An introduction to R:
Organisation and Basics of Algorithmics

Noémie Becker, Benedikt Holtmann & Dirk Metzler [1]

nbecker@bio.lmu.de - holtmann@bio.lmu.de

Winter semester 2016-17

[1] Special thanks to: Prof. Dr. Martin Hutzenthaler and Dr. Sonja Grath for course development

1. Organisation of the course

2. Basic algorithmics
   - Algowhat?
   - Variables
   - Read and write
   - Tests and logic
   - Loops
   - Take home message

Where to find information about the course?

- Webpage: http://evol.bio.lmu.de/_statgen/Rcourse/ws1617/
- see **Syllabus_R-course_2017.pdf** for global information
- All course material (presentations, scripts, exercise sheets) will be posted on the website.

When and where?

- Course from February 27 to March 10.
- **Lectures and correction of the exercises**: every morning from 9 to 12 am in G00.001
- Exception: March 6 in B00.019 and March 9 in B01.027
- **Exercise sessions**: every day from 1 to 5 pm in C00.005, G00.037 or D00.021 (laptop). You can also work from home of course.

How do I get my 3 ECTS?

- **Exam** on March 10 at 10 am in B00.019
- Recap on April 3 at 9 am - register before March 25.

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

- Monday Feb 27: Bascis of algorithmics
- Tuesday Feb 28: Getting started with R
- Wednesday March 1: Data types and structure
- Thursday March 2: Data types and structure (continued)
- Friday March 3: Programming in R

Course outline - Week 2

- Monday March 6: Programming in R (continued)
- Tuesday March 7: Data manipulation
- Wednesday March 8: Data visualization and graphics
- Thursday March 9: Basic statistics with R
- Friday March 10: Exam

Contents

Aknowledgment

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

Why are computers binary?

Who has experience with programming?

Why are computers binary?

Who has experience with programming?

Computers can treat binary information only.
Can you cite examples of binary variables?

Why are computers binary?

Who has experience with programming?

Computers can treat binary information only.
Can you cite examples of binary variables?

The memory of the computers is made of electronic components that can be charged or uncharged. This is the reason why they register information as binary variables symbolized by humans as 0 and 1.

Previous experience with algorithms

Who has already executed an algorithm?

Previous experience with algorithms

Who has already executed an algorithm?

Examples:

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

Previous experience with algorithms

Who has already executed an algorithm?

Examples:

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

Who has already written an algorithm?

Previous experience with algorithms

Who has already executed an algorithm?

Examples:

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

Who has already written an algorithm?

Examples:

- 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.

Definition

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

Definition

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).

But in our case, it is easier as computers are all as stupid and do not have cultural background etc...

Algorithmics and Programming

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

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

Contents

Variables what for?

Variables are made to store information

- given by the user
- result of the program

Variables what for?

Variables are made to store information

- given by the user
- result of the program

In practice:

- computer assigns space in memory for information
- computer assigns a label to this space with a binary adress

Variable declaration

Many languages require declaration of variables:

- usually at the beginning of the program
- need to specify the type of variable to allocate enough memory (ex: integer vs decimal)

Variable declaration

Many languages require declaration of variables:

- usually at the beginning of the program
- need to specify the type of variable to allocate enough memory (ex: 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 (easier for user but less effective in terms of memory).

Types of variables

The space allocated in memory depends on the type.

- numerical: integer, decimal
- character/string
- boolean = binary variable

Affectation

Attribute a value to the variable.

Value must belong to defined type (not in R).

Affectation

Attribute a value to the variable.

Value must belong to defined type (not in R).

Can be written in pseudo-code as:
`Var1 <- 24` (in pseudo-code but also in R)

Affectation

Attribute a value to the variable.

Value must belong to defined type (not in R).

Can be written in pseudo-code as:
`Var1 <- 24` (in pseudo-code but also in R)

Can also affect the value of another variable:

```
Var2 <- Var1
Var2 <- Var1 + 4
```

Affectation

Attribute a value to the variable.

Value must belong to defined type (not in R).

Can be written in pseudo-code as:
`Var1 <- 24` (in pseudo-code but also in R)

Can also affect the value of another variable:

```
Var2 <- Var1
Var2 <- Var1 + 4
```

The order of the instructions plays a role of course:

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

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

What is the value of A?

Operators

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

Operators

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

Possible operators depend on the type of variable

- numeric: +, -, *, /, ^
- text: & to concatenate (in pseudo-code not in R)
- boolean: and (& in R), or (| in R), no (! in R)

Contents

Read and write

Necessary to communicate with the user.

Read and write

Necessary to communicate with the user.

