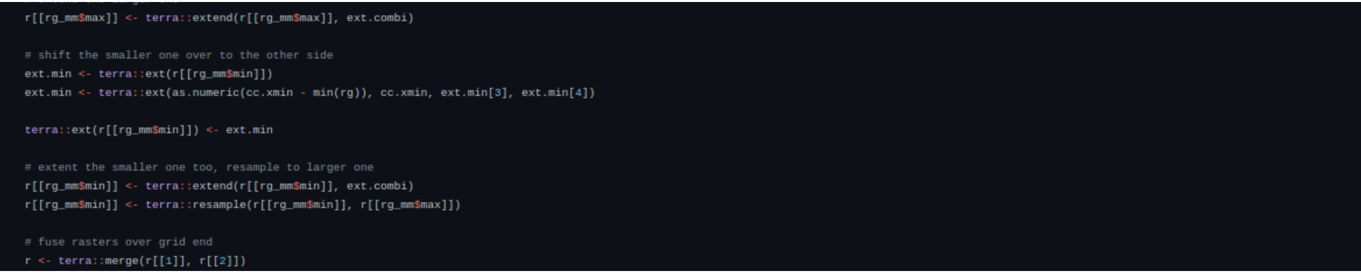


```r
r[[rg_mm$max]] <- terra::extend(r[[rg_mm$max]], ext.combi)

# shift the smaller one over to the other side
ext.min <- terra::ext(r[[rg_mm$min]])
ext.min <- terra::ext(as.numeric(cc.xmin - min(rg)), cc.xmin, ext.min[3], ext.min[4])

terra::ext(r[[rg_mm$min]]) <- ext.min

# extent the smaller one too, resample to larger one
r[[rg_mm$min]] <- terra::extend(r[[rg_mm$min]], ext.combi)
r[[rg_mm$min]] <- terra::resample(r[[rg_mm$min]], r[[rg_mm$max]])

# fuse rasters over grid end
r <- terra::merge(r[[1]], r[[2]])
```

EAGLE M.Sc., MB2 2023/24

# Introduction into programming using R

Part II: Concepts, code & practical exercises

## Short recap

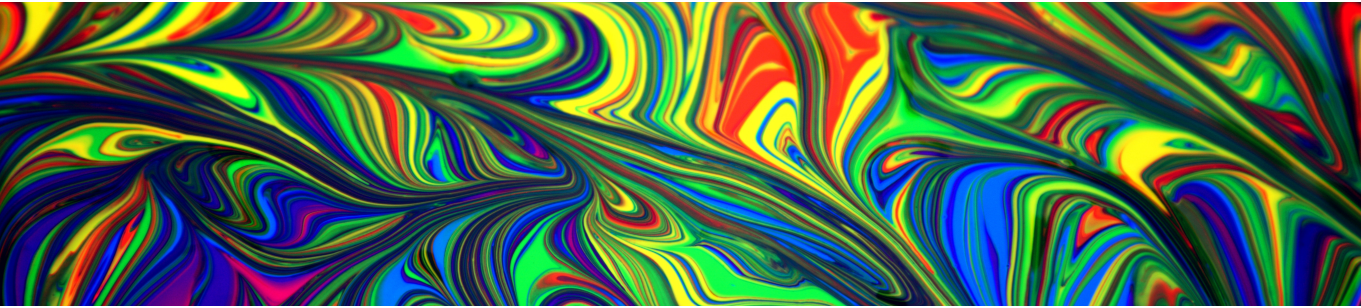
Last week and today morning, you have learned

- what types, structures and classes are in R
- what scalars and (atomic) vectors are in R
- that atomic vectors can be named, adding *attributes* to the vector
- how different structures can be indexed
- how to filter our energy dataset using logical operators for indexing
- how to summarize our dataset, e.g. to count occurences of values per variable
- more indexing
- how to use ggplot2, a powerful plotting library, for data visualization
- how to use ggplot2 extensions such as gganimate

Are there any questions/issues regarding the last topics?

## Agenda

- Background: Explicit and implicit type conversion in R
- Practical: Exploration of the course's example ENTSO-E enegery dataset
- Task: Generic plotting/ggplot2



## Explicit and implicit type conversions in R and why it concerns us

## What is type conversion and how can one do it in R?

## Explicit and implicit type conversions

- Type conversion: Converting between types such as *numeric*, *character* or *logical*
- Explicit type conversions: A type conversion is explicitly issued, i.e. you tell R to convert an object from one type to another
- Implicit type conversions: R automatically converts from one type to another without telling you

## Examples of implicit type conversions

Explicit type conversion:

```r
a <- c(1.2, 2.1, 3.4, 4.9, 5)
is.numeric(a)
typeof(a)
class(a)

a <- as.character(a)
is.numeric(a)
typeof(a)
class(a)
```

## Examples of implicit type conversions

Implicit type conversion (1):

```r
a <- c(1.2, 2.1, 3.4, 4.9, 5)
is.numeric(a)
typeof(a)
class(a)

a[1] <- "1.2"
is.numeric(a)
typeof(a)
class(a)
```

What happend here? Did we just assign a character scalar to a double vector? How does R react to this? Try!

