An Introduction to Rcpp

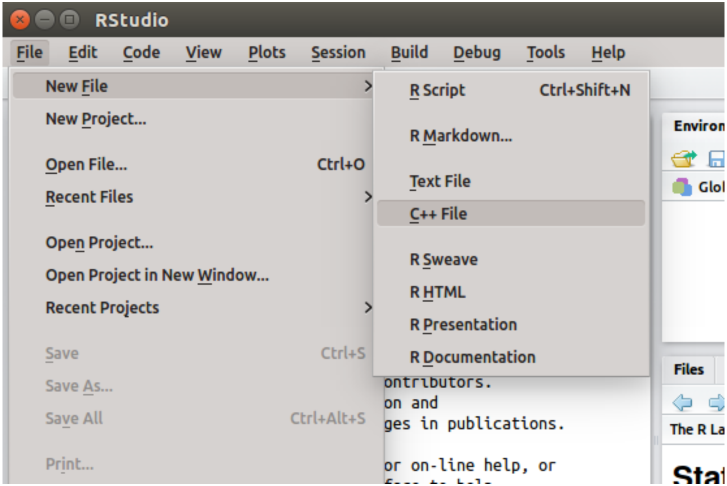
Dr Dirk Eddelbuettel
dirk@eddelbuettel.com
@eddelbuettel

Invited Presentation
Orange County R Users Group
20 May 2014

Outline

A First Example

RStudio makes starting very easy:



```
ontributors.
on and
ges in publications.

or on-line help, or
```

A First Example: Cont'ed

The following file gets created:

```cpp
#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;
}
```

A First Example: Cont'ed

We can easily deploy the file ("press the button") and call the resulting function:

```
Rcpp::sourceCpp('files/timesTwo.cpp')
timesTwo(21)

## [1] 42
```

A First Example: Cont'ed

So what just happened?

- We defined a simple C++ function
- It operates on a single integer argument
- We asked **Rcpp** to 'source it' for us
- Behind the scenes **Rcpp** creates a wrapper
- **Rcpp** then compiles, links, and loads the wrapper
- The function is available in R under its C++ name

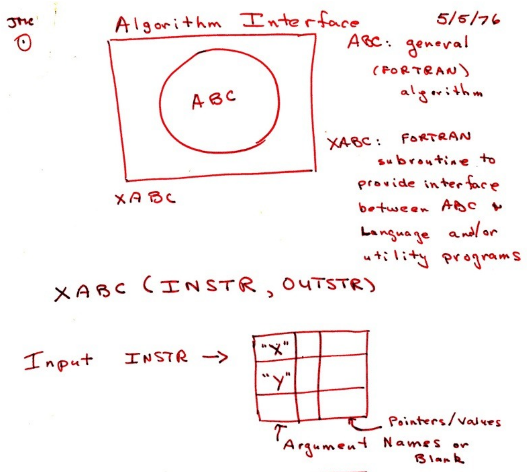
A First Example: Related

Two related functions related to `sourceCpp()`:

```
evalCpp("2 * 2")

## [1] 4

cppFunction("int times2(int x) { return 2*x;}")
times2(123)

## [1] 246
```

Outline

A “vision” from Bell Labs from 1976


JMC
1
Algorithm Interface
5/5/76
ABC
XABC
ABC: general (FORTRAN) algorithm
XABC: FORTRAN subroutine to provide interface between ABC & language and/or utility programs
XABC (INSTR, OUTSTR)
Input INSTR →
"X"
"Y"
Pointers/Values
Argument Names or Blank

Source: John Chambers’ talk at Stanford in October 2010; personal correspondence.

Outline

Rcpp maps R object to C++ objects – both ways

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

// [[Rcpp::export]]
Rcpp::IntegerVector timesTwoI(Rcpp::IntegerVector x) {
   for (int i=0; i<x.size(); i++) {
     x[i] = x[i] * 2;
   }
}

