Урок 5. Числовые векторы и операции с ними

В этом и в следующем уроках мы поговорим о ключевой структуре данных в R – о векторе. Вектор – это просто некоторый список значений одного типа, аналог массивов в Python. Например, такой:

```
v <- c(2, 5, 8)
```

Вектор создается с помощью оператора `c()`, а элементы вектора перечисляются в скобках через запятую.

Почему векторы в R важны и почему мы не можем начать с чего-то более интересного, например, с датафреймов (таблиц с данными, в R они сохраняются в объект data.frame)? Дело в том, что вектор является основной единицей хранения данных и даже датафрейм воспринимается R как набор векторов: каждый столбец — отдельный вектор. То же происходит и со статистическими выдачами: их внутреннее представление часто выглядит как набор векторов, например, вектор с коэффициентами модели, вектор с техническими характеристиками модели и вектор с результатами.

Начнем наше знакомство с векторами с числовых векторов. Создадим вектор со значениями объемов продаж разных фирм (в тысячах) и посмотрим на него:

```
sales <- c(30, 80, 24, 67, 90, 32, 24)
sales
```

```
## [1] 30 80 24 67 90 32 24
```

Посчитаем число элементов — длину вектора:

```
length(sales)
```

```
## [1] 7
```

Да, в Python в списках было бы просто `len()`, главное не перепутать.

Теперь выберем первый элемент вектора по его индексу:

```
sales[1] # индекс в квадратных скобках
```

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

Обратите внимание, в отличие от Python и других языков программирования нумерация в R начинается с единицы, а не с нуля. Если мы вызовем нулевой элемент, R не выдаст ошибку, а покажет пустой элемент:

```
sales[0]
```

```
## numeric(0)
```

Здесь указан `numeric()`, поскольку вектор состоит из чисел.

В R, в отличие от Python, нет удобной возможности отсчитывать элементы с конца, поэтому последний элемент произвольного вектора придется выбирать так:

```
sales[length(sales)] # через длину вектора
```

```
## [1] 24
```

Изменить элемент вектора просто: нужно выбрать его по индексу и присвоить новое значение:

```
sales[1] <- 44
sales
```

```
## [1] 44 80 24 67 90 32 24
```

Для выбора нескольких элементов используются срезы:

```
sales[1:3] # выберем элементы с первого по третий
```

```
## [1] 44 80 24
```

В отличие от Python, оба конца среза включаются.

Полуоткрытые и открытые срезы, к сожалению, не работают:

```
sales[2:] # ошибка
```

```
## Ошибка: неожиданный ']' in "sales[2:]"
```

В квадратных скобках мы можем указывать не только индексы элементов, но и условия, в соответствии с которыми мы хотим отфильтровать элементы. Например, выберем все элементы, равные 24. Для этого укажем название вектора в скобках и пропишем условие равенства (как вы помните из Python, для проверки равенства используется двойной знак =):

```
sales[sales == 24]
```

```
## [1] 24 24
```

Или все элементы не более 30:

```
sales[sales <= 30]
```

```
## [1] 24 24
```

Конечно, условия в скобках можно сочетать, делать их более сложными, но давайте обсудим это позже, в отдельном уроке.

Если нам нужно узнать индекс конкретного элемента, пригодится функция `which()` и условие равенства. Узнаем, на каких местах в векторе `sales` стоит число 24:

```
which(sales == 24)
```

```
## [1] 3 7
```

Таким же образом можно узнавать индексы элементов, которые удовлетворяют определенным условиям. Например, узнаем, на каких местах стоят элементы больше 30:

```
which(sales > 30)
```

```
## [1] 1 2 4 5 6
```

В конце этого урока хотелось бы показать, что вектора не только важны для понимания структур данных в R, но и очень удобны на практике. Если мы хотим применить некоторую операцию к каждому элемента вектора, нам не понадобится никаких вспомогательных конструкций, мы сможем применить ее ко всему вектору сразу (*векторизованные операции*). Например, возведем каждый элемент вектора `sales` в квадрат:

```
sales ** 2
```

```
## [1] 1936 6400  576 4489 8100 1024  576
```

Или сложим два вектора одинаковой длины:

```
v1 <- c(0, 1, 4)
v2 <- c(8, 9, 2)
v1 + v2  # элементы суммируются попарно
```

```
## [1]  8 10  6
```

Или вычтем из первого вектора второй:

```
v1 - v2
```

```
## [1] -8 -8  2
```

Или применим какую-нибудь функцию, например, округлим все значения в векторе:

```
round(c(3.4, 5.7, 0.1))
```

```
## [1] 3 6 0
```

Что еще можно делать с числовыми векторами? Находить сумму всех элементов или их произведение:

```
v2
```

```
## [1] 8 9 2
```

```
sum(v2) # сумма
```

```
## [1] 19
```

```
prod(v2) # произведение
```

```
## [1] 144
```

Сортировать, как в порядке возрастания, так и в порядке убывания:

```
sort(v2)
```

```
## [1] 2 8 9
```

```
sort(v2, decreasing = TRUE) # убывание, опция decreasing
```

```
## [1] 9 8 2
```

Скорее всего, многие действия выше напомнили вам операции с массивами NumPy в Python. Это неслучайно. Действительно, о векторах можно думать как об одномерных массивах. Это наборы элементов одного типа (мы работали с векторами, полностью состоящими из чисел), над которыми можно выполнять векторизованные операции. В следующем уроке мы подробно поговорим о разных типах данных в R, а пока предлагаю поработать с векторами и выполнить практическое задание. Желаю удачи!