Advanced R Programming - Lecture 6
Rcpp

Krzysztof Bartoszek
(slides based on Leif Jonsson's and Måns Magnusson's)

Linköping University

krzysztof.bartoszek@liu.se

29 IX, 1 X 2020 (Zoom)

Today

Rcpp

Memoization

Questions since last time?

Rcpp

Using C++ code in R

Need C++ compiler (look `http://adv-r.had.co.nz/Rcpp.html`)

Often called interfacing

Similar can be done with Java and Fortran

Extremely fast!

But just handle bottlenecks!

Fibonacci

$$f(n) = \begin{cases} n, & \text{if } n<2 \\ F(n-1) + F(n-2), & \text{otherwise} \end{cases}$$

Fibonacci R

```
fr <- function(n) {
  if (n < 2) return(n)
  fr(n-1) + fr(n-2)
}
```

```
system.time(fr(33))
user  system elapsed
3.312    0.008    3.32
```

Fibonacci C++

```
library(Rcpp)

cppFunction(code = '
  int fcpp(int n) {
    if (n < 2) return(n);
    return(fcpp(n-1) + fcpp(n-2));
  }
')

system.time(fcpp(33))
user  system elapsed
0.019   0.000   0.019
```

Memoization

A simple optimization technique
Example of a general technique in optimization of trading memory for computation

Memoization stores (caches) results of function calls

If called again, returns old value

Depends on functional programming

Useful in recursive programming
(instead of manual look–up structure)

Memoise in R

```
> library(memoise)
> a <- function(x) runif(1)
> replicate(3, a())
[1] 0.6709919 0.3490709 0.4772027
> b <- memoise(a)
> replicate(3, b())
[1] 0.1867441 0.1867441 0.1867441
```

Memoise in R

```
> c <- memoise(function(x) {Sys.sleep(1); runif(1)})
> system.time(print(c()))
[1] 0.7816399
user  system elapsed
0.003   0.004   1.001
> system.time(print(c()))
[1] 0.7816399
user  system elapsed
0.001   0.000   0.000
> forget(c)
[1] TRUE
> system.time(print(c()))
[1] 0.9234995
user  system elapsed
0.003   0.004   1.001
```

Memoise Fibonacci

```
library(memoise)

frm <- memoise(function(n) {
  if (n < 2) return(n)
  frm(n-1) + frm(n-2)
})

system.time(frm(33))
user  system elapsed
0.029   0.000   0.029
```

The End... for today.
Questions?
See you next time!