Циклы в R, хотя лучше их избегать
========

Посмотрим, как записывается в R for цикл

```
for(var in seq) { expr }
```

seq обозначает некоторый вектор, содержащий набор значений, который пробегает счетчик цикла var.

Примеры того, как может выглядеть вектор seq.

```
1:10
c(1,7,19)
```

Фигурные скобки { } используются, чтобы "упаковать" несколько операторов в один. Они используются в if операторах (смотри ниже) с той же целью.

В некоторых ситуациях фигурные скобки можно опускать. Тем не менее, не стоит пижонить, начинающим рекомендуется всегда ставить скобки.

Простой пример цикла
Пример 1.

```
> for (i  in 1:10) print(i)
```

В R часто существуют способы написания команд, которые предпочтительнее, чем использование циклов. Рекомендуется использовать матричные операции вместо операторов цикла.

Другие варианты оператора цикла:

```
repeat <expression>          ## где-то внутри должен быть выход из цикла
while (x>0) <expression>
```

Здесь под <expression> имеется в виду R statement, or a sequence of statements, заключенное в скобки.

Пример 2.

Сначала посмотрим вариант суммирования трех чисел 31, 51 and 91.

```
> answer <- 0
> for (j in c(31,51,91)){answer <- j+answer}
> answer
[1]  173
```

Вычисления последовательно изменяют объект answer, используя элементы вектора (31,51,91). В рабочей переменной последовательно накапливается сумма. Сначала полагаем j=31, и объекту answer присваивается значение 31 + 0 = 31. Затем j=51, и answer становится равным 51 + 31 = 82. Наконец, j=91, а answer становится равным 91 + 81 = 173. На этом процедура заканчивает свою работу, но результат на экране не высвечивается, что свойственно интровертированному характеру пакета R. Чтобы посмотреть на значение answer, набираем answer и нажимаем клавишу Enter.

Приведенный выше вариант скрипта является нерациональным в R. Эти же вычисления можно проделать проще и эффективнее:

```
> sum(c(31,51,91))
[1] 173
```

Еще раз. Продвинутые пользователи R стараются избегать циклов. Как и в рассмотренном только что примере, часто существует лучший вариант действий.

Пример 3.

Сосчитаем значения температуры по Фаренгейту, которые соответствуют температурам 25, 26, ..., 30 градусов по Цельсию:

Первый вариант использует for цикл.

```
> # Celsius to Fahrenheit
> for (celsius in 25:30)
+ print(c(celsius, 9/5*celsius + 32))
[1]    25 77
[1]    26.0 78.8
[1]    27.0 80.6
[1]    28.0 82.4
[1]    29.0 84.2
[1]    30 86
```

Второй вариант более экономен.

```
> Celsius <- 25:30
> fahrenheit <- 9/5*celsius+32
> conversion <- data.frame(Celsius=celsius, Fahrenheit=fahrenheit)
> print(conversion)
Celsius        Fahrenheit
25             77.0
26             78.8
27             80.6
28             82.4
29             84.2
30             86.0
```

Пример 4. Возможная ошибка

Задача.

Имеется последовательность чисел от 1 до 10. Написать программу для нахождения скользящего среднего данной последовательности при ширине окна, равной 5.

Сделайте то же самое при нечетной ширине окна.

Подзадача 1.

Найти ошибку в программе

```
wind.1 <- 5                 #   ширина окна
xxx <- c(1:10)              #   сглаживаемый ряд

xxx.smooth <- xxx

for (i in 1:wind.1-1)
  {     xxx.smooth <- xxx.smooth + xxx[-1:-i]   }

movavr.centr <- c(rep(NA,2), xxx.smooth[1:(length(xxx.smooth)-
wind.1+1)]/wind.1, rep(NA,2))
```

Комментарий к используемым командам.

xxx[-1:-9] == используй сокращенный вариант вектора xxx без первых девяти координат.

length == функция, которая вычисляет длину вектора.

Если вычисляем не длину вектора, а размерность data.frame, применяем функцию dim.

rep(a,b) == создать вектор, последовательно b раз склеивая вектор a, например

rep(0,10) создаст вектор из 10 нулей.

rep(c(5,7), 3) создаст вектор 5 7 5 7 5 7

Подзадача 2.

