talk02 练习与作业

目录

0.1 练习和作业说明 1

0.2 talk02 内容回顾 1

0.3 练习与作业：用户验证 2

0.4 练习 1：vector 的基本类型与简单算术 2

0.5 练习 2：`vector` 操作 10

0.6 练习 3：逻辑检验和运算 16

0.7 练习 4：matrix、计算及相关函数 18

0.8 练习 5：特别值 22

0.1 练习和作业说明

将相关代码填写入以 “‘{r} “‘ 标志的代码框中，运行并看到正确的结果；

完成后，用工具栏里的”Knit” 按键生成 PDF 文档；

将生成的 PDF 改为：姓名`-`学号`-talk02` 作业`.pdf`，并提交到老师指定的平台/钉群。

0.2 talk02 内容回顾

- R language basic
  - 基本数据类型

– 简单算术
– 特别值

0.3 练习与作业：用户验证

请运行以下命令，验证你的用户名。

如你当前用户名不能体现你的真实姓名，请改为拼音后再运行本作业！

```
Sys.info()[["user"]]
```

```
## [1] "lucas"
```

```
Sys.getenv("HOME")
```

```
## [1] "/Users/lucas"
```

0.4 练习 1：vector 的基本类型与简单算术

- **用 class 命令确定以下 vector 的类型；**

```
c(100, 20, 30)
c(" 字符串", " 数组"," 是我")
c(TRUE, FALSE, TRUE, T, F)
```

```
## 将代码写在此处，并运行，比如：
class(c(100, 20, 30));
```

```
## [1] "numeric"
```

```
class(c(" 字符串", " 数组"," 是我"));
```

```
## [1] "character"
```

```r
class(c(TRUE, FALSE, TRUE, T, F))
```

```
## [1] "logical"
```

- **用 class 命令确定以下 vector 的类型;**

```r
c(45, TRUE, 20, FALSE, -100)
```

```r
c("string a", FALSE, "string b", TRUE)
```

```r
c("a string", 1.2, "another string", 1e-3)
```

```r
## 将代码写在此处，并运行
class( c(45, TRUE, 20, FALSE, -100) );
```

```
## [1] "numeric"
```

```r
class(c("string a", FALSE, "string b", TRUE));
```

```
## [1] "character"
```

```r
class(c("a string", 1.2, "another string", 1e-3));
```

```
## [1] "character"
```

请解释为什么整个 vector 的结果与单个成员的类型并不完全一致?

答:

由于类似于"2"，或者"True" 这类变量可以适用于多个类型，但是一整个 vector 只能对应一种类型，所以当一个 vector 的成员中含有多种类型时，系统会自动将他们转换为固定的"charactor"或者"numeric"类型的变量。具体转换规则如下:

- 逻辑符 logical→ 数字类型 numeric;
- 逻辑符 logical→ 字符串 character;
- 数字类型 numeric→ 字符串 character。

- **运行以下代码:**

```
x <- c(10,100,1000, 10000);
( y <- sqrt( x ) * 4 + 10 );
```

```
## 代码写在此处并运行
x = c(10,100,1000, 10000)
( y = sqrt( x ) * 4 + 10 )
```

```
## [1]  22.64911  50.00000 136.49111 410.00000
```

问:第二行代码最外层的括号有什么作用?

答:

不添加括号的话此代码无法将结果打印在控制台中

- **以下两个 vector,计算它们的乘积:**

```
x <- c(4,6,5,7,10,9,4,15)
y <- c(0,10,1,8,2,3,4,1)
```

```
## 代码写在此处并运行
x = c(4,6,5,7,10,9,4,15);
y = c(0,10,1,8,2,3,4,1);
x * y
```

```
## [1]  0 60  5 56 20 27 16 15
```

- **以下两个 vector ，计算：a <= b ：**

```
a <- c(1,5,4,3,6)
b <- c(3,5,2,1,9)
```

```
## 代码写在此处并运行
a = c(1,5,4,3,6);
b = c(3,5,2,1,9);
a <= b
```

```
## [1]  TRUE  TRUE FALSE FALSE  TRUE
```

