

**op\_join** — Row- and column-join operators

[Description](#)    [Syntax](#)    [Remarks and examples](#)    [Conformability](#)  
[Diagnostics](#)    [Also see](#)

## Description

, and \ are Mata's row-join and column-join operators.

## Syntax

$a , b$

$a \setminus b$

## Remarks and examples

stata.com

Remarks are presented under the following headings:

*Comma and backslash are operators*

*Comma as a separator*

*Warning about the misuse of comma and backslash operators*

### Comma and backslash are operators

That , and \ are operators cannot be emphasized enough. When one types

```
: (1, 2 \ 3, 4)
      1  2
```

1	1	2
2	3	4

one is tempted to think, "Ah, comma and backslash are how you separate elements when you enter a matrix." If you think like that, you will not appreciate the power of , and \.

, and \ are operators in the same way that \* and + are operators.

, is the operator that takes a  $r \times c_1$  matrix and a  $r \times c_2$  matrix, and returns a  $r \times (c_1 + c_2)$  matrix.

\ is the operator that takes a  $r_1 \times c$  matrix and a  $r_2 \times c$  matrix, and returns a  $(r_1 + r_2) \times c$  matrix.

, and \ may be used with scalars, vectors, or matrices:

```
: a = (1 \ 2)
```

```
: b = (3 \ 4)
```

```

: a, b
      1  2
1   1  3
2   2  4

```

```

: c = (1, 2)
: d = (3, 4)
: c \ d
      1  2
1   1  2
2   3  4

```

, binds more tightly than \, meaning that  $e, f \setminus g, h$  is interpreted as  $(e, f) \setminus (g, h)$ . In this, , and \ are no different from \* and + operators: \* binds more tightly than + and  $e*f + g*h$  is interpreted as  $(e*f)+(g*h)$ .

Just as it sometimes makes sense to type  $e*(f+g)*h$ , it can make sense to type  $e, (f \setminus g), h$ :

```

: e = 1 \ 2
: f = 5 \ 6
: g = 3
: h = 4
: e, (g \ h), f
      1  2  3
1   1  3  5
2   2  4  6

```

## Comma as a separator

, has a second meaning in Mata: it is the argument separator for functions. When you type

```

: myfunc(a, b)

```

the comma that appears inside the parentheses is not the comma row-join operator; it is the comma argument separator. If you wanted to call myfunc() with second argument equal to row vector (1,2), you must type

```

: myfunc(a, (1,2))

```

and not

```

: myfunc(a, 1, 2)

```

because otherwise Mata will think you are trying to pass three arguments to myfunc(). When you open another set of parentheses inside a function's argument list, comma reverts to its usual row-join meaning.

## Warning about the misuse of comma and backslash operators

Misuse or mere overuse of `,` and `\` can substantially reduce the speed with which your code executes. Consider the actions Mata must take when you code, say,

$$a \setminus b$$

First, Mata must allocate a matrix or vector containing `rows(a)+rows(b)` rows, then it must copy `a` into the new matrix or vector, and then it must copy `b`. Nothing inefficient has happened yet, but now consider

$$(a \setminus b) \setminus c$$

Picking up where we left off, Mata must allocate a matrix or vector containing `rows(a)+rows(b)+rows(c)` rows, then it must copy `(a \setminus b)` into the new matrix or vector, and then it must copy `c`. Something inefficient just happened: `a` was copied twice!

Coding

$$res = (a \setminus b) \setminus c$$

is convenient, but execution would be quicker if we coded

```
res = J(rows(a)+rows(b)+rows(c), cols(a), .)
res[1,.] = a
res[2,.] = b
res[3,.] = c
```

We do not want to cause you concern where none is due. In general, you would not be able to measure the difference between the more efficient code and coding `res = (a \setminus b) \setminus c`. But as the number of row or column operators stack up, the combined result becomes more and more inefficient. Even that is not much of a concern. If the inefficient construction itself is buried in a loop, however, and that loop is executed thousands of times, the inefficiency can become important.

With a little thought, you can always substitute predeclaration using `J()` (see [M-5] `J()`) and assignment via subscripting.

## Conformability

$a, b$ :

$a$ :	$r \times c_1$
$b$ :	$r \times c_2$
<b>result:</b>	$r \times (c_1 + c_2)$

$a \setminus b$ :

$a$ :	$r_1 \times c$
$b$ :	$r_2 \times c$
<b>result:</b>	$(r_1 + r_2) \times c$

## Diagnostics

`,` and `\` abort with error if `a` and `b` are not of the same broad type.

## Also see

[M-2] [exp](#) — Expressions

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

Stata, Stata Press, and Mata are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2023 StataCorp LLC, College Station, TX, USA. All rights reserved.

