'cpp11armadillo': An 'R' Package to Use the Armadillo C++ Library

Mauricio Vargas Sepúlveda[1,2*], Jonathan Schneider Malamud[3]

1 Department of Political Science, University of Toronto,
2 Munk School of Global Affairs and Public Policy, University of Toronto,
3 Department of Electrical and Computer Engineering, University of Toronto,

☯These authors contributed equally to this work.

¤Current Address: Dept/Program/Center, Institution Name, City, State, Country

†Deceased

¶Membership list can be found in the Acknowledgments section.

*

Abstract

"This article introduces 'cpp11armadillo', an R package that integrates the highly efficient Armadillo C++ linear algebra library with R through the 'cpp11' interface. Designed to offer significant performance improvements for computationally intensive tasks, 'cpp11armadillo' simplifies the process of integrating C++ code into R. This package is particularly suited for R users requiring efficient matrix operations, especially in cases where vectorization is not possible. Our benchmark demonstrate substantial speed gains over native R functions and Rcpp-based setups."

Introduction

As R continues to grow in popularity for statistical computing and data analysis, it can create bottlenecks when working with large datasets. The need to interface with lower-level languages like C++ to bypass these bottlenecks has led to the development of tools like 'Rcpp' [1], and more recently, 'cpp11' [2].

'cpp11armadillo' is a novel package that builds upon the existing Armadillo C++ library [3] to provide high-level access to efficient linear algebra operations directly from R. The existing 'RcppArmadillo' [4] already offers this, and 'cpp11armadillo' offers an alternative with modern C++ features, where we highlight vendoring, that can improve performance and simplify the its usage.

Features

'cpp11armadillo' simplifies C++ integration by leveraging the 'cpp11' package [2], which is designed to reduce the complexity of including C++ code in R. Its core features include:

- High-Performance Matrix Operations: By using Armadillo, cpp11armadillo allows users to perform matrix multiplications, inversions, and decompositions with minimal overhead. These operations are, for some cases, several times faster than their R counterparts.
- Simplified Syntax: Armadillo offers a user-friendly syntax similar to that of 'MATLAB' [3], making it accessible to R users without requiring an extensive background in C++.
- Modern C++ Support: 'cpp11armadillo' supports C++11 and newer standards, enabling efficient memory management, parallel computations, and advanced templates, all integrated into an R workflow.
- Integration with R: The package allows direct manipulation of R data structures (e.g., vectors, matrices) in C++, providing a seamless workflow between the two languages without requiring manual data conversion (e.g., such as exporting data to CSV files and continously reading them in R to use the C++ output or vice versa).

Background

The R programming language was not designed for high-performance computing, it was created to offer ready-made statistical functions for the end-user, and its interpreted nature can lead to slow execution times for computationally intensive tasks [5, 6]. The same applies to Python and other high-level languages.

Compiled languages like C++ offer significant performance improvements due to their direct access to hardware resources and efficient memory management. The counterpart is that C++ has a steep learning curve and it is less user-friendly than R. C++ goals are different, it is designed to be fast and efficient, and it is not focused on statistical computing but on general-purpose programming [7].

However, bottlenecks in R can be solved by using vectorized operations instead of writing loops, which treats data objects as a whole instead of element-wise [8]. There are cases where vectorization is not possible or challenging to implement [8], and this is where 'cpp11armadillo' comes in and it offers data structures and functions that are not available in R that facilitate the implementation of loops and other operations [6, 3].

Even in spite of these difficulties to work with large datasets or computationally intensive tasks, R is a growing popularity language for statistical computing and data analysis, and its community has created tools to integrate it with SQL databases for memory-efficient data access [9]. There are multiple ways to measure a language popularity, and one of them is the number of R questions in Stack Overflow, which has been increasing over the years as shown in the following plot [10].

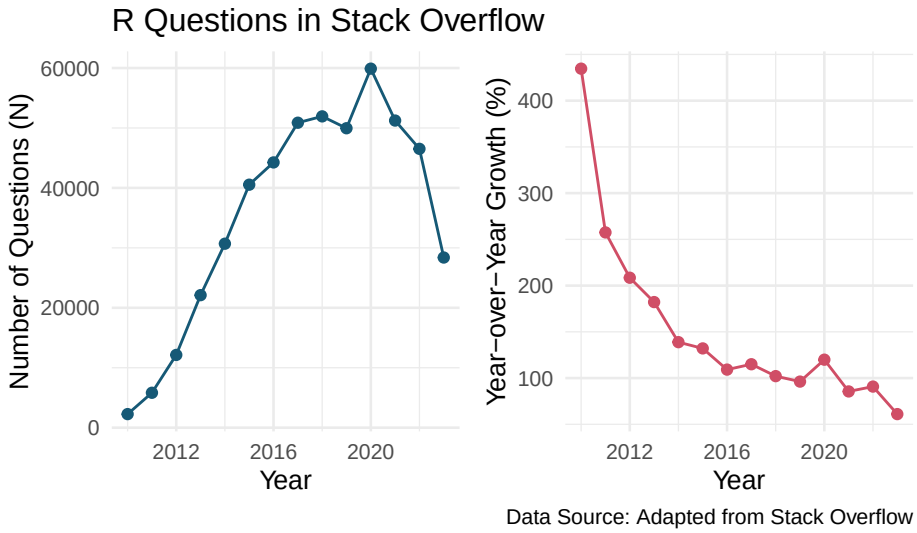

Figure 1. R Questions in Stack Overflow increase from 2,260 in 2010 to 28,385 in 2023 with a peak of 59,895 in 2020.

