

UseR!

Dirk Eddelbuettel

# Seamless R and C++ Integration with Rcpp

 Springer

# Use R!

*Series Editors:*

Robert Gentleman   Kurt Hornik   Giovanni G. Parmigiani

For further volumes:

<http://www.springer.com/series/6991>



Dirk Eddelbuettel

# Seamless R and C++ Integration with Rcpp

 Springer

Dirk Eddelbuettel  
River Forest  
Illinois, USA

ISBN 978-1-4614-6867-7      ISBN 978-1-4614-6868-4 (eBook)  
DOI 10.1007/978-1-4614-6868-4  
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2013933242

© The Author 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

*To Lisa, Anna and Julia*



# Preface

**Rcpp** is an R add-on package which facilitates extending R with C++ functions.

It is being used for anything from small and quickly constructed add-on functions written either to fluidly experiment with something new or to accelerate computing by replacing an R function with its C++ equivalent to large-scale bindings for existing libraries, or as a building block in entirely new research computing environments.

While still relatively new as a project, **Rcpp** has already become widely deployed among users and developers in the R community. **Rcpp** is now the most popular language extension for the R system and used by over 100 CRAN packages as well as ten BioConductor packages.

This book aims to provide a solid introduction to **Rcpp**.

## Target Audience

This book is for R users who would like to extend R with C++ code. Some familiarity with R is certainly helpful; a number of other books can provide refreshers or specific introductions. C++ knowledge is also helpful, though not strictly required. An appendix provides a very brief introduction for C++ to those familiar only with the R language.

The book should also be helpful to those coming to R with more of a C++ programming background. However, additional background reading may be required to obtain a firmer grounding in R itself. [Chambers \(2008\)](#) is a good introduction to the philosophy behind the R system and a helpful source in order to acquire a deeper understanding.

There may also be some readers who would like to see how **Rcpp** works internally. Covering that aspect, however, requires a fairly substantial C++ content and is not what this book is trying to provide. The focus of this book is clearly on how to use **Rcpp**.



## Historical Context

**Rcpp** first appeared in 2005 as a (fairly small when compared to its current size) contribution by Dominick Samperi to the **RQuantLib** package started by Eddelbuettel in 2002 (Eddelbuettel and Nguyen 2012). **Rcpp** became a CRAN package in its own name in early 2006. Several releases (all provided by Samperi) followed in quick succession under the name **Rcpp**. The package was then renamed to **RcppTemplate**; several more releases followed during 2006 under the new name. However, no new releases were made during 2007, 2008, or most of 2009. Following a few updates in late 2009, the **RcppTemplate** package has since been archived on CRAN for lack of active maintenance.

Given the continued use of the package, Eddelbuettel decided to revitalize it. New releases, using the original name **Rcpp**, started in November 2008. These included an improved build and distribution process, additional documentation, and new functionality—while retaining the existing “classic **Rcpp**” interface. While not described here, this API will continue to be provided and supported via the **RcppClassic** package (Eddelbuettel and François 2012c).

Reflecting evolving C++ coding standards (see Meyers 2005), Eddelbuettel and François started a significant redesign of the code base in 2009. This added numerous new features, many of which are described in the package via different vignettes. This redesigned version of **Rcpp** (Eddelbuettel and François 2012a) has become widely used with over ninety CRAN packages depending on it as of November 2012. It is also the version described in this book.

**Rcpp** continues to be under active development, and extensions are being added. The content described here shall remain valid and supported.

## Related Work

Integration of C++ and R has been addressed by several authors; the earliest published reference is probably Bates and DebRoy (2001). The “Writing R Extensions” manual (R Development Core Team 2012d) has also been mentioning C++ and R integration since around that time. An unpublished paper by Java et al. (2007) expresses several ideas that are close to some of our approaches, though not yet fully fleshed out. The **Rserve** package (Urbanek 2003, 2012) acts as a socket server for R. On the server side, **Rserve** translates R data structures into a binary serialization format and uses TCP/IP for transfer. On the client side, objects are reconstructed as instances of Java or C++ classes that emulate the structure of R objects.

