



四、矩阵的运算

1. 调用函数

命令	随机生成的 x				计算结果			
x = randi(5,3,4)	2	3	2	2	7.3891	20.0855	7.3891	7.3891
exp(x)	5	4	2	5	148.4132	54.5982	7.3891	148.4132
% 计算自然常数 e 的指数	5	3	3	1	148.4132	20.0855	20.0855	2.7183
x = rand(4, 3)	0.2259	0.3111	0.9049		0.2300	0.3100	0.9000	
round(x, 2)	0.1707	0.9234	0.9797		0.1700	0.9200	0.9800	
% 四舍五入保留两位小数	0.2277	0.4302	0.4389		0.2300	0.4300	0.4400	
	0.4357	0.1848	0.1111		0.4400	0.1800	0.1100	
x = randi(10,3)	9	10	3		0	1	0	
mod(x, 3) % 计算 x 中各	10	7	6		1	1	0	
元素除以 3 的余数	2	1	10		2	1	1	

四、矩阵的运算



1.调用函数

函数名	函数的作用
sum	求和函数
prod	求乘积函数 (product)
cumsum	计算累积和 (cumulative sum)
diff	计算差分 (difference)
mean	计算平均值
median	计算中位数
mode	计算众数
var	计算方差 (variance)
std	计算标准差 (standard deviation)
min	求最小值 (minimum value)
max	求最大值 (maximum value)



四、矩阵的运算

1.调用函数

sum : 求和函数

(1) 如果 A 是一个向量, 则 `sum(A)` 可以计算 A 中所有元素的和。

<code>A = 1:100;</code> <code>sum(A)</code>	5050
--	------

(2) 如果 A 是一个矩阵, 则 `sum(A,dim)` 可以计算 A 矩阵沿维度 `dim` 中所有元素的和。

- `dim = 1` 表示沿着行方向进行计算, 即计算矩阵每一列的和, 返回一个行向量;
- `dim = 2` 表示沿着列方向进行计算, 即计算矩阵每一行的和, 返回一个列向量。

当 `dim = 1` 时, `sum(A,1)` 可以简写成 `sum(A)`. (sum 函数的帮助文档上有一个示意图)

<code>A = randi(10,3,4)</code>	4	6	6	8
	7	4	3	3
	8	2	1	5
% 计算每一列的和 <code>sum(A)</code> % 或者写成 <code>sum(A,1)</code>	19	12	10	16
% 计算每一行的和 <code>sum(A,2)</code>	24			
	17			
	16			



四、矩阵的运算

1.调用函数

(3) 计算一个矩阵中所有元素的总和。

可以先用一次 `sum` 函数计算矩阵 `A` 每一列的和，返回一个行向量；然后再用一次 `sum` 函数计算这个行向量的和；也可以先使用 `A(:)` 语句把 `A` 中的所有元素按照线性索引的顺序拼接成一个向量，然后直接计算这个向量的和。

A = randi(5,2,3)	<table><tr><td>1</td><td>2</td><td>4</td></tr><tr><td>3</td><td>4</td><td>1</td></tr></table>	1	2	4	3	4	1
1	2	4					
3	4	1					
sum(sum(A))	15						
sum(A(:)) % 更推荐这种写法	15						
补充：从 MATLAB2018b 版本开始，可以使用 sum(A, 'all')来计算所有元素的和。							



四、矩阵的运算

1.调用函数

prod : 求乘积函数 (product)

prod 函数的用法和 sum 函数的用法相同，它是用来计算乘积的，我们直接来看例子。

v = [2,4,5,1,10]; % 向量 prod(v) % 直接向量中所有元素的乘积	400												
v = 1:10; prod(v) % 计算 10 的阶乘 % 熟练的话可以直接写成 prod(1:10)	3628800												
% 下面看矩阵的例子 A = randi(10,3,4)	<table><tr><td>1</td><td>1</td><td>5</td><td>5</td></tr><tr><td>3</td><td>10</td><td>6</td><td>10</td></tr><tr><td>9</td><td>8</td><td>3</td><td>6</td></tr></table>	1	1	5	5	3	10	6	10	9	8	3	6
1	1	5	5										
3	10	6	10										
9	8	3	6										
prod(A) % 计算每列的乘积 % 也可以写成 prod(A, 1)	<table><tr><td>27</td><td>80</td><td>90</td><td>300</td></tr></table>	27	80	90	300								
27	80	90	300										
prod(A,2) % 计算每行的乘积	<table><tr><td>25</td></tr><tr><td>1800</td></tr><tr><td>1296</td></tr></table>	25	1800	1296									
25													
1800													
1296													



四、矩阵的运算

1. 调用函数

<pre>% 如果计算时忽略 NaN 值，可以在最后面 加上'omitnan'参数 prod(v, 'omitnan')</pre>	80
<pre>A = [5 3 -8 4 1 5 NaN 10; 3 6 18 9]; % 矩阵中存在 NaN 值 prod(A) % 计算每一列的乘积</pre>	15 90 NaN 360
<pre>prod(A, 'omitnan') % 忽略 NaN 值</pre>	15 90 -144 360
<pre>% 忽略 NaN 值计算每一行的乘积 prod(A, 2, 'omitnan')</pre>	-480 50 2916



四、矩阵的运算

1.调用函数

cumsum : 计算累积和 (cumulative sum)

(1) 如果 A 是一个向量, 则 cumsum(A)可以计算向量 A 的累积和 (累加值)。

A = [1 5 3 4 -5 0 8]; cumsum(A) % 计算 A 的累积和	1	6	9	13	8	8	16
--	---	---	---	----	---	---	----

(2) 如果 A 是一个矩阵, 则 cumsum(A,dim)可以计算 A 沿维度 dim 中所有元素的累积和, 具体的使用方法和 sum 函数类似。

A = randi(10,3,4)	4	6	6	8
	7	4	3	3
	8	2	1	5
% 计算每一列的累积和 cumsum(A) % 或者写成 cumsum(A,1)	4 11 19	6 10 12	6 9 10	8 11 16
% 计算每一行的累积和 cumsum(A,2)	4 7 8	10 11 10	16 14 11	24 17 16



四、矩阵的运算

1. 调用函数

(3) 也可以在最后加一个输入参数: 'omitnan', 这样计算时会忽略 NaN 值。

<pre>A = [5 3 -8 4 1 5 NaN 10; 3 6 18 9]; cumsum(A) % 计算每一列的累积和</pre>	<pre>5 3 -8 4 6 8 NaN 14 9 14 NaN 23</pre>
<pre>% 忽略 NaN 值计算每一列的累积和 cumsum(A,'omitnan')</pre>	<pre>5 3 -8 4 6 8 -8 14 9 14 10 23</pre>
<pre>cumsum(A, 2) % 计算每一行的累积和</pre>	<pre>5 8 0 4 1 6 NaN NaN 3 9 27 36</pre>
<pre>% 忽略 NaN 值计算每一行的累积和 cumsum(A, 2, 'omitnan')</pre>	<pre>5 8 0 4 1 6 6 16 3 9 27 36</pre>



四、矩阵的运算

1. 调用函数

diff : 计算差分 (difference)

MATLAB 中计算差分的函数是 `diff`，我们可以使用 `diff(A,n)` 命令计算向量 `A` 的 `n` 阶差分，当 `n` 等于 1 时，可以直接写成 `diff(A)`。

<code>w = [60 65 66 70 68 72 64 70];</code> <code>diff(w)</code> % 1 阶差分, <code>diff(w,1)</code>	5	1	4	-2	4	-8	6
<code>diff(w,2)</code> % 2 阶差分	-4	3	-6	6	-12	14	
<code>diff(w,3)</code> % 3 阶差分	7	-9	12	-18	26		



四、矩阵的运算

1.调用函数

diff 函数也可以用在矩阵上面: `diff(A,n,dim)`表示沿矩阵 A 的维度 `dim` 方向上计算差分, 当 `dim=1` 时沿着行方向计算, 即得到每列的 `n` 阶差分; 当 `dim=2` 时沿着列方向计算, 即得到每行的 `n` 阶差分。类似的, `dim=1` 时, `diff(A,n,1)`也可以简写成 `diff(A,n)`。

<code>A = randi(10,3,4)</code>	4	8	7	6
	6	3	9	2
	3	6	10	2
% 计算每列的 1 阶差分 <code>diff(A)</code> % 也可写成 <code>diff(A,1)</code> 或 <code>diff(A,1,1)</code>	2	-5	2	-4
	-3	3	1	0
% 计算每列的 2 阶差分 <code>diff(A,2)</code> % 也可写成 <code>diff(A,2,1)</code>	-5	8	-1	4
% 计算每行的 1 阶差分 <code>diff(A,1,2)</code>	4	-1	-1	
	-3	6	-7	
	3	4	-8	
% 计算每行的 2 阶差分 <code>diff(A,2,2)</code>	-5	0		
	9	-13		
	1	-12		



四、矩阵的运算

1. 调用函数

mean : 计算平均值 (mean/average value)

假设向量 $y = [y_1 \ y_2 \ \cdots \ y_n]$, 即向量 y 中有 n 个元素, 那么它的平均值等于 $\frac{1}{n} \sum_{i=1}^n y_i$.

