目录

讲解视频：可以在 bilibili 搜索“MATLAB 教程新手入门篇——数学建模清风主讲”。

https://www.bilibili.com/video/BV1dN4y1Q7Kt/

配套的讲义和代码下载方式：

微信公众号《数学建模学习交流》后台发送 701365 六个数字

第 3 章：MATLAB 矩阵的操作

3.1 复习矩阵的基础知识

MATLAB 是 matrix 和 laboratory 两个词的组合，意为矩阵实验室，MATLAB 中有大量内置函数都是基于矩阵进行计算的。在数学建模中，也有许多模型需要借助矩阵的知识，例如在层次分析法和主成分分析法中，都需要计算矩阵的特征值。因此，要想学好 MATLAB，必须要学和矩阵相关的操作。

本节将帮大家简单复习《线性代数》中与矩阵相关的基础知识点。《线性代数》是大学基础的数学课之一，除了文科专业不开设外，其他专业一般都需要学习，没有学过的低年级同学可以在 b 站搜索相关的视频提前自学。

【矩阵的定义】由$m\times n$个元素组成的m行n列的数表$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$称为一个$m\times n$阶矩阵，记为$A=(a_{ij})_{m\times n}$，当$m=n$时，我们称矩阵$A$为$n$阶方阵（或$n$阶矩阵）。

【同型矩阵】当矩阵A，B的行数和列数都相同时，我们称矩阵A,B为同型矩阵。

【转置矩阵】设$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$，$\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$称为矩阵$A$的转置矩阵，记为$A^T$或$A'$.

【向量的定义】由n个数构成的数表$[a_1,a_2,\cdots,a_n]$或$[a_1,a_2,\cdots,a_n]^T$称为n维行向量或n维列向量。显然，向量是矩阵的特例，行向量的行数为 1，列向量的列数为 1.

【向量的模】设向量$\alpha=[a_1,a_2,\cdots,a_n]^T$,称$\sqrt{{a_1}^2+{a_2}^2+\cdots+{a_n}^2}$为向量$\alpha$的模，记为$|\alpha|$.

【矩阵的加减法】设矩阵$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$，$B=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$，则$A\pm B=$

$\begin{bmatrix} a_{11}\pm b_{11} & a_{12}\pm b_{12} & \cdots & a_{1n}\pm b_{1n} \\ a_{21}\pm b_{21} & a_{22}\pm b_{22} & \cdots & a_{2n}\pm b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\pm b_{m1} & a_{m2}\pm b_{m2} & \cdots & a_{mn}\pm b_{mn} \end{bmatrix}$。显然，只有两个矩阵同型时才能相加减。

【数与矩阵的乘法】设$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$，则$kA=\begin{bmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{bmatrix}$.

【矩阵与矩阵的乘法】设矩阵$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$，$B=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1s} \\ b_{21} & b_{22} & \cdots & b_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{ns} \end{bmatrix}$，则$AB=$

$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{ms} \end{bmatrix}$，其中，$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\cdots+a_{in}b_{nj}(i=1,2,\cdots,m;j=1,2\cdots,s)$.显然，两个矩阵的乘法必须满足左边矩阵的列数与右边矩阵的行数相等。

【单位矩阵】称$E=\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$为单位矩阵。（部分教材用符号$I$表示单位矩阵。单位矩阵一定为方阵）

【矩阵的逆】设A是n阶矩阵，若存在n阶矩阵B，使得$BA=E$或$AB=E$，则称矩阵A可逆，矩阵B称为矩阵A的逆矩阵，记为$B=A^{-1}$.

【特征值和特征向量】设n阶方阵A满足以下条件：存在数λ（λ可为复数）和非零n维列向量x，使得$Ax=\lambda x$成立，则称数λ是方阵A的特征值，称x为方阵A对应于特征值λ的特征向量。注意，特征向量并不唯一，$kx(k\neq 0)$也是方阵A对应于特征值λ的特征向量。

3.2 MATLAB 中的向量

上一节我们知道，向量可以视为矩阵的一个特例：若一个矩阵的行数为 1，则它可以视为一个行向量；若列数为 1，则可以视为列向量；如果行数和列数同时为 1，那么它就是一个标量（又称为常量、常数）。

学过其他编程语言的同学应该听过数组这个概念，在程序设计中，为了处理方便，我们需要把具有相同类型的若干元素按有序的形式组织起来，这些有序排列的同类型数据元素的集合就被称为数组。

在 MATLAB 的官方文档中，有时候也会出现数组这个概念。在 MATLAB 里，向量可以被称为一维数组，而我们常见到的由多行多列构成的矩阵可以被称为二维数组，这两个维度由矩阵的行和列表示。在 MATLAB 中，一维数组可以视为二维数组的一个特例。另外，在 MATLAB 中，我们还可以定义多维数组，多维数组是指具有两个以上维度的数组，绝大多数情况下我们不会用到，有兴趣的同学可以查看官方帮助文档。

以后在不引起误会的情况下，我们将 MATLAB 中的矩阵和数组视为同一个概念。

3.2.1 向量的创建方法

在 MATLAB 中，向量的创建方法主要有三种，分别是：直接输入法、冒号法和利用 MATLAB 的函数创建。大家可以打开本节的配套代码：“code_3_2_1”进行学习，下面我们来进行介绍：

（1）直接输入法

向量元素需要用**英文的中括号“[ ]”**括起来，元素之间用空格、逗号、分号或按回车键分隔，就可以创建对应的向量。**若元素之间用空格、逗号分隔，则创建的是行向量；若用分号、回车键分隔，则创建的是列向量。**（注意：这里的逗号和分号都是英文输入法下输入的，不能用中文的逗号或分号）

举例：a = [1 3 5]和 a = [1,3,5]都可以创建包含元素 1,3,5 的行向量，并将这个行向量的值赋值给 a；而 b = [1;3;5]创建的是包含元素 1,3,5 的列向量。

（2）冒号法：最常用

我们可以利用命令：`A:step:B` 来创建一个行向量。（冒号也要是英文的！）

其中，A 是起始值，step 是每次递增或递减的步长，B 是终止值（不一定刚好停在这里）。**若 step 等于 1，则可以直接简写成 A:B。**

直接看上面的概念不够直观，下面我们举几个例子，大家根据例子来理解会很轻松。

代码	结果和相应的解释
1:2:7	[1 3 5 7]　　% 每次增加 2，直到最后到了 7
1:2:8	[1 3 5 7]　% 每次增加 2，到了 7 后，如果再增加 2 的话结果等于 9，比 8 要大，所以到了 7 就停止了。
1:2:9	[1 3 5 7 9]
2:3:18	[2 5 8 11 14 17]
0:0.1:1	[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]　% 每次增加 0.1
0:0.01:1	[0 0.01 0.02 0.03 …… 0.98 0.99 1]　　% 每次增加 0.01
1:1:100 或简写成 1:100	[1 2 3 4 5 6 7 8 …… 98 99 100]　% 步长为 1 时可以省略
1:10:3	1　% 从 1 开始，增加 10 等于 11，比 3 还要大，所以返回 1
5:2:1	空的 1×0 double 行向量　　% 若 A > B 且步长 step > 0，则会返回空的向量。
10:-1:6	[10 9 8 7 6]　% 步长为-1，因此会从 10 开始递减
10:-2:5	[10 8 6]　% 步长为-2，从 10 开始递减，到了 6 后，如果再减去 2 就等于 4，比 5 还要小，所以到了 6 就停止了。
10:-100:5	10　% 步长为-100,因为 10-100 = -90 比 5 还要小，所以返回 10
10:-10:50	空的 1×0 double 行向量　% 若 A < B 且步长 step < 0，则会返回空的向量。
1:0:2	空的 1×0 double 行向量　% 若 step = 0，则返回空的向量。

大家根据上面的例子应该很容易发现规律。下面再给大家补充两个知识点：

第一：上表中，有三种情况都会导致 MATLAB 返回空的向量：空的 1×0 double 行向量。怎么理解这个返回结果呢？这个“1×0”指的是向量的维度，你可以理解为 1 行 0 列，即这个向量是空的，不存在元素。**在 MATLAB 中，我们可以直接使用命令[ ]创建空的向量。**

第二：MATLAB 返回空的向量时，出现了一个英文单词：double。这里的 double 表示双精度浮点型，我在这门课中并没有特意去介绍数值的类型，原因是这一块的知识比较底层，涉及到数值在计算机中的存储方式。同时，和 C、C++、JAVA 等语言相比，MATLAB 是偏应用的一门语言，其对数值类型的要求较弱。对数值类型感兴趣的同学可以先在 b 站学习 C 语言的公开课，然后再从 MATLAB 官网搜索数值类型的帮助文档进行学习。

（3）利用 MATLAB 函数创建

我们主要介绍两个函数: **linspace** 和 **logspace**，它们分别用来创建等差数列和等比数列。

首先介绍 **linspace 函数**，它有两种用法，区别在于是否给定第三个输入参数 n，如果我们不指定 n，则 MATLAB 会默认 n=100。这个函数使用的频率也很高，大家需要掌握。

- `linspace(a,b)`：该命令用来创建一个行向量，向量中的第一个元素为 a，最后一个元素为 b，形成总数为 100 个元素的线性间隔的向量。
- `linspace (a,b,n)`：该命令用来创建一个行向量，向量中的第一个元素为 a，最后一个元素为 b，形成总数为 n 个元素的线性间隔的向量。

代码	结果和相应的解释
linspace(1,100,10)	[1 12 23 34 45 56 67 78 89 100]　% 第一个数为 1，最后一个数为 100，整个向量构成了一个等差数列，由 10 个元素组成
linspace(1,99,10)	[1 11.8889 22.7778 33.6667 44.5556 55.4444 66.3333 77.2222 88.1111 99] % 第一个数为 1，最后一个数为 99，整个向量由 10 个元素组成，构成了一个等差数列，MATLAB 会自动计算等差数列的步长。
linspace(0,2*pi,10)	[0 0.69813 1.3963 2.0944 2.7925 3.4907 4.1888 4.8869 5.5851 6.2832] % 第一个数为 0，最后一个数为 2*pi，注意中间的乘号千万不能省略！
linspace(1,10)	[1 1.0909 1.1818 1.2727 1.3636　……　9.8182 9.9091 10] % 如果不指定第三个输入参数 n，则默认生成 100 个元素的等差数列
linspace(100,1,10)	[100 89 78 67 56 45 34 23 12 1]　% 如果 a>b,则步长是负数

初学者可能搞不懂 **linspace (a,b,n)和冒号法 a:step:b 生成向量的区别**，我这里为大家总结：

（1） linspace 不需要指定步长，MATLAB 会根据你给定的元素个数 n 自动计算出来；而使用冒号法可以自己指定步长。

（2） linspace 生成的向量的最后一个元素一定是 b，而使用冒号法 a:step:b 生成的向量的最后元素不一定是 b。

（3） 后续章节讲解循环语句时，冒号法使用的频率最高；而在绘制函数图形时，使用 linspace 得到的 x 轴的范围要比冒号法稍微准确一点。例如：我们要绘制 sin(x)在区间[0, 2π]上的图形，x 的范围是 0 到2π，我们使用 linspace(0,2*pi)生成的向量的最后一个元素一定是2π；如果使用冒号法令 x=0:0.1:2*pi，那么 x 向量的最后一个元素和2π有一个微小的差异，当然，如果我们将 step 取得更小，例如取成 0.01，那么这个差异几乎可以忽略。

另一个函数是 **logspace 函数**，它使用的频率不高，大家了解即可。它有两种常见的用法：

- `logspace(a,b)`：创建一个行向量，其第一个元素为10^a，最后一个元素为10^b，形成总数为 50 个元素的等比数列向量。

- logspace(a, b, n)：创建一个行向量，其第一个元素为10^a，最后一个元素为10^b，形成总数为 n 个元素的等比数列向量。

下面我们来看几个例子：

代码	结果和相应的解释
logspace(1,2,10)	[10 12.915 16.681 21.544 27.826 35.938 46.416 59.948 77.426 100] % 第一个数为 10^1，最后一个数为 10^2， 10 个元素组成的等比数列
logspace(2,1,5)	[100 56.234 31.623 17.783 10] % 第一个数为 10^2，最后一个数为 10^1，5 个元素组成的等比数列
logspace(log10(2),log10(1024),10)	[2 4 8 16 32 64 128 256 512 1024] % 第一个数为 2，最后一个数为 1024，10 个元素组成的等比数列

大家可以思考：如何创建一个包含 n 个元素的等比数列，其第一项为 a，最后一项为 b。（这里 n、a 和 b 都是正数）？这个问题留作本章的课后习题。

3.2.2 向量元素的引用

对向量元素的引用（即提取向量指定位置的值）有两种情形，分别是提取向量中的单个元素和提取向量中的多个元素。在正式讲解之前，我们先来介绍索引（或下标）的概念。

我们知道，向量分为行向量和列向量，它们在 MATLAB 中只有一个维度，因此我们可以利用向量中包含的元素个数来描述一个向量的大小。**在 MATLAB 中，可以使用 length 函数或 numel 函数来计算向量中包含的元素个数。**

例如：a = [1,3,8,9,7]; length(a)或 numel(a)的返回结果是 5，因为向量 a 中有五个元素。

假如我们有一个行向量 a，里面包含了 n 个元素（n 是大于等于 1 的常数），它们分别是$a_1, a_2, \dots, a_n$，那么我们可以列一个表格：

向量的元素	a_1	a_2	a_3	$\cdots$	a_{n-1}	a_n
索引（下标）	1	2	3	$\cdots$	$n-1$	n

从上表可以看出，索引就是指某一个元素在向量中对应的位置，也可以称为元素在向量中所处的下标，在 MATLAB 中，向量的索引是从 1 开始的。

举个具体的例子，假设向量 a=[2 4 8 16 32 64 128 256 512 1024]，那么 a 中有 10 个元素，因此 a 的最大索引是 10。

（1）单个元素引用

我们提取向量 a 中单个元素的方法很简单，只需要利用 a(ind)命令，小括号中的 ind 就是你要提取的对应元素的索引。（注意：创建向量用中括号，提取元素要用小括号哦！）

例如：a(1)的结果为 2，因为 a 中第 1 个位置（索引或下标等于 1）的元素是 2；类似的，a(9)等于 512，因为 a 中第 9 个位置的元素是 512。

有些同学可能会好奇，如果我取索引为 11，即输入 a(11)会出现什么情况？

MATLAB 会报错：“索引超出数组元素的数目(10)”，即告诉我们，现在这个向量中元素的数目只有 10 个，即最大索引是 10，而你取了索引 11 的元素，超出了取值范围。

另外，如果我们将 ind 取成 0、负数或者小数，例如输入 a(0)、a(-1)、a(1.5)，MATLAB 也会报错：“数组索引必须为正整数或逻辑值”。这里出现了“逻辑值”的概念，我们在本章后面小节中会介绍。

（2）多个元素引用

类似的，我们也可以利用向量的索引来同时提取多个位置的元素，这时候只需要将 ind 设置成一个向量，ind 中放入我们想要提取的元素的索引，然后使用 a(ind)命令即可。

例如，我们令 ind = [1 3 5 7 9]，那么 a(ind)的结果为[2 8 32 128 512]，即我们提取了向量 a 中奇数位置的元素。熟悉向量冒号创建方法的同学应该能够看出，ind 等于 1:2:9，因此我们可以直接将 a(ind)写成 a(1:2:9)，这就表示提取 a 中奇数位置的元素；类似的，提取 a 中偶数位置元素的命令是 a(2:2:10)，如果你不熟练的话，可以分成两步写，即先令 ind=2:2:10，然后再使用 a(ind)的命令。当然，对于同一个位置的元素，我们也可以提取多次，例如：ind = [1 2 2 3 3 3]，那么 a(ind)得到的结果应该是[2 4 4 8 8 8]，以后熟悉的话可以直接写成 a([1 2 2 3 3 3])。

技巧：使用 end 索引

有同学会想，假如我不知道向量 a 中有多少个元素，也不想使用 length 函数或者 numel 函数来计算向量中元素的个数，那我能不能提取出 a 中奇数位置的元素呢？这时候就需要用到一个特殊的关键字：end。它有很多种用法，在这里 **end 可以用来替代向量的最后一个索引**。

例如，我现在要访问 a 中第五个至最后一个元素，那么我们可以直接使用 a(5:end)，这里的 end 就表示了 a 的最后一个索引；另外，我们还可以对 end 进行计算，例如要访问 a 中第五个至倒数第三个元素，我们可以使用 a(5:end-2)，得到的结果为[32 64 128 256]。

这里有一个易错点，如果使用了 end，不能将要取元素的索引赋值给 ind。例如，还是要访问 a 中第五个至最后一个元素，如果你令 ind=5:end，MATLAB 就会报错。因此，我们只能在 a 后面的小括号中使用 end 来替代数组的最后一个索引。

那我们回到上面的问题，在不知道 a 中有多少个元素的前提下，我们可以使用 a(1:2:end)。

最后请大家思考：如何将一个向量倒序？例如原来的向量是[1 5 8 4]，倒序后是[4 8 5 1]。这个问题留作本章课后习题。

3.2.3 向量元素的修改和删除

前面我们介绍了向量元素的引用，我们可以利用等号赋值的方法对引用位置的元素进行修改和删除。令向量 a=[2 4 8 16 32 64 128 256 512 1024]，请大家依次执行下面的代码：

请依次执行下面的代码	修改后的向量 a
a(1) = 4　% 第一个元素改成 4	[4 4 8 16 32 64 128 256 512 1024]
a([1,3]) = [50 60] % 第 1 个位置元素改成 50；第 3 个位置元素改成 60	[50 4 60 16 32 64 128 256 512 1024]
a(1:3) = [5 6] % 赋值时，左右两侧的元素个数要相同，左边引用了 3 个位置，右侧的向量长度为 2	MATLAB 报错：无法执行赋值，因为左侧和右侧的元素数目不同。
a(2:4) = 100　% 如果右边为常数，则将指定位置的元素全部变成这个常数。	% 第 2 至 4 号位置的元素改为了 100 [50 100 100 100 32 64 128 256 512 1024]
a(13) = 88　% 把索引为 13 的元素赋值为 88，如果超过了最大索引，则会自动拓展向量的大小	[2 4 8 16 32 64 128 256 512 1024 0 0 88] % 索引 11 和 12 的位置会自动用 0 进行赋值

如果我们将等号右侧变成空向量[ ]，则表示删除对应位置的元素。

a(1) = [ ]　% 删除 a 的第一个元素	[100 100 100 32 64 128 256 512 1024]
a(end-1:end) = [ ]　% 删除 a 中最后两个元素	[100 100 100 32 64 128 256]

3.3 MATLAB 中的矩阵

上一节中介绍了 MATLAB 中向量的基本操作，本节介绍 MATLAB 中矩阵的相关知识。因为向量可以看成矩阵的一个特例，所以它们有许多类似的操作。

3.3.1 矩阵的创建方法

在 MATLAB 中，矩阵的创建方法主要有三种，分别是：直接输入法、函数创建法和导入本地文件中的数据。大家可以打开本节的配套代码：“code_3_3_1”进行学习。

（1）直接输入法

我们先来看直接输入法，直接输入法适用于矩阵中元素数量较少的情况。

输入矩阵时要以中括号“[ ]”作为标识符号，矩阵的所有元素必须都在中括号内。矩阵的同行元素之间用空格或逗号分隔，行与行之间用分号或回车键分隔。

例如：命令 *a = [1 2 3; 4 5 6];* 可以在工作区创建出变量名为 a 的矩阵 $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$。

（2）函数创建法

MATLAB 提供了一些函数，这些函数可以用来生成某些特定的矩阵，我们这里介绍几组最常用到的函数。

第一组函数： zeros、ones 和 eye。这三个函数分别用来创建全为 0 的矩阵、全为 1 的矩阵和单位矩阵。

以 zeros 函数为例，其常见的用法有两种：(1) zeros(n)可以创建一个 n 行 n 列全为 0 的矩阵；(2) zeros(m,n)可以创建一个 m 行 n 列全为 0 的矩阵。

例如：

命令	结果
a = zeros(3)	0 0 0 0 0 0 0 0 0
b = zeros(2,3)	0 0 0 0 0 0

ones 和 eye 的用法类似，大家可以看配套的讲解视频或者查看 MATLAB 的帮助文档。

第二组函数： rand、randi 和 randn。这三个函数分别用来创建均匀分布的随机数、均匀分布的随机整数和标准正态分布的随机数，以后会大量用到，请大家熟记。（数据的分布是概率论里面的知识点，没学过的同学可以搜索关键词自学）

rand 函数用来创建区间 0 和 1 内均匀分布的随机数，其最常用的方法有两种：(1) rand(n)可以创建一个 n 行 n 列的随机数矩阵；(2) rand(m,n)可以创建一个 m 行 n 列的随机数矩阵。由 rand 函数创建的随机数矩阵的每个元素都随机取样自 0 和 1 之间的均匀分布。

randi 函数用来创建均匀分布的随机整数，其最一般的用法为：randi([imin, imax],m,n)，该命令可创建一个 m 行 n 列的随机数矩阵，随机数矩阵中的每个元素都是从区间[imin, imax]内随机抽取的整数。举个例子，假设我们要模拟投掷 100 次骰子，骰子有 6 个面，那么我们可以使用 randi([1,6],1,100)得到一个长度为 100 的行向量，向量中的每个元素都是取自 1,2,3,4,5,6 中的一个整数。另外，如果 imin 等于 1，那么可以简写为 randi(imax,m,n)；如果 m 和 n 相同，即生成一个 n 行 n 列的方阵，那么可以直接写成 randi([imin, imax],n)。

randn 函数用来创建标准正态分布的随机数，其使用方法和 rand 函数类似：(1) randn(n) 可以创建一个 n 行 n 列的随机数矩阵；(2) randn(m, n) 可以创建一个 m 行 n 列的随机数矩阵。由 randn 函数创建的随机数矩阵的每个元素都随机取样自标准正态分布。

注意：因为我们生成的是随机数，所以每次运行的结果可能会变化。除了上述这几个函数外，MATLAB 还提供了其他一些与随机数生成相关的函数，感兴趣的同学可以在 MATLAB 官网搜索关键词：随机数。

第三组函数：diag 和 blkdiag。

diag 函数可用来创建对角矩阵或者获取矩阵的对角元素

情况 1：如果输入的第一个参数是向量，则表示创建对角矩阵。

diag(v, k) 将向量 v 的元素放置在第 k 条对角线上，其他位置元素为 0。

k=0 表示主对角线，k>0 位于主对角线上方，k<0 位于主对角线下方。

如果 k=0, 可以直接写成 diag(v)。

命令	结果
diag([1,2,3])	1 0 0 0 2 0 0 0 3
diag([1,2,3], -1)	0 0 0 0 1 0 0 0 0 2 0 0 0 0 3 0
diag([1,2,3], 2)	0 0 1 0 0 0 0 0 2 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0

情况 2：如果输入的第一个参数是矩阵，则表示获取矩阵的对角元素。

diag(A,k) 返回 A 的第 k 条对角线上元素的构成的列向量。

命令	结果
A = [1,2,3,4; 5,6,7,8; 9,10,11,12; 13,14,15,16]	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
diag(A)	1 6 11 16
diag(A, -1)	5 10 15
diag(A, 1)	2 7 12

blkdiag 函数可用来创建分块对角矩阵。

分块对角矩阵是相对于常规的对角矩阵而言的，常规的对角矩阵沿对角线具有单个元素，而分块对角矩阵的对角线的元素是矩阵。我们可采用以下形式表示一个分块对角矩阵：

$$\begin{bmatrix} \mathrm{A1} & O & \cdots & O \\ O & \mathrm{A2} & \cdots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \cdots & \mathrm{AN} \end{bmatrix} (\mathrm{A1},\mathrm{A2},\cdots,\mathrm{AN}\text{各自可以是大小不同的矩阵})$$

命令	结果
A1 = [1,2,3;4,5,6]	1 2 3 4 5 6
A2 = [7,8;9,10]	7 8 9 10
A3 = [11,12;13,14;15,16]	11 12 13 14 15 16
blkdiag(A1,A2,A3)	1 2 3 0 0 0 0 4 5 6 0 0 0 0 0 0 0 7 8 0 0 0 0 0 9 10 0 0 0 0 0 0 0 11 12 0 0 0 0 0 13 14 0 0 0 0 0 15 16

（3）导入本地文件中的数据

MATLAB 可读取本地的文件，支持的常见格式如下：

- .txt、.dat 或 .csv（适用于带分隔符的文本文件）
- .xls、.xlsb、.xlsm、.xlsx、.xltm、.xltx 或 .ods（适用于电子表格文件）

由于这一块的内容比较丰富且可能涉及我们没学过的知识点，所以会放在后面的章节进行讲解。到时候我们会重点学习 MATLAB 菜单栏：“主页——导入数据”这个功能。

3.3.2 矩阵元素的引用

在讲解矩阵元素的引用之前，我们先来回顾一下矩阵的表示方式：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}_{m\times n}$$

上方给出了一个m行n列的矩阵，对于第i行第j列的元素，我们用a_{ij}表示。

因此，我们可以使用矩阵元素所处的行(row)和列(column)来进行引用矩阵的某一个元素，方式为：a(row_ind, column_ind).