The packages **rcppbind** (Liang 2008), **RAbstraction** (Armstrong 2009a), and **RObjects** (Armstrong 2009b) are all implemented using C++ templates. None of them have matured to the point of a CRAN release. **CXXR** (Runnalls 2009) approaches this topic from the other direction: its aim is to completely refactor R on a stronger C++ foundation. **CXXR** is therefore concerned with all aspects of the R interpreter, read-eval-print loop (REPL), and threading; object interchange be-

tween R and C++ is but one part. A similar approach is discussed by [Temple Lang \(2009a\)](#) who suggests making low-level internals extensible by package developers in order to facilitate extending R. [Temple Lang \(2009b\)](#), using compiler output for references on the code in order to add bindings and wrappers, offers a slightly different angle. Lastly, the **rdyncall** package ([Adler 2012](#)) provides a direct interface from R into C language APIs. This can be of interest if R programmers want to access lower-level programming interfaces directly. However, it does not aim for the same object-level interchange that is possible via C++ interfaces, and which we focus on with **Rcpp**.

## Typographic Convention

The typesetting follows the usage exemplified both by the publisher, and by the *Journal of Statistical Software*. We use

- Sans-serif for programming language such as R or C++
- Boldface for (CRAN or other) software packages such as **Rcpp** or **inline**
- Courier for short segments of code or variables such as `x <- y + z`

We make use of a specific environment for the short pieces of source code interwoven with the main text.

River Forest, IL, USA

Dirk Eddelbuettel



# Acknowledgements

**Rcpp** is the work of many contributors, and a few words of thanks are in order.

Dominick Samperi contributed the original code which, while much more limited in scope than the current **Rcpp**, pointed clearly in the right direction of using C++ templates to convert between R and C++ types.

Romain François has shown impeccable taste in designing and implementing very large parts of **Rcpp** as it is today. The power of the current design owes a lot to his work and boundless energy. Key components such as modules and sugar, as well as lot a of template “magic,” are his contributions. This started as an aside to make object interchange easier for our **RProtoBuf** package—and it has taken us down a completely different, but very exciting road. It has been a pleasure to work with Romain, I remain in awe of his work, and I look forward to many more advances with **Rcpp**.

Doug Bates has been a help from the very beginning: had it not been for some simple macros to pick list components out of **SEXP** types, I may never have started **RQuantLib** a decade ago. Doug later joined this project and has been instrumental in a few key decisions regarding **Rcpp** and **RcppArmadillo** and has taken charge of the **RcppEigen** project.

John Chambers became a key supporter right when *Rcpp modules* started and contributed several important pieces at the gory intersection between R and C++. It was very flattering for Romain and me to hear from John how **Rcpp** is so close to an original design vision of a whole-object interchange between systems which was already present on a hand-drawn Bell Labs designs from the 1970s.

JJ Allaire has become a very important contributor to **Rcpp** and a key supporter of the same idea of an almost natural pairing between R and C++. The *Rcpp attributes* which he contributed are showing a lot of promise, and we expect great things to be built on top of this.

Several other members of the R Core team—notably Kurt Hornik, Uwe Ligges, Martyn Plummer, Brian Ripley, Luke Tierney, and Simon Urbanek—have helped at various points with anything from build issues and portability to finer points of R internals. Last but not least, there would of course be no **Rcpp** if there was no R system to build upon and to extend.

Finally, many members of the **R** and **Rcpp** communities have been very supportive at different workshops, conference presentations, and via the mailing lists. Numerous good questions and suggestions have come this way. And, of course, it is seeing this work being used so actively which motivates us and keeps us moving forward with **Rcpp**.

