第二周——向量和矩阵

题目目的

（一）掌握向量的创建方法。

（二）掌握向量的操作与运算。

（三）掌握矩阵的创建方法。

（四）掌握矩阵的操作与运算。

题目

题目一：用冒号运算符、c 函数、rep 函数和 seq 函数创建向量。创建脚本文件 test0201.R，完成下面操作。

- 使用冒号运算符创建一个包含 1 到 10 的向量，并赋值给 vector1。
- 使用 c 函数创建一个包含”apple”, “banana”, “cherry” 的向量，并赋值给 vector2。
- 使用 rep 函数创建一个包含数值 1 到 5 重复 2 次的向量，并赋值给 vector3。
- 使用 seq 函数创建一个从 10 到 50 以 10 为间隔的向量，并赋值给 vector4。
- 打印出所有创建的向量。
- 计算 vector1 和 vector2 的长度，并将结果打印出来。

- 使用冒号运算符选择 vector1 中的第 3 个到第 7 个元素，并将结果打印出来。
- 使用 c 函数将 vector1 和 vector2 合并，并将结果打印出来。
- 使用 rep 函数将 c(-5,3,7) 中的元素分别重复 3、4、2 次，并将结果打印出来。
- 使用 seq 函数创建一个从 1 到 10 的向量，其间隔设置为 2，并将结果打印出来。

题目二：使用 R 语言执行向量的索引操作、算术运算和排序操作。创建脚本文件 test0202.R，完成下面操作。

- 使用 R 语言创建一个包含以下数字的向量：1, 10, 3, 7, 2, 8。将其命名为 numbers。
- 使用索引操作选择 numbers 向量中的第 3 个到第 5 个元素，并将结果打印出来。
- 计算 numbers 向量的平均值，并将结果打印出来。
- 计算 numbers 向量的总和，并将结果打印出来。
- 使用 sort 函数对 numbers 向量进行降序排序，并将排序后的结果打印出来。
- 创建一个新的向量 squares，其中元素值为 numbers 向量中每个元素的平方，并将结果打印出来。
- 计算 numbers 的 5 次方除以向量 c(3,7,-13,17,-5,11) 的余数，并将结果打印出来。

题目三：使用 R 语言掌握向量元素的增加、删除、更新与命名。新建脚本文件 test0203.R，完成下面的操作。

- 创建一个名为 vect 的数值向量，包含 5 个元素：10, 20, 30, 40, 50。

- 用中括号运算符，在 vect 向量的末尾增加两个新元素 60 和 70。
- 用 append 函数，在 vect 向量的末尾增加两个新元素 100 和 200。
- 删除 vect 向量中的第 3 个和第 7 个元素。
- 将 vect 向量中的第 3 个元素更新为 300。
- 把 vect 向量中的元素分别用"a"、"b"、"c"、"d"、"e"、"f"、"g" 命名，并把结果打印出来。
- 创建一个长度为 100 的向量 x，其中 x 是首项为 5 间隔为 3 的等差序列构成，然后把 x 中的偶数位置的元素用向量 c(-10,-20) 更新，并把结果打印出来。

题目四：使用 R 语言掌握向量的条件筛选。创建脚本文件 test0204.R，完成下面的操作。

- 创建一个名为 numbers 的数值向量，包含以下元素：2, 4, 5, 2, 2, 3。
- 找出最大元素所在的位置。
- 提取 numbers 向量中所有等于 2 和大于 4 的元素的位置。
- 创建向量 x=rpois(300,100) 创建长度为 300 的向量 x，然后把第 3，6，… …，300 号元素筛选出来，并将选择的结果打印出来。

题目五：使用 matrix 和 diag 函数创建矩阵。新建脚本文件 test0205.R，完成下面的操作。

- 使用 matrix 函数创建一个 4 行 3 列的矩阵，元素从 1 到 12，要求按行排列。
- 创建如下形式的矩阵。

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    1    2    3
[3,]    1    2    3
```

- 创建如下形式的矩阵，且行名称为"r1", "r2", "r3"，列名称为"c1", "c2", "c3"。