## Examples of implicit type conversions

Implicit type conversion (2):

```r
b <- letters[1:10]
is.character(b)
typeof(b)
class(b)

b[1] <- 1
is.character(b)
typeof(b)
class(b)
```

And what happend here? Any difference to the former example?

## Examples of implicit type conversions

Implicit type conversion (3):

```r
d <- c(TRUE, TRUE, TRUE, FALSE, TRUE)
is.logical(d)
typeof(d)
class(d)

d[1] <- 1
is.logical(d)
typeof(d)
class(d)
```

Insights on this one?

## Implicit type conversions

**Quick assesment of this behavior:**

- In certain situations, R converts from one type to another without telling you
- Where other languages would throw an error, R remains silent and silently converts things
- There seems to be a direction of implicit conversions from one type to another, usually differing types are converted into *character* $\rightarrow$ conversion rules

Why does R do that? Can you think of any scenario where (a) this is usefull and (b) where it is dangerous?

## Examples of implicit type conversions

A rather usefull case of implicit type conversion:

```r
year <- 2023
x <- "EAGLE students"

paste("The", year, x, "like to code :-)")
```

```
[1] "The 2023 EAGLE students like to code :-)"
```

To those proficient in Pyhton: How would the python interpreter react to the Python version of this example?

```python
year = 2023
x = "EAGLE students"
"The " + 2023 + " " + x + " like to code :-)"
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

## Examples of implicit type conversions

A dangerous case of implicit type conversion (pls. never do this):

```r
x <- "0.3"
y <- 0.01
y < x

x <- "0.3"
y <- 0.000000000000001
y < x
```

What happens here under the hood that we cannot see?

```r
as.character(0.3)
as.character(0.000000000000001)

"1e-15" < "0.3"
#[1] FALSE
1e-15 < 0.3
#[1] TRUE
```

A binary opeartor used on characters is comparing them lexicographically!

## Examples of implicit type conversions

Binary operators on characters lead to a lexicographic comparison of them, depending on the locale set in your OS (e.g. “en-US” using the US alphabet):

```r
"A" > "B"
"C" > "B"

"Adam" > "Bertha"
"Chris" > "Bertha"
"Adam" > "Ariel"
```

What happens here? A vector element is treated as a full word. If the leading character is equal, the second character is collated.

```r
"42" > "8"
"1002" > "2"
1002 > "2" # implicit type conversion to character
"1002" > 2 # implicit type conversion to character
1002 > 2

"A" > TRUE # implicit type conversion to character
```

Conclusion: (1) Make sure your objects really are handed over in the mode you believe them to be! (2) There are nearly no cases where you would need to compare characters using binary operators except for checking alphabetic order.

## Coding

Working with the ENTSO-E dataset

## Joint coding

Energy Production Dataset: Exploration

## Task: Energy Production Dataset Exploration

Task: Use the **Energy Production Dataset** from last session:

- Use ggplot2 or generic plotting to create summary graphics of the dataset, for example:
  - Plot actual vs. installed capacity – DONE
  - Plot number of plants per production type (counts graph) – DONE
  - Plot production capacity per type – DONE
  - Plot actual generation per production type
  - Plot production capacity vs. acutal production per type
  - Plot actual production over the course of the week
  - Plot installed capacity over the course of the week
  - ...
- Interpret the plots – what do they tell about the data?
- Save your plots to show them later-on

Extras: (1) Find out what the temporal resolution of the dataset is. (2) Try to find out the number of records per each power plant. Are they all the same? Do some report more often then others? (3) Create an animation using *gganimate* that shows how actual generation per type changes day-wise.

Any questions?

## Feedback

Anonymous feedback via slido.com, event code **#mb2training**.

What we would like to know from you:

- Is the speed ok?
- Is the session structure ok?
- Is there any content you'd like to be covered?
- Is it boring or not?
- Is it helpful or not?
- Is the presentation style ok?
- Anything else?

Go to slido.com & join #mb2training

**Please give feedback using the feedback box.**

**Coffee time – see you next week**