

J() — Matrix of constants

Description
Diagnostics

Syntax
Also see

Remarks and examples

Conformability

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 1×1 .

Syntax

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

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

Remarks and examples

stata.com

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

	1	2	3
1	0	0	0
2	0	0	0

$J()$ must be typed in uppercase.

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

Function	Returns
$J(2, 3, 4)$	2×3 real matrix, each element = 4
$J(2, 3, 4+5i)$	2×3 complex matrix, each element = $4 + 5i$
$J(2, 3, "hi")$	2×3 string matrix, each element = "hi"
$J(2, 3, \&x)$	2×3 pointer matrix, each element = address of x

Also, `J()` can create void matrices:

<code>J(0, 0, .)</code>	<code>0 × 0 real</code>
<code>J(0, 1, .)</code>	<code>0 × 1 real</code>
<code>J(1, 0, .)</code>	<code>1 × 0 real</code>
<code>J(0, 0, 1i)</code>	<code>0 × 0 complex</code>
<code>J(0, 1, 1i)</code>	<code>0 × 1 complex</code>
<code>J(1, 0, 1i)</code>	<code>1 × 0 complex</code>
<code>J(0, 0, "")</code>	<code>0 × 0 string</code>
<code>J(0, 1, "")</code>	<code>0 × 1 string</code>
<code>J(1, 0, "")</code>	<code>1 × 0 string</code>
<code>J(0, 0, NULL)</code>	<code>0 × 0 pointer</code>
<code>J(0, 1, NULL)</code>	<code>0 × 1 pointer</code>
<code>J(1, 0, NULL)</code>	<code>1 × 0 pointer</code>

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\3,4)`, 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*rows(mat)` rows and `c*cols(mat)` columns.

Note that `J(r, c, mat)` creates a void matrix if any of `r`, `c`, `rows(mat)`, or `cols(mat)` are zero.

Conformability

J(*r*, *c*, *val*):

<i>r</i> :	1×1
<i>c</i> :	1×1
<i>val</i> :	1×1
<i>result</i> :	$r \times c$

J(*r*, *c*, *mat*):

<i>r</i> :	1×1
<i>c</i> :	1×1
<i>mat</i> :	$m \times n$
<i>result</i> :	$r*m \times c*n$

Diagnostics

J(*r*, *c*, *val*) and J(*r*, *c*, *mat*) abort with error if $r < 0$ or $c < 0$, or if $r \geq .$ or $c \geq ..$ Arguments *r* and *c* are interpreted as `trunc(r)` and `trunc(c)`.

Also see

- [M-5] [missingof\(\)](#) — Appropriate missing value
- [M-4] [standard](#) — Functions to create standard matrices