这里的 row_ind 表示要引用的元素的行索引，column_ind 表示列索引。如果 row_ind 和 column_ind 都是一个常数，则表示提取矩阵中的单个元素；如果 row_ind 或 column_ind 是包含有多个元素的向量，则表示同时提取多个位置的元素。与向量类似，end 也可以用来替代最后一个索引，通常和冒号法一起使用。

下面我们来举两个例子，大家也可以打开本节的配套代码学习：“code_3_3_2”。

命令	结果
a = randi([2,10],4,5) % 生成一个随机整数矩阵	7 9 3 6 5 5 3 4 4 2 5 8 6 6 8 7 10 10 7 6
a(1,2) % 第一行第二列的元素	9
a(2,end) % 第二行最后一列的元素	2
a(2,[1 3]) % 第二行、第一三列的元素	5 4
a(3,1:5) % 第三行、一至五列的元素	5 8 6 6 8
a([1 3], [1 3 5]) % 第一三行、第一三五列的元素	7 3 5 5 6 8
a(1:2:end, 1:2:end) % 奇数行且奇数列对应的元素	7 3 5 5 6 8

前面我们学过，可以使用 length 函数和 numel 函数来计算向量中包含的元素个数。那么，怎样计算一个矩阵的大小呢？我们可以使用 **size 函数**，它有两种常见的用法：

（1）size(A)返回一个行向量，其元素是 A 的各维度的长度。若 A 是一个 3×4 的矩阵，则 size(A)返回向量[3 4]；如果让[r,c] = size(A)，那么 r = 3,c = 4。

（2）size(A,dim)返回在维度 dim 上的长度。dim = 1 时表示行；dim = 2 时表示列。若 A 是一个 3×4 的矩阵，则 size(A,1)返回 3，size(A,2)返回 4。

（length 函数和 numel 函数也可以用在矩阵上。length 函数会返回行和列的较大值：对上面的 A 矩阵，length(A)返回 4；numel 函数会返回矩阵中元素的总数，numel(A) 返回 12）

有时候我们需要取出矩阵的某一行或者某一列。以取出矩阵 A 的第一行为例，我们可以使用代码 A(1, 1:end)，即 row_ind 取 1 表示第一行，column_ind 取 1:end 表示从 1 到最后一列的索引。这时候我们可以直接将其简写为：A(1,:)，逗号后面是列索引的位置，加一个冒号就表示取出每一列的元素。同理，要取第一列的所有元素，我们可以使用代码：A(:, 1).

总结：

- **A(:, n) 表示矩阵 A 的第 n 列的所有元素。**
- **A(m, :) 表示矩阵 A 的第 m 行的所有元素。**

练习：

（1）生成一个 5 行 7 列的随机整数矩阵，各元素在区间[1,10]上均匀分布

```
A = randi(10,5,7)
```

（2）取出 A 的第 2, 5 两列的元素

```
A(:,[2,5])
```

（3）取出 A 的最后一列的元素

```
A(:,end)
```

（4）取出 A 的偶数行的元素

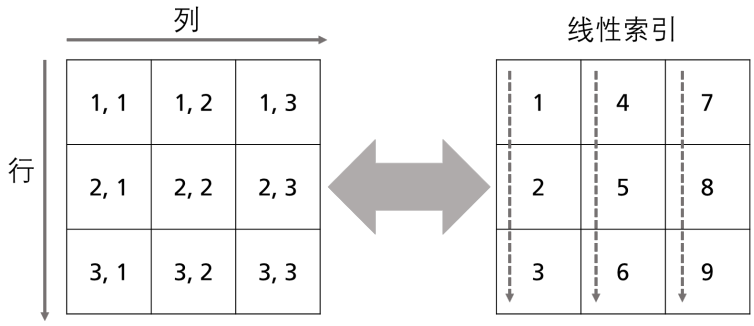
```
A(2:2:end, :)
```

（5）取出 A 的奇数行、3 的倍数列的元素

```
A(1:2:end, 3:3:end)
```

前面我们介绍的是对矩阵的双下标进行索引，即同时指定行索引和列索引，中间用逗号隔开。有同学会问，我们能不能使用单个下标对矩阵进行索引呢？在 MATLAB 中是可以的，这种**单下标的索引方法称为线性索引**。

事实上，在 MATLAB 中，矩阵的数据在计算机的内存中被存储为单列。以下图为例，下面的矩阵虽然显示为 3×3 矩阵，但 MATLAB 在内存中将它存储为单列，由它的各列顺次连接而成。例如，第 2 行第 2 列的元素的线性索引为 5，第 2 行第 3 列的元素的线性索引为 8。



我们可以利用线性索引来取出矩阵中的元素，尽管这种方式并不那么直观。

命令	结果
A = randi(100, 3, 5)	39 24 40 78 32 7 21 6 17 33 36 82 38 92 21
A(5)	21
A(2: 7)	7 36 24 21 82 40
A(1: 2: end)	39 36 21 40 38 17 32 21

另外，**A(:)命令可以将 A 中的所有元素按照线性索引的顺序重构成一个列向量。**

命令	结果
A = randi(100, 2, 3)	59 31 24 21 48 85
A(:)	59 21 31 48 24 85
% 对向量也有同样的效果 a = [2 5 8 1]; a(:)	2 5 8 1

最后，sub2ind 和 ind2sub 函数可用于在矩阵的原始索引(双下标)和线性索引之间进行转换。他们的功能刚好相反，sub2ind 将矩阵的下标转换为线性索引；ind2sub 将线性索引转换为下标。

（1） ind = sub2ind(sz,row,col) 针对大小为 sz 的矩阵返回由 row 和 col 指定的行列下标的对应线性索引 ind。此处，sz 是包含两个元素的向量，其中 sz(1) 指定行数，sz(2) 指定列数。

（2） [row,col] = ind2sub(sz,ind) 返回数组 row 和 col，其中包含与大小为 sz 的矩阵的线性索引 ind 对应的等效行和列下标。此处，sz 是包含两个元素的向量，其中 sz(1) 指定行数，sz(2) 指定列数。

我们举两个例子：

命令	结果
ind = sub2ind([3,3],2,2)	5
ind = sub2ind([3,3],1,3)	7
[row,col] = ind2sub([3,3], 5)	row = 2　col = 2
[row,col] = ind2sub([3,3], 7)	row = 1　col = 3

3.3.3 矩阵元素的修改和删除

我们可以直接利用等号赋值的方法对矩阵中引用位置的元素进行修改，用法和对向量元素的修改类似。

依次运行下面的命令	结果
A = [1:4; 2:5; 3:6]	1 2 3 4 2 3 4 5 3 4 5 6
A(2,3) = 10 % 第二行第三列的元素修改成 10	1 2 3 4 2 3 10 5 3 4 5 6
A(3,:) = 100　% 如果右边为常数，则将指定位置的元素全部变成这个常数	1 2 3 4 2 3 10 5 100 100 100 100
% 左右两侧的大小要匹配 A([1,3],[2,3]) = [8 88; 888 8888]	1 8 88 4 2 3 10 5 100 888 8888 100

当然，你也可以使用线性索引（单下标的索引）的方式对矩阵的元素进行修改：

依次运行下面的命令	结果
A = [1:4;2:5;3:6]	1 2 3 4 2 3 4 5 3 4 5 6
A(4) = 10　% 将线性索引为 4 的元素修改成 10	1 10 3 4 2 3 4 5 3 4 5 6
A(1:2:end) = 0　% 如果右边为常数，则将指定位置的元素全部变成这个常数。	0 10 0 4 2 0 4 0 0 4 0 6
% 左右两侧的元素数量要匹配 A([3,5,6]) = [8 88 888]	0 10 0 4 2 88 4 0 8 888 0 6

注意，如果你在赋值时将一个或多个元素置于矩阵现有的行和列索引的边界之外，则会将矩阵的大小进行拓展，MATLAB 会将没有赋值的位置的元素自动用 0 填充，使其保持为完整的矩形。

例如，A 是一个 2 行 3 列的矩阵，在 A 的第三行第四列的位置插入一个元素 88，矩阵 A 会自动进行拓展。

依次运行下面的命令	结果
A = [10 20 30; 60 70 80]	10 20 30 60 70 80
A(3,4) = 88	10 20 30 0 60 70 80 0 0 0 0 88

此外，我们还可以通过在现有索引范围之外插入一个新的矩阵来扩展原始矩阵的大小。

% 接上面的代码，在 A 的四五行、五六列加上一个新的矩阵 A(4:5, 5:6) = [2 3; 4 5]	10 20 30 0 0 0 60 70 80 0 0 0 0 0 0 88 0 0 0 0 0 0 2 3 0 0 0 0 4 5

以上就是修改矩阵元素的方法，下面我们再来介绍删除矩阵元素的方法。

如果我们将等号右侧变成空向量[ ]，则可以删除对应位置的元素。需要注意的是，**通常只能删除矩阵的整行或者整列，否则会报错。**

请看下面的例子：

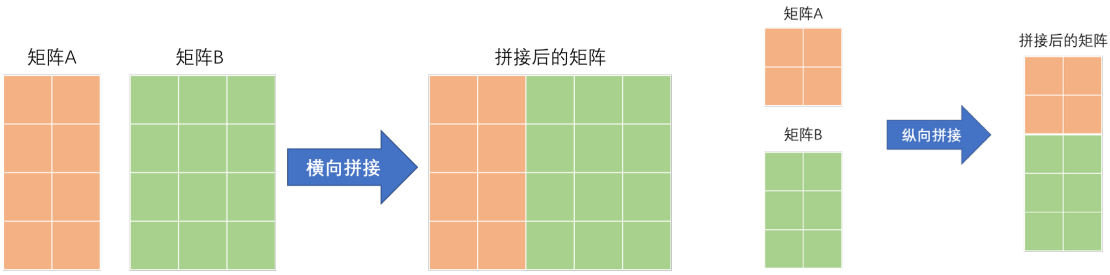
命令	结果
A = [1:4; 2:5; 3:6]	1 2 3 4 2 3 4 5 3 4 5 6
A(:,1) = [ ] % 删除第一列	2 3 4 3 4 5 4 5 6
A = [1:4; 2:5; 3:6] A(:, [2,end]) = [ ] % 删除第二列和最后一列	1 3 2 4 3 5
A = [1:4; 2:5; 3:6] A(2,2) = [ ]	MATLAB 报错：空赋值只能具有一个非冒号索引。
A = [1:4; 2:5; 3:6] A([1,3],[1,2]) = [ ]	MATLAB 报错：空赋值只能具有一个非冒号索引。

注意，也可以通过线性索引来删除矩阵的元素。使用线性索引删除后，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向量中。另外，使用线性索引可以删除任意位置的元素，不需要删除矩阵的一整行或者一整列。

命令	结果
A = [1:4; 2:5; 3:6]	1 2 3 4 2 3 4 5 3 4 5 6
A(1:3) = [ ] % 也可以使用线性索引删除	2 3 4 3 4 5 4 5 6
% 使用线性索引不需要删除一整行或者一整列 A = [1:4; 2:5; 3:6]; A(1:4) = [ ]	3 4 3 4 5 4 5 6

3.3.4 矩阵的拼接和重复

有时候我们需要对多个矩阵进行拼接，变成一个大的矩阵。根据矩阵拼接的方向，我们可以分为横向(水平)拼接和纵向(垂直)拼接，如下图所示：



横向拼接(horizontal catenate)　　纵向拼接(vertical catenate)

如上图所示：**横向拼接要求矩阵的行数相同；纵向拼接要求矩阵的列数相同。**

在 MATLAB 中，我们可以使用命令[A, B] 或 [A B]对矩阵 A 和 B 进行横向拼接，也可以使用 MATLAB 中的内置函数：horzcat(A,B)；类似的，我们可以使用命令[A; B]对矩阵 A 和 B 进行纵向拼接，也可以使用 MATLAB 中的内置函数：vertcat(A,B)。

事实上，horzcat 和 vertcat 两个函数来源自 cat 函数，这里的 cat 不是猫的意思，而是单词 catenate 的缩写，可以翻译成连接。

cat 函数的用法如下：

命令 cat(dim,A,B)表示沿着维度 dim 方向将矩阵 B 拼接到矩阵 A 的末尾。

dim = 1 时表示沿着行方向从上往下进行拼接，即纵向拼接，因此 cat(1,A,B)等价于 vertcat(A,B)；

dim = 2 时表示沿着列方向从左自右进行拼接，即横向拼接，因此 cat(2,A,B)等价于 horzcat(A,B)。

（horzcat 函数中的 horz 取自英文单词 horizontal，表示水平的意思；vertcat 函数中的 vert 取自英文单词 vertical，表示竖直的意思）

总结：若 A 和 B 的行数相同，那么使用[A, B]、[A B]、horzcat(A,B)以及 cat(2,A,B)都能将 A 和 B 横向拼接成一个大的矩阵；若 A 和 B 的列数相同，那么使用[A; B]、vertcat(A,B)以及 cat(1,A,B)都能将 A 和 B 纵向拼接成一个大的矩阵。

下面举几个例子：

（1） 横向拼接的例子

命令	结果
A = [1 6 7; 4 5 7] B = [3 1; 5 10]	A = 1 6 7 4 5 7 B = 3 1 5 10
[A, B] [A B] % 用空格隔开，可以有多个空格 horzcat(A,B) cat(2,A,B)	% 左侧四种方法都可以用于横向拼接 1 6 7 3 1 4 5 7 5 10

（2） 纵向拼接的例子

命令	结果
A = [2 4 5;2 2 4] B = [1 8 6;6 3 10;1 5 5]	A = 2 4 5 2 2 4 B = 1 8 6 6 3 10 1 5 5
[A; B] [A B] % 也可以使用回车键进行纵向拼接 vertcat(A,B) cat(1,A,B)	% 左侧四种方法都可以用于纵向拼接 2 4 5 2 2 4 1 8 6 6 3 10 1 5 5

（3） 拼接时维度不一致导致的报错

如果横向拼接时矩阵的行数不相同，或者纵向拼接矩阵的列数不相同，那么 MATLAB 就会报错。

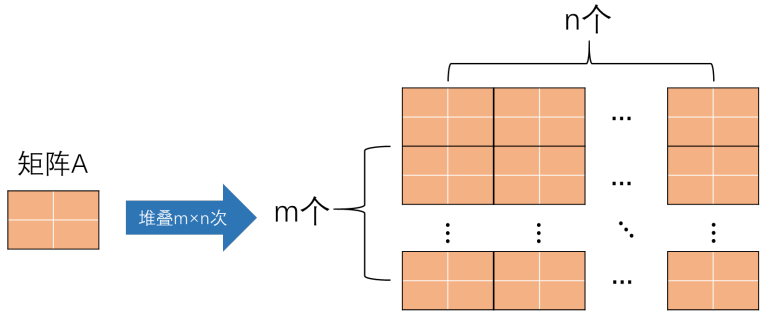
命令	结果
A = ones(2) B = ones(3)	A = 1 1 1 1 B = 1 1 1 1 1 1 1 1 1
[A, B]	错误使用 horzcat 要串联的数组的维度不一致。
[A; B]	错误使用 vertcat 要串联的数组的维度不一致。

注意，如果要拼接的矩阵的个数大于 2，也能使用上面的方法进行拼接，请看下面的例子：

命令	结果
A = ones(2) B = zeros(2) C = eye(2)	A = 1 1 1 1 B = 0 0 0 0 C = 1 0 0 1

[A B C] [A, B, C] cat(2,A,B,C) horzcat(A,B,C)	% 多个矩阵进行横向拼接 1 1 0 0 1 0 1 1 0 0 0 1
[A; B; C] [A B C] cat(1,A,B,C) vertcat(A,B,C)	% 多个矩阵进行纵向拼接 1 1 1 1 0 0 0 0 1 0 0 1

除了对矩阵进行拼接外，有时候我们需要对同一个矩阵进行重复的堆叠。如下图所示，我们将矩阵 A 重复堆叠 m×n 次，得到一个新的矩阵：



在 MATLAB 中，对同一个矩阵进行重复的堆叠的代码为 repmat(A,m,n)。

（如何记住 repmat 这个函数？ repeat 表示重复，matrix 表示矩阵）

我们举两个例子：

命令	结果
A = 1:4; B = repmat(A,3,1)	1 2 3 4 1 2 3 4 1 2 3 4
C = [1,2; 3,4]; D = repmat(C,2,3)	1 2 1 2 1 2 3 4 3 4 3 4 1 2 1 2 1 2 3 4 3 4 3 4

除了对整个矩阵进行重复的堆叠外，**MATLAB 还可以对向量或者矩阵中的元素进行重复，使用到的函数是 repelem**。（如何记住 repelem: repeat 重复 + element 元素）

repelem 函数有两种用法：

（1） 重复向量 v 中的元素：repelem(v, n)

当 n 为一个正整数时，表示把向量 v 中的每一个元素都重复 n 次；n 也可以为一个向量，其长度必须和 v 的长度相同，它可以将 v 中第 i 个位置的元素 v(i)重复 n(i)次，其中 n(i)表示 n 中第 i 个位置的元素。

命令	结果
v = [5,3,8]; repelem(v, 2) %　将 v 中的每个元素都重复 2 次	5　5　3　3　8　8
repelem(v, [2,1,4]) %　将 v 中的第一个元素重复 2 次，第二个元素重复 1 次，第三个元素重复 4 次	5　5　3　8　8　8　8

（2）　重复矩阵 A 中的元素：repelem(A,m,n)

m 和 n 分别表示沿着行方向(从上至下)以及沿着列方向(从左至右)将矩阵元素重复的次数，这里的 m 和 n 可以是正整数，也可以是向量。如果 m 是向量，则 m 的长度要和矩阵 A 的行数相同；如果 n 是向量，则 n 的长度要和矩阵 A 的列数相同。

命令	结果
A = [2,3,5; 　　8,4,7]; repelem(A,3,2)　　%　沿着行方向将 A 的元素重复 3 次，沿着列方向重复 2 次	2　2　3　3　5　5 2　2　3　3　5　5 2　2　3　3　5　5 8　8　4　4　7　7 8　8　4　4　7　7 8　8　4　4　7　7
repelem(A,2,[1,2,3]) % 沿着行方向：每一行都重复 2 次；　沿着列方向：第一列重复 1 次，第二列重复 2 次，第三列重复 3 次	2　3　3　5　5　5 2　3　3　5　5　5 8　4　4　7　7　7 8　4　4　7　7　7
repelem(A,[2,3],[1,2,3]) % 沿着行方向：第一行重复 2 次，第二行重复 3 次；　沿着列方向：第一列重复 1 次，第二列重复 2 次，第三列重复 3 次	2　3　3　5　5　5 2　3　3　5　5　5 8　4　4　7　7　7 8　4　4　7　7　7 8　4　4　7　7　7

3.3.5 矩阵的重构和重新排列

这一小节将介绍一些和矩阵的重构或重新排列相关的函数，下表给出了本小节要学的函数的名称和作用：

函数名	主要作用
`reshape`	更改矩阵的形状
`sort`	对向量或者矩阵进行排序
`sortrows`	基于矩阵的某一列对矩阵进行排序，同一行的元素不会改变
`flip / fliplr / flipud`	将矩阵进行翻转，fliplr 是左右翻转，flipud 是上下翻转
`rot90`	将矩阵按逆时针方向旋转 90 度或者 90 度的倍数

（1）reshape 函数

reshape 函数可以改变矩阵的形状，其常用语法为 reshape(A, m, n)或者 reshape(A,[m,n])，这可以将矩阵 A 的形状更改为 m 行 n 列，前提是转换前后的两个矩阵的元素总数要相同。

例如有一个矩阵 A，它原来的形状是 2 行 6 列，如果我们需要将其形状变成 3 行 4 列，就可以使用命令：reshape(A, 3, 4).

