Vectors & Data frames. Functions.

cinkova@ufal.mff.cuni.cz

Vector classes + coercion

- logical + numeric = numeric
- boolean values translate to 0 and 1

```
boolean_vec <- c(TRUE, TRUE, FALSE) # boolean
numeric_vec <-  c(3, 2, 6) # numeric
c(boolean_vec, numeric_vec)
[1] 1 1 0 3 2 6
class(c(boolean_vec, numeric_vec))
[1] "numeric"
```

Vector classes + coercion

- anything + character = character
- even digits can be characters

```
mix_vec <- c(boolean_vec,
             numeric_vec,
             "June 8", "3 ")
class(mix_vec)
[1] "character"
```

Vector subsetting by positions

```
vec <- c("John", "Mary", "Paul")
vec[1]
[1] "John"
vec[2:3]
[1] "Mary" "Paul"
vec[c(1,3)]
[1] "John" "Paul"
```

Vector subsetting with logical operators

```
vec <- c(1, 20, 3, 4)
vec[vec < 20]
[1] 1 3 4
vec[vec < 20 & vec > 1] # and
[1] 3 4
vec[vec > 15 | vec < 3 ] # or
[1]  1 20
```

Vector recycling

- many functions proceed element by element
- nothing gets recycled with equally long vectors

```
c(2,4,6,8)/c(2,2,2,2)
[1] 1 2 3 4
c(1000, 100, 10) / c(100, 10, 1)
[1] 10 10 10
```

Vector recycling

- The second vector contains just one value and that must serve each element of the first vector
- `2` gets *recycled*

```
c(2,4,6,8) / 2
[1] 1 2 3 4
```

Vector recycling

- the second vector gets recycled once
- each of its element must serve twice
- R believes you want it this way

```
c(2,4,6,8) / c(2,1)
[1] 1 4 3 8
```

Vector recycling

- Did you really want this? Warning.

```
c(10, 10, 10) * c(1,2)
Warning in c(10, 10, 10) * c(1, 2): longer
object length is not a multiple of
shorter object length
[1] 10 20 10
```

Exercise

- Feed this function with 2 or 3 vectors you create in your own script.
- Experiment with the numbers of their elements to get a feel for recycling.

```
library(stringr)
vec1 <- c("a", "b", "c")
vec2 <- c("A", "B", "C")
str_c(vec1, vec2, sep = "*and*")
[1] "a*and*A" "b*and*B" "c*and*C"
```

Functions - `?sort`

sort {base} R Documer

Sorting or Ordering Vectors

Description

Sort (or *order*) a vector or factor (partially) into ascending or descending order. For ordering along more than one variable, e.g., for sorting data frames, see `order`.

Usage

```
sort(x, decreasing = FALSE, ...)
```

Function arguments

```
sort(x = c(2, 1, NA), decreasing =
FALSE)
[1] 1 2
```

Arguments

`x`	for `sort` an **R** object with a class or a numeric, complex, character or logical vector. For `sort.int`, a numeric, complex, character or logical vector, or a factor.
`decreasing`	logical. Should the sort be increasing or decreasing? Not available for partial sorting.
`...`	arguments to be passed to or from methods or (for the default methods and objects without a class) to `sort.int`.
`na.last`	for controlling the treatment of `NA`s. If `TRUE`, missing values in the data are put last; if `FALSE`, they are put first; if `NA`, they are removed.

Arguments' order

- The function expects the arguments in the defined order.
- If you keep this order, you can just type their values.

```
sort(c("b", "a", "d"), # this was `x`
     FALSE) # this was `decreasing`
[1] "a" "b" "d"
```

Arguments' order

- If not sure about their order, type argument names.

```
sort(decreasing = FALSE,
     x = c("b", "a", "d"))
[1] "a" "b" "d"
```

Arguments' order

- otherwise errors or, worse, unexpected results

```
sort(FALSE, c(1,2,3))
Error in sort(FALSE, c(1, 2, 3)):
'decreasing' must be a length-1 logical
vector.
Did you intend to set 'partial'?
```

Default values

- Using a function for the first time, check defaults!

```
sort(c(100, 99, 98))
[1]  98  99 100
sort(c(100, 99, 98),
     decreasing = TRUE)
[1] 100  99  98
```

Exercise

- Study functions under `list.files`.
- Check that your working directory is your home.
  - `getwd()`
  - `setwd("~")` when you are elsewhere
- List all files and folders you see in your home folder.

More exercises in quiz

Get your cell phones ready to scan a QR code!

Copy teacher’s folder home

- use function `file.copy`
- path: `"../cinkova/2024-10-11__02"`

Look up and use functions

- `readLines`
  - to read the first 10 lines of the csv file `Atlantyk`…
  - look up `read.delim` and pick a function to read the entire file as a table

A madly difficult exercise

- create this output with the function `stringr::str_c`

```
a <- c("lemon peels", "banana pulp",
"cherry sap")
b <- c(1, 2, 3)
c <- c("tbsp", "cups", "glasses" )
```

```
[1] "1 tbsp lemon peels, 2 cups banana
pulp, 3 glasses cherry sap"
```

Clue to Madly difficult exercise

```
a <- c("lemon peels", "banana pulp",
"cherry sap")
b <- c(1, 2, 3)
c <- c("tbsp", "cups", "glasses" )
str_c(b, c, a, sep = " ", collapse = ", ")
[1] "1 tbsp lemon peels, 2 cups banana
pulp, 3 glasses cherry sap"
```