



# Rcpp

MAKING R APPLICATIONS GO FASTER AND FURTHER

---

Dirk Eddelbuettel

*EARL 2015* Keynote Address

September 16, 2015

Released under a Creative Commons Attribution-ShareAlike 4.0 International License.

# INTRODUCTION

---

# A VERY KIND TWEET







**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

# AND ANOTHER TWEET



**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

# AND YET ANOTHER TWEET



**Pat Schloss**

@PatSchloss



Follow

Thanks to @eddelbuettel's Rcpp and @hadleywickham AdvancedR Rcpp chapter I just sped things up 750x. You both rock.

RETWEETS

3

FAVORITES

5



11:55 AM - 29 May 2015



# AND WHY NOT ANOTHER TWEET



**Rich FitzJohn**

@rgfitzjohn



Follow

Writing some code using [#rstats](#) plain C API and realising/remembering quite how much work Rcpp saves - thanks [@eddelbuettel](#)

RETWEETS

5

FAVORITES

8



5:45 PM - 6 Mar 2015



# AND LAST BUT NOT LEAST



**Romain François**

@romain\_francois



Following

"Rcpp is one of the 3 things that changed how I write #rstats code". @hadleywickham at #EARL2014

RETWEETS

3

FAVORITES

7



3:19 AM - 16 Sep 2014

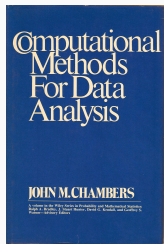


# EXTENDING R

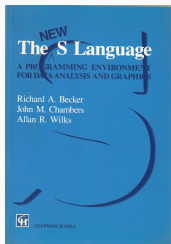
---



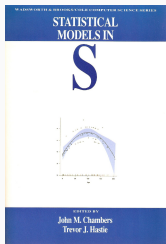
# WHY R? : PROGRAMMING WITH DATA



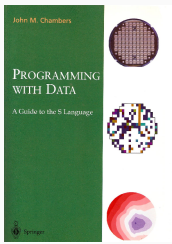
Chambers,  
*Computational  
Methods for Data  
Analysis*. Wiley,  
1977.



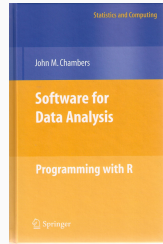
Becker, Chambers,  
and Wilks. *The  
New S Language*.  
Chapman & Hall,  
1988.



Chambers and  
Hastie. *Statistical  
Models in S*.  
Chapman & Hall,  
1992.



Chambers.  
*Programming with  
Data*. Springer,  
1998.



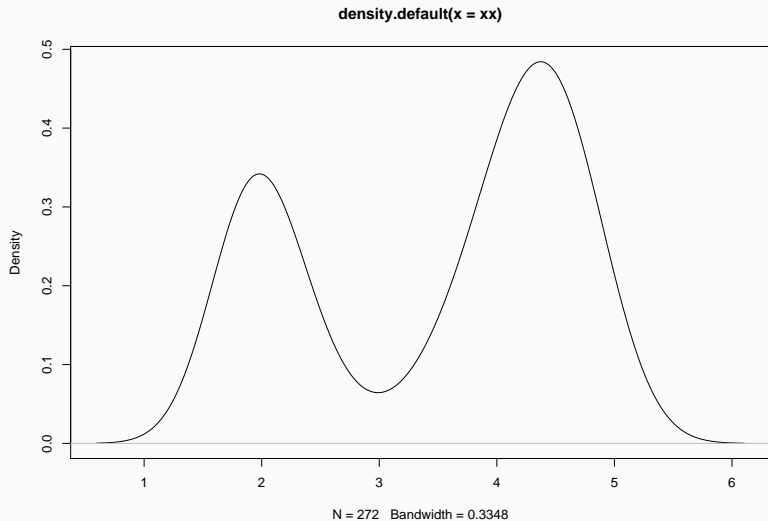
Chambers.  
*Software for Data  
Analysis:  
Programming with  
R*. Springer, 2008

Thanks to John Chambers for sending me high-resolution scans of the covers of his books.

