Lecture 26: C++ and Rcpp

A snippet of C++ code in R

```r
Rcpp::cppFunction('int add(int x, int y, int z) {
  int sum = x + y + z;
  return sum;
}')

add(1, 2, 3)
```

```
[1] 6
```

What is this code doing?

C++ code

```cpp
int add(int x, int y, int z) {
  int sum = x + y + z;
  return sum;
}
```

What are some differences between C++ and R code?

C++ code

Here’s another function:

```cpp
int signC(int x) {
  if (x > 0) {
    return 1;
  } else if (x == 0) {
    return 0;
  } else {
    return -1;
  }
}
```

What similarities do you notice between C++ and R?

C++ code

```cpp
double sumC(NumericVector x) {
  int n = x.size();
  double total = 0;
  for(int i = 0; i < n; ++i) {
    total += x[i];
  }
  return total;
}
```

What is this code doing?

Comparing R and C++ speed

```r
Rcpp::cppFunction('double sumC(NumericVector x) {
  int n = x.size();
  double total = 0;
  for(int i = 0; i < n; ++i) {
    total += x[i];
  }
  return total;
}')

x <- rnorm(1000)
bench::mark(
  sum(x),
  sumC(x)
)
```

```
# A tibble: 2 × 6
  expression      min   median `itr/sec` mem_alloc `gc/sec`
  <bch:expr> <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
1 sum(x)     113.33µs 113.58µs     8701.        0B        0
2 sumC(x)      2.25µs   3.12µs   320880.    2.49KB        0
```

C++ code

```
NumericVector col_meansC(NumericMatrix x) {
  int n_cols = x.ncol();
  int n_rows = x.nrow();
  NumericVector col_means(n_cols);

  double total = 0;

  for(int j = 0; j < n_cols; ++j){
    total = 0;
    for(int i = 0; i < n_rows; ++i){
      total += x(i,j);
    }
    col_means[j] = total/n_rows;
  }

  return col_means;
}
```

Comparing R and C++ speed

```r
x <- matrix(rnorm(1000*150), ncol=150)

bench::mark(
  colMeans(x),
  col_meansC(x)
)
```

```
# A tibble: 2 × 6
  expression         min   median `itr/sec` mem_alloc `gc/sec`
  <bch:expr>    <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
1 colMeans(x)     4.04ms   4.07ms      244.   25.45KB        0
2 col_meansC(x) 123.21µs  124.5µs     7907.    3.71KB        0
```

Some key points

- 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
- `NumericVector` objects are the equivalent of vectors in R
- `NumericMatrix` objects are the equivalent of matrices in R

Class activity

https://sta279-f23.github.io/class_activities/ca_lecture_26.html