在 MATLAB 中, mean 函数可以用来计算平均值, 它的使用方法和 sum 函数类似。

(1) 如果 A 是一个向量, 则 mean(A) 可以计算向量 A 的平均值。

```
y = [1 3 5 7 9];
```

```
mean(y) % 计算 y 的平均值
```

```
5
```

```
% (1+3+5+7+9) / 5 = 5
```



四、矩阵的运算

1.调用函数

(2) 如果 A 是一个矩阵, 则 $\text{mean}(A, \text{dim})$ 可以计算 A 沿维度 dim 中所有元素的平均值。

- 当 $\text{dim}=1$ 时沿着行方向进行计算, 即得到每列的平均值;
- 当 $\text{dim}=2$ 时沿着列方向进行计算, 即得到每行的平均值。

类似的, $\text{dim}=1$ 时, $\text{mean}(A,1)$ 也可以简写成 $\text{mean}(A)$.

$A = \text{randi}(10,3,4)$	7	1	2	10
	8	3	9	1
	5	10	6	5
% 求每列的平均值 $\text{mean}(A)$ % 或者写成 $\text{mean}(A,1)$	6.6667	4.6667	5.6667	5.3333
$\text{mean}(A,2)$ % 求每行的平均值	5.0000	5.2500	6.5000	



四、矩阵的运算

1. 调用函数

(3) 也可以在最后加一个输入参数: 'omitnan', 这样计算时会忽略 NaN 值。

<pre>A = [5 3 -8 4 1 5 NaN 10; 3 6 18 9]; mean(A) % 计算每一列的平均值</pre>	<pre>3.0000 4.6667 NaN 7.6667</pre>
<pre>% 忽略 NaN 值计算每一列的平均值 mean(A,'omitnan') % 也可以写成 mean(A,1,'omitnan')</pre>	<pre>3.0000 4.6667 5.0000 7.6667 % 第三列平均数为 5, 由(-8+18)/2 得到, 分母为 2</pre>
<pre>mean(A, 2) % 计算每一行的平均值</pre>	<pre>1 NaN 9</pre>
<pre>% 忽略 NaN 值计算每一行的平均值 mean(A, 2, 'omitnan')</pre>	<pre>1.0000 5.3333 9.0000</pre>



四、矩阵的运算

1.调用函数

median : 计算中位数

在 MATLAB 中，median 函数可以用来计算中位数，它的使用方法和 mean 函数类似。

<code>y = [6 80 1 5 100 20 1000];</code> <code>median(y) % 计算 y 的中位数</code>	20
<code>A = [5 3 -8 4</code> <code>1 5 NaN 10;</code> <code>3 6 18 9];</code> <code>median(A) % 计算每一列的中位数</code>	3 5 NaN 9
<code>% 忽略 NaN 值计算每一列的中位数</code> <code>median(A,'omitnan')</code> <code>% 也可以写成 median(A,1,'omitnan')</code>	3 5 5 9 <code>% 第三列忽略 NaN 值后，只剩下 -8 和 18，中位数为 5</code>
<code>% 忽略 NaN 值计算每一行的中位数</code> <code>median(A, 2, 'omitnan')</code>	3.5000 5.0000 7.5000 <code>% 第二行忽略 NaN 值后，剩下 1、5 和 10，中位数为 5</code>



四、矩阵的运算

1.调用函数

mode : 计算众数

众数是指一组数据中出现次数最多的数。一组数据可以有多个众数，例如向量[1 3 -1 2 1 3]中，1 和 3 都出现了两次，它们都是这组数据中的众数。

MATLAB 中可以使用 mode 函数计算数据的众数，调用方法也和 mean 函数类似，但是 mode 函数可以有多个返回值。

以计算向量 A 的众数为例，直接调用 mode(A)会返回 A 中出现次数最多的值。如果有多个值以相同的次数出现，mode 函数将返回其中最小的值。

```
A = [1 3 -1 2 1 3];  
mode(A) % 计算 A 的众数
```

1



四、矩阵的运算

1. 调用函数

如果 A 是一个矩阵，则 `mode(A,1)` 或者 `mode(A)` 可以沿着行方向进行计算，得到每一列的众数；`mode(A,2)` 可以沿着列方向进行计算，得到每一行的众数，这里的 1 和 2 表示维度 `dim`。

<code>A = randi(10,3,4)</code>	4	6	6	8
	7	2	3	3
	8	2	6	5
<code>% 计算每一列的众数</code> <code>mode(A) % 或者写成 mode(A,1)</code>	4	2	6	3
<code>% 计算每一行的众数</code> <code>mode(A, 2)</code>	6	3	2	



四、矩阵的运算

1.调用函数

var : 计算方差 (variance)

方差是概率论与数理统计里面的知识点，我们先简单回顾一下相关的内容。

先来看个例子：第一组数据是 6、8、10、12、14；第二组数据是-10、0、10、20、30。显然两组数据的均值都是 10，但第二组数据的离散程度更大一些。

方差就是用来描述这种离散程度的一个统计量，当两组数据的平均值相同时，方差较大的一组数据的离散程度更大。有一个常举的例子：一个射击队要从两名运动员中选拔一名参加比赛，选拔赛上两人各打了 10 发子弹，在得分均值相差不大的情况下，应选择方差更小的队员。



四、矩阵的运算

1. 调用函数

在现实生活中，我们收集到的数据可分为下面两类：

- ◇ 总体数据：所要考察对象的全部个体叫做总体；
 - ◇ 样本数据：从总体中所抽取的一部分个体叫做总体的一个样本。
- 根据收集的数据类型的不同，我们计算方差的公式也有所区别。

- 如果数据 $X: \{X_1, X_2, \dots, X_n\}$ 是总体数据（例如普查结果）

那么总体均值： $E(X) = \frac{\sum_{i=1}^n X_i}{n}$ ，总体方差： $\sigma_X^2 = \frac{\sum_{i=1}^n [X_i - E(X)]^2}{n}$

- 如果数据 $Y: \{Y_1, Y_2, \dots, Y_n\}$ 是样本数据（例如抽样调查的结果）

那么样本均值： $\bar{Y} = \frac{\sum_{i=1}^n Y_i}{n}$ ，样本方差： $S_Y^2 = \frac{\sum_{i=1}^n (Y_i - \bar{Y})^2}{n-1}$

从上方的计算公式可以看出，总体方差和样本方差在计算时的区别在于分母上是否要减 1。



四、矩阵的运算

1.调用函数

MATLAB 中使用 **var** 函数计算方差：

(1) 如果 **A** 是一个向量，那么 **var(A, w)**可以计算 **A** 的方差，当 **w=0** 时，表示计算样本方差，**w=1** 时表示计算总体方差，另外，**var(A, 0)**也可以直接简写为 **var(A)**。

(2) 如果 **A** 是一个矩阵，则 **var(A, w, dim)**可以计算矩阵 **A** 沿维度 **dim** 上的方差。

- **dim = 1** 时表示沿着行方向进行计算，即得到每一列的方差；
- **dim = 2** 时表示沿着列方向进行计算，即得到每一行的方差。

当 **dim** 为 1 时，**var(A, w, 1)**可以简写为 **var(A,w)**；若 **w** 为 0，则可以进一步简写为 **var(A)**，即默认情况下 **MATLAB** 会沿行方向计算得到每一列的样本方差。

(3) 如果数据中存在 **NaN** 值，可以在 **var** 函数的最后加上 **'omitnan'**参数来忽略 **NaN**。

<code>v = [6 8 10 12 14];</code> <code>var(v) % 样本方差，等价于 var(v,0)</code>	10
<code>var(v,1) % 总体方差</code>	8