## A SIMPLE EXAMPLE

```
xx <- faithful[, "eruptions"]  
fit <- density(xx)  
plot(fit)
```

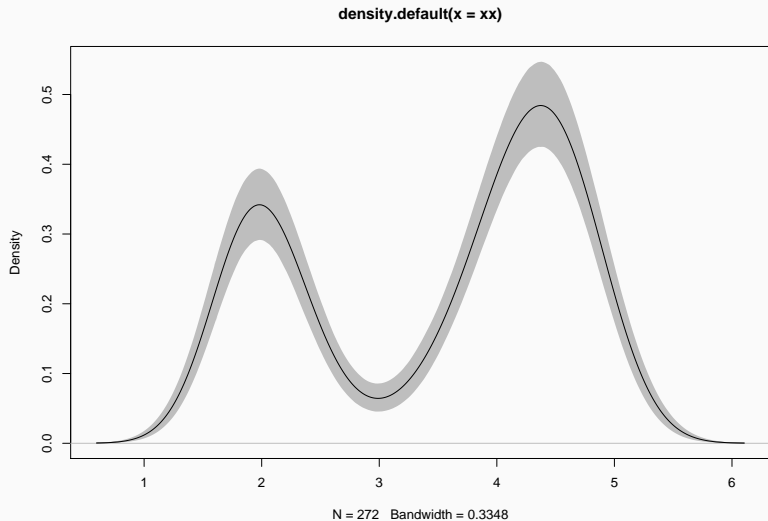
# A SIMPLE EXAMPLE



## A SIMPLE EXAMPLE - REFINED

```
xx <- faithful[, "eruptions"]
fit1 <- density(xx)
fit2 <- replicate(10000, {
  x <- sample(xx, replace=TRUE);
  density(x, from=min(fit1$x), to=max(fit1$x))$y
})
fit3 <- apply(fit2, 1, quantile, c(0.025, 0.975))
plot(fit1, ylim=range(fit3))
polygon(c(fit1$x, rev(fit1$x)), c(fit3[1,], rev(fit3[2,])),
  col='grey', border=F)
lines(fit1)
```

# A SIMPLE EXAMPLE - REFINED



# So WHY R?

R enables us to

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

making it the key language for statistical computing, and a preferred environment for many data analysts.

# So WHY R?

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

So while *in theory* this always worked – it was tedious *in practice*

## WHY EXTEND R?

Chambers (2008), opens Chapter 11 *Interfaces I: Using C and Fortran*:

*Since the core of R is in fact a program written in the C language, it's not surprising that the most direct interface to non-R software is for code written in C, or directly callable from C. All the same, including additional C code is a serious step, with some added dangers and often a substantial amount of programming and debugging required. You should have a good reason.*



# WHY EXTEND R?

Chambers (2008), opens Chapter 11 *Interfaces I: Using C and Fortran*:

*Since the core of R is in fact a program written in the C language, it's not surprising that the most direct interface to non-R software is for code written in C, or directly callable from C. All the same, including additional C code is a serious step, with some added dangers and often a substantial amount of programming and debugging required. You should have a good reason.*

# WHY EXTEND R?

Chambers proceeds with this rough map of the road ahead:

- Against:
  - It's more work
  - Bugs will bite
  - Potential platform dependency
  - Less readable software
- In Favor:
  - New and trusted computations
  - Speed
  - Object references

# WHY EXTEND R?

The *Why?* boils down to:

- **speed**: Often a good enough reason for us ... and a focus for us in this workshop.
- **new things**: We can bind to libraries and tools that would otherwise be unavailable in R
- **references**: Chambers quote from 2008 foreshadowed the work on *Reference Classes* now in R and built upon via Rcpp Modules, Rcpp Classes (and also RcppR6)

## AND WHY C++?

- Asking Google leads to about ~ 50 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++* (maybe 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 Standard Template Library* (STL) is a specific template library which is powerful but has its own conventions.
- *C++11* (and C++14 and beyond) add enough to be called a fifth language.

