Основы программирования в R

R как калькулятор, переменные и типы переменных в R

Алла Тамбовцева

Содержание

R как калькулятор

Сложение, умножение, деление:

```
12 + 18
```

```
## [1] 30
```

```
5 * 3
```

```
## [1] 15
```

```
5 / 8
```

```
## [1] 0.625
```

Возведение в степень (два способоа):

```
7 ^ 2
```

```
## [1] 49
```

```
7 ** 2
```

```
## [1] 49
```

Квадратный корень:

```
sqrt(16)
```

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

```
sqrt(24)
```

```
## [1] 4.898979
```

Корень произвольной степени:

```
16 ^ (1/4)
```

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

Округление до целого числа:

```
round(5.5)
```

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

Округление до произвольного знака после запятой (в примере — до первого):

```
round(5.56, 1)
```

```
## [1] 5.6
```

Математические константы π и e:

```
pi
```

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

```
exp(1)
```

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

Натуральный логарифм:

```
log(100)
```

```
## [1] 4.60517
```

Десятичный логарифм:

```
log10(100)
```

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

Логарифм по произвольному основанию `base`:

```
log(100, base = 4)
```

```
## [1] 3.321928
```

Переменные в R

Названия переменных в R могут содержать буквы, цифры, точки и знаки подчёркивания, при этом название переменной не может начинаться с цифры. Название переменной не должно совпадать со служебными словами (операторами) в R: `if`, `else`, `for`, `while` и другимим.

Оба оператора `<-` и `=` используются для присваивания, но `<-` является общепринятым в R, и знак `=` для присваивания в хорошем коде и официальных тьюотриалах мы не увидим.

```
age.1 <- 26
age.1
```

```
## [1] 26
```

Иногда оператор `<-` «переворачивают» и превращают в `->`, но вообще в программировании принято, чтобы переменная стояла слева, а присваимое значение — справа, поэтому такое можно увидеть довольно редко:

```
2 -> x
```

Мы можем изменить значение переменной и сохранить её под тем же именем:

```
x <- x + 1
x
```

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

Типы переменных

Основными типами переменных в R являются:

- числовой (`numeric`);
- целочисленный (`integer`);
- текстовый (`character`);
- логический (`logical`) - только два значения: `TRUE` и `FALSE`.

Важно: В дробных числах в R в качестве разделителя используется точка.

Создадим переменную `x1` и присвоим ей значение 9.5.

```
x1 <- 9.5
```

Проверим, является ли `x1` числом:

```
is.numeric(x1)  # yes
```

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

Проверим, является ли `x1` целым числом:

```
is.integer(x1)  # no
```

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

Аналогичным образом после префикса `is.` можем поставить `character` и `logical`.

Создадим переменную `x2`:

```
x2 <- "welcome"
```

Узнаем, какого она типа:

```
class(x2)
```

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

Узнаем более конкретный тип:

```
typeof(x2)
```

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

Важно. Если забыли, что делает та или иная функция, можно спросить это у R:

```
?class
```

```
help(class)
```

Тип переменной можно менять — выполнять приведение типов. Например, превратим строку “2” в число 2:

```
two <- "2"
as.numeric(two)
```

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

Логические переменные легко превращаются в числовые:

```
u <- TRUE
e <- FALSE
```

```
as.numeric(u)
```

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

```
as.numeric(e)
```

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

Конечно, не у любой переменной мы можем поменять тип. Например, строку “abc” превратить в число не получится:

```
as.numeric("abc")
```

```
## Warning:                        NA
```

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

Числовые переменные

С числовыми переменными можно делать то же, что и с числами:

```
a <- 25
b <- 15
sum_ab <- a + b
sum_ab
```

```
## [1] 40
```

```
power_ab <- a ^ b
power_ab
```

```
## [1] 9.313226e+20
```

Здесь `e` — это число 10. Запись `9.313226e+20` означает, что число 9.313226 надо умножить на 10^{20}. Если, напротив, R нужно было бы выдать очень маленькое число, 10 стояло бы в отрицательной степени:

```
1 / a ^ b
```

```
## [1] 1.073742e-21
```

Текстовые переменные (строки)

Что можно делать с текстовыми переменными? Например, в текстовых переменных можно заменять одни символы на другие. Для этого существует функция `sub()`. На первом месте указываем, что заменяем, на втором — на что заменяем, на третьем — где заменяем:

```
group <- "group#1 group#2 group#3"
sub("#", "-", group)
```

```
## [1] "group-1 group#2 group#3"
```

Однако функция `sub()` позволяет изменить только первое совпадение. Для того, чтобы заменить все встречающиеся в тексте символы, нужно воспользоваться `gsub()`:

```
gsub("#", "_", group)
```

```
## [1] "group_1 group_2 group_3"
```

Логические выражения

Необходимы для проверки или формулировки условий.

```
x <- 2
y <- 10
```

Привычные операции сравнения:

```
x > y
```

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

```
x < y
```

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

```
x <= y
```

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

Для проверки равенства, как обычно в программировании, используется двойной знак равенства:

```
x == y
```

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

Для отрицания равенства используется оператор `!=`:

```
x != y
```

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

Для проверки одновременного выполнения условий используется оператор `&` (аналог `and` в Python):

```
x & y < 5
```

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

Для проверки выполнения хотя бы одного условия используется оператор `|` (аналог `or` в Python):

```
x | y < 5
```

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

Для проверки выполнения ровно одного условия используется функция `xor()`, так как в R нет соответствующего оператора:

```
xor(x < 5, y < 5)
```

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

Чтобы заметить разницу между операциями «хотя бы одно верно» и «ровно одно верно», изменим значение `y` так, чтобы оба условия `x < 5` и `y < 5` были верными:

```
y <- 3
```

Теперь «хотя бы одно верно» истинно (оба верны), а «ровно одно верно» ложно (более одного верно):

```
x | y < 5
```

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

```
xor(x < 5, y < 5)
```

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