Advanced Programming in R

Maria Kubara

Ewa Weychert

Contact information

Maria Kubara

maria.kubara@uw.edu.pl

Ewa Weychert

e.weychert@uw.edu.pl

Aim of the class

- Last chance to catch up with your programming skills, if you are still a beginner
- Learn how to write automatized code that is readable and efficient
- The tools presented here are useful for other courses!

Plan for the course

week	topic	who?
1	conditionals and loops	Ewa
2	creating functions, environments	Ewa
3	funcions 2: defensive programming, assertions, invisible, binary operators	Ewa
4	benchmarking and profiling	Ewa
5	script and report automation (RMarkdown) - documents, presentations and dashboards (flexdashboard)	Ewa
6	object oriented programming, S3 system, generic functions and methods	Maria
7	object oriented programming, S4 (and R6?)	Maria
8	vectorization and purrr functions	Maria
9	basics of Shiny	Ewa
10	shiny 2 - multipanel applications	Ewa
11	advanced data processing with dplyr, dtplyr, tidyr	Maria
12	Using C++ in R (Rcpp)	Maria
13	RCpp2: advanced usage of Rcpp	Maria
14	Creating and testing own package	Maria
15	presentations	both

Passing requirements

The following topics (tools) will be presented during the course:

1. Writing own functions in R (including defensive programming)
2. Object-oriented programming - creating own classes, methods and generic functions of the S3, S4 and R6 systems
3. Advanced data processing with dplyr, dtplyr, tidyr
4. Automation of scripts and reports (RMarkdown)
5. Shiny + creating analytical dashboards
6. Use of C++ in R (Rcpp)
7. Vectorization of the code
8. Creating own R packages

Passing requirements

- **The final grade** for the course will be based on the final project.
- Final projects should be prepared in teams of **at most 3 people**. They can refer to any topic, but they should use **at least four of the tools listed above** (NOTE - **at least two tools for each person in the team**, so a team of three people should use at least the **six tools** above).
- Each team should provide a presentation (prepared in RMarkdown, no more than 10 slides) + complete R codes.

Deadlines

- The team members and a **short initial description of the project** should be sent to the teacher(s) by **April 30, 2022** at the latest.
- **Presentation days:**

Tuesday: **June 14, 2022**

Thursday: **June 16, 2022**

- The deadline for submitting projects is midnight ending the day

June 24, 2022

Plan for todays class: Conditionals and loops

1. Logical expressions

2. Conjunction and alternatives

conjunction - & (and)

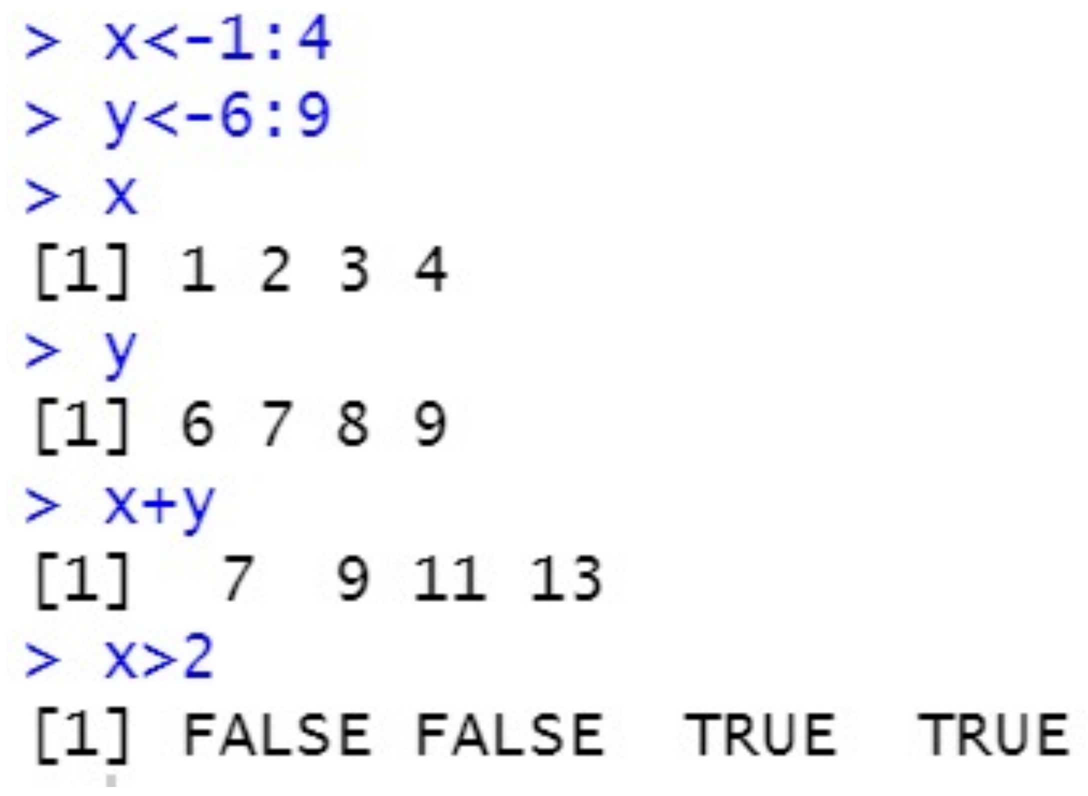
Alternative - | (or)

