An introduction to R
WS 2018/2019

Organisation and Basics of Algorithmics

Dr. Sonja Grath
Dr. Eliza Argyridou

Special thanks to:
Prof. Dr. Martin Hutzenthaler and Dr. Benedikt Holtmann for significant contributions to course development, lecture notes and exercises and Dr. Noémie Becker for the preparation of this lecture

What should you know after day 1?

Part I: Course organisation

- Course instructors
- Lectures and exercises
- How to get credits for the course

Part II: Basic algorithmics

- Algowhat?
- Variables
- Read and write
- Tests and logic
- Loops
- Take home message

Part I
Course organisation

Where can you find information on the course?

- Webpage: http://evol.bio.lmu.de/_statgen/Rcourse/ws1819/
- Summary: Syllabus_R-course_2019.pdf
- All course material (presentations, scripts, exercise sheets) will be posted on the website.

Disclaimer:
The handouts we will put online may contain only a summary of the contents of the slides shown in the lecture.
More detailed explanations are given on the whiteboard and with practical demonstrations during the lectures. The questions in the exam refer to the whole content of the course including lectures and exercises.

When and where?

- Course: March 4 – March 15, 2019
- Lectures and correction of the exercises: every morning from 9 to 12 am in B 01.027
- Exercise sessions: every day from 1 to 5 pm in C00.005, G00.037 or D00.021 (laptop/Mac users). You can also work from home or elsewhere.

How do I get my 3 ECTS?

You have to pass the exam.

- Final exam: March 15, 2019 – 10-12 am in B00.019
- Make-up exam: April 12, 2019 – 10-12 am in tba
- For both exams, you are allowed to bring a two-sided A4 "formula" sheet in your own handwriting

If you want to write the make-up exam, please register via email (grath@bio.lmu.de) before April 1.

Attendance is not mandatory but we strongly recommend to attend both lectures and exercise sessions to learn efficiently.

The most efficient way to learn programming is to program.

Course outline

Week 1

March 4: Basics of algorithmics
March 5: Getting started with R
March 6: Data types and structure
March 7: Reading and writing data
March 8: Manipulating datasets

Week 2

March 11: Plots
March 12: Programming in R – Part I
March 13: Programming in R – Part II
March 14: Basic statistics with R
March 15: **EXAM**

Part II
Basic algorithmics

Acknowledgements
For this lecture I would like to thank Christophe Darmangeat from Université Paris-Diderot for his online material (in French). And Noémie Becker for providing this material.

Algowhat?

Who has already executed an algorithm?

- Follow a recipe from a book to cook.
- Decipher instructions to configurate a device.
- Assemble ikea furniture following the instructions.

Who has already written/developed an algorithm?

- Indicate the way to the English Garden to a lost tourist.
- Prepare a treasure hunt for the birthday party of your little sibling.
- Write instructions for your grandma / grandpa / parents on how to use the printer / email / dvd player.

What is an algorithm?

An algorithm is a list of instructions that, upon correct execution, lead to a wanted result.

- To be useful, the algorithm shall contain only instructions understandable by who has to execute it (think again of some of our examples above).
- In our case, it is easier as computers are rather stupid/logical and do not have cultural background etc...

Algorithms and Programming

The algorithm is independent of the specific programming language. It is the logic structure of the program.

- You should have a precise idea of instructions the program should contain before starting to type the code in your chosen language.
- The algorithm can be written in pseudo-code and later translated into your chosen language to be executed.

Some definitions

Algorithm:
Systematic logical approach which is a well-defined, step-by-step procedure that allows a computer to solve a problem.

Pseudo-code:
It is a simpler version of a programming code in plain English which uses short phrases to write code for a program before it is implemented in a specific programming language.

Program:
It is exact code written for problem following all the rules of the programming language.

https://www.geeksforgeeks.org/difference-between-algorithm-pseudocode-and-program/

Motivation

Problem:
Determine GC content of a DNA sequence consisting of the nucleotides A, C, G, T.

Input: ACCGGTACAT
Output: 0.5

➡ We will now define an algorithm, write some pseudo-code and write a small program to solve this problem

Algorithm vs. Pseudo-Code vs. Program

Problem: Determine GC content of a DNA sequence

Algorithm
1. Set counter to 0, take DNA sequence and start from the leftmost element, one by one check if element eqals C or G
2. If element equals C or G, increase counter by 1
3. Give counter divided by the length of the DNA sequence (the GC content)

Pseudo-code
```
Input: DNA sequence
counter = 0
For i from 1 to length(dna) then
    If dna[i] = "C" or dna[i] = "G"
        counter = counter + 1
    End of If
gc_content = counter/length(dna)
End of For
Output: gc_content
```

Programs in Python and R

python
```
def gcContent(dna):
  counter = 0
  for nuc in dna:
    if nuc == 'G' or nuc == 'C':
      counter = counter + 1
  return counter / len(dna)
```

```
gcContent <- function(dna, counter = 0){
    dna <- unlist(strsplit(dna, ""))
    for(i in 1:length(dna)){
        if(dna[i] == "C" | dna[i] == "G")
            {counter = counter + 1}
    }
    return(counter/length(dna))
}
```

We need variables to store information

Information can be

- given by the user *or*
- a result of the program

In practice:
- The computer assigns space in memory for information.
- The computer assigns a label to this space with a binary address.
- Computers can only process binary information as they are made of electronic components that can be uncharged or charged (symbolized by 0 and 1).

