Writing code with Rcpp and RcppEigen

Stephen Berg

University of Wisconsin-Madison

12/7/2018

Introduction

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:

```
    test replications elapsed  relative user.self sys.self
4  GSLGibbs(N, thn)           10   7.845  1.000000
3 RcppGibbs(N, thn)           10  12.218  1.557425
2   RCgibbs(N, thn)           10 312.296 39.808286    3
1    Rgibbs(N, thn)           10 420.953 53.658764    4
```

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

```
#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:

```
#include <Rcpp.h>

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

In the R or RStudio console, type:

```
sourceCpp("C:/.../firstProgram.cpp")
helloWorld()
```

Another speed comparison

In R

```
addUp_R<-function(n){
    val=0
    for (i in 1:n){
        val=val+i
    }
    return(val)
}
```

Speed comparison continued

In C++

```
#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;
}
```

Speed comparison continued

Running

```
microbenchmark(addUp_R(10000))
microbenchmark(addUp_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

Eigen

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

```
#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

```
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 1000×1000 matrix. Which should be faster:

```
t(x)%*%A%*%y
```

or

```
quadraticForm(x,A,y)
```

?

Quadratic form speed comparison

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

```
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"

```
#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

Solution

The issue is much easier to debug in Visual Studio.

```
#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

- Douglas Bates, Dirk Eddelbuettel (2013). Fast and Elegant Numerical Linear Algebra Using the RcppEigen Package. Journal of Statistical Software, 52(5), 1-24. URL http://www.jstatsoft.org/v52/i05/.
- http://dirk.eddelbuettel.com/blog/2011/07/14/, post by Dirk Eddelbuettel, source for Gibbs sampling example and timings
- Dirk Eddelbuettel and Romain Francois (2011). Rcpp: Seamless R and C++ Integration. Journal of Statistical Software, 40(8), 1-18. URL http://www.jstatsoft.org/v40/i08/.
- Eddelbuettel, Dirk (2013) Seamless R and C++ Integration with Rcpp. Springer, New York. ISBN 978-1-4614-6867-7.
- Dirk Eddelbuettel and James Joseph Balamuta (2017). Extending R with C++: A Brief Introduction to Rcpp. PeerJ Preprints 5:e3188v1. URL https://doi.org/10.7287/peerj.preprints.3188v1.