

An Introduction to Rcpp

Dr. Dirk Eddelbuettel

`dirk.eddelbuettel@R-Project.org`

`edd@debian.org`

`@eddelbuettel`

Bay Area userR Group
San Jose, 27 January 2015

Outline

- 1 Motivation
 - C++
 - Vision
 - Features
 - Interface

A tweet about Rcpp from IQSS / Harvard



Research Consulting

@iqssrtc



Follow

Using [#Rcpp](#) to leverage the speed of c++ with the ease and clarity of R. Thanks, [@eddelbuettel](#)

Reply Retweet Favorited More

RETWEET

1

FAVORITE

1



10:29 AM - 19 Mar 2012

Another tweet about Rcpp



Peter Hickey

@PeteHaitch



Follow

Love that my reaction almost every time I rewrite R code in Rcpp is "holy shit that's fast" thanks @eddelbuettel & @romain_francois #rstats

Reply Retweeted Favorited More

RETWEETS

6

FAVORITES

8



9:08 PM - 18 Oct 2013

Alternative Title of Talk

Rcpp: The Good Parts

- Ease of use
- R -> C++
- C++ as Glue

Why R?

Interactive

R enables us to

- work interactively
- explore and visualize data
- access, retrieve and/or generate data
- summarize and report into pdf, html, ...

making it a preferred environment for many data analysts.

Why R?

Extensible

R has always been extensible via

C via a bare-bones interface described in
Writing R Extensions

Fortran which is also used internally by R

Java via **rJava** by S Urbanek

C++ but essentially at the bare-bones level of C

So 'in theory' this worked – yet tedious 'in practice'.

Why C++?

- Asking Google [currently] leads to about 42 million hits.
- **Wikipedia:** *C++ is a statically typed, free-form, multi-paradigm, compiled, general-purpose, powerful programming language.*
- C++ is industrial-strength, vendor-independent, widely-used, and *still evolving*.
- In science & research, one of the most frequently-used languages: If there is something you want to use / connect to, it probably has a C/C++ API.
- As a widely used language it also has good tool support (debuggers, profilers, code analysis).

Why C++?

Scott Meyers: *“View C++ as a federation of languages”*

C provides a rich inheritance and interoperability as Unix, Windows, ... are all build on C.

Object-Oriented C++ just to provide endless discussions about exactly what OO is or should be.

Templated C++ which is mighty powerful; template meta programming unequalled in other languages.

The STL which is a specific template library which is powerful but has its own conventions.

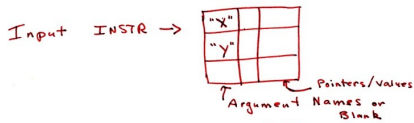
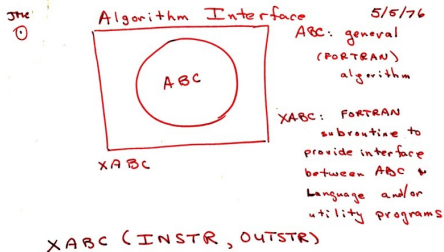
C++11 adds enough to be called a fifth language.

NB: Meyers original list of four language appeared years before C++11.

Why C++?

- Mature yet current
- Strong performance focus:
 - “You don’t pay for what you don’t use”
 - “Leave no room for a language between the machine level and C++”
- Yet also powerfully abstract and high-level
- C++11 and beyond are a big deal giving us new language features
- While there are complexities, Rcpp users are mostly shielded

Interface Vision



Source: John Chambers, personal communication.

Interface Vision

- Use trusted numerical libraries (mostly/exclusively written in Fortran)
- Provide environment which statistician could use more easily
- Enable interactive and iterative data exploration
- Make it extensible for research into statistical methods
- C.f. John Chambers (2008) regarding “Mission” and “Directive”

Interface Vision

R offers us the best of both worlds:

Compiled code with

- Access to proven libraries and algorithms in C/C++/Fortran
- Extremely high performance (in both serial and parallel modes)

Interpreted code with

- An accessible high-level language made for *Programming with Data*
- An interactive workflow for data analysis
- Support for rapid prototyping, research, and experimentation

Why Rcpp?

