Основы программирования в R

Векторы в R: часть 1

Алла Тамбовцева, НИУ ВШЭ

Содержание

Числовые векторы

Вектор в R – список некоторых объектов. Для тех, кто знаком с Python: вектор можно сравнить с массивом (*array*) или списком (*list*), состоящим из элементов одного типа. Технически, элементы вектора могут иметь разный тип, но тогда вектор будет уже не тот — увидим это позже.

Создадим вектор `v`, состоящий из целых чисел. Для объединения элементов в вектор используется базовая функция `c()`:

```
v <- c(1, 0, 0, 2)
v
```

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

Какие характеристики вектора мы можем получить? Из самых простых — тип его элементов и длина. Определим тип:

```
class(v)
```

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

А теперь длину — число элементов вектора:

```
length(v)
```

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

Перейдём к более интересным характеристикам числового вектора. Посчитаем сумму его элементов:

```
sum(v)
```

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

Произведение элементов (`prod()` от *product*):

```
prod(v)
```

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

Среднее арифметическое:

```
mean(v)
```

```
## [1] 0.75
```

И медиану:

```
median(v)
```

```
## [1] 0.5
```

Кроме того, если рассмотреть числовой вектор как выборку, мы сможем продолжить список описательных статистик: выборочные квантили, дисперсия, стандартное отклонение, минимум, максимум... Посмотрим, как всё это вычислять в R.

Для вычисления квантилей в R есть функция с очень логичным названием `quantile()`. На вход эта функция принимает числовой вектор и уровень квантиля. Создадим вектор побольше и найдём для него квантиль уровня 0.2:

```
w <- c(9, 4, 5, 6, 1, 0, -1, 2, 3)
quantile(w, 0.2)
```

```
## 20%
## 0.6
```

Итак, 20% значений `w` не превышают значения 0.6.

В качестве аргумента, который соответствует уровню квантиля, можно подавать не только отдельные числа, но и векторы. Например, запросим нижний квартиль, медиану и верхний квартиль:

```
quantile(w, c(0.25, 0.5, 0.75))
```

```
## 25% 50% 75%
##   1   3   5
```

Теперь вычислим показатели разброса значений — дисперсию и стандартное отклонение:

```
# variance
var(w)
```

```
## [1] 9.944444
```

```
# standard deviation
sd(w)
```

```
## [1] 3.153481
```

И вернёмся к показателям попроще — найдём минимум и максимум:

```
min(w)
```

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

```
max(w)
```

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

Если продолжать разговор об описании выборки, то можно вспомнить о ранжировании наблюдений. R умеет вычислять ранги, для этого существует функция `rank()`:

```
rank(w)
```

```
## [1] 9 6 7 8 3 2 1 4 5
```

Если в векторе есть повторяющиеся значения, для них R считает усреднённые ранги. Посмотрим на это на примере вектора `v`:

```r
rank(v)
```

```
## [1] 3.0 1.5 1.5 4.0
```

Где ранжирование элементов — там и сортировка. Для сортировки в R используется функция `sort()`.

```r
sort(w)
```

```
## [1] -1  0  1  2  3  4  5  6  9
```

Если нас интересует сортировка по убыванию, пригодится аргумент `decreasing`:

```r
sort(w, decreasing = TRUE)
```

```
## [1]  9  6  5  4  3  2  1  0 -1
```

Теперь давайте рассмотрим операции, которые можно выполнять сразу с несколькими числовыми векторами. Создадим два вектора:

```r
x <- c(1, 0, 0, 2)
y <- c(0, 1, 1, 2)
```

С числовыми векторами в R мы можем осуществлять те же операции, что и с векторами в математике. Но результаты не всегда совпадают. Например, обычное умножение векторов с помощью знака `*` не соответствует ни скалярному, ни векторному произведению векторов. Это происходит потому, что арифметические операции с векторами выполняются поэлементно.

Сложим два вектора:

```r
x + y
```

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

А теперь перемножим:

```r
x * y
```

```
## [1] 0 0 0 4
```

Если вас интересует скалярное произведение векторов, как в математике, то оператор для умножения уже будет другим (полезный факт, тот же оператор можно использовать для умножения матриц):

```r
# dot product
x %*% y
```

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

Для объединения векторов, любых, не только числовых, используется уже знакомая нам функция `c()` (аналог методов `.append()` и `.extend()` в Python):

```r
result <- c(x, y)
result
```

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

Текстовые векторы

Векторы могут быть текстовыми, то есть содержать элементы типа `character`:

```r
name <- c("Ann", "Mike", "Ben")
name
```

```
## [1] "Ann"  "Mike" "Ben"
```

О том, как работать с текстом в R, мы ещё будем говорить отдельно, а пока посмотрим на что-то совсем простое — перевод всех букв в строчные или заглавные:

```
tolower(name)
```

```
## [1] "ann"  "mike" "ben"
```

```
toupper(name)
```

```
## [1] "ANN"  "MIKE" "BEN"
```

