R 语言-4

郑泽靖　zzjstat2023@163.com

北京师范大学统计学院

2025 年 11 月 30 日

R 语言中的条件语句

在 R 语言中，常用的条件控制语句是 `if` 语句，它有三种主要形式。

- **(1) 单条件 (if)**：基础判断。
- **(2) 复合条件 (if/else)**：二选一判断。
- **(3) 多条件 (if/else if/else)**：多重分支判断。

单条件语句 (if)

调用格式:

```
if (条件表达式) {
    # 仅当条件表达式为 TRUE 时执行
    程序代码
}
```

执行过程：若“条件表达式”成立（结果为 TRUE），则执行“程序代码”；否则，跳过执行。

单条件语句示例

```r
x <- 1  # 初始值

if (x < 1) {  # 1 < 1 为 FALSE
    x <- x + 1
}
print(x)  # x 仍为 1

if (x < 2) {  # 1 < 2 为 TRUE
    x <- x + 1
}
print(x)  # x 变为 2
```

要点提示：条件表达式必须返回一个 ** 长度为 1 的逻辑值 **（TRUE 或 FALSE）。若输入长度大于 1 的向量，将仅检查第一个元素，并给出警告。

复合条件语句 (if/else)

调用格式:

```
if （条件表达式）{
    程序代码1   # 条件成立时执行
} else {
    程序代码2   # 条件不成立时执行
}
```

注意事项:

- else 必须紧跟在 if 代码块的右花括号 } 之后，并在同一行。

多条件语句 (if/else if/else)

调用格式:

```
if (条件表达式1) {
    程序代码1
} else if (条件表达式2) {
    程序代码2
} else {
    程序代码3   # 以上条件都不满足时执行
}
```

执行过程：程序依次检查条件表达式，一旦发现某个条件成立，则执行相应的程序代码，** 然后跳出整个结构 **。

多条件语句示例：判断数字符号

编写代码判断一个数字是正数、负数还是零。

```r
x <- -5
sign_status <- ""

if (x > 0) {
    sign_status <- "正数"
} else if (x < 0) { # 检查是否为负数
    sign_status <- "负数"
} else {              # 既不大于 0 也不小于 0，则为 0
    sign_status <- "零"
}
print(sign_status) # 输出 "负数"
```

循环语句概述

循环语句用于重复执行一段程序代码，是实现迭代和批量操作的重要手段。R 语言主要提供两种循环语句：

- **for 循环**：适用于已知重复次数或需要遍历向量元素的场景。
- **while 循环**：适用于重复次数不确定，依赖条件表达式来控制终止的场景。

for 循环语句

调用格式：

```
for(i in 向量v) {
    # 循环体 expr
}
```

执行过程：

1. 循环变量 i 依次取向量 v 中的每一个元素值。
2. 每取一个值，就执行一次循环体内的程序代码。

while 循环语句

调用格式:

```
while(条件表达式) {
    循环体
    # 确保循环体内有能改变条件表达式的代码，防止死循环
}
```

执行过程:

1. 计算条件表达式。
2. 若为 TRUE，则运行循环体，然后返回第 1 步。
3. 若为 FALSE，则结束循环。

示例：模拟随机变量 X (掷骰子直到 6 点)

模拟 X: 掷骰子，直到掷出 6 点为止，记录掷骰子的次数 X。

使用 while 循环和 break 语句：

```r
X_observations <- numeric(10) # 存储 10 次观测值
for(i in 1:10) {
    n <- 0  # 实验次数
    while(TRUE) { # 设置无限循环
        n <- n + 1
        y <- sample(1:6, 1)  # 模拟掷骰子
        if (y == 6) {        # 满足终止条件
            X_observations[i] <- n
            break            # 终止 while 循环
        }
    }
}
print(X_observations) # 显示 10 次模拟结果
```

函数 (Function) 定义

函数是封装代码逻辑，实现代码重用和模块化的重要工具。

```
name <- function(arg1, arg2, ...) {
    # 函数体：程序代码
    return(result) # 可选，但推荐使用
}
```

