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;
}
```

Comparing R and C++ speed

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

```
## # A tibble: 2 x 6
##   expression             min   median `itr/sec` mem_alloc
##   <bch:expr>        <bch:tm> <bch:tm>     <dbl> <bch:byt>
## 1 my_lcg(1000, 1)  105.41us  108.4us     9089.    84.85KB
## 2 my_lcgC(1000, 1)   5.37us    5.9us   161325.     7.86KB
```

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

```
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] -20.68701
```

`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)