Типы элементов и приведение типов

Так как вектор умеет хранить данные только одного типа, в случае, если в векторе встречаются два типа, один из них неизбежно «вытеснит» другой. Так, текстовый тип (`character`) «вытеснит» числовой (`numeric`):

```
mixed <- c(2, "one", 3)
class(mixed)
```

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

При работе с данными нужно всегда внимательно относиться к типу объектов. Например, если у нас есть результаты опроса студентов, и мы видим, что пять респондентов не указали свой возраст, не стоит кодировать эти пропущенные значения словом «нет». Лучше оставить значения пропущенными или закодировать их заведомо невозможным значением (например, 1000), чтобы потом ответы этих респондентов спокойно отфильтровать. Социологи так обычно и поступают: ответы на вопросы кодируются небольшими числами (например, от 1 до 6, от 1 до 10), а пропущенные значения кодируются числами 98 или 99.

Как и у обычных переменных, у векторов можно изменять тип, то есть тип всех элементов вектора. Например, можем преобразовать текстовый вектор в числовой с помощью функции `as.numeric()`:

```
text <- c("2", "3", "5")
as.numeric(text)
```

```
## [1] 2 3 5
```

Или в целочисленный:

```
as.integer(text)
```

```
## [1] 2 3 5
```

При этом, если среди элементов есть дробное число, записанное, как текст, то все элементы вектора преобразуются в дробные числа:

```
as.numeric(c("2.3", "5", "6"))
```

```
## [1] 2.3 5.0 6.0
```

Грустная новость: если бы в `"2.3"` разделителем являлась запятая, ничего бы не получилось – R в качестве разделителя разрядов признает только точку:

```
as.numeric(c("2,3", "5", "6"))
```

```
## Warning:                     NA
```

```
## [1] NA  5  6
```

Что делать? В таком случае нужно сначала заменить запятую на точку с помощью функции `gsub()`, а уже потом преобразовывать:

```
as.numeric(gsub(",", ".", c("2,3", "5", "6")))
```

```
## [1] 2.3 5.0 6.0
```

Работа с элементами вектора

Для начала создадим текстовый вектор:

```
people <- c("Mary", "Peter", "John")
people
```

```
## [1] "Mary"  "Peter" "John"
```

Для того, чтобы выбрать элементы вектора по их индексу (положению в векторе), нужно учитывать, что в R нумерация начинается с 1, а не с 0, как в Python и многих языках программирования.

```
people[1]
```

```
## [1] "Mary"
```

Если мы выберем элемент с индексом 0, R вернёт нам пустой элемент:

```
people[0]
```

```
## character(0)
```

Для выбора нескольких элементов, стоящих подряд, можно использовать срезы:

```
people[1:2]
```

```
## [1] "Mary"  "Peter"
```

Если нужно выбрать элементы, которые следуют в векторе не подряд, индексы интересующих нас элементов нужно оформить в виде вектора:

```
people[c(1, 3)]
```

```
## [1] "Mary" "John"
```

При желании выбор элементов подряд и не подряд можно объединить:

```
people[c(1:2, 2:3)]
```

```
## [1] "Mary"  "Peter" "Peter" "John"
```

Обратите внимание: в отличие от Python, в R правый конец среза включается. Например, код `names[1:2]` вернёт первые два элемента, а не только элемент с индексом 1.

А теперь мы будем отбирать элементы вектора по их значению. Для этого необходимо указывать интересующие критерии выбора (условия) в квадратных скобках. Создадим новый вектор `v`:

```
v <- c(1, 8, 9, 2, 3, 0, -1)
```

Выберем элементы `v`, которые больше 3:

```
v[v > 3]
```

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

Как работает этот код? Давайте отдельно запустим строку с выражением, которое стоит в квадратных скобках:

```
v > 3
```

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

R вернул нам вектор из `TRUE` и `FALSE`, где `TRUE` стоит на месте элементов, для которых условие выполняется. Соответственно, при запуске кода `v[v > 3]` R отбирает только те элементы, на месте которых в `v > 3` стоят `TRUE`.

Усложним задачу. Будем выбирать только чётные элементы вектора `v`. Для этого нам понадобится оператор для определения остатка от деления: `%%`. Чётные элементы – те, которые делятся на 2 без остатка. Значит, остаток от деления их на 2 должен быть равен нулю:

```
v[v%%2 == 0]
```

```
## [1] 8 2 0
```

Условия можно сочетать:

```
v[v > 3 & v %% 2 == 0]
```

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

```
v[v > 3 | v %% 2 == 0]
```

```
## [1] 8 9 2 0
```

Иногда нам нужно не найти элемент вектора по его номеру или по определённым критериям, а выполнить обратную задачу: вернуть индекс элемента (его порядковый номер в векторе). Это можно сделать с помощью функции `which()`:

```
which(v > 8)
```

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

```
which(v > 3)
```

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

```
which(v > 10)
```

```
## integer(0)
```

Обратите внимание: в случае, если сразу несколько элементов удовлетворяет некоторому условию, функция `which()` возвращает индексы всех, а не только первого подходящего элемента.