# Contents

## Part I Introduction

<b>1</b>	<b>A Gentle Introduction to Rcpp</b> .....	3
1.1	Background: From R to C++ .....	3
1.2	A First Example .....	7
1.2.1	Problem Setting .....	7
1.2.2	A First R Solution .....	7
1.2.3	A First C++ Solution .....	8
1.2.4	Using Inline .....	9
1.2.5	Using Rcpp Attributes .....	11
1.2.6	A Second R Solution .....	12
1.2.7	A Second C++ Solution .....	12
1.2.8	A Third R Solution .....	14
1.2.9	A Third C++ Solution .....	14
1.3	A Second Example .....	15
1.3.1	Problem Setting .....	15
1.3.2	R Solution .....	15
1.3.3	C++ Solution .....	16
1.3.4	Comparison .....	17
1.4	Summary .....	18
<b>2</b>	<b>Tools and Setup</b> .....	19
2.1	Overall Setup .....	19
2.2	Compilers .....	20
2.2.1	General Setup .....	20
2.2.2	Platform-Specific Notes .....	21
2.3	The R Application Programming Interface .....	22
2.4	A First Compilation with Rcpp .....	23
2.5	The Inline Package .....	25
2.5.1	Overview .....	25
2.5.2	Using Includes .....	27

2.5.3	Using Plugins	29
2.5.4	Creating Plugins	30
2.6	Rcpp Attributes	31
2.7	Exception Handling	32

## Part II Core Data Types

<b>3</b>	<b>Data Structures: Part One</b>	39
3.1	The RObject Class	39
3.2	The IntegerVector Class	41
3.2.1	A First Example: Returning Perfect Numbers	42
3.2.2	A Second Example: Using Inputs	43
3.2.3	A Third Example: Using Wrong Inputs	44
3.3	The NumericVector Class	45
3.3.1	A First Example: Using Two Inputs	45
3.3.2	A Second Example: Introducing clone	46
3.3.3	A Third Example: Matrices	47
3.4	Other Vector Classes	48
3.4.1	LogicalVector	48
3.4.2	CharacterVector	49
3.4.3	RawVector	49
<b>4</b>	<b>Data Structures: Part Two</b>	51
4.1	The Named Class	51
4.2	The List aka GenericVector Class	52
4.2.1	List to Retrieve Parameters from R	53
4.2.2	List to Return Parameters to R	54
4.3	The DataFrame Class	55
4.4	The Function Class	56
4.4.1	A First Example: Using a Supplied Function	56
4.4.2	A Second Example: Accessing an R Function	56
4.5	The Environment Class	57
4.6	The S4 Class	58
4.7	ReferenceClasses	59
4.8	The R Mathematics Library Functions	60

## Part III Advanced Topics

<b>5</b>	<b>Using Rcpp in Your Package</b>	65
5.1	Introduction	65
5.2	Using Rcpp.package.skeleton	66
5.2.1	Overview	66
5.2.2	R Code	67
5.2.3	C++ Code	68
5.2.4	DESCRIPTION	69
5.2.5	Makevars and Makevars.win	69

