R的数据类型

温灿红

中国科学技术大学管理学院

大纲

- 类型变换
- 数值型
- 逻辑性
- 字符型
- 日期时间型
- 因子型

R的数据类型

R中包含有以下几种数据类型：

- 整型(int)：存储整型的数据，如1L
- 数值型/双整型(numeric, double)：存储数值型的数据，如1,1.1
- 逻辑型(logical)：只有两个值TRUE和FALSE, 缺失时为NA。
- 字符型(character)：存储一小段文本，用双引号包住，其中单个元素称之为字符串(string)，如"Hello","1"
- 复数类型(complex)：存储复数，如1+3i
- 日期时间类型(Date, POSIXct, POSIX1t)：存储日期和时间的，如Sys.time()
- 因子类型(factor)：存储分类数据，如性别、学历程度等，factor("male)
- 特殊符号： NA(Not Avaiable), NaN(Not a Number), Inf(infinite), NULL

类型变换

数据的类型转换（一）

- typeof()返回数据类型
- is.foo()判断是否属于某种类型foo，如是，则返回TRUE，否则返回FALSE
- as.foo()强制转换成foo类型

```
typeof(1)
```

```
## [1] "double"
```

```
is.numeric(1)
```

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

```
is.integer(1)
```

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

```
is.integer(1L)
```

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

数据的类型转换（二）

```
is.na(1)
```

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

```
is.na(0/0)
```

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

```
is.na(1/0)
```

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

```
is.infinite(1/0)
```

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

- 为什么1/0不是NA，而0/0是?

数据的类型转换（三）

```
is.character(1)
```

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

```
is.character("1")
```

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

```
is.character("one")
```

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

数据的类型转换（四）

```
as.character(2/3)
```

```
## [1] "0.666666666666667"
```

```
as.numeric(as.character(2/3))
```

```
## [1] 0.6666667
```

```
as.numeric(as.character(2/3)) == 2/3
```

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

- 为什么最后一行返回的是FALSE?

数値型

四则运算及数学函数（一）

```
1+2
```

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

```
1-2
```

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

```
1*2
```

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

```
1/2
```

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

```
2^2   # 幂运算
```

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

四则运算及数学函数（二）

```
5%%2          # 取余运算
```

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

```
5%/%2         # 整除运算
```

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

```
sqrt(2)       # 平方根
```

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

```
exp(1)        # 指数
```

```
## [1] 2.718282
```

```
log10(100)    # 对数
```

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

四则运算及数学函数（三）

```
round(3.14159)      # 四舍五入取整
```

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

```
round(3.14159,2)    # 四舍五入到两位小数
```

```
## [1] 3.14
```

```
floor(3.14159)      # 向下取整
```

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

```
ceiling(3.14159)    # 向上取整
```

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

```
ceiling(-3.14159)   # 向上取整
```

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

四则运算及数学函数（四）

```
pi
```

```
## [1] 3.141593
```

```
sin(pi/2)
```

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

```
tan(pi/4)
```

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

```
asin(1)
```

```
## [1] 1.570796
```

逻辑型

逻辑运算（一）

```
7 > 5
```

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

```
7 < 5
```

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

```
7 >= 7
```

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

```
7 == 5
```

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

```
7 != 5
```

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

逻辑运算（二）

- 判断两个浮点型对象是否完全相同，不能直接采用==和identical()，而应该用all.equal()

```
0.45 == 3*0.15
```

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

```
0.45 - 3*0.15
```

```
## [1] 5.551115e-17
```

```
all.equal(0.45, 3*0.15)
```

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

```
identical(0.45, 3*0.15)
```

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

逻辑运算（三）

```
(5 > 7) & (6*7 == 42)
```

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

```
(5 > 7) | (6*7 == 42)
```

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

```
!(5 > 7)
```

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

字符型数据

字符型数据的处理之字符串长度

- nchar()：计算字符串的长度

```
    nchar("coffee")
```

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

```
    nchar("code monkey")
```

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

```
    nchar("code monkey\t")
```

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

```
    nchar("R统计软件")                    # 以字符个数为单位
```

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

```
    nchar("R统计软件", type = "bytes")  # 以字节为单位
```

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

字符型数据的处理之转换大小写

- 在文本分析中常常需要把大小写的统一成一个字符，然后基于该字符进行词频等的分析。
- toupper()函数把字符型向量内容转为大写，tolower()函数转为小写。

```
a <- "Applied Statistical Software"
tolower(a)
```

```
## [1] "applied statistical software"
```

```
toupper(a)
```

```
## [1] "APPLIED STATISTICAL SOFTWARE"
```