四、矩阵的运算

1. 调用函数

<pre>% 下面看矩阵的例子 A = randi(10,4,5)</pre>	<pre>9 7 10 10 5 10 1 10 5 10 2 3 2 9 8 10 6 10 2 10</pre>
<pre>var(A) % 每一列的样本方差 % 也可以写成 var(A,0)或者 var(A,0,1)</pre>	<pre>14.9167 7.5833 16.0000 13.6667 5.5833</pre>
<pre>var(A,0,2) % 每一行的样本方差</pre>	<pre>4.7000 16.7000 11.7000 12.8000</pre>
<pre>A = [9 7 10 10 NaN; 10 NaN 10 5 10; 2 3 2 9 8; 10 6 10 2 10]; % 矩阵中存在 NaN 值 var(A, 0, 2) % 计算每一行的样本方差</pre>	<pre>NaN NaN 11.7000 12.8000</pre>
<pre>% 忽略 NaN 值计算每一行的样本方差 var(A, 0, 2, 'omitnan')</pre>	<pre>2.0000 6.2500 11.7000 12.8000</pre>



四、矩阵的运算

1.调用函数

std : 计算标准差 (standard deviation)

标准差是方差的算术平方根,它也是用来反应数据离散程度的一个统计量。那么问题来了,既然有了方差为什么还需要标准差呢?这是因为方差和数据原本的量纲(即单位)是不一致的,对方差的计算公式进行量纲分析容易看出,方差的量纲是原始数据量纲的平方,因此对方差开根号,得到的标准差的量纲和原始数据的量纲一致。

在 MATLAB 中,我们可以使用 std 函数计算样本标准差和总体标准差,它和 var 函数的使用方法完全相同。

v = [6 8 10 12 14]; std(v) % 样本标准差, 等价于 std(v,0)	3.1623
std(v,1) % 总体标准差	2.8284



四、矩阵的运算

1. 调用函数

<pre>% 下面看矩阵的例子 A = randi(10,4,5)</pre>	<pre>9 7 10 10 5 10 1 10 5 10 2 3 2 9 8 10 6 10 2 10</pre>
<pre>std(A) % 每一列的样本标准差 % 也可以写成 std(A,0)或者 std(A,0,1)</pre>	<pre>3.8622 2.7538 4.0000 3.6968 2.3629</pre>
<pre>std(A,0,2) % 每一行的样本标准差</pre>	<pre>2.1679 4.0866 3.4205 3.5777</pre>
<pre>A = [9 7 10 10 NaN; 10 NaN 10 5 10; 2 3 2 9 8; 10 6 10 2 10]; % 矩阵中存在 NaN 值 std(A, 0, 2) % 计算每一行的样本标准差</pre>	<pre>NaN NaN 3.4205 3.5777</pre>
<pre>% 忽略 NaN 值计算每一行的样本标准差 std(A, 0, 2, 'omitnan')</pre>	<pre>1.4142 2.5000 3.4205 3.5777</pre>



四、矩阵的运算

1. 调用函数

min : 求最小值 (minimum value)

min 函数主要有两种用法:

用法一: 求两个矩阵对应位置元素的最小值: $\min(A,B)$ 。

$A = \begin{bmatrix} 2 & 6 & 10 \\ 6 & 7 & 5 \\ 5 & 3 & 7 \\ 2 & 6 & 5 \end{bmatrix};$	$B = \begin{bmatrix} 1 & 1 & 6 \\ 3 & 6 & 9 \\ 9 & 4 & 9 \\ 4 & 8 & 2 \end{bmatrix};$	% min(A,B)的结果: $\begin{bmatrix} 1 & 1 & 6 \\ 3 & 6 & 5 \\ 5 & 3 & 7 \\ 2 & 6 & 2 \end{bmatrix}$
--	---	--

矩阵 A 和矩阵 B 的大小可以不一样, 只要保证矩阵 A 和矩阵 B 具有兼容的大小就能够计算, MATLAB 矩阵运算中支持的兼容模式会在“3.1.2 算术运算”这一小节中详细介绍。

下面再举两个例子: 表中第三列是运行 $\min(A, B)$ 后返回的结果。

$A = \begin{bmatrix} 2 & 6 & 10 \\ 6 & 7 & 5 \\ 5 & 3 & 7 \\ 2 & 6 & 5 \end{bmatrix};$	% B 可以是一个标量 $B = 5;$	$\begin{bmatrix} 2 & 5 & 5 \\ 5 & 5 & 5 \\ 5 & 3 & 5 \\ 2 & 5 & 5 \end{bmatrix}$ % A 中每一个元素和 5 比较大小, 取较小的值。
$A = \begin{bmatrix} 2 & 6 & 10 \\ 6 & 7 & 5 \\ 5 & 3 & 7 \\ 2 & 6 & 5 \end{bmatrix};$	% B 可以是一个具有三个元素的行向量 $B = [4 \ 5 \ 6];$	$\begin{bmatrix} 2 & 5 & 6 \\ 4 & 5 & 5 \\ 4 & 3 & 6 \\ 2 & 5 & 5 \end{bmatrix}$ % A 中第 k 列的元素和 B 中第 k 个元素比较大小 ($k=1,2,3$), 取较小的值。



四、矩阵的运算

1.调用函数

用法二：求向量或者矩阵中的最小值，可以指定沿什么维度计算并返回索引。

具体用法有以下三种：

(1) 如果 A 是向量，则 `min(A)` 返回 A 中的最小值。如果 A 中有复数，则比较的是复数的模长。

<code>A = [5 6 7 3 5 3 10];</code> <code>min(A)</code>	3
<code>B = [4.5 3+4i 6];</code> <code>min(B)</code>	4.5000

(2) 如果 A 是矩阵，则 `min(A, [], 1)` 沿着 A 的行方向求每一列的最小值，也可以简写为 `min(A)`; `min(A, [], 2)` 沿着 A 的列方向求每一行的最小值。这里的 1 和 2 表示矩阵的维度(dim)。

<code>A = randi(10,3,4)</code>	7	9	2	8
	1	5	1	3
	6	5	8	2
<code>min(A, [], 1)</code> % 求每一列的最小值 % 可以简写成 <code>min(A)</code>	1	5	1	2
<code>min(A, [], 2)</code> % 求每一行的最小值	2	1	2	



四、矩阵的运算

1. 调用函数

(3) 在求向量或矩阵的最小值时，min 函数可以有两个返回值：[m, ind] = min(A). 第一个返回值 m 是我们要求的最小值，ind 是最小值在所在维度上的索引。如果最小元素出现多次，则 ind 是最小值第一次出现位置的索引。

<pre>% 向量的例子 A = [5 6 7 3 5 3 10]; [min_A, ind] = min(A)</pre>	<pre>min_A = 3 ind = 4 % A 中有两个 3，但返回了第一个 3 的索引</pre>
<pre>% 矩阵的例子 A = [7 9 2 8 1 5 1 3 6 5 8 2]; [min_A, ind] = min(A, [], 2) % 求每一行的最小值并返回索引</pre>	<pre>min_A = ind = 2 3 1 1 2 4</pre>
<pre>[min_A, ind] = min(A) % 求每一列的最小值并返回索引</pre>	<pre>min_A = 1 5 1 2 ind = 2 2 2 3</pre>



四、矩阵的运算

1. 调用函数

max 函数和 min 函数的用法完全相同，它是用来求最大值的函数，下面我们举几个例子。

$A = \begin{bmatrix} 2 & 6 & 10 \\ 6 & 7 & 5 \\ 5 & 3 & 7 \\ 2 & 6 & 5 \end{bmatrix};$	$B = \begin{bmatrix} 1 & 1 & 6 \\ 3 & 6 & 9 \\ 9 & 4 & 9 \\ 4 & 8 & 2 \end{bmatrix};$	$\% \max(A,B)$ 的返回结果 $\begin{bmatrix} 2 & 6 & 10 \\ 6 & 7 & 9 \\ 9 & 4 & 9 \\ 4 & 8 & 5 \end{bmatrix}$
$A = [5 \ 6 \ 10 \ 7 \ 3 \ 5 \ 3 \ 10];$ $\% \text{ 计算向量 } A \text{ 的最大值并返回其索引}$ $[\max_A, \text{ind}] = \max(A)$	$\max_A = 10$ $\text{ind} = 3$	
$A = \text{randi}(10,3,4)$	$\begin{bmatrix} 6 & 9 & 2 & 8 \\ 5 & 5 & 1 & 3 \\ 6 & 9 & 8 & 2 \end{bmatrix}$	
$\max(A, [], 1) \quad \% \text{ 求每一列的最大值}$ $\% \text{ 可以简写成 } \max(A)$	$\begin{bmatrix} 6 & 9 & 8 & 8 \end{bmatrix}$	
$\max(A, [], 2) \quad \% \text{ 求每一行的最大值}$	$\begin{bmatrix} 9 \\ 5 \\ 9 \end{bmatrix}$	