命令	结果
A = randi(10,2,6)　% 生成随机整数矩阵	2 2 10 3 5 1 4 1 4 4 7 9
B = reshape(A,3,4)　% 改变形状	2 1 3 7 4 10 4 1 2 4 5 9

从上面的运行结果可以看出， reshape 函数实际上是按矩阵的线性索引来重新组织矩阵元素的。也就是说，它先取矩阵 A 的第一列，然后是第二列，依此类推，再按新的维度重新组织这些元素。因此，转换后的 B 矩阵中的元素和 A 矩阵中的元素是完全相同的，即 A(:)和 B(:)的结果完全相同。

另外，我们不需要自己来计算转换后的矩阵有多少行或多少列。可以只给出转换后的行数，列数用空向量[ ]代替；或者只给出转换后的列数，行数用空向量[ ]代替。MATLAB 会自动帮我们计算转换后的矩阵大小。例如：若 A 是一个由 12 个元素组成的矩阵，命令 reshape(A,3,[ ])、reshape(A,[ ],4)可以实现和 reshape(A,3,4)一样的效果。

如果你给出的转换后的行数和列数的乘积不等于原始矩阵中元素的个数，那么 MATLAB 就会报错：

命令	结果
A = randi(10,3,6); B = reshape(A,5,8)	错误使用 reshape。元素数不能更改。请使用 [] 作为大小输入之一，以自动计算该维度的适当大小。
A = randi(10,3,6); B = reshape(A,[ ],8)	错误使用 reshape 已知维度的乘积 8 不能被元素总数 18 整除。

（2）sort 函数

sort 函数是用于对向量或矩阵进行排序的。如果输入的参数是矩阵的话，还可以对矩阵的每一行或每一列分别进行排序。

①对向量排序

我们先来学习 sort 函数对向量排序，假设 v 是一个向量，有下面两种基础的用法：

- sort(v) 可以将向量 v 按照从小到大的顺序进行升序排列；
- sort(v, 'descend')可以将向量 v 按照从大到小的顺序进行降序排列。

命令	结果
v = [10 24 16 8 50 40]; v1 = sort(v)	8 10 16 24 40 50
v2 = sort(v,'descend')	50 40 24 16 10 8
% 换个列向量测试 sort 函数 v = [5;-5;7;0;4;5]; v1 = sort(v)	-5 0 4 5 5 7

注意，上面的用法中，sort 函数只有一个返回值，即排序后的向量；事实上，sort 函数可以有两个返回值，基本用法为：`[sort_v, ind] = sort(v)`。这里，sort_v 是排序后的向量，ind 是排序后的向量（即 sort_v）中的每个元素在原向量（即 v）中的索引（即下标、位置）。我们来看一个具体的例子：

v = [10 24 16 8 50 40]; [sort_v , ind] = sort(v)	sort_v = [8 10 16 24 40 50] ind = [4 1 3 2 6 5]

在上面的例子中，我们让 sort 函数返回了两个变量；sort_v 和 ind。它们是两个长度相等的向量，向量的方向和 sort 函数中输入的 v 向量的方向一致，都是行向量。向量 v 中所有元素的最小值为 8，而 8 在 v 中的索引是 4，因此 sort_v 中第一个元素为 8，ind 的第一个元素为 4；向量 v 中第二小的值为 10，而 10 是 v 中的第 1 个元素，因此 sort_v 中第二个元素为 10，ind 的第二个元素为 1；依次类推，可以得到 sort_v 和 ind 向量的值。事实上，这里有一个恒等关系成立：v(ind)运行的结果和 sort_v 的结果完全一样，大家可以自行验证。

下面我们看一个具体的应用场景。假设清风班上有 10 名同学，序号分别是 1 号、2 号一直到 10 号。已知这 10 名同学的成绩构成的向量为：[84 70 61 90 69 78 88 74 92 76]，问：清风班上成绩排名前三的同学的序号是什么？分数分别是多少？

score = [84 70 61 90 69 78 88 74 92 76]; [sort_score, ind] = sort(score,'descend')	sort_score = [92 90 88 84 78 76 74 70 69 61] ind = [9 4 7 1 6 10 8 2 5 3]

根据 MATLAB 返回的结果可以看出：9 号、4 号和 7 号这三名同学的分数排名前三，分别是 92、90 和 88 分。

上面这个问题比较简单，我们再来提一个问题：我们能不能知道这 10 名同学在班上的排名？例如：1 号同学 84 分，在班上排名第 4；2 号同学 70 分，在班上排名第 8；3 号同学 61 分，排名第 10；4 号同学 90 分，排名第 2；…… ；依此类推，最终我们想要得到的排名为：[4 8 10 2 9 5 3 7 1 6].

大家观察 ind 的值和我们想得到的排名的值，应该可以发现如下规律：1 号同学排名第 4，而 ind 中等于 1 的元素的索引也为 4；2 号同学排名第 8，而 ind 中等于 2 的元素的索引也为 8；3 号同学排名第 10，而 ind 中等于 3 的元素的索引也为 10；4 号同学排名第 2，而 ind 中等于 4 的元素的索引也为 2，依次类推，我们可以根据 ind 得到想要的排名。

根据上面的分析，我们可以将 ind 这个向量按照从小到大的顺序排列，排序后的向量是[1,2,3,⋯,10]，且排序后的向量中的每个元素在向量 ind 中的索引就是我们要得到的排名！

因此，我们只需要加下面这行代码，new_ind 就是我们想要计算的排名：

[sort_ind, new_ind] = sort(ind)	sort_ind = [1 2 3 4 5 6 7 8 9 10] new_ind = [4 8 10 2 9 5 3 7 1 6]

注意：如果存在同学的成绩相同的情况，那么这个代码就会存在问题，要想解决这个问题可以用到我们本章后面要学的 ismember 函数，这个问题也将放到本章最后的课后习题中。

② 对矩阵排序

上面介绍的是 sort 函数对向量进行排序的应用，下面我们再来介绍 sort 函数对矩阵 A 进行排序的用法：**sort(A, dim)**.

- dim = 1 时，沿着行方向(从上至下)对矩阵的每一列升序排列
- dim = 2 时，沿着列方向(从左至右)对矩阵的每一行升序排列

注意：（1）当 dim=1 时，sort(A,1)可以直接写成 sort(A)；（2）默认是升序排列的，我们可以在最后面加一个输入参数'descend'，变成从大到小的降序排列；（3）可以有两个返回值，代表的含义和对向量排序类似，表示排序后的元素在原矩阵所在行或所在列中的索引。

命令	结果
A = randi(10,4,6)	4 1 8 7 8 2 3 7 8 6 4 2 8 7 8 4 7 6 1 6 3 1 8 5
% 对矩阵 A 的每一列升序排列 sort(A) sort(A,1)	1 1 3 1 4 2 3 6 8 4 7 2 4 7 8 6 8 5 8 7 8 7 8 6 % 每一列的元素都是按照升序进行排列
% 对矩阵 A 的每一行升序排列 sort(A, 2)	1 2 4 7 8 8 2 3 4 6 7 8 4 6 7 7 8 8 1 1 3 5 6 8 % 每一行的元素都是按照升序进行排列
% 对矩阵 A 的每一行降序排列 sort(A, 2, 'descend')	8 8 7 4 2 1 8 7 6 4 3 2 8 8 7 7 6 4 8 6 5 3 1 1 % 每一行的元素都是按照降序进行排列
% 返回两个值 A = [4 1 8 7 8 2; 3 7 8 6 4 2; 8 7 8 4 7 6; 1 6 3 1 8 5]; [sort_A, ind] = sort(A)	sort_A = 1 1 3 1 4 2 3 6 8 4 7 2 4 7 8 6 8 5 8 7 8 7 8 6
	ind = 4 1 4 4 2 1 2 4 1 3 3 2 1 2 2 2 1 4 3 3 3 1 4 3
	这里是对 A 中每一列进行升序排列。 我们先来解释 ind 的第一列四个元素的含义，第一列的四个元素分别是 4 2 1 3；其中第一个元素 4 表示排序后的 sort_A 中第一列的第一个元素 1 在原始矩阵 A 中第一列的第四个位置；-第二个元素 2 表示排序后的 sort_A 中第一列的第二个元素 3 在原始矩阵 A 中第一列的第二个位置；第三个元素 1 表示排序后的 sort_A 中第一列的第三个元素 4 在原始矩阵 A 中第一列的第一个位置；第四个元素 3 表示排序后的 sort_A 中第一列的第四个元素 8 在原始矩阵 A 中第一列的第三个位置。依次类推，可以解释其他各列 ind 的值的含义。

（3）sortrows 函数

sortrows 函数可以基于矩阵的某一列对矩阵进行排序，排序后得到的新矩阵的同一行元素不会改变。这个函数的用法较多，下面我们直接用一个具体的实例来讲解它的主要用法。

假设清风老师有 6 名学生，下面这个矩阵保存着这六名同学在四门科目上的成绩。矩阵的每一行代表一名学生。这六名同学的四门科目的成绩对应着四列，例如第一名同学的第一科成绩为 95，第二科成绩为 80，依此类推。

$$\text{score} = \begin{bmatrix} 95 & 80 & 85 & 79 \\ 95 & 67 & 78 & 90 \\ 95 & 67 & 78 & 75 \\ 95 & 67 & 64 & 73 \\ 86 & 85 & 82 & 84 \\ 86 & 87 & 84 & 88 \end{bmatrix}$$

请解决下面的问题：

（1） 请基于第一科的成绩按升序对这六名同学进行排序，得到排序后的成绩矩阵。若第一科成绩相同，则基于第二科成绩升序排列。如果第二科成绩还相同，就基于第三科成绩进行排序，依此类推。

```
score = [95 80 85 79;95 67 78 90;95 67 78 75;95 67 64 73;86 85 82 84;86 87 84 88];
sort_score1 = sortrows(score)
```

```
86 85 82 84
86 87 84 88
95 67 64 73
95 67 78 75
95 67 78 90
95 80 85 79
```

（2） 请基于第一科的成绩按升序对这六名同学进行排序。当第一科成绩相同时，请保持其在矩阵中出现的先后顺序。

```
sort_score2 = sortrows(score,1)
```

```
86 85 82 84
86 87 84 88
95 80 85 79
95 67 78 90
95 67 78 75
95 67 64 73
```

（3） 请基于第二科的成绩按升序对这六名同学进行排序。当第二科成绩相同时，请保持其在矩阵中出现的先后顺序。

```
sort_score3 = sortrows(score,2)
```

```
95 67 78 90
95 67 78 75
95 67 64 73
95 80 85 79
86 85 82 84
86 87 84 88
```

（4） 请基于第一科的成绩按升序对这六名同学进行排序。当第一科成绩相同时，基于第三科成绩升序排列。如果第一科和第三科都相同，就保持在矩阵中出现的先后顺序。

```
sort_score4 = sortrows(score,[1,3])
```

```
86 85 82 84
86 87 84 88
95 67 64 73
95 67 78 90
95 67 78 75
95 80 85 79
```

事实上，sortrows(score)等价于 sortrows(score, 1:size(score,2))，即 sortrows(score, [1,2,3,4]).

（5） 请基于第一科的成绩对这六名同学进行降序排列。当第一科成绩相同时，基于第三科成绩降序排列。如果第一科和第三科都相同，就保持在矩阵中出现的先后顺序。

```
sort_score5 =
  sortrows(score,[1,3],'descend')
```

```
    95    80    85    79
    95    67    78    90
    95    67    78    75
    95    67    64    73
    86    87    84    88
    86    85    82    84
```

（6） 请基于第一科的成绩对这六名同学进行降序排列。当第一科成绩相同时，基于第三科成绩升序排列。如果第一科和第三科都相同，就保持在矩阵中出现的先后顺序。

```
sort_score6 =
sortrows(score,[1,3],{'descend','ascend'})
% 使用元胞数组分别表示多个列的方向
```

```
    95    67    64    73
    95    67    78    90
    95    67    78    75
    95    80    85    79
    86    85    82    84
    86    87    84    88
```

（7） 请基于第一科的成绩按升序对这六名同学进行排序。当第一科成绩相同时，请保持其在矩阵中出现的先后顺序，并返回索引值。

```
[sort_score7 , ind7] = sortrows(score,1)
```

```
sort_score7 =
    86    85    82    84
    86    87    84    88
    95    80    85    79
    95    67    78    90
    95    67    78    75
    95    67    64    73
ind7 =
     5
     6
     1
     2
     3
     4
```

ind7 的第一个元素为 5，这表示 sort_score7 的第一行元素在原矩阵中位于第 5 行；ind7 的第二个元素为 6，这表示 sort_score7 的第二行元素在原矩阵中位于第 6 行；依次类推，可以解释 ind7 每一个元素的含义。

因此 sort_score7 和 score(ind7, :)得到的结果完全相同。

通过上面的例子可以看出，**sortrows 函数和 sort 函数的区别**在于：sort 函数会对矩阵的每一列分别进行排序；而 sortrows 函数是基于某一列进行排序的，排序后得到的新矩阵的同一行元素不会改变。

命令	结果
A = randi(10,4,6)	4 1 8 7 8 2 3 7 8 6 4 2 8 7 8 4 7 6 1 6 3 1 8 5
sort(A)	sortrows(A)　% sortrows(A,1:6)
1 1 3 1 4 2 3 6 8 4 7 2 4 7 8 6 8 5 8 7 8 7 8 6	1 6 3 1 8 5 3 7 8 6 4 2 4 1 8 7 8 2 8 7 8 4 7 6

在实际的应用场景中，sort 函数通常只用于对向量进行排序；如果是对矩阵或者表格数据进行排序，我们一般使用 sortrows 函数。如果大家熟悉 Excel 的话，就会发现 Excel 中对数据的排序就和 sortrows 函数类似。在以后的章节中，我们会专门讲解 MATLAB 中的表格数据类型，到时候还会用到 sortrows 函数。

以下是 sortrows 函数的常用用法的总结：

（1） sortrows(A)基于矩阵 A 中第一列元素的值按升序对矩阵进行排序。当第一列包含重复值时，sortrows 会根据下一列中的值进行升序，并对后续的重复值重复此行为。另外，sortrows(A)等价于 sortrows(A, 1:size(A,2)).

（2） sortrows(A,column)基于向量 column 中指定的列对矩阵 A 进行排序。例如，sortrows(A,2)会基于第二列中的元素按升序对矩阵 A 进行排序，如果第二列中有相同的元素，则保持其在矩阵中出现的先后顺序。sortrows(A,[2 3])首先基于第二列中的元素升序，若第二列元素相同，再基于第三列中的元素，对 A 矩阵进行升序排序。如果第三列也出现数值相同的情况，就保持其在矩阵中出现的先后顺序。

（3） 排序时可以指定每一列的排序方向，sortrows(___,direction)按 direction 指定的顺序对 A 进行排序。direction 可以是'ascend'（默认值，对于升序排序）或 'descend'（对于降序排序）。direction 也可以是元素为 'ascend' 和 'descend' 的元胞数组，例如，sortrows(A,[1 3],{'ascend', 'descend'}) 首先基于第一列按升序对 A 进行排序，如果第一列中有数值相同，就基于第三列按降序排序。

（4） 除了返回排序后的矩阵，还可以返回排序后的各行在原矩阵中的位置索引。例如[sorted_A, index] = sortrows(A, column)，此时 A(index, :)的运行结果等于 sorted_A。

（4）flip / fliplr / flipud 函数

下面我们来学习 flip / fliplr / flipud 这三个函数，它们可以用来对向量或矩阵进行翻转操作。其中，flip 函数是一个通用的翻转函数，而 fliplr 和 flipud 是其特例，分别用于从左到右和从上到下的翻转。flip 翻译成中文是翻转，而 fliplr 函数可以拆解为 filp+ 左边 left + 右边 right，flipud 则可以拆解为 flip + 上边 upper + 下边 down，大家可以根据英文来进行记忆。

flip 函数有两种用法：

用法 1： flip(A)

- 如果 A 为向量，flip(A) 将翻转向量中各元素的顺序，向量的方向不变。
- 如果 A 为矩阵，flip(A) 将对矩阵进行上下翻转。

命令	结果
A = [5 2 7 8 9]; % 行向量 flip(A) % 等价于 A(end:-1:1)	9 8 7 2 5
A = [5;2;7;8;9]; % 列向量 flip(A)	9 8 7 2 5

A = [5 8 7; 4 2 6; 3 5 8; 6 4 1]; flip(A) % 对矩阵进行上下翻转	6 4 1 3 5 8 4 2 6 5 8 7

用法 2： flip(A, dim)

flip(A,dim) 沿维度 dim 翻转 A 中元素的顺序。

- dim 为 1 时表示行，此时 flip(A,1) 将沿着行方向对矩阵 A 上下翻转。
- dim 为 2 时表示列，此时 flip(A,2) 将沿着列方向对矩阵 A 左右翻转。

命令	结果
A = [5 8 7; 4 2 6; 3 5 8; 6 4 1];	5 8 7 4 2 6 3 5 8 6 4 1
flip(A,1) % 对矩阵进行上下翻转 % 等价于 flip(A)或者 flipud(A)	6 4 1 3 5 8 4 2 6 5 8 7
flip(A,2) % 对矩阵进行左右翻转 % 等价于 fliplr(A)	7 8 5 6 2 4 8 5 3 1 4 6

请思考：若 A 是一个行向量，flip(A,1)返回的结果为什么还是 A？

答案：此时 A 被当成了一个一行的矩阵，对 A 进行上下翻转不会有变化。因此这提示我们，要对向量进行翻转，直接使用 flip(A)即可。

flip 函数用法总结：

（1） 若要对向量 A 中的元素进行翻转且向量的方向不变，那么可以直接使用 flip(A).

（2） 若要对矩阵 A 进行翻转，那么 flip(A)、flip(A,1)和 flipud(A)都能对矩阵 A 进行上下翻转；flip(A,2)和 fliplr(A)能对矩阵 A 进行左右翻转。

（5）rot90 函数

rot90 函数是对矩阵进行旋转的函数，它源于英文 rotate 一词，中文翻译为旋转。rot90 函数允许我们按 90 度或其倍数**逆时针**旋转矩阵。它的用法非常简单，rot90(A,k)将矩阵 A 按逆时针方向旋转 k*90 度，其中 k 是一个整数；不提供 k 时 k 默认取 1。我们来看几个例子：

命令		结果	
C = [5 8 7; 4 2 6; 3 5 8; 6 4 1];		5 8 7 4 2 6 3 5 8 6 4 1	
rot90(C) % 等价于 rot90(C,1)	rot90(C, 2) % 180°	rot90(C, 3) % 270° 等价于顺时针旋转 90°	
7 6 8 1 8 2 5 4 5 4 3 6	1 4 6 8 5 3 6 2 4 7 8 5	6 3 4 5 4 5 2 8 1 8 6 7	

3.4 矩阵的运算

本节我们将介绍与矩阵相关的一系列运算，包括：调用函数运算、算术运算、关系运算、逻辑运算和集合运算。

3.4.1 调用函数

在上一章中，我们介绍了常见的数学运算函数，例如 abs, sin, round, log 等。这些函数可以直接应用到矩阵上，所表示的含义是：对矩阵中的每个元素分别运用这些数学运算函数，因此返回的结果也是一个矩阵。下面我们来举几个例子：

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

除了这些最基础的数学运算函数外，这一小节我们还要学习下表这些使用频率较高的函数，大家需要熟练掌握它们的用法（点击函数名称可以快速跳转到相应的位置）：

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）

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。

A = 1:100; sum(A)	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）

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

（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)	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。	

（4）指定如何处理 NaN 值

NaN 指不定值或缺失值（Not a Number）。默认情况下，求和时有一个元素为 NaN 值，那么最终的和也为 NaN；我们可以在最后加一个输入参数: 'omitnan', 这样计算时会忽略 NaN 值。

A = [1 5 NaN 10]; % 向量中存在 NaN sum(A)	NaN
sum(A, 'omitnan') % 忽略 NaN 值	16
A = [5 3 -8 4 1 5 NaN 10; 3 6 18 9]; % 矩阵中存在 NaN sum(A) % 计算每一列的和	9 14 NaN 23
sum(A, 'omitnan') % 忽略 NaN 值	9 14 10 23
sum(A, 2) % 计算每一行的和	4 NaN 36
% 忽略 NaN 值 sum(A, 2, 'omitnan')	4 16 36

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)	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)	27 80 90 300
prod(A,2) % 计算每行的乘积	25 1800 1296
v = [2,4,NaN,1,10]; % 有 NaN 值 prod(v)	NaN
% 如果计算时忽略 NaN 值，可以在最后面加上'omitnan'参数 prod(v, 'omitnan')	80
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	15 90 NaN 360
prod(A, 'omitnan') % 忽略 NaN 值	15 90 -144 360
% 忽略 NaN 值计算每一行的乘积 prod(A, 2, 'omitnan')	-480 50 2916

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 6 6 8 11 10 9 11 19 12 10 16
% 计算每一行的累积和 cumsum(A,2)	4 10 16 24 7 11 14 17 8 10 11 16

（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。

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

diff ：计算差分（difference）

差分运算在和时间相关的数据中用的比较多。在原始序列中用下一个数值减去上一个数值可以得到一个新的序列，这个过程就是一阶差分；在一阶差分结果的基础上再进行一次一阶差分，就是二阶差分，举个例子，下表是清风老师 8 年来的体重变化，我们可以计算一阶差分和二阶差分的结果：

年份	体重	一阶差分	二阶差分
2015	60	\	\
2016	65	5	\
2017	66	1	-4
2018	70	4	3
2019	68	-2	-6
2020	72	4	6
2021	64	-8	-12
2022	70	6	14

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).

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

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).

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

注意，diff 函数不支持使用'omitnan'参数来忽略向量或者矩阵中的 NaN 值。

w = [60 65 NaN 70 68 72 64 70]; diff(w)	5 NaN NaN -2 4 -8 6
% 尝试使用'omitnan'参数 diff(w, 'omitnan')	错误使用 diff 差分阶数 N 必须为正整数标量。
% 加上差分阶数 diff(w, 1, 'omitnan')	错误使用 diff 维度参数必须是处于索引范围内的正整数标量。
% 加上维度 diff(w, 1, 2,'omitnan')	错误使用 diff 输入参数太多。

mean ：计算平均值（mean/average value）

假设向量$y = [y_1 \quad y_2 \quad \cdots \quad y_n]$，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

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

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

类似的，dim=1 时，mean(A,1)也可以简写成 mean(A).

