Portability of Using C++20 in R Packages

Sherman Lo
Queen Mary, University of London

About This Talk

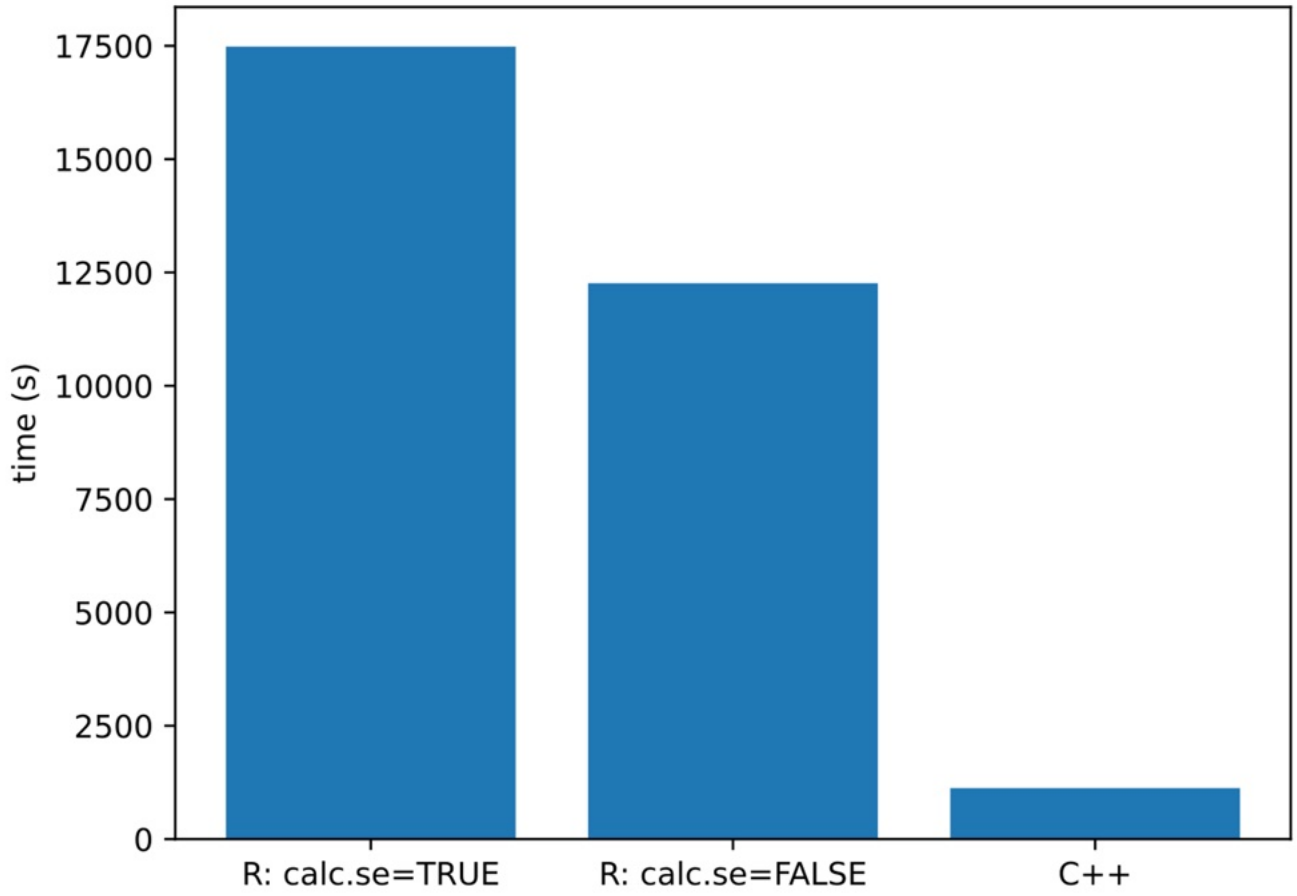
- Tell a story of building and maintaining a R package with C++
- Friction points
  - Portability
  - CRAN

About Me

- Research Software Engineer
- Centralised
- HPC facilities

The Task at Hand

- A user's code is very slow
- Uses the R package **poLCA**
- Rewrite the underlying calculations in C++ with Rcpp
- **poLCAParallel**