```
   c1 c2 c3
r1  1  1  1
r2  2  2  2
r3  3  3  3
```

- 创建如下形式的矩阵。

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    1    2
[2,]    1    2    1    2
```

- 用 diag 创建如下矩阵（先用帮助文档查看 diag 函数的用法）。

```
     [,1] [,2] [,3]
[1,]    3    0    0
[2,]    0    4    0
[3,]    0    0    5
```

题目六：矩阵的基本操作。打开脚本文件 test0206.R，补充代码，实现以下操作。

- 给矩阵 mat 的行/列命名，行名称分别为 "第 1 行"、"第 2 行"、"第 3 行"、"第 4 行"，列名称分别为 "第 1 列"、"第 2 列"、"第 3 列"。
- 修改矩阵 mat 中第 2 行第 3 列的元素值为-15。
- 把矩阵 mat 除开第 3 行的子矩阵赋值给变量 y。
- 把矩阵 mat 第 1 行元素值改为-10、第 3 行元素值改为-20，要求只用一条语句。
- 把矩阵 mat 第 1 列元素值改为-100、第 2 列元素值改为-200，要求只用一条语句。

题目七：矩阵的算术运算和代数积运算。打开脚本文件 test0207.R，补充代码，实现以下操作。

- 计算矩阵 x 与矩阵 u 的和。
- 计算矩阵 x 与矩阵 y 的和，观察返回结果。
- 计算矩阵 x 与向量 w 的积。
- 计算矩阵 x 与矩阵 z 的积，观察返回结果。
- 计算矩阵 x 与矩阵 y 的代数积。
- 计算矩阵 y 与矩阵 x 的代数积，观察返回结果。
- 计算矩阵 x 与矩阵 z 的代数积。
- 计算矩阵 x 与向量 w 的代数积。
- 提取矩阵 x 对角线上的元素。
- 对矩阵 y 进行转置。
- 计算矩阵 x 对应的行列式的值。
- 提到矩阵 x 的特征值。
- 用 eigen 函数计算矩阵 y 的特征值与特征向量，观察返回结果。

题目八：矩阵相关函数，包括 dim 函数、apply 函数、ncol 函数与 nrow 函数、colSums 函数、rowSums 函数、colMeans 函数、rowMeans 函数。打开脚本文件 test0208.R，补充代码，完成下面任务。

- 计算矩阵的行数与列数。
- 计算矩阵 x 的维度。
- 计算矩阵 x 每行的和。
- 计算矩阵 x 每列的平均值。

- 用 apply 函数计算矩阵 x 每行的方差。
- 计算矩阵 x 所有元素的平方和。

题目九：综合题。创建脚本文件 test0209.R，补充代码，完成下面任务。

空间三角形 ABC 的顶点坐标分别为 A(3,4,5)、B(-3,2,4)、C(5,-3,4)，请计算三角形三边的长度以及三个夹角的余弦值。

向量 x=c(9, 8, 10, 7, 12, 11, 12, 6. 10, 3, 11, 13, 7, 10, 9)，请计算 x 中的元素两两相乘之积的和，例如 $x = c(x1, x2, x3)$ 则两两相乘之积的和为 $x1x2 + x2x3 + x1x3$。