# WHY C++?

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

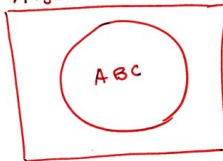
# INTERFACE VISION

---

JTC  
①

## Algorithm Interface

5/5/76



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



# 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** as it really does not have to be that complicated – we will see numerous few examples
- **Easy to use** as it avoids build and OS system complexities thanks to the R infrastrucure
- **Expressive** as 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*

# SPEED

---

## SPEED EXAMPLE (DUE TO STACKOVERFLOW)

Consider a function defined as

$$f(n) \text{ such that } \begin{cases} n & \text{when } n < 2 \\ f(n-1) + f(n-2) & \text{when } n \geq 2 \end{cases}$$

# SPEED EXAMPLE IN R

R implementation and use:

```
f <- function(n) {  
  if (n < 2) return(n)  
  return(f(n-1) + f(n-2))  
}
```

```
## Using it on first 11 arguments  
sapply(0:10, f)
```

```
## [1] 0 1 1 2 3 5 8 13 21 34 55
```

# SPEED EXAMPLE TIMED

Timing:

```
library(rbenchmark)  
benchmark(f(10), f(15), f(20))[,1:4]
```

##	test	replications	elapsed	relative
## 1	f(10)	100	0.026	1.000
## 2	f(15)	100	0.327	12.577
## 3	f(20)	100	3.796	146.000

## SPEED EXAMPLE IN C / C++

A C or C++ solution can be equally simple

```
int g(int n) {  
    if (n < 2) return(n);  
    return(g(n-1) + g(n-2));  
}
```

But how do we call it from R?

# MATT'S EXAMPLE FROM USER! 2015

```
#include <R.h>
#include <Rinternals.h>

int fibonacci_c_impl(int n) {
    if (n < 2) return n;
    return fibonacci_c_impl(n - 1) + fibonacci_c_impl(n - 2);
}

SEXP fibonacci_c(SEXP n) {
    SEXP result = PROTECT(allocVector(INTSXP, 1));
    INTEGER(result)[0] = fibonacci_c_impl(asInteger(n));
    UNPROTECT(1);
    return result;
}

/*
## need to compile, link, load, ...
fibonacci <- function(n) .Call("fibonacci_c", n)
sapply(0:10, fibonacci)
*/
```



# ONE MINOR MODIFICATION TO MATT'S EXAMPLE

```
#include <R.h>
#include <Rinternals.h>

int fibonacci_c_impl(int n) {
    if (n < 2) return n;
    return fibonacci_c_impl(n - 1) + fibonacci_c_impl(n - 2);
}

// [[Rcpp::export]]
SEXP fibonacci_c(SEXP n) {
    SEXP result = PROTECT(allocVector(INTSXP, 1));
    INTEGER(result)[0] = fibonacci_c_impl(asInteger(n));
    UNPROTECT(1);
    return result;
}

/*** R
sapply(0:10, fibonacci_c)
*/
```

## SPEED EXAMPLE IN C / C++

But Rcpp makes this *much* easier:

```
Rcpp::cppFunction("int g(int n) {  
    if (n < 2) return(n);  
    return(g(n-1) + g(n-2)); }")  
sapply(0:10, g)
```

```
## [1] 0 1 1 2 3 5 8 13 21 34 55
```

## SPEED EXAMPLE COMPARING R AND C++

Timing:

```
Rcpp::cppFunction("int g(int n) {  
    if (n < 2) return(n);  
    return(g(n-1) + g(n-2)); }")  
library(rbenchmark)  
benchmark(f(25), g(25), order="relative")[,1:4]
```

##	test	replications	elapsed	relative
## 2	g(25)	100	0.099	1.000
## 1	f(25)	100	47.787	482.697

A nice gain of a few orders of magnitude.

