# Contents



# R 语言基本运算

---

在这一章里面，我们将介绍一些简单的 R 软件运算，包括基本数字运算、向量运算与统计运算，让读者们对 R 软件的基本计算功能先有一个初步的印象。

## 简单的数字与字符串运算

R 软件的简单运算是通过程序语言通用的运算符符号来完成的。

```
1+1
```

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

```
1*3.4
```

```
## [1] 3.4
```

```
1/2
```

```
## [1] 0.5
```

```
1%/%2
```

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

余数 ( modulus) :

```
5 %% 2
```

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

三角函数运算：

```
cos(1.0)
```

```
## [1] 0.5403023
```

幂次计算:

```
2 ^ 0.5
```

```
## [1] 1.414214
```

```
sqrt(2)
```

```
## [1] 1.414214
```

科学计数符号：

```
x = 1.2e-5
x * 10000
```

```
## [1] 0.12
```

逻辑运算会产生逻辑向量：

```
x = c(1,2,3,4,5)
x>3
```

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

### 有序数列：规则性的数字集合

在 R 软件中，如果想要构建规则性的数字或向量，可以使用以下函数：

- sequence（有序数列）运算符
- seq（起始值，结束值，by：递增值）: sequence 函数，例如，seq（5）会产生（1，2，3，4，5）。若加上 length：k，则会产生 k 个等距数据
- rep () 函数

```
1:9
```

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

```
x = 1:9
x
```

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

```
1.5:10
```

```
## [1] 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5
```

```
c(1.5:10,11)
```

```
##  [1]  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 11.0
```

```
prod(1:8)
```

```
## [1] 40320
```

```
seq(1,5)
```

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

```
seq(1,5,by=0.5)
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
seq(1,5,length = 7)
```

```
## [1] 1.000000 1.666667 2.333333 3.000000 3.666667 4.333333 5.000000
```

```
rep(10,5)
```

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

```
rep(c("A","B","C","D"),2)
```

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

```
rep(1:4,times = 3,each =2)
```

```
##  [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

```
rep(1:4,each =2,length = 12)
```

```
##  [1] 1 1 2 2 3 3 4 4 1 1 2 2
```

```
matrix(rep(0,16),nrow = 4)
```

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

```
matrix(rep(0,16),nrow = 4)
```

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

```
matrix(0,nrow = 4,ncol = 4)
```

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

### 基本向量运算

以下函数可用于 vector 变量：

- length：向量中的元素个数
- sum：向量中所有元素求和
- prod：向量中所有元素求积
- cumsum、cumprod：累积相加和累积相乘
- sort：向量中的元素排序
- rank：显示元素排序后的“排序顺位”，输出为向量

```
x=c(1,2.0,3);x
```

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

```
(x=c(1.0,2.3,3))
```

```
## [1] 1.0 2.3 3.0
```

```
x = c(1,2,3)
x + 1
```

```
## [1] 2 3 4
```

```
x - 1.2
```

```
## [1] -0.2  0.8  1.8
```

```
x * 2
```

```
## [1] 2 4 6
```

```
x * x
```

```
## [1] 1 4 9
```

```
y = c(4,5,6,7)
x * y
```

```
## Warning in x * y: longer object length is not a multiple of shorter object
## length
```

```
## [1]  4 10 18  7
```

```
x = c(1,2,3,4)
y = c(5,6,7,8)
y / x
```

```
## [1] 5.000000 3.000000 2.333333 2.000000
```

```
y - x
```

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

```
x ^ y
```

```
## [1]     1    64  2187 65536
```

```
cos(x*pi)+cos(y*pi)
```

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

```
s = c(1,2,3,4,5,6)
length(s)
```

```
## [1] 6
```

```
sum(s)
```

```
## [1] 21
```

```
prod(s)
```

```
## [1] 720
```

```
cumsum(s)
```

```
## [1]  1  3  6 10 15 21
```

```
x = c(1,2,3,4)
y = c(5,6,7)
z = c(x,y)
z
```

```
## [1] 1 2 3 4 5 6 7
```

### 向量的指标用法

一个向量 x 的第 i 个元素可以用 x[i] 表示。

```
x = c(11,12,13)
x[2]
```

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

