**Syntax**

transmorphic matrix \( J(\text{real scalar } r, \text{ real scalar } c, \text{ scalar } val) \)

transmorphic matrix \( J(\text{real scalar } r, \text{ real scalar } c, \text{ matrix } mat) \)

**Description**

\( J(r, c, val) \) returns an \( r \times c \) matrix with each element equal to \( val \).

\( J(r, c, mat) \) returns an \((r*\text{rows(mat)}) \times (c*\text{cols(mat)})\) matrix with elements equal to \( mat \).

The first, \( J(r, c, val) \), is how \( J() \) is commonly used. The first is nothing more than a special case of the second, \( J(r, c, mat) \), when \( mat \) is \( 1 \times 1 \).

**Remarks and examples**

Remarks are presented under the following headings:

*First syntax: \( J(r, c, val) \), \( val \) a scalar

*Second syntax: \( J(r, c, mat) \), \( mat \) a matrix

**First syntax: \( J(r, c, val) \), \( val \) a scalar**

\( J(r, c, val) \) creates matrices of constants. For example, \( J(2, 3, 0) \) creates

\[
\begin{array}{ccc}
1 & 2 & 3 \\
1 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 \\
\end{array}
\]

\( J() \) must be typed in uppercase.

\( J() \) can create any type of matrix:

<table>
<thead>
<tr>
<th>Function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J(2, 3, 4) )</td>
<td>( 2 \times 3 ) real matrix, each element = 4</td>
</tr>
<tr>
<td>( J(2, 3, 4+5i) )</td>
<td>( 2 \times 3 ) complex matrix, each element = 4 + 5i</td>
</tr>
<tr>
<td>( J(2, 3, &quot;hi&quot;) )</td>
<td>( 2 \times 3 ) string matrix, each element = &quot;hi&quot;</td>
</tr>
<tr>
<td>( J(2, 3, &amp;x) )</td>
<td>( 2 \times 3 ) pointer matrix, each element = address of ( x )</td>
</tr>
</tbody>
</table>
Also, J() can create void matrices:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>J(0, 0, .)</td>
<td>0 × 0 real</td>
</tr>
<tr>
<td>J(0, 1, .)</td>
<td>0 × 1 real</td>
</tr>
<tr>
<td>J(1, 0, .)</td>
<td>1 × 0 real</td>
</tr>
<tr>
<td>J(0, 0, 1i)</td>
<td>0 × 0 complex</td>
</tr>
<tr>
<td>J(0, 1, 1i)</td>
<td>0 × 1 complex</td>
</tr>
<tr>
<td>J(1, 0, 1i)</td>
<td>1 × 0 complex</td>
</tr>
<tr>
<td>J(0, 0, &quot;&quot;)</td>
<td>0 × 0 string</td>
</tr>
<tr>
<td>J(0, 1, &quot;&quot;)</td>
<td>0 × 1 string</td>
</tr>
<tr>
<td>J(1, 0, &quot;&quot;)</td>
<td>1 × 0 string</td>
</tr>
<tr>
<td>J(0, 0, NULL)</td>
<td>0 × 0 pointer</td>
</tr>
<tr>
<td>J(0, 1, NULL)</td>
<td>0 × 1 pointer</td>
</tr>
<tr>
<td>J(1, 0, NULL)</td>
<td>1 × 0 pointer</td>
</tr>
</tbody>
</table>

When J(r, c, val) is used to create a void matrix, the particular value of the third argument does not matter. Its element type, however, determines the type of matrix produced. Thus, J(0, 0, .), J(0, 0, 1), and J(0, 0, 1/3) all create the same result: a 0 × 0 real matrix. Similarly, J(0, 0, ""), J(0, 0, "name"), and J(0, 0, "?") all create the same result: a 0 × 0 string matrix. See [M-2] void to learn how void matrices are used.

Second syntax: J(r, c, mat), mat a matrix

J(r, c, mat) is a generalization of J(r, c, val). When the third argument is a matrix, that matrix is replicated in the result. For instance, if X is \((1,2\backslash3,4)\), then J(2, 3, X) creates

```
1 2 3 4 5 6
1 1 2 1 2 1 2
2 3 4 3 4 3 4
3 1 2 1 2 1 2
4 3 4 3 4 3 4
```

J(r, c, val) is a special case of J(r, c, mat); it just happens that mat is 1 × 1. The matrix created has \(r*\text{rows(mat)}\) rows and \(c*\text{cols(mat)}\) columns. Note that J(r, c, mat) creates a void matrix if any of \(r, c, \text{rows(mat)}, \text{cols(mat)}\) are zero.
Conformability

\[ J(r, c, val) : \]
\[
\begin{align*}
  r &: 1 \times 1 \\
  c &: 1 \times 1 \\
  val &: 1 \times 1 \\
  result &: r \times c
\end{align*}
\]

\[ J(r, c, mat) : \]
\[
\begin{align*}
  r &: 1 \times 1 \\
  c &: 1 \times 1 \\
  mat &: m \times n \\
  result &: r^m \times c^n
\end{align*}
\]

Diagnostics

\[ J(r, c, val) \text{ and } J(r, c, mat) \text{ abort with error if } r < 0 \text{ or } c < 0, \text{ or if } r \geq m \text{ or } c \geq n. \text{ Arguments } r \text{ and } c \text{ are interpreted as } \text{trunc}(r) \text{ and } \text{trunc}(c). \]

Also see

[M-5] missingof() — Appropriate missing value

[M-4] standard — Functions to create standard matrices