

## subscripts — Use of subscripts

Description	Syntax	Remarks and examples	Conformability
Diagnostics	Reference	Also see	

## Description

Subscripts come in two styles.

In `[subscript]` syntax—called list subscripts—an element or a matrix is specified:

<code>x[1,2]</code>	the 1,2 element of $x$ ; a scalar
<code>x[(1\3\2), (4,5)]</code>	the $3 \times 2$ matrix composed of rows 1, 3, and 2 and columns 4 and 5 of $x$ :

$$\begin{bmatrix} x_{14} & x_{15} \\ x_{34} & x_{35} \\ x_{24} & x_{25} \end{bmatrix}$$

In `[|subscript|]` syntax—called range subscripts—an element or a contiguous submatrix is specified:

<code>x[ 1,2 ]</code>	same as <code>x[1,2]</code> ; a scalar
<code>x[ 2,3 \ 4,7 ]</code>	$3 \times 4$ submatrix of $x$ :

$$\begin{bmatrix} x_{23} & x_{24} & x_{25} & x_{26} & x_{27} \\ x_{33} & x_{34} & x_{35} & x_{36} & x_{37} \\ x_{43} & x_{44} & x_{45} & x_{46} & x_{47} \end{bmatrix}$$

Both style subscripts may be used in expressions and may be used on the left-hand side of the equal-assignment operator.

## Syntax

`x[real vector r, real vector c]`

`x[|real matrix sub|]`

Subscripts may be used on the left or right of the [equal-assignment operator](#).

## Remarks and examples

Remarks are presented under the following headings:

*List subscripts*

*Range subscripts*

*When to use list subscripts and when to use range subscripts*

*A fine distinction*

**List subscripts**

List subscripts—also known simply as subscripts—are obtained when you enclose the subscripts in square brackets, [ and ]. List subscripts come in two basic forms:

$x[ivec, jvec]$	matrix composed of rows <i>ivec</i> and columns <i>jvec</i> of matrix <i>x</i>
$v[kvec]$	vector composed of elements <i>kvec</i> of vector <i>v</i>

where *ivec*, *jvec*, *kvec* may be a vector or a scalar, so the two basic forms include

$x[i, j]$	scalar <i>i, j</i> element
$x[i, jvec]$	row vector of row <i>i</i> , elements <i>jvec</i>
$x[ivec, j]$	column vector of column <i>j</i> , elements <i>ivec</i>
$v[k]$	scalar <i>k</i> th element of vector <i>v</i>

Also missing value may be specified to mean all the rows or all the columns:

$x[i, .]$	row vector of row <i>i</i> of <i>x</i>
$x[. , j]$	column vector of column <i>j</i> of <i>x</i>
$x[ivec, .]$	matrix of rows <i>ivec</i> , all columns
$x[. , jvec]$	matrix of columns <i>jvec</i> , all rows
$x[. , .]$	the entire matrix

Finally, Mata assumes missing value when you omit the argument entirely:

$x[i, ]$	same as $x[i, .]$
$x[ivec, ]$	same as $x[ivec, .]$
$x[, j]$	same as $x[. , j]$
$x[, jvec]$	same as $x[. , jvec]$
$x[, ]$	same as $x[. , .]$

Good style is to specify *ivec* as a column vector and *jvec* as a row vector, but that is not required:

$x[(1\2\3), (1,2,3)]$	good style
$x[(1,2,3), (1,2,3)]$	same as $x[(1\2\3), (1,2,3)]$
$x[(1\2\3), (1\2\3)]$	same as $x[(1\2\3), (1,2,3)]$
$x[(1,2,3), (1\2\3)]$	same as $x[(1\2\3), (1,2,3)]$

Similarly, good style is to specify *kvec* as a column when *v* is a column vector and to specify *kvec* as a row when *v* is a row vector, but that is not required and what is returned is a column vector if *v* is a column and a row vector if *v* is a row:

$rowv[(1,2,3)]$	good style for specifying row vector
$rowv[(1\2\3)]$	same as $rowv[(1,2,3)]$
$colv[(1\2\3)]$	good style for specifying column vector
$colv[(1,2,3)]$	same as $colv[(1\2\3)]$

Subscripts may be used in expressions following a variable name:

```
first = list[1]
multiplier = x[3,4]
result = colsum(x[,j])
```

Subscripts may be used following an expression to extract a submatrix from a result:

```
allneeded = invsym(x)[(1::4), .] * multiplier
```

Subscripts may be used on the left-hand side of the equal-assignment operator:

```
x[1,1] = 1
x[1,.] = y[3,.]
x[(1::4), (1..4)] = I(4)
```

## Range subscripts

Range subscripts appear inside the difficult to type `[|` and `|]` brackets. Range subscripts come in four basic forms:

<code>x[ i,j ]</code>	<i>i,j</i> element; same result as <code>x[i,j]</code>
<code>v[ k ]</code>	<i>k</i> th element of vector; same result as <code>v[k]</code>
<code>x[ i,j \ k,l ]</code>	submatrix, vector, or scalar formed using ( <i>i,j</i> ) as top-left corner and ( <i>k,l</i> ) as bottom-right corner
<code>v[ i \ k ]</code>	subvector or scalar of elements <i>i</i> through <i>k</i> ; result is row vector if <i>v</i> is row vector, column vector if <i>v</i> is column vector

Missing value may be specified for a row or column to mean all rows or all columns when a  $1 \times 2$  or  $1 \times 1$  subscript is specified:

<code>x[ i, . ]</code>	row <i>i</i> of <i>x</i> ; same as <code>x[i, .]</code>
<code>x[  . , j ]</code>	column <i>j</i> of <i>x</i> ; same as <code>x[ . , j]</code>
<code>x[  . , . ]</code>	entire matrix; same as <code>x[ . , .]</code>
<code>v[  .  ]</code>	entire vector; same as <code>v[.]</code>

Also missing may be specified to mean the number of rows or the number of columns of the matrix being subscripted when a  $2 \times 2$  subscript is specified:

<code>x[ 1,2 \ 4, . ]</code>	equivalent to <code>x[ 1,2 \ 4, cols(x) ]</code>
<code>x[ 1,2 \ . , 3 ]</code>	equivalent to <code>x[ 1,2 \ rows(x), 3 ]</code>
<code>x[ 1,2 \ . , . ]</code>	equivalent to <code>x[ 1,2 \ rows(x), cols(x) ]</code>

With range subscripts, what appears inside the square brackets is in all cases interpreted as a matrix expression, so in

```
sub = (1,2)
... x[|sub|] ...
```

`x[sub]` refers to `x[1,2]`.

## 4 **subscripts** — Use of subscripts

---

Range subscripts may be used in all the same contexts as list subscripts; they may be used in expressions following a variable name

```
submat = result[|1,1 \ 3,3|]
```

they may be used to extract a submatrix from a calculated result

```
allneeded = invsym(x)[|1,1 \ 4,4|]
```

and they may be used on the left-hand side of the equal-assignment operator:

```
x[|1,1 \ 4,4|] = I(4)
```

### **When to use list subscripts and when to use range subscripts**

Everything a range subscript can do, a list subscript can also do. The one seemingly unique feature of a range subscript,

```
x[|i1,j1 \ i2,j2|]
```

is perfectly mimicked by

```
x[(i1:i2), (j1..j2)]
```

The range-subscript construction, however, executes more quickly, and so that is the purpose of range subscripts: to provide a fast way to extract contiguous submatrices. In all other cases, use list subscripts because they are faster.

Use list subscripts to refer to scalar values:

```
result = x[1,3]
x[1,3] = 2
```

Use list subscripts to extract entire rows or columns:

```
obs = x[., 3]
var = x[4, .]
```

Use list subscripts to permute the rows and columns of matrices:

```
: x = (1,2,3,4 \ 5,6,7,8 \ 9,10,11,12)
```

```
: y = x[(1\3\2), .]
```

```
: y
```

	1	2	3	4
1	1	2	3	4
2	9	10	11	12
3	5	6	7	8

```
: y = x[., (1,3,2,4)]
```

```
: y
```

	1	2	3	4
1	1	3	2	4
2	5	7	6	8
3	9	11	10	12

```
: y=x[(1\3\2), (1,3,2,4)]
```

```

: y
      1   2   3   4
1   [ 1   3   2   4 ]
2   [ 9  11  10  12 ]
3   [ 5   7   6   8 ]

```

Use list subscripts to duplicate rows or columns:

```
: x = (1,2,3,4 \ 5,6,7,8 \ 9,10,11,12)
```

```
: y = x[(1\2\3\1), .]
```

```

: y
      1   2   3   4
1   [ 1   2   3   4 ]
2   [ 5   6   7   8 ]
3   [ 9  10  11  12 ]
4   [ 1   2   3   4 ]

```

```
: y = x[., (1,2,3,4,2)]
```

```

: y
      1   2   3   4   5
1   [ 1   2   3   4   2 ]
2   [ 5   6   7   8   6 ]
3   [ 9  10  11  12  10 ]

```

```
: y = x[(1\2\3\1), (1,2,3,4,2)]
```

```

: y
      1   2   3   4   5
1   [ 1   2   3   4   2 ]
2   [ 5   6   7   8   6 ]
3   [ 9  10  11  12  10 ]
4   [ 1   2   3   4   2 ]

```

## A fine distinction

There is a fine distinction between  $x[i, j]$  and  $x[|i, j|]$ . In  $x[i, j]$ , there are two arguments,  $i$  and  $j$ . The comma separates the arguments. In  $x[|i, j|]$ , there is one argument:  $i, j$ . The comma is the [column-join operator](#).