// [[Rcpp::export]]
Rcpp::NumericVector timesTwoN(Rcpp::NumericVector x) {
   for (int i=0; i<x.size(); i++) {
     x[i] = x[i] * 2;
   }
}
```

Rcpp maps R object to C++ objects – both ways

We can also work on the whole object: * operator knows that every vector element needs to be multiplied by two.

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

// [[Rcpp::export]]
Rcpp::IntegerVector timesTwoI(Rcpp::IntegerVector x) {
   return x * 2;
}

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

Object Mapping

Rcpp maps between C++ and standard R types

list
vector
matrix
data.frame
...

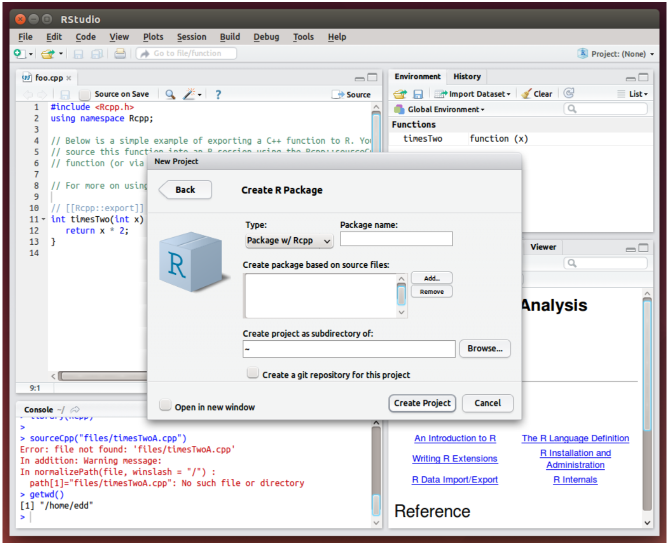
as well as standard C++ variants such as `std::vector<>` — and R types such as `S4 classes`.

Moreover, packages can define their own mapping using `as<>()` and `wrap()`. Popular examples are **RcppArmadillo** and **RcppEigen**.

Outline

Packages and Rcpp

Best way to organize R code with **Rcpp** is via a package:



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

// Below is a simple example of exporting a C++ function to R. Yo
// source this function into an R session using the Rcpp::sourceC
// function (or via

// For more on using
|
// [[Rcpp::export]]
int timesTwo(int x)
   return x * 2;
}
```

Functions

timesTwo   function (x)

New Project

Back   **Create R Package**

Type: Package w/ Rcpp

Package name:

Create package based on source files: Add... Remove

Create project as subdirectory of: ~   Browse...

Create a git repository for this project

Open in new window   Create Project   Cancel

```
> library(Rcpp)
>
> sourceCpp("files/timesTwoA.cpp")
Error: file not found: 'files/timesTwoA.cpp'
In addition: Warning message:
In normalizePath(file, winslash = "/") :
  path[1]="files/timesTwoA.cpp": No such file or directory
> getwd()
[1] "/home/edd"
> 
```

Packages and Rcpp

`Rcpp.package.skeleton()` and its derivatives. e.g. `RcppArmadillo.package.skeleton()` create working packages.

```
// another simple example: outer product of a vector,
// returning a matrix
//
// [[Rcpp::export]]
arma::mat rcpparma_outerproduct(const arma::colvec & x) {
    arma::mat m = x * x.t();
    return m;
}

// and the inner product returns a scalar
//
// [[Rcpp::export]]
double rcpparma_innerproduct(const arma::colvec & x) {
    double v = arma::as_scalar(x.t() * x);
    return v;
}
```

Outline

Well-known packages using Rcpp

Amelia	by G King et al
lme4	by D Bates, M Maechler et al
forecast	by R Hyndman et al
RStan	by A Gelman et al
rugarch	by A Ghalanos
plyr	by H Wickham (plus **roxygen2**, **dplyr**, ...)
httpuv	by J Cheng / RStudio
MTS	by R Tsay

Rcpp is currently used by 214 CRAN packages, and a further 27 BioConductor packages.

Outline

Cumulative Sum

http://gallery.rcpp.org/articles/vector-cumulative-sum/

A basic looped version:

```
#include <Rcpp.h>
#include <numeric>    // for std::partial_sum
using namespace Rcpp;

// [[Rcpp::export]]
NumericVector cumsum1(NumericVector x){
    // initialize an accumulator variable
    double acc = 0;

    // initialize the result vector
    NumericVector res(x.size());

    for(int i = 0; i < x.size(); i++){
         acc += x[i];
         res[i] = acc;
    }
    return res;
}
```

Cumulative Sum

See `http://gallery.rcpp.org/articles/vector-cumulative-sum/`

An STL variant:

```
// [[Rcpp::export]]
NumericVector cumsum2(NumericVector x){
    // initialize the result vector
    NumericVector res(x.size());
    std::partial_sum(x.begin(), x.end(), res.begin());
    return res;
}
```

Cumulative Sum

http://gallery.rcpp.org/articles/vector-cumulative-sum/

Or just **Rcpp** sugar:

```
// [[Rcpp::export]]
NumericVector cumsum_sug(NumericVector x){
    return cumsum(x);  // compute + return result vector
}
```

Of course, all results are the same.

```
cppFunction('NumericVector cumsum2(NumericVector x) {
                                   return cumsum(x); }')
x <- 1:10
all.equal(cumsum(x), cumsum2(x))

## [1] TRUE
```