四、矩阵的运算

1. 调用函数

<code>max(A, 2) % 返回每个元素和 2 相比的较大值</code>	<table><tr><td>6</td><td>9</td><td>2</td><td>8</td></tr><tr><td>5</td><td>5</td><td>2</td><td>3</td></tr><tr><td>6</td><td>9</td><td>8</td><td>2</td></tr></table>	6	9	2	8	5	5	2	3	6	9	8	2
6	9	2	8										
5	5	2	3										
6	9	8	2										
<code>[max_A, ind] = max(A,[],2)</code> <code>% 求每一行的最大值并返回索引</code>	<table><tr><td>max_A =</td><td>ind =</td></tr><tr><td>9</td><td>2</td></tr><tr><td>5</td><td>1</td></tr><tr><td>9</td><td>2</td></tr></table>	max_A =	ind =	9	2	5	1	9	2				
max_A =	ind =												
9	2												
5	1												
9	2												
<code>[max_A, ind] = max(A)</code> <code>% 求每一列的最大值并返回索引</code>	<table><tr><td>max_A =</td><td>6</td><td>9</td><td>8</td><td>8</td></tr><tr><td>ind =</td><td>1</td><td>1</td><td>3</td><td>1</td></tr></table>	max_A =	6	9	8	8	ind =	1	1	3	1		
max_A =	6	9	8	8									
ind =	1	1	3	1									
<code>% 存在缺失值的例子</code> <code>A = [5 6 7 NaN 5 3 10 5];</code> <code>[max_A, ind] = max(A)</code>	<code>max_A = 10</code> <code>ind = 7</code>												



四、矩阵的运算

2. 算术运算

矩阵的加法：
五种算术运算的兼容模式：

情形	示例：计算 A+B		计算结果	解释
两个大小完全相同的输入	A = 6 5 6 2 9 2	B = 7 8 9 8 6 2	ans = 13 13 15 10 15 4	将 A 和 B 对应位置的元素相加
有一个输入是标量（常数）	A = 2 1 3 7 2 4	B = 4	ans = 6 5 7 11 6 8	矩阵的每个元素都加上这个标量
一个输入是矩阵，另一个输入是具有相同行数的列向量	A = 3 6 5 2 6 8	B = 6 5	ans = 9 12 11 7 11 13	把 B 堆叠成完全相同的三列，然后再和 A 相加 相当于 repmat(B,1,3)
一个输入是矩阵，另一个输入是具有相同列数的行向量	A = 3 5 6 6 9 4	B = 3 9 6	ans = 6 14 12 9 18 10	把 B 堆叠成完全相同的两行，然后再和 A 相加 相当于 repmat(B,2,1)
一个输入是列向量，另一个输入是行向量。	A = 2 5	B = 1 8 3	ans = 3 10 5 6 13 8	把 A 堆叠成完全相同的三列，把 B 堆叠成完全相同的两行，然后相加 相当于 repmat(A,1,3) repmat(B,2,1)



四、矩阵的运算

2. 算术运算

矩阵的减法：
五种算术运算的兼容模式：

命令	随机生成的 A	随机生成的 B	计算结果
A = randi(10,2,3) B = randi(10,2,3) A - B	9 2 7 10 10 1	3 10 2 6 10 10	6 -8 5 4 0 -9
A = randi(10,1,3) B = randi(10,2,3) A - B	10 8 10	7 9 7 1 10 8	3 -1 3 9 -2 2
A = randi(10,1,3) B = randi(10,2,1) A - B	10 5 9	2 5	8 3 7 5 0 4



四、矩阵的运算

2. 算术运算

矩阵的乘法:

$A*B$

$A.*B$

命令	随机生成的 A	随机生成的 B	计算结果
A = randi(10,2,3) B = randi(10,3,1) A * B	5 1 4 2 5 9	7 1 2	44 37
A = randi(10,1,3) B = randi(10,3,1) A * B	10 8 6	5 1 7	100
A = randi(10,2,3) B = randi(10,2,3) A .* B	5 9 6 2 3 9	8 9 8 3 1 2	40 81 48 6 3 18
A = randi(10,2,3) B = randi(10,1,3) A .* B	9 8 4 4 8 6	3 9 2	27 72 8 12 72 12



四、矩阵的运算

2. 算术运算

矩阵的除法：
/(右除)和\ (左除)
./ (点除)

命令	随机生成的 A		随机生成的 B		计算结果	
A = randi([0,10],3,2)	0	10	0	6	NaN	1.6667
B = randi([0,10],3,2)	4	7	8	0	0.5000	Inf
A ./ B	2	4	10	10	0.2000	0.4000
A = randi([0,10],3,2)	2	3	无		0.5000	0.3333
1 ./ A	4	5			0.2500	0.2000
% 对 A 的元素求倒数	6	2			0.1667	0.5000

“A ./ B” 表示用A的每个元素除以B的对应元素



四、矩阵的运算

2. 算术运算

矩阵的乘方也有两种用法，分别是“ \wedge ”和“ \wedge ”。

其中，“ \wedge ”表示矩阵的幂运算，例如 A 是一个方阵，那么 $A \wedge 3$ 等价于 $A * A * A$ ；“ \wedge ”表示对矩阵中的每一个元素分别进行乘方计算，例如 $A \wedge 0.5$ 表示对矩阵 A 中的每一个元素开根号，等价于 $\text{sqrt}(A)$ 。

命令	随机生成的 A	计算结果
$A = \text{randi}([0,10],2,2)$ $A \wedge 2$	$\begin{bmatrix} 1 & 4 \\ 3 & 0 \end{bmatrix}$	$\begin{bmatrix} 13 & 4 \\ 3 & 12 \end{bmatrix}$
$A = \text{randi}([0,10],2,3)$ $A \wedge 0.5$	$\begin{bmatrix} 1 & 4 & 8 \\ 2 & 9 & 2 \end{bmatrix}$	$\begin{bmatrix} 1.0000 & 2.0000 & 2.8284 \\ 1.4142 & 3.0000 & 1.4142 \end{bmatrix}$
$A = \text{randi}([0,10],2,3)$ $A \wedge 2$	$\begin{bmatrix} 9 & 2 & 8 \\ 4 & 1 & 6 \end{bmatrix}$	$\begin{bmatrix} 81 & 4 & 64 \\ 16 & 1 & 36 \end{bmatrix}$

四、矩阵的运算

2. 算术运算

特别地，如果 A 是一个可逆的方阵，那么 A^{-1} 可用来计算 A 的逆矩阵 (inverse matrix)。另外，MATLAB 中的 `inv` 函数也可以计算逆矩阵，它们的计算结果相同。

命令	结果
$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 4 & 3 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 4 & 3 \end{bmatrix}$
$B1 = A^{-1}$	$\begin{bmatrix} 1.0000 & 3.0000 & -2.0000 \\ -1.5000 & -3.0000 & 2.5000 \\ 1.0000 & 1.0000 & -1.0000 \end{bmatrix}$
$B2 = \text{inv}(A)$	$\begin{bmatrix} 1.0000 & 3.0000 & -2.0000 \\ -1.5000 & -3.0000 & 2.5000 \\ 1.0000 & 1.0000 & -1.0000 \end{bmatrix}$
$B1 * A$	$\begin{bmatrix} 1.0000 & 0.0000 & 0 \\ -0.0000 & 1.0000 & -0.0000 \\ 0.0000 & 0 & 1.0000 \end{bmatrix}$



四、矩阵的运算

2. 算术运算

最后我们再来介绍**矩阵的转置**运算，矩阵的转置符号为英文的单引号：“’”，它也可以在前面加上点变成“.’”，两者的区别在于对矩阵中复数的处理，使用“’”会在转置的同时将复数变为共轭复数（实部不变虚部反号），使用“.’”则会保持原来的复数。

