R基础学习-控制流、函数、pply族

林静

Sunday, November 20, 2016

控制流

在正常情况下，R程序中的语句是从上至下顺序执行。但有时你可能希望重复执行某些语句，或仅在满足特定条件的情况下执行另外的语句。
这时候呢，就是控制流结构发挥作用的地方了。
控制流结构包括两个部分：条件、循环。
statement:语句，一条或一组复合语句（包含在{}里的一组语句，使用分号隔开）
cond:条件，最终被解析为真或假
expr:是一条数值或字符串的求值语句
seq:是一个数值或字符串序列

条件执行

条件执行结构，一条或一组语句仅在满足一个指定条件时执行。条件执行结构包括：if-else、ifelse、switch。

if-else结构

if (cond) statement if (cond) statement1 else statement2

ifelse结构

ifelse(cond,statement1,statement2)

switch结构

switch(expr,……)类似开关 eg:switch(“happy”,happy=“i am glad you are happy”,afraid=“there is nothing to fear”,sad=“cheer up”,angry=“calm down now”)

循环执行

for结构

for结构重复地执行一个语句，直到某个变量的值不再包含在序列seq中为止。 for (variable in seq) statement eg:for (i in 1:10) print(“Hello”)

while结构

while结构重复地执行一个语句，直到条件不为真为止 while (cond) statement eg:i<-10 while (i>0) {print(“Hello”),i=i-1}

function

编写函数与调用函数

编写函数：

myfunction<-function(arg1,arg2=xx,……){ statements return(object) }

tips：1、函数中的对象只在函数内部使用 2、使用一个表达式来捕获错误输入的参数值（类型不符）通常来说是一个好主意(warning()、message()、stop()) 3、return意味着函数将返回return(object)的形式 4、arg2=xx，指的是默认情况下为xx类型

eg:输出指定的日期格式

```
mydate<-function(type="long"){
switch(type,long=format(Sys.time(),"%A %B %d %Y"),short=format(Sys.time(),"%m-%d-%y"),cat(type,"is not
 a recognized type\n"))
}
mydate1<-function(type="long"){
switch(type,long=format(Sys.time(),"%A %B %d %Y"),short=format(Sys.time(),"%m-%d-%y"),warning("is not
 a recognized type"),call.=TRUE)
}
```

type="long",指的是默认情况下为long类型 cat()仅在输入的日期格式类型不匹配时执行，用来捕获错误输入的参数值

调用函数：

```
mydate()
```

```
## [1] "星期日 十一月 20 2016"
```

```
mydate("long")
```

```
## [1] "星期日 十一月 20 2016"
```

```
mydate("short")
```

```
## [1] "11-20-16"
```

```
mydate("foolish")
```

```
## foolish is not a recognized type
```

```
mydate1("foolish")
```

```
## Warning in mydate1("foolish"): is not a recognized type
```

一个练习

给定一个N样本的数据，求样本均值与标准差,并输出

```
mystats<-function(x){
center<-median(x)
spread<-sd(x)
result<-list(center=center,spread=spread)
return(result)
}
x<-rnorm(100)
x
```

```
##   [1]  0.571830034  2.157314599  0.561336583 -1.874718175  0.503240186
##   [6] -0.753524030  1.903481117  1.371445943  0.357564045 -0.003399645
##  [11] -1.229453123  2.357407628  0.072369375 -0.551610322 -0.344544482
##  [16]  0.707347274  0.188896211 -0.685907433 -0.150490723  1.008400708
##  [21]  1.161569560  0.564662611  0.729560656 -1.704272581  0.581224126
##  [26] -0.158393963 -0.653136085 -0.745896009 -0.581393119  0.474308418
##  [31] -0.311746694 -1.016254581 -0.788270604  0.648497484  1.122215132
##  [36] -0.366866985 -0.418889340  1.557058540  0.457785037 -0.111345506
##  [41]  0.029174191  0.630712075 -0.442653447  1.312260395  1.782818278
##  [46]  0.670994030 -1.616341804 -0.782929061 -1.095317123 -0.941562635
##  [51] -1.120277428  1.258859163 -0.966648087 -1.015632495 -0.387249484
##  [56]  1.011815188 -1.394521940  0.481623322  1.687962920 -0.828618570
##  [61]  1.033324297  0.620993317  0.772140989 -0.604576255  0.496940744
##  [66] -0.177588007 -0.114657560  0.301904477 -1.317011123 -0.636297109
##  [71]  0.654560874 -0.765921177 -1.254641997  0.713408422 -0.154672948
##  [76]  1.759488906 -0.124937226 -0.228205302  0.484379194 -0.875067930
##  [81]  0.079595780  1.466603619  0.216532946  0.191567915 -0.753109664
##  [86] -0.319940460  0.905805815 -0.527142182 -0.508363334 -1.243982710
##  [91]  0.716604011  0.988892958 -0.645956310  1.147931283 -0.505523916
##  [96] -0.846197171  0.348918052 -0.017626714 -1.299034817  0.726378284
```

```
mystats(x)
```

```
## $center
## [1] -0.01051318
##
## $spread
## [1] 0.9288755
```

