## Matrix Utility Functions

### Complex

- **Re()**
  - Function: Re()
  - Purpose: real part
- **Im()**
  - Function: Im()
  - Purpose: imaginary part
- **C()**
  - Function: C()
  - Purpose: make complex

### Shape & Type

- **rows()**
  - Function: rows()
  - Purpose: number of rows
- **cols()**
  - Function: cols()
  - Purpose: number of columns
- **length()**
  - Function: length()
  - Purpose: number of elements of vector
- **eltype()**
  - Function: eltype()
  - Purpose: element type of object
- **orgtype()**
  - Function: orgtype()
  - Purpose: organizational type of object
- **classname()**
  - Function: classname()
  - Purpose: class name of a Mata class scalar
- **structname()**
  - Function: structname()
  - Purpose: struct name of a Mata struct scalar
- **isreal()**
  - Function: isreal()
  - Purpose: object is real matrix
- **iscomplex()**
  - Function: iscomplex()
  - Purpose: object is complex matrix
- **isstring()**
  - Function: isstring()
  - Purpose: object is string matrix
- **ispointer()**
  - Function: ispointer()
  - Purpose: object is pointer matrix
- **isrealvalues()**
  - Function: isrealvalues()
  - Purpose: whether matrix contains only real values
- **isview()**
  - Function: isview()
  - Purpose: whether matrix is view

### Properties

- **issymmetric()**
  - Function: issymmetric()
  - Purpose: whether matrix is symmetric (Hermitian)
- **issymmetriconly()**
  - Function: issymmetriconly()
  - Purpose: whether matrix is mechanically symmetric
- **isdiagonal()**
  - Function: isdiagonal()
  - Purpose: whether matrix is diagonal
- **diag0cnt()**
  - Function: diag0cnt()
  - Purpose: count 0s on diagonal
Utility — Matrix utility functions

Selection

select()  
select() select rows or columns
st_select() select rows or columns of view
selectindex() select indices

Missing values

missing()  
missing() count of missing values
rowmissing() count of missing values, by row
colmissing() count of missing values, by column
nonmissing() count of nonmissing values
rownonmissing() count of nonmissing values, by row
colnonmissing() count of nonmissing values, by column
hasmissing() whether matrix has missing values

missingof()  
missingof() appropriate missing value

Range, sums, & cross products

minmax()  
rowmin() minimum, by row
colmin() minimum, by column
min() minimum, overall
rowmax() maximum, by row
colmax() maximum, by column
max() maximum, overall
rowminmax() minimum and maximum, by row
colminmax() minimum and maximum, by column
minmax() minimum and maximum, overall
rowmaxabs() rowmax(abs())
colmaxabs() colmax(abs())

minindex()  
minindex() indices of minimums
maxindex() indices of maximums

sum()  
rowsum() sum of each row
colsum() sum of each column
sum() overall sum
quadrowsum() quad-precision sum of each row
quadcolsum() quad-precision sum of each column
quadsum() quad-precision overall sum
### Range, sums, & cross products, continued

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>runningsum()</code></td>
<td>running sum of vector</td>
</tr>
<tr>
<td><code>quadrunningsum()</code></td>
<td>quad-precision running sum of vector</td>
</tr>
<tr>
<td><code>cross()</code></td>
<td>$X'X$, $X'Z$, etc.</td>
</tr>
<tr>
<td><code>crossdev()</code></td>
<td>$(X: -x)'(X: -x)$, $(X: -x)'(Z: -z)$, etc.</td>
</tr>
<tr>
<td><code>quadcross()</code></td>
<td>quad-precision cross()</td>
</tr>
<tr>
<td><code>quadcrossdev()</code></td>
<td>quad-precision crossdev()</td>
</tr>
</tbody>
</table>

### Programming

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>reldif()</code></td>
<td>relative difference</td>
</tr>
<tr>
<td><code>mreldif()</code></td>
<td>max. relative difference between matrices</td>
</tr>
<tr>
<td><code>mreldifsym()</code></td>
<td>max. relative difference from symmetry</td>
</tr>
<tr>
<td><code>mreldifre()</code></td>
<td>max. relative difference from real</td>
</tr>
<tr>
<td><code>all()</code></td>
<td>sum(!$L$)==0</td>
</tr>
<tr>
<td><code>any()</code></td>
<td>sum($L$)!=0</td>
</tr>
<tr>
<td><code>allof()</code></td>
<td>all($P$:==$s$)</td>
</tr>
<tr>
<td><code>anyof()</code></td>
<td>any($P$:==$s$)</td>
</tr>
<tr>
<td><code>panelsetup()</code></td>
<td>initialize panel-data processing</td>
</tr>
<tr>
<td><code>panelstats()</code></td>
<td>summary statistics on panels</td>
</tr>
<tr>
<td><code>panelsubmatrix()</code></td>
<td>obtain matrix for panel $i$</td>
</tr>
<tr>
<td><code>panelsubview()</code></td>
<td>obtain view matrix for panel $i$</td>
</tr>
<tr>
<td><code>_negate()</code></td>
<td>fast negation of matrix</td>
</tr>
</tbody>
</table>

### Constants & tolerances

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mindouble()</code></td>
<td>minimum nonmissing value</td>
</tr>
<tr>
<td><code>maxdouble()</code></td>
<td>maximum nonmissing value</td>
</tr>
<tr>
<td><code>smallestdouble()</code></td>
<td>smallest $\epsilon &gt; 0$</td>
</tr>
<tr>
<td><code>epsilon()</code></td>
<td>unit roundoff error</td>
</tr>
<tr>
<td><code>floatround()</code></td>
<td>round to float precision</td>
</tr>
<tr>
<td><code>solve_tol()</code></td>
<td>tolerance used by solvers and inverters</td>
</tr>
</tbody>
</table>
**Description**

Matrix utility functions tell you something about the matrix, such as the number of rows or whether it is diagonal.

**Remarks and examples**

There is a thin line between utility and manipulation; also see

[M-4] **Manipulation** Matrix manipulation functions

**Also see**

[M-4] **Intro** Categorical guide to Mata functions