DASS. Loops in R

Alla Tambovtseva

For-loop in R

Let's start from a short recall. How to create a sequence of subsequent integers in R? It is simple:

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Now imagine that we want to print all the integers from 1 to 10, each number on a new line. How to do it? Of course, we can repeat code lines like this:

```
print(1)
```

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

```
print(2)
```

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

But it is inconvenient. No need to be a programmer to see that it is unwise. So, we can use a special construction called *a loop* that helps us to avoid repeating code. We will use a more common loop, *a for-loop*. First, we have to specify a vector (any set, any sequence) that R will "walk along". In our case it is `1:10`, a sequence. Then we should decide what R should do with the elements of this sequence (it is a body of a loop).

```
# go from i=1 to i=10
# and print this i
for (i in 1:10){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

A body of a loop is usually written in curly braces, it is a set of operations that R should do at every step of a loop. In the example above we have only one command, the function `print()`. The variable `i` here is "a frame" (formally, an iterator) that runs inside a vector `1:10`. It changes its value from step to step, so at the beginning it equals 1, then 2, etc, and finally 10. No matter how this variable is called, we can use any letter or any word instead of `i`, R will always understand that we want him to do. For example, we can write a word `number` instead:

```
for (number in 1:10){
  print(number)
}
```

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

```
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

The results are the same. Now we can add one more operation, one more line of code to the body of our loop:

```
for (number in 1:10){
  square <- number^2 # get a number squared
  print(square)  # print it
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
## [1] 36
## [1] 49
## [1] 64
## [1] 81
## [1] 100
```

In the end let's consider one more example. Suppose we have a data set with information on respondents and we want to create a column with a text id that looks like this: `respondent_1`. There are 100 respondents. To obtain such ids, we can take a word `respondent` and then append numbers from a sequence `1:n` where `n` is the number of respondents. We already discussed such a problem at our first lectures, but we did not use loops. Recall:

```
r <- "respondent"
paste(r, 1, sep = "_")  # join using _
```

```
## [1] "respondent_1"
```

Now we have to create an empty vector of length 100, a vector of 100 NA's:

```
resp <- rep(NA, 100)
```

and replace all NA's with an id. We can generate ids using a for-loop and then assign these values to the corresponding elements in `resp`:

```
for (i in 1:100){
  r <- paste("responsent", i, sep = "_")  # generate
  resp[i] <- r  # put r at the i-th place
}

head(resp)
```

```
## [1] "responsent_1" "responsent_2" "responsent_3" "responsent_4"
## [5] "responsent_5" "responsent_6"
```

Now it is done! This technique of replacing values will be useful in the next lecture on statistical laws.