Types of variables

The space that will be allocated in memory depends on the type of the variable,

Examples in R:

- integer
- double
- complex
- character
- logical (binary information, boolean variables, TRUE/FALSE)

Variable declaration

Many programming languages require declaration of variables:
Usually at the beginning of the program, you need to specify the type of variable to allocate enough memory (example: integer vs decimal).

In R there is no need to declare the variables.

The memory is allocated the first time the variable is assigned a value. We also do not need to define *a priori* the type of variable.

Example:
```
age <- 25
student <- "Anna"
```
↑ value assignment (more tomorrow)

Value assignment

Can be written as:
```
Var1 <- 24
```

Can also affect the value of another variable:
```
Var2 <- Var1
Var2 <- Var1 + 4
```

The order of the instructions plays a role of course:

Pseudo-code:

```
Begin      Begin
A <- 2     A <- 25
A <- 25    A <- 2
End        End
```

R:

```
A <- 2     A <- 25
A <- 25    A <- 2
```

What is the value of A?

Note:
In many programming languages, a value must belong to a defined type – not so in R. More on this matter later in the course.

Operators

Definition:
An operator is a sign linking two values to produce a result.

Possible operators depend on the type of the variable, for example.

- numeric: +, -, *, /, ^
- text: & to concatenate (in pseudo-code, not in R)
- boolean:

Pseudo-code:	R:
and	&
or	|
not	!

Read and write

Necessary to communicate with the user.

Write: prompt or save something (result or question to the user)
Read: read value given by the user or from file

Example (in pseudo-code):

```
Write "Please enter your name"
Read Name
```


Control structures

We have already learned:
- How to ask information from the user.
- How to save information into variables.
- How to make operations on variables.
- How to print results.

Next:
The algorithm should be flexible and should allow for different options

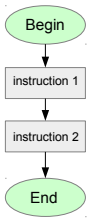
Example:
Black/White or Color Printing?

If your document has colors: tick colors
else: tick black and white

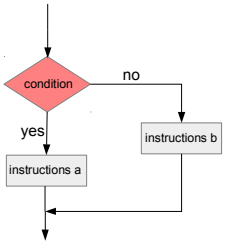
Overview on control structures

3 structures of an algorithm:

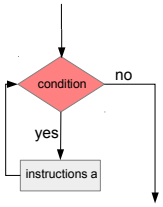
I. Sequential execution



II. Conditional execution



III. Repetition and looping



What is a condition?

A condition is an expression that resolves to a boolean variable (TRUE or FALSE).

A condition can be given as
- direct variable of type `boolean`
- result of a function that gives back a value of type boolean
- relational operator or logical operator

Relational operators:

Pseudo-code	R
=	==
≠	!=
<	<
≤	<=
>	>
≥	>=

Logical operators:

Pseudo-code	R
NOT	!
AND	&
OR	|

Conditional execution – structure of a test

IF …

Pseudo-code

```
If BooleanVariable then
    Instructions
End of If
```

R

```
if (BooleanVariable) {
   # statement(s) will
   # execute if the boolean
   # expression is true.
}
```

IF … ELSE …

```
If BooleanVariable then
    Instructions
Else then
    Instructions 2
End of If
```

```
if(BooleanVariable) {
   # statement(s) will execute if the
   # boolean expression is true.
} else {
   # statement(s) will execute if the
   # boolean expression is false.
}
```

Chained tests

Examples:

```
Begin
Write "Enter water temperature:"
Read Temp
If Temp ≤ 0 then
    Write "This is ice"
End of If
If Temp > 0 AND Temp < 100 then
    Write "This is liquid"
End of If
If Temp ≥ 100 then
    Write "This is vapor"
End of If
End
```

```
Begin
Write "Enter water temperature:"
Read Temp
If Temp ≤ 0 then
    Write "This is ice"
Else then
    If Temp < 100 then
        Write "This is liquid"
    Else then
        Write "This is vapor"
    End of If
End of If
End
```

What could be a good graphical representation for this code?

A bit more about logic

- Parentheses are important:
  `(A and B) or C` is different from `A or (B and C)`
- AND versus OR

Boolean algebra:

```
A     and   !B    =>    C
!A    or    B     =>    !C
```

Example:

```
If it is too hot AND it is not raining then
   Open the window
Else then Do not open the window

If it is not too hot OR it is raining then
   Do not open the window
Else then Open the window
```

A It is too hot
B It is raining
C Open the window

Repetition and looping

There are two types of loops:
- Specific number of iterations – `For`
- No specific number of iterations – `While`

Structure of the `For` loop:

```
For i varying from 1 to 70 then
   Hello Name(student i)!
End For
```

Structure of the `While` loop:

```
While number < 60 then
   getPlace = true
   number = number + 1
End While
```

- You can specify how `i` should vary (e.g., by 2).
- In R, you can simply give a vector of values for `i`.
- You can name the variable something different from `i`.

Take-home message

- An algorithm is a list of instructions to execute.
- You need to have the algorithm in mind before starting to write it as a script in the chosen language.

What you should do now

```
Begin
While it is earlier than 17:00
   Work on Exercise Sheet 1
   If you are done with Exercise Sheet 1 then
      If you are still motivated for the R course then
         Install R
      Else then
         Enjoy your evening
      End of If
   Else then
      Enjoy your evening
   End of If
End of While
End
```