A	A'的结果	A.'的结果
$A = \begin{bmatrix} 3 & 1 + 3i \\ 2 & 5 \\ 4 - 2i & 6 \end{bmatrix}$	$\begin{bmatrix} 3 & 2 & 4 + 2i \\ 1 - 3i & 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 3 & 2 & 4 - 2i \\ 1 + 3i & 5 & 6 \end{bmatrix}$

当然，通常情况下我们的矩阵中全是实数，那么使用“’”和“.’”的效果相同。



四、矩阵的运算

3. 关系运算

MATLAB 中的关系运算符有以下六个：

==	~=	>	>=	<	<=
等于	不等于	大于	大于等于	小于	小于等于

关系运算符可以用来比较两个数组中元素的关系，如果比较的结果为真，则 MATLAB 会返回**逻辑值 1**；如果结果为假，则会返回**逻辑值 0**。这里的逻辑值 1 和 0 实际上就是其他编程语言（例如 C 语言、Java 等）中的布尔型(bool)变量，布尔型变量的值只有真(true)和假(false)。逻辑值 1 和逻辑值 0 可以简称为逻辑 1 和逻辑 0，对应的英文为 logical 1 和 logical 0。

下面我们举一个例子，大家可以观察 MATLAB 的返回结果：

```
A = 5;  
B = 4;  
A == B  
% 一个等号是给变量赋值，两个等号才是判断  
是否相等的关系运算符
```

```
ans = logical  
      0
```



四、矩阵的运算

3. 关系运算

上一小节中，我们介绍了 MATLAB 支持的算术运算的兼容模式，这些兼容模式在关系运算中也支持，下面举几个例子：

A				B				A==B 的结果			
4	4	1	1	1	4	1	2	0	1	1	0
				4	1	1	4	1	0	1	0
				5	4	4	4	0	1	0	0
2	3	5		3				0	1	0	
				2				1	0	0	
NaN				NaN				0			

注意：（1）上面表格第三列的 0 和 1 实际上是逻辑值 0 和逻辑值 1；（2）NaN(不定值或缺失值)相互之间不相等。



四、矩阵的运算

3. 关系运算

易错点：连续使用关系运算符：有许多初学者喜欢连续使用多个关系运算符，大家可以试试下面四条命令，观察 MATLAB 返回的结果是什么：

(1) $0 == 0 == 0$ (2) $1 == 1 == 1$ (3) $-1 < 0 < 1$ (4) $1 > 0 > -1$

在 MATLAB 中，连续使用多个关系运算符可能会产生意想不到的结果，上面四条命令的返回结果如下：

```
0 == 0 == 0    % 返回 logical 0 (false)
1 == 1 == 1    % 返回 logical 1 (true)
-1 < 0 < 1     % 返回 logical 0 (false)
1 > 0 > -1     % 返回 logical 1 (true)
```

这些不符合预期的结果源于 MATLAB 对多个连续关系运算符的解析方式。以第一条命令为例： $0 == 0 == 0$ 实际上被解析为 $(0 == 0) == 0$ 。由于 $0 == 0$ 为真 (logical 1)，因此最终的表达式变为 $1 == 0$ ，结果为假 (logical 0)；第三条命令也是类似的， $-1 < 0 < 1$ 实际上被解析为 $(-1 < 0) < 1$ 。由于 $-1 < 0$ 为真 (logical 1)，因此最终的表达式变为 $1 < 1$ ，结果为假 (logical 0)。第二条和第四条命令也是类似的解释，大家可以自己尝试。



四、矩阵的运算

4. 逻辑运算

(1) 逻辑运算函数

运算方法	函数名	运算符	运算规则（针对逻辑值）	示例
逻辑与	and	&	都为 1 时返回 1，只要有一个是 0 返回 0	and(1, 1) % 返回 1 and(1, 0) % 返回 0 and(0, 0) % 返回 0 1 & 1 % 返回 1 1 & 0 % 返回 0 0 & 0 % 返回 0
逻辑或	or	 	只要有一个为 1 返回 1，都是 0 时返回 0	or(1, 1) % 返回 1 or(1, 0) % 返回 1 or(0, 0) % 返回 0 1 1 % 返回 1 1 0 % 返回 1 0 0 % 返回 0
逻辑非	not	~	原来为 1 时返回 0，为 0 时返回 1	not(1) % 返回 0 not(0) % 返回 1 ~1 % 返回 0 ~0 % 返回 1
逻辑异或	xor	无	不相同取 1，相同时取 0	xor(1, 0) xor(0, 1) % 返回 1 xor(1, 1) xor(0, 0) % 返回 0



四、矩阵的运算

4. 逻辑运算

(1) 逻辑运算函数

下面我们再来介绍 MATLAB 中另外两个使用频率很高的逻辑运算符：**&&**和**||**。

这两个运算符和“逻辑与&”和“逻辑或|”作用相同，但它们有两个非常重要的区别：

(1) **&&**和**||**只能对**标量**（只有一个元素）进行逻辑运算，不能对有多个元素的向量或者矩阵进行运算，而**&**和**|**可以。比如我们上面那个练习题，你只能使用**&**和**|**进行运算。

(2) **&&**和**||**进行逻辑运算时具有**短路功能**，可以提高运行效率：

- 计算 $A \&\& B$ 时，如果 A 为逻辑 0，则 B 不会被判断，因为最后的结果一定是逻辑 0；
- 计算 $A \|\ B$ 时，如果 A 为逻辑 1，则 B 不会被判断，因为最后的结果一定是逻辑 1。

举个例子：假设 a 等于 10， b 等于 3，现在要计算： $(a+b < 10) \&\& (a/b > 1)$ ，那么 MATLAB 首先会判断前面一项： $(a+b < 10)$ ，因为这一项计算的结果为逻辑 0，所以后面的 $(a/b > 1)$ 这一项不会被计算，MATLAB 会直接返回逻辑 0；如果你使用的是： $(a+b < 10) \& (a/b > 1)$ ，那么这两项都会被计算，这样的话效率会低一点。在下一章中，我们会介绍 if 判断语句和 while 循环语句，和**&**、**|**相比，**&&**和**||**在 if 和 while 语句中使用频率更高。



四、矩阵的运算

4. 逻辑运算

(2) 利用逻辑值引用矩阵的元素

提取随机矩阵A中所有小于等于3的元素：

命令	结果												
A = randi(10,3,4)	<table><tr><td>3</td><td>5</td><td>6</td><td>2</td></tr><tr><td>4</td><td>1</td><td>8</td><td>4</td></tr><tr><td>6</td><td>7</td><td>7</td><td>10</td></tr></table>	3	5	6	2	4	1	8	4	6	7	7	10
3	5	6	2										
4	1	8	4										
6	7	7	10										
L = (A <= 3)	<div>3×4 logical 数组</div> <table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	1	0	0	1	0	1	0	0	0	0	0	0
1	0	0	1										
0	1	0	0										
0	0	0	0										
A(L)	<table><tr><td>3</td></tr><tr><td>1</td></tr><tr><td>2</td></tr></table>	3	1	2									
3													
1													
2													



四、矩阵的运算

4. 逻辑运算

(3) 使用逻辑值修改或删除矩阵元素

最后一个例子可以看出，使用逻辑索引删除矩阵中的元素后，MATLAB会将矩阵中剩下的元素按照线性索引的顺序放入到一个行向量中。

依次运行下面的命令	结果
<code>v = randi(10,1,6)</code>	10 5 3 9 5 6
<code>v(v > 6) = 100</code> % 将 v 中大于 6 的元素全部修改成 100	100 5 3 100 5 6
<code>v(v <= 5) = []</code> % 删除 v 中小于等于 5 的元素	100 100 6
<code>A = [1:4; 2:5; 3:6]</code>	1 2 3 4 2 3 4 5 3 4 5 6
<code>A(mod(A,3) == 0) = 10</code> % 将能被 3 整除的元素修改成 10	1 2 10 4 2 10 4 5 10 4 5 10
<code>A(A < 3) = 0</code> % 将小于 3 的元素修改成 0	0 0 10 4 0 10 4 5 10 4 5 10
<code>A(A <= 4) = []</code> % 删除小于等于 4 的元素	10 10 10 5 5 10