- **将函数 dim, is.numeric, is.character, is.logical,length 应用到下面的 vector‘，并展示结果；**

```
x <- 1:12
y <- LETTERS[1:12]
z <- c(F, T, FALSE);
```

```
## 代码写在此处并运行
x = 1:12;
y = LETTERS[1:12];
z = c(F, T, FALSE);
dim(x);
```

```
## NULL
```

```
dim(y);
```

```
## NULL
```

```
dim(z);
```

```
## NULL
```

```
is.numeric(x);
```

```
## [1] TRUE
```

```
is.numeric(y);
```

```
## [1] FALSE
```

```
is.numeric(z);
```

```
## [1] FALSE
```

```
is.character(x);
```

```
## [1] FALSE
```

```
is.character(y);
```

```
## [1] TRUE
```

```
is.character(z);
```

```
## [1] FALSE
```

```
is.logical(x);
```

```
## [1] FALSE
```

```
is.logical(y);
```

```
## [1] FALSE
```

```
is.logical(z);
```

```
## [1] TRUE
```

```
length(x);
```

```
## [1] 12
```

```
length(y);
```

```
## [1] 12
```

```
length(z);
```

```
## [1] 3
```

- 以下两个 vector ，计算：which(!is.finite(x/y))：

```
x <- c(12:4)
y <- c(0,1,2,0,1,2,0,1,2)
```

```
## 代码写在此处并运行
x = c(12:4);
y = c(0,1,2,0,1,2,0,1,2);
which(!is.finite(x/y));
```

```
## [1] 1 4 7
```

提问：请解释输出结果的含义?

答：

输出 x/y 的算式无意义的位置

- **以下两个 vector ，计算：x > y ：**

```
x <- letters[1:10]
```

```
y <- letters[15:24]
```

```
## 代码写在此处并运行
x = letters[1:10];
y = letters[15:24];
x > y;
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

- **以下 vector ：**

```
x <- c(4,6,5,7,10,9,4,15)
```

计算：

```
x < 7
```

```
x < c( 8, 4 ); ## 与第二个 vector 进行
```

```
## 代码写在此处并运行
x = c(4,6,5,7,10,9,4,15);
x < 7;
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE
```

```
x < c(8, 4)
```

```
## [1]  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
```

问：请问第二个 vector 成员的循环规则是什么？这种循环在 R 里被称为什么？

答：

第二个 vector 的循环规则是将数据自动循环使用。这种循环在 R 里被称为 vectorisation。

- **练习阶乘和取余操作：**

```
2 ^ 6
```

```
1:10 ^ 2
```

```
5 %% 2
```

```
100:110 %% 2
```

```
## 代码写在此处并运行
2 ^ 6;
```

```
## [1] 64
```

```
1:10 ^ 2;
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```

```
5 %% 2;
```

```
## [1] 1
```

```
100:110 %% 2;
```

```
##  [1] 0 1 0 1 0 1 0 1 0 1 0
```

- 将函数 `is.vector` 应用到以下数据:

```
c( 8, 9, 10)
```

```
T
```

```
7
```

问: 后两个的输出结果是什么? TRUE or FALSE? 为什么?

答:

输出结果均为 TRUE, 因为 R 里的所有对象都是 vector。

0.5 练习 2: vector 操作

- 合并:

```
a <- 1:3;
b <- LETTERS[1:3];
( ab <- c(a,b) );
```

```
## 代码写在此处并运行
a = 1:3;
b = LETTERS[1:3];
ab = c(a,b)
```

- **用至少两个函数检测上面生成的变量 ab 的数据类型；**

```
## 代码写在此处并运行
class(ab);
```

```
## [1] "character"
```

```
is.character(ab);
```

```
## [1] TRUE
```

```
is.array(ab);
```

```
## [1] FALSE
```

- **取 vector 的一部分**

先生成一个 vector，并对其每个成员进行命名:

```
v <- 1:10;
names( v ) <- letters[1:10];
v; ## 显示 v 的内容
```

```
## 代码写在此处并运行
v = 1:10;
names(v) = letters[1:10];
v
```

```
##  a  b  c  d  e  f  g  h  i  j
##  1  2  3  4  5  6  7  8  9 10
```