- Easy to learn** it really does not have to be that complicated – we will look at a few examples
- Easy to use** as it avoids build and OS system complexities thanks to the R infrastructure
- Expressive** it allows for *vectorised* C++ using *Rcpp Sugar*
- Seamless** access to all R objects: vector, matrix, list, S3/S4/RefClass, Environment, Function, ...
- Speed gains** for a variety of tasks **Rcpp** excels precisely where R struggles: loops, function calls, ...
- Extensions** greatly facilitates access to external libraries using eg *Rcpp modules*

Rcpp Playground: Defined by .Call

R side

```
res <- .Call("nameOfFunction", a, b, c,  
             ..., PACKAGE="name")
```

C++ side

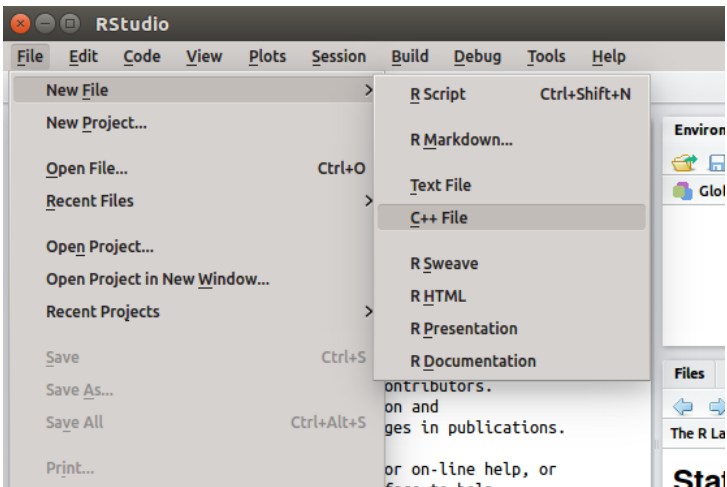
```
extern "C"  
SEXP nameOfFunction(SEXP a, SEXP b,  
                    SEXP c, ...,)
```

Outline

- 2 Rcpp
 - How
 - Example: VAR(1)
 - Packages

How do we use Rcpp?

RStudio makes it very easy: Single File



How do we use Rcpp?

RStudio example cont'ed

The following file used to get created:

```
#include <Rcpp.h>
using namespace Rcpp;

// Below is a simple example of exporting a C++ function to R.
// You can source this function into an R session using the
// Rcpp::sourceCpp function (or via the Source button on the
// editor toolbar)

// For more on using Rcpp click the Help button on the editor
// toolbar

// [[Rcpp::export]]
int timesTwo(int x) {
    return x * 2;
}
```

How do we use Rcpp?

RStudio example cont'ed

The following file gets created now:

```
#include <Rcpp.h>
using namespace Rcpp;

// Below is a simple example of exporting a C++ function to R.
// You can source this function into an R session using the
// Rcpp::sourceCpp function (or via the Source button on the
// editor toolbar)

// For more on using Rcpp click the Help button on the editor toolbar

// [[Rcpp::export]]
NumericVector timesTwo(NumericVector x) {
  return x * 2;
}
```

How do we use Rcpp?

Rcpp Attributes: evalCpp, cppFunction, sourceCpp

```
## evaluate a C++ expression, retrieve result
evalCpp("2 + 2")

## [1] 4

## create ad-hoc R function 'square'
cppFunction('int square(int x) { return x*x;}')
square(7L)

## [1] 49

## or source an entire file (including R code)
#sourceCpp("code/squareWithRCall.cpp")
```

When would we use Rcpp?

Easy speed gain: VAR(1) Simulation

Let's consider a simple possible VAR(1) system of k variables.

