Vectors in R

Alla Tambovtseva

Vectors

A vector in R is a list of elements. It is created in the following way:

```
x <- c(1, 0, 0, 1, 2) # vector x
```

We can look at this vector:

```
x
```

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

And define its type (the type of its elements):

```
class(x)
```

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

Also we can define a length of a vector, i.e. a number of its elements:

```
length(x)
```

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

With vectors in R we can do the same operations as with vectors in mathematics. But results might not coincide. For example, multiplication via * does not correspond neither to dot product, nor to cross product. This is because operations in R are applied element-wise:

```
x + x # element by element addition
```

```
## [1] 2 0 0 2 4
```

```
x * x # element by element multiplication
```

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

We can also calculate a sum of elements:

```
sum(x)
```

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

Not only numeric vectors can be created, character ones are possible:

```
names <- c('Ann', 'Tom', 'Jane')
names
```

```
## [1] "Ann"  "Tom"  "Jane"
```

Besides, it possible to include elements of different types in a vector:

```
mix <- c("Anna", 0, 1, "Jane")
mix
```

```
## [1] "Anna" "0"    "1"    "Jane"
```

However, in this case one type ‘conquers’ another, and all vector elements become character:

```
class(mix)
```

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

While working with data we should pay attention to the types of objects. For example, if we have results of a survey and see that five respondents did not indicate their age, it is not wise to code these missing values with the word "no". It is better to keep them missing or code them with impossible values (1000 or 500) so as to freely filter these respondents.

Factor vectors

Sometimes numbers in vectors do not correspond to quantitative indicators. Using numbers we can encode qualitative information. For instance, number 1 will stand for male respondents and number 2 - to female respondents.

```
gender <- c(1, 1, 2, 1, 2, 2, 1)
```

In such cases we often want R to regard these values not as numbers, but as nominal information, as a vector with words "male" and "female". To do this, we can use the special R type called factor:

```
g <- factor(gender)
g
```

```
## [1] 1 1 2 1 2 2 1
## Levels: 1 2
```

```
class(g)
```

```
## [1] "factor"
```

Factor vectors have levels, unique values that are used for encoding qualitative variables. There are two levels "male" and "female" coded as 1 and 2.

Working with vector elements

So as to choose an element of a vector by its index (position in a vector), we should remember that numeration in R starts from 1, not from 0 like in other programming languages.

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

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

```
names[1] # 1st element of the vector names
```

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

```
names[0] # no such elements
```

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

We can choose multiple elements at the same time using slices:

```
names[1:2] # 1st and 2nd elements
```

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

If elements do not follow each other, we can list their numbers as a vector:

```
names[c(1, 3)] # 1st and 3rd elements
```

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

It is possible to combine slices as well:

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

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

Now let's choose elements by their values. To do this we should indicate criteria in square brackets. Let's take the vector v:

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

```
## [1]  1  8  9  2  3  0 -1
```

Choose elements of v that are greater than 3:

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

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

Consider a more diffucult problem. Now we will choose only even elements of v. So we will need the operator %% to return a remainder of division. Even elements are those that are divided by 2 with no remainder. It means that the remainder should be equal 0:

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

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

We can combine conditions as well:

```
v[v > 3 & v%%2 == 0] # even elements greater than 3
```

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

Sometimes we face the opposite problem: we have to return indices of elements that suit some requirements instead of choosing elements themselves.

```
names
```

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

Let's ask R which elements of names are equal "Jane":

```
which(names =='Jane') # no such elements
```

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

```
which(names =='Peter') # 3rd element
```

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

Or which elements of v exceed 3:

```
which(v > 3) # indices, not elements themselves
```

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

Indices of even elements:

```
which(v%%2 == 0)
```

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

So as to add an element to a vector, it is enough to know the position where this element should be placed:

```
p <- c(1, 2)
p[3] = 7 # add 3rd element
p
```

```
## [1] 1 2 7
```

To remove an element by its value, we can simply filter elements that are not equal to this value:

```
v[v != 8] # everything except 8
```

```
## [1]  1  9  2  3  0 -1
```

Such an approach works if there are repeated values:

```
w <- c(6, 6, 6, 7)
w[w != 6]
```

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