6. Control structures and functions in R

Principles of Data Science with R

Dr. Uma Ravat

PSTAT 10

Announcement

Next we will see...

We've seen

- Data Types
- Data structures
- Plotting

Now:

- Control structures
- (User defined) functions

Motivation

Conditionals, Iterators and functions

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$$

To code the function or plot it's graph succinctly we need

1. **Conditionals :** "If x>0 do this... otherwise do that..."
2. **Iterators/loops :** "Repeat this action several times"
3. **user-defined functions**

1 and 2 are called control structures, since they control how code is executed by the computer.

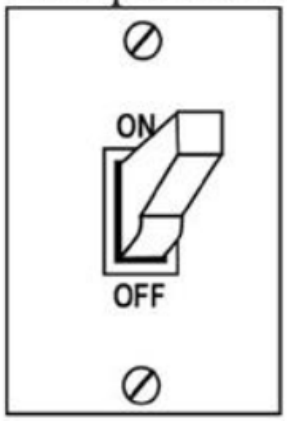
User defined functions may need control structures along with other lines of sequential code.

Conditionals in everyday language:

If you commit a crime

Then you go to jail

If I flip on the switch



Then the lights go on

`if` Syntax:

```
if (test expression){
  statement
}
```

- If the test expression is `TRUE`, the statement is executed.

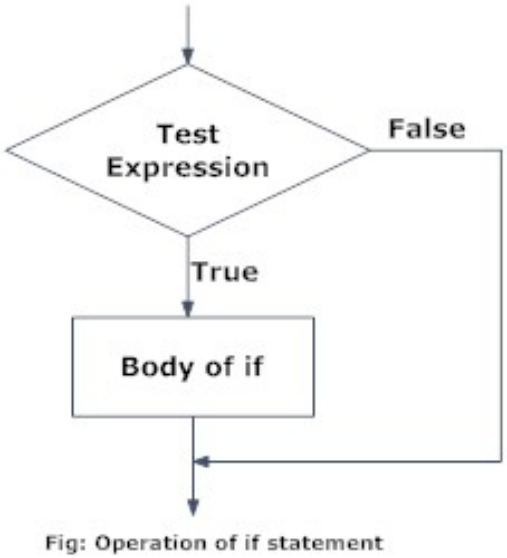


Fig: Operation of if statement

`if-else` Syntax:

- An `if` statement can be followed by an optional `else` statement

```
if (test expression1) {
  statement 1
} else {
  statement 2
}
```

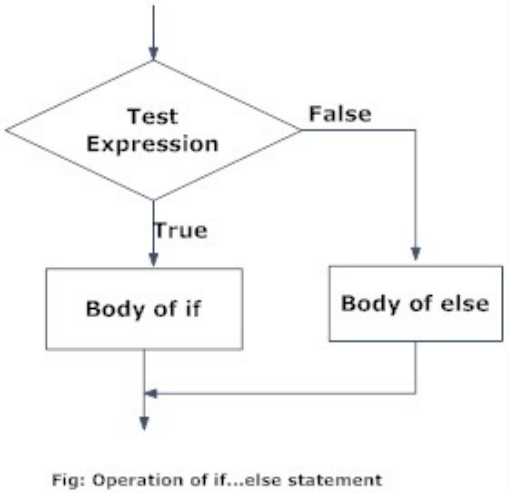
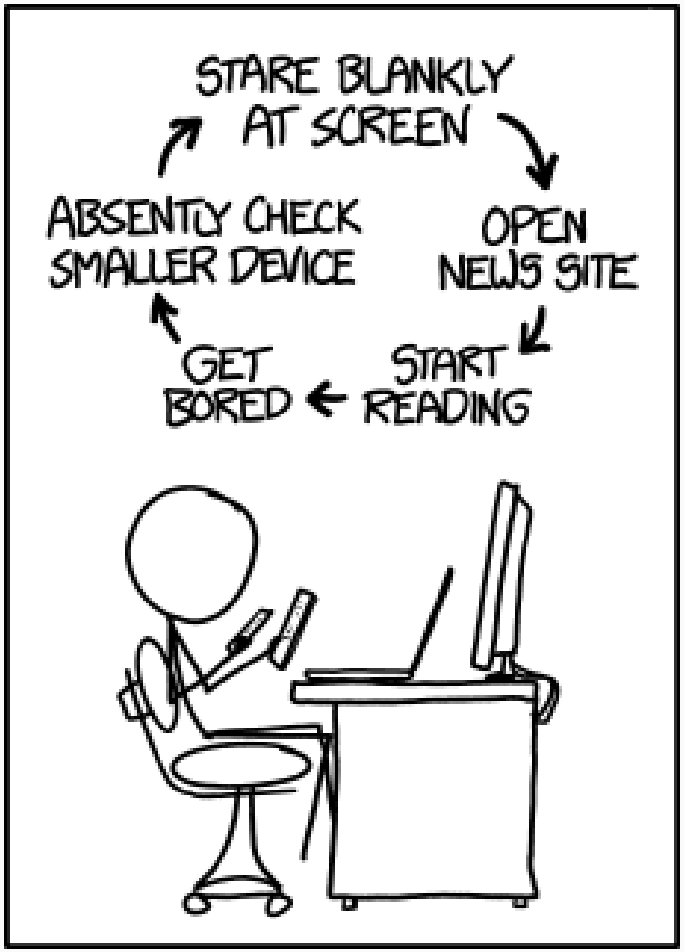
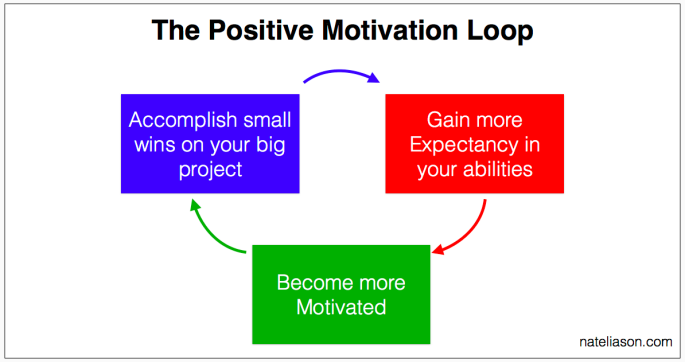


Fig: Operation of if...else statement

–>

Loops or Iterators





- Loops are used when we need to execute a block of code several times.
- Statements are executed sequentially. The first statement in a function is executed first, followed by the second, and so on.
- A loop statement allows us to execute a statement multiple times.

