0)
range(>0 <=50)
```

numlist_element	Example	Expands to	Definition
#	`3.82`	3.82	a number
.	.	.	a missing value
$\#_1$/$\#_2$	`4/6` `2.3/5.7`	4 5 6 2.3 3.3 4.3 5.3	starting at $\#_1$, increment by 1 to $\#_2$
$\#_1$(`:`)$\#_2$	`2:10` `4.8:9.9`	2 5 8 4.8 6.9 9	starting at $\#_1$, increment by $\#_2$ to $\#_3$
$\#_1$`[:]`$\#_2$	`2[3]10` `4.8[2.1]9.9`	2 5 8 4.8 6.9 9	starting at $\#_1$, increment by $\#_2$ to $\#_3$
$\#_1$ $\#_2$ `:` $\#_3$	`5 7 : 13` `1.1 2.4 : 5.8`	5 7 9 11 13 1.1 2.4 3.7 5	starting at $\#_1$, increment by $(\#_2 - \#_1)$ to $\#_3$
$\#_1$ $\#_2$ `to` $\#_3$	`5 7 to 13` `1.1 2.4 to 5.8`	same	same

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Options

`ascending` indicates that the user must give the numeric list in ascending order without repeated values. This is different from the `sort` option.

`descending` indicates that the numeric list must be given in descending order without repeated values.

`integer` specifies that the user may give only integer values in the numeric list.

`missingokay` indicates that missing values are allowed in the numeric list. By default, missing values are not allowed.

`min(#)` specifies the minimum number of elements allowed in the numeric list. The default is `min(1)`. If you want to allow empty numeric lists, specify `min(0)`.

`max(#)` specifies the maximum number of elements allowed in the numeric list. The default is `max(1600)`, which is the largest allowed maximum.

`range(`*operator* # [*operator* #]`)` specifies the acceptable range for the values in the numeric list. The *operator*s are < (less than), <= (less than or equal to), > (greater than), and >= (greater than or equal to). No space is allowed between the *operator* and the #.

`sort` specifies that the returned numeric list be sorted. This is different from the `ascending` option, which places the responsibility for providing a sorted list on the user who will not be allowed to enter a nonsorted list. `sort`, on the other hand, puts no restriction on the user and takes care of sorting the list. Repeated values are also allowed with `sort`.

Remarks and examples

Programmers rarely use the `numlist` command because `syntax` also expands numeric lists, and it handles the rest of the parsing problem, too, at least if the command being parsed follows standard syntax. `numlist` is used for expanding numeric lists when what is being parsed does not follow standard syntax.

▷ Example 1

We demonstrate the `numlist` command interactively.

```
. numlist "5.3 1.0234 3 6:18 -2.0033 5.3/7.3"
. display "`r(numlist)'"
5.3 1.0234 3 6 9 12 15 18 -2.0033 5.3 6.3 7.3
. numlist "5.3 1.0234 3 6:18 -2.0033 5.3/7.3", integer
invalid numlist has noninteger elements
r(126);
. numlist "1 5 8/12 15", integer descending
invalid numlist has elements out of order
r(124);
. numlist "1 5 8/12 15", integer ascending
. display "`r(numlist)'"
1 5 8 9 10 11 12 15
```

```
. numlist "100 1 5 8/12 15", integer ascending
invalid numlist has elements out of order
r(124);
. numlist "100 1 5 8/12 15", integer sort
. display "`r(numlist)'"
1 5 8 9 10 11 12 15 100
. numlist "3 5 . 28 -3(2)5"
invalid numlist has missing values
r(127);
. numlist "3 5 . 28 -3(2)5", missingokay min(3) max(25)
. display "`r(numlist)'"
3 5 . 28 -3 -1 1 3 5
. numlist "28 36", min(3) max(6)
invalid numlist has too few elements
r(122);
. numlist "28 36 -3 5 2.8 7 32 -8", min(3) max(6)
invalid numlist has too many elements
r(123);
. numlist "3/6 -4 -1 to 5", range(>=1)
invalid numlist has elements outside of allowed range
r(125);
. numlist "3/6", range(>=0 <30)
. display "`r(numlist)'"
3 4 5 6
```

Stored results

`numlist` stores the following in `r()`:

Macros

`r(numlist)`	expanded numeric list

Also see

[P] **syntax** — Parse Stata syntax

[U] **11.1.8 numlist**

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.

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