```
x[4]
```

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

```
x[c(1,3)]
```

```
## [1] 11 13
```

```
x[1:3]
```

```
## [1] 11 12 13
```

```
y = x[1:2]
y
```

```
## [1] 11 12
```

### 基本统计计算

- mean：期望（平均值）
- var：样本方差
- sd：样本标准差

```
x = c(11,12,13)
mean(x)
```

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

```
max(x)
```

```
## [1] 13
```

```
min(x)
```

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

```
var(x)
```

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

```
sd(x)
```

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

```
sum(x)
```

```
## [1] 36
```

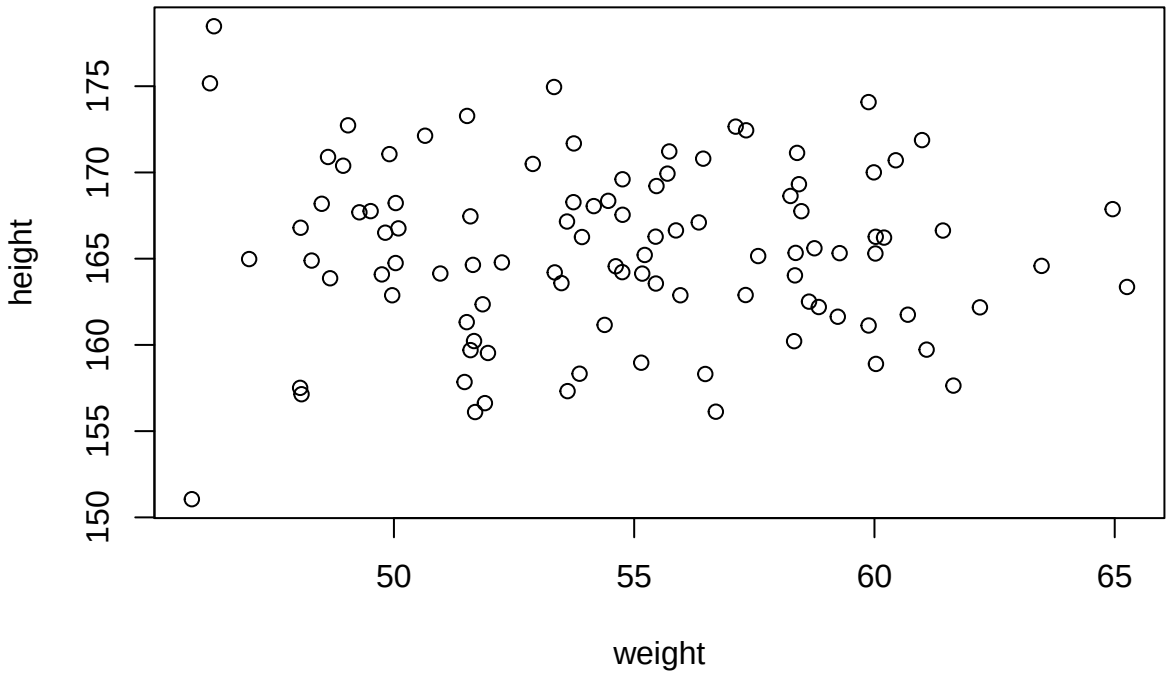
也可以不使用 sd 函数，而是用自定义函数计算标准差：

```
my.sd <- function(y)
{
  n=length(y)
  s=sqrt((sum(y^2)-n*mean(y)^2)/(n-1))
  return(s)
}
my.sd(x)
```

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

模拟 100 个人的身高体重数据（正态分布）

```r
weight =rnorm(100,55,5)
height = rnorm(100,165,5)
plot(weight,height)
```




```r
summary(lm(height~weight))
```

```
##
## Call:
## lm(formula = height ~ weight)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.711  -2.994   0.054   3.021  12.727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 167.49837    6.12071  27.366   <2e-16 ***
## weight       -0.03783    0.11161  -0.339    0.735
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.019 on 98 degrees of freedom
## Multiple R-squared:  0.001171,   Adjusted R-squared:  -0.009021
## F-statistic: 0.1149 on 1 and 98 DF,  p-value: 0.7353
```

### 数据对象