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

（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。

A = [5 3 -8 4 1 5 NaN 10; 3 6 18 9]; mean(A) % 计算每一列的平均值	3.0000 4.6667 NaN 7.6667
% 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')	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
mean(A, 2) % 计算每一行的平均值	1 NaN 9
% 忽略 NaN 值计算每一行的平均值 mean(A, 2, 'omitnan')	1.0000 5.3333 9.0000

median：计算中位数

中位数又称中值，我们将数据按从小到大的顺序排列，在排列后的数据中居于中间位置的数就是中位数。

假设有一个向量 $y = [y_1 \quad y_2 \quad \cdots \quad y_n]$，向量$y$中有$n$个元素，我们先将向量$y$按照从小到大的顺序排序，得到新的向量 $y' = [y'_1 \quad y'_2 \quad \cdots \quad y'_n]$，那么当$n$为奇数时，中位数为 $y'_{(n+1)/2}$；当n为偶数时，中位数为 $\frac{y'_{n/2}+y'_{n/2+1}}{2}$.

下面是手算中位数的步骤：

y	y'	n	中位数
[6 80 1 5 100 20 1000]	[1 5 6 20 80 100 1000]	7	20
[3 6 80 1 5 10 20 100]	[1 3 5 6 10 20 80 100]	8	（6+10）/2=8

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。

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

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

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。

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

注意：使用 mode 函数计算众数时会自动忽略 NaN 值，我们不能额外添加'omitnan'参数，否则会报错。

A = [2 3 -1 NaN 1 NaN NaN 3]; mode(A)	3
mode(A, 'omitnan')	低版本 MATLAB：DIM 参数必须为用于指定 X 的维度的标量。 高版本 MATLAB：维度参数必须为正整数标量、由唯一正整数组成的向量或 'all'。 % 不同版本的 MATLAB 报错信息可能有区别，这是因为高版本 MATLAB 中的 mode 函数支持更多种用法。大家可以使用 doc mode 命令来查看你的版本对应的帮助文档。
mode(A, 2, 'omitnan')	错误使用 mode 输入参数太多。

现在我们来看 mode 函数有两个返回值的例子，如果 A 是一个向量，[M,F] = mode(A)得到的 M 表示向量 A 的众数，F 表示众数 M 在向量 A 中出现的次数。

A = [-1 2 0 8 -1 0 2 1 8 0 8]; [M, F] =mode(A)	M = 0　F = 3

上面这个例子中，有两个众数，分别是 0 和 8，它们出现的次数都是 3 次，此时 MATLAB 会返回最小的那个数作为众数。有同学会想，MATLAB 能不能把这些众数都输出呢？

当然可以，我们需要用到 mode 函数的第三个返回值：[M,F,C] = mode(A), 这里的 C 是一个元胞数组，元胞数组里面有一个列向量，列向量中的每个元素都是向量 A 的众数。（注意，元胞数组(cell array)是使用大括号{}括起来的，元胞数组里面的元素可以包含不同的数据类型，例如标量、向量、矩阵、字符串等，第五章中我们会专门讲解元胞数组的用法）

```
A = [-1 2 0 8 -1 0 2 1 8 0 8];
[M, F, C] =mode(A)
```

```
M = 0     F = 3
C =   1×1 cell 数组
      {2×1 double}
```

变量 C 就是元胞数组，它的大小为 1×1，即一行一列，因此里面只有一个数据，这个数据是一个 2 行 1 列的列向量。我们可以使用大括号索引的方式来引用元胞数组中的数据：

```
C{1}
```

```
0
8
```

命令 C{1}可以提取出元胞数组 C 中的第一个数据：列向量[0; 8]。这个列向量中的元素 0 和 8 都是向量 A 的众数。

以上是 A 为向量时的情况，如果 A 是一个矩阵，mode 函数也可以有两个返回值或三个返回值，后两个返回值代表的含义与 A 为向量时类似。我们直接看下面的例子：

```
A =   [3 3 1 4;
       0 0 1 1;
       0 1 6 4;
       1 5 6 8];
[M, F, C] = mode(A) % 计算每一列的众数
```

```
M =
     0     0     1     4
F =
     2     1     2     2
C =   1×4 cell 数组
    {[0]}    {4×1 double}    {2×1 double}    {[4]}
```

M 是一个包含四个元素的行向量，里面第 k 个元素表示第 k 列的众数，例如第 1 列的众数为 0；如果有多个众数，那么会返回最小的那个值，因此第 2 列返回的众数为 0。

F 也是一个包含四个元素的行向量，它表示 M 中的各个众数在其所在列中出现的次数，例如第 1 列的众数 0 在第 1 列出现了 2 次。

C 是一个大小为 1×4 的元胞数组，里面的第 k 个数据表示第 k 列的所有众数。从元胞数组 C 的结果来看，第一个和第四个数据分别为 0 和 4，这说明第 1 列和第 4 列都只有一个众数，分别是 0 和 4；C 中第二个数据是一个 4 行 1 列的列向量，这说明第 2 列有四个众数；C 中第三个个数据是一个 2 行 1 列的列向量，这说明第 3 列有两个众数。

我们可以使用大括号索引提取 C 中的每个数据：

```
C{1}
```

```
0
```

```
C{2}
```

```
0
1
3
5
% 会自动对所有的众数按照从小到大的顺序排序
```

```
C{3}
```

```
1
6
```

```
C{4}
```

```
4
```

类似的，我们也可以计算 A 矩阵每一行的所有众数，结果如下：

```
[M, F, C] = mode(A, 2)   %计算 A 矩阵每一行的众数
```

```
M =
     3
     0
     0
     1
% 每一行的众数
```

```
F =
     2
     2
     1
     1
% 每一行众数出现的次数
```

```
C =
  4×1 cell 数组
    [     3]
    [2×1 double]
    [4×1 double]
    [4×1 double]
```

C{1}	C{2}	C{3}	C{4}
3	0 1	0 1 4 6	1 5 6 8

下面对 mode 函数做一个总结：

（1） mode 函数可用来计算一个向量或者矩阵中的众数。如果 mode 函数只有一个返回值，那么它的用法和 mean 函数、median 函数类似。

（2） 如果 A 是一个向量，那么 M＝mode(A)可以计算向量 A 的众数，若存在多个众数，则只会返回其中的最小值。

（3） 如果 A 是一个矩阵，那么 M = mode(A, dim)可以计算矩阵 A 沿着维度 dim 中所有元素的众数。当 dim=1 时沿着行方向进行计算，即得到每列的众数；当 dim=2 时沿着列方向进行计算，即得到每行的众数。另外，mode(A,1)可以简写成 mode(A)。

（4） mode 函数不能使用'omitnan'参数，MATLAB 在计算众数时会自动忽略向量或者矩阵中的 NaN 值。

（5） mode 函数最多能有三个返回值：[M,F,C]＝mode(A). 如果 A 是向量，那么 M 表示向量 A 的众数，F 表示众数 M 在向量 A 中出现的次数，如果 A 中存在多个众数，MATLAB 会返回最小的那个数作为 M，第三个返回值 C 则是一个元胞数组，里面包含了 A 的所有众数。如果 A 是矩阵，那么 M、F 和 C 分别代表每一列的众数、每一列众数在所在列中出现的次数以及每一列的所有众数构成的元胞数组。

另外，本小节用到了元胞数组的知识，元胞数组中的元素用大括号{}括起来，各元素可以是不同的数据类型。要引用元胞数组中的数据可以使用大括号进行索引，例如，要提取元胞数组 C 中的第一个数据，可以使用 C{1}。在第五章中，我们会系统学习元胞数组的用法，感兴趣的同学可以提前在 MATLAB 官网搜索关键词进行学习。

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。

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$X:\{X_1,X_2,\cdots,X_n\}$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$Y:\{Y_1,Y_2,\cdots,Y_n\}$是样本数据（例如抽样调查的结果）

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

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。

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.

v = [6 8 10 12 14]; var(v)　% 样本方差，等价于 var(v,0)	10
var(v,1)　% 总体方差	8
% 下面看矩阵的例子 A = randi(10,4,5)	9　7　10　10　5 10　1　10　5　10 2　3　2　9　8 10　6　10　2　10
var(A)　% 每一列的样本方差 % 也可以写成 var(A,0)或者 var(A,0,1)	14.9167　7.5833　16.0000　13.6667　5.5833
var(A,0,2)　% 每一行的样本方差	4.7000 16.7000 11.7000 12.8000
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	NaN NaN 11.7000 12.8000
% 忽略 NaN 值计算每一行的样本方差 var(A, 0, 2, 'omitnan')	2.0000 6.2500 11.7000 12.8000

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
% 下面看矩阵的例子 A = randi(10,4,5)	9　7　10　10　5 10　1　10　5　10 2　3　2　9　8 10　6　10　2　10
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)	3.8622　2.7538　4.0000　3.6968　2.3629
std(A,0,2)　% 每一行的样本标准差	2.1679 4.0866 3.4205 3.5777
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	NaN NaN 3.4205 3.5777
% 忽略 NaN 值计算每一行的样本标准差 std(A, 0, 2, 'omitnan')	1.4142 2.5000 3.4205 3.5777

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 = [2　6　10 6　7　5 5　3　7 2　6　5];	B = [1　1　6 3　6　9 9　4　9 4　8　2];	% min(A,B)的结果： 1　1　6 3　6　5 5　3　7 2　6　2

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 = [2　6　10 6　7　5 5　3　7 2　6　5];	% B 可以是一个标量 B = 5;	2　5　5 5　5　5 5　3　5 2　5　5 % A 中每一个元素和 5 比较大小，取较小的值。
A = [2　6　10 6　7　5 5　3　7 2　6　5];	% B 可以是一个具有三个元素的行向量 B = [4 5 6];	2　5　6 4　5　5 4　3　6 2　5　5 % 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。

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。

A = [5 6 7 3 5 3 10]; min(A)	3
B = [4.5　3+4i　6]; min(B)	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）。

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

有同学可能会问：为什么中间要加一个空向量[ ]？如果不加的话，就是将 A 中每个元素和 1 或者 2 比较大小，并返回较小值。

min(A, 1)	1　1　1　1 1　1　1　1 1　1　1　1
min(A, 2)	2　2　2　2 1　2　1　2 2　2　2　2

（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。

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

上面我们介绍了 min 函数的两种用法，如果向量或者矩阵中存在 NaN 值，min 函数会自动忽略，大家不需要单独对 NaN 值进行处理。

% 存在缺失值的例子 A = [5 6 7 NaN 5 3 10]; [min_A , ind] = min(A)	min_A = 3 ind = 6

% 求 x 和 y 对应元素的最小值
x = [1 NaN 3 5 2 NaN 1];
y = [2 5 4 NaN 2 NaN 9];
min(x, y)
	1 5 3 5 2 NaN 1

% 求两个矩阵对应位置元素的最小值时，min 函数只能有一个返回值！ [min_xy, ind] = min(x, y)	错误使用 min 不支持具有两个要比较的矩阵和两个输出参数的 MIN。

max :求最大值（maximum value）

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 = [2 6 10 6 7 5 5 3 7 2 6 5];	B = [1 1 6 3 6 9 9 4 9 4 8 2];	% max(A,B)的返回结果 2 6 10 6 7 9 9 4 9 4 8 5

A = [5 6 10 7 3 5 3 10]; % 计算向量 A 的最大值并返回其索引 [max_A, ind] = max(A)	max_A = 10 ind = 3
A = randi(10,3,4)	6 9 2 8 5 5 1 3 6 9 8 2
max(A, [ ], 1) % 求每一列的最大值 % 可以简写成 max(A)	6 9 8 8
max(A, [ ], 2) % 求每一行的最大值	9 5 9
max(A, 2) % 返回每个元素和 2 相比的较大值	6 9 2 8 5 5 2 3 6 9 8 2
[max_A , ind] = max(A,[ ],2) % 求每一行的最大值并返回索引	max_A = ind = 9 2 5 1 9 2
[max_A , ind] = max(A) % 求每一列的最大值并返回索引	max_A = 6 9 8 8 ind = 1 1 3 1
% 存在缺失值的例子 A = [5 6 7 NaN 5 3 10 5]; [max_A , ind] = max(A)	max_A = 10 ind = 7

3.4.2 算术运算

MATLAB 的基本算术运算符有：＋(加)、－(减)、∗(乘)、/(右除)、\(左除)、^(乘方)和′(转置)，下面我们分别进行介绍。

首先是**矩阵的加法**。在线性代数中，只有两个大小完全相同的矩阵才可以进行相加运算，而在 MATLAB 中，只要两个矩阵的大小兼容，就能够进行计算。

以矩阵的加法为例，下表我们给出了 MATLAB 支持的**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)

事实上，执行加法运算时，MATLAB 会将大小兼容的矩阵隐式扩展为相同的大小，然后再将对应位置的元素相加，这种计算方式在 MATLAB 中称为“按对应位置的元素运算”。我们在上一节介绍的 min 函数和 max 函数在计算两个矩阵对应位置元素的最小值和最大值时也支持上表五种模式。

类似的，**矩阵的减法**也支持上表五种兼容模式，其计算方式也是“按对应位置的元素运算”，我们看下面的例子：

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

矩阵的乘法则有所不同，在 MATLAB 中，**矩阵的乘法分为两种**：第一种是线性代数中定义的矩阵的乘法，使用的运算符号是**乘号“*”**，例如矩阵 A*B，矩阵的乘法必须要满足前面矩阵 A 的列数和后面矩阵 B 的行数相等；第二种是按对应位置的元素相乘，我们需要使用运算符号**点乘“.*”**，例如 A.*B，此时 A 和 B 的大小只需要满足上方表格介绍的五种兼容模式。特别地，如果一个矩阵和标量（常数）相乘，那么使用乘号“*”和点乘“.*”得到的结果相同。

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

下面我们再来介绍**矩阵的除法**。事实上，在我们学的线性代数中，矩阵并不支持除法的运算，但 MATLAB 中定义了矩阵除法的计算规则。

在 MATLAB 中，除号有两种，分别是 / (右除)和 \ (左除)，命令“x = B/A”表示对线性方程组 x*A = B 求解 x；命令“x = A\B”则表示对线性方程组 A*x = B 求解 x。这两个符号使用频率很低，大家不需要强记，需要用时查询即可。

我们平时使用更多的是对矩阵进行点除的操作，即将两个矩阵按对应位置的元素做除法。其中命令**“A./B”表示用 A 的每个元素除以 B 的对应元素**，A 和 B 的大小必须兼容；命令“A.\B”则表示用 B 的每个元素除以 A 的对应元素，这个用法不太符合我们的习惯。因此，大家只需要掌握“A./B”的用法即可。

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

特别地，如果 B 是标量，那么 A./B 的结果和 A/B 的结果相同。

矩阵的乘方也有两种用法，分别是“^”和“.^”。

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

命令	随机生成的 A	计算结果
A = randi([0,10],2,2) A ^ 2	1 4 3 0	13 4 3 12
A = randi([0,10],2,3) A .^ 0.5	1 4 8 2 9 2	1.0000 2.0000 2.8284 1.4142 3.0000 1.4142
A = randi([0,10],2,3) A .^ 2	9 2 8 4 1 6	81 4 64 16 1 36

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 = [1 2 3; 2 2 1; 3 4 3]	1 2 3 2 2 1 3 4 3
B1 = A ^ (-1)	1.0000 3.0000 -2.0000 -1.5000 -3.0000 2.5000 1.0000 1.0000 -1.0000
B2 = inv(A)	1.0000 3.0000 -2.0000 -1.5000 -3.0000 2.5000 1.0000 1.0000 -1.0000
B1*A	1.0000 0.0000 0 -0.0000 1.0000 -0.0000 0.0000 0 1.0000

根据线性代数中逆矩阵的定义，互为逆矩阵的两个矩阵的乘积为单位矩阵（主对角线为 1，其余位置为 0 的方阵），我们通过上表的第三行代码验证了这一点。注意：由于浮点数运算的误差，计算结果中的元素可能和 0 或 1 有微小的差异，我们使用 format long g 命令显示更多的小数点：

命令	结果
format long g B1*A	1 1.77635683940025e-15 0 -8.88178419700125e-16 1 -8.88178419700125e-16 4.44089209850063e-16 0 1

例如第一行第二个元素，理论上应为 0，但 MATLAB 计算结果约为 1.776×10^{-15}。

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 = [3 1 + 3i; 2 5; 4 - 2i 6];	3 2 4 + 2i 1 - 3i 5 6	3 2 4 - 2i 1 + 3i 5 6

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。

（我个人觉得：矩阵的转置应该属于线性代数中的知识点，但是 MATLAB 官网将其视为算术运算的一种。因此，本书也是参考官网的安排将其放在了这一小节）

学完了矩阵的算术运算后，我们来做一些练习。

（1）计算用来评价预测效果好坏的一些指标

假设真实值是向量$y=[y_1,y_2,\cdots,y_n]$，拟合值或预测值是向量$\hat{y}=[\hat{y}_1,\hat{y}_2,\cdots,\hat{y}_n]$

```
% 例如我们举一个 n 为 10 的例子
y = [100 102 108 117 135 178 198 241 290 349];
y_hat = [93 108 118 117 141 170 196 249 296 359];
n = length(y);      % 10
```

- SSE: 误差(或残差)平方和（Sum of Squares due to Error）

$$SSE = \sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

```
SSE = sum( (y-y_hat).^2 )   % 489
```

- MSE: 均方误差（Mean Square Error）

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

```
MSE = 1/n*(sum( (y-y_hat).^2 ))   % 48.9
```

- RMSE: 均方根误差（Root Mean Square Error）

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$$

```
RMSE = sqrt( 1/n*(sum((y-y_hat).^2)) )   % 6.9929
```

- MAE: 平均绝对误差（Mean Absolute Error）

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|$$

```
MAE = 1/n*( sum( abs(y-y_hat) ) )   % 6.3
```

- MAPE: 平均绝对百分比误差（Mean Absolute Percentage Error）

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i-\hat{y}_i}{y_i}\right|$$

```
fz = y-y_hat;   % 分子
MAPE = 1/n*(sum(abs(fz ./ y)))   % 0.0403
```

- SMAPE: 对称平均绝对百分比误差（Symmetric Mean Absolute Percentage Error）

$$SMAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i-\hat{y}_i|}{(|y_i|+|\hat{y}_i|)/2}$$

```
fz = abs(y-y_hat);   % 分子
fm = (abs(y)+abs(y_hat))/2;   % 分母
SMAPE = 1/n*(sum(fz./ fm ))   % 0.0399
```

- R^2：决定系数（Coefficient of determination）

$$R^2=1-\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}, \text{式中}\bar{y}=\frac{1}{n}\sum_{i=1}^{n}y_i$$

```
fz = sum((y - y_hat).^2);   % 分子
fm = sum((y - mean(y)).^2);     % 分母
R2 = 1 - fz/fm   % 0.9928
```

（2）计算优化算法中常见的测试函数（假设公式中的 x 向量为 1:10）

函数名	函数表达式
Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$
Rastrigin	$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$
Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$
Rosenbrock	$f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$

参考答案如下：不熟练的同学请看配套的视频讲解，视频中讲解的很详细：

```
% Sphere 函数
x = 1:10;
y = sum(x.^2)

% Rastrigin 函数
x = 1:10;
y = sum(x.^2-10*cos(2*pi*x)+10)

% Griewank 函数
x = 1:10;
n = length(x);
y = 1/4000*sum(x.*x)-prod(cos(x./sqrt(1:n))) + 1

% Rosenbrock 函数
x = 1:10;
tem1 = x(1:end-1);
tem2 = x(2:end);
y = sum(100 * (tem2-tem1.^2).^2 + (tem1-1).^2)
```

3.4.3 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
```

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 4 1 1 4 5 4 4 4	0 1 1 0 1 0 1 0 0 1 0 0
2 3 5	3 2	0 1 0 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。

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。

因此，为了避免不必要的错误，我们应该使用下一小节中介绍的逻辑运算函数（逻辑与&、逻辑或|、逻辑非~等）连接多个关系表达式，例如判断 a<b<c 对应的命令应为(a<b)&(b<c)，这里的&表示逻辑与，只有当 a<b 和 b<c 同时成立时才返回逻辑值 1，下一小节会详细介绍。

另外，从上面的例子可以看出，逻辑值 1 和逻辑值 0 不仅仅用于表示真 true 和假 false，它们在进行计算时也可以被视为数值 1 和数值 0。

请看下面的例子：

命令	结果
x = randi(10, 3, 4)	2 9 6 7 4 1 5 3 9 4 7 5
y = x > 5 % 也可以写成 y = (x>5)，加括号会让运算的顺序更清晰	3×4 logical 数组 0 1 1 1 0 0 0 0 1 0 1 0
sum(y) % 计算 y 矩阵每一列的和	1 1 2 1 % x 的每一列中大于 5 的元素个数
sum(y(:)) % 计算所有元素的和	5 % x 中大于 5 的元素个数有 5 个

再来看一个例子：模拟投硬币 10 万次，计算出现正面的次数和频率。