Loops in `R`

- `for` loop : Executes a statement or sequence of statements multiple times. Tests the condition at the **end** of the loop.
- while loop Repeats a statement or sequence of statements while a given condition is true. Tests the condition **before** executing the loop.
- repeat loop Executes a statement or sequence of statements multiple times until a stop condition is met.

For Loop SYNTAX

```
for (counter in counter-vector)
  {
    statements #body of for loop
  }
```

Fig: operation of for loop

Construct a table of logarithms

Use a `for` loop to construct a table of logarithms from 1 to 10.

```
table.of.logarithms <- vector(length=10, mode="numeric") #empty/null vector
table.of.logarithms
```

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

```
for (i in 1:length(table.of.logarithms)) {
  table.of.logarithms[i] <- log(i)
}

names(table.of.logarithms) <- 1:length(table.of.logarithms)
table.of.logarithms
```

```
##         1         2         3         4         5         6         7         8
## 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595 1.9459101 2.0794415
##         9        10
## 2.1972246 2.3025851
```

Tracing through the for loop is useful

counter: `i`	`table.of.logarithms[i]`
1	log(1)
2	log(2)
3	log(3)
...	...
10	log(10)

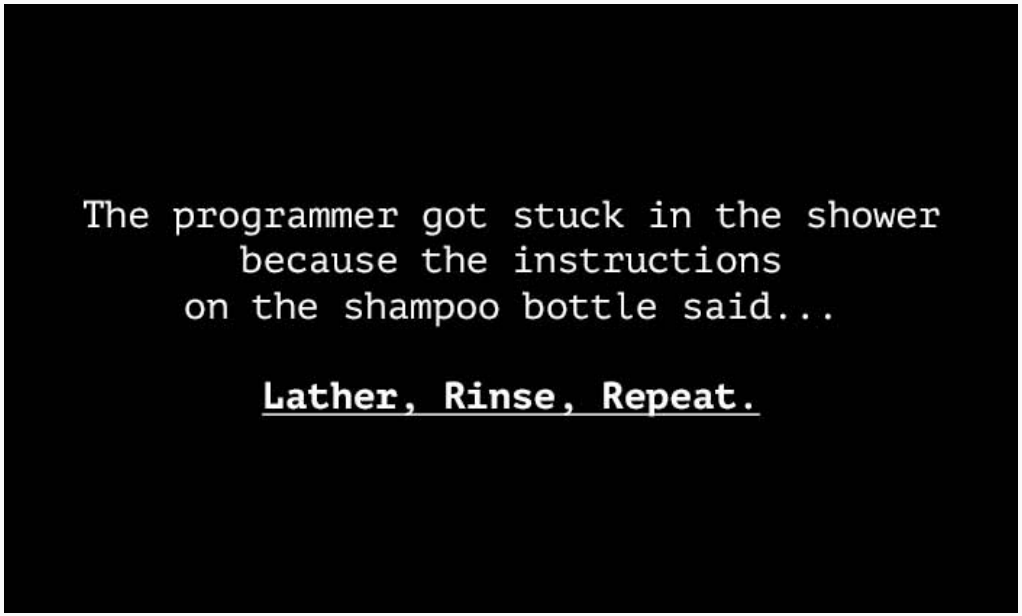
"**iterates over** the counter-vector"

Note, there is a better way to do this job!

The body can contain just about anything, including:

- `if()` clauses
- other `for()` loops (nested iteration)
- `for` loops are not limited to numeric vectors in the counter vector. We can pass character vectors, logical vectors or expressions

Programming humor



Function in R

Functions in R are either

- built-in (free for you to use!)
- user-defined (you need to code them up)

Functions are (most often) verbs, followed by what they will be applied to in parentheses

function and arguments

```
do_this(to_this)
```

Here `do_this` is the function and `to_this` is the **argument** to the function

```
do_that(to_this, to_that, with_those)
```

Here `do_that` is the function and `to_this, to_that, with_those` are the three **arguments** to the `do_that` function

What should be a function?

- Things you're going to re-run, especially if it will be re-run with changes
- Chunks of code which are small parts of bigger analyses
- Chunks which are very similar to other chunks

SYNTAX:

```
function_name <- function(arg1, arg2, ...)
  {
    code that does something
    return(object)
  }
```

1. **function Name:** choose a name for your function
2. **arguments** `arg1`, `arg2`, ...
   - An argument is a placeholder
   - When a function is called or invoked, you pass a value to an argument.
3. **function body:** Write code that does something
4. **return** or **print**: The result

questions you should be able to answer

Your job is to learn the syntax and develop an acumen to choose the best control structure/function for a given situation.

Summary:

1. Control structures
    - Conditionals: `if`, `if-else`, `ifelse`
    - Iterators: `for`, `while`, `repeat`
2. Functions

Maintain a glossary of functions used.

Congratulations!

This completes the first part of the course

An Introduction to R programming for Data science

Next we will look at

Introductory Probability and Statistics in R.