Loops in R

Sean Hellingman ©

Data Visualization and Manipulation through Scripting (ADSC1010)

shellingman@tru.ca

Fall 2024

THOMPSON RIVERS UNIVERSITY

Topics

Introduction

- Loops are a staple of all programming languages.
- R is very good at performing repetitive tasks.
- Can execute loops for a specified number of times or until a specified condition is met.
- Three main types in R:
  1. *for* loop
  2. *while* loop
  3. *repeat* loop

`for` Loops

- Used to repeat a task for a defined number of times.
- The basic structure of a for loop in R is:

```
for (index in range) { Block of Code }
```

- `range` is a vector of values for `index`.
  - Example: `1:10` or `seq(1,10,by=2)` or `c(1,5,9,8)`.

Example 1

- Use a *for* loop to print sequential numbers from 1 to 5.

`for` Loop Advice

- Before writing a loop, write the first few lines of code to visualise the patterns.
- Change the index values manually within the loop and check the results.
- `for` loops are computationally demanding!

`while` Loops

- The basic structure of a `while` loop in R is:

```
while (expression) { Block of Code }
```

- `expression` is a relational statement.
  - Example: `Balance > 0` or `length(p) < 1000`
- Code is executed until expression becomes false.
- Note that there is no index variable being incremented.
  - Often need to do this ourselves.

Example 2

- Use a *while* loop to print sequential numbers until we reach 10.

`repeat` Loops

- The basic structure of a `repeat` loop in R is:

```
repeat { Block of Code }
```

- A `break` command needs to be written inside a `repeat` loop.
  - Example: `if (x == 10) {break}`
- Code is executed until the exit condition is satisfied.
- *We will not be focusing on `repeat` loops.*

Example 3

- Use a *repeat* loop to print sequential numbers until we reach 10.

Nested Loops

- A **nested loop** is simply a loop within the body (block of code) of another loop.
- The *outer* loop may contain more than one nested loop.
- The nested loop runs completely within each iteration of the outer loop.
- Both the nested and outer loops may contain break statements.

Example 4

- Create a function that uses nested loops to print the possible combinations of elements of two numeric vectors.
  - Include a `break` in the nested loop if the two elements' sum is greater than 50.

Comments

- Loops can be very computationally demanding & easy to make mistakes.
- In general, data scientists should avoid using loops whenever possible.
- Loops do not need to be iterated over numeric values.
- There are some situations where using loops is *necessary*.
- The base R functions: `apply()`, `lapply()`, `tapply()`, `sapply()`, `vapply()`, and `mapply()` may be more useful than looping.
- Alternatives are also found in the *dplyr* and *tidyr* packages.

`lapply()` Function

- The `apply()`/`lapply()` functions are often a good alternative to loops.
- Goes through each element of an object to perform a task (function).
- Structure: `lapply(Object, Function)`
- The output of `lapply()` is represented as a list.

Example 5

- Apply the `lapply()` function to print sequential numbers until we reach 10.

Exercise 1

- Consider the vector $x = (1, 2, 3, ..., 50)$.
- Use a `for` loop to store x in the variable `x` in three different ways:
  - `for (i in 1:50) {line of code}`
  - `for (i in 0:49) {line of code}`
  - `for (i in seq(from=3,to=101,by=2)) {line of code}`
- In each case you should initialize `x <- rep(NA,50)` outside the loop.
  - If we don't initialize `x <- c()`, i.e. an empty vector, outside the loop, how should we update the line of code inside the loop?

Exercise 2

- Use a `for` loop to store each of the following vectors in an eponymous variable (e.g. store w in a variable named `w`).
  - The 1×50 vector $w = (1, 4, 9, ..., 2500)$.
  - The 1×30 vector $x = (-1, 1, -1, 1, ..., -1, 1)$.
  - The 1×70 vector $y = (1, -1, 1, -1, ..., 1, -1)$.
- Ensure that you can construct each vector in two different ways:
  - Beginning your index variable at `0`.
  - Beginning your index variable at `1`.
- Good practice to initialize outside of the loop (e.g. `w <- rep(NA,50)`).

Exercise 3

- Use a `while` loop to determine how many deposits of $500 it will take you to reach $10000.
- Keep in mind, there are (at least) two ways to do this:
  - Create a vector that gets longer with each deposit, stop as soon as last component exceeds the threshold.
  - Keep track of current balance and number of deposits that have been made, stop as soon as current balance exceeds the threshold.

Exercise 4

- Apply the `lapply()` function to print sequential **even** numbers until we reach 50.

Exercise 5

- Explore the base R functions: `apply()`, `lapply()`, `tapply()`, `sapply()`, `vapply()`, and `mapply()` and their possible applications.

References & Resources

1. Douglas, A., Roos, D., Mancini, F., Couto, A., & Lusseau, D. (2023). *An introduction to R*. Retrieved from https://intro2r.com/

- for loops
- apply()