17500
15000
12500
10000
7500
5000
2500
0
time (s)
R: calc.se=TRUE
R: calc.se=FALSE
C++

OPEN ACCESS PEER-REVIEWED

RESEARCH ARTICLE

Ethnic differences in early onset multimorbidity and associations with health service use, long-term prescribing, years of life lost, and mortality: A cross-sectional study using clustering in the UK Clinical Practice Research Datalink

Fabiola Eto, Miriam Samuel, Rafael Henkin, Meera Mahesh, Tahania Ahmad, Alisha Angdembe, R. Hamish McAllister-Williams, Paolo Missier, Nick J. Reynolds, Michael R. Barnes, Sally Hull, Sarah Finer, Rohini Mathur

Published: October 27, 2023 • https://doi.org/10.1371/journal.pmed.1004300

Acknowledgments

The authors would like to thank Sherman Lo, research software engineer in the ITS Research, Queen Mary University of London, for assistance in expanding and improving the poLCA R library by creating the poLCAParallel R package and enabling a much faster run of our codes.

Finishing Touches

- Added documentation
- The user tested it
- Job done!

npj | precision oncology

Article | Open access | Published: 23 June 2023

Latent class analysis-derived classification improves the cancer-specific death stratification of molecular subtyping in colorectal cancer

Wen Zhou, Ming-Ming He, Feng Wang, Rui-Hua Xu, Fang Wang ✉ & Qi Zhao ✉

npj Precision Oncology **7**, Article numbe

7072 Accesses | **4** Citations | **1** Altm

than or equal to 0.5 was used to assign a class to each participant. The class with the largest posterior probability was assigned to that participant. All LCAs were conducted using poLCA in R version 4.0.3 (RRID: SCR_003005). The poLCAParallel[69], a reimplementation of poLCA, was used to speed up the running. The latent class models were estimated with the default parameters (graphs=FALSE, tol=1e-10, na.rm=TRUE, calc.se=TRUE), except for nclass=1–6, maxiter=1000, and nrep=30. Additionally, as the numerical order of the estimated latent classes in the model output is determined solely by the start values of the expectation-maximization (EM) algorithm, the poLCA.reorder command was used to ensure consistency

nature communications

Article | Open access | Published: 24 November 2023

Epidemiology, mortality, and health service use of local-level multimorbidity patterns in South Spain

Javier Alvarez-Galvez ✉, Esther Ortega-Martin, Begoña Ramos-Fiol, Victor Suarez-Lledo & Jesus Carretero-Bravo

Nature Communications 14, Art

Given the large sample size and the number of conditions, we computed the LCA models with the *POLCAParallel*[69,70] package, which implements the usual LCA using all processor cores, thus decreasing computation time. We established the number of appropriate patterns in each LCA model, considering three criteria[71]. Initially, we assessed the goodness-of-fit indices of the models, taking into consideration BIC, ABIC and CAIC. A lower value of these indices indicates a better fit of the model. Additionally, we examined the probability of membership in each class (i.e., multimorbidity pattern) and assessed the clinical interpretability of the

Annals of Intensive Care
Volume 16, 2026, 100031

Research

Characterizing ICU-like profiles of very old patients hospitalized in medical intermediate care units in France: A clustering analysis of a nationwide population-based study

