op_logical — Logical operators

Description

The operators above perform logical comparisons, and operator ! performs logical negation. All operators evaluate to 1 or 0, meaning true or false.

Syntax

a == *b*	true if *a* equals *b*
a != *b*	true if *a* not equal to *b*
a > *b*	true if *a* greater than *b*
a >= *b*	true if *a* greater than or equal to *b*
a < *b*	true if *a* less than *b*
a <= *b*	true if *a* less than or equal to *b*
!*a*	logical negation; true if *a*==0 and false otherwise
a & *b*	true if *a*!=0 and *b*!=0
a	*b*	true if *a*!=0 or *b*!=0
a && *b*	synonym for *a* & *b*
a || *b*	synonym for *a*	*b*

Remarks and examples

Remarks are presented under the following headings:

Introduction

The operators above work as you would expect when used with scalars, and the comparison operators and the not operator have been generalized for use with matrices.

a==*b* evaluates to true if *a* and *b* are p-conformable, of the same type, and the corresponding elements are equal. Of the same type means *a* and *b* are both numeric, both strings, or both pointers. Thus it is not an error to ask if a 2 $\times$ 2 matrix is equal to a 4 $\times$ 1 vector or if a string variable is equal to a real variable; they are not. Also *a*==*b* is declared to be true if *a* or *b* are p-conformable and the number of rows or columns is zero.

a!=*b* is equivalent to !(*a*==*b*). *a*!=*b* evaluates to true when *a*==*b* would evaluate to false and evaluates to true otherwise.

The remaining comparison operators `>`, `>=`, `<`, and `<=` work differently from `==` and `!=` in that they require *a* and *b* be p-conformable; if they are not, they abort with error. They return true if the corresponding elements have the stated relationship, and return false otherwise. If *a* or *b* is complex, the comparison is made in terms of the length of the complex vector; for instance, $a>b$ is equivalent to `abs(`*a*`)>abs(`*b*`)`, and so `-3>2+0i` is true.

`!`*a*, when *a* is a scalar, evaluates to 0 if *a* is not equal to zero and 1 otherwise. Applied to a vector or matrix, the same operation is carried out, element by element: `!(-1,0,1,2,.)` evaluates to `(0,1,0,0,0)`.

`&` and `|` (*and* and *or*) may be used with scalars only. Because so many people are familiar with programming in the C language, Mata provides `&&` as a synonym for `&` and `||` as a synonym for `|`.

Use of logical operators with pointers

In a pointer expression, `NULL` is treated as false and all other pointer values (address values) are treated as true. Thus the following code is equivalent

```
pointer x                        pointer x
...                               ...
if (x) {                         if (x!=NULL) {
        ...                          ...
}                                }
```

The logical operators *a*`==`*b*, *a*`!=`*b*, *a*`&`*b*, and *a*`|`*b* may be used with pointers.

Conformability

a`==`*b*, *a*`!=`*b*:

a:	$r_1 \times c_1$
b:	$r_2 \times c_2$
result:	1×1

$a>b$, $a>=b$, $a<b$, $a<=b$:

a:	$r \times c$
b:	$r \times c$
result:	1×1

`!`*a*:

a:	$r \times c$
result:	$r \times c$

a`&`*b*, *a*`|`*b*:

a:	1×1
b:	1×1
result:	1×1

Diagnostics

a`==`*b* and *a*`!=`*b* cannot fail.

$a>b$, $a>=b$, $a<b$, $a<=b$ abort with error if *a* and *b* are not p-conformable, if *a* and *b* are not of the same general type (numeric and numeric or string and string), or if *a* or *b* are pointers.

!*a* aborts with error if *a* is not real.

a&*b* and *a*|*b* abort with error if *a* and *b* are not both real or not both pointers. If *a* and *b* are pointers, pointer value NULL is treated as false and all other pointer values are treated as true. In all cases, a real equal to 0 or 1 is returned.

Also see

[M-2] **exp** — Expressions

[M-2] **Intro** — Language definition

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