Strings and vectorized code

Given strings `a='hello'` and `b='jello'` …

- No. of characters that are the same in `a` and `b`: `sum(a==b)`
- No. of characters that are different in `a` and `b`: `sum(a~=b)`

Vectorized (logical) operations – 1d

```
a= [4 2 6 1 3];
b= [5 3 6 5 3];
```

Relational ops:

```
   L= a==b
   M= a>b
```

Arithmetic ops:

```
   c= a-b
```

Extraction:

```
   d= a(a==b)
   e= b(b>3 & rem(b,2)==1)
```

Vectorized (logical) operations – 1d

```
a= [4 2 6 1 3];

b= [5 3 6 5 3];
```

Relational ops:

```
L= a==b      [0 0 1 0 1]          } Type: Logical

M= a>b       [0 0 0 0 0]
```

Arithmetic ops:

```
c= a-b       [-1 -1 0 -4 0]      Type: double
```

Extraction:

```
d= a(a==b)   [6 3]

e= b(b>3 & rem(b,2)==1)  [5 5]
```

odd

↑ for vectorized code

4

Vectorized (logical) operations – 2d

```
m= [ 2 3   5 7; …
     -2 1   0 7; …
      5 2 -1 8]


L= m>3


P= m>3 | m<0


a= m(m>4)


b= (m>4).*m
```

Vectorized (logical) operations – 2d

```
m= [ 2 3   5 7; …
    -2 1   0 7; …
     5 2  -1 8]
```

L= m>3

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ & 0 & & 1 \end{bmatrix}$$
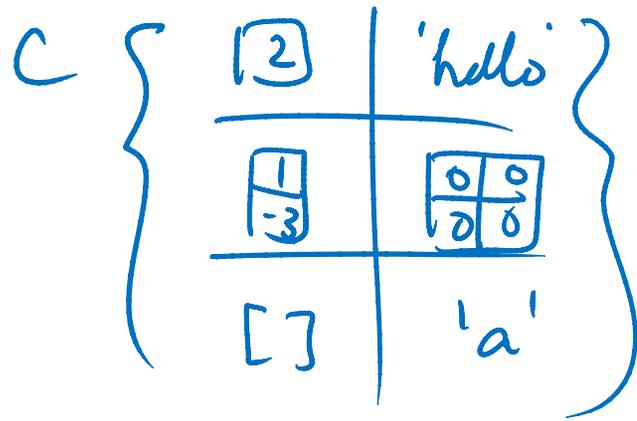
← vectorized code

P= m>3 | m<0

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

a= m(m>4)

$$\begin{bmatrix} 5 \\ 5 \\ 7 \\ 7 \\ 8 \end{bmatrix}$$

b= (m>4).*m

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} .* \begin{bmatrix} 2 & 3 & 5 & 7 \\ -2 & 1 & 0 & 7 \\ 5 & 2 & -1 & 8 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 5 & 7 \\ 0 & 0 & & 7 \\ 5 & 0 & 0 & 8 \end{bmatrix}$$

6

$$C \left\{ \begin{array}{c|c} \boxed{2} & \text{'hello'} \\ \hline \boxed{\begin{array}{c} 1 \\ -3 \end{array}} & \boxed{\begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array}} \\ \hline [\ ] & \text{'a'} \end{array} \right\}$$

C


```
C = cell(3,2);  % unnecessary

C{1,1} = 2;
C{1,2} = 'hello';
C{2,1} = [1; -3];
    ⋮
C{3,1} = [];  % not necessary b/c
              % function cell
              % initializes each cell
              % to be empty vector

    ⋮
```
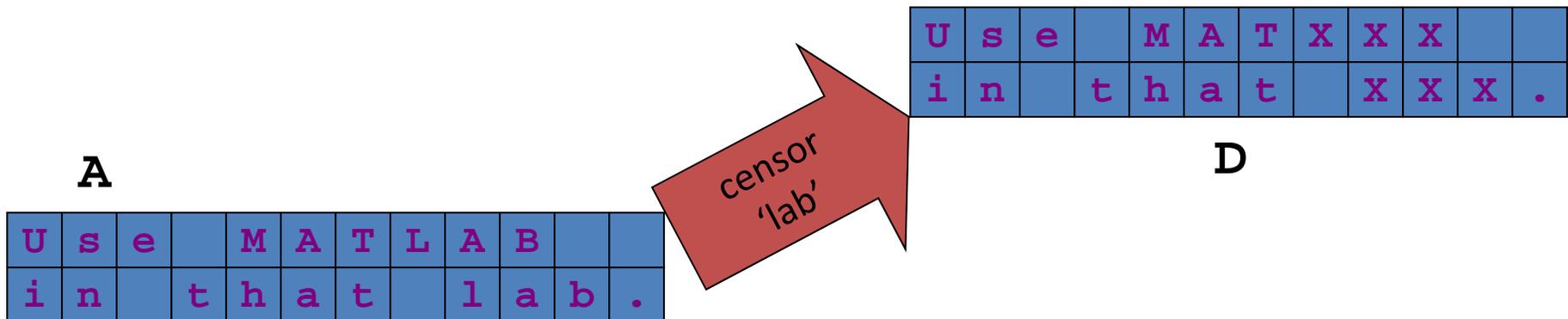
Example: censoring words

```
function D = censor(str, A)
% Replace all occurrences of string str in
% character matrix A with X's, regardless of
% case.
% Assume str is never split across two lines.
% D is A with X's replacing str.
```


A

D

censor 'lab'

```matlab
function D = censor(str, A)
% Replace all occurrences of string str in character matrix A,
% regardless of case, with X's.
% A is a matrix of characters.
% str is a string.  Assume that str is never split across two lines.
% D is A with X's replacing the censored string str.

D= A;
B= lower(A);
s= lower(str);
ns= length(str);
[nr,nc]= size(A);

% Build a string of X's of the right length



% Traverse the matrix to censor string str
```

```
function D = censor(str, A)
% Replace all occurrences of string str in character matrix A,
% regardless of case, with X's.
% A is a matrix of characters.
% str is a string.  Assume that str is never split across two lines.
% D is A with X's replacing the censored string str.

D= A;
B= lower(A);
s= lower(str);
ns= length(str);
[nr,nc]= size(A);

% Build a string of X's of the right length
Xs= char( zeros(1,ns));
for k= 1:ns
    Xs(k)= 'X';
end

% Traverse the matrix to censor string str
```

zeros *returns an array of type* **double**

```matlab
function D = censor(str, A)
% Replace all occurrences of string str in character matrix A,
% regardless of case, with X's.
% A is a matrix of characters.
% str is a string.  Assume that str is never split across two lines.
% D is A with X's replacing the censored string str.

D= A;
B= lower(A);
s= lower(str);
ns= length(str);
[nr,nc]= size(A);

% Build a string of X's of the right length
Xs= char( zeros(1,ns));
for k= 1:ns
    Xs(k)= 'X';
end

% Traverse the matrix to censor string str
for r= 1:nr
    for c= 1:nc-ns+1
        if  strcmp( s , B(r, c:c+ns-1) )==1
            D(r, c:c+ns-1)= Xs;
        end
    end
end
```