Rcpp

Jacob Mortensen

March 6, 2018

Simon Fraser University

Introduction

Introduction

Computation is a frequent bottleneck in statistics research

- R is often too slow
- C can become very complex, and can be very intimidating if you haven’t dealt with it before

Enter Rcpp!

c++ vs. R

R	c++
1. Weakly typed	1. Strongly typed
2. Interpreted	2. Compiled
3. 1-indexed	3. 0-indexed

Advantages of Rcpp

Advantages of Rcpp

Advantages of Rcpp:

- Speed comparable to pure C
- Much easier to use
    - Memory management and data structures
    - Syntactic "sugar"
    - Can use inline code or an external file

Memory management and data structures

R:

```
output <- matrix(0, nrow = sigma2length*mu2length, ncol = 5)
```

C:

```
double* output = (double*) malloc(5*mu2len*sigma2len*sizeof(double));
```

Rcpp:

```
NumericMatrix output(sigma2length*mu2length, 5);
```

Memory management and data structures

Data structures available through Rcpp:

- NumericVector
- IntegerVector
- CharacterVector
- LogicalVector

Equivalent data structures exist for R matrices. All of these data structures are pointers, and so nothing is copied unless needed. Memory is allocated and freed automatically.

Memory management and data structures

Initialize vector:

```
NumericVector x(10);
NumericVector x = NumericVector::create(1,2,3,4,5);
```

Initialize matrix:

```
NumericMatrix x(10, 10);
```

Memory management and data structures

Access and set vector elements:

```
x[0];
x[1] = 4.2;
```

Access and set matrix elements:

```
x(0,0);
x(0,1) = 42;
x(0, _); // returns first row of matrix as a NumericVector
```

Note that these data structures are all indexed by zero!

Syntactic sugar

Many functions from R have been vectorised and implemented in c++

```
NumericVector euclid_dist(double x, NumericVector ys) {
  return sqrt(pow((x - ys), 2));
}
```

Syntactic sugar

R functions implemented in Rcpp:

- Arithmetic and logical operators: *, +, -, /, pow, <, <=, >, >=, ==, !=, !.
- Math functions: abs(), beta(), exp(), gamma(), …
- Summary functions: mean(), min(), max(), sum(), sd() and var()
- d/p/q/r for all standard distributions in R

Code examples

Rcpp Armadillo

Rcpp Armadillo

RcppArmadillo is a linear algebra library, so anytime you need to

- invert a matrix
- perform matrix algebra
- decompose a matrix

this is where you should look.

Rcpp Armadillo Data Structures

Use `arma::colvec` and `arma::mat` in place of `NumericVector` and `NumericMatrix`.

Can convert between the two by using:

```
NumericVector x = wrap(y);
arma::colvec y = as<arma::colvec>(x);
```

and

```
NumericMatrix x = wrap(y);
arma::mat y = as<arma::mat>(x);
```

Rcpp Armadillo Functions

If you need to do any kind of linear algebra operation, it is probably available, and the **documentation is really good**

Code examples

Misc

Using Other C++ Libraries

```
// [[Rcpp::depends(RcppGSL)]]

#include <RcppGSL.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>
```

Calling R Functions in Rcpp

```
RObject callWithOne(Function f) {
  return f(1);
}
> callWithOne(function(x) x + 1)
# [1] 2
```

Resources

- Hadley Wickham's Advanced R Site
- Official Rcpp Website
- Armadillo Documentation