```
x <- seq(0,1,by = 0.2)
y <- seq(0,1,by = 0.2)
y[4]
```

```
## [1] 0.6
```

```
x[3]
```

```
## [1] 0.4
```

```
1 - x[3]
```

```
## [1] 0.6
```

```
y[4] > 1 - x[3]
```

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

### 向量

1. 向量赋值

```
x <- c(1,3,5,7,9)
x
```

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

```
v <- paste("x",1:5,sep="")
v
```

```
## [1] "x1" "x2" "x3" "x4" "x5"
```

2. 向量运算

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

```
## [1]  2 12 30 56 90
```

```
x %*% y
```

```
##      [,1]
## [1,]  190
```

3. 生成有规则序列

```
(t <- 1:10)
```

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

```
(r <- 5:1)
```

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

```
2 * 1:5
```

```
## [1]  2  4  6  8 10
```

```
seq(1,10,2)
```

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

```
seq(1,by = 2,length = 10)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

4. 向量常见函数

```
x <- c(1,3,5,7,9)
length(x)
```

```
## [1] 5
```

```
y <- c(2,6,3,7,5)
sort(y)
```

```
## [1] 2 3 5 6 7
```

```
rev(y)
```

```
## [1] 5 7 3 6 2
```

```
append(y,10:15,after = 3)
```

```
##  [1]  2  6  3 10 11 12 13 14 15  7  5
```

```
sum(x)
```

```
## [1] 25
```

```
max(y)
```

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

5. 向量索引

```
x <- c(1,3,5,7,9)
x[2]
```

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

```
x[c(1,3)] <- c(9,11)
x
```

```
## [1]  9  3 11  7  9
```

```
x[x < 9]
```

```
## [1] 3 7
```

```
y <- 1:10
y[-(1:5)]
```

```
## [1]  6  7  8  9 10
```

**矩阵**

```
matrix(1:12,nrow = 4,ncol = 3)
```

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

```r
matrix(1:12,nrow = 4,ncol = 3,byrow = T)
```

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

```r
(A <- matrix(1:12,nrow = 3,ncol = 4))
```

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

```r
t(A)
```

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

```r
A * A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   16   49  100
## [2,]    4   25   64  121
## [3,]    9   36   81  144
```

```r
A %*% t(A)
```

```
##      [,1] [,2] [,3]
## [1,]  166  188  210
## [2,]  188  214  240
## [3,]  210  240  270
```

```r
diag(A)
```

```
## [1] 1 5 9
```

```r
diag(diag(A))
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    5    0
## [3,]    0    0    9
```

```r
diag(3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

```r
(B <- matrix(rnorm(16),4,4))
```

```
##            [,1]        [,2]       [,3]       [,4]
## [1,]  0.3615026  0.84670864  1.4653460  0.9078788
## [2,]  1.5922525 -0.02679871  0.8492639 -2.2765339
```

```
## [3,] -0.9098172 -1.15760997 -1.5875298  0.3921660
## [4,]  0.9260390  0.34014771  1.1583710  0.9136625
```

```
solve(B)
```

```
##            [,1]       [,2]       [,3]       [,4]
## [1,] -2.2991498 -0.4737069 -1.1954476  1.6173951
## [2,] -2.3628876 -1.1468434 -2.4112177  0.5253418
## [3,]  2.9192986  1.0141891  1.7777307 -1.1368518
## [4,] -0.4912072 -0.3787393 -0.1445506  0.7009497
```

```
(B.eigen <- eigen(B,symmetric = T))
```

```
## $values
## [1]  2.493394  1.353617 -1.496660 -2.689514
##
## $vectors
##            [,1]        [,2]         [,3]       [,4]
## [1,] -0.6889714 -0.07981623  0.688242649 -0.2127671
## [2,] -0.5835978 -0.27341633 -0.706401128 -0.2926680
## [3,]  0.2132177  0.46290284  0.001143505 -0.8603824
## [4,] -0.3731892  0.83940088 -0.165282169  0.3589119
```