Arthur Kassa-Sombo [a 1], Adrien I
Grammatico-Guillon [c e], Antoine

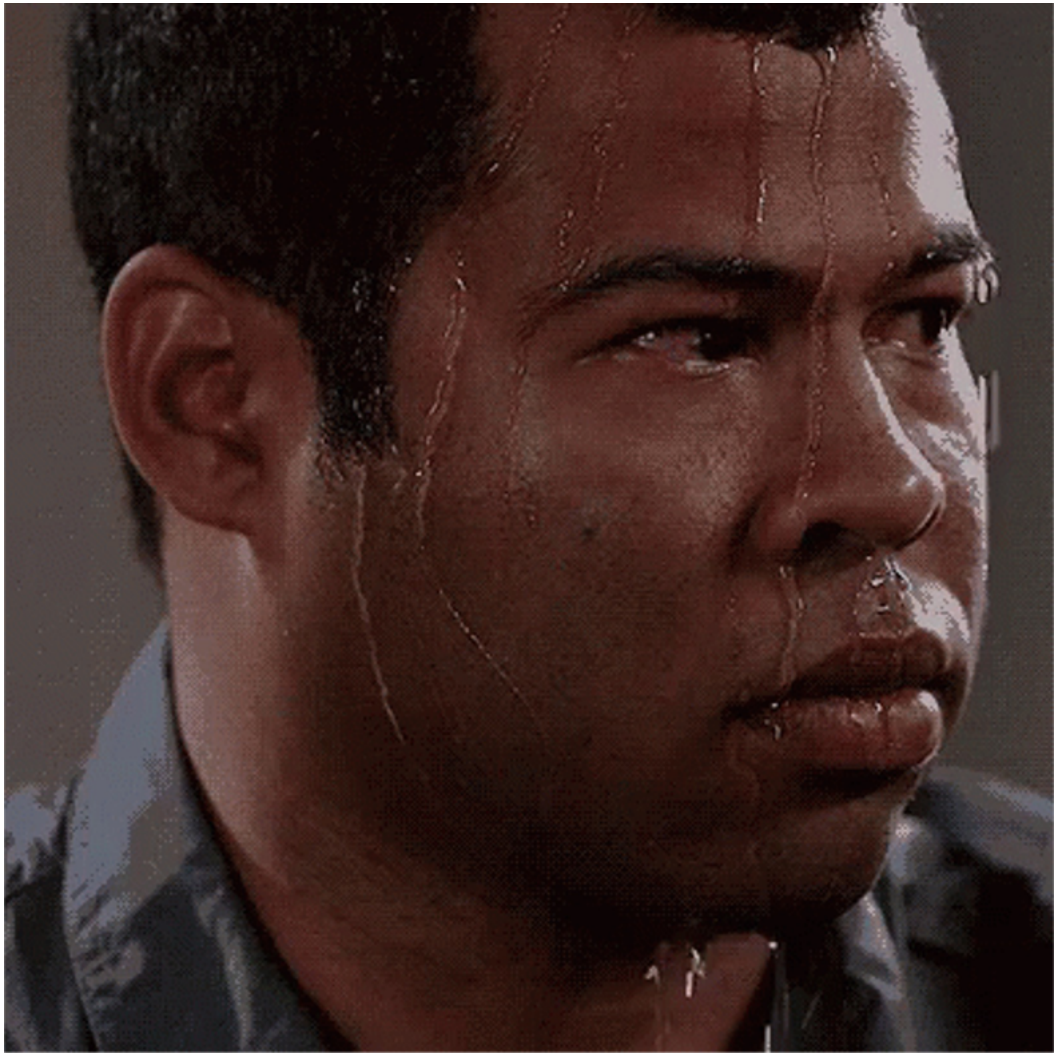
The AIC, BIC and aBIC served as criteria for class number selection with lower value indicating a better trade-off between fit and complexity of model. Entropy measures the clarity of classification, and the separation of classes ranged from 0 to 1. An entropy of ≥0.80 indicates a well-defined classification with clear separation between classes. The LMRa test was used to assess whether a model with k classes provided a significantly better fit than a model with k-1 classes. LCA was performed using poLCA in R Version 4.3.2. The poLCAParallel, a reimplementation of poLCA, was used to speed up the running.

Psychology

- This is amazing! Time to celebrate!

Psychology

- This is amazing! Time to celebrate!
- OMG!!! PEOPLE ARE ACTUALLY USING MY CODE!!!



Issue from ONS

- Missing data is supported in poLCA
- Missing data crashes poLCAParallel

Maintenance

- Add missing data support
- **Modernise C++ standards**
- Add testing
- **Submit to CRAN**

