Rcpp

Hynek Handlíř, Jan Labský

O čem to bude

- Základní info
- Jak to funguje
- Praktický příklad: definice funkce v R pomocí C++
- K čemu je to dobré?

Rcpp

- Balíček pro integraci programovacího jazyku C++ do R

- Tvorba balíčků
  - 1000+ balíčků na CRAN využívá funkcí Rcpp

- Dirk Eddelbuettel, Romain Francois, 2011
  - Rcpp: Seamless R and C++ Integration
  - https://www.jstatsoft.org/article/view/v040i08

Jak to funguje

- Různé jazyky používají různé míry abstrakce
  - Garbage collector
  - Efficiency vs Universality
- Převod do strojového jazyka (0100101…)

- High level (R, Python, C#) → low level (C, C++) → assembly language („assembler“) → strojový kód

- Low level jazyky šetří výkon

Příklad

- Počítání Fibonacciho řady
  - 1, 1, 2, 3, 5, 8, 13, 21, 34, 55…
  - Náročné na výpočet

$$f(n) = \begin{cases} 0 & if \quad n = 0 \\ 1 & if \quad n = 1 \\ F(n-1) + F(n-2) & if \quad n > 1 \end{cases}$$

- Porovnání rychlosti C++ a R pomocí balíčku benchmark

```r
# Definice nové funkce ----
Fibonacci_R <- function(n) { # vrátí n-té číslo z Fibonacciho posloupnosti
  if (n < 2) return(n)
  return(Fibonacci_R(n-1) + Fibonacci_R(n-2))
}
```

```
> # Volání nové R funkce ----
> Fibonacci_R(4)
[1] 3

> Fibonacci_R(6)
[1] 8

> sapply(0:10, Fibonacci_R)
 [1]  0  1  1  2  3  5  8 13 21 34 55

> # Čas běhu R funkce
> benchmark(sapply(0:20, Fibonacci_R))[ , 1:4]
                       test replications elapsed relative
1 sapply(0:20, Fibonacci_R)          100     4.2        1
```

```
# Definice funkce pomocí C++ ----
cppFunction(
  "int Fibonacci_C(int n) {
      if (n < 2) return(n);
      return(Fibonacci_C(n-1) + Fibonacci_C(n-2));
  }"
)
```

```
> # Volání nové C++ funkce ----
> Fibonacci_C(4)
[1] 3

> Fibonacci_C(6)
[1] 8

> sapply(0:10, Fibonacci_C)
 [1]  0  1  1  2  3  5  8 13 21 34 55

> # Čas běhu C++ funkce ----
> benchmark(sapply(0:20, Fibonacci_C))[ , 1:4]
                        test replications elapsed relative
1 sapply(0:20, Fibonacci_C)          100    0.03        1
```

```
> # Porovnání obou funkcí
> benchmark(sapply(0:20, Fibonacci_C), sapply(0:20, Fibonacci_R))[ , 1:4]
                        test replications elapsed relative
1 sapply(0:20, Fibonacci_C)          100    0.03    1.000
2 sapply(0:20, Fibonacci_R)          100    4.25  141.667
```