四、矩阵的运算



4.逻辑运算

(4) all、any和find函数

函数名	作用
all	判断数组元素是否全为非零值（可指定沿什么维度判断），全为非零值时返回逻辑 1，否则返回逻辑 0
any	判断数组元素中是否存在至少一个非零值（可指定沿什么维度判断），是的话返回逻辑 1，否则返回逻辑 0
find	查找数组中的非零元素，并返回其索引



四、矩阵的运算

4.逻辑运算

(4) all、any和find函数

其中，all 函数和 any 函数的用法类似，以 **all 函数** 为例，它的用法如下：

(1) 如果 A 是一个向量，那么当所有元素均为非零值时，all(A)返回逻辑值 1 (true)，当存在一个或多个元素为零时，返回逻辑值 0 (false)。

(2) 如果 A 是一个矩阵，那么 all(A,dim) 沿着 dim 维来判断元素是否全为非零值，dim 等于 1 时沿着行方向来判断每一列是否全为非零值，并将结果返回为一个全为逻辑值的行向量；dim 等于 2 时表示沿着列方向判断每一行是否全为非零值，并将结果返回为一个全为逻辑值的列向量。特别地，当 dim 等于 1 时，可以直接简写成 all(A)。

(事实上，all 函数和 any 函数的用法和我们之前讲解的 sum 函数非常像)



四、矩阵的运算

4. 逻辑运算

(4) all、any和find函数

A	命令	结果(这里的 0 和 1 都是逻辑值)
[2 4 4 -2 2 -3 -5 3]	all(A)	1
[5 -5 0 -1 0 3 -2 3]	all(A)	0
<pre>[0 2 -1 2; -3 0 1 -2; -2 -2 -2 3]</pre>	all(A) % 等价于 all(A,1)	0 0 1 1
	all(A,2)	0 0 1
	all(A(:)) % 等价于 all(all(A)) % 判断整个矩阵的元素是否全 都为非零值	0



四、矩阵的运算

4. 逻辑运算

(4) all、any和find函数

A	命令	结果(这里的 0 和 1 都是逻辑值)
[5 -5 0 -1 0 3 -2 3]	any(A)	1
[0 0 0 0 0 0]	any(A)	0
<pre>[0 0 -1 2; -3 0 1 -2; 0 0 0 0]</pre>	any(A) % 等价于 any(A,1)	1 0 1 1
	any(A,2)	1 1 0
	any(A(:)) %等价于 any(any(A)) % 判断矩阵 A 的所有元素中是 % 否存在至少一个非零值	1

四、矩阵的运算



5.集合运算

函数名	功能
<code>unique</code>	返回数组中的唯一值
<code>ismember</code>	判断一个数组的元素是否在另一个数组内
<code>intersect</code>	返回两个数组的交集
<code>union</code>	返回两个数组的并集
<code>setdiff</code>	返回两个数组的差集
<code>setxor</code>	返回两个数组的对称差集



四、矩阵的运算

4.逻辑运算

(1) unique 函数

unique 函数可用来提取数组中的唯一值，它可以用于我们学过的向量和矩阵上，也可以用于我们后续章节要学的表格类型的数据上。

先以向量为例，unique 函数的用法如下：

如果 A 是一个向量， $C = \text{unique}(A)$ 会对向量 A 进行去重操作，即提取向量 A 的唯一值。返回的向量 C 和输入的向量 A 的方向相同，向量 C 的每一个元素都来自向量 A 且互不相同，同时，MATLAB 会自动将 C 中元素升序排列。（显然， $\text{numel}(A) \geq \text{numel}(C)$ 对任意的 A 都成立）

命令	结果
$A1 = [5 \ -6 \ 5 \ 10 \ 8 \ -6 \ -3 \ -3];$ $C1 = \text{unique}(A1)$	$C1 =$ -6 -3 5 8 10
$A2 = [1:5, 3:-1:1];$ $C2 = \text{unique}(A2)$	$C2 =$ 1 2 3 4 5
$A3 = [5;1;3;1;3;5;3];$ $C3 = \text{unique}(A3)$	$C3 =$ 1 3 5



四、矩阵的运算

4.逻辑运算

另外，unique 函数可以有最多三个返回值： $[C, ia, ic] = \text{unique}(A)$ ，这里的 ia 和 ic 都是索引向量， ia 是 C 中的每个元素在 A 中的索引值， ic 是 A 中的每个元素在 C 中的索引值。因此有下面的等式成立： $A(ia)$ 等于 C 且 $C(ic)$ 等于 A 。

命令	结果		
$A = [5 \ -6 \ 5 \ 10 \ 8 \ -6 \ -3 \ -3];$ $[C, ia, ic] = \text{unique}(A)$	$C =$ -6 -3 5 8 10	$ia =$ 2 7 1 5 4	$ic =$ 3 1 3 5 4 1 2 2

$C = [-6 \ -3 \ 5 \ 8 \ 10]$ 是 A 中的唯一值；

$ia = [2; 7; 1; 5; 4]$ 是一个列向量：第一个元素 2 代表的含义是 C 中第一个元素 -6 在 A 中的索引值是 2（ A 中有两个 -6，MATLAB 会返回第一个 -6 的索引）；第二个元素是 7，表示 C 中第二个元素 -3 在 A 中的索引值是 7；第三个元素是 1，表示 C 中第三个元素 5 在 A 中的索引值是 1；依此类推。

$ic = [3; 1; 3; 5; 4; 1; 2; 2]$ 也是一个列向量：第一个元素 3 代表的含义是 A 中第一个元素 5 在 C 中的索引值是 3；第二个元素 1 代表的含义是 A 中第二个元素 -6 在 C 中的索引值是 1；依此类推。



四、矩阵的运算

4.逻辑运算

上面得到的唯一值向量 C 都会自动进行升序排列，如果我们不希望 MATLAB 自动排序，可以在 `unique` 函数的输入最后增加一个参数 '`stable`'，这样 MATLAB 会按照与 A 中相同的顺序返回 C 中的值。

命令	结果
<pre>A = [5 -6 5 10 8 -6 -3 -3]; C = unique(A, 'stable')</pre>	<pre>C = 5 -6 10 8 -3</pre>



四、矩阵的运算

4.逻辑运算

`unique` 函数还可以作用到矩阵上，它的用法如下：

如果 A 是一个矩阵，那么 `unique(A)` 的结果和 `unique(A(:))` 的结果相同。但是，如果我们加一个输入参数 `'rows'`，那么 `unique(A, 'rows')` 会将 A 的每一行视为一个整体，会返回 A 矩阵的唯一行。注意，MATLAB 默认会对唯一行进行排序，排序规则如下：优先按照第一列元素升序排列，第一列元素相同时，会按第二列元素升序排列，依此类推。当然，如果你希望按照与 A 中相同的顺序返回唯一值，则可以在输入的最后加一个参数 `'stable'`。

A	unique(A)	unique(A, 'rows')	unique(A, 'rows', 'stable')
3 5	1		
6 8	2	2 5	3 5
3 5	3	2 8	6 8
4 1	4	3 5	4 1
2 8	5	4 1	2 8
2 5	6	6 8	2 5
4 1	8		

`unique` 函数用于矩阵上时，也可以有最多三个返回值，但这种情况我们用的非常少，感兴趣的同学可以在 MATLAB 中进行测试。



四、矩阵的运算

4. 逻辑运算

(2) ismember 函数

$h = \text{ismember}(A, B)$ 可以判断数组 A 中的元素是否在数组 B 中，返回值 h 是一个和 A 大小相同的逻辑数组，逻辑数组 h 中的元素为逻辑值 1 时说明该位置的 A 元素在 B 中存在，为逻辑值 0 时说明在 B 中不存在。

命令	结果
<pre>A = [4 1 3 4 8]; B = [3 7 4 5 4 9 6]; h = ismember(A,B)</pre>	<pre>1×5 logical 数组 1 0 1 1 0</pre>
<pre>A = [3, 1; 8, 5; 4, 0]; B = [3 7 4 5 4 9 6]; h = ismember(A,B)</pre>	<pre>3×2 logical 数组 1 0 0 1 1 0</pre>
<pre>A = [3,1; 8,5; 4,0]; B = [3 7 4 5; 1 4 9 6]; h = ismember(A,B)</pre>	<pre>3×2 logical 数组 1 1 0 1 1 0</pre>