Modern C++ features

'cpp11' is a modern rewrite of the C++ interface for R, designed to improve safety, performance, and ease of use. It enforces copy-on-write semantics to prevent unintended modifications to data, ensuring that changes to objects do not affect other references. Additionally, 'cpp11' provides safer access to R's C API, reducing runtime errors in C++ code. It also supports ALTREP objects for efficient memory management and deferred computations, making it ideal for handling large datasets. By using UTF-8 strings throughout, 'cpp11' ensures robust handling of datasets created in different countries where encodings vary [2].

Built on C++11 features like smart pointers and lambdas, 'cpp11' offers a more straightforward and efficient implementation compared to previous bindings. Its header-only design eliminates ABI compatibility issues, making it easier to integrate and manage in projects. 'cpp11' also compiles faster, uses less memory, and grows vectors more efficiently, optimizing performance when dealing with large amounts of data. These improvements make 'cpp11' a powerful, streamlined tool for developers who need reliable, high-performance C++ bindings for R [2].

Following from 'cpp11', 'cpp11armadillo' leverages these modern C++ features to provide a seamless interface between R and the Armadillo library, offering high-performance linear algebra operations with minimal overhead. Besides the technical aspects, 'cpp11armadillo' offers vendoring, something not available in 'RcppArmadillo', which can simplify the installation process and reduce dependency issues, especially when working in environments with restricted access to the Internet. For instance, the Niagara Supercomputer that we use at the University of Toronto has restricted access to the Internet, and vendoring can simplify the installation process.

Vendoring is a well-known concept in the Go community, and it consists in copying the dependency code directly into a project's source tree. This approach ensures that dependencies remain fixed and stable, preventing any external changes from

inadvertently breaking the project. While vendoring offers stability, it comes with trade-offs. The primary advantage is that updates to the 'cpp11armadillo' library will not disrupt existing code, and it also copies 'cpp11' C++ headers. However, the drawbacks include an increase in package size and the loss of automatic updates, meaning that bug fixes and new features will only be available when manually updated.

In other words, vendoring allows the package creator to provide a dependency-free package that can be installed in any environment without requiring the end user to install 'cpp11' nor 'cpp11armadillo'. This approach makes 'cpp11armadillo' a dependency for the developer but not for the end user.

Usage

To use 'cpp11armadillo', users must first install the package from CRAN or GitHub. The package includes the Armadillo library, no additional installation is required. The following code shows how to install the package:

```
install.packages("cpp11armadillo")

# or
remotes::install_github("pachadotdev/cpp11armadillo")
```

Once installed, users can use the provided package template function to create a new package that uses C++ code with Armadillo. The package template includes simple examples and all the necessary files to compile the code and install the new R package. The following code shows how to create a new package:

```
cpp11armadillo::create_package("cpp11newpackage")
```

The package vignettes cover the directories organization and the necessary steps to build an R package with relatively low setup efforts.

It is important to note that 'cpp11armadillo' only work within an R package, and it is not possible to compile individual C++ scripts on the fly. This design choice ensures that the R and C++ code is organized following the organization described in **(author?)** [11].

Benchmarks

In order to evaluate the performance of 'cpp11armadillo', we compared it with native R functions and 'RcppArmadillo'. We focused on a single test consisting in computing the Balassa Index [12].

The R code for the Balassa Index computation is as follows:

```
balassa_r <- function(X) {
  B <- t(t(X / rowSums(X)) / (colSums(X) / sum(X)))
  B[B < 1] <- 0
  B[B >= 1] <- 1
  B
}
```

The C++ code for the Balassa Index computation from an R matrix is as follows:

```
Mat<double> balassa_armadillo(const doubles_matrix<>& x) {
  mat X = as_Mat(x);
  mat B = X.each_col() / sum(X, 1);
  B = B.each_row() / (sum(X, 0) / accu(X));
  B.elem(find(B < 1)).zeros();
  B.elem(find(B >= 1)).ones();
  return B;
}
```

After implementing the C++ computation in 'cpp11armadillo' and 'RcppArmadillo', we compared the execution time of obtaining the Balassa index year by year for the 2002-2020 period, which involves 19 matrices with an approximate dimension of $230 \times 5,100$. After a build time of 4.4s for 'cpp11armadillo' and 9.5s for 'RcppArmadillo'. The following plot reveals almost identical execution times for the Armadillo function called from 'cpp11armadillo' and 'RcppArmadillo' with a marginal advantage for 'cpp11armadillo', and both are significantly faster than the base R function.