字符型数据的处理之取子串

- substr(x, start, stop)从字符串x中取出从第start个到第stop个的子串
- substring(x, start)可以从字符串x中取出从第start个到末尾的子串

```
phrase = "我们一起上实用统计软件课程"
substr(phrase, 1, 4)
```

```
## [1] "我们一起"
```

```
substr(phrase, nchar(phrase)-3, nchar(phrase))
```

```
## [1] "软件课程"
```

```
substring(phrase, nchar(phrase)-3)
```

```
## [1] "软件课程"
```

```
substr(phrase, nchar(phrase)+1, nchar(phrase)+10)
```

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

字符型数据的处理之替代（一）

- substr()替代特定位置

```
substr(phrase, 1, 1) = "你"
phrase
```

```
## [1] "你们一起上实用统计软件课程"
```

```
substr(phrase, 100, 1004) = "我是多余的"
phrase
```

```
## [1] "你们一起上实用统计软件课程"
```

```
substr(phrase, 1, 2) = "大家"
phrase
```

```
## [1] "大家一起上实用统计软件课程"
```

```
substr(phrase, 1, 2) = "我"
phrase
```

```
## [1] "我家一起上实用统计软件课程"
```

字符型数据的处理之替代（二）

- gsub(pattern, replacement, x)：通过在x中匹配找到与pattern对应的字符并替换成replacement，常常用在文本数据的清洗过程。

```
gsub("软件", "算法", phrase)
```

```
## [1] "我家一起上实用统计算法课程"
```

```
gsub("实用", "", phrase)
```

```
## [1] "我家一起上统计软件课程"
```

```
salary <- c("22万", "30万", "50万", "120万", "11万")
salary0 <- gsub("万","0000", salary)
mean(as.numeric(salary0))
```

```
## [1] 466000
```

字符型数据的处理之合并

- paste()：连接两个字符型对象，默认用空格连接。

```
    paste("Spider", "Man")                                  # 默认方式
```

```
## [1] "Spider Man"
```

```
    paste("Spider", "Man", sep="-")                         # 以破折号连接
```

```
## [1] "Spider-Man"
```

```
    paste("Spider", "Man", "does whatever", sep=", ")       # 以逗号加空格连接
```

```
## [1] "Spider, Man, does whatever"
```

```
    paste("Spider", "Man", sep = "")                        # 两个字符直接相连，中间没
```

```
## [1] "SpiderMan"
```

```
    paste0("Spider", "Man")                                 # 更有效的方式
```

```
## [1] "SpiderMan"
```

字符型数据的处理之拆分

- strsplit()：拆分两个及以上的字符型对象。

```
cities <- "北京，上海，广州，深圳，合肥"
strsplit(cities, split="，")[[1]]
```

```
## [1] "北京" "上海" "广州" "深圳" "合肥"
```

```
course <- "实用的统计的软件"
strsplit(course, split="的")[[1]]
```

```
## [1] "实用" "统计" "软件"
```

```
strsplit("实用统计软件", split="")[[1]]       # 仅仅是把每一个字符分开
```

```
## [1] "实" "用" "统" "计" "软" "件"
```

```
jiebaR::segment("早上好，我们一起上统计软件课程!", jiebaR::worker())  # 中文分
```

```
## [1] "早上好" "我们"   "一起"   "上"     "统计"   "软件"   "课程"
```

字符型数据的处理之正则表达式（一）

- 正则表达式(regular expression)是一种匹配某种字符串模式的方法。 包括
  - 元字符： *, ^, $, \d等等
  - 普通字符： a-z, A-Z, 0-9, 空格等
- 可以从字符串中查找某种模式的出现位置，或替换某种模式，如 grep(), grepl(), sub(), gsub()
- 例如:
  - [0-9]表示数字，等价于\d
  - [a-zA-Z0-9]表示数字和所有的英文字母，等价于\w和[:alnum:]
  - [:cntrl:]表示ASCII控制字符
  - [:punct:]表示既不属于[:alnum:]也不属于[:cntrl:]的
  - [:space:]表示任意空格
  - + 表示一个或多个，？表示零个或一个，^x 表示除x外的任意字符

字符型数据的处理之正则表达式（二）

```
    gsub("[ae]", "o", "Give me a break")
```

```
## [1] "Givo mo o brook"
```

```
    gsub('[[:space:]]+', ' ', 'Give    me a break')
```

```
## [1] "Give me a break"
```

```
    strsplit("For tough problems, you need R.", split = " ")[[1]]
```

```
## [1] "For"       "tough"     "problems," "you"       "need"      "R."
```

```
    strsplit("For tough problems, you need R.", split = "[[:space:]]|[[:punct:]]")
```

```
## [1] "For"      "tough"    "problems" ""         "you"      "need"     "R"
```

```
    grep("[[:digit:]]{3}", c("1", "12", "123", "1234"))
```

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

日期时间型

日期时间类型数据

- 在**R**中，日期可以保存为Date类型， 一般用整数保存，数值为从1970-1-1经过的天数。
- R中用一种叫做POSIXct和POSIXlt的特殊数据类型保存日期和时间，可以仅包含日期部分，也可以同时有日期和时间。
    - POSIXct把日期时间保存为从1970年1月1日零时到该日期时间的时间间隔秒数
    - POSIXlt把日期时间保存为一个包含年、月、日、星期、时、分、秒等成分的列表