3. Vectorization

4. If statements in R

5. For loops

Vectorization



```
> x<-1:4
> y<-6:9
> x
[1] 1 2 3 4
> y
[1] 6 7 8 9
> x+y
[1]  7  9 11 13
> x>2
[1] FALSE FALSE  TRUE  TRUE
```

- Aim – in R it makes operation simpler

(add first element of the vector A with the first element of vector b)

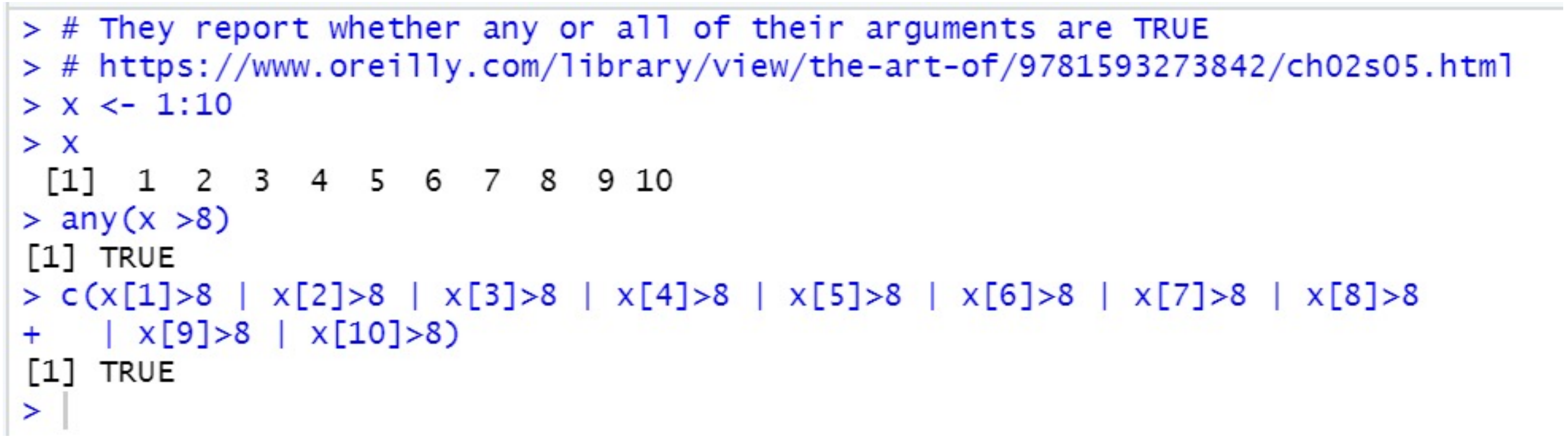
https://www.youtube.com/watch?v=tTUBLwjwriU

Why do we sometimes use && or || instead & or |?

- A good code should be easy to check. It should have many comments that state why we do certain procedure but also it should state what we expect as a result of our code.
- Using & o | → we expect vector of values (several values in c())
- Using && or || → we expect scalar (single value)
- So it depends on what we intend to return as a result of our operations in a code.

Why do we sometimes use && or || instead & or |?

- Thanks to this code is more readable and less prone to mistakes!



```
> # They report whether any or all of their arguments are TRUE
> # https://www.oreilly.com/library/view/the-art-of/9781593273842/ch02s05.html
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> any(x >8)
[1] TRUE
> c(x[1]>8 | x[2]>8 | x[3]>8 | x[4]>8 | x[5]>8 | x[6]>8 | x[7]>8 | x[8]>8
+    | x[9]>8 | x[10]>8)
[1] TRUE
> 
```

If statements in R

- If is an instruction for R to conduct
- One line if statement without {} → I do not recommend. Danger that we will forget to add {} if we expand our condition!!
- Normally we build

```
if (<logical_expression>){
   <functions and commands>
 <functions and commands>
  <functions and commands>
}
```

If else

- Only one condition will be conducted the order matters!!!!

```
# The longer version (but else if can be unlimited):

# if (<logical_expression>){ # mandatory
#    <functions and commands>
#    <functions and commands>
#    <functions and commands>
# } else if (logical_expression>) { # optional
#    <functions and commands>
#    <functions and commands>
#    <functions and commands>
# } else if (logical_expression>) { # optional
#    <functions and commands>
#    <functions and commands>
#    <functions and commands>
# } else { # optional
#    <functions and commands>
#    <functions and commands>
#    <functions and commands>
# }
```

Loops

```
wek <- 1:20
for (i in 1:10) {
print(wek)
}
```

A temporary variable i created in a loop for which we execute the code. It has different value for each iterations

Range of a loop (number of loops (how many iterations)