Writing code with Rcpp and RcppEigen

Stephen Berg

University of Wisconsin-Madison

12/7/2018
Today, hope to answer the following:

- Why/when should I use C++ rather than R?
- How should I integrate R and C++?

Some examples and possibly useful tips.
Why

R programs tend to be

- Fast and easy to write
- Easy to test and debug
- Convenient for reading in, cleaning, analyzing data

but can also be (when I write them)

- disorganized—programs may not have clearly delineated parts, copy and pasted code, object oriented programming forgotten
- slow, unless programmed in careful and sometimes unnatural ways ("vectorization")

Worse, R may still be slow no matter how carefully it is programmed
An alternative

C++ is a modern, widely used programming language

Pros

▶ “object-oriented”—may be easier to organize complicated programs through user-defined classes
▶ can generally write faster code than with R

Cons

▶ slower and harder to write than R
▶ Harder to test and debug
When does it matter?

R already calls fast functions for matrix multiplication, matrix decompositions, etc.
So basic linear algebra like

\[ A\%*\%B \]

may not see any speed increase in C++.
On the other hand, iteration that cannot be vectorized can become much faster in C++/another compiled language.
Example: Gibbs sampling

The Gibbs sampler is a Markov chain Monte Carlo (MCMC) sampling scheme. It involves lots of iterative updates on components of a vector. The iterations depend on each other and need to be done sequentially-hard (impossible?) to vectorize.

Timings from http://dirk.eddelbuettel.com/blog/2011/07/14/ for a bivariate Gibbs sampling example:

<table>
<thead>
<tr>
<th>test</th>
<th>replications</th>
<th>elapsed</th>
<th>relative</th>
<th>user.self</th>
<th>sys.self</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSLGibbs(N, thn)</td>
<td>10</td>
<td>7.845</td>
<td>1.000000</td>
<td>7.84</td>
<td>0</td>
</tr>
<tr>
<td>RcppGibbs(N, thn)</td>
<td>10</td>
<td>12.218</td>
<td>1.557425</td>
<td>12.22</td>
<td>0</td>
</tr>
<tr>
<td>RCgibbs(N, thn)</td>
<td>10</td>
<td>312.296</td>
<td>39.808286</td>
<td>311.98</td>
<td>0</td>
</tr>
<tr>
<td>Rgibbs(N, thn)</td>
<td>10</td>
<td>420.953</td>
<td>53.658764</td>
<td>420.59</td>
<td>0</td>
</tr>
</tbody>
</table>
Rcpp and RcppEigen

- **Rcpp**: package that lets user pass data back and forth between R and C++ programs
- **RcppEigen**: provides access to Eigen C++ template library

```r
# to install
install.packages("Rcpp")
install.packages("RcppEigen")
# may need to install Rtools as well
```
Getting started: the sourceCpp() function

In the text file "firstProgram.cpp", type:

```cpp
#include <Rcpp.h>

// [[Rcpp::export]]
void helloWorld()
{
    Rcpp::Rcout << "Hello world" << std::endl;
}
```

In the R or RStudio console, type:

```r
sourceCpp("C:/.../firstProgram.cpp")
helloWorld()
```
Another speed comparison

In R

```r
addUp_R <- function(n) {
  val = 0
  for (i in 1:n) {
    val = val + i
  }
  return(val)
}
```
#include <Rcpp.h>

// [[Rcpp::export]]
int addUp_Cpp(int n){
    int start=0;
    for (int i=1;i<=n;i++){
        start=start+i;
    }
    return start;
}
Running

\[ \text{microbenchmark(\text{addUp}_R(10000))} \]
\[ \text{microbenchmark(\text{addUp}_\text{Cpp}(10000))} \]

on my computer gave a mean evaluation time of 307 microseconds for the R version and 3.16 microseconds for the C++ version, respectively.
What is Eigen?

- C++ template library for linear algebra
- includes useful functionality we’d rather not program ourselves
- efficient, actively maintained
- included with RcppEigen download

What does it do?

- implements matrices
- matrix multiplication, matrix addition
- matrix decompositions, elementwise operations on matrices
- gives us access to many linear algebra functions we are used to in R
Example: compute quadratic form

In the file "eigenExample.cpp", type

```cpp
#include <RcppEigen.h>
/**/[Rcpp::depends(RcppEigen)]

// a function to compute x^tAy for vectors x,y
// and a matrix A
/**/[Rcpp::export]
double quadraticForm(Eigen::VectorXd vec1,
                     Eigen::MatrixXd A,
                     Eigen::VectorXd vec2){
    return vec1.adjoint()*A*vec2;
}
```
Running `quadraticForm()`

In the R console, type

```r
x=c(1,2)
y=c(3,4)
A=matrix(c(5,6,7,8),ncol=2)
sourceCpp("eigenExample.cpp")
quadraticForm(x,A,y)
```

# for comparison:
`t(x)%*%A%*%y`
Quadratic form speed comparison

Suppose $x$ and $y$ are length 1000 vectors, and $A$ is a $1000 \times 1000$ matrix. Which should be faster:

$$t(x)^\%*%A^\%*%y$$

or

$$\text{quadraticForm}(x,A,y)$$

?
I found averages of about 2 and 5 ms for the R and C++ functions, respectively. So the R code is not actually slower than the C++ code in this instance.
Building a package

For larger projects, it may be useful to build a package to keep things organized. In the R console, the command

```r
RcppEigen.package.skeleton("packageName")
```

will build a basic package with some example functions that can be deleted or modified.

Making the directory an R project in RStudio lets you build and compile the whole package at once.
In practice

While RStudio does do code completion and help with compilation for C++, I recommend (from experience) developing in a dedicated IDE like Microsoft Visual Studio (for Windows) or XCode (for Mac OS)
Example: the function "fail.cpp"

```cpp
#include <RcppEigen.h>
//@[[Rcpp::depends(RcppEigen)]]
//@[[Rcpp::export]]
Eigen::VectorXd memory(int n){
  Eigen::VectorXd x;
  for (int i=0; i<n;i++){
    x(i)=1;
  }
  return(x);
}
```
Example continued

Running

```
sourceCpp("fail.cpp")
memory(100)
```

in the R console causes the R session to abort
The issue is much easier to debug in Visual Studio.

```cpp
#include <RcppEigen.h>
//[[Rcpp::depends(RcppEigen)]]

//[[Rcpp::export]]
Eigen::VectorXd memory(int n){
    Eigen::VectorXd x;
    x.setZero(n); // problem: needed to
                // reserve space for x

    for (int i=0; i<n;i++){
        x(i)=1;
    }
    return(x);
}
```
Thank you
References

- http://dirk.eddelbuettel.com/blog/2011/07/14/, post by Dirk Eddelbuettel, source for Gibbs sampling example and timings