R avancé

partie 2 : Fonctions

Romain François

romain@r-enthusiasts.com

@romain_francois

Quiz

- Quels sont les trois composants d'une fonction ?
- Que retourne ce code:

```
y <- 10
f1 <- function(x){
  function(){
    x + 10
  }
}
f1(1)()
```

- Comment écrit-on ce code classiquement:

```
`+`( 1, `*`(2 ,3) )
```

Uwe Ligges. R Journal

Quiz

- Comment rendre ce code plus facile à lire:

```
mean( , TRUE, x = c(1:10, NA) )
```

- Est-ce que cette fonction retourne une erreur:

```
f2 <- function(a, b){
  a * 10
}
f2( 10, stop("This is an error") )
```

Composants d'une fonction

- Le `body()`: Le code à l'intérieur de la fonction
- Le `formals()`: La liste des arguments
- Le `environment()` : Une "table" des emplacements des variables d'une fonction

Une fonction du globalenv()

```
> f <- function(x) x^2
> f
function(x) x^2

> formals(f)
$x

> body(f)
x^2

> environment(f)
<environment: R_GlobalEnv>
```

Une fonction d'un package

```
> rnorm
function (n, mean = 0, sd = 1)
.Call(C_rnorm, n, mean, sd)
<bytecode: 0x7fe9ed01b720>
<environment: namespace:stats>

> formals(rnorm)
$n

$mean
[1] 0

$sd
[1] 1

> body(rnorm)
.Call(C_rnorm, n, mean, sd)
> environment(rnorm)
<environment: namespace:stats>
```

Quiz

- Quelle fonction indique si un object est une fonction ?

?

- Ce code donne la liste des fonctions de base

```
objs <- mget( ls("package:base"), inherits = TR
funs <- Filter( is.function, objs )
```

Utilisez funs pour répondre à :

- Quelle fonction a le plus grand nombre d'arguments
- Combien de fonctions n'ont aucun argument

```
x <- 2
g <- function(){
  y <- 1
  c(x, y)
}
g()
```

```
x <- 1
h <- function(){
  y <- 2
  i <- function(){
    z <- 3
      c(x,y,z)
  }
  i()
}
h()
```

```
j <- function(x){
  y <- 2
  function(){
    c(x,y)
  }
}
k <- j(1)
k()
```

```
n <- function(x) x/2
o <- function(){
  n <- 10
  n(n)
}
o()
```

```
f <- function() x
x <- 15
f()
```

```
x <- 20
f()

>codetools::findGlobals(f)
[1] "x"
```

Quiz

- Que retourne cette fonction ? Quelle est la signification de chacun des "c"

```
c <- 10
c(c=c)
```

- Et ça:

```
f <- function(x){
  f <- function(x){
    f <- function(x){
      x^2
    }
    f(x+1)
  }
  f(x*2)
}
f(3)
```

Arguments des fonctions

- 1) Match exact
- 2) Match partiel
- 3) Position

```
f <- function(abcdef, bcde1, bcde2){
  list(a=abcdef, b1=bcde1, b2=bcde2)
}
str(f(1,2,3))
str(f(2,3,abcdef=1))
str(f(2,3,a=1))
str(f(1,3,b=1))
```

```
mean(1:10)
mean(1:10, trim=.05)

mean(x=1:10)
```

```
mean(1:10, n=T)
mean(1:10,, FALSE)
mean(1:10, 0.05)
mean(,TRUE, x=c(1:10,NA))
```

```
args <- list( 1:10, na.rm=TRUE)
do.call( mean, args )

# equivalent à:
mean( 1:10, na.rm = TRUE)
```

```
g <- function(a=1, b=a*2){
  c(a,b)
}
g()
g(10)
g(1,4)
g(b=2)
```

Evaluation paresseuse (lazy evaluation)

```
f <- function(x){
   10
}
f( stop("🔥") )
```

Evaluation paresseuse (lazy evaluation)

```
f <- function(x=ls()){
  a <- 1
  x
}
f()
f(ls())
```

Plus d'informations

Advanced R, chapitre 6