R is a functional language

Marco Pascucci

25/10/2018

list of function syntax

```r
summary <- function(x) {
  funs <- c(mean, median, sd, mad, IQR)
  lapply(funs, function(f) f(x, na.rm = TRUE) )
}

y <- summary(1:10)
str(y)
```

```
## List of 5
##  $ : num 5.5
##  $ : num 5.5
##  $ : num 3.03
##  $ : num 3.71
##  $ : num 4.5
```

Closures

functions that write functions

FORMULA

Exercice

expliquer le code suivant:

```r
library(purrr)
iris %>% group_by(Species) %>%
nest(.key = Data) %>%
mutate(Model = purrr::map(Data, ~ lm(data = .,
  Sepal.Length ~ Petal.Length))) %>%
mutate(Summary = purrr::map(Model, summary)) %>%
mutate(`R squared` = purrr::map_dbl(Summary, ~ .$r.squared))
```

comme les “fonctions anonimes”...

```r
increment <- function(x) {
  x+1
}

v = 0:9

v %>% increment
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
v %>% (function(x) {x+1})
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Formula et l'operateur ~

Il existe une autre syntaxe, la `formula`, qui décrit une relation entre des variables, mais à difference d'une fonction, elle est seulement formelle.

```
y ~ x1 + x2
```

```
## y ~ x1 + x2
```

cette `formula` veut seulement dire `y` depend de `x1` et `x2`.

Certaines fonctions, comme par exemple `map` peut prendre des `formulae` aussi bien que des fonctions en argument.

```
before <- 0:9
map(before, ~.x+1) %>% cbind(before, after=.)
```

```
##       before after
##  [1,] 0      1
##  [2,] 1      2
##  [3,] 2      3
##  [4,] 3      4
```

coercion

explicit coercion (casting)

```r
as.character(1)
```

```
## [1] "1"
```

```r
as.logical(1)
```

```
## [1] TRUE
```

implicit coercion happens, for example, when data of different class are inserted in the same vector.

```r
c(1, TRUE, "a")
```

```
## [1] "1"    "TRUE" "a"
```

in this example all becomes character.

Imprecise coercion results in `NA`

```r
as.numeric('f')
```

Factors

```r
unordered <-factor(c("one", "two", "three"))
ordered <- factor(c("one", "two", "three"),
                  c("one", "two", "three"))

str(unordered)
```

```
##  Factor w/ 3 levels "one","three",..: 1 3 2
```

```r
str(ordered)
```

```
##  Factor w/ 3 levels "one","two","three": 1 2 3
```

factor are assigned in alphabetical order