Основы программирования в R

Функции в R

Алла Тамбовцева, НИУ ВШЭ

Функции

На самом деле с функциями в R мы уже знакомы. Когда мы создаем вектор, мы берём функцию `c()`, когда мы определяем максимальное значение в векторе, вызываем функцию `max()`, когда выводим сообщение на экран, используем `print()` или `cat()`.

Но часто бывает, что встроенных функций R не хватает для полноценной работы, поэтому приходится дописывать некоторые функции самим.

При создании функции нужно учитывать три момента:

- что функция получает на вход (аргументы или параметры функции);
- что функция делает;
- что функция возвращает, выдает на выходе (результат функции).

Для примера напишем свою функцию, которая будет принимать на вход числовой вектор и считать его среднее значение. Для простоты будем считать, что пропущенных значений в векторе нет. Создадим функцию `my_mean()`.

Важно: своим функциям нужно давать названия, отличные от названий функций, уже встроенных в R. В случае совпадения R не выдаст ошибку, сохранит функцию с тем же названием, но потом могут возникнуть проблемы, да и самим легко запутаться.

Создание функции в R несильно отличается от создания функции в Python. Разница лишь в синтаксисе: здесь вместо оператора `def` мы используем функцию `function()`, которая сообщает R, объект какого типа мы создаем, плюс, в строке с результатом функции `return()` тоже является функцией, а не оператором.

```
my_mean <- function(x){
  avg <- sum(x) / length(x)
  return(avg)
}
```

Применим функцию к вектору `test1` и сохраняем результат в переменную `r1`:

```
test1 <- c(3, 6, 7)
r1 <- my_mean(test1)
r1
```

```
## [1] 5.333333
```

Все работает. На самом деле, можно было обойтись и без `return()`: R по умолчанию возвращает то, что указано в последней строчке в теле функции (в фигурных скобках).

```
my_mean <- function(x){
  avg <- sum(x) / length(x)
  avg
}
```

Тоже все работает:

```
my_mean(test1)
```

```
## [1] 5.333333
```

Главное не забыть важную вещь: в R, как и в Python (да и вообще в программировании) действие «функция возвращает какое-то значение» отличается от действия «функция выводит на экран какое-то значение». Функция может просто выводить результат на экран и никуда его не сохранять, а может, наоборот, сохранять результат, но на экран его не выводить.

Например, эта функция просто выводит среднее на экран, не сохраняя его:

```
my_mean2 <- function(x){
  avg <- sum(x) / length(x)
  cat(avg)
}
```

Результат выводится на экран, но не сохраняется:

```
my_mean2(test1)
```

```
## 5.333333
```

```
r2 <- my_mean2(test1)
```

```
## 5.333333
```

```
r2
```

```
## NULL
```

Здесь нас может спасти функция `print()`. Как мы обсуждали, эта функция не просто выводит объект на экран, но и сохраняет его. Проверим:

```
my_mean3 <- function(x){
  avg <- sum(x) / length(x)
  print(avg)
}
```

Результат выводится на экран:

```
my_mean3(test1)
```

```
## [1] 5.333333
```

И сохраняется!

```
r3 <- my_mean3(test1)
```

```
## [1] 5.333333
```

```
r3
```

```
## [1] 5.333333
```

Можно привести такую аналогию. Если я что-то говорю и не записываю, что говорю – я просто вывожу нечто на экран с помощью `cat()`: результат все видят, но обратиться к нему позже не получится. Если я по ходу рассказа делаю некоторые заметки – я возвращаю опредёленный результат, сохраняю записи, которые позже можно будет прочитать.

Однако никто не мешает сочетать и основной результат, и какие-то побочные действия. Напишем «внимательную» функцию для среднего, которая будет считать среднее и напоминать нам, что нужно спать.

```r
my_considerate_mean <- function(x){
  avg <- sum(x) / length(x)
  cat("You have to go to sleep!")
  return(avg)
}
```

Получаем и основной результат, и сообщение на экране:

```r
r4 <- my_considerate_mean(test1)
```

```
## You have to go to sleep!
```

```r
r4
```

```
## [1] 5.333333
```

Напоследок обсудим ещё одно отличие функций в R от функций в Python. Так как R не умеет автоматически объединять перечисленные через запятую объекты в какие-то структуры (Python запись 2, 3, 4 превратит в кортеж (2, 3, 4)), функции в R не умеют возвращать несколько объектов в `return()`. Их необходимо самим объединять либо в вектор, либо в список, либо в какую-то ещё структуру.

Проверим: напишем функцию `my_power()`, которая принимает на вход два целых числа `a` и `b` и возвращает пару чисел a^b и b^a.

```r
my_power <- function(a, b){
  return(a ** b, b * a)
}
```

Функция создана, но она не работает (ошибка):

```r
my_power(2, 5)
```

Объединим пару результатов в вектор:

```r
my_power <- function(a, b){
  return(c(a ** b, b * a))
}

my_power(2, 5)
```

```
## [1] 32 10
```

Ура! Всё работает!

Если мы хотим сделать выдачу более *user friendly*, в `return()` можем записать поименованный вектор:

```r
my_power <- function(a, b){
  return(c(result1 = a ** b, result2 = b * a))
}

my_power(2, 5)
```

```
## result1 result2
##      32      10
```