Rcpp

Oliver Heidmann
Supervised by: Julian Kunkel

University of Hamburg

oliverheidmann@hotmail.de

June 14, 2016
Overview

1. What is C++?
2. What is Rcpp?
3. Rcpp basics
   - RObject and SEXP
     - Rcpp basics: Type Mappings
     - conversion methods
     - Calling a function
4. Writing R code
   - Rcpp.package.skeleton
5. Example parts
   - cpp Export function
   - c++ function
   - calling the function
   - package inline
   - sourceCpp
   - RuntimeComparison
6. Conclusion
7. sources
What is C++?

- object orientated programming language
- 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() template is specialized
conversion methods

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

\[
\text{template <typename } T \text{> SEXP wrap(const } T & \text{ object)}
\]

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

\[
\text{template <typename } T \text{> } T \text{ as(SEXP } x)\]
Calling done with `.call(...)`

- r function call
- calls the c function for r

```r
.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
rcpp skeleton package guide

rough guide in package included

▶ edit the help file skeletons in 'man', possibly combining help files for multiple functions.
▶ edit the exports in 'NAMESPACE', and add necessary imports.
▶ put any c/c++/fortran code in 'src'.
▶ R CMD build to build the package tarball.
▶ R CMD check to check the package tarball.
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

```cpp
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')
}

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

add(-0.01,1.02)

output: 1.01
#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

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

myCmean(10.01, 1.1, 2.26)
```

output:
13.37
Runtime comparison

$x = \text{vector with 10000 entries of type numeric/double}$
each function was executed 1000 times

unit: microseconds

<table>
<thead>
<tr>
<th>function</th>
<th>min</th>
<th>median</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>r inlineMean(x)</td>
<td>12.054</td>
<td>14.5440</td>
<td>49.531</td>
</tr>
<tr>
<td>r mean(x)</td>
<td>2062.821</td>
<td>2151.4090</td>
<td>3021.454</td>
</tr>
<tr>
<td>c mean(x)</td>
<td>9.179</td>
<td>9.8325</td>
<td>21.462</td>
</tr>
</tbody>
</table>
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

- adv-r.had.co.nz/Rcpp.html 13.6.2016