## ANOTHER ANGLE ON SPEED

Run-time performance is just one example.

*Time to code* is another metric.

We feel quite strongly that helps you code more succinctly, leading to fewer bugs and faster development.

A good environment helps. RStudio integrates R and C++ development quite nicely (eg the compiler error message parsing is very helpful) and also helps with package building.

# SPEED EXAMPLE FOOTNOTE ALSO DUE TO MATT

```
#include <Rcpp.h>

// [[Rcpp::plugins("cpp11")]]

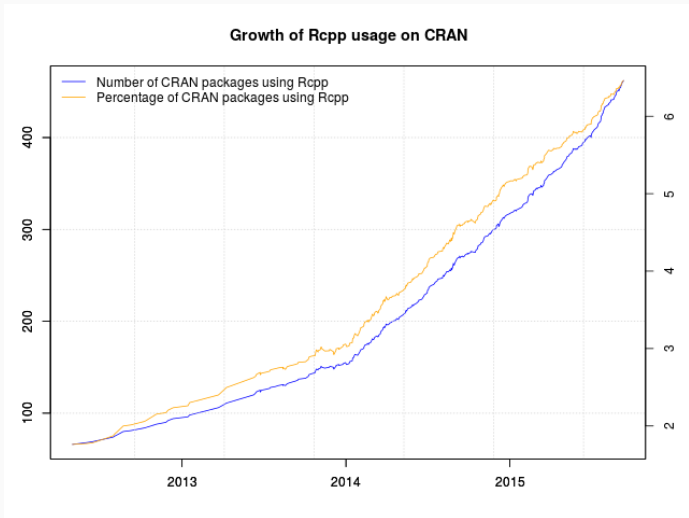
constexpr int fibonacci_recursive_constexpr(const int n) {
    return n < 2 ? n : (fibonacci_recursive_constexpr(n - 1) +
                        fibonacci_recursive_constexpr(n - 2));
}

// [[Rcpp::export]]
int constexprFib() {
    const int N = 42;
    constexpr int result = fibonacci_recursive_constexpr(N);
    return result;
}
```

# POPULARITY

---

# USED BY 462 CRAN PACKAGES AS OF LAST WEEKEND

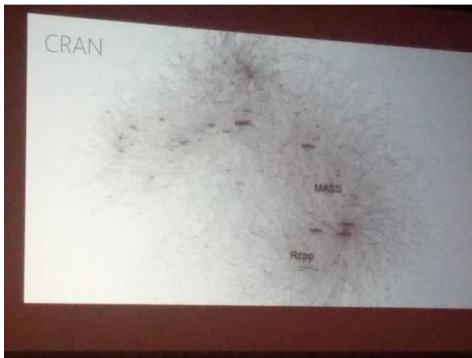


# PAGE RANK ONE (ACCORDING TO ANDRIE DE VRIES)



**Dirk Eddelbuettel**  
@eddelbuettel

Achievement unlocked: @revoandrie says  
#Rcpp has page rank 1 on CRAN!  
#useR2015