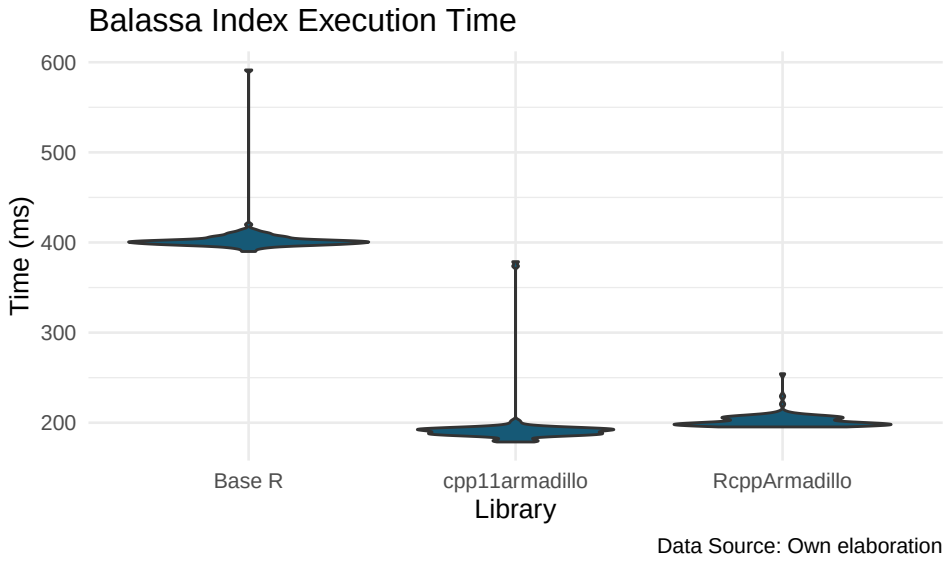

Figure 2. Balassa Index execution time reveals speed gains of around 50% for Armadillo implementations compared to base R.

The benchmarks were conducted on a ThinkPad X1 Carbon Gen 9 with the following specifications:

- Processor: Intel Core i7-1185G7 with eight cores
- Memory: 16 GB LPDDR4Xx-4266
- Operating System: Pop!_OS 22.04 based on Ubuntu 22.04
- R Version: 4.4.1
- BLAS Library: OpenBLAS 0.3.20

Conclusion

'cpp11armadillo' provides a simple and efficient way to integrate C++ code with R, leveraging the 'cpp11' package and the Armadillo library. It simplifies the process of writing C++ code for R users, allowing them to focus on the logic of the algorithm rather than the technical details of the integration. It can help to solve performance bottlenecks in 'R' code by using the efficient linear algebra operations provided by Armadillo in cases where vectorization is challenging. 'RcppArmadillo' is a popular package for this purpose, it has existed for nearly a decade, 'cpp11armadillo' offers a different approach with similar performance.

References

1. Eddelbuettel D, Francois R. Rcpp: Seamless R and C++ Integration. Journal of Statistical Software. 2011;40:1–18. doi:10.18637/jss.v040.i08.
2. Vaughan D, Hester J, François R. 'cpp11': A C++11 Interface for R's C Interface; 2023. Available from: https://CRAN.R-project.org/package=cpp11.
3. Sanderson C, Curtin R. Armadillo: a template-based C++ library for linear algebra. Journal of Open Source Software. 2016;1(2):26. doi:10.21105/joss.00026.
4. Eddelbuettel D, Sanderson C. 'rcpparmadillo': Accelerating R with high-performance C++ linear algebra. Computational Statistics & Data Analysis. 2014;71:1054–1063. doi:10.1016/j.csda.2013.02.005.
5. Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, et al. Welcome to the Tidyverse. Journal of Open Source Software. 2019;4(43):1686. doi:10.21105/joss.01686.
6. 'R' Core Team. 'R': A Language and Environment for Statistical Computing; 2024. Available from: https://www.R-project.org/.
7. Stroustrup B. Tour : Standard C++; 2012. Available from: https://isocpp.org/tour.
8. Burns P. The R inferno. Lulu; 2011.
9. Wickham H, Ooms J, Müller K. 'rpostgres': C++ Interface to PostgreSQL; 2023. Available from: https://CRAN.R-project.org/package=RPostgres.
10. Stack Overflow. Stack Exchange Data Explorer; 2024. Available from: https://data.stackexchange.com/stackoverflow/query/new.
11. Wickham H, Bryan J. R packages. "O'Reilly Media, Inc."; 2023.

12. Vargas Sepulveda M. economiccomplexity: Computational Methods for Economic Complexity. Journal of Open Source Software. 2020;5(46):1866. doi:10.21105/joss.01866.