- 5.2.6 NAMESPACE ..... 71
- 5.2.7 Help Files ..... 71
- 5.3 Case Study: The **wordcloud** Package ..... 73
- 5.4 Further Examples ..... 74
- 6 Extending Rcpp** ..... 75
  - 6.1 Introduction ..... 75
  - 6.2 Extending Rcpp::wrap ..... 76
    - 6.2.1 Intrusive Extension ..... 76
    - 6.2.2 Nonintrusive Extension ..... 77
    - 6.2.3 Templates and Partial Specialization ..... 78
  - 6.3 Extending Rcpp::as ..... 78
    - 6.3.1 Intrusive Extension ..... 78
    - 6.3.2 Nonintrusive Extension ..... 79
    - 6.3.3 Templates and Partial Specialization ..... 79
  - 6.4 Case Study: The **RcppBDT** Package ..... 80
  - 6.5 Further Examples ..... 82
- 7 Modules** ..... 83
  - 7.1 Motivation ..... 83
    - 7.1.1 Exposing Functions Using **Rcpp** ..... 83
    - 7.1.2 Exposing Classes Using Rcpp ..... 84
  - 7.2 Rcpp Modules ..... 86
    - 7.2.1 Exposing C++ Functions Using Rcpp Modules ..... 86
    - 7.2.2 Exposing C++ Classes Using Rcpp Modules ..... 90
  - 7.3 Using Modules in Other Packages ..... 98
    - 7.3.1 Namespace Import/Export ..... 98
    - 7.3.2 Support for Modules in Skeleton Generator ..... 99
    - 7.3.3 Module Documentation ..... 100
  - 7.4 Case Study: The **RcppCNPY** Package ..... 100
  - 7.5 Further Examples ..... 102
- 8 Sugar** ..... 103
  - 8.1 Motivation ..... 103
  - 8.2 Operators ..... 105
    - 8.2.1 Binary Arithmetic Operators ..... 105
    - 8.2.2 Binary Logical Operators ..... 106
    - 8.2.3 Unary Operators ..... 106
  - 8.3 Functions ..... 107
    - 8.3.1 Functions Producing a Single Logical Result ..... 107
    - 8.3.2 Functions Producing Sugar Expressions ..... 107
    - 8.3.3 Mathematical Functions ..... 113
    - 8.3.4 The d/q/p/q Statistical Functions ..... 114
  - 8.4 Performance ..... 115
  - 8.5 Implementation ..... 116



8.5.1	The Curiously Recurring Template Pattern	117
8.5.2	The VectorBase Class	117
8.5.3	Example: sapply	118
8.6	Case Study: Computing $\pi$ Using <i>Rcpp sugar</i>	122

## Part IV Applications

<b>9</b>	<b>RInside</b>	127
9.1	Motivation	127
9.2	A First Example: Hello, World!	128
9.3	A Second Example: Data Transfer	131
9.4	A Third Example: Evaluating R Expressions	132
9.5	A Fourth Example: Plotting from C++ via R	133
9.6	A Fifth Example: Using RInside Inside MPI	134
9.7	Other Examples	135
<b>10</b>	<b>RcppArmadillo</b>	139
10.1	Overview	139
10.2	Motivation: FastLm	140
10.2.1	Implementation	140
10.2.2	Performance Comparison	142
10.2.3	A Caveat	144
10.3	Case Study: Kalman Filter Using <b>RcppArmadillo</b>	146
10.4	RcppArmadillo and Armadillo Differences	152
<b>11</b>	<b>RcppGSL</b>	155
11.1	Introduction	155
11.2	Motivation: FastLm	156
11.3	Vectors	158
11.3.1	<b>GSL</b> Vectors	158
11.3.2	RcppGSL::vector	159
11.3.3	Mapping	161
11.3.4	Vector Views	161
11.4	Matrices	163
11.4.1	Creating Matrices	163
11.4.2	Implicit Conversion	163
11.4.3	Indexing	163
11.4.4	Methods	164
11.4.5	Matrix Views	164
11.5	Using <b>RcppGSL</b> in Your Package	164
11.5.1	The <code>configure</code> Script	165
11.5.2	The <code>src</code> Directory	166
11.5.3	The <code>R</code> Directory	167
11.6	Using <b>RcppGSL</b> with <b>inline</b>	168
11.7	Case Study: <b>GSL</b> -Based B-Spline Fit Using <b>RcppGSL</b>	169

- 12 RcppEigen** ..... 177
  - 12.1 Introduction ..... 177
  - 12.2 Eigen classes ..... 178
    - 12.2.1 Fixed-Size Vectors and Matrices ..... 178
    - 12.2.2 Dynamic-Size Vectors and Matrices ..... 179
    - 12.2.3 Arrays for Per-Component Operations ..... 180
    - 12.2.4 Mapped Vectors and Matrices and Special Matrices ..... 181
  - 12.3 Case Study: Kalman filter using RcppEigen ..... 182
  - 12.4 Linear Algebra and Matrix Decompositions ..... 183
    - 12.4.1 Basic Solvers ..... 183
    - 12.4.2 Eigenvalues and Eigenvectors ..... 184
    - 12.4.3 Least-Squares Solvers ..... 185
    - 12.4.4 Rank-Revealing Decompositions ..... 185
  - 12.5 Case Study: C++ Factory for Linear Models in **RcppEigen** ..... 186