Calling an R function from C++

http://gallery.rcpp.org/articles/r-function-from-c++/

```
#include <Rcpp.h>

using namespace Rcpp;

// [[Rcpp::export]]
NumericVector callFunction(NumericVector x,
                           Function f) {
    NumericVector res = f(x);
    return res;
}

/*** R
callFunction(x, fivenum)
*/
```

Using Boost via BH

http://gallery.rcpp.org/articles/using-boost-with-bh/

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

// One include file from Boost
#include <boost/date_time/gregorian/gregorian_types.hpp>

using namespace boost::gregorian;

// [[Rcpp::export]]
Rcpp::Date getIMMDate(int mon, int year) {
    // compute third Wednesday of given month / year
    date d = nth_day_of_the_week_in_month(
                       nth_day_of_the_week_in_month::third,
                       Wednesday, mon).get_date(year);
    date::ymd_type ymd = d.year_month_day();
    return Rcpp::wrap(Rcpp::Date(ymd.year, ymd.month, ymd.day));
}
```

Using Boost via BH

http://gallery.rcpp.org/articles/boost-foreach/

```
#include <Rcpp.h>
#include <boost/foreach.hpp>
using namespace Rcpp;
// [[Rcpp::depends(BH)]]

// the C-style upper-case macro name is a bit ugly
#define foreach BOOST_FOREACH

// [[Rcpp::export]]
NumericVector square( NumericVector x ) {

  // elem is a reference to each element in x
  // we can re-assign to these elements as well
  foreach( double& elem, x ) {
    elem = elem*elem;
  }
  return x;
}
```

C++11 now has something similar in a smarter `for` loop.

Vector Subsetting

http://gallery.rcpp.org/articles/subsetting/

New / improved in **Rcpp** 0.11.1:

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

// [[Rcpp::export]]
NumericVector positives(NumericVector x) {
    return x[x > 0];
}

// [[Rcpp::export]]
List first_three(List x) {
    IntegerVector idx = IntegerVector::create(0, 1, 2);
    return x[idx];
}

// [[Rcpp::export]]
List with_names(List x, CharacterVector y) {
    return x[y];
}
```

Armadillo Eigenvalues

http://gallery.rcpp.org/articles/armadillo-eigenvalues/

```cpp
#include <RcppArmadillo.h>

// [[Rcpp::depends(RcppArmadillo)]]

// [[Rcpp::export]]
arma::vec getEigenValues(arma::mat M) {
    return arma::eig_sym(M);
}
```

```r
set.seed(42)
X <- matrix(rnorm(4*4), 4, 4)
Z <- X %*% t(X)
getEigenValues(Z)

# R gets the same results (in reverse)
# and also returns the eigenvectors.
```

Converting C to C++: A plyr example

http://gallery.rcpp.org/articles/plyr-c-to-rcpp/

The job of `split_indices()` *is simple: given a vector* x *of integers, it returns a list where the i-th element of the list is an integer vector containing the positions of x equal to i.*

I will spare you the C API version.

Converting C to C++: A plyr example

http://gallery.rcpp.org/articles/plyr-c-to-rcpp/

```cpp
#include <Rcpp.h>

using namespace Rcpp;

// [[Rcpp::export]]
std::vector<std::vector<int> >
split_indices(IntegerVector x, int n = 0) {
    if (n < 0) stop("n must be a pos. int.");

    std::vector<std::vector<int> > ids(n);

    int nx = x.size();
    for (int i = 0; i < nx; ++i) {
        if (x[i] > n) {
           ids.resize(x[i]);
        }
        ids[x[i] - 1].push_back(i + 1);
    }
    return ids;
}
```

Creating xts objects in C++

http://gallery.rcpp.org/articles/creating-xts-from-c++/

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

NumericVector createXts(int sv, int ev) {
    IntegerVector ind = seq(sv, ev);     // values

    NumericVector dv(ind);               // date(time)s == reals
    dv = dv * 86400;                     // scaled to days
    dv.attr("tzone")    = "UTC";         // index has attributes
    dv.attr("tclass")   = "Date";

    NumericVector xv(ind);               // data has same index
    xv.attr("dim")         = IntegerVector::create(ev-sv+1,1);
    xv.attr("index")       = dv;
    CharacterVector cls = CharacterVector::create("xts","zoo");
    xv.attr("class")       = cls;
    xv.attr(".indexCLASS") = "Date";
    // ... some more attributes ...