```
n = 1e5;                % 科学计数法表示 100000
x = randi([0,1], n, 1); % 0 表示正面，1 表示反面
s = sum(x == 0)         % 出现正面的次数
pl = s / n              % 频率
```

下面我们再来看一个易错点：**判断浮点数是否相等**

先给大家简单介绍下什么是浮点数，浮点数是计算机科学中用于近似表示实数的一种数值表示法。简单来说，浮点数由两部分组成：一个表示数字的部分（称为尾数）和一个表示数字所在位置的指数。例如，数字 1234.56 可以写成 1.23456×10^3，其中 1.23456 是尾数，3 是指数。

有编程基础的同学应该知道，计算机中的数据是使用二进制 0 和 1 来保存的，例如十进制表示的 3 在二进制中表示为 11。虽然十进制能精确表示如 0.1 这样的数，但在二进制中，0.1 却需要表示为一个无限循环的小数 0.0001100110011001…。由于计算机的内存是有限的，这导致了它无法精确表示 0.1 这样的数值。计算机在处理这类数字时会进行截断或舍入，所以使用浮点数计算时可能会产生误差。

MATLAB 中的浮点数分为两种，分别是双精度(double)浮点数和单精度(single)浮点数，两种浮点数能表示的数值的范围有所不同。默认情况下我们创建的向量或者矩阵中的元素都是使用双精度浮点数表示的，对这部分内容感兴趣的同学可点击下方链接，MATLAB 的官网有详细介绍两种的区别：https://ww2.mathworks.cn/help/matlab/matlab_prog/floating-point-numbers.html。

大家可以尝试计算0.5 − 0.4 − 0.1，这是我的 MATLAB 返回的结果：

C = 0.5 - 0.4 - 0.1 C == 0	C = -2.7756e-17 ans = logical 0

理论上 C 应该等于 0，但由于浮点数计算的误差，MATLAB 得到的 C 约为-2.7756×10^{-17}，这是一个非常接近 0 的数。当我们试图判断 C 是否等于 0 时，MATLAB 返回逻辑 0，即 MATLAB 认为 C 不等于 0，这显然不是我们期望的结果。

那么我们应该怎样判断两个浮点数相等呢？

解决方法：使用一个很小的正数来比较浮点数，而不是直接使用双等号（==）判断。我们将这个很小的正数称为容差（tolerance，简称为 tol），例如要比较 A 和 B 两个数是否相等，只需要满足：$|A-B| \le tol$，这里的容差 tol 通常取一个非常小的正数，例如 tol 可取成 1e-12(10 的-12 次方)。tol 越小，判断两个浮点数相等的要求越严格。

```
abs(C-0) <= 1e-12
```

```
ans =  logical
   1
```

当然，并非 MATLAB 中所有的浮点数计算都会出现误差。例如，如果我们仅交换 0.1 和 0.4 的位置：

```
% 仅交换 0.1 和 0.4 的位置
C = 0.5 - 0.1 - 0.4
C == 0
```

```
C =
     0
ans =  logical
   1
```

总之，当我们需要判断两个浮点数是否相等时，务必要考虑到计算的误差。感兴趣的同学可以进一步搜索和学习有关浮点数的知识。

3.4.4 逻辑运算

3.4.4.1 逻辑运算函数

在上一节介绍关系运算时，我们提到过：MATLAB 中使用逻辑值表示布尔变量，逻辑值 1 代表真（true），逻辑值 0 代表假（false）。而逻辑运算就是对逻辑值进行的运算。大部分的编程语言都会涵盖四个最基本的逻辑运算方法：**逻辑与、逻辑或、逻辑非和逻辑异或。**

下表给出了 MATLAB 中的定义（注意：下表中的 1 和 0 是逻辑值 1 和 0）：

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	不相同时取 1，相同时取 0	xor(1,0)　xor(0,1) % 返回 1 xor(1,1)　xor(0,0) % 返回 0

注意，上表中函数名和对应的运算符可以执行相同的功能，除了“逻辑异或”没有相应的运算符外，剩下三个运算方法都有对应的运算符。

这四个运算方法的使用方法较为相似，以“逻辑与”为例，大家可以查看“逻辑与”的帮助文档：A & B 对数组 A 和 B 执行逻辑 AND 操作，并返回包含设置为逻辑值 1（true）或逻辑值 0（false）的元素的数组。如果 A 和 B 在相同的数组位置都包含非零元素，则返回的数组中对应位置的元素将设置为逻辑值 1（true）。如果不是，则将数组元素设置为 0。and(A,B) 是执行 A & B 的替代方法，但很少使用。（注：该帮助文档来自于 MATLAB2023 之前的版本，新版本增加对表格类型数据的计算）

从帮助文档可以得知：（1）MATLAB 推荐大家直接使用运算符进行计算，因此&、|和~这三个符号的功能大家要牢记，我们主要使用这三个符号而不是对应的函数进行计算。（2）“逻辑与&”是对数组 A 和 B 进行计算的，计算时会比较 A 和 B 对应位置的元素。数组 A 和 B 的大小不一定要完全相同，只需要符合算术运算中介绍的五种兼容模式即可。（3）“逻辑与&”不仅可以作用在逻辑值 0 和 1 上，还可以用于普通的数值上，这时候，**MATLAB 会将非零数值视为逻辑 1，将数值零视为逻辑 0 进行运算。**例如：3&5 返回逻辑值 1，-4&0 返回逻辑值 0。我们来看两个例子：

A	B	A&B 的计算结果
-2 0 7 0 0	1 3 5 0 1	1×5 logical 数组 1 0 1 0 0
6 0 0	0 0 1 -1 2 0	2×3 logical 数组 0 0 0 1 0 0

有同学可能会有疑问，为什么 MATLAB 会将非零数值视为逻辑 1，将数值零视为逻辑 0？这是因为 MATLAB 在进行逻辑运算之前，在计算机内部自动将数值转换成了逻辑值。我们也可以使用 **logical 函数**手动进行转换：

L = logical(A) 将 A 转换为一个逻辑值数组。A 中的任意非零元素都将转换为逻辑值 1（true），零则转换为逻辑值 0（false）。复数值和 NaN 不能转换为逻辑值。

举个例子，我们随机生成一个 4 行 3 列的矩阵，将矩阵中非 0 位置的元素转换成逻辑值 1，等于 0 的位置的元素转换成逻辑值 0：

代码	A 的结果	转换后的结果
A = randi([-5,5],4,3) logical(A)	-5 -1 0 -2 4 -2 0 2 -4 2 5 1	4×3 logical 数组 1 1 0 1 1 1 0 1 1 1 1 1

特别地，我们还可以**使用 true 和 false 函数分别创建全为逻辑 1 和逻辑 0 的逻辑矩阵**。以 true 函数为例，它的主要用法有两种：（1）true(n)可以生成一个 n 行 n 列全为逻辑值 1 的方阵，特别地，当 n 等于 1 时可以简写为 true，此时表示一个常量；（2）true(m,n)可以生成一个 m 行 n 列全为逻辑值 1 的矩阵。

命令	结果
true	logical 1

true(2) % 等价于 logical(ones(2))	2×2 logical 数组 1 1 1 1
false(2,3) % 等价于 logical(zeros(2,3))	2×3 logical 数组 0 0 0 0 0 0

除了“逻辑与&”外，剩下的三个逻辑运算函数也可以用于普通的数值上，MATLAB 也会将非零数值视为逻辑 1，将数值零视为逻辑 0 进行运算，下面我们举几个例子：

命令	结果
A = randi([-3,3],2,4)	-1 -3 0 3 0 -3 2 -3
B = randi([-3,3],1,4)	-3 2 0 0
B | 0	1×4 logical 数组 1 1 0 0
A | B	2×4 logical 数组 1 1 0 1 1 1 1 1
~A	2×4 logical 数组 0 0 1 0 1 0 0 0
xor(3,4)	logical 0 % 有同学会说返回逻辑 1，因为表中说过：不相同时取 1，相同时取 0。这是错的！xor 函数比较的是数值的逻辑值，而不是数值本身。这里的 3 和 4 在 MATLAB 内部都会被转换成逻辑值 1，这时候就相当于计算 xor(1,1)，结果为逻辑 0.
xor(A,B)	2×4 logical 数组 0 0 0 1 1 0 1 1

a = [0 1 NaN 1];
b = [0 1 1 1];
a & b
a | b
~a
xor(a,b)
	% 如果数据中存在 NaN 值，执行逻辑运算会报错
	NaN 值无法转换为逻辑值。

a = 1+3i;
b = 1;
a & b
a | b
~a
xor(a,b)
	% 如果数据中存在复数，执行逻辑运算也会报错
	操作数必须为实数。

另外，我们有时候也会使用**“逻辑与 &”和“逻辑或 |”进行连续运算**，例如 1 & 2 & 3 和 0 | 3 | 0 的结果都是逻辑 1，下面我们再看几个例子：

命令	结果
A = randi([-3,3],2,4)	0 1 -2 1 -2 0 2 2
B = randi([-3,3],1,4)	-2 2 0 3
C = randi([-3,3],1,4)	1 0 -3 3
A & B & C	2×4 logical 数组 0 0 0 1 1 0 0 1
A | B | C	2×4 logical 数组 1 1 1 1 1 1 1 1
A & B | C	2×4 logical 数组 1 1 1 1 1 0 1 1
A & (B | C) % 使用小括号指定计算的先后顺序	2×4 logical 数组 0 1 1 1 1 0 1 1

下面我们再次谈谈运算优先级的问题，MATLAB 中的运算符有不同的计算优先级，优先级高的先进行计算，例如 3+4*2 等于 11 而不是 14，这是因为乘法的优先级高于加法。类似的，关系运算符(例如>、==、~=)的优先级要高于逻辑运算符&和|，例如 3 > 4 & 2 > -1 的返回结果是逻辑 0。大家可以去 MATLAB 官网查看运算符优先级的帮助文档，但没有必要刻意去记，我们只需要养成一个好的习惯：**使用小括号来指定计算的先后顺序**，例如我们可以将上面的代码改成(3 > 4) & (2 > -1)，这样计算的先后顺序看起来会清楚很多。另外有一个特殊的优先级顺序需要大家了解， & 运算符的优先级要高于 | 运算符。尽管 MATLAB 通常按从左到右的顺序计算表达式，但表达式 a|b&c 按 a|(b&c) 形式计算，因此，对于同时包含 & 和 | 的语句，比较好的做法是使用小括号显式地指定期望的语句优先级。

我们来做一个小练习：随机生成一个具有 20 个元素的向量，用来表示 20 名同学的成绩（假设成绩为满分 100 分的整数制）。请通过代码计算得到一个具有 20 个元素的逻辑向量，向量中对应位置的元素为逻辑值 1 时表示该同学的成绩在区间[60,80)内，为逻辑值 0 时表示成绩在区间[60,80)外。

答案如下：

```
A = randi([0,100],1,20)
res = (A >= 60) & (A < 80)
```

一定要注意：不能写成：60 <= A < 80 哦！

如果要找出[0 , 60) ∪ [80 , 100]分的同学呢？

答案如下(下面两种方法都可以)：

```
方法 1：(A < 60) | (A >= 80)
方法 2：~((A >= 60) & (A < 80))
```

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。

思考题：前面我们介绍过 logical 函数，它能将数值转换为逻辑值，但如果输入的是 NaN，那么 MATLAB 会报错，请大家思考：分别运行(10 > 3) | logical(NaN)和(10 > 3) || logical(NaN)的结果是什么？

答案如下：

运行 (10>3)|logical(NaN) 会报错，因为前后两项都会被计算，所以当 MATLAB 运行到后面的 logical(NaN)时会报错；运行 (10>3)||logical(NaN) 会返回逻辑 1，因为前面一项（10>3）返回逻辑 1，此时会触发||的短路机制，这时候就会直接返回逻辑 1。

3.4.4.2 利用逻辑值引用矩阵的元素

前面我们学过 MATLAB 中的向量和矩阵，并介绍过如何通过元素所处的位置索引来引用（提取）所需的元素。例如：若 a 是一个向量，则 a(1:2:end)表示提取 a 中所有奇数位置的元素；若 A 是一个矩阵，则 A(1:2:end, :)表示提取 A 中所有奇数行的元素。

本小节我们介绍引用矩阵的元素的另一种方式：利用逻辑值。

假设有一个 m 行 n 列的矩阵 A，我们要提取其指定位置的元素，那么我们可以生成一个和 A 同样大小的逻辑矩阵 L，L 中的元素要么是逻辑值 1，要么是逻辑值 0，其中：等于逻辑值 1 的元素所处的位置是我们所需要的。接着我们只需要使用命令 A(L)，就能够在 A 中提取出指定位置的元素。通常逻辑矩阵 L 是由一系列的逻辑运算或者条件运算得来的。

举个例子，我们要提取 A 中所有小于等于 3 的元素，这里的 A 是我随机生成的一个矩阵：

命令	结果
A = randi(10,3,4)	3 5 6 2 4 1 8 4 6 7 7 10
L = (A <= 3)	3×4 logical 数组 1 0 0 1 0 1 0 0 0 0 0 0
A(L)	3 1 2

这里的 L 就是通过关系运算得到的一个逻辑矩阵，它的大小和 A 相同，L 中为逻辑值 1 的位置就是矩阵 A 中要引用的位置。

注意：命令 A(L)得到的结果是一个列向量，有同学会问：为什么 A(L)返回的列向量中的元素顺序是 3,1,2，而不是其他的顺序？这是因为 MATLAB 会按照线性索引的顺序来返回提取的元素，就类似于 A(:)命令可以将 A 中的所有元素按照线性索引的顺序返回一个列向量。

以后大家熟练的话，可以直接写成 A(A<=3)来提取 A 中所有小于等于 3 的元素。再举个例子，如果要提取 A 中大于等于 5 且小于 8 的元素，我们可以使用 A(A>=5 & A < 8)，千万不能写成 A(5 <= A < 8)，也不能使用&&运算符：A(A>=5 && A < 8)，&&运算符只能用于标量的逻辑运算。

注意，这里有一个易错点：**L 必须是逻辑矩阵，即里面的 0 和 1 都必须是逻辑值，不能是由数字 0 和 1 构成的数值矩阵。如果 L 是数值矩阵，可以使用 logical 函数进行转换。**

练习题：清风班上有 20 名同学，这 20 名同学的编号为数字 1 至 20。现在清风老师要随机请一些同学去吃饭，每名同学被抽中的概率都是 50%，请帮助清风老师随机抽取这些同学（学过概率论的同学应该能够看出来，每名同学是否能被选中均服从一个伯努利分布，就类似于独立的投 20 次篮，每次投中的概率都是 0.5）。

答案如下：

```
A = 1:20;  % 这 20 名同学的编号为 1,2,……,20
x = randi([0,1],1,20);  % 随机生成 20 个由 0 和 1 构成的随机数
L = logical(x);  % 转换成逻辑值
A(L)  % 使用逻辑向量 L 引用 A 中的元素
```

为什么上面的答案可行？第二行代码随机生成了一个包含 20 个元素的行向量 x，向量 x 中的每个元素都是随机生成的 0 或 1，其中 0 代表同学没有被选中，1 代表同学被选中。由于 randi 函数用来生成随机的均匀分布的整数，因此出现 0 和 1 的概率相同，这就满足了每名同学被抽中的概率都是 50%。第三行代码使用 logical 函数将 x 这个数值向量转换成了逻辑向量，这样才能通过逻辑值对向量元素引用。大家可以自己尝试，如果不使用 logical 函数进行转换的话，MATLAB 就会报错：数组索引必须为正整数或逻辑值。

除了以上方法外，还有一种更为通用的方法，不妨假设每名同学被抽中的概率都是 p，其中 0≤p≤1，那么我们可以借助下面的代码来抽取这些同学：

更通用的答案：

```
p = 0.5;  % 每名同学被抽中的概率
A = 1:20;  % 这 20 名同学的编号为 1,2,……,20
r = rand(1,20);  % 随机生成 20 个在 0 至 1 区间上均匀分布的随机数
L = (r < p);  % r 和 p 比较大小
A(L)  % 使用逻辑向量 L 引用 A 中的元素
```

在更通用的答案中，我们通过比较由 rand(1,20)生成的、位于[0, 1]区间上的 20 个随机数 r 与每名同学被抽中的概率 p 来构建逻辑数组 L。对于每一个元素 r(i)，如果 r(i) < p，则 L(i)为逻辑值 1，表示第 i 个同学被抽中；反之，如果 r(i) >= p，则 L(i)为逻辑值 0，表示第 i 个同学未被抽中。最终，逻辑数组 L 的逻辑值 1 和逻辑值 0 会被用来从原始数组 A 中提取元素，即 A(L)会返回所有被抽中同学的编号。

下面我们再来看一个练习题：随机生成 200 名同学的考试得分(0-100 整数)，统计得分大于等于 60 分的同学人数，并计算得分位于区间[60,80]内的所有同学的平均分。

答案如下：

```
A = randi([0,100],200,1);   % 随机生成 200 名同学的分数
sum(A>=60)     % 大于等于 60 分的人数
tmp = A(A<=80 & A >=60);   % 区间[60,80]内的所有分数
mean(tmp)   % 计算平均值
```

3.4.4.3 使用逻辑值修改或删除矩阵元素

上一小节我们介绍了利用逻辑值引用矩阵的元素，我们也可以对引用的元素进行修改或删除。下面我们直接看例子：

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

从上表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。

我们再来看一个有趣的问题：**缺失值的识别和填补**。

举个例子：假设清风老师要连续一周测量早上 6 点室外的温度，结果清风老师周二和周五睡过了头，那两天的温度没有测量，剩下五天的温度分别是 10°、5°、2°、8°和 5°，周二和周五的温度成了缺失值。现在清风老师想利用有数据的剩余五天的平均气温来代替周二和周五这两天的温度，于是他计算出这五天的平均气温为（10+5+2+8+5）/5=6°，这时候就完成了对缺失值的填补。

那么我们怎样在 MATLAB 中实现这个过程呢？我们可以先定义一个向量 A 用来保存这一周的温度：A = [10 NaN 5 2 NaN 8 5]，其中第二个元素和第五个元素为 NaN，代表周二和周五的温度数据是缺失的。现在需要大家将 A 中所有的 NaN 值替换成所有非缺失值的平均值。

答案只需要一行代码：A(isnan(A)) = mean(A(~isnan(A)))。这里用到了 **isnan 函数，它可以判断数组中的元素是否为不定值 NaN，并返回一个和输入的数组大小相同的逻辑数组。**

例如，这里的 isnan(A)返回的结果就是[0 1 0 0 1 0 0]这个逻辑向量。

有同学会想：为什么不直接用命令 A==NaN 来找 A 中的缺失值？这是因为在 MATLAB 中，NaN 相互之间不相等，运行 NaN == NaN 会输出逻辑值 0。

那么，如何找出 A 中所有非缺失值的元素呢？我们可以对 isnan(A)的结果进行“逻辑非”运算，即~isnan(A)，然后再利用这个逻辑向量对 A 进行索引：A(~isnan(A))。

详细的介绍大家可以看本书的配套视频。后续章节中我们会更系统地介绍缺失值的知识点，现在只是小试牛刀。

3.4.4.4 all、any 和 find 函数

下面我们介绍三个非常重要的函数，它们的作用请看下表：

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

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）

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
[0 2 -1 2; -3 0 1 -2; -2 -2 -2 3]	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

可以看出，all 函数相当于对向量或者矩阵的元素进行‘逻辑与&’运算，只有全为非零值时才返回逻辑值 1。而 any 函数则相当于对元素进行‘逻辑或|’运算，存在至少一个非零值时就会返回逻辑值 1。

我们来看 any 函数的例子：

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
[0 0 -1 2; -3 0 1 -2; 0 0 0 0]	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

事实上，all 函数和 any 函数很少直接运用在数值矩阵上，它常常配合逻辑矩阵来实现特定的功能。我们来看下面的练习题：

（1）请随机生成一个 100 行 3 列的矩阵，用来记录学生的考试成绩：矩阵每一行代表一名同学，每一列代表一门科目的成绩，矩阵中的每个元素都是区间[50,100]内的随机整数。

```
score = randi([50,100],100,3)
```

（2）三门科目的成绩都不低于 85 分的同学可以获得奖学金评选的资格，请指出哪些同学可以获得资格。要求返回一个包含 100 个元素的逻辑向量，元素为逻辑值 1 的位置对应的同学有评选资格。

```
all(score >= 85,2)
```

（3）请指出哪些同学挂科了，至少有一门科目没过 60 分就算挂科。要求返回一个包含 100 个元素的逻辑向量，元素为逻辑值 1 的位置对应的同学挂科了。

```
any(score < 60,2)
```

（4）这三门科目中是否存在科目没有人挂科（所有同学的这一门科目的成绩都高于 60 分）。要求返回一个包含 3 个元素的逻辑向量，元素为逻辑 1 的位置对应的科目表示没有人挂科。

```
all(score >= 60)
```

大家应该注意到了，上面问题的答案有一点冗余。例如第二问我们关心的是哪些同学可以获得评选资格，但是答案返回的结果是一个长度为 100 的逻辑向量，向量中也包含了没有获得评选资格的同学，他们用逻辑值 0 表示。那么有没有一种方法能够找到这个向量中所有非零元素呢？find 函数可以帮助我们实现！下面是 MATLAB 官方文档对于 **find 函数**的介绍：

说明

k = find(X) 返回一个包含数组 X 中每个非零元素的线性索引的向量。

- 如果 X 为向量，则 find 返回方向与 X 相同的向量。
- 如果 X 为多维数组，则 find 返回由结果的线性索引组成的列向量。

k = find(X,n) 返回与 X 中的非零元素对应的前 n 个索引。

k = find(X,n,direction) (其中 direction 为 'last') 查找与 X 中的非零元素对应的最后 n 个索引。direction 的默认值为 'first'，即查找与非零元素对应的前 n 个索引。

[row,col] = find(___) 使用前面语法中的任何输入参数返回数组 X 中每个非零元素的行和列下标。

[row,col,v] = find(___) 还返回包含 X 的非零元素的向量 v。

默认情况下，find 函数会返回所有非零元素的索引，如果只给 find 函数一个返回值，那么会返回所有非零元素的线性索引；如果给两个返回值，那么会返回非零元素对应的行和列下标；如果给三个返回值，那么还会返回非零元素构成的向量。另外，大家也可以指定返回前 n 个非零元素的索引，只需要给定第二个输入参数 n，此时会返回前 n 个非零元素的索引，如果要返回后 n 个非零元素的索引，那么需要使用 find(X,n,'last')。

X	命令	结果
[4 0 11 -3 0 2 -4]	ind = find(X)	1 3 4 6 7
同上	[r, c] = find(X)	r = 1 1 1 1 1 c = 1 3 4 6 7
[0 -1 0 0; 4 0 0 3; 0 1 0 0]	ind = find(X)	2 4 6 11 % 线性索引的知识大家可以参考本章 3.3.2 节：矩阵元素的引用
同上	ind = find(X, 2)	2 4
同上	ind = find(X, 2, 'last')	6 11
同上	[row,col] = find(X)	row = 2 1 3 2 col = 1 2 2 4
同上	[row,col,v] = find(X)	row = 2 1 3 2 col = 1 2 2 4 v = 4 -1 1 3

回到上一页练习题的第二个问题，我们可以使用下面的命令对代码进行改进，这样就可以返回获得资格的同学的索引：

```
tmp = all(score >= 85,2);
ind = find(tmp)
```

练习题（接着上题来）：

（5）找出恰好挂了两门科目的同学的编号。

```
tmp = sum(score < 60,2);  % 每位同学挂科的数目
find(tmp == 2)
```

（6）找到总分超过 260 分的同学的编号。

