Основы программирования в R

Ввод и вывод в R

Алла Тамбовцева, НИУ ВШЭ

Содержание

Вывод сообщения на экран

В R для вывода сообщения на экран есть две функции: `print()` и `cat()`. С `print()` мы уже знакомы:

```
print(8)
```

```
## [1] 8
```

```
print("hello")
```

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

Или так:

```
x <- 5
print(x)
```

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

Как можно заметить, когда с помощью `print()` на экран выводится текст, он печатается в кавычках. При желании это можно изменить, добавив аргумент `quote=FALSE`:

```
print("hello", quote=FALSE)
```

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

А что делает `cat()`? Тоже выводит сообщение на экран, только сразу без кавычек:

```
cat("hello")
```

```
## hello
```

Видно, что `cat()` при печати к тому же не выводит номер строки. Но это всё мелочи. В чём состоит принципиальное отличие `print()` от `cat()`?

Во-первых, `cat()`, в отличие от `print()`, умеет «склеивать» то, что мы подаём ему на вход. Сравним:

```
print(1, 8) # выводится только первый элемент
```

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

```
cat(1, 8) # выводятся все элементы
```

```
## 1 8
```

Во-вторых, `cat()` умеет работать только с одномерными объектами (числа, строки, вектора), а `print()` может вывести на экран что угодно. Рассмотрим пример с векторами, где обе функции сработают:

```
v <- c(3, 6, 9)
print(v) # ок
```

```
## [1] 3 6 9
```

```
cat(v) # ок
```

```
## 3 6 9
```

А теперь рассмотрим пример с матрицами, где `print()` сработает, а `cat()` выведет все элементы матрицы в одну строчку:

```
m <- matrix(3, nrow = 2, ncol = 3)
print(m) # сохранил матрицу
```

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

```
cat(m) # склеил все элементы в одну строку
```

```
## 3 3 3 3 3 3
```

И, наконец, пример со списками, когда `print()` сработает, а `cat()` — нет:

```
L <- list(c(1, 2, 3), c(0, 1))
print(L)
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] 0 1
```

```
cat(L) # ошибка
```

В-третьих, `cat()` просто выводит объект на экран, не сохраняя информацию о нем, а `print()` сохраняет сам объект. Другими словами, функция `cat()` просто выводит информацию на экран, ничего не возращая, то есть возвращая пустой объект типа `NULL`, а функция `print()` выводит объект на экран и возвращает сам объект:

```
a <- cat(9)
```

```
## 9
```

```
a
```

```
## NULL
```

```
class(a)
```

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

```
b <- print(9)
```

```
## [1] 9
```

```
b
```

```
## [1] 9
```

```
class(b)
```

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

В функции `cat()` можно настраивать вид вывода, например, указывать разделитель между элементами в выдаче, выставив аргумент `sep`:

```
cat("Hello", ",", "world", sep = "") # пустота
```

```
## Hello,world
```

```
cat(24, 1, 2020, sep = "-") # дефис
```

```
## 24-1-2020
```

Как быть, если мы хотим вывести некоторый текст и значение переменной? Для этого можем использовать `cat()`, потому что он умеет «склеивать» записи:

```
name <- "Alla"
cat("Hello, ", name)
```

```
## Hello,  Alla
```

А если мы хотим, чтобы переменная была где-то внутри текста? Для этого есть функция `sprintf()`:

```
cat(sprintf("Hello, %s!", name)) # s - string
```

```
## Hello, Alla!
```

В то место, куда мы хотим добавить значение переменной, мы вписываем знак `%` и сокращённое название типа это переменной: `s` для строк (*string*), `i` для целочисленных (*integer*), `f` для чисел с плавающей точкой (*float*).

```
cat(sprintf("Age: %i", 25)) # i - integer
```

```
## Age: 25
```

```
cat(sprintf("Age: %f", 25.5)) # f - float
```

```
## Age: 25.500000
```

У чисел с плавающей точкой можно контролировать число знаков после запятой:

```
cat(sprintf("Age: %f", 25.5)) # по умолчанию 6 знаков
```

```
## Age: 25.500000
```

```
cat(sprintf("Age: %.3f", 25.5)) # 3 знака
```

```
## Age: 25.500
```

Ввод данных с клавиатуры

Как быть, если мы хотим запрашивать информацию у пользователя, а затем её использовать? Для этого есть функция `readline()`, аналог функции `input()` в Python. Для примера попросим пользователя ввести свое имя:

```
name <- readline(prompt = "Enter your name: ")
```

И оформим приветствие:

```
cat("Hello, ", name, "!", sep = "")
```

```
## Hello, Alla!
```

Также как и в Python, тут важно учитывать, что функция `readline()` всегда возвращает текст, то есть объект типа `character`, даже если пользователь ввёл число:

```
n <- readline(prompt = "Enter a number: ")
```

```
class(n)
```

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

Но исправлять такие вещи — конвертировать строки в числа — мы умеем.

```
n <- as.numeric(readline(prompt = "Enter your name: "))
```

```
## Enter your name:
```

```
class(n)
```

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

А что делать, если введено сразу несколько элементов, например несколько чисел через пробел?

```
inputs <- readline(prompt = "Enter something: ") # 2 3 4
```

Воспользуемся функцией `strsplit()` и разобьём получившуюся строку по пробелу:

```
res <- strsplit(inputs, split = " ")
```

```
res
```

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

Результат исполнения этой функции — это список (*list*). Про списки мы поговорим позже, пока к ним можно относиться как к вложенным спискам в Python. Для удобства работы превратим список в вектор:

```
res_v <- unlist(res) # раскрываем список и сохраняем в вектор
res_v
```

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

Сделаем все элементы вектора числами:

```
res_v <- as.numeric(res_v) # сделаем числовым
res_v
```

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

Получилось!