取部分操作:

```
v[1]; ## index based method
```

```
v[ 2:5 ];
```

```
v[ c(1,3,9,2,5];
```

```
v[ "a" ];
```

```
v[ c( "a", "c", "b") ];
```

注: 运行上述代码，并于每次运行后，显示 v 的当前值;

```
## 代码写在此处并运行
v [1];
```

```
## a
## 1
```

```
v [2:5];
```

```
## b c d e
## 2 3 4 5
```

```
v [c(1,3,9,2,5)];
```

```
## a c i b e
## 1 3 9 2 5
```

```
v ["a"];
```

```
## a
## 1
```

```
v [c("a", "c", "b")]
```

```
## a c b
## 1 3 2
```

- 替换

```
v[ 1 ] <- 100;
```

```
v[2:3] <- 100;
```

```
v[ 3:5 ] <- c( 100, 200 );
```

```
v[ c(1, 5, 3 ) ] <- c(100, 500, 300);
```

注：运行上述代码，并于每次运行后，显示 v 的当前值；

```
## 代码写在此处并运行
v [1] = 100;
v;
```

```
##   a   b   c   d   e   f   g   h   i   j
## 100   2   3   4   5   6   7   8   9  10
```

```
v[2:3] = 100;
v;
```

```
##   a   b   c   d   e   f   g   h   i   j
## 100 100 100   4   5   6   7   8   9  10
```

```
v [3:5] = c(100,200);
```

```
## Warning in v[3:5] = c(100, 200): number of items to replace is not a multiple
## of replacement length
```

```
v;
```

```
##   a   b   c   d   e   f   g   h   i   j
## 100 100 100 200 100   6   7   8   9  10
```

```
v[c(1, 5, 3)] = c(100, 500 ,300);
v;
```

```
##   a   b   c   d   e   f   g   h   i   j
## 100 100 300 200 500   6   7   8   9  10
```

- **在 vector 的后面增加一个成员；此操作会改变 vector 的长度；**

```
a <- sample(1:20, 10);
length(a);
a[ length(a) + 1] <- 666;
length(a);
a;
```

```
## 代码写在此处并运行
a = sample(1:20, 10);
length(a);
```

```
## [1] 10
```

```
a[ length(a) + 1] = 666;
length(a);
```

```
## [1] 11
```

```
a;
```

```
##  [1]  13  10  11   9  14   3  15   2  19  18 666
```

- **以下两个 vector 相加，并查看结果；**

```
p <- c (3, 5, 6, 8)
q <- c (3, 3, 3)
```

```
## 代码写在此处并运行
p = c(3,5,6,8);
q = c(3,3,3);
p + q;
```

```
## Warning in p + q: longer object length is not a multiple of shorter object
## length
```

```
## [1]  6  8  9 11
```

- **取出下面 vector 中数据大于 20 的成员，并显示：**

a <- sample(1:50, 20);

```
## 代码写在此处并运行
a = sample(1:50, 20);
a[a > 20];
```

```
##  [1] 47 25 33 36 38 42 41 23 27 31 43 21
```

0.6 练习 3：逻辑检验和运算

- 用函数 isTRUE 计算以下数值或表达式，查看结果；

```
T | F
```

```
T & F
```

```
5 | 0
```

```
5 & 6
```

```
## 代码写在此处并运行
isTRUE(T | F);
```

```
## [1] TRUE
```

```
isTRUE(T & F);
```

```
## [1] FALSE
```

```
isTRUE(5 | 0);
```

```
## [1] TRUE
```

```
isTRUE(5 & 6);
```

```
## [1] TRUE
```

问题:为什么 isTRUE(5) 为 FALSE,isTRUE(6) 也为 FALSE,但 isTRUE(5 & 6) 是 TRUE?