RETWEETS  
8

FAVORITES  
38





# APPLICATION SPOTLIGHT: RBLPAPI

---

# HISTORY: BASIC PACKAGE USING THE C (DIRK)

r\_lim\_bloomberg.pdf


1 of 19 61.84%

---


## ***Programming with financial data: Connecting R to Lim and Bloomberg***

Dirk Eddelbuettel  
B of A and Debian  
edd@debian.org

Presentation at the

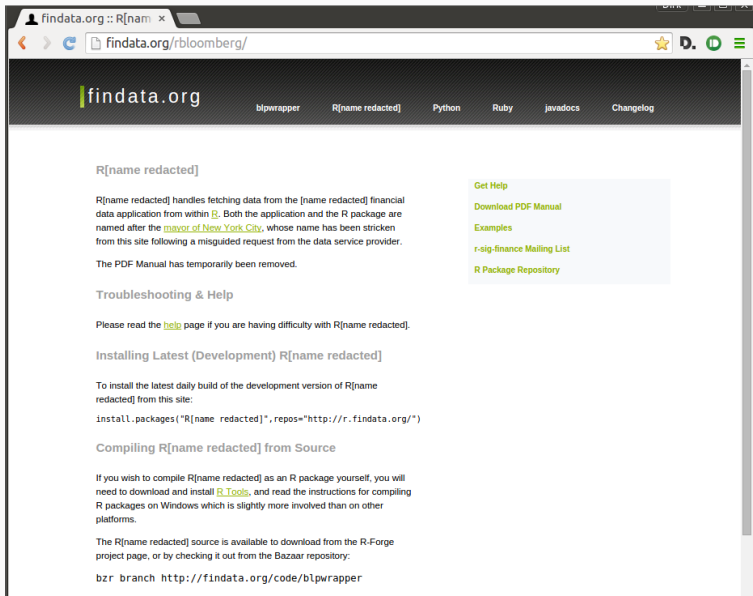


conference, Vienna, May 20-22, 2004



Programming with financial data: Connecting R to Lim and Bloomberg – p. 1

# HISTORY: KEY PACKAGE USING JAVA (ANA, THEN JOHN)



The screenshot shows a web browser window with the address bar displaying `findata.org/rbloomberg/`. The website has a dark header with the `findata.org` logo and navigation links for `blpwrapper`, `R[name redacted]`, `Python`, `Ruby`, `javadocs`, and `Changelog`.

The main content area features the package name `R[name redacted]` in a large font. Below it, a paragraph explains that the package handles fetching data from the `[name redacted]` financial data application. It mentions that both the application and the R package are named after the [mayor of New York City](#), whose name has been stricken from the site following a misguided request from the data service provider.

A note states: "The PDF Manual has temporarily been removed."

A sidebar on the right contains links: [Get Help](#), [Download PDF Manual](#), [Examples](#), [r-sig-finance Mailing List](#), and [R Package Repository](#).

The **Troubleshooting & Help** section begins with the instruction: "Please read the [help](#) page if you are having difficulty with R[name redacted]."

The **Installing Latest (Development) R[name redacted]** section provides instructions on installing the latest daily build of the development version of the package from the site. It includes the following R code snippet:

```
install.packages("R[name redacted]", repos="http://r.findata.org/")
```

The **Compiling R[name redacted] from Source** section explains that if you wish to compile the package as an R package yourself, you will need to download and install [R Tools](#), and read the instructions for compiling R packages on Windows, which is slightly more involved than on other platforms.

It also states that the `R[name redacted]` source is available for download from the R-Forge project page, or by checking it out from the Bazaar repository:

```
bzr branch http://findata.org/code/blpwrapper
```

# HISTORY: BUT THE VENDOR API KEEPS IMPROVING

The screenshot shows a web browser window with the address bar displaying `www.bloomberglabs.com/api/documentation/`. The page header includes the Bloomberg Labs logo and a navigation menu with links for 'ABOUT', 'API LIBRARIES', 'DOCUMENTATION' (which is highlighted), 'PUBLICATIONS', 'FAQ', and 'UPDATES'. The main content area features a large heading 'BLOOMBERG API' followed by a subheading 'Bloomberg's Open Market Data Initiative is part of the company's ongoing effort to foster open solutions for the financial services industry.' Below this, there is a section titled 'DOCUMENTATION' with a brief description of the practical tutorials and reference materials available. To the right of this section is a 'Developer's Guide' section, which is also highlighted. Below the 'Developer's Guide' is an 'API References' section, which includes a list of links for C++ and Java. The C++ section lists 'Version 3.8 (current)' and 'Version 3.7'.

