Short introduction to Rcpp

Sherman Ip

University of Warwick

2nd March 2017

Table of Contents

Rcpp

- Rcpp is an R package which allows you to compile and run C++ code in R.
- Eddelbuettel et al. (2011)

C++

Some key words:

C++

Some key words:

- Strongly typed

C++

Some key words:

- Strongly typed
- Memory allocation

C++

Some key words:

- Strongly typed
- Memory allocation
- Pointers

C++

Some key words:

- Strongly typed
- Memory allocation
- Pointers
- Memory leaks

C++

Some key words:

- Strongly typed
- Memory allocation
- Pointers
- Memory leaks
- Object oriented

Simple example

hello_world.cpp

```cpp
#include <iostream>

int main(){
    std::cout << "Hello World!" << std::endl;
}
```

Simple example

hello_world.cpp

```cpp
#include <iostream>

// [[Rcpp::export]]
void hello_world(){
    std::cout << "Hello World from C++!" << std::endl;
}
```

Simple example

```
> library(Rcpp);
> sourceCpp('hello_world.cpp');
> hello_world();
Hello World from C++!
>
```

Inputs

hello.cpp

```cpp
#include <iostream>

// [[Rcpp::export]]
void hello(std::string name){
    std::cout << "Hello " + name << std::endl;
}
```

Inputs

```
> library(Rcpp);
> sourceCpp('hello.cpp');
> hello('Bob');
Hello Bob
>
```

Inputs and outputs

input_output.cpp

```cpp
// [[Rcpp::export]]
int input_output(int n){

    int x_before = 0;
    int x = 1;
    int x_new;

    if (n <= 0){
        return 0;
    }

    else{
        for (int i=0; i<(n-1); i++){
            x_new = x + x_before;
            x_before = x;
            x = x_new;
        }
        return x;
    }

}
```

Inputs and outputs

```
> library(Rcpp);
> sourceCpp('input_output.cpp');
> x = rep(0,10);
> for (i in 1:10){
+   x[i] = input_output(i);
+ }
> print(x);
 [1]  1  1  2  3  5  8 13 21 34 55
> 
```

Matrices

- `armadillo` is a C++ library for matrix operations. Sanderson et al. (2016)

Matrices

- `armadillo` is a `C++` library for matrix operations. Sanderson et al. (2016)
- `RcppArmadillo` is an `R` package wich allows you to compile and run `C++` code, which uses `arma`, in `R`. Eddelbuettel et al. (2014)

Matrices

transpose.cpp

```cpp
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>
using namespace arma;
using namespace Rcpp;


void doTranspose(double* output, double* input, int nrow, int ncol){

    memcpy(output, input, nrow*ncol*sizeof(double));

    Mat<double> X_t (output, nrow, ncol, false);

    inplace_trans(X_t);
}


// [[Rcpp::export]]
NumericMatrix transpose(NumericMatrix X){

    NumericMatrix X_t (X.ncol(),X.nrow());

    doTranspose(X_t.begin(), X.begin(), X.nrow(), X.ncol());

    return X_t;
}
```

Matrices

```
> library(Rcpp);
> library(RcppArmadillo);
> sourceCpp('transpose.cpp');
> X = matrix(1:12,4,3);
> print(X);
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> X_t = transpose(X);
> print(X_t);
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
>
```

Lists

matrix_list.cpp

```cpp
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>
using namespace arma;
using namespace Rcpp;


// [[Rcpp::export]]
List matrix_list(int n){

    List list;
    for (int i=0; i<n; i++){
        list.push_back(NumericMatrix(i+1,i+1));
    }

    return list;
}
```

Lists

```
> library(Rcpp);
> library(RcppArmadillo);
> setwd("~/Documents/rug_rcpp/lists")
> sourceCpp('matrix_list.cpp');
> matrix_list(3);
[[1]]
     [,1]
[1,]    0

[[2]]
     [,1] [,2]
[1,]    0    0
[2,]    0    0

[[3]]
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0

>
```

Some advice

- C++11 and armadillo libraries are very useful and well documented.

Some advice

- C++11 and armadillo libraries are very useful and well documented.
- Spilt your programe into pure C++ code and code involving Rcpp.

Some advice

- C++11 and armadillo libraries are very useful and well documented.
- Spilt your programe into pure C++ code and code involving Rcpp.
- Learn C++ first before Rcpp.