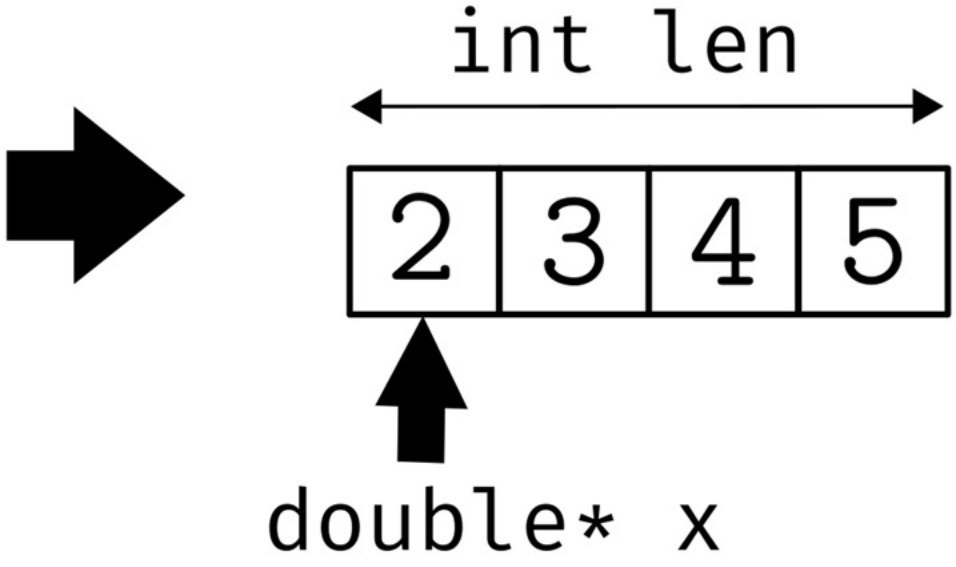
Confession

Confession

- I used raw pointers in my C++ code

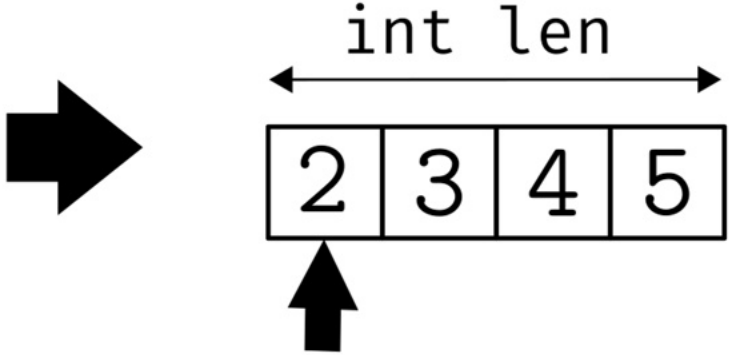
$$\begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix}$$

R NumericMatrix



$$\begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix}$$

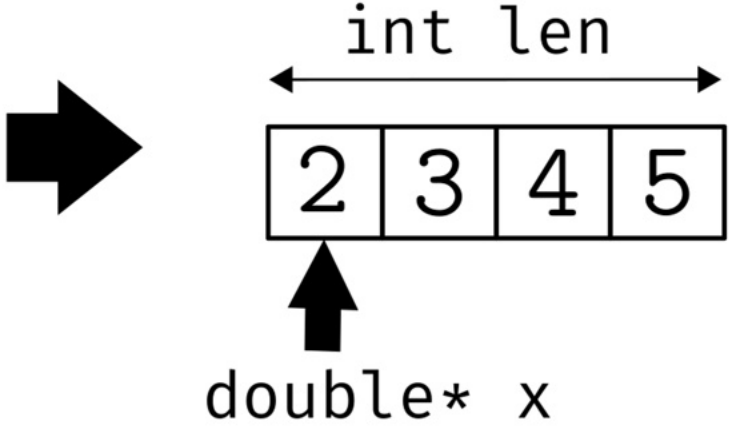
R NumericMatrix




```
x + 1;
x[1] = some_number;
x[100] = some_number;
```

$$\begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix}$$

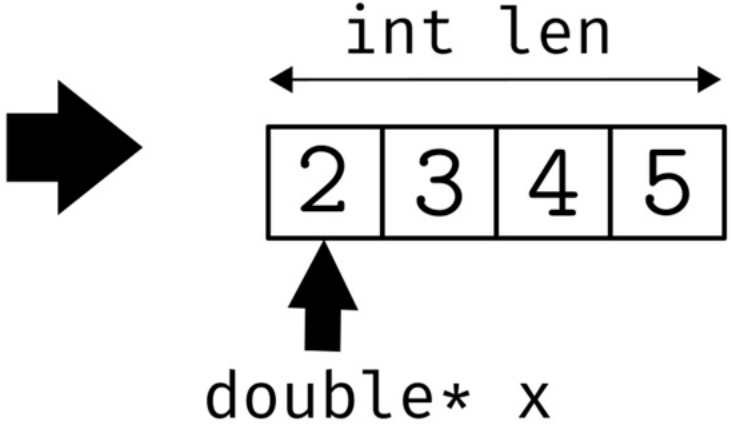
R NumericMatrix




```
x + 1;
x[1] = some_number;
x[100] = some_number;
```