For $k = 2$:

$$X_t = X_{t-1}B + E_t$$

where X_t is a row vector of length 2, B is a 2 by 2 matrix and E_t is a row of the error matrix of 2 columns.

When do we use Rcpp?

Easy speedup:: VAR(1) Simulation

In R code, given both the coefficient and error matrices (revealing k and n):

```
rSim <- function(B,E) {  
  n <- nrow(E); k <- ncol(E)  
  X <- matrix(0, n, k)  
  for (r in 2:n) {  
    X[r,] = X[r-1, ] %*% B + E[r, ]  
  }  
  return(X)  
}
```

When do we use Rcpp?

Easy speed gain: VAR(1) Simulation

```
cppFunction ('
arma::mat cppSim(const arma::mat& B,
                 const arma::mat& E) {
  int n = E.n_rows; int k = E.n_cols;
  arma::mat X = arma::zeros<arma::mat>(n,k);
  for (int r=1; r < n; r++) {
    X.row(r) = X.row(r-1) * B + E.row(r);
  }
  return X;
}', depends="RcppArmadillo")
```

When do we use Rcpp?

Easy speed gain: VAR(1) Simulation

```
a <- matrix(c(0.5, 0.1, 0.1, 0.5), nrow=2)
e <- matrix(rnorm(10000), ncol=2)
all.equal(cppSim(a, e), rSim(a, e))

## [1] TRUE

benchmark(cppSim(a, e), rSim(a, e),
           order="relative")[, 1:4]

##           test  replications  elapsed  relative
## 1  cppSim(a, e)           100    0.044    1.000
## 2    rSim(a, e)           100    2.633   59.841
```


How do we use Rcpp?

RStudio makes it very easy (using Rcpp.package.skeleton())

The screenshot shows the RStudio interface with a C++ source file named `foo.cpp` open in the editor. The code includes `<Rcpp.h>` and uses the `Rcpp` namespace. It defines a function `tinesTwo` that takes an integer `x` and returns `x * 2`. A `[[Rcpp::export]]` attribute is placed above the function definition. The `Console` window at the bottom shows the execution of `sourceCpp("files/tinesTwoA.cpp")`, which results in an error: `Error: file not found: 'files/tinesTwoA.cpp'`. The `Environment` pane on the right shows the `Global Environment` with the `tinesTwo` function listed. The `Viewer` pane on the right displays a reference page for Rcpp, including links to [An Introduction to R](#), [The R Language Definition](#), [Writing R Extensions](#), [R Installation and Administration](#), [R Data Import/Export](#), and [R Internals](#).

The **Create R Package** dialog box is open, showing the following options:

- Type:** Package w/ Rcpp
- Package name:** [Empty text field]
- Create package based on source files:** [Empty list box with Add... and Remove buttons]
- Create project as subdirectory of:** [Text field containing "~" and a Browse... button]
- Create a git repository for this project
- Open in new window
- Create Project** and **Cancel** buttons

How do we use Rcpp?

Key Features

As we just saw in the small example:

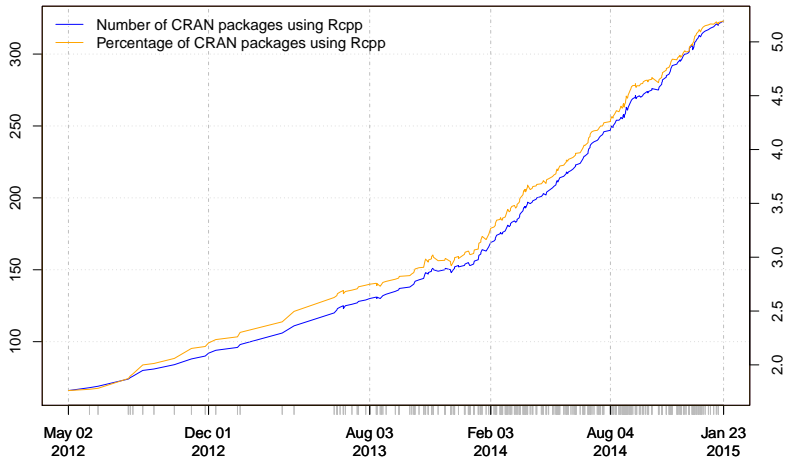
- No build system issues: We just use R
- No operating system dependencies
- No (manual or explicit) memory management
- No complicated C++ code (though we could if we wanted to)
- Easy transition from exploration (“one-liners”) to deployment (“packages”)