```
total_score = sum(score,2); % 计算每位同学的总分
find(total_score > 260)
```

3.4.5 集合运算

集合是指具有某种特定性质的具体的或抽象的对象汇总而成的集体。其中，构成集合的这些对象则称为该集合的元素。集合中的元素具有三个性质：确定性（给定一个集合，任给一个元素，都可以确定该元素是否属于该集合）、互异性（任何两个元素都是不相同的）和无序性（每个元素的地位都是相同的，元素之间是没有顺序的）。

学过 Python 的同学应该知道，Python 中可以使用 set 函数或者大括号来创建集合。然而在现版本的 MATLAB 中，还没有正式推出集合这个数据结构。目前 MATLAB 使用普通的数组来模拟集合的功能，但大家要清楚，数组中的元素可以相同而且是有先后顺序的，因此不满足集合中元素的互异性和无序性，因此数组并不能当成真正的集合。MATLAB 官网列举了一些集合运算相关的函数，官方是这样介绍集合运算的：*集合运算比较两个集合中的元素，以找出共性或差异*。大家一定要注意，官方所说的集合要当作普通的数组来理解，用户可以通过 MATLAB 提供的与集合运算相关的函数来实现集合中的大多数操作。

我们先通过表格来展示这一小节要学到的和集合运算相关的函数：

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

（1）unique 函数

unique 函数可用来提取数组中的唯一值，它可以用在我们学过的向量和矩阵上，也可以用在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 = 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，numel(A)≥numel(C)对任意的 A 都成立）

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

另外，unique 函数可以有最多三个返回值：[C, ia, ic] = 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] = 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。

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
A = [5 -6 5 10 8 -6 -3 -3]; C = unique(A, 'stable')	C = 5 -6 10 8 -3

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 6 8 3 5 4 1 2 8 2 5 4 1	1 2 3 4 5 6 8	2 5 2 8 3 5 4 1 6 8	3 5 6 8 4 1 2 8 2 5

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。

（2）ismember 函数

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

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。

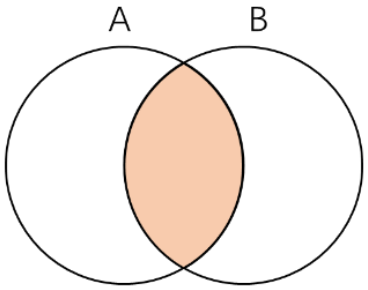
命令	结果
A = [4 1 3 4 8]; B = [3 7 4 5 4 9 6]; [h, ib] = ismember(A,B)	h = 1×5 logical 数组 1 0 1 1 0 ib = 3 0 1 3 0
A = [3, 1; 8, 5; 4, 0]; B = [3 7 4 5 4 9 6]; [h, ib] = ismember(A,B)	h = 3×2 logical 数组 1 0 0 1 1 0 ib = 1 0 0 4 3 0
A = [3,1; 8,5; 4,0]; B = [3 7 4 5; 1 4 9 6]; [h, ib] = ismember(A,B)	h =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

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。

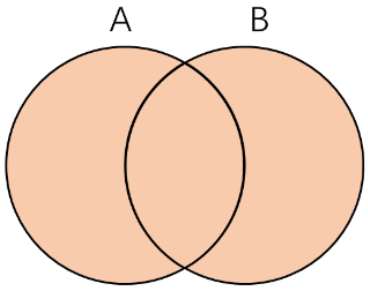
命令	结果
A = [6 8; 3 5; 4 1]; B = [1 2 3 5 3 8 6 1]; [h, ib] = ismember(A, B)	h = 3×2 logical 数组 1 1 1 1 0 1 ib = 4 7 2 6 0 1
A = [6 8; 3 5; 4 1]; B = [1 2 3 5 3 8 6 1]; [h, ib] = ismember(A, B, 'rows')	h = 3×1 logical 数组 0 1 0 ib = 0 2 0

（3）intersect、union、setdiff 和 setxor 函数

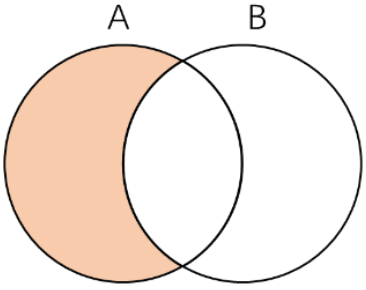
这四个函数分别用于计算两个数组之间的交集、并集、差集和对称差集，下面给出了这四个函数对应的维恩图（Venn diagram）。



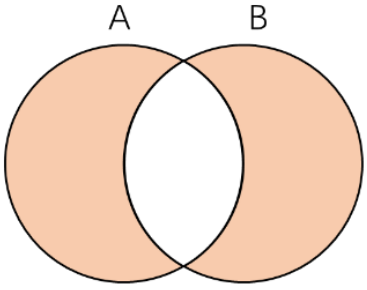
intersect(A,B) 返回 A 和 B 的交集，但是不包含重复项。



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。

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

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
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	C = 1　0　3 3　3　3
% A 和 B 同上 C = intersect(A, B, 'rows', 'stable')	C = 3　3　3 1　0　3

另外，intersect 函数可以有最多三个返回值，下面截图来自 MATLAB 官网：

[C,ia,ib] = intersect(___) 还使用上述任何语法返回索引向量 ia 和 ib。

- 一般情况下，C = A(ia) 且 C = B(ib)。
- 如果指定了 'rows' 选项，则 C = A(ia,:) 且 C = B(ib,:)。

通常情况下我们不会用到三个返回值，感兴趣的同学可以在 MATLAB 中进行测试。

剩余三个函数的用法我们举几个简单的例子，更多例子大家可以自己查看帮助文档学习。

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

3.5 线性代数相关的函数

本小节我们将介绍 MATLAB 在线性代数中的一些常用函数，以下内容大多仅涉及到初等线性代数的知识，大家可以查阅线性代数的书籍复习相关概念。

函数名	功能
det	计算方阵的行列式（determinant）
rank	计算矩阵的秩
trace	计算方阵的迹（对角线元素的和）
rref	将矩阵变换成行最简型矩阵（Reduced row echelon form 简化行阶梯形）
inv	计算方阵的逆矩阵（inverse matrix），inv(X)的结果和 X^(-1)相同
transpose	返回转置矩阵（有复数的话转置后虚部符号保持不变），transpose(A)和 A.'的结果相同
triu	返回矩阵的上三角部分，triangle(三角形) + upper(高、上方)
tril	返回矩阵的下三角部分，triangle(三角形) + lower(低、下方)
eig	计算方阵的特征值和特征向量（eigenvalue and eigenvector）
norm	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）

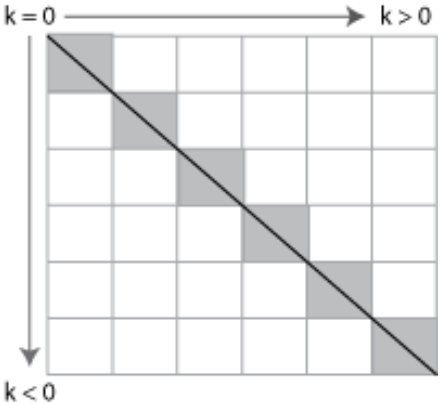
最前面的六个函数都很简单，我们直接来看例子：

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

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。



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

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。

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。

具体代码的解释如下：对于一个 n 阶的对称矩阵，它的上三角部分（不包括主对角线）有 n*(n-1)/2 个元素，上方代码中将其赋值给 num；矩阵 A 初始化为一个 n 阶全为 0 的矩阵；true(n)创建了一个 n 阶的逻辑矩阵，其中所有元素都为逻辑值 1；triu(true(n),1)返回了一个不包括主对角线的上三角的逻辑矩阵，我们用它对 A 矩阵进行逻辑索引； randi([0,9],num,1)生成了一个有 num 个元素的向量，其中每个元素都是位于区间[0, 9]中的随机整数；A(triu(true(n),1))选取了矩阵 A 的上三角部分（不包括主对角线），然后将其赋值为随机生成的整数；A'是矩阵 A 的转置，由于我们已经填充了 A 的上三角部分，将它转置我们可以确保下三角部分与上三角部分是对称的；diag(randi([0,9],n,1))生成了一个主对角线上的元素为区间[0, 9]中的随机整数，而其它地方都为 0 的 n 阶矩阵；将这三个矩阵相加，即可得到最终的对称矩阵。

以后我们学了循环语句后，还可以利用循环语句生成，虽然循环语句更直观且代码易于编写，但直接基于矩阵操作在 MATLAB 中通常更高效，尤其是对于大型矩阵。

eig 函数可用来计算方阵的特征值和特征向量，它有两种最基础的用法：

- `e = eig(A)` 返回一个列向量，e 中包含方阵 A 的所有特征值。
- `[V,D] = 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。

我们用 MATLAB 来验算下：

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

有同学会有疑惑，为什么 MATLAB 算出来的特征向量这么奇怪？特征值-1 对应的特征向量是[-0.7071; 0.7071]，特征值 3 对应的特征向量是[0.7071; 0.7071]。这是因为特征值对应的特征向量不是唯一的，前面可以乘以任意的非零常数。在 MATLAB 中，eig 函数会对特征向量进行缩放，使得各特征值对应的特征向量的模长均为 1。

再来看个例子：

命令	结果
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)	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

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。

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 = 5×5
   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 = 5×5
   -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
```

另外，MATLAB 计算得到的特征值或特征向量可能会有一定的误差，这是 MATLAB 的浮点数运算误差导致的，大家以后遇到了有印象即可。

下面我们来介绍最后一个函数：norm 函数，它可以用来计算向量或者矩阵的范数。范数这个概念大家可能没听过，它在机器学习中用的较多。

我们这里只介绍向量的范数，假设向量$x = [x_1, x_2, \cdots, x_n]$，常用的向量范数有下面三种：

1-范数：$\|x\|_1 = \sum_{i=1}^{n}|x_i|$，即对$x$的元素求绝对值后再求和。

2-范数：$\|x\|_2 = \sqrt{\sum_{i=1}^{n}x_i^2}$，即对$x$的元素求平方后再求和，然后再开方。

p-范数：$\|x\|_p = (\sum_{i=1}^{n}|x_i|^p)^{\frac{1}{p}}$,即对$x$元素的绝对值求$p$次方，再求和，最后再算$\frac{1}{p}$次幂。

在 MATLAB 中的调用方法分别为：norm(x, 1)、norm(x, 2)和 norm(x, p)。事实上，1-范数和 2-范数都属于 p-范数的特例，MATLAB 中 p 默认取 2，即 norm(x)是 norm(x, 2)的简写形式，表示计算 2-范数。

我们来看几个例子：

命令	结果
x = [2 -4 -4 0]; a1 = norm(x,1)	10
a2 = norm(x,2) % 等价于 a2 =norm(x)	6
a3 = norm(x,3)	5.1426

事实上，如果你不会 norm 函数，也可以用我们之前的方法计算向量的范数：p-范数等价于：sum(abs(x).^p)^(1/p)，这里的 1/p 要加括号。特别地，将 p 分别取 1 和 2 即可得到 1-范数和 2-范数。

拓展：范数和距离的联系

（1）欧式距离（Euclidean Distance）

欧式距离是最常见的两点之间距离定义的方法，又被称为欧几里得度量，它定义于欧几里得空间中。点$x = (x_1, x_2, \cdots, x_n)$和点$y = (y_1, y_2, \cdots, y_n)$之间的欧氏距离为：

$$d_{x,y}^{\text{欧氏距离}} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

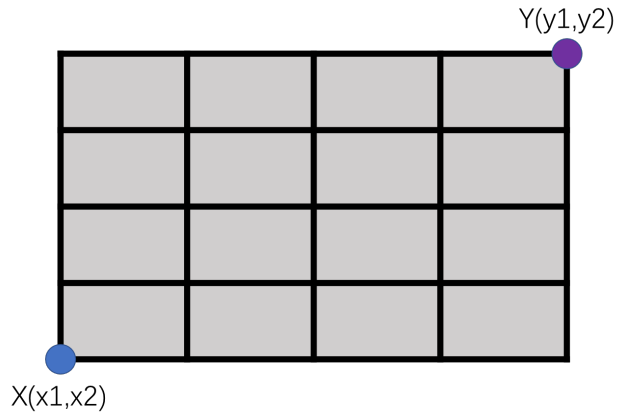
$$= \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

显然，把x和y视为两个向量，则$d_{x,y}^{\text{欧氏距离}} = \|x - y\|_2$.（将$x$和$y$向量作差后计算 2-范数）

例如$x(1, 2)$和$y(4, 6)$直接计算欧氏距离结果 5，将它们视为向量：$x = [1, 2]$，$y = [4, 6]$，那么作差后$x - y = [-3, -4]$，向量$[-3, -4]$对应的 2-范数也为 5，和欧氏距离的结果相同。

（2）曼哈顿距离（Manhattan Distance）

假设城市所有的道路是平行且垂直的，如下图所示，你需要驾车从 X 点到达 Y 点，那么你的驾驶距离应该如何计算？



你能直接计算 X 和 Y 两点之间的欧式距离作为你的驾驶距离吗？当然不行，除非你乘坐的飞机。这时候，我们可以使用曼哈顿距离来表示实际的驾驶距离。

点$x = (x_1, x_2, \cdots, x_n)$和点$y = (y_1, y_2, \cdots, y_n)$之间的曼哈顿距离定义为：

$$d_{x,y}^{\text{曼哈顿距离}} = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|$$

$$= \sum_{i=1}^{n}|x_i - y_i|$$

把x和y视为两个向量，则$d_{x,y}^{\text{曼哈顿距离}} = \|x - y\|_1$.（将$x$和$y$向量作差后计算 1-范数）

例如$x(1, 2)$和$y(4, 6)$直接计算曼哈顿距离结果 7，将它们视为向量作差后得到$[-3, -4]$，向量$[-3, -4]$对应的 1-范数也为 7，和曼哈顿距离的结果相同。

（3）闵可夫斯基距离 (Minkowski Distance)

闵可夫斯基距离（简称为闵氏距离）不是一种特定的距离，而是一组距离的定义。x和y之间的闵氏距离为：

$$d_{x,y}^{\text{闵式距离}} = (|x_1 - y_1|^p + |x_2 - y_2|^p + \cdots + |x_n - y_n|^p)^{\frac{1}{p}}$$
$$= \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

其中p是一个正数，当$p = 1$时就是曼哈顿距离，当$p = 2$时，就是欧氏距离。

把x和y视为两个向量，则$d_{x,y}^{\text{闵式距离}} = \|x - y\|_p$.（将$x$和$y$向量作差后计算 p-范数）

3.6 本章小节

本章我们主要介绍了 MATLAB 对于矩阵的相关操作，大家需要有线性代数这门科目的基础。

由于本章内容很多，我为大家准备了一套思维导图，大家可以照着思维导图来回顾本章的知识点。

（下载的方法大家可以看本书的第一页，在目录的下面）

本章涵盖了 MATLAB 中许多常用的内置函数。为了加深理解，我专门设计了一些题目，分为三个部分来帮助大家巩固和掌握本章的知识（**不要使用循环、判断等本章没学的命令**）：

- ✧ **基础篇：**帮助大家再次复习基础的内容，题目的答案大部分都能直接在本章的知识点中找到。
- ✧ **提高篇：**在本章课上练习题的基础上进行了一点变形，考验大家对于知识点的熟练程度。这部分的习题不算难，相信大多数的同学都能应对得了。
- ✧ **挑战篇：**这部分的习题有一定的难度，主要训练大家的编程思维。这部分的习题如果你能独立地做出来，那就说明你非常有编程天赋，大家可以尝试挑战一下。

3.7 课后习题

基础篇

Q1.填空题

（1） MATLAB 中矩阵的元素应包括在____括号中，矩阵的同行元素之间用_____分隔，行与行之间用_____分隔。

（2） 命令 0:3:10 生成的向量是_____；命令 10:-2: 5 生成的向量是_____；命令 10: 5 生成的向量是_____。

（3） A 是一个向量，要计算 A 中包含的元素个数，使用的命令是_____；如果 A 是一个矩阵，要计算 A 中包含的元素个数的命令是_____。

（4） A 是一个矩阵，命令_____可返回 A 的行数；命令_____可返回 A 的列数。

（5） 要生成一个包含 50 个元素的等差数列，数列的第一个数是 0，最后一个数是 2π，我们可以使用命令_____。

（6） 提取向量 A 中第三个位置的元素到最后一个位置的元素，可以使用命令_____；提取向量 A 第三个位置的元素到倒数第二个位置的元素可以使用命令_____。
（7） 提取矩阵 A 中奇数行的元素可以使用命令_____。
（8） 删除 A 矩阵的第二列和最后一列可以使用命令_____。
（9） A 矩阵的大小为 3×3，其线性索引为 4 的元素位于第____行第____列，如何使用代码得到这个结果：_____。
（10） 命令_____可以将矩阵 A 中的所有元素按照线性索引的方式重构成一个列向量。
（11） 命令_____可以生成一个大小为 3×4 的全为 0 的矩阵；_____可以生成一个大小为 4×4 的全为 1 的矩阵；_____可以生成一个大小为 3×3 的单位矩阵。
（12） 命令_____用来创建一个 100 行 2 列的随机矩阵，矩阵中的每个元素都在区间 0 和 1 内均匀分布。
（13） 要模拟投掷有 6 个面的均匀骰子 100 次，那么我们可以使用命令_____得到一个长度为 100 的行向量，向量中的每个元素都是随机的取自[1,6]中的整数。
（14） 命令_____用来创建一个 3 行 3 列的随机矩阵，矩阵中的每个元素都随机取样自标准正态分布。
（15） 如果 A 是一个向量，那么命令 diag(A)的作用是_____；如果 A 是矩阵，则作用是_____。
（16） 如果矩阵 A 和 B 的行数相同，命令_____可以对 A 和 B 横向拼接；如果矩阵 A 和 B 的列数相同，命令_____可以对 A 和 B 纵向拼接。
（17） 命令 repelem(1:3,1:3)得到的结果是_____。
（18） 将一个向量 a 沿着行方向进行重复的堆叠 5 次，可以使用命令_____。
（19） 矩阵 A 大小是 4 行 5 列，将其形状变成 2 行 10 列可以使用命令_____。
（20） 命令_____可以将矩阵 A 进行左右翻转，命令_____可以将 A 上下翻转。
（21） 要将矩阵 A 顺时针（注意不是逆时针）旋转 90 度可以使用命令_____。
（22） 将向量 v 按照从大到小的顺序进行降序排列可以使用命令_____。
（23） 计算 A 矩阵每一行的最大值的命令为_____。
（24） 计算累积和、乘积、差分的函数分别是_____。
（25） A 是一个方阵，A^3 的作用是_____，A.^3 的作用是_____。
（26） MATLAB 中关系运算符有六个，分别是_____。
（27） 运算优先级非常重要，我们可以通过_____来改变运算的先后顺序。
（28） 要返回数组中出现的唯一值，我们可以使用函数_____。
（29） 判断一个数组的元素是否在另一个数组内，我们可以使用函数_____。
（30） 要计算方阵 A 的特征值和特征向量，命令为_____。
（31） 交集、并集、差集和对称差集的四个函数分别为_____。
（32） 返回矩阵的上三角部分和下三角部分的函数分别为_____。
（33） 将一个普通数组转换成逻辑值数组的函数是_____。
（34） 判断数组中的元素是否为不定值 NaN 的函数是_____。
（35） 计算方阵 A 的逆矩阵的命令为_____。
（36） 对 A 矩阵的所有元素求倒数的命令为_____。
（37） 命令 0 == 0 == 0 返回的结果为_____；命令-1 < 0 < 1 返回的结果为_____。
（38） xor(3,4)返回的结果为_____。
（39） 计算矩阵的行列式、秩和迹的函数分别为_____。
（40） 计算向量 x 的 1-范数、2-范数和 p-范数的命令为_____。

Q2.请完成下面的一系列任务：

（1） 生成一个 6 行 3 列的随机矩阵 A，矩阵中每个元素都是位于区间[50,100]之间的随机整数，下面我们假设矩阵 A 的每一行代表一名学生，这六名同学的三门科目的成绩对应着三列；

（2） 将第一门科目六名同学的成绩赋值给变量 B，对 B 进行降序排列，排序后的向量记为 BB，并返回 BB 中的每个元素在 B 中的索引向量 ind；

（3） 计算 A 中所有成绩的自然对数；

（4） 请基于第二科的成绩按升序对这六名同学进行排序，当第二科成绩相同时，请保持其在矩阵中出现的先后顺序；

（5） 计算六名同学的总分以及每门科目的平均分；

（6） 计算每门科目的最低分，并返回是第几位同学取得的分数；

（7） 计算每名同学在哪门科目上分数最高、并返回最高分；

（8） 假设这三门科目的权重分别是 0.2, 0.5, 0.3，请计算每名同学的加权平均分（即三门科目的成绩分别乘以对应的权重，然后再求和）；

（9） 判断这三门科目是否有同学不及格（低于 60 分），如果有任意一名同学在某个科目中不及格就返回逻辑值 1，否则返回逻辑值 0；

（10） 统计每门科目不及格的人数；

（11） 将 A 中低于 60 分的成绩全部改成 60 分；

（12） 在上一问的基础上重新计算六名同学的总分，并找出总分最高的同学。

Q3.简答题

（1） 如何将一个向量倒序？例如原来的向量是[1 5 8 4]，倒序后是[4 8 5 1]。原向量的方向（行向量和列向量）会影响你的结果吗？（本题答案有至少两种方法）

（2） 请说明&&和&的用法和主要区别。

（3） MATLAB 中浮点数的计算可能存在误差，应该怎样判断两个浮点数相等？

（4） 简述 MATLAB 中*和.*的作用，并介绍加法运算中所支持的几种兼容模式。

（5） 简述 sum、mean、median 这三个函数使用方法的共同特点，它们和 max 函数在沿行和列方向计算时的命令有什么区别？

提高篇

Q1.如果 x 是一个常数，A 是一个矩阵，请给出至少两种方法来判断 A 矩阵中是否存在 x？存在则返回逻辑值 1，否则返回逻辑值 0.