```
svd(B)
```

```
## $d
## [1] 3.4679048 2.7533421 0.7844148 0.1675902
##
## $u
##            [,1]       [,2]       [,3]       [,4]
## [1,] -0.3512988 -0.5514050 -0.1536711 -0.7408960
## [2,] -0.6279048  0.6952859  0.2262759 -0.2666690
## [3,]  0.6074264  0.1602265  0.5731773 -0.5261449
## [4,] -0.3366867 -0.4322702  0.7724326  0.3211422
##
## $v
##            [,1]       [,2]        [,3]       [,4]
## [1,] -0.5741825  0.1313533  0.63557199  0.4991082
## [2,] -0.3167061 -0.2971035 -0.68452700  0.5855335
## [3,] -0.6927372 -0.3532476 -0.06142907 -0.6257457
## [4,]  0.3002112 -0.8773209  0.35170655  0.1283893
```

```
dim(A)
```

```
## [1] 3 4
```

```
nrow(B)
```

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

```
det(B)
```

```
## [1] -1.255226
```

```
A[row(A) < col(A)] = 0
A
```

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

```
## [3,]    3    6    9    0
```

```
apply(A,1,sum)
```

```
## [1]  1  7 18
```

```
apply(A,2,mean)
```

```
## [1] 2.000000 3.666667 3.000000 0.000000
```

**数组**

矩阵只能是 2 维的，数组是多维的。一维数组就是向量，二维数组就是矩阵。

```
(xx <- array(1:24,c(3,4,2)))
```

```
## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]   13   16   19   22
## [2,]   14   17   20   23
## [3,]   15   18   21   24
```

```
xx[2,3,2]
```

```
## [1] 20
```

```
xx[2,1:3,2]
```

```
## [1] 14 17 20
```

```
dim(xx)
```

```
## [1] 3 4 2
```

数组的运算和矩阵类似。

**因子**

```
y <- c(" 女"," 男"," 男"," 女"," 女"," 女"," 男"," 女"," 女"," 男")
(f <- factor(y))
```

```
##  [1] 女 男 男 女 女 女 男 女 女 男
## Levels: 男 女
```

```
levels(f)
```

```
## [1] "男" "女"
```

**列表**

如果一个数据对象需要包含不同的数据类型，则可以采用列表（List）

```
x <- c(1,1,2,2,3,3,3)
y <- c("male","female","female","male","female","male","male")
z <- c(80,85,92,76,61,95,83)
(stu <- list(class = x, sex = y, score = z))
```

```
## $class
## [1] 1 1 2 2 3 3 3
##
## $sex
## [1] "male"   "female" "female" "male"   "female" "male"   "male"
##
## $score
## [1] 80 85 92 76 61 95 83
```

```
stu[[3]]
```

```
## [1] 80 85 92 76 61 95 83
```

```
stu$sex
```

```
## [1] "male"   "female" "female" "male"   "female" "male"   "male"
```

**数据框**

数据框（data frame）是一种矩阵形式的数据，但各列可以是不同类型的数据，可以看做是矩阵的推广，类似于关系数据库的形式。

```
(student <- data.frame(class = x, sex = y, score = z))
```

```
##   class    sex score
## 1     1   male    80
## 2     1 female    85
## 3     2 female    92
## 4     2   male    76
## 5     3 female    61
## 6     3   male    95
## 7     3   male    83
```

```
row.names(student) <- c("zhao","qian","sun","li","zhou","wu","zhen")
student
```

```
##      class    sex score
## zhao     1   male    80
## qian     1 female    85
## sun      2 female    92
## li       2   male    76
## zhou     3 female    61
## wu       3   male    95
## zhen     3   male    83
```

```
student[,"score"]
```

```
## [1] 80 85 92 76 61 95 83
```

```
student[,2]
```

```
## [1] male   female female male   female male   male
## Levels: female male
```

```
student$score
```

```
## [1] 80 85 92 76 61 95 83
```

```
student[["class"]]
```

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

```
student[[3]]
```

```
## [1] 80 85 92 76 61 95 83
```

数据框绑定 attach 函数

```
#score
#Error: object 'score' not found
attach(student)
score
```

```
## [1] 80 85 92 76 61 95 83
```

```
detach()
#score
#Error: object 'score' not found
```