R 中的因子变量

周世祥

2015 年 12 月 8 日

因子

统计中的数据变量一般分为三种类型：数值数据，次序数据和分类数据。次序数据和分类数据可以从有限集中取值，但该数据集的数据形式必须按顺序排列。比如，学位的等级，低等，中等，高等；头发的颜色也是一个分类数据。在 R 中次序数据和分类数据的类型称为因子，这些因子的所有可能值构成因子的水平。

为何使用因子

使用因子的理由：一是很多统计模拟的行为依赖于变量的输入输出类型，所以需要区分不同的数据类型；其次，使用因子存储数据非常有效。

除了因子存储的数据类型可取有限个值外，其他方面因子与字符型数据区别不是很大。但 R 处理因子和字符型数据的方式不同。为了产生一个因子，将 factor 命令作用于数据 x。x 的区分值用来表示相应的水平。也可以通过 levels 命令指定水平。用 is.factor(x) 检查一个目标对象是否为因子，用 levels(x) 给出其所有的水平。

```
  hair <- c("blond","black","brown","brown","black","gray","none")
 is.character(hair)
```

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

```
is.factor(hair)
```

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

```
hair <- factor(hair)
 levels(hair)
```

```
## [1] "black" "blond" "brown" "gray"  "none"
```

```
hair <- factor(hair,levels = c("black","gray","brown","blond","white","none"))
 table(hair)
```

```
## hair
## black  gray brown blond white  none
##     2     1     2     1     0     1
```

table 可以用来计算每个因子水平出现的次数。默认情况下，R 按照字母顺序排列一个因子的所有水平。如果你自己指定水平，R 会按照你指定的顺序排列。

注意，若，给出的水平以数字开头，会导致一些意想不到的结果。

```
phys.act <-c("L","H","H","L","M","M")
 phys.act <- factor(phys.act,levels = c("L","M","H"),ordered = TRUE)

 phys.act[2] > phys.act[1]
```

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

通常用略缩字或数值型符号表示一个因子的水平。可以用参数 labels 来改变水平的名字。

```
phys.act <- factor(phys.act,levels = c("L","M","H"),
                   labels = c("Low","Medium","High"),ordered = TRUE)

 table(phys.act)
```

```
## phys.act
##    Low Medium   High
##      2      2      2
```

```
which(phys.act == "High")
```

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

因子的存储方式

R 程序通常以我们赋予的水平报告因子的运行结果，但在程序内部 R 实质上以整数的形式表示因子。

```
hair
```

```
## [1] blond black brown brown black gray  none
## Levels: black gray brown blond white none
```

```
 as.vector(hair)
```

```
## [1] "blond" "black" "brown" "brown" "black" "gray"  "none"
```

```
 as.numeric(hair)
```

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

```
 c(hair,5)
```

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

```
 x <- factor(c(0.8,1.1,0.7 ,1.4,1.4,0.9))
 as.numeric(x) # 不能恢复 x
```

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

```
 as.numeric(levels(x))[x] # 恢复 x
```

```
## [1] 0.8 1.1 0.7 1.4 1.4 0.9
```

```r
as.numeric(as.character(x)) # 恢复 x
```

```
## [1] 0.8 1.1 0.7 1.4 1.4 0.9
```

框架和环境

为了在更复杂的环境下使用 R 程序，理解 R 如何组织已创建和包含对象的知识十分必要。

R 使用框架来组织组织内部创建的对象。框架是一种把对象名称和其 R 表现形式联系起来的工具。环境是一种将其他环境，母环境等相结合的框架。环境是一种嵌套结构，母环境是直接包含当前环境的环境。

当 R 启动时，一个工作空间目录被自动创建，该工作空间被称为全局环境，随后创建的对象在缺省情况下均被放入该空间。当程序包被载入时他们会产生与本身有关的子环境或子空间，该空间路径会加入 R 的搜索路径中。

路径搜索的内容可以用下面方式查询

```r
search()
```

```
## [1] ".GlobalEnv"        "package:stats"     "package:graphics"
## [4] "package:grDevices" "package:utils"     "package:datasets"
## [7] "package:methods"   "Autoloads"         "package:base"
```

当一个函数被调用时，R 会创建相对于当前环境的封闭的新环境。此时函数参数中涉及的对象将被从当前环境传入新环境中。而在新环境中创建的对象不能在母环境中使用.

有效数字

双精度型数字在以 10 为基数的情况下大致有 16 个有效数字，对于整数的运算范围在 $-(2^{53}-1)$ 与 $2^{53}-1$ 之间，大概是 -10^{16} 到 10^{16}, 一旦使用超过了这个范围的数字或者分数，由于舍入误差的原因你将得到一个与原数有一些误差的数字。

例如 1.1 没有一个有限的二进制数与其对应，所以在双精度型下其二进制数大致是 1.00011001100...001，其误差大概是 2^{-53}.

```r
.Machine$integer.max # 能记录的最大整数
```

```
## [1] 2147483647
```

```r
1.2e3 # 科学计数法，不能与指数 e 混淆
```

```
## [1] 1200
```

```r
1/0
```

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

```r
0/0
```

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

```
2^-1074 == 0
```

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

```
1/(2^-1074)
```

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

```
2^1023+2^1022+2^1021
```

```
## [1] 1.572981e+308
```

```
x <- 1+2^-52
x-1
```

```
## [1] 2.220446e-16
```

```
y <- 1+2^-53
y-1
```

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

在双精度型数中，最小的非零正数是 2^{-1074}，最大的数是 $2^{1023}(2-2^{-53})$，有时称其为最大浮点数。

可以用来使 1 区别于 $1+x$ 的最小数 x 是 $2^{-53} \approx 2.220446 * 10^{-16}$，这个数称为机器误差。在以 10 为基数的时候，双精度型大约有 16 个有效数字，指数部分的取值范围可以达到 ∓ 308