Rcpp

Oliver Heidmann
Supervised by: Julian Kunkel

University of Hamburg

oliverheidmann@hotmail.de

June 15, 2016

Overview

What is C++?

- object orientated programming languge
- intermediate level language
  - high level language features
    and also
  - low level language features
- designed to be
  - fast at executing
  - efficient with memory
  - flexible in the way it can be used

What is Rccp?

- easy integration of c++ code
- mappings from r objects to c classes
- package skeleton creation
- flexible error and exception handling

RObject and SEXP

- Rcpp::RObject is a very thin wrapper around an SEXP.
- Rcpp::RObject defines set of functions applicable to any r object
- SEXP only member of Rcpp::RObject.
- SEXP represents r object
- SEXP is guarded from Garbage collection through Rcpp::RObject

Rcpp basics: Type Mappings

What is Wrappable?

- int double bool to r atomic vectors
- std::string to R atomic character vectors
- stl containers
- and any class that has a SEXP() operator for conversion
- or any class in which wrap() tamplate is specialized

wrap() for converting c++ types to R

```
template <typename T> SEXP wrap(const T &  object)
```

Rcpp::as for converting R types to c++ types

```
template <typename T> T as(SEXP x)
```

Calling a c++ function in r

- calling done with .call(...)
- calls the c++ function in r
- overloading not supported
- parameters must be convertible
- return type has to be convertible or void

```
.call(
    "function_name", parameter_1, ...,
    parameter_n, package="packagename"
)
```

Rcpp.package.skeleton

- r command to create skeleton rcpp package
- already in Rcpp package
- can be created with example functions
- very easy to get into
- rough guide in created package
- c++ functons get compiled when package is built

Rcpp and inline

- requires inline package
- compiles at each execution

Rcpp cppSource

- simple way of including c++ functions
- function needs to be marked with [[Rcpp::export]]
- handles wrapping automatically
- slightly faster than inline functions
- compiles at each execution
- less code to be written
- very convenient

cpp Export function

```
RcppExport SEXP test_add_lists(SEXP r_list1, SEXP r_list2)
{
    std::vector<int> cpp_vector1;
    std::vector<int> cpp_vector2;

    BEGIN_RCPP
    Rcpp::RObject __result;
    Rcpp::RNGScope __rngScope;

    cpp_vector1 = Rcpp::as<std::vector<int>>(r_list1);
    cpp_vector2 = Rcpp::as<std::vector<int>>(r_list2);

    __result = add_lists(cpp_vector1, cpp_vector2);

    return Rcpp::wrap(__result);
    END_RCPP
}
```

c++ function

```
std::vector<int> add_lists(std::vector<int> vec1,
                           std::vector<int> vec2) {

    std::vector<int> result;
    unsigned long max_length;

    max_length = std::min(vec1.size(), vec2.size());

    for (unsigned long i = 0; i < max_length; i++)
    {
        result.push_back(vec1[i] + vec2[i]);
    }
    return result;}
```

calling the function

```
 add_lists <-function(vec1, vec2) {
    .Call( "test_add_lists", vec1, vec2, PACKAGE = 'test')
}
```

package inline

```
requires(inline)
add <- cppFunction("
    double add(double x, double y)
    {
    double sum = x + y;
    return sum;
    }
")

add(-0.01,1.02)
```

output: 1.01

sourceCpp

simple-c-functions.cpp

```
#include <Rcpp.h>
#include <iostream>

// [[Rcpp::export]]
double myCmean(Rcpp::NumericVector x)
{

    double result = 0;
    for(auto v : x)
    {
        result += v;
    }
    return result/x.length();
}
```

sourceCpp

main.R

```
require(Rcpp)
sourceCpp("simple-c-functions.cpp")

myCmean(10.01,1.1,2.26)
```

output:
13.37

Runtime comparison

x = vector with 10000 entrys of type numeric/double
each function was executed 1000 times

unit: microseconds

function	min	median	max
r inlineMean(x)	12.054	14.5440	49.531
r mean(x)	2062.821	2151.4090	3021.454
c mean(x)	9.179	9.8325	21.462

Conclusion

- Rcpp is easy to get into
- package creation is fast and easy
- sourceCpp() easiest way of using small cpp functions
- c++ code is a lot faster

sources

- http://dirk.eddelbuettel.com/code/rcpp/Rcpp-introduction.pdf
  13.6.2016
- http://dirk.eddelbuettel.com/code/rcpp/Rcpp-package.pdf
  13.6.2016
- http://dirk.eddelbuettel.com/code/rcpp/Rcpp-FAQ.pdf
  13.6.2016
- https::/www.techopedia.com/definition/26184/c-programming-language
  13.6.2016
- adv-r.had.co.nz/Rcpp.html
  13.6.2016
- https://cran.r-project.org/web/packages/microbenchmark/microbenchmark.pdf
  13.6.2016