答案及解析

题目一：

```
# 为了方便理解，这里把需要一次性打印的向量分开打印
vector1 <- 1:10
print(vector1)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
vector2 <- c("apple", "banana", "cherry")
print(vector2)
```

```
[1] "apple"  "banana" "cherry"
```

```
vector3 <- rep(1:5, times = 2)
# 将向量 (1:5) 重复两次
print(vector3)
```

```
 [1] 1 2 3 4 5 1 2 3 4 5
```

```
vector4 <- seq(from = 10, to = 50, by = 10)
# 首项为 10，尾项为 50，步长为 10 组成的等差数列组成的向量
print(vector4)
```

```
[1] 10 20 30 40 50
```

```
# 利用 length(vector) 来算向量内元素的个数
print(length(vector1))
```

```
[1] 10
```

```
print(length(vector2))
```

```
[1] 3
```

```
print(vector1[3:7])
```

```
[1] 3 4 5 6 7
```

```
# 当两个向量需要组合在一起的时候，可以用 c 来连接
print(c(vector1, vector2))
```

```
 [1] "1"      "2"      "3"      "4"      "5"      "6"      "7"      "8"
 [9] "9"      "10"     "apple"  "banana" "cherry"
```

```
# 将-5 重复 3 次，3 重复 4 次，7 重复 2 次
print(rep(c(-5, 3, 7), times = c(3, 4, 2)))
```

```
[1] -5 -5 -5  3  3  3  3  7  7
```

```
print(seq(from = 1, to = 10, by = 2))
```

```
[1] 1 3 5 7 9
```

> 归纳创建向量 (Vector) 方法：
>
> - 用 $x:y$，创建一个 $(x, x+1, ..., y-1, y)$ 的向量。
> - c(元素）
> - 当元素需要重复时:rep()
> - 当创建一个等差数列的向量时：seq()

题目二：

```
numbers <- c(1, 10, 3, 7, 2, 8)

print(numbers[3:5])
```

```
[1] 3 7 2
```

```
print(mean(numbers))
```

```
[1] 5.166667
```

```
print(sum(numbers))
```

```
[1] 31
```

```
#sort 为排序, decreasing 为排序顺序,T 为升序, F 为降序
print(sort(numbers, decreasing = TRUE))
```

```
[1] 10  8  7  3  2  1
```

```r
squares <- numbers^2
# 对向量进行平方运算时，每个元素单独进行平方
print(squares)
```

```
[1]   1 100   9  49   4  64
```

```r
# 对 numbers 中的每个元素进行 5 次方，然后分别与后面向量的元素进行模运算
print(numbers^5 %% c(3, 7, -13, 17, -5, 11))
```

```
[1]  1  5 -4 11 -3 10
```

我们利用 vector[] 来索引向量内的元素

不同于其他语言从 0 开始编号，R 语言从 1 开始编号

向量运算示例	作用
sum()	向量内元素求和
mean()	求平均数
max(),min()	求最大最小值
length()	求向量内元素个数
sort()	向量内元素排序（默认升序，T 升 F 降）
sd(),var()	计算标准差，方差

题目三：

```r
vect <- c(10, 20, 30, 40, 50)

vect <- c(vect, 60, 70)
```

```
vect <- append(vect, c(100, 200))

vect <- vect[-c(3, 7)]

vect[3] <- 300



names(vect) <- c("a", "b", "c", "d", "e", "f", "g")
print("Named vect:")
```

```
[1] "Named vect:"
```

```
print(vect)
```

```
  a   b   c   d   e   f   g
 10  20 300  50  60 100 200
```

```
x <- seq(from = 5, by = 3, length.out = 100)
x[seq(2, length(x), by = 2)] <-
  rep(c(-10, -20), length.out = length(x) / 2)
print("Updated x:")
```

```
[1] "Updated x:"
```

```
print(x)
```

```
  [1]   5 -10  11 -20  17 -10  23 -20  29 -10  35 -20  41 -10  47 -20  53 -10
 [19]  59 -20  65 -10  71 -20  77 -10  83 -20  89 -10  95 -20 101 -10 107 -20
 [37] 113 -10 119 -20 125 -10 131 -20 137 -10 143 -20 149 -10 155 -20 161 -10
 [55] 167 -20 173 -10 179 -20 185 -10 191 -20 197 -10 203 -20 209 -10 215 -20
 [73] 221 -10 227 -20 233 -10 239 -20 245 -10 251 -20 257 -10 263 -20 269 -10
 [91] 275 -20 281 -10 287 -20 293 -10 299 -20
```