- **name**：自定义函数名。
- **arg1, ...**：函数的输入参数。
- **return(result)**：显式指定函数返回值。若省略 return，默认返回函数体中最后一行代码的计算结果。

函数示例：计算前 n 个自然数之和

使用公式法定义函数 mysum：

```r
mysum <- function(n) {
    # 检查输入是否为正整数
    if (n < 1 | n != round(n)) {
        stop("参数n必须是正整数。")
    }
    s <- n * (n + 1) / 2
    return(s)
}

# 调用示例
print(mysum(50))
# 输出 [1] 1275
```

定积分计算：integrate 函数

R 语言通过内置的 integrate 函数进行一元函数的数值积分。

计算定积分 $\int_0^1 x^2 \, \mathrm{d}x$:

```
# integrate 的第一个参数是一个函数对象
result <- integrate(function(x) x^2,
                    lower = 0, upper = 1)
print(result)
```

运行结果：

$$\$value = 0.3333333 \text{ with absolute error } < 3.7 \times 10^{-15}$$

注意：积分的返回值是一个包含结果和误差信息的列表。

均匀分布 $U(a,b)$ 的期望值

设随机变量 $\xi \sim U(0,60)$，理论期望值 $\mathrm{E}(\xi) = \frac{0+60}{2} = 30$。

使用积分计算 $\mathrm{E}(\xi) = \int_{-\infty}^{\infty} x \cdot f(x)\mathrm{d}x$:

```r
# f(x) 为均匀分布的密度函数 dunif(x, min, max)
result <- integrate(function(x)
                    x * dunif(x, min = 0, max = 60),
                    lower = -Inf, upper = Inf)
print(result$value)
```

计算结果:

$$\mathrm{E}(\xi) \approx 30.00000 \quad \text{(高精度逼近)}$$

概率分布函数概述 (d/p/q/r)

在 R 中，大多数概率分布函数都遵循 **d/p/q/r** 的命名规则：

- `dxxx(x)`: **D**ensity / **D**istribution function (概率密度或概率质量函数)。
- `pxxx(q)`: **P**robability function (累积分布函数 CDF)，表示 $P(X \leq q)$。
- `qxxx(p)`: **Q**uantile function (分位数函数)，求解 $F(x) = p$ 的 x 值。
- `rxxx(n)`: **R**andom number generation (随机数生成函数)。

(xxx 为分布的缩写，如 'norm', 'binom', 'unif' 等)

二项分布（Binomial）示例

- `dbinom(x, size, prob)`
- `pbinom(q, size, prob)`
- `rbinom(n, size, prob)`

```
> dbinom(5, size = 10, prob = 0.5)
# P(X=5) = 0.246
> pbinom(5, size = 10, prob = 0.5)
# P(X<=5) = 0.623
> rbinom(2, size = 10, prob = 0.5)
# [1] 7 8 (示例值)
```

正态分布（Normal）示例

- dnorm(x, mean, sd)
- pnorm(q, mean, sd)
- rnorm(n, mean, sd)

```
> dnorm(0.5, mean = 0, sd = 1)
# 0.3520...
> pnorm(1.96, mean = 0, sd = 1)
# P(Z<=1.96) = 0.975
> rnorm(3, mean = 0, sd = 1)
# [1] -0.62... 1.03... 0.70...
```

练习题：斐波那契数列函数

编写一个名为 `fibonacci` 的 R 函数，该函数接受一个正整数 n 作为输入参数，并返回一个长度为 n 的斐波契数列。

斐波那契数列定义：

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2} \quad (n \geq 2)$$

示例： 如果输入 `n = 5`，则输出应为：

```
[1] 0 1 1 2 3
```

提示：

- 使用 `numeric()` 函数初始化结果向量。
- 使用 `if` 语句处理 $n = 1$ 和 $n = 2$ 的边界情况。
- 使用 `for` 循环来计算 $n \geq 3$ 的值。

Thanks!