Módulo IV. Acceso lógico

- Arreglos lógicos.
- Indexación numérica y lógica.

Indexación con rangos

Extraer secciones de un arreglo:

```
A = 0 : 0.5 : 5;
A(3 : 5) % Los valores de A entre las posiciones 3 y 5
```

```
ans = 1×3
    1.0000    1.5000    2.0000
```

```
A(6 : end) % Desde la posición 6 hasta la última
```

```
ans = 1×6
    2.5000    3.0000    3.5000    4.0000    4.5000    5.0000
```

```
A(2 : 2 : end) % Desde la posición 2 hasta la última saltando cada 2 posiciones
```

```
ans = 1×5
    0.5000    1.5000    2.5000    3.5000    4.5000
```

Unir múltiples vectores en uno (verificar que dimensiones coincidan):

```
S = [0, 0.2, 1, 9, 2]
```

```
S = 1×5
         0    0.2000    1.0000    9.0000    2.0000
```

```
U = [99, 98, 95]
```

```
U = 1×3
    99    98    95
```

```
T = [S, U] % Pueden unirse S y U en vector fila porque ambos son vectores fila
```

```
T = 1×8
         0    0.2000    1.0000    9.0000    2.0000   99.0000   98.0000   95.0000
```

```
R = [14, 9, 150]
```

```
R = 1×3
    14     9   150
```

```
V = [U; R] % U y R pueden ponerse uno encima del otro porque tienen la misma
```

```
V = 2×3
    99    98    95
    14     9   150
```

```
            % cantidad de columnas
```

Extraer secciones de una matriz:

```
M = eye( 7 ); % Matriz identidad
M(:, 5) % Todas las filas de la columna 5
M(3 : 4, :) % Todas las columnas, pero sólo filas 3 y 4
M(:, 2 : end) % Todas las filas pero solo columnas desde la dos hasta la última
```

Indexación lógica

Indicar cuáles datos tomar (1) y cuáles no (0) de algún arreglo. El arreglo de indexación debe ser arreglo lógico (hacer casting con la función logical()):

```
V = [1, 5, 9]
```

```
V = 1×3
     1     5     9
```

```
V( logical([1, 0, 1]) ) % Sólo la primera y la última posición
```

```
ans = 1×2
     1     9
```

```
V( ~logical([1, 0, 1]) ) % Sólo la posición de la mitad
```

```
ans = 5
```

Generar índices lógicos que cumplan con alguna condición:

```
r_1 = A > 0.5 % ¿Cuáles elementos de A son mayores a 0.5?
```

```
r_1 = 1×11 logical array
   0   0   1   1   1   1   1   1   1   1   1
```

```
A( r_1 ) % Extraer esos valores que son mayores que 0.5
```

```
ans = 1×9
    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000    4.5000 ···
```

```
K = magic(8)
```

```
K = 8×8
    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     1
```

```
r_2 = K <= 15 %¿En qué posiciones de K hay elementos menores o iguales a 15?
```

```
r_2 = 8×8 logical array
   0   1   1   0   0   1   1   0
   1   0   0   1   1   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   1   1   0   0   1   1   0
   1   0   0   1   1   0   0   1
```

```
K_2 = K(r_2) %Tome todos esos valores que son menores o iguales a 15
```

```
K_2 = 15×1
     9
     8
     2
    15
     3
    14
    12
     5
    13
     4
     ⋮
```

Extraer los valores de la diagonal de unal matriz:

```
J = randn(20, 20) % Matriz aleatoria
```

```
J = 20×20
    0.2916    0.6003   -0.0438   -0.4256    1.3244    0.0596   -1.2087    2.0500 ···
   -0.7777   -1.3615    0.9608    1.0486   -0.2132   -0.4113   -0.2971    0.1205
    0.5667    0.3476    1.7382    0.6607   -0.1345   -0.3680   -3.2320   -0.9899
   -1.3826   -0.1818   -0.4302    2.5088   -1.1714   -1.3610   -1.0870    1.1978
```

```
    0.2445   -0.9395   -1.6273    1.0635   -1.3853    0.7796   -1.4264   -0.5927
    0.8084   -0.0375    0.1663    1.1569    0.3105    0.4394   -1.0145   -0.4698
    0.2130   -1.8963    0.3763    0.0530   -0.2495   -0.0896   -0.2133    0.8864
    0.8797   -2.1280   -0.2270   -1.2884    0.5037    1.0212   -0.3253   -1.3852
    2.0389   -1.1769   -1.1489   -0.3712   -0.8927   -0.8740    1.9444   -1.9568
    0.9239   -0.9905    2.0243   -0.7578    1.9085    0.4147   -0.5718    0.4207
     ⋮
```

```
I_mat = eye(20); % Usar una matriz identidad del mismo tamaño
I_logic = logical( I_mat ) % Convertirla a matriz de índices lógicos
```

```
I_logic = 20×20 logical array
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
    ⋮
```

```
J_diag = J( I_logic ) % Extraer los valores de J que están  en la diagonal
```

```
J_diag = 20×1
    0.2916
   -1.3615
    1.7382
    2.5088
   -1.3853
    0.4394
   -0.2133
   -1.3852
    0.8017
    0.5411
     ⋮
```