例子

- 获取当前日期

```
Sys.Date()
```

```
## [1] "2024-02-29"
```

```
class(Sys.Date())
```

```
## [1] "Date"
```

- 获取当前时间

```
Sys.time()
```

```
## [1] "2024-02-29 11:31:53 CST"
```

```
class(Sys.time())
```

```
## [1] "POSIXct" "POSIXt"
```

从字符串生成日期数据

- as.Date()可将字符型数据转换成日期型数据
- 通过format =指定输入的格式，默认的格式中分隔符为-或/
- 如果不是标准格式，可通过以下方式来指定

代码	意义
%d	日
%m	月（数字格式）
%b	月（英文简称）
%B	月（英文全称）
%y	年（两位）
%Y	年（四位）

- 提取组成部分：weekdays, months, days, quarters

例子

```
as.Date("2021/9/10")
```

```
## [1] "2021-09-10"
```

```
as.Date("2021-09-10")
```

```
## [1] "2021-09-10"
```

```
as.Date("9/10/2021", format = "%m/%d/%Y")
```

```
## [1] "2021-09-10"
```

```
weekdays(as.Date("2021/9/10"))
```

```
## [1] "星期五"
```

```
quarters(as.Date("2021/9/10"))
```

```
## [1] "Q3"
```

从字符串生成时间数据

- as.POSIXlt() 或 as.POSIXct()

```
as.POSIXlt("2021-9-10 18:47:22")
```

```
## [1] "2021-09-10 18:47:22 CST"
```

```
as.POSIXct("2021-9-10 18:47:22")
```

```
## [1] "2021-09-10 18:47:22 CST"
```

- 提取成分：

```
y <- as.POSIXlt("2021-9-10 18:47:22")
cat(1900+y$year, y$mon+1, y$mday, y$hour, y$min, y$sec, '\n')
```

```
## 2021 9 10 18 47 22
```

日期时间类型数据的运算

```
as.Date("2021-9-10") + 10
```

```
## [1] "2021-09-20"
```

```
as.Date("2021-9-10") - 10
```

```
## [1] "2021-08-31"
```

```
as.POSIXlt("2021-9-10 18:47:22") - 30
```

```
## [1] "2021-09-10 18:46:52 CST"
```

```
difftime(as.Date("2021-9-10"), as.Date("2020-9-10"), units = 'days')
```

```
## Time difference of 365 days
```

```
difftime(as.POSIXlt("2021-9-10 18:47:22") , as.POSIXlt("2021-8-10 10:47:22") , uni
```

```
## Time difference of 31.33333 days
```

因子型

因子类型数据（一）

- **R**中用因子来存储分类变量，如性别、学历程度等。分类数据分为有序和无序的:
  - 有序变量是指每一类之间有一个大小顺序关系，如打分结果“A”，“B"，” C，“D"，我们仅仅知道“A”>“B">” C>“D"，但不知道每一类之间的差异是多少，即不能进行如下运算"A"-"B"。
  - 无序变量是指每类之间没有顺序关系，如性别，职业等

```
factor(c("男", "女"))
```

```
## [1] 男 女
## Levels: 男 女
```

```
factor(LETTERS[1:3])
```

```
## [1] A B C
## Levels: A B C
```

```
factor(LETTERS[1:3], ordered = TRUE)
```

```
## [1] A B C
## Levels: A < B < C
```

因子类型数据（二）

- 因子的levels()（水平值）属性是一个映射， 把整数值1,2,映射成这些水平值，因子在保存时会保存成整数值1,2,等与水平值对应的编号。
- 可以节省存储空间， 在建模计算的程序中也比较有利于进行数学运算。

```
levels(factor(c("男", "女")))
```

```
## [1] "男" "女"
```

```
factor(LETTERS[1:3], ordered = TRUE, levels = c("C","B","A"))
```

```
## [1] A B C
## Levels: C < B < A
```

变量（一）

- 上面我们讲到的基本都是常量，即直接写在命令行中的量。
- 下面我们讨论变量，变量是用来存储输入的值或者计算得到的值，前面提到的所有的数据类型都可以保存。
- 变量必须有变量名，在**R**中，变量名是由字母、数字、下划线和句点有机组成，命名规则有：
    - 变量名的第一个字符不能取为数字。
    - 变量名区分大小写，如a和A是两个不同的变量名。
- 变量名例子： `a, a1, app.stat, app_stat, AppStat`

变量（二）

- 用<-赋值的方法定义变量。<-也可以写成=，但是更推荐用<-。

```
r <- 2
area <- pi * r^2
area
```

```
## [1] 12.56637
```

```
area <- 10
area
```

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

谢 谢