$$\begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix}$$

R NumericMatrix



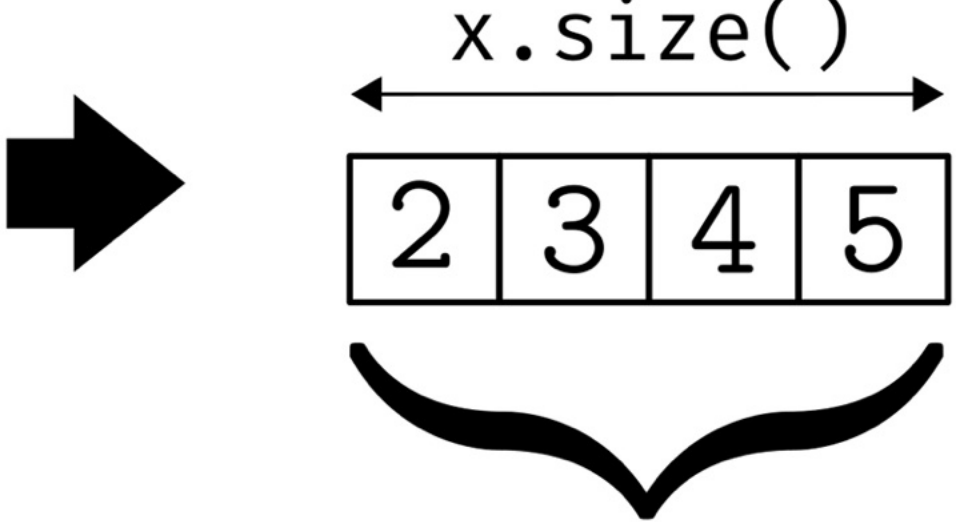

```
for (int i=0; i<len; i++) {
  x[i];
}
```