Documentation - Bloomberg Labs

www.bloomberglabs.com/api/documentation/

Bloomberg the Company | Bloomberg Anywhere Login

Bloomberg Labs

SELECT A TOPIC

## BLOOMBERG API

Bloomberg's Open Market Data Initiative is part of the company's ongoing effort to foster open solutions for the financial services industry.

ABOUT API LIBRARIES DOCUMENTATION PUBLICATIONS FAQ UPDATES

### DOCUMENTATION

The practical tutorials and comprehensive reference materials in BLPAPI's technical documentation help developers learn to use all features in the Bloomberg API.

[f](#) [in](#) [g+](#) [tw](#)

### Developer's Guide

The BLPAPI Developer's Guide is a tutorial for developing applications with BLPAPI in C++, Java and C# (.NET). It documents how the SDK libraries connect to the Bloomberg network, data schemas, events and messages, and much more.

Read the [BLPAPI Developer's Guide](#).

### API References

Online API references are available for C++, Java and C# (.NET) users. Follow the links below to find the reference that applies to the BLPAPI version and language you are using in your applications.

### C++

- Version 3.8 (current)
- Version 3.7

# PRESENT .. AND FUTURE (WHIT, DIRK, AND JOHN)

The screenshot shows the GitHub repository page for `armstrtw/Rblpapi`. The repository description is "an api to fetch data from a certain vendor which has a habit of asking for its name to be redacted". It has 108 commits, 1 branch, 0 releases, and 2 contributors. A merge pull request #5 from `eddelbuettel/feature/boost-headers` is highlighted. The file list includes `R`, `inst/include`, `man`, `src`, `.Rbuildignore`, `.gitignore`, `DESCRIPTION`, `NAMESPACE`, `README.rst`, `cleanup`, and `rblpapi.Rproj`. The right sidebar shows options to view code, issues, pull requests, wiki, pulse, and graphs, along with a download ZIP button.

armstrtw / Rblpapi

an api to fetch data from a certain vendor which has a habit of asking for its name to be redacted

108 commits 1 branch 0 releases 2 contributors

branch: master Rblpapi +

Merge pull request #5 from eddelbuettel/feature/boost-headers

armstrtw authored on Dec 16, 2014 latest commit 85451d8ec4

R	also offer choice of return type for getBars(); bump Version	3 months ago
inst/include	update bbg headers to version 3.7.5.1	8 months ago
man	also offer choice of return type for getBars(); bump Version	3 months ago
src	use the BH package as only Boost headers are used	2 months ago
.Rbuildignore	make local exclusion work better	3 months ago
.gitignore	actually make local/ work in .gitignore	3 months ago
DESCRIPTION	use the BH package as only Boost headers are used	2 months ago
NAMESPACE	new accessor functions getBars and getTicks	3 months ago
README.rst	add README.rst	2 years ago
cleanup	add a trivial cleanup script	4 months ago
rblpapi.Rproj	new function fieldSearch	3 months ago

