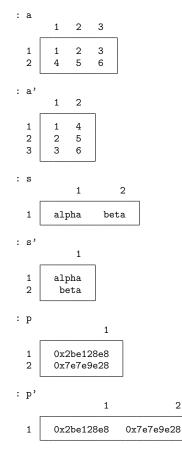## Description

A' returns the transpose of A or, if A is complex, the conjugate transpose.

Syntax

A'

Remarks and examples

The $'$ postfix operator may be used on any type of matrix or vector: real, complex, string, or pointer:

```
: a
         1    2    3
     ┌                  ┐
   1 │   1    2    3    │
   2 │   4    5    6    │
     └                  ┘

: a'
         1    2
     ┌            ┐
   1 │   1    4   │
   2 │   2    5   │
   3 │   3    6   │
     └            ┘

: s
            1         2
     ┌                     ┐
   1 │   alpha     beta    │
     └                     ┘

: s'
            1
     ┌            ┐
   1 │   alpha    │
   2 │    beta    │
     └            ┘

: p
               1
     ┌                  ┐
   1 │   0x2be128e8     │
   2 │   0x7e7e9e28     │
     └                  ┘

: p'
               1              2
     ┌                              ┐
   1 │   0x2be128e8    0x7e7e9e28   │
     └                              ┘
```

```
: z
             1         2

1      1 + 2i    3 + 4i
2      5 + 6i    7 + 8i

: z'
             1         2

1      1 - 2i    5 - 6i
2      3 - 4i    7 - 8i
```

When $'$ is applied to a complex, returned is the conjugate transpose. If you do not want this, code conj(z') or conj(z)$'$—it makes no difference; see [M-5] **conj( )**,

```
: conj(z')
             1         2

1      1 + 2i    5 + 6i
2      3 + 4i    7 + 8i
```

Or use the transposeonly( ) function; see [M-5] **transposeonly( )**:

```
: transposeonly(z)
             1         2

1      1 + 2i    5 + 6i
2      3 + 4i    7 + 8i
```

transposeonly( ) executes slightly faster than conj(z').

For real and complex A, also see [M-5] **_transpose( )**, which provides a way to transpose a matrix in place and so saves memory.

Conformability

A':

$$A: \quad r \times c$$
$$result: \quad c \times r$$

Diagnostics

The transpose operator cannot fail, but it is easy to use it incorrectly when working with complex quantities.

A user wanted to form $A*x$ but when he tried, got a conformability error. He thought x was a column vector, but it turned out to be a row vector, or perhaps it was the other way around. Anyway, he then coded $A*x'$, and the program worked and, even better, produced the correct answers. In his test, x was real.

Later, the user ran the program with complex x, and the program generated incorrect results, although it took him a while to notice. Study and study his code he did, before he thought about the innocuous $A*x'$. The transpose operator not only had changed x from being a row into being a column but also had taken the conjugate of each element of x! He changed the code to read $A*\texttt{transposeonly}(x)$.

The user no doubt wondered why the $'$ transpose operator was not defined at the outset to be equivalent to `transposeonly()`. If it had been, then rather than telling the story of the man who was bitten by conjugate transpose when he only wanted the transpose, we would have told the story of the woman who was bitten by the transpose when she needed the conjugate transpose. There are, in fact, more of the latter stories than there are of the former.

Also see

[M-2] **exp** — Expressions

[M-5] **conj( )** — Complex conjugate

[M-5] **_transpose( )** — Transposition in place

[M-5] **transposeonly( )** — Transposition without conjugation

[M-2] **Intro** — Language definition