答:

isTRUE 的本质是 { is.logical(x) && length(x) == 1 && !is.na(x) && x }，而在单独进行判断时，系统内部会将所有数值型的变量转换为逻辑值。同理，& 运算符也是完成一个类似于判断真假的操作，所以 5 & 6 这次运算由于两个数字都不是 0，所以返回值为 1，从而导致 isTRUE(5 & 6) 是 TRUE。

- **将 isTRUE 应用于以下数值，并查看结果:**

-1

-100

0

1

100

```
## 代码写在此处并运行
isTRUE(-1);
```

```
## [1] FALSE
```

```
isTRUE(-100);
```

```
## [1] FALSE
```

```
isTRUE(0);
```

```
## [1] FALSE
```

```
isTRUE(1);
```

```
## [1] FALSE
```

```
isTRUE(100)
```

```
## [1] FALSE
```

0.7 练习 4：matrix、计算及相关函数

- 生成一个 matrix，并查看结果，注意 dimnames 的用法：

```
m <- matrix( c(20, 30.1, 2, 45.8, 23, 14), nrow = 2, dimnames
= list( c("row_A", "row_B"), c("A", "B", "C")));
```

```
## 代码写在此处并运行
m = matrix( c(20, 30.1, 2, 45.8, 23, 14), nrow = 2, dimnames = list( c("row_A", "row_B"
m
```

```
##          A    B  C
## row_A 20.0  2.0 23
## row_B 30.1 45.8 14
```

- 在上面生成的变量 m 上运行以下函数：

```
dim
```

```
nrow
```

```
ncol
```

```
range
```

```
summary
```

colnames

rownames

t

```
## 代码写在此处并运行
dim(m);
```

```
## [1] 2 3
```

```
nrow(m);
```

```
## [1] 2
```

```
ncol(m);
```

```
## [1] 3
```

```
range(m);
```

```
## [1]  2.0 45.8
```

```
summary(m);
```

```
##        A               B               C
##  Min.   :20.00   Min.   : 2.00   Min.   :14.00
##  1st Qu.:22.52   1st Qu.:12.95   1st Qu.:16.25
##  Median :25.05   Median :23.90   Median :18.50
##  Mean   :25.05   Mean   :23.90   Mean   :18.50
##  3rd Qu.:27.57   3rd Qu.:34.85   3rd Qu.:20.75
##  Max.   :30.10   Max.   :45.80   Max.   :23.00
```

```
colnames(m);
```

```
## [1] "A" "B" "C"
```

```
rownames(m);
```

```
## [1] "row_A" "row_B"
```

```
t(m);
```

```
##   row_A row_B
## A    20  30.1
## B     2  45.8
## C    23  14.0
```

- **用代码实现以下操作：**

a. 取第一行

b. 取第二列

c. 同时取第三、二列，注意取的顺序；

并且，用 `class` 函数检验得到结果的数据类型；

```
## 代码写在此处并运行
m[1,];
```

```
##  A  B  C
## 20  2 23
```

```
m[,2];
```

```
## row_A row_B
##   2.0  45.8
```

```
m[,c(3,2)];
```

```
##         C    B
## row_A 23  2.0
## row_B 14 45.8
```

- **用代码实现以下操作：**

a. 用 1-1000 之间随机数值（用 `sample` 函数取值）替换第一行；

b. 用 1-1000 之间随机数值（用 `sample` 函数取值）替换第二列；

```
## 代码写在此处并运行
m[1,] = sample(1:1000, 3);
m;
```

```
##           A     B   C
## row_A 953.0 722.0 322
## row_B  30.1  45.8  14
```

```
m[,2] = sample(1:1000, 2);
m;
```

```
##           A   B   C
## row_A 953.0 804 322
## row_B  30.1 641  14
```

0.8 练习 5：特别值

- 用以下函数或命令式检测特别值构成的 `vectorsp` ，报告输出结果

```
sp <- (NA, NaN, Inf, -Inf)
```

```
is.finite
```

```
! is.infinite
```

```
is.na
```

```
is.nan
```

```
## 代码写在此处并运行
sp = (c(NA, NaN, Inf, -Inf));
is.infinite(sp);
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

```
! is.infinite(sp);
```

```
## [1]  TRUE  TRUE FALSE FALSE
```

```
is.na(sp);
```

```
## [1]  TRUE  TRUE FALSE FALSE
```

```
is.nan(sp);
```

```
## [1] FALSE  TRUE FALSE FALSE
```