**Part V Appendix**

- A C++ for R Programmers** ..... 195
  - A.1 Compiled Not Interpreted ..... 195
  - A.2 Statically Typed ..... 197
  - A.3 A Better C ..... 198
  - A.4 Object-Oriented (But Not Like S3 or S4) ..... 200
  - A.5 Generic Programming and the STL ..... 201
  - A.6 Template Programming ..... 203
  - A.7 Further Reading on C++ ..... 204

**References** ..... 207

**Subject Index** ..... 211

**Software Index** ..... 217

**Author Index** ..... 219



# List of Tables

Table 1.1	Run-time performance of the recursive Fibonacci examples . .	10
Table 1.2	Run-time performance of the different VAR simulation implementations . . . . .	17
Table 8.1	Run-time performance of <i>Rcpp sugar</i> compared to <b>R</b> and manually optimized <b>C++</b> . . . . .	116
Table 8.2	Run-time performance of <i>Rcpp sugar</i> compared to <b>R</b> for simulating $\pi$ . . . . .	124
Table 11.1	Correspondence between <b>GSL</b> vector types and templates defined in <b>RcppGSL</b> . . . . .	161
Table 11.2	Correspondence between <b>GSL</b> vector view types and templates defined in <b>RcppGSL</b> . . . . .	162
Table 12.1	Mapping between <b>Eigen</b> matrix and vector types, and corresponding array types . . . . .	181
Table 12.2	lmBenchmark results for the <b>RcppEigen</b> example . . . . .	191



# List of Figures

Figure 1.1	Plotting a density in R .....	5
Figure 1.2	Plotting a density and bootstrapped confidence interval in R ..	6
Figure 1.3	Fibonacci spiral based on first 34 Fibonacci numbers .....	8
Figure 9.1	Combining <b>RInside</b> with the Qt toolkit for a GUI application .....	135
Figure 9.2	Combining <b>RInside</b> with the Wt toolkit for a web application .....	136
Figure 10.1	Object trajectory and Kalman filter estimate .....	149
Figure 11.1	Artificial data and B-spline fit .....	175



# List of Listings

1.1	Plotting a density in R	4
1.2	Plotting a density and bootstrapped confidence interval in R	4
1.3	Fibonacci number in R via recursion	7
1.4	Fibonacci number in C++ via recursion	8
1.5	Fibonacci wrapper in C++	9
1.6	Fibonacci number in C++ via recursion, using inline	9
1.7	Fibonacci number in C++ via recursion, using Rcpp attributes	11
1.8	Fibonacci number in C++ via recursion, via Rcpp attributes and sourceCpp	11
1.9	Fibonacci number in R via memoization	12
1.10	Fibonacci number in C++ via memoization	12
1.11	Fibonacci number in R via iteration	14
1.12	Fibonacci number in C++ via iteration	14
1.13	VAR(1) of order 2 generation in R	16
1.14	VAR(1) of order 2 generation in C++	16
1.15	Comparison of VAR(1) run-time between R and C++	17
2.1	A first manual compilation with Rcpp	23
2.2	A first manual compilation with Rcpp using Rscript	24
2.3	Using the first manual compilation from R	24
2.4	Convolution example using inline	26
2.6	Using inline with include=	27
2.5	Program source from convolution example using inline in verbose mode	28
2.7	A first RcppArmadillo example for inline	29
2.8	Creating a plugin for use with inline	30
2.9	Example of new cppFunction	31
2.10	Example of new cppFunction with plugin	32
2.11	C++ example of throwing and catching an exception	32
2.12	Using C++ example of throwing and catching an exception	33
2.13	C++ example of example from Rcpp-type checks	33
2.14	C++ macros for Rcpp exception handling	34