Title

transposeonly( ) — Transposition without conjugation

Syntax

> *numeric matrix*   transposeonly(*numeric matrix A*)
>
> *void*   _transposeonly(*numeric matrix A*)

Description

transposeonly(*A*) returns *A* with its rows and columns interchanged. When *A* is real, the actions of transposeonly(*A*) are indistinguishable from coding A'; see [M-2] **op_transpose**. The returned result is the same, and the execution time is the same, too. When *A* is complex, however, transposeonly(*A*) is equivalent to coding conj(A'), but transposeonly() obtains the result more quickly.

_transposeonly(*A*) interchanges the rows and columns of *A* in place—without use of additional memory—and returns the transposed (but not conjugated) result in *A*.

Remarks and examples

transposeonly() is useful when you are coding in the programming, rather than the mathematical, sense. Say that you have two row vectors, a and b, and you want to place the two vectors together in a matrix R, and you want to turn them into column vectors. If a and b were certain to be real, you could just code

```
R = (a', b')
```

The above line, however, would result in not just the organization but also the values recorded in R changing if a or b were complex. The solution is to code

```
R = (transposeonly(a), transposeonly(b))
```

The above line will work for real or complex a and b. If you were concerned about memory consumption, you could instead code

```
R = (a \ b)
_transposeonly(R)
```

Conformability

transposeonly(*A*):

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

$\_$transposeonly(*A*):
>   *input*:
>>   *A*:   *r* × *c*
>   *output*:
>>   *A*:   *c* × *r*

Diagnostics

$\_$transposeonly(*A*) aborts with error if *A* is a view.

Also see

[M-2] **op$\_$transpose** — Conjugate transpose operator

[M-5] **$\_$transpose( )** — Transposition in place

[M-4] **manipulation** — Matrix manipulation