    return xv;
}
```

Passing user-defined C(++) functions R to C++

http://gallery.rcpp.org/articles/passing-cpp-function-pointers/

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

// [[Rcpp::export]]
arma::vec fun_cpp(const arma::vec& x) { return(10*x); }

typedef arma::vec (*funcPtr)(const arma::vec& x);

// [[Rcpp::export]]
Rcpp::XPtr<funcPtr> putFunPtrInXPtr() {
    return(Rcpp::XPtr<funcPtr>(new funcPtr(&fun_cpp)));
}

// [[Rcpp::export]]
arma::vec callViaXPtr(const arma::vec x, SEXP xpsexp) {
    Rcpp::XPtr<funcPtr> xpfun(xpsexp);
    funcPtr fun = *xpfun;
    arma::vec y = fun(x);
    return(y);

}
```

Passing user-defined C(++) functions R to C++

http://gallery.rcpp.org/articles/passing-cpp-function-pointers/

Quick illustration:

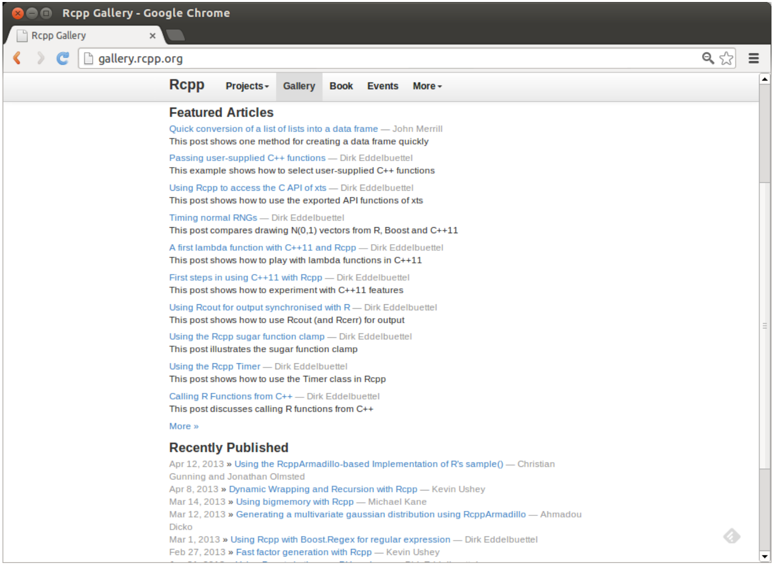
```
fun <- putFunPtrInXPtr()
callViaXPtr(1:4, fun)

##      [,1]
## [1,]   10
## [2,]   20
## [3,]   30
## [4,]   40
```

Outline

Documentation

- The **Rcpp** 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** is closing in 500 **Rcpp** posts.
- And a number of **blog posts** introduce/discuss features.
- Plus . . .

Rcpp Gallery



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 Eddelbuettel
This example shows how to select user-supplied C++ functions

Using Rcpp to access the C API of xts — Dirk Eddelbuettel
This post shows how to use the exported API functions of xts

Timing normal RNGs — Dirk Eddelbuettel
This post compares drawing N(0,1) vectors from R, Boost and C++11

A first lambda function with C++11 and Rcpp — Dirk Eddelbuettel
This post shows how to play with lambda functions in C++11

First steps in using C++11 with Rcpp — Dirk Eddelbuettel
This post shows how to experiment with C++11 features

Using Rcout for output synchronised with R — Dirk Eddelbuettel
This post shows how to use Rcout (and Rcerr) for output

Using the Rcpp sugar function clamp — Dirk Eddelbuettel
This post illustrates the sugar function clamp

Using the Rcpp Timer — Dirk Eddelbuettel
This post shows how to use the Timer class in Rcpp

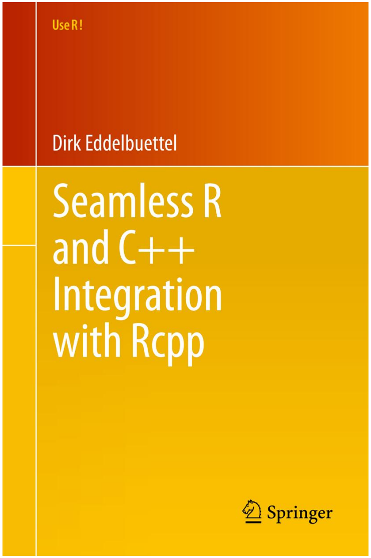
Calling R Functions from C++ — Dirk Eddelbuettel
This post discusses calling R functions from C++

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 Eddelbuettel
Feb 27, 2013 » Fast factor generation with Rcpp — Kevin Ushey

The Rcpp book



Available since June 2013