pply族

*R*中利用*pply*族（*apply*、*lapply*、*sapply*、*tapply*）、*by*、*aggregate*等函数进行分组/分类统计 pply簇相比与for循环在R中速度要快，并且能够有效的缩减代码量和使代码更加容易修改。

apply

apply函数是对一个数组按行/列进行function计算，apply的返回值是一个向量 apply(X,MARGIN,FUN,……)

X为数据框类型 MARGIN为一个向量，1表示行，2表示列，c(1,2)表示行、列都计算 FUN即为function

```
apply(mtcars[,c(1,3,4,5,6,7)],2,median)
```

```
##     mpg    disp      hp    drat      wt    qsec
##  19.200 196.300 123.000   3.695   3.325  17.710
```

lapply、sapply

lapply函数是对一个list对象中的每个元素进行function计算,lapply的返回值是一个和X有相同长度的list lapply(X,FUN,……)

X为list类型，R可自动转换 FUN即为function

sapply是lapply的一个特殊情形 sapply(X,FUN,…….,simplify=TRUE/FALSE)

X为list类型，R可自动转换 FUN即为function 若simplify=FALSE,返回值也是list，则sapply等价于lapply 若simplify=TRUE,返回向量

tapply

tapply函数是根据INDEX进行function计算 tapply(X,INDEX,FUN,simplify=TRUE/FALSE)

X通常是一个向量 INDEX是一个list,list中的每一个元素是与X有同样长度的因子 FUN即为function 若simplify=FALSE,返回值是list 若simplify=TRUE,返回向量

```
tapply(mtcars$mpg,mtcars$cyl,mean)
```

```
##        4        6        8
## 26.66364 19.74286 15.10000
```

利用tapply还可以实现类似于excel里的数据透视表的功能

```
tapply(mtcars$mpg,list(mtcars$cyl,mtcars$gear),mean)
```

```
##       3      4    5
## 4 21.50 26.925 28.2
## 6 19.75 19.750 19.7
## 8 15.05     NA 15.4
```

by结构也可以实现类似于tapply的效果，但结果不够美观 with(X,by(X,INDEX,FUN))

```
with(mtcars,by(mtcars$mpg,mtcars$cyl,mean))
```

```
## mtcars$cyl: 4
## [1] 26.66364
## ------------------------------------------------------------
## mtcars$cyl: 6
## [1] 19.74286
## ------------------------------------------------------------
## mtcars$cyl: 8
## [1] 15.1
```

aggregate结构也可以实现类似于tapply的效果 aggregate(X,by=list(),FUN,simplify=TRUE)

```
aggregate(mtcars$mpg,by=list(mtcars$cyl,mtcars$gear),mean)
```

```
##   Group.1 Group.2      x
## 1       4       3 21.500
## 2       6       3 19.750
## 3       8       3 15.050
## 4       4       4 26.925
## 5       6       4 19.750
## 6       4       5 28.200
## 7       6       5 19.700
## 8       8       5 15.400
```