Q2.生成包含 3 个元素的行向量 A，A 中每个元素都是位于 1-10 之间的随机整数；再生成包含 10 个的元素的行向量 B，B 中的每个元素也都是位于 1-10 之间的随机整数。

（1） A 中的这三个元素哪些出现在 B 中？请返回下表中的结果列。例如：

A	B	结果	解释
[3 10 6]	[2 6 6 7 10 7 9 4 4 1]	2, 3	A 中的第二个元素和第三个元素出现在了 B 中
[5 4 6]	[7 10 8 8 2 2 9 3 5 9]	1	A 中只有第一个元素出现在了 B 中
[9 8 7]	[1 6 5 4 4 10 3 1 10 2]	[ ]	A 中没有元素出现在了 B 中

（2） A 中每个元素是否都包含在 B 中？请返回下表中的结果列。例如：

A	B	结果	解释
[2 7 6]	[2 6 6 7 10 7 9 4 4 1]	逻辑值 1	A 中的所有元素都在 B 中能找到
[3 5 6]	[1 8 8 4 2 7 9 3 5 9]	逻辑值 0	A 中的 6 在 B 中不能找到

Q3.如何创建一个包含 n 个元素的等比数列，其第一项为 a，最后一项为 b。例如 a=2, b=1024, n=10 时，创建的等比数列为：[2 4 8 16 32 64 128 256 512 1024].

Q4.命令 rand(1)可生成一个位于 0 和 1 之间均匀分布的随机数，那么 10*rand(1)生成的随机数所在的范围是___和___之间；2+3*rand(1)生成的随机数所在的范围是___和___之间。考虑一般情况，命令_____可以创建一个 m 行 n 列的随机矩阵，矩阵中的每个元素都在区间[a, b]上(a<b)均匀分布。

Q5.本题需要用到概率论中正态分布的一个性质：假设$X \sim N(u, \sigma^2)$，且a, b是两个实数，那么$aX + b \sim N(au + b, (a\sigma)^2)$，MATLAB 中的_____函数可生成标准正态分布的随机数，其均值为 0 方差为 1。利用上面的性质，命令_____可以创建一个 m 行 n 列的随机矩阵，矩阵中的每个元素都服从均值为p,方差为q^2的正态分布。

Q6.A 是一个包含至少两个元素的向量，请你判断 A 中是否所有元素都互不相同。例如：A = [6 5 0 3 6 2]时返回逻辑值 0，因为里面包含了重复的元素 6；A = [1 2 0 4 -2 3 7]时返回逻辑值 1，因为所有的元素都互不相同。（你能给出两种甚至更多种解答方法吗?）

Q7.找到向量 x 中最大值对应的位置索引，如果有多个最大值，将它们的索引全部返回。例如 x = [2 4 8 1 4 8 3]，那么你需要返回[3, 6]，因为第 3 个位置和第 6 个位置都是最大值 8。

Q8.给出一个实矩阵 A，请判断 A 是否为对称矩阵。（实矩阵是指矩阵中的每一个元素都是实数，不含复数；若 A 和 A 的转置相同，则 A 为对称矩阵）

Q9.给定两个同型方阵 A 和 B，请判断 A 和 B 是否互为对方的逆矩阵。（提示：如果 A*B 或者 B*A 的结果是单位矩阵，则 A 和 B 互逆）

Q10.对于任意一个实对称矩阵 A，判断 A 是否为正定矩阵。（提示：如果实对称矩阵 A 的所有特征值都大于 0，那么这个实对称矩阵是正定矩阵）

Q11.怎样将一个方阵 A 的主对角线元素重新赋值为 0。例如：

原来的 A：				现在的 A：			
3	3	3	8	0	3	3	8
4	8	7	7	4	0	7	7
2	1	9	8	2	1	0	8
5	3	7	9	5	3	7	0

Q12.下面是斐波那契数列的通项公式，请使用该通项公式计算 n=1,2,⋯,10 的前 10 项。

$$a_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Q13.拉马努金是印度历史上最著名的数学家之一，他没有接受过正规的数学教育，但有着令人惊异的数学天赋，似乎可以感知到大量数字关系背后的规律。请你验证他发现的下面两个等式是否成立（考虑浮点数计算的误差）。

（1）$\sqrt{\sqrt[3]{28}-3} = \frac{\sqrt[3]{98}-\sqrt[3]{28}-1}{3}$

（2）$-\sqrt[3]{\cos\left(\frac{\pi}{9}\right)} + \sqrt[3]{\cos\left(\frac{2\pi}{9}\right)} + \sqrt[3]{\cos\left(\frac{4\pi}{9}\right)} = \sqrt[3]{\frac{3\sqrt[3]{9}-6}{2}}$

Q14.下面是拉马努金发现的计算圆周率的公式，请你计算等式右侧级数前三项的和，并将等式右侧取倒数来计算一个近似圆周率. (提示：factorial 函数也可用来计算阶乘)

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{99^2}\sum_{k=0}^{\infty}\frac{(4k)!}{(k!)^4}\frac{26390k+1103}{396^{4k}}$$

Q15.清风老师每年年初都会在银行存 1000 元私房钱，银行每年支付 2%的利息。

（1）请分别计算第 1 年、第 2 年和第 3 年的年末清风老师银行账户的余额。

（2）根据上一小问的规律，尝试使用一行代码计算清风老师前 10 年每一年年末的银行账户余额。

Q16.在机器学习算法中，我们经常会将一些数据进行特征缩放(Feature Scaling)来加快算法的收敛速度。下面我们介绍特征缩放中使用最多的两个方法：

方法 1：Min-Max Scaling：$x^* = \frac{x-x_{\min}}{x_{\max}-x_{\min}}$，其中$x_{\min}$和$x_{\max}$分别是这一组数据的最小值和最大值。可以看到，该公式对原始数据进行线性变换，使结果映射到[0, 1]的范围，来实现对原始数据的等比缩放。因此，该方法中文常被翻译为“线性函数归一化”。

方法 2：Z-score Normalization：$x^* = \frac{x-u}{\sigma}$，这里的u和σ分别是这一组数据的均值和标准差（通常为样本标准差），该方法常被翻译为“Z-score 标准化”。

以方法 1 给大家举例，假设原来的数据为[3 5 6 2 1]，最小值为 1，最大值为 6，因此将数据中的每个数都减去最小值 1，然后再除以 5(由 6-1 计算得到)，就能够得到特征缩放的结果：[0.4 0.8 1 0.2 0].

请大家解决下面两个问题：

（1） 生成包含 30 个元素的随机列向量 a（a 中每个元素都是 0 到 100 间均匀分布的随机数），分别使用上面两种方法对 a 进行特征缩放；

（2） 生成一个 30 行 3 列的随机矩阵 A（A 中第一列每个元素都是 0 到 100 间均匀分布的随机数，第二列和第三列每个元素都是均值为 50，标准差为 20 的正态分布随机数），分别使用上面两种方法对 A 的每一列进行特征缩放，得到的结果是一个和 A 大小相同的矩阵。

挑战篇

Q1.A 是一个矩阵，A(:)可以按照线性索引的顺序（沿着行方向依次遍历各列元素）将 A 中所有元素重构成一个向量，请问怎样沿着列方向依次遍历各行将 A 中的所有元素重构成一个向量。例如，$A=\begin{bmatrix}1 & 4 & 2\\ 3 & 6 & 8\end{bmatrix}$，需要输出的向量为[1 4 2 3 6 8].（输出的结果为行向量和列向量都可以）

Q2.利用 MATLAB 模拟随机丢一枚骰子 N 次，骰子有均匀的六个面。

（1） 取 N = 6000，将这 6000 次的结果保存到向量 A 中

（2） 统计 A 中出现 6 点的次数，记为 n

（3） 利用公式$p=\frac{n}{N}$计算 6 点出现的频率，并计算频率与实际概率之间的偏差：bias=$\left|p-\frac{1}{6}\right|$

（4） 请百度“伯努利大数定律”这个概念，这是概率论与数理统计中的知识点。如何利用本题来直观的说明伯努利大数定律的成立。

Q3.清风班上有 20 名同学，这 20 名同学的编号分别是 1、2、…、20。假设现在清风老师要随机请班上同学去吃饭、唱歌或者去看电影，已知每个同学被抽中去吃饭的概率是 30%，被抽中去唱歌的概率为 20%，剩下 50%的概率是去看电影。请帮助清风老师随机抽取这些同学，并分别返回吃饭、唱歌和看电影的同学的编号。

Q4.接上一题的题干，请学过概率论与数理统计的同学接着做本题。将上一题中抽取同学去吃饭作为一个随机事件，记随机变量 X 为每次抽出来去吃饭的同学人数，显然：随机变量 X 的期望值为 6 人，即用 20 乘以 0.3 计算得到。请大家验证辛钦大数定律的成立。提示：将上述抽取同学去吃饭这个随机事件重复 N 次（N 取得尽量大一点，例如 1000 次），并求出这 N 次独立事件抽出来的去吃饭的同学人数的平均值，计算该平均值和真实的期望的偏差有多大，增加 N 会出现怎样的情况。

Q5.**（这道题非常重要！）**MATLAB 中有一个非常有用的函数：randperm 函数，它能够将一个数字序列进行随机打乱。它有两种常见的用法：

用法 1：randperm(n)可以将向量 1:n 中元素的顺序随机打乱，生成一个长度仍为 n 的新向量，所有可能出现的情况共 n!种（全排列）。例如，当你运行 randperm(4)时，你可能得到[1 4 3 2]，也可能得到[3 2 4 1]。

用法 2：randperm(n,k)表示从打乱的 1:n 序列中随机的选择 k 个数出来，显然这 k 个数都不相同，且 k 要小于等于 n。例如，当你运行 randperm(10,3)时，你可能得到[5 3 10]，也可能得到[6 5 8]。**（randperm 函数在后续章节也会经常用到）**

请回答下面的问题**（做不出来的同学可以看本书的最后一页，有讲解视频）**：

（1） 根据上面的介绍，请你在 MATLAB 中测试 randperm 函数的功能。特别地，如果 n 是负数或者小数会出现怎样的情况？如果 k 大于 n 会出现怎样的情况？

（2） 假设一个商品推销员要去 10 个不同的城市推销商品，该推销员随机选择一个城市出发，依次经过其他所有的城市后，回到出发的城市（中途经过的城市不重复），为了方便，这 10 个城市就用数字 1 至 10 表示。请你为该推销员随机的生成一条路线（例如 2 10 8 9 1 5 7 6 3 4 2）。

（3） 使用代码模拟下列场景：假设你是一名数学老师，你正在给同学们讲不定积分的计算。这时候你的 PPT 上出现了你备课时准备的 4 道练习题，你需要随机抽取 4 名幸运同学到黑板上进行计算。已知你的班上共有 50 名同学，他们的学号分别是 2023001 至 2023050，你在 MATLAB 中运行了你写的这个程序，这四名同学的学号在 MATLAB 中被随机地抽取出来。

（4） 假设某公司在年会上设置了抽奖环节。主办方准备了一个抽奖用的不透明盒子，盒子内有 10 张奖券，其面值分别为[1 2 5 10 20 50 100 200 500 1000]，每名员工从中随机地抽取 3 张，将这 3 张奖券的面值相加就是他能获得的现金奖励。请设计一个程序，模拟清风老师在该抽奖环节中抽取一次能获得多少钱。

（5） 一副扑克牌有 54 张，其中大王和小王各一张，A,2,3,4,5,6,7,8,9,10,J,Q,K 各有 4 张。假设我们不考虑桃杏梅方这四种花色，请你设计一个随机的发牌程序，为地主发 20 张牌，两个农民各发 17 张牌。为了方便，A,2,3,4,5,6,7,8,9,10,J,Q,K 分别用数字 1 至 13 代替，小王用 14 代替，大王用 15 代替。进一步地，请你判断地主的牌是否有炸弹（有炸弹是指手上有双王或者有四张相同的牌例如 4 张 3）？

Q6.最近短视频上有一个有趣的街头抽奖游戏，规则如下：摆摊的店家准备了 24 个大小相同的玻璃球，其中红黄蓝各 8 个装进一个不透明的袋子里。玩家从袋子中随机的抓出 12 个球，然后计算每种颜色球的个数，颜色数量多的球放在前面。比如 5 个红色 4 个蓝色 3 个黄色，这样就属于 543。玩家抓出的 12 个球的颜色分布情况一定在下表所示的 13 种情况中。假设参与这个游戏是免费的，如果抽中了相应的情况，店家需要向玩家支付表中第二行所对应的金额。注意，除了 543 这一种情况玩家要赔给店家 30 元外，其余的 12 种情况玩家都是赚钱的，如果你运气好抓到了 840 这种情况（例如抽出了 8 个黄球 4 个红球），你可以赚 300 元。请模拟这个游戏，代码应返回你每次玩这个游戏获得的金额。多次运行代码，观察你是否能赚钱，你能从理论上解释原因吗？

情况	840	831	822	750	741	732	660	651	642	633	552	543	444
金额	300	200	200	80	60	15	100	10	5	5	5	-30	5

Q7.层次分析法是数学建模中一个常用的模型，它主要用于解决评价类问题，也可以为评价体系中的指标确定权重。在层次分析法中会用到判断矩阵，判断矩阵具有以下三个特点。特点 1：它是一个方阵，即行数和列数相同；特点 2：它的每个元素只能是 1,2,⋯,9 或者它们的倒数；特点 3：若它是一个 n 阶的方阵，那么对于任意的$i, j = 1,2,\cdots,n$，均满足$a_{ij} \times a_{ji} = 1$.

现在给你一个矩阵 A，请判断 A 是否符合层次分析法中判断矩阵的三个特点。

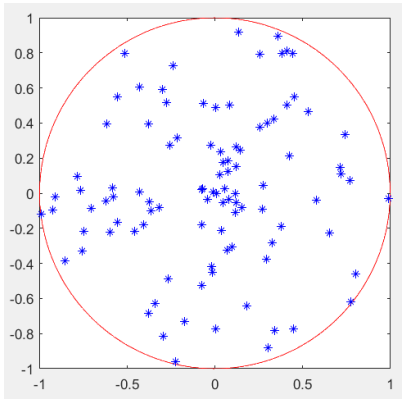
Q8.在上一题的基础上，请写出一段代码能够随机生成一个 n 阶（例如 n=5）的判断矩阵，该判断矩阵要满足上一问中的三个特点。

提示：（1）判断矩阵的主对角线元素一定是 1，为什么？（2）本章 3.5 节中我们学会了如何生成一个随机的对称矩阵，一个 n 阶的对称矩阵的上三角部分需要生成 n*(n-1)/2 个随机数，你需要思考本题中这个 n 阶的判断矩阵需要生成多少个随机数？

Q9.生成 100 个随机的点，这些点都位于单位圆内：$x^2 + y^2 \leq 1$。注意，要求返回两个长度为 100 的向量 x 和 y，x 和 y 分别表示这 100 个点的横坐标和纵坐标。

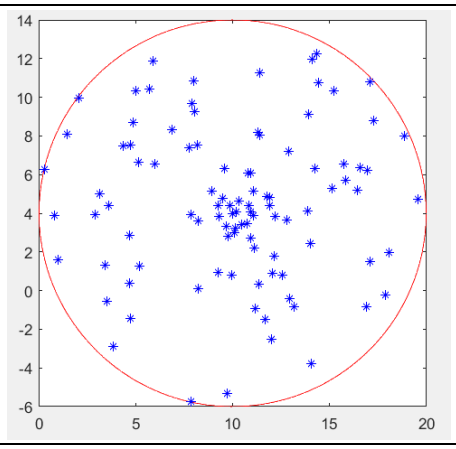
提示：我们可以借助圆的极坐标公式来生成随机数。下方我为大家准备了一段绘图的代码，大家将自己生成随机数的核心代码放入答题区域内，然后运行整段程序，就会出现类似于右侧的图形。（注意：没有要求大家生成的随机点在圆内均匀分布）

```
close all
theta = 0:0.01:2*pi;
xx = cos(theta);
yy = sin(theta);
plot(xx,yy,'r')   % 画一个单位圆
hold on
axis equal
axis([-1 1 -1 1])
% 答题区域---------开始

% 答题区域---------结束
plot(x,y,'*b')
hold off
```



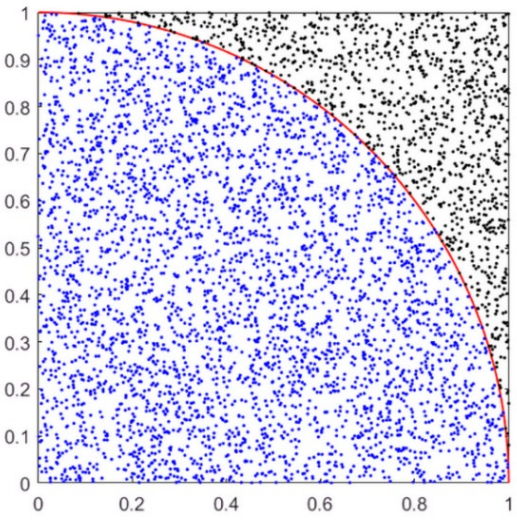
Q10.和上一题题干类似，请生成 100 个随机的点，这些点都位于圆内（不要求在圆内均匀分布）。其中，圆心的坐标为(10,4)，半径为 10。

```
close all
cc = [10,4];   % 圆心
radius = 10;   % 半径
theta = 0:0.01:2*pi;
xx = cc(1) + radius * cos(theta);
yy = cc(2) + radius * sin(theta);
plot(xx,yy,'r')
hold on;   axis equal
axis([cc(1)-radius cc(1)+radius cc(2)-radius cc(2)+radius])
% 答题区域---------开始

% 答题区域---------结束
plot(x,y,'*b')
hold off
```



Q11.蒙特卡罗模拟是一种以概率和统计理论为基础的计算方法，它能通过随机数来解决许多计算问题。蒙特卡罗模拟将所求解的问题同一定的概率模型相联系，用计算机实现统计模拟或抽样，以获得问题的近似解。本题将利用蒙特卡罗模拟计算圆周率π。

如右图所示，在边长为 1 的正方形内随机地生成 N 个随机点（图中 N 取的是 5000），由于每一个随机点都有横纵坐标，因此我们可以将它们绘制在图中，并用图中的小点表示。如果小点落在半径为 1 的 1/4 圆的范围内，其颜色为蓝色；否则颜色为黑色。因此蓝色小点的个数和黑色小点的个数加起来恰好等于 N。我们不妨记蓝色小点的个数为 n，那么根据蒙特卡罗模拟的思想，$\frac{1/4\text{ 圆的面积}}{\text{正方形的面积}} \approx \frac{\text{蓝色小点的个数}}{\text{所有点的个数}}$，替换成数学符号则有以下的关系式成立：$\frac{\pi/4}{1} \approx \frac{n}{\mathrm{N}}$，将其变形后可得：$\pi \approx \frac{4n}{\mathrm{N}}$。



有同学可能会好奇为什么这个关系式会成立，实际上这里用到了概率论与数理统计中几何概型和大数定律的思想，我们这里不深入探究。

现在请大家使用 MATLAB 生成 N 个随机点（N 取 1000），每个随机点的横纵坐标都分别在区间[0,1]上均匀分布，接下来统计出位于 1/4 圆内的随机点的数量 n，然后套用上面的公式计算出一个近似的圆周率。将你的 N 分别变成之前的 10 倍、100 倍和 1000 倍，你计算出来的圆周率的精度有何变化？

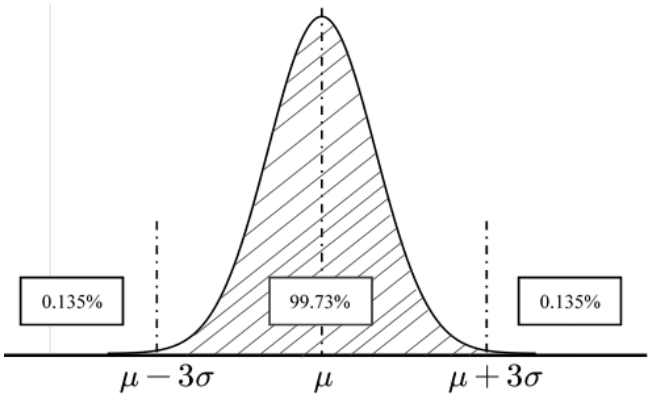
Q12.学过不定积分的同学应该知道，有一些不定积分的原函数无法用初等函数表示，例如$\int \frac{x}{e^x-1}dx$，对于这种类型的不定积分，如果我们要计算其在特定区间的定积分，只能通过某些数值分析的方法，例如梯形法或辛普森积分法。蒙特卡罗模拟也能帮助我们计算定积分，其基本的思路类似于上一题。现在请大家根据上一题的求解步骤来计算定积分$\int_0^2 \frac{x}{e^x-1}dx$, 在此之前，你可以先在草稿纸上画出被积函数在积分区间上的草图。注意：一个较为精确的答案是 1.2139，大家可以自己对对答案看看你计算的精度如何。

Q13.清风订了一份报纸，送报人可能在早上 6：30 至 7：30 之间把报纸送到他家门口，而清风出门的时间在早上 7：00 到 8：00 之间（假设送报人到达的时间和清风出发的时间在对应的时间区间内都是均匀分布的）。请使用蒙特卡罗模拟来估计清风在离开家之前能拿到报纸的概率。注意：这是概率论中几何概型的一道经典题目，准确的概率是 7/8.（思路：你可以模拟这个送报纸的过程 N 次，其中能拿到报纸的次数为 n，那么频率 n/N 就能看成概率的近似值，理论依据是伯努利大数定律）

Q14.异常值或离群值是指在一组数据中与其他数值相比差异较大的一个或几个数值。举个极端一点的例子，[4600,0,5000,5200,4700,4300,6000,5400,100000,6200]这一组数据中，我们可以认为 0 和 100000 这两个数就是异常值，因为剩下的数都集中在 5000 附近。异常值的识别和处理是数据清洗的重要环节，异常值的存在可能会导致后续的数据分析和建模工作出现偏差。本题将教给大家两种识别异常值的方法。

方法一：3σ原则识别异常值

学过概率论的同学应该知道，正态分布的概率密度函数图像是关于均值点处对称的，假设总体服从均值为μ，标准差为σ的正态分布，那么从该总体中随机抽取一个样本点，该点落在区间$[\mu-3\sigma,\mu+3\sigma]$上的概率约为 99.73%，而超出这个范围的可能性仅占不到 0.3%，是典型的小概率事件，所以这些超出该范围的数据可以认为是异常值。这就是3σ原则识别异常值的理论基础。

