Funkce v R

Většina funkcionality R je dána vestavěnými funkcemi. Je-li to nutné, je možné definovat vlastní funkce, např.:

```
sectiCtverce <- function(a, b){
        # vrací součet čtverců čísel "a" a "b"
        return(a ^ 2 + b ^ 2)
}
```

Lokální, globální proměnné

- Globální proměnné platí všude v rámci scriptu. V R se jim raději vyhněte. Pokud se bez nich ale neobejdete, definujete je takto `x <<- 1 # x je globální proměnnou`
- Ostatní proměnné jsou lokální.
- Je-li proměnná definovaná uvnitř funkce, platí jen tam.
- Použijeme-li ve funkci proměnnou, která v ní není definována, R hledá proměnnou tohoto jména o level výš.

```
x <- 1
f <- function(x){return(2 * x)}

f(x = 5) # 10
x # 1
```

Varování

- Textová hláška vrácená funkcí nebo procedurou
- Nejde o maligní chybu `log(-5) # NaN; In log(-5) : NaNs produced.`
- Lze zavést i vlastní, například

```
logaritmuj <- function(x){
        # vrací přirozený logaritmus čísla "x"
        if(x <= 0){
            print( "x je nekladné, bude vráceno NaN" )
        }
        return(suppressWarnings(log(x)))
}

logaritmuj(-5) # NaN; x je nekladné, bude vráceno NaN
```

Chyby

- Ochranný mechanismus funkcí, který zabrání dalšímu provádění kódu

```
"1" + "1" # Error: non-numeric argument to binary

# operator lze zavést i vlastní, například
sectiCtverce <- function(a, b){
        # vrací součet čtverců čísel "a" a "b"
        if(!is.numeric(a)){stop("a musí být číslo!")}
        if(!is.numeric(b)){stop("b musí být číslo!")}
        return(a ^ 2 + b ^ 2)
}

sectiCtverce(1, 2) # 5
sectiCtverce(1, "2") # Error: b musí být číslo!
```

Rodina funkcí apply()

- Jde o funkce dobře optimalizované tak, že v rámci svého vnitřního kódu „co nejdříve“ volají C++ ekvivalenty R-kové funkce
- Díky tomu jsou exekučně rychlé
- Nejužitečnější je `apply()` a `lapply()`

```
# vrací průměry nad všemi sloupci "mtcars"
x <- apply(mtcars, 2, mean)

# méně šikovně to samé
x <- NULL
for(i in 1:dim(mtcars)[2]){
    x <- c(x, mean(mtcars[, i]))
}
names(x) <- colnames(mtcars)
```

apply()

- Vrací vektor výsledků funkce FUN nad maticí či datovou tabulkou X, kterou čte po řádcích (MARGIN = 1), nebo sloupcích (MARGIN = 2)
- Syntaxe je apply(X, MARGIN, FUN, ...)

```
apply(mtcars, 2, mean)
my_start <- Sys.time()
x <- apply(mtcars, 2, mean)
my_stop <- Sys.time(); my_stop - my_start # 0.019s

my_start <- Sys.time()
x <- NULL
for(i in 1:dim(mtcars)[2]){
    x <- c(x, mean(mtcars[, i]))
    names(x)[length(x)] <- colnames(mtcars)[i]
}
my_stop <- Sys.time(); my_stop - my_start # 0.039s
```

lapply()

- Vrací list výsledků funkce FUN nad vektorem či listem X
- Syntaxe je lapply(X, FUN, ...)
- Skvěle se hodí pro přepis for() cyklu do vektorizované podoby!
- Vhodná i pro adresaci v listu

```
set.seed(1)
my_long_list <- lapply(
    sample(c(80:120), 100, TRUE),
    function(x) sample(
        c(50:150), x, replace = TRUE
    )
) # list vektorů náhodné délky generovaných z náhodných čísel

lapply(my_long_list, "[[", 14) # z každého prvku listu (vektoru) vybírám jen jeho 14. prvek
```