In Mata, comma means mostly the column-join operator:

```

newvec = oldvec, addedvalues
qsum = (x, 1)'(x, 1)

```

There are, in fact, only two exceptions. When you type the arguments for a function, the comma separates one argument from the next:

```
result = f(a,b,c)
```

In the above example,  $f()$  receives three arguments:  $a$ ,  $b$ , and  $c$ . If we wanted  $f()$  to receive one argument,  $(a, b, c)$ , we would have to enclose the calculation in parentheses:

```
result = f((a,b,c))
```

That is the first exception. When you type the arguments inside a function, comma means argument separation. You get back to the usual meaning of comma—the column-join operator—by opening another set of parentheses.

The second exception is in [list subscripting](#):

$$x[i,j]$$

Inside the list-subscript brackets, comma means argument separation. That is why you have seen us type vectors inside parentheses:

$$x[(1\ 2\ 3), (1, 2, 3)]$$

These are the two exceptions. Range subscripting is not an exception. Thus in

$$x[| i, j |]$$

there is one argument,  $i, j$ . With range subscripts, you may program constructs such as

```
IJ      = (i,j)
RANGE  = (1,2 \ 4,4)
...
... x[|IJ|] ... x[|RANGE|] ...
```

You may not code in this way with list subscripts. In particular,  $x[IJ]$  would be interpreted as a request to extract elements  $i$  and  $j$  from vector  $x$ , and would be an error otherwise.  $x[RANGE]$  would always be an error.

We said earlier that list subscripts  $x[i, j]$  are a little faster than range subscripts  $x[|i, j|]$ . That is true, but if  $IJ=(i, j)$  already,  $x[|IJ|]$  is faster than  $x[i, j]$ . You would, however, have to execute many millions of references to  $x[|IJ|]$  before you could measure the difference.

## Conformability

$x[i, j]$ :

$x$ :	$r \times c$			
$i$ :	$m \times 1$	or	$1 \times m$	(does not matter which)
$j$ :	$1 \times n$	or	$n \times 1$	(does not matter which)
result:	$m \times n$			

$x[i, .]$ :

$x$ :	$r \times c$			
$i$ :	$m \times 1$	or	$1 \times m$	(does not matter which)
result:	$m \times c$			

$x[. , j]$ :

$x$ :	$r \times c$			
$j$ :	$1 \times n$	or	$n \times 1$	(does not matter which)
result:	$r \times n$			

$x[. , .]$ :

$x$ :	$r \times c$			
result:	$r \times c$			

$x[i]$ :

$x$ :	$n \times 1$		$1 \times n$
$i$ :	$m \times 1$	or	$1 \times m$
<i>result</i> :	$m \times 1$		$1 \times m$ or $m \times 1$

$x[.]$ :

$x$ :	$n \times 1$		$1 \times n$
<i>result</i> :	$n \times 1$		$1 \times n$

$x[|k|]$ :

$x$ :	$r \times c$
$k$ :	$1 \times 2$
<i>result</i> :	$1 \times 1$ if $k[1] < .$ and $k[2] < .$ $r \times 1$ if $k[1] >= .$ and $k[2] < .$ $1 \times c$ if $k[1] < .$ and $k[2] >= .$ $r \times c$ if $k[1] >= .$ and $k[2] >= .$

$x[|k|]$ :

$x$ :	$r \times c$
$k$ :	$2 \times 2$
<i>result</i> :	$k[2,1] - k[1,1] + 1 \times k[2,2] - k[1,2] + 1$ (in the above formula, if $k[2,1] >= .$ , treat as if $k[2,1] = r$ , and similarly, if $k[2,2] >= .$ , treat as if $k[2,2] = c$ )

$x[|k|]$ :

$x$ :	$r \times 1$	$1 \times c$
$k$ :	$2 \times 1$	$2 \times 1$
<i>result</i> :	$k[2] - k[1] + 1 \times 1$ (if $k[2] >= .$ , treat as if $k[2] = r$ )	$1 \times k[2] - k[1] + 1$ (if $k[2] >= .$ , treat as if $k[2] = c$ )

## Diagnostics

Both styles of subscripts abort with error if the subscript is out of range, if a reference is made to a nonexisting row or column.

## Reference

Gould, W. W. 2007. *Mata Matters: Subscripting*. *Stata Journal* 7: 106–116.

## Also see

[M-2] [intro](#) — Language definition