Outline

3 Growth

Rcpp on CRAN: Now at 323 packages

Growth of Rcpp usage on CRAN



Outline

4

Doc

- Basics
- Gallery
- Book

What Else?

Basic Documentation

- The package comes with **eight pdf vignettes**, and numerous help pages.
- The introductory vignettes are now **published** (Rcpp and RcppEigen in *J Stat Software*, RcppArmadillo in *Comp. Stat. & Data Anal.*).
- The **rcpp-devel** list is *the* recommended resource, generally very helpful, and fairly low volume.
- **StackOverflow** has over 670 posts too.
- Several blog posts introduce/discuss features.

What Else?

Rcpp Gallery: 90+ working and detailed examples

Rcpp Gallery - Google Chrome

Rcpp Gallery x

gallery.rcpp.org

Rcpp Projects Gallery Book Events More

Featured Articles

- [Quick conversion of a list of lists into a data frame](#) — John Merrill
This post shows one method for creating a data frame quickly
- [Passing user-supplied C++ functions](#) — Dirk Eddebuettel
This example shows how to select user-supplied C++ functions
- [Using Rcpp to access the C API of xts](#) — Dirk Eddebuettel
This post shows how to use the exported API functions of xts
- [Timing normal RNGs](#) — Dirk Eddebuettel
This post compares drawing $N(0,1)$ vectors from R, Boost and C++11
- [A first lambda function with C++11 and Rcpp](#) — Dirk Eddebuettel
This post shows how to play with lambda functions in C++11
- [First steps in using C++11 with Rcpp](#) — Dirk Eddebuettel
This post shows how to experiment with C++11 features
- [Using Rcout for output synchronised with R](#) — Dirk Eddebuettel
This post shows how to use Rcout (and Rcerr) for output
- [Using the Rcpp sugar function clamp](#) — Dirk Eddebuettel
This post illustrates the sugar function clamp
- [Using the Rcpp Timer](#) — Dirk Eddebuettel
This post shows how to use the Timer class in Rcpp
- [Calling R Functions from C++](#) — Dirk Eddebuettel
This post discusses calling R functions from C++

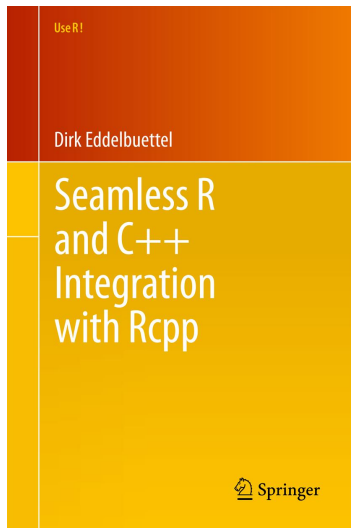
[More »](#)

Recently Published

- Apr 12, 2013 » [Using the RcppArmadillo-based Implementation of R's sample\(\)](#) — Christian Gunning and Jonathan Olmsted
- Apr 8, 2013 » [Dynamic Wrapping and Recursion with Rcpp](#) — Kevin Ushey
- Mar 14, 2013 » [Using bigmemory with Rcpp](#) — Michael Kane
- Mar 12, 2013 » [Generating a multivariate gaussian distribution using RcppArmadillo](#) — Ahmadou Dicko
- Mar 1, 2013 » [Using Rcpp with Boost.Regex for regular expression](#) — Dirk Eddebuettel
- Feb 27, 2013 » [Fast factor generation with Rcpp](#) — Kevin Ushey

What Else?

The Rcpp book



In print since June
2013