> 增加（删除）向量元素方法:
>
> 可以直接创建一个新向量，把旧向量和新元素放入
> append() 函数可以直接在尾部增加新元素
> 如该向量所含元素为 m，可以直接使用 vector[m+1] 来赋值
> 当我们要删除元素的时候，可以用”-“删除

题目四：

```r
numbers <- c(2, 4, 5, 2, 2, 3)
# 找到最大值的索引
print(which.max(numbers))
```

```
[1] 3
```

```r
print(which(numbers == 2 | numbers > 4))
```

```
[1] 1 3 4 5
```

```r
print("Selected elements from x:")
```

```
[1] "Selected elements from x:"
```

```r
print(x[seq(3, 300, by = 3)])
```

```
  [1]  11 -10  29 -20  47 -10  65 -20  83 -10 101 -20 119 -10 137 -20 155 -10
 [19] 173 -20 191 -10 209 -20 227 -10 245 -20 263 -10 281 -20 299  NA  NA  NA
 [37]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
 [55]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
 [73]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
 [91]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
```

Table 2: 基本逻辑运算符

运算符	逻辑	作用
!	非（NOT）	用于取反
&	与（AND）	逐元素进行逻辑与运算
|	或（OR）	逐元素进行逻辑或运算
&&	与（AND）	仅对第一个元素进行运算（常用于条件判断）
||	或（OR）	仅对第一个元素进行运算（常用于条件判断）

题目五：

```r
mat1 <- matrix(1:12, nrow = 4, ncol = 3, byrow = TRUE)
# 当 byrow 为 TRUE 时，按照行排列，FALSE 时，按照列排列
# 可以通过 nrow 和 ncol 来分别规定矩阵的行数和列数
print(mat1)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

```r
mat2 <- matrix(c(1, 2, 3), nrow = 3, ncol = 3)
print(mat2)
```

```
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    2    2
[3,]    3    3    3
```

```r
mat3 <- matrix(1:3, nrow = 3, ncol = 3,byrow = F)
rownames(mat3) <- c("r1", "r2", "r3")
colnames(mat3) <- c("c1", "c2",'c3')
# 利用 rownames 和 colnames 来命名矩阵的行和列
print(mat3)
```

```
   c1 c2 c3
r1  1  1  1
r2  2  2  2
r3  3  3  3
```

```r
mat4 <- matrix(c(1, 2), nrow = 2, ncol = 4,byrow = T)
# 每个元素将重复两次以填满每一行（当元素不足时）
print(mat4)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    1    2
[2,]    1    2    1    2
```

```r
mat5 <- diag(3:5)
# 当你向 diag 函数提供一个向量时，它会创建一个对角矩阵，
# 其中对角线上的元素是向量的元素，其余元素都是 0。
print(mat5)
```

```
     [,1] [,2] [,3]
[1,]    3    0    0
[2,]    0    4    0
[3,]    0    0    5
```

题目六：

```
set.seed(1)
# 用于设置随机数生成器的种子。种子值为 1。
x = rpois(12,30)
# 生成一个长度为 12 的随机向量，其中每个元素都来自参数为 30 的泊松分布。
mat = matrix(x,nrow = 4)
mat
```

```
     [,1] [,2] [,3]
[1,]   26   21   28
[2,]   37   32   38
[3,]   36   34   32
[4,]   32   33   26
```

```
rownames(mat) <- c(" 第 1 行", " 第 2 行", " 第 3 行", " 第 4 行")
colnames(mat) <- c(" 第 1 列", " 第 2 列", " 第 3 列")
# 重命名

mat[" 第 2 行", " 第 3 列"] <- -15
y <- mat[-3, ]
# 提取矩阵中除第 3 行之外的所有行，结果存储在变量 y 中
print(y)
```

```
     第1列 第2列 第3列
第1行    26    21    28
第2行    37    32   -15
第4行    32    33    26
```