Найти ошибку в программе, воспользовавшись распечаткой результатов выполнения программы

```
rm(wind.1, xxx.smooth, xxx, movavr.centr, i )

wind.1 <- 5          #   ширина окна
xxx <- c(1:10)       #   сглаживаемый ряд
```

```
xxx.smooth <- xxx

print(c(' xxx.smooth0', xxx.smooth))            #  начальное значение вектора

for (i in 1:wind.1-1)
  {
   print(c(i, ' xxx.smooth1', xxx.smooth))      #  первое слагаемое
   print(c(i, 'xxx', xxx[-1:-i]))               #  второе слагаемое
   xxx.smooth <- xxx.smooth + xxx[-1:-i]
   print(c(i, ' xxx.smooth2', xxx.smooth))      #  сумма
  }
movavr.centr <- c(rep(NA,2), xxx.smooth[1:(length(xxx.smooth)-
wind.1+1)]/wind.1, rep(NA,2))
```

Диагноз. Оказывается, ошибка в строке `for (i in 1:wind.1-1)`, поскольку вычитание производится не только из последней переменной `wind.1` , но из всего вектора значений индексов `1:wind.1` . Причина ошибки: оператор " **:** " имеет очень высокий приоритет, выше чем у арифметических операций, а значит сначала создается вектор, а затем из его координат вычитается единица!

Исправление. Надо заменить строку

```
for (i in 1:wind.1-1)
```

на

```
for (i in 1 : (wind.1-1))
```

Комментарий. Не стесняйтесь ставить скобки, когда есть сомнения. В первую очередь это касается фигурных скобок при использовании операторов `for` и `if` .

Окончательный вариант

```
wind.1 <- 5          #   ширина окна
xxx <- 1:10  #   сглаживаемый ряд

xxx.smooth <- xxx

for (i in 1:(wind.1-1))
  {   xxx.smooth <- xxx.smooth + xxx[-1:-i]   }

movavr.centr <- c(rep(NA,2), xxx.smooth[1:(length(xxx.smooth)-
wind.1+1)]/wind.1, rep(NA,2))

movavr.right <- c(rep(NA, wind.1-1),
       xxx.smooth [1 : (length(xxx.smooth)-wind.1+1)]/wind.1 )
```

2.11 Примеры

1. Предскажите результат каждой последовательности команд, приведенной ниже. Проверьте себя, выполнив вычисления.

a)
```
answer <- 0
for (j in 3:5){ answer <- j+answer }
```

b)
```
answer <- 10
for (j in 3:5){ answer <- j+answer }
```

c)
```
answer <- 10
for (j in 3:5){ answer <- j*answer }
```

2. Посмотрите описание функции prod(), и используйте функцию prod(), чтобы решить задачу примера **1**(c) выше.

3. Найдите сумму всех натуральных чисел от 1 до 100 двумя способами, используя оператор цикла for и функцию sum.

4. Найдите произведение всех натуральных чисел от 1 до используя оператор цикла for и функцию prod.

5. Объем сферы радиуса *r* определяется по формуле 4*pi*r^3/3. Для последовательности сфер с радиусами 3,4,5, ..., 20 найдите объемы и выдайте результаты в виде таблицы. Используйте метод из раздела 2.1.5 для построения таблицы данных со столбцами радиус и объем.

6. Use sapply() to apply the function is. factor to each column of the supplied data frame tinting.

For each of the columns that are identified as factors, determine the levels. Which columns are ordered factors?

[Use is.ordered()].

Замечание.

Особенно легко заменить цикл на матричную операцию, если ко всем элементам объекта применяется одна и та же операция. Например, если необходимо заменить все отрицательные элементы матрицы на нули, можно организовать цикл, который переберет все строки, затем еще один цикл, просматривающий все столбцы, по дороге сравнивая все элементы, один за другим с нулем.

Быстрее и проще сделать так

```
xmat[xmat < 0] <- 0
```

Описание циклов в руководстве пользователя

Control

Control Flow

Description

These are the basic control-flow constructs of the R language. They function in much the same way as control statements in any algol-like language.

Usage

```
if(cond) expr
if(cond) cons.expr else alt.expr
for(var in seq) expr
while(cond) expr
repeat expr
break
next
```

Details

Note that `expr` and `cons.expr`, etc, in the *Usage* section above means an expression in a formal sense. This is either a simple expression or a so called compound expression, usually of the form `{ expr1 ; expr2 }`.

Note that it is a common mistake to forget putting braces (`{ .. }`) around your statements, e.g. after `if(..)` or `for(....)`. For that reason, one (somewhat extreme) attitude of defensive programming uses braces always, e.g. for if clauses.

The index `seq` in a for loop is evaluated at the start of the loop; changing it subsequently does not affect the loop.

See Also

`Syntax` for the basic R syntax and operators, `Paren` for parentheses and braces; further, `ifelse, switch`.

Examples

```
for(i in 1:5) print(1:i)

for(n in c(2,5,10,20,50)) {
x <- rnorm(n)
cat(n,":", sum(x^2),"\n")
}
```