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

Example:

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

Contents

Motivating example

Now we can ask info from the user, save it into variables, make operations on variables and print results.

But the algorithm must be flexible and allow for different options.

Example: print in black and white or in colors?

If your doc has colors: tick colors
else: tick black and white.

Structure of a test

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

Structure of a test

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

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

Structure of a test

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

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

Boolean = Expression with value `TRUE` or `FALSE`
Can be a variable or a condition.

Structure of a test

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

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

Boolean = Expression with value `TRUE` or `FALSE`
Can be a variable or a condition.

What would be the pseudo-code for the example above?

Condition

A condition is a comparison.

Value1 + comparison operator + Value2

operators: $=$, $\neq$, $\geq$, $\leq$, $>$, $<$

Examples:

- `"t" < "w"`
- `a` $\geq$ `3`
- `b = 3` in R ==

Composed condition

How to express $5 \leq x \leq 7$.

Composed condition

How to express $5 \leq x \leq 7$.

- AND: means both conditions must be true (& in R)

Composed condition

How to express $5 \leq x \leq 7$.

- AND: means both conditions must be true (& in R)
- OR: means one or both are ture or at least one is true (| in R)

Composed condition

How to express $5 \leq x \leq 7$.

- AND: means both conditions must be true (& in R)
- OR: means one or both are ture or at least one is true (| in R)
- NO: if A ist TRUE then NO(A) is FALSE (! in R)

Composed condition

How to express $5 \leq x \leq 7$.

- AND: means both conditions must be true (& in R)
- OR: means one or both are ture or at least one is true (| in R)
- NO: if A ist TRUE then NO(A) is FALSE (! in R)

Example: ”Document has colors AND you want to print the colors.”

Example: ”Document has colors AND NO(you want to print the colors).”

Chained tests

Example:

```
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
```

Chained tests

Example:

```
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
```

Chained tests

Example:

```
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 can be a good graphical representation for this?

A bit more about logic

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

A bit more about logic

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

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

A bit more about logic

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

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

Can also be written as:

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

Contents

Motivation example

```
Write "Do you like this course?  (Y/N)
Read Answer
If Answer = Y then
    Write "Me too"
Else then
    Write "What a pity!"
End If
```

What happens if the user answers something stupid?

Motivation example

```
Write "Do you like this course?  (Y/N)
Read Answer
If Answer = Y then
    Write "Me too"
Else then
    Write "What a pity!"
End If
```

What happens if the user answers something stupid?

”What a pity!”

1st solution: Test

You want to check if the answer is correct.

```
Write "Do you like this course?  (Y/N)
Read Answer
If Answer NO Y and Answer NO N then
    Write "Please answer by Y or N"
    Read Answer
End If
If Answer = Y then
    Write "Me too"
Else then
    Write "What a pity!"
End If
```

1st solution: Test

You want to check if the answer is correct.

```
Write "Do you like this course?  (Y/N)
Read Answer
If Answer NO Y and Answer NO N then
    Write "Please answer by Y or N"
    Read Answer
End If
If Answer = Y then
    Write "Me too"
Else then
    Write "What a pity!"
End If
```

What happens if the user answers something stupid twice?

1st solution: Test

You want to check if the answer is correct.

```
Write "Do you like this course?  (Y/N)
Read Answer
If Answer NO Y and Answer NO N then
    Write "Please answer by Y or N"
    Read Answer
End If
If Answer = Y then
    Write "Me too"
Else then
    Write "What a pity!"
End If
```

What happens if the user answers something stupid twice?

”What a pity!”

Real solution: Loop

You want to check if the answer is correct.

```
Write "Do you like this course?  (Y/N)
Read Answer
While Answer NO Y and Answer NO N then
    Write "Please answer by Y or N"
    Read Answer
End While
If Answer = Y then
    Write "Me too"
Else then
    Write "What a pity!"
End If
```

`While` loop means that you repeat instruction as long as the condition is TRUE.

Types of loops

There are two types of loop.

- No specific number of iterations
  `While`.
- Specific number of iterations
  `For`

Types of loops

There are two types of loop.

- No specific number of iterations
  `While`.
- Specific number of iterations
  `For`

Structure of the `For` loop:

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

Types of loops

There are two types of loop.

- No specific number of iterations
  `While`.
- Specific number of iterations
  `For`

Structure of the `For` loop:

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

You can specify that i should vary 2 by 2 or any other.
In R you simply give a vector of values for i.
Of course you can name the variable something else as i.

More loops

You can also imbed a loop into another loop.

Many errors due to the name of the counter in `For` loops.

More about that when we will use these notions in R next week.

Contents

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.