Lecture 5: Using C++ in R

Ciaran Evans

Previously

```
arma::vec my_lcgC(int n, uint32_t x0,
                  uint64_t m = 4294967296,
                  uint32_t a = 1664525,
                  uint32_t c = 1013904223){
  arma::uvec x(n);

  x[0] = x0;
  for(int i = 1; i < n; i++){
    x[i] = (a*x[i-1] + c) % m;
  }

  arma::vec u = arma::conv_to<arma::vec>::from(x);
  return u/m;
}
```

Get the function into R

defines an R function with C++ source code

package allows us to use C++ source code in R

```
Rcpp::cppFunction('arma::vec my_lcgC(int n, uint32_t x0,
                           uint64_t m = 4294967296,
                           uint32_t a = 1664525,
                           uint32_t c = 1013904223){
  arma::uvec x(n);

  x[0] = x0;
  for(int i = 1; i < n; i++){
    x[i] = (a*x[i-1] + c) % m;
  }

  arma::vec u = arma::conv_to<arma::vec>::from(x);
  return u/m;
}', depends = "RcppArmadillo")

my_lcgC(5, 1)
```

Armadillo library

(pass C++ function definition as a string)

package in R that allows us to use Armadillo library

call it in R session

Now we have my_lcgC in R

```
## [1] 2.328306e-10 2.364555e-01 3.692707e-01 5.042420e-01
```

Comparing R and C++ speed

benchmarking: process of comparing code speed & memory use

compares different lines of code

```
bench::mark(
  my_lcg(1000, 1),
  my_lcgC(1000, 1),
  check=F
)
```

(R) my_lcg — two different pieces of code to compare

(C++) my_lcgC

check=F — don't require the output to be identical

median

time to run code

iterations per second

```
## # A tibble: 2 x 6
##   expression                min   median `itr/sec` mem_allo
##   <bch:expr>           <bch:tm> <bch:tm>     <dbl> <bch:byt
## 1 my_lcg(1000, 1)     105.53us 108.77us     9082.   84.85K
## 2 my_lcgC(1000, 1)      5.37us   6.11us   158570.    7.86K
```

runs each piece of code multiple times, aggregates the results

Use benchmarking to assess performance of code

Some key points

- C++ can be faster than an equivalent implementation in R, especially loops/iteration
- C++ can be more general-purpose, and provides a wider variety of certain data types
- C++ *always* needs to know the type of an object
  - This is true for inputs, outputs, *and* any variables you create
- In C++, indexing begins at 0
- C++ needs a ; at the end of each line
- The Armadillo library provides many useful objects and functions that behave similarly to R counterparts

Using C++ in R

```r
Rcpp::cppFunction('double sumC(arma::vec x) {
  int n = x.n_elem;
  double total = 0;
  for(int i = 0; i < n; ++i) {
    total += x[i];
  }
  return total;
}', depends = "RcppArmadillo")

x <- rnorm(10000)
sumC(x)
```

```
## [1] 70.06122
```

`Rcpp` and `RcppArmadillo` allow us to write functions in R with C++ source code (and the Armadillo library)

What does this require?

- C++ installed on your computer
- `Rcpp` and `RcppArmadillo` R packages installed

Challenge: Getting `Rcpp`, and especially `RcppArmadillo`, to work on your personal computer can very difficult (particularly with Macs)

Solution: we are going to use an RStudio Server provided by the DEAC cluster

Steps

1. Log on to class DEAC OnDemand site: https://sta379.deac.wfu.edu/
2. Open RStudio app
3. Request resources from DEAC cluster to initialize RStudio session
4. Work in RStudio session through your browser
5. Save work, commit and push to GitHub
6. Quit RStudio session

Full, detailed instructions provided on course website

Practice questions and Homework 2

Practice questions:

https://sta379-s25.github.io/practice_questions/pq_4.html

HW 2:

https://sta379-s25.github.io/homework/hw2.html

- Practice writing functions with C++ source code
- Accept the assignment through GitHub classroom
- Work on RStudio server; clone the GitHub repo in your RStudio server session (see instructions on course website)