四、矩阵的运算

4. 逻辑运算

`ismember` 函数可以有两个返回值: `[h, ib] = ismember(A,B)`。 `h` 就是上面的那个逻辑数组。 `ib` 是和 `A` 大小相同的一个数组, 对于 `A` 中属于 `B` 的成员的每一个值, `ib` 会包含该值在 `B` 中的最小索引; 如果值为 0 表示 `A` 不是 `B` 的成员。

命令	结果
<pre>A = [4 1 3 4 8]; B = [3 7 4 5 4 9 6]; [h, ib] = ismember(A,B)</pre>	<pre>h = 1×5 logical 数组 1 0 1 1 0 ib = 3 0 1 3 0</pre>
<pre>A = [3, 1; 8, 5; 4, 0]; B = [3 7 4 5 4 9 6]; [h, ib] = ismember(A,B)</pre>	<pre>h = 3×2 logical 数组 1 0 0 1 1 0 ib = 1 0 0 4 3 0</pre>
<pre>A = [3,1; 8,5; 4,0]; B = [3 7 4 5; 1 4 9 6]; [h, ib] = ismember(A,B)</pre>	<pre>h = 3×2 logical 数组 1 1 0 1 1 0 ib = 1 2 0 7 4 0 % 注意 ib 中的索引是 B 矩阵的线性索引</pre>



四、矩阵的运算

4. 逻辑运算

如果 B 和 A 的列数相同,那么我们可以在 `ismember` 函数的输入最后增加一个参数 'rows', 这时候 `ismember(A, B, 'rows')` 会将 A 的每一行视为一个整体, 然后在 B 中查找。

命令	结果
<pre>A = [6 8; 3 5; 4 1]; B = [1 2 3 5 3 8 6 1]; [h, ib] = ismember(A, B)</pre>	<pre>h = 3×2 logical 数组 1 1 1 1 0 1 ib = 4 7 2 6 0 1</pre>
<pre>A = [6 8; 3 5; 4 1]; B = [1 2 3 5 3 8 6 1]; [h, ib] = ismember(A, B, 'rows')</pre>	<pre>h = 3×1 logical 数组 0 1 0 ib = 0 2 0</pre>

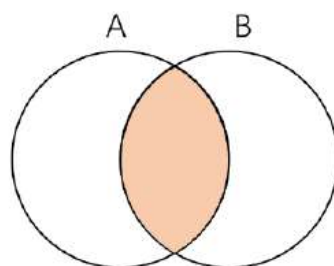
四、矩阵的运算



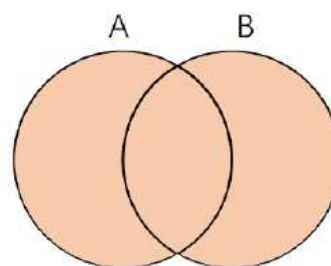
4.逻辑运算

(3) intersect、union、setdiff 和 setxor 函数

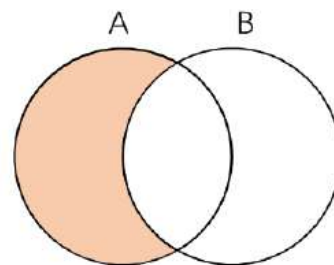
这四个函数分别用于计算两个数组之间的交集、并集、差集和对称差集，下面给出了这四个函数对应的维恩图 (Venn diagram)。



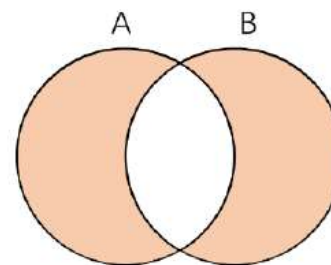
`intersect(A,B)` 返回 A 和 B 的交集，但是不包含重复项。



`union(A,B)` 返回 A 和 B 的并集，但是不包含重复项。



`setdiff(A,B)` 返回A中存在但B中不存在的数据，不包含重复项。



`setxor(A,B)` 返回 A和B的对称差集，不包含重复项。



四、矩阵的运算

4. 逻辑运算

因为这四个函数的用法类似，所以下面以 `intersect` 函数为例，我们介绍它的用法。

`C = intersect(A, B)` 会返回数组 `A` 和 `B` 的共同数据，但是不包含重复项，返回的 `C` 默认会排序。我们还可以增加一个输入参数 `'stable'`，这样会按照在 `A` 中出现的顺序返回 `C` 中的值。

命令	结果
<pre>A = [5 3 1 4 2 7 2]; B = [2 8 6 1 0 3 9 5]; C = intersect(A, B) % 交集</pre>	<pre>C = 1 2 3 5</pre>
<pre>% A 和 B 同上 C = intersect(A, B, 'stable')</pre>	<pre>C = 5 3 1 2</pre>
<pre>A = [1 2 4; 3 4 8]; B = [1 3; 8 7; 1 2]; C = intersect(A, B) % 交集</pre>	<pre>C = 1 2 3 8</pre>
<pre>% A 和 B 同上 C = intersect(A, B, 'stable')</pre>	<pre>C = 1 3 2 8</pre>



四、矩阵的运算

4.逻辑运算

如果 A 和 B 的列数相同，那么我们可以加一个输入参数 'rows'，这时候 intersect (A, B, 'rows') 会将 A 和 B 的每一行视为一个整体，然后返回 A 和 B 共同的行。注意，MATLAB 默认会对共同的行进行排序，排序规则如下：优先按照第一列元素升序排列，第一列元素相同时，会按第二列元素的升序排列，依此类推。当然，如果你希望按照与 A 中相同的顺序返回唯一值，则可以在输入的最后加一个参数 'stable'。

命令	结果
<pre>A = [3 3 3; 0 0 1; 1 0 3; 1 1 1]; B = [1 0 3; 3 3 3; 0 0 0]; C = intersect(A, B, 'rows') % 交集</pre>	<pre>C = 1 0 3 3 3 3</pre>
<pre>% A 和 B 同上 C = intersect(A, B, 'rows', 'stable')</pre>	<pre>C = 3 3 3 1 0 3</pre>



四、矩阵的运算

4.逻辑运算

命令	结果
A = [5 3 1 7 2]; B = [2 8 6 1 5]; C = union(A, B) % 并集	% A 和 B 元素的并集, 不包含重复值, 且会自动排序 C = 1 2 3 5 6 7 8
% A 和 B 同上 C = union(A, B, 'stable') % 并集	% 按照出现的先后顺序返回结果 C = 5 3 1 7 2 8 6
A = [6 3; 5 1; 3 7]; B = [4 2; 5 1]; C = union(A, B, 'rows') % 行的并集	% A 和 B 中行的并集 C = 3 7 4 2 5 1 6 3
A = [5 3 1 7 2]; B = [2 8 6 1 5]; D1 = setdiff(A, B) % 差集	% A 中有 B 中没有的元素 D1 = 3 7
% A 和 B 同上 D2 = setdiff(B, A) % 差集	% B 中有 A 中没有的元素 D2 = 6 8
% A 和 B 同上 E = setxor(A, B) % 对称差集	% 出现在 A 或 B 中, 但不是同时出现的元素 E = 3 6 7 8

五、线性代数相关的函数



函数名	功能
<code>det</code>	计算方阵的行列式 (determinant)
<code>rank</code>	计算矩阵的秩
<code>trace</code>	计算方阵的迹 (对角线元素的和)
<code>rref</code>	将矩阵变换成行最简型矩阵 (Reduced row echelon form 简化行阶梯形)
<code>inv</code>	计算方阵的逆矩阵 (inverse matrix), <code>inv(X)</code> 的结果和 X^{-1} 相同
<code>transpose</code>	返回转置矩阵 (有复数的话转置后虚部符号保持不变), <code>transpose(A)</code> 和 <code>A.'</code> 的结果相同
<code>triu</code>	返回矩阵的上三角部分, <code>triangle(三角形) + upper(高、上方)</code>
<code>tril</code>	返回矩阵的下三角部分, <code>triangle(三角形) + lower(低、下方)</code>
<code>eig</code>	计算方阵的特征值和特征向量 (eigenvalue and eigenvector)
<code>norm</code>	计算向量或者矩阵的范数 (这个概念大家可能没接触过, 它在机器学习中用的较多)