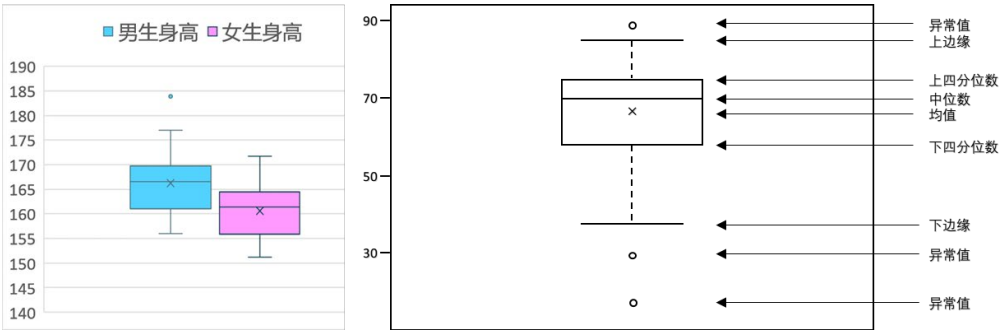


下面总结3σ原则识别异常值的步骤：（1）计算这组数据的均值μ和标准差σ（注意：我们得到的数据一般是样本数据，因此这里的标准差通常为样本标准差。如果总体标准差是已知的，那么就用总体的标准差）。（2）判断这组数据中的每个值是否都位于$[\mu-3\sigma,\mu+3\sigma]$这个区间内，如果不在这个区间内就标记为异常值。

注意事项：使用3σ原则确定异常值时，样本数据要来自正态分布总体或者近似于正态分布总体，这一点需要根据历史经验或统计检验来进行判断。

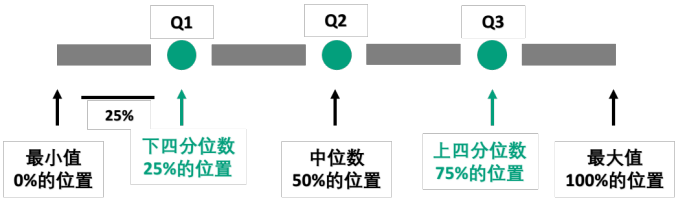
方法二：箱线图识别异常值

箱线图又称为盒须图、盒式图或箱形图，是一种用于显示数据分散情况资料的统计图，因形状如箱子而得名。下方左侧给出了一个用来反映某班男女同学身高分布情况的箱线图，右侧是箱线图上各元素所代表的含义。可以看到，箱线图可以反映数据的许多统计信息，例如均值、中位数、上四分位数和下四分位数。另外，箱线图中规定了数据的异常值，因此我们可以借助箱线图来识别数据的异常值，下面我们来介绍箱线图中异常值的定义方法。（注意：箱线图的画法不唯一，下图是一种典型的画法）



首先回顾下中位数的定义：我们将数据按从小到大的顺序排列，在排列后的数据中居于中间位置的数就是中位数，我们用 Q2 表示。

下四分位数则是位于排列后的数据 25%位置上的数，我们用 Q1 表示；上四分位数则是处在排列后的数据 75%位置上的数，我们用 Q3 表示。（MATLAB 中可以直接计算出 Q1 和 Q3，大家不用担心计算问题，请接着看后面的内容）



然后我们要定义一个叫做四分位距（IQR: interquartile range）的指标，它是上四分位数（Q3，即位于 75%）与下四分位数（Q1，即位于 25%）的距离，因此 IQR=Q3-Q1。四分位距反映了中间 50%的数据的离散程度，其数值越小，说明中间的数据越集中；其数值越大，说明中间的数据越分散。

接下来的工作和3σ原则识别异常值类似，我们需要给出一个合理的区间，位于该区间内的值是正常的数值，而在区间外的值就是异常值。在箱线图中，该区间一般为$[Q1-k\times IQR, Q3+k\times IQR]$，k是控制区间长度的一个正数，通常k取为 1.5。因此，我们只需要判断这组数据中的每个值是否都位于$[Q1-1.5\times IQR,\ Q3+1.5\times IQR]$这个区间内，如果不在这个区间内就将其标记为异常值。另外，如果我们将k取为 3，那么在区间$[Q1-3\times IQR,\ Q3+3\times IQR]$外的异常值被称为极端异常值。

和3σ原则相比，箱线图并没有对数据服从的分布作任何限制性要求（3σ原则要求数据服从正态分布或近似服从正态分布），其判断异常值的标准主要以四分位数和四分位距为基础。在总体分布未知的情况下，使用箱线图识别异常值的结果更加客观。

根据上面的介绍，请完成下面的问题：

（1） 清风开了一家撸猫店，他统计了最近一个月以来每天进店撸猫的人数（本故事纯属虚构）。请大家用下面这三行代码来生成一个随机的向量 x，向量 x 就表示清风记录的最近一个月以来每天进店的人数。从下一问开始，请你假装不知道这个数据是我们随机生成的，就把它当成真实的数据。

```
x = fix(50+5*randn(30,1));
x(randi(30,2,1)) = randi([50,90],2,1);
x(randi(30,2,1)) = randi([20,50],2,1);
```

（2） 计算这一个月来，撸猫人数最多的一天和最少的一天各有多少人？平均每天有多少人？标准差是多少？下四分位数、中位数和上四分位数各是多少？（提示：MATLAB 中可以使用函数 prctile 来计算分位数，请大家自己搜索它的用法）

（3） 无论是使用3σ原则还是箱线图识别异常值，均需要计算正常数值所在的区间，请分别算出这两种方法正常数值所在的区间，哪种方法的区间范围更大？

（4） 分别用两种方法识别出 x 中存在的异常值，并返回哪些天是异常值。你可以多次运行第一小问的代码生成新的 x，观察哪种方法识别出来的异常值更多？

（5） 以箱线图识别的异常值为例，分别完成下面两个任务：

（a）删除 x 中的异常值，将剩下的正常数值保存到向量 y 中；

（b）计算所有正常数值的平均值，并将 x 中的异常值替换成四舍五入后的平均值，将结果保存到向量 z 中。

Q15.在本章 3.3.5 小节介绍 sort 函数时，我们留下了一个问题：如果存在同学的成绩相同的情况，那么课上讲的代码将会失效，我们算出来的排名无法区分相同成绩的同学。下表给出了两种不同的排名结果，成绩越高排名越靠前，成绩相同则排名一样，但普通排名的并列排名会占据名次的数字位置，而中国式排名中的并列排名不占用名次。现给定第一行的成绩向量，请大家分别算出第二行和第三行的两种排名。

（提示：这个题目可以使用 ismember 函数解决，大家不要想的过于复杂。对于每种排名，参考答案只需要两三行代码就能解决）

成绩	20	60	80	50	90	90	60	80	80	70	50	30
普通排名	12	7	3	9	1	1	7	3	3	6	9	11
中国式排名	7	4	2	5	1	1	4	2	2	3	5	6

Q16.行向量 A 中包含至少两个元素且 A 中不含元素 0，请判断 A 中能否找到两个元素互为相反数，能找到则返回逻辑 1，不能找到则返回逻辑 0。例如 A = [3 1 -1 2]返回逻辑 1， A = [-3 4 1 2 6 -5]返回逻辑 0。进一步思考，如果向量 A 中可能包含 0，你的代码还适用吗？如果不适用请修改你的代码。例如 A=[3 4 -2 0 1]返回逻辑 0，A=[3 4 0 -2 0 1]和 A=[-6 3 0 -5 3 2 6 1]返回逻辑 1.（你能使用两种不同的思路求解吗？）

Q17.请完成下面的一系列任务：

（1） 随机生成两个长度为 6 的行向量 s 和 v，s 和 v 中的元素都是位于区间[1,10]之间的随机整数，例如随机生成的 s 和 v 分别为[2 4 1 7 4 9]和[7 8 3 2 4 5]；

（2） 如果 s 和 v 中相同位置的元素相同，则将该位置的元素从 s 和 v 中同时剔除掉；如果不存在这种情况则不剔除。例如：上面 s 和 v 中第五个位置的元素都是 4，那么将 4 剔除后，新的 s 和 v 分别为[2 4 1 7 9]和[7 8 3 2 5]；

（3） 经过上一步后，记向量 s 和 v 的长度为 m，请判断 s 和 v 中是否存在$\mathrm{s}_i = \mathrm{s}_j$且$v_i = v_j$，这里的$i, j \le m$且$i \ne j$。如果存在，则从 s 和 v 中剔除掉重复的值，仅保留一组即可。在上面的 s 和 v 的例子中，m 等于 5，不存在这样的情况。换个例子，如果 s 和 v 分别为[2 4 9 7 9]和[7 8 3 2 3]，那么i和j分别取 3 和 5 时，$\mathrm{s}_i = \mathrm{s}_j = 9$且$v_i = v_j = 3$，我们剔除后，新的 s 和 v 分别为[2 4 9 7]和[7 8 3 2].

（4） 经过上一步后，记向量 s 和 v 的长度为 n，请判断 s 和 v 中是否存在$\mathrm{s}_i = \mathrm{v}_j$且$v_i = s_j$，这里的$i, j \le n$且$i \ne j$。例如 s 和 v 分别为[2 4 9 7]和[7 8 3 2]时，n 等于 4，可以找到i=1,j=4 满足$\mathrm{s}_i = \mathrm{v}_j = 2$且$v_i = s_j = 7$.

Q18.本题带大家了解枚举法和网格搜索法。

枚举法在我们日常生活中使用的频率很高，它的核心思想就是枚举所有可能来找到正确或者最优的情况。

举个简单的例子：现在有一个两位数，十位数是 8，个位数未知，但是知道这个数可以被 9 整除，请回答它的个位数是什么？显然，个位上的数只能是$0,1,2,\cdots,9$中的一个，因此你可以依次尝试，这里面只有 81 符合条件。用枚举法解决问题，通常可以从以下两个方面思考：（1）找出枚举范围：我们需要列出问题中所有可能的解。在上面的例子中，枚举范围就是个位上可能存在的 0 到 9 这 10 个整数。（2）找出约束条件：我们需要找出问题的解应该满足的约束条件。在上面的例子中，约束条件就是这个数能被 9 整除。下面使用 MATLAB 来完成这个过程，仅需三行代码：

```
x = 0:9;  % 枚举范围
tmp = (mod(x+80, 9) == 0); % tmp 是一个长度为 10 的逻辑向量
x(tmp)+80  % 输出符合约束条件的解
```

以上是一个非常简单的使用枚举法的例子，那么枚举法有什么缺点呢？最容易想到的缺点就是：如果枚举范围中的情况非常的多，那么枚举所有情况就非常耗费时间。例如我们要判断整数 10 到 1 亿里面哪些数是质数，这个范围就非常的大。枚举法的另一个缺点就是可能会做很多无用功导致效率低下，例如判断质数这个例子中，我们没有必要列举 10 到 1 亿中为偶数的情况，因为它们一定不是质数。另外，如果枚举范围中有无数种可能，那么我们就无法使用枚举法来解决这个问题，例如使用计算机验证哥德巴赫猜想（任一大于 2 的偶数都可写成两个质数之和）是不可能的，因为这里面有无数个待验证的数。

但有一种情况例外，即使枚举范围中有无数种可能，我们仍可以挑选出有限种方案进行求解，这样能保证最后获得一个近似解。举个例子，我们要求函数$y = 2x - e^x$在区间$[0,1]$上的最大值。如果你学过高数的话，应该很容易求出最大值在$x = \ln 2$处取得。现在假设你没有学过高数，你应该怎样借助计算机求解这个问题？

如果我们使用枚举法，由于x在区间[0,1]上有无数种可能的取值，我们不可能全部列举出来，但我们可以在区间[0,1]上按一定的步长取有限个点作为x（例如$x = 0,0.01,0.02,\ldots,0.99,1$），然后依次算出在这些点上$y$的取值，从里面挑选出最大的那个$y$所对应的$x$来作为一个的近似解，这种策略被称为网格搜索法。

因为这里只有x这一个参数，所以搜索起来比较简单。显然，选取的步长越小，找到的解会越精确，例如x取为 0:0.0001:1 得到的解比 0:0.01:1 更精确。但要注意：步长设置的越小，网格搜索所需的时间会越长，因此这里有一个权衡取舍关系：我们需要根据要获得的解的精度和求解的时间综合考虑来设置一个合适的搜索步长。

另外，如果搜索的参数很多，例如我们要求出二元函数$z = f(x,y)$在x和y都属于[0,1]区间时它的最大值，那么我们使用网格搜索时就要对x和y可能的取值进行排列组合，假设x和y各有一万种可能的取值，那么我们的搜索次数将会高达一亿次。

如果有更多要搜索的参数，那么这个计算量将会非常的大，搜索次数随着要搜索的参数数量呈指数增长。因此，研究优化理论的学者提出了很多高效的优化算法，这些算法能够在可接受的计算时间下给出待解决的优化问题的一个可行解。做数学建模的同学可能听过启发式算法，常见的启发式算法有模拟退火、粒子群算法、遗传算法等。总之，优化算法的内容非常丰富，本题仅通过介绍枚举法和网格搜索法来引出相关的概念，有兴趣的同学可以在网上查阅相关的资料。

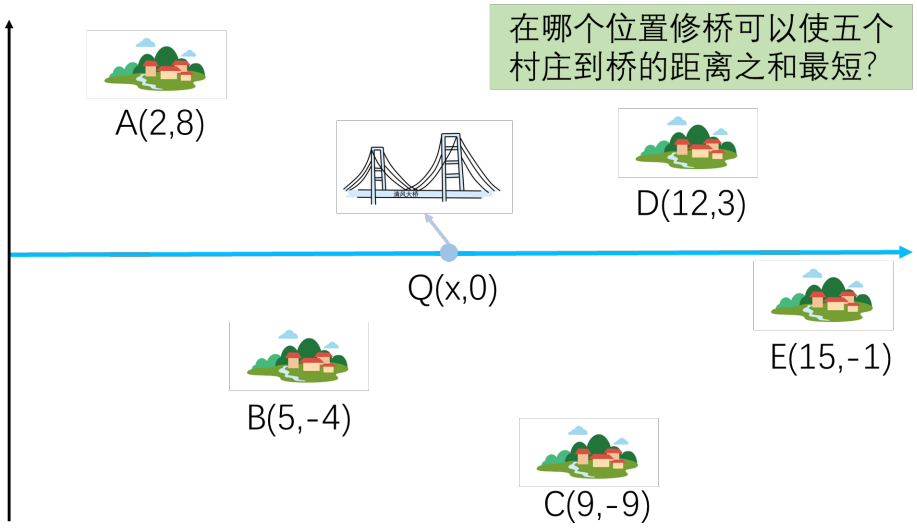
下面请大家完成下面的练习题：

（1） 使用网格搜索法求出$y = 2x - e^x$在区间[0,1]上的最大值，并求出此时的x。

（2） 使用网格搜索法求出$f(x,y) = x^3 - y^3 + 18x^2 + 12y^2 - 9x - 9y$在$x$和$y$都位于区间$[-1,1]$上的最小值，并求出此时$x$和$y$的值。（提示：你可能需要使用 meshgrid 函数生成xy平面上的二维网格坐标，该函数具体的用法可以查看 MATLAB 帮助文档。本题的参考答案为-2.854，最小值在(0.245,0.394)处取得）

（3） 假设你知道$y = 2x - e^x$在区间[0,1]上的图形是先递增后递减，你将如何改进你的搜索策略来提高搜索效率。（不需要写代码，可以用文字描述这种策略）

（4） 如下图所示，图上标出了五个村庄的坐标，现在要在横轴上（看成一条河流）选取某一点修建一座桥，请问在哪个位置修桥可以使这五个村庄的距离到桥的距离之和最短，这里的距离请用曼哈顿距离计算。



Q19.本题考察大家收集资料以及自学的能力。请大家查询相关资料并回答下面几个问题：

（1）本章中我们学过如何生成随机数，请大家搜索生成可重复的随机数的方法。

（2）如果大家学过概率论与数理统计，那么一定知道泊松分布和指数分布，请大家搜索生成这两个分布的随机数的方法。

（3）请大家自学 MATLAB 中排列组合的两个函数：nchoosek 和 perms。

Q20.本题将带大家学习皮尔逊相关系数和斯皮尔曼相关系数的计算步骤。

（1） 皮尔逊相关系数(Pearson correlation coefficient)

皮尔逊相关系数用于度量两组数据 X 和 Y 之间的线性相关的程度，它的范围为-1到$+1$，其中0代表无线性相关性，负值代表线性负相关，正值代表线性正相关。

在现实生活中，我们收集到的数据可以分为总体数据和样本数据，根据收集的数据类型不同，我们计算皮尔逊相关系数的公式也有所区别。通常我们收集到的数据都是样本数据，因此下面给出样本皮尔逊相关系数的计算公式。

假设我们搜集了两组样本数据 X:$[X_1, X_2, \cdots, X_n]$和 Y: $[Y_1, Y_2, \cdots, Y_n]$，那么 X 和 Y 之间的样本皮尔逊相关系数$r_{XY} = \frac{\mathrm{Cov}(X,Y)}{S_X S_Y}$，其中$\mathrm{Cov}(X,Y)$表示 X 和 Y 之间的样本协方差，计算公式$\mathrm{Cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{n-1}$，这里的$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$和$\bar{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$分别表示 X 和 Y 的样本均值，n 为样本个数；S_X和S_Y分别表示 X 和 Y 的样本标准差，$S_X = \sqrt{\frac{\sum_{i=1}^{n}(X_i-\bar{X})^2}{n-1}}$，$S_Y = \sqrt{\frac{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}{n-1}}$。请你根据公式计算下表给出的 X 和 Y 的样本皮尔逊相关系数。

X	92	79	78	96	64	88	88	69	78	53	52	77	89
Y	58	29	41	47	19	40	35	42	34	27	17	41	38

（提示：答案是 0.7629，你也可以使用 MATLAB 内置的 corrcoef 函数计算 X 和 Y 的皮尔逊相关系数，根据公式算出的结果应和内置函数的返回结果相同）

（2） 斯皮尔曼等级相关系数(Spearman's rank correlation coefficient)

斯皮尔曼等级相关系数简称斯皮尔曼相关系数，或称为秩相关系数，它是一种非参数方法，用于衡量两个变量依赖性的强度和方向。与皮尔逊相关系数不同，斯皮尔曼相关系数不要求两个变量是线性相关的，它主要用于衡量两个变量之间单调关系的强度和方向。斯皮尔曼相关系数的范围为-1到$+1$，如果当 X 增加时，Y 趋向于增加，则斯皮尔曼相关系数为正；如果当 X 增加时，Y 趋向于减少，则斯皮尔曼相关系数为负；斯皮尔曼相关系数为 0 表明当 X 增加时 Y 没有任何趋向性。

在介绍斯皮尔曼相关系数的计算公式之前，我们先介绍秩（或称为等级）的概念。一个数的秩就是将这个数所在的那组数据按照从小到大的顺序重新排列，这个数所处的位置下标；若出现相同的数，则将它们的位置下标计算平均值作为它们的秩（这里介绍的秩和矩阵的秩没有任何关系，大家不要弄混了）。

（例 1）计算 18、23、9、11、3、20 这组数据中各元素的秩。

首先从小到大进行排序：3、9、11、18、20、23；那么，18 所处的位置下标是 4；23 所处的位置下标是 6；9 所处的位置下标是 2；11 所处的位置下标是 3；3 所处的位置下标是 1；20 所处的位置下标是 5；因此，这组数据的秩就是：4、6、2、3、1、5。

（例 2）计算 18、23、9、11、11、20 这组数据中各元素的秩。

首先从小到大进行排序：9、11、11、18、20、23；那么，18 所处的位置下标是 4；23 所处的位置下标是 6；9 所处的位置下标是 1；11 所处的位置有两个，下标分别是 2 和 3，将 2 和 3 求平均值等于 2.5； 20 所处的位置下标是 5；因此，这组数据的秩就是：4、6、1、2.5、2.5、5。

介绍完秩的概念后，我们就能计算斯皮尔曼相关系数，它的计算方法有两种：（1）直接套用公式计算；（2）计算 X 和 Y 的秩之间的皮尔逊相关系数，将这个结果当成原始数据 X 和 Y 之间的斯皮尔曼相关系数。

第一种方法是基于以下公式直接计算：$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}$，式中的$d_i$为$X_i$和$Y_i$之间的秩差，$n$为 X 或 Y 中包含的元素个数。

举个例子，假设要计算 X = [3 8 4 7 2]和 Y = [5 10 8 10 6]这两组数据的斯皮尔曼相关系数，那么我们可以先求出 X 和 Y 对应的秩，然后算出秩差和秩差的平方：

X	Y	X 的秩	Y 的秩	秩差	秩差的平方
3	5	2	1	1	1
8	10	5	4.5	0.5	0.25
4	8	3	3	0	0
7	10	4	4.5	-0.5	0.25
2	6	1	2	-1	1

接下来只需要套用公式算出$r_s = 1 - \frac{6\times(1+0.25+0+0.25+1)}{5\times24} = 0.875.$

第二种方法是计算 X 和 Y 的秩之间的皮尔逊相关系数，即计算数据[2 5 3 4 1]和数据[1 4.5 3 4.5 2]的皮尔逊相关系数，大家可以自行计算，得到的结果为 0.8721。

可以看出，两种方法得到的结果有细微差别，这是因为原来的 Y 数据中有重复的元素。可以证明的是：如果 X 中的元素各不相同，且 Y 中的元素也各不相同时，两种方法得到的结果是相等的。

下面请大家编写代码求解下表中 X 和 Y 的斯皮尔曼相关系数，你需要使用上面介绍的两种方法分别计算。

X	92	79	78	96	64	88	88	69	78	53	52	77	89
Y	58	29	41	47	19	40	35	42	34	27	17	41	38

（提示：计算一组数据的秩需要用到 sort 函数，你需要思考如何处理存在相同数值的情况；当然，如果你绞尽脑汁也没有想出来怎么处理，你可以在 MATLAB 官网搜索 tiedrank 函数的用法。参考答案：第一种方法得到的结果为 0.6401，第二种方法得到的结果为 0.6386。另外，大家可以在 MATLAB 官网搜索 corr 函数，该函数可用来计算斯皮尔曼相关系数）

以上就是本章的课后习题，大家一定要认真做，这些题目都是按照本章知识点精心编写的，对大家编程能力的提高非常有帮助。

题目答案的讲解视频在 b 站观看（一定要先自己做，然后再对答案）：

https://www.bilibili.com/video/BV1bh4y1z7sa