Code

Issues 0

Pull Requests 0

Wiki

Pulse

Graphs

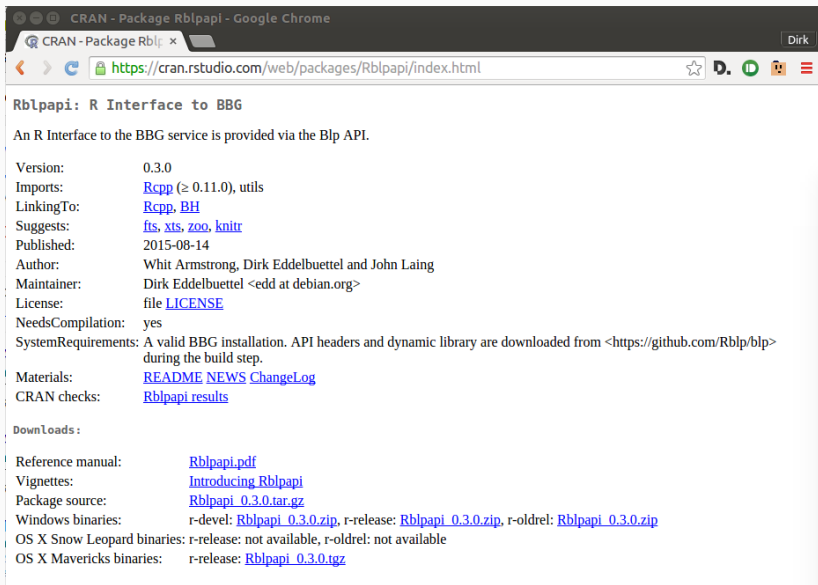
SSH clone URL

git@github.com:arm

You can clone with HTTPS, SSH, or Subversion

Download ZIP

# EVEN BETTER PRESENT AND FUTURE



The screenshot shows a web browser window with the title "CRAN - Package Rblpapi - Google Chrome". The address bar displays the URL "https://cran.rstudio.com/web/packages/Rblpapi/index.html". The page content is for the "Rblpapi: R Interface to BBG" package. It includes a description, version information, imports, linking details, suggestions, publication date, author, maintainer, license, compilation requirements, system requirements, materials, and CRAN check results. There are also links for downloads, including a reference manual, vignettes, and package sources for different operating systems.

**Rblpapi: R Interface to BBG**

An R Interface to the BBG service is provided via the Blp API.

Version: 0.3.0

Imports: [Rcpp](#) (≥ 0.11.0), utils

LinkingTo: [Rcpp](#), [BH](#)

Suggests: [fts](#), [xts](#), [zoo](#), [knitr](#)

Published: 2015-08-14

Author: Whit Armstrong, Dirk Eddelbuettel and John Laing

Maintainer: Dirk Eddelbuettel <edd at debian.org>

License: file [LICENSE](#)

NeedsCompilation: yes

SystemRequirements: A valid BBG installation. API headers and dynamic library are downloaded from <<https://github.com/Rblp/blp>> during the build step.

Materials: [README](#) [NEWS](#) [ChangeLog](#)

CRAN checks: [Rblpapi results](#)

**Downloads:**

Reference manual: [Rblpapi.pdf](#)

Vignettes: [Introducing Rblpapi](#)

Package source: [Rblpapi\\_0.3.0.tar.gz](#)

Windows binaries: r-devel: [Rblpapi\\_0.3.0.zip](#), r-release: [Rblpapi\\_0.3.0.zip](#), r-oldrel: [Rblpapi\\_0.3.0.zip](#)

OS X Snow Leopard binaries: r-release: not available, r-oldrel: not available

OS X Mavericks binaries: r-release: [Rblpapi\\_0.3.0.tgz](#)

# THIRD TIME LUCKY: THE RBLPAPI PACKAGE

The new rewrite is different:

- Lighter – no longer uses or requires Java
- Simpler – leverages Rcpp
- More flexible – easy to add new functionality with C++

# STATUS: THE RBLPAPI PACKAGE

Where we are at now:

- Robust and fast
- Implements most widely-used features
- (Basic) documentation for everything
- Travis CI integration
- On GitHub and in the ghrr repository

*NB: And now on CRAN too, see below.*



## Core Functions known from other API accessors:

- `bdp(c("ESA Index", "SPY US Equity"), c("PX_LAST", "VOLUME"))`
- `bds("GOOG US Equity", "TOP_20_HOLDERS_PUBLIC_FILINGS")`
- `bdh("SPY US Equity", c("PX_LAST", "VOLUME"),  
start.date=Sys.Date()-31)`
- `getBars("ESA Index", startTime=ISOdatetime(2015,1,1,0,0,0))`
- `getTicks("ESA Index", "TRADE", Sys.time()-60*60)`
- `fieldSearch("VWAP")`

## [THEN] CURRENT STATUS OF THE RBLPAPI PACKAGE

### Things we addressed

- Fixed-dimension retrieval very easy
- Now include shared library with `rpath`-encoded path
- Builds “everywhere” including on Travis CI

## [THEN] CURRENT STATUS OF THE RBLPAPI PACKAGE

Things we [then] need[ed] to address:

- DataFrame class caused trouble, need something new
- Builds on “that other OS” very difficult while (vendor) API library built with VC++
- More features: subscriptions, screens, portfolios...
- Pull requests welcome!