```
mat[c(" 第 1 行", " 第 3 行"), ] <- c(-10, -20)
# 将矩阵中第 1 行和第 3 行的所有元素分别设置为 -10 和 -20
mat[, c(" 第 1 列", " 第 2 列")] <- c(-100, -200)
# 将矩阵中第 1 列和第 2 列的所有元素分别设置为 -100 和 -200
print(mat)
```

	第1列	第2列	第3列
第1行	-100	-100	-10
第2行	-200	-200	-15
第3行	-100	-100	-20
第4行	-200	-200	26

题目七：

```
set.seed(100)
x = matrix(rpois(9,10),nrow = 3)
set.seed(200)
y = matrix(rpois(12,10),nrow = 3)
z = matrix(1:3,nrow = 3)
w = c(3,7,1)
u = matrix(1:9,nrow = 3)

print(x + u)
```

```
     [,1] [,2] [,3]
[1,]    9   16   15
[2,]   12   15   20
[3,]   12   17   16
```

```
#print(x + y)
# 错误于 x + y: 非整合陈列

print(x * w)
```

```
     [,1] [,2] [,3]
[1,]   24   36   24
[2,]   70   70   84
[3,]    9   11    7
```

```
#print(x * z)
# 错误于 x * z: 非整合陈列

print(x %*% y)
```

```
     [,1] [,2] [,3] [,4]
[1,]  288  280  192  220
[2,]  332  318  212  256
[3,]  277  272  184  214
```

```
#print(y %*% x)
# 错误于 y %*% x: 非整合参数
print(x %*% w)
```

```
     [,1]
[1,]  116
[2,]  112
[3,]  111
```

```
print(diag(x))
```

```
[1]  8 10  7
```

```
print(t(y))
```

```
     [,1] [,2] [,3]
[1,]   10   10   11
[2,]   11   10    9
[3,]    6    8    6
[4,]    9    7    8
```

```
print(det(x))
```

```
[1] 120
```

```
print(eigen(x)$values)
```

```
[1] 29.184179 -2.606891 -1.577288
```

```
#print(eigen(y))
# 错误于 eigen(y): eigen 里的矩阵不是正方形的
```

题目八：

```
x = matrix(1:30,nrow = 5)
ncol(x)
```

```
[1] 6
```

```
dim(x)
```

```
[1] 5 6
```

```
nrow(x)
```

```
[1] 5
```

```
ncol(x)
```

```
[1] 6
```

```
dim(x)
```

```
[1] 5 6
```

```
rowSums(x)
```

```
[1]  81  87  93  99 105
```

```
colMeans(x)
```

```
[1]  3  8 13 18 23 28
```

```
# 对矩阵 x 的每一行应用方差函数 var
apply(x,1,var)
```

```
[1] 87.5 87.5 87.5 87.5 87.5
```

```
sum(x^2)
```

```
[1] 9455
```

题目九：

```
A = c(3,4,5)
B = c(-3,2,4)
C = c(5,-3,4)
AB <- sqrt((-3 - 3)^2 + (2 - 4)^2 + (4 - 5)^2)
BC <- sqrt((5 - (-3))^2 + (-3 - 2)^2 + (4 - 4)^2)
CA <- sqrt((3 - 5)^2 + (4 - (-3))^2 + (5 - 4)^2)
cos_A <- (AB^2 + CA^2 - BC^2) / (2 * AB * CA)
print(cos_A)
```

```
[1] 0.06375767
```

```
cos_B <- (AB^2 + BC^2 - CA^2) / (2 * AB * BC)
print(cos_A)
```

```
[1] 0.06375767
```

```
cos_C <- (CA^2 + BC^2 - AB^2) / (2* CA * BC)
print(cos_C)
```

```
[1] 0.7356619
```

```
x = c(9,8,10,7,12,11,12,6,10,3,11,13,7,10,9)
sum_of_products <- sum(combn(x, 2, FUN = prod))
print(sum_of_products)
```

```
[1] 8838
```