五、线性代数相关的函数



A	代码	结果
$\begin{bmatrix} 6 & 2 & 1 \\ 10 & 10 & 8 \\ 4 & 3 & 3 \end{bmatrix}$	det(A)	30
$\begin{bmatrix} 6 & 2 & 1 \\ 10 & 10 & 8 \end{bmatrix}$	det(A)	错误使用 det 矩阵必须为方阵。
$\begin{bmatrix} 9 & 8 & 8 & 5 \\ 7 & 4 & 2 & 6 \\ 7 & 4 & 2 & 6 \end{bmatrix}$	rank(A)	2
$\begin{bmatrix} 4 & 6 & 5 \\ 1 & 3 & 3 \\ 6 & 7 & 6 \end{bmatrix}$	trace(A)	13
$\begin{bmatrix} 2 & 4 & 4 & 1 \\ 6 & 10 & 8 & 7 \\ 1 & 8 & 9 & 1 \end{bmatrix}$	rref(A)	$\begin{bmatrix} 1 & 0 & 0 & -0.5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -2.5 \end{bmatrix}$
$\begin{bmatrix} 2 & 3 & 4 \\ 3 & 2 & 5 \\ 3 & 7 & 8 \end{bmatrix}$	inv(A)	$\begin{bmatrix} 3.8 & -0.8 & -1.4 \\ 1.8 & -0.8 & -0.4 \\ -3 & 1 & 1 \end{bmatrix}$
$\begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 1 & 4 \end{bmatrix}$	transpose(A)	$\begin{bmatrix} 1 & 3 & 1 \\ 2 & 8 & 4 \end{bmatrix}$
$\begin{bmatrix} 3 & 1+3i \\ 2 & 5 \\ 4-2i & 6 \end{bmatrix}$	transpose(A)	$\begin{bmatrix} 3 & 2 & 4-2i \\ 1+3i & 5 & 6 \end{bmatrix}$

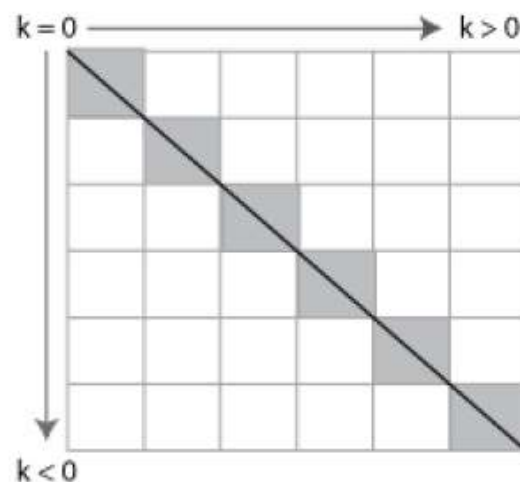
五、线性代数相关的函数



triu 函数和 **tril** 函数用法相同，可分别用来返回矩阵的上三角部分和下三角部分。

- **triu**(A,k) 返回 A 的第 k 条对角线上以及该对角线上方的元素，其他位置元素用 0 填充，k 等于 0 时可以简写成 **triu**(A)。
- **tril**(A,k) 返回 A 的第 k 条对角线上以及该对角线下方的元素，其他位置元素用 0 填充，k 等于 0 时可以简写成 **tril**(A)。

默认值 $k = 0$ 是主对角线， $k > 0$ 位于主对角线上方，而 $k < 0$ 位于主对角线下方。



五、线性代数相关的函数



A					代码	结果				
9	6	8	6	2	triu(A) 或 triu(A, 0)	9	6	8	6	2
9	1	2	4	0		0	1	2	4	0
7	2	5	1	4		0	0	5	1	4
6	8	6	5	2		0	0	0	5	2
9	5	1	2	1		0	0	0	0	1
同上					triu(A, -1)	9	6	8	6	2
						9	1	2	4	0
						0	2	5	1	4
						0	0	6	5	2
						0	0	0	2	1
同上					tril(A)	9	0	0	0	0
						9	1	0	0	0
						7	2	5	0	0
						6	8	6	5	0
						9	5	1	2	1
同上					tril(A, 1)	9	6	0	0	0
						9	1	2	0	0
						7	2	5	1	0
						6	8	6	5	2
						9	5	1	2	1

五、线性代数相关的函数



天津大学
Tianjin University

下面我们用这两个函数来做一个有趣的练习题：生成一个 n 阶（例如 $n=4$ ）的对称矩阵，里面的每个元素都是位于区间 $[0, 9]$ 中的随机整数。

五、线性代数相关的函数



答案如下：

```
n = 4;  
num = n*(n-1)/2;  
A = zeros(n);  
A(triu(true(n),1)) = randi([0,9],num,1);  
A = A+A'+diag(randi([0,9],n,1))
```

% 随机生成的对称矩阵：

7	6	0	9
6	3	8	6
0	8	6	7
9	6	7	1

这几行代码综合性非常强，核心的思路就是将这个对称矩阵分成三个部分：首先随机生成对称矩阵的上半部分（不包括主对角线），这是一个上三角矩阵；然后将其转置来确保矩阵的元素是对称的；最后使用 `diag` 函数生成一个随机的对角矩阵。将上半部分、下半部分以及对角矩阵相加即可得到这个对称矩阵。



五、线性代数相关的函数

eig 函数 用来计算方阵的特征值和特征向量，它有两种最基础的使用法：

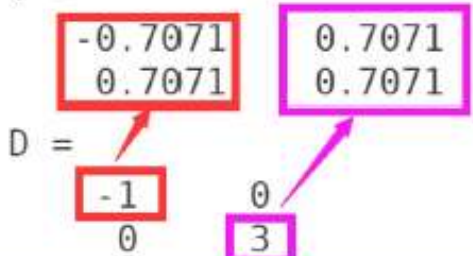
- $e = \text{eig}(A)$ 返回一个列向量， e 中包含方阵 A 的所有特征值。
- $[V, D] = \text{eig}(A)$ 返回特征向量构成的矩阵 V 和特征值构成的对角矩阵 D ， V 中的每一列就是 D 中对应特征值的特征向量。

举个最简单的例子：学过线性代数的同学应该会求矩阵 $A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ 的特征值和特征向量，

这是手算的结果： A 的两个特征值分别为 -1 和 3 ，其中 -1 对应的特征向量为 $k_1 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ ， 3 对应的

特征向量为 $k_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ，这里的 k_1 和 k_2 均为非零常数。

五、线性代数相关的函数

命令	结果
<pre>A = [1 2; 2 1]; e = eig(A)</pre>	<pre>e = -1 3</pre>
<pre>A = [1 2; 2 1]; [V, D] = eig(A)</pre>	<pre>V = -0.7071 0.7071 0.7071 0.7071 D = -1 0 0 3</pre> 

五、线性代数相关的函数



命令	结果
<pre>A = [17 24 1 8 15; 23 5 7 14 16; 4 6 13 20 22; 10 12 19 21 3; 11 18 25 2 9]; [V,D] = eig(A)</pre>	<pre>V = -0.4472 0.0976 -0.6330 0.6780 -0.2619 -0.4472 0.3525 0.5895 0.3223 -0.1732 -0.4472 0.5501 -0.3915 -0.5501 0.3915 -0.4472 -0.3223 0.1732 -0.3525 -0.5895 -0.4472 -0.6780 0.2619 -0.0976 0.6330 D = 65.0000 0 0 0 0 0 -21.2768 0 0 0 0 0 -13.1263 0 0 0 0 0 21.2768 0 0 0 0 0 13.1263</pre>

练习题：请你将上方A矩阵的特征值从大到小降序排列，对应的特征向量的顺序也要跟着特征值的顺序改变。

五、线性代数相关的函数



答案如下:

```
[V, D] = eig(A)
e = diag(D); % 取出特征值
[sort_e, ind] = sort(e, 'descend'); % 将特征值降序排列
D_new = diag(sort_e)
V_new = V(:, ind)
```

计算结果如下:

```
D_new = diag(sort_e)
```

```
D_new = 5x5
    65.0000         0         0         0         0
         0    21.2768         0         0         0
         0         0    13.1263         0         0
         0         0         0   -13.1263         0
         0         0         0         0   -21.2768
```

```
V_new = V(:, ind)
```

```
V_new = 5x5
   -0.4472    0.6780   -0.2619   -0.6330    0.0976
   -0.4472    0.3223   -0.1732    0.5895    0.3525
   -0.4472   -0.5501    0.3915   -0.3915    0.5501
   -0.4472   -0.3525   -0.5895    0.1732   -0.3223
   -0.4472   -0.0976    0.6330    0.2619   -0.6780
```