## [THEN] SUMMARY: THE RBLPAPI PACKAGE

Concluding:

- Bloomberg provides a first-rate API and infrastructure
- So the R Community came up with good packages
- Language/OS choice matter: some vendors still “different”
- We prefer Open Source; package may not go onto CRAN
- But we have alternatives in GitHub-hosted repositories

*NB: That was then ...*

## [NOW] SUMMARY AND STATE OF THE RBLPAPI PACKAGE

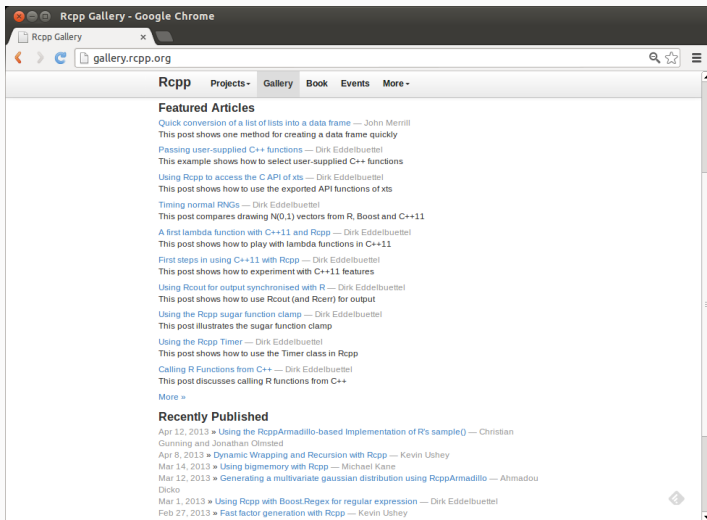
What we learned:

- Powerful APIs are compelling
- Providing working code is key
- Unexpectedly, we got a pull request for Windows support
- With some additional work, this got onto CRAN
- Supporting Linux, OS X and Windows “out of the box”

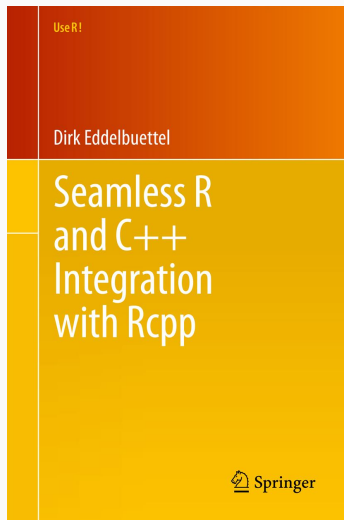
# THE END

---

- The Rcpp package comes with nine pdf vignettes, and numerous help pages.
- The introductory vignettes are now published (for Rcpp and RcppEigen in *J Stat Software*, for RcppArmadillo in *Comp Stat & Data Anlys*)
- The rcpp-devel list is *the* recommended resource, generally very helpful, and fairly low volume.
- StackOverflow has almost 900 posts too.
- And a number of blog posts introduce/discuss features.







# Thank You!

`dirk@eddelbuettel.com`

`http://dirk.eddelbuettel.com/presentations/`

Made using

- TeXlive 20141024
- Beamer with mtheme
- Pandoc 1.12.4.2
- R 3.2.2
- rmarkdown 0.7
- Emacs 24.4
- Ubuntu 15.04