C++ 20

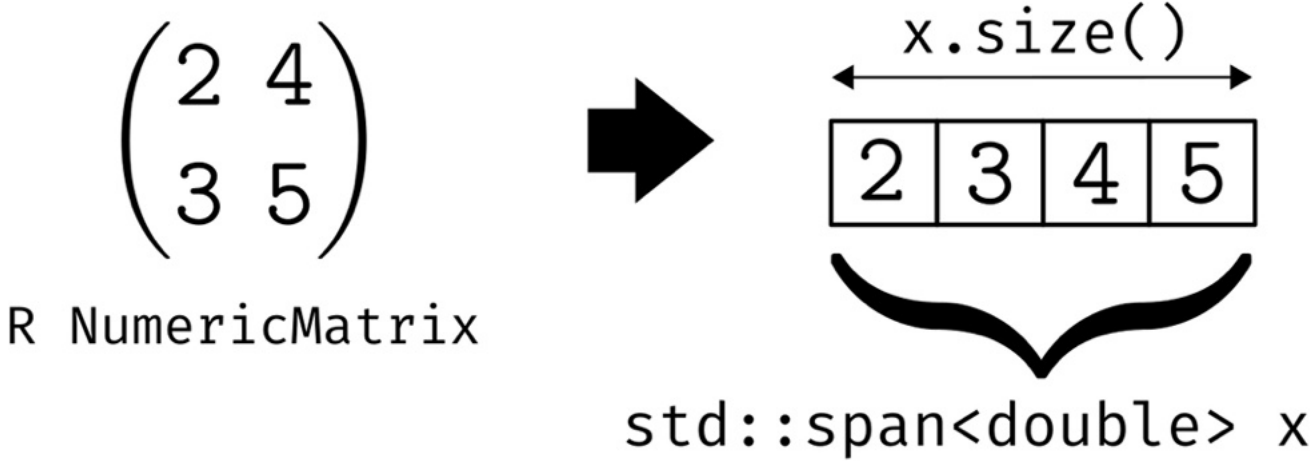
- std::span

$$\begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix}$$

R NumericMatrix



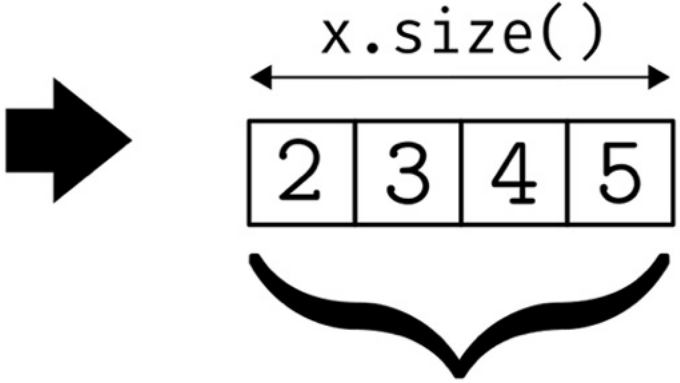
std::span<double> x




```
x[1] = some_number;
x[100] = some_number;
```

$$\begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix}$$

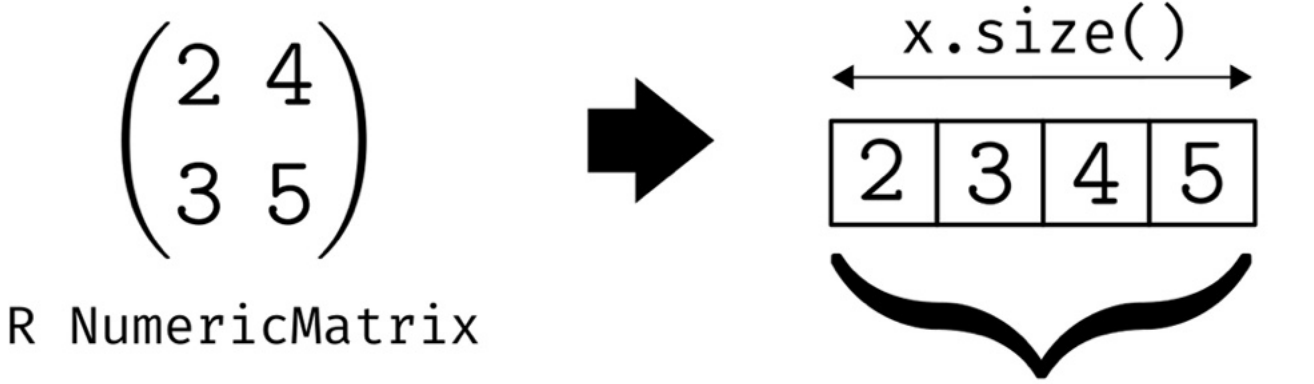
R NumericMatrix



std::span<double> x

```
x[1] = some_number;
x[100] = some_number;
```

Now fails with -D_GLIBCXX_DEBUG




```
for (auto x_i : x) {
  x_i;
}
```

Explicit Construction

```
std::span<double>(double*, int)
```

Implicit Construction

```
std::span<double>(R NumericMatrix)
```

```
106  106       // fit using EM algorithm
107  107       polca_parallel::EmAlgorithmArray fitter(
108       -        std::span<const double>(features.cbegin(), features.size()),
109       -        std::span<const int>(responses.cbegin(), responses.size()),
110       -        std::span<const double>(initial_prob.cbegin(), initial_prob.size()),
111       -        n_data, n_feature, n_outcomes, n_cluster, n_rep, n_thread, max_iter,
112       -        tolerance, std::span<double>(posterior.begin(), posterior.size()),
113       -        std::span<double>(prior.begin(), prior.size()),
114       -        std::span<double>(estimated_prob.begin(), estimated_prob.size()),
115       -        std::span<double>(regress_coeff.begin(), regress_coeff.size()));
     108  +        features, responses, initial_prob, n_data, n_feature, n_outcomes,
     109  +        n_cluster, n_rep, n_thread, max_iter, tolerance, posterior, prior,
     110  +        estimated_prob, regress_coeff);
116  111
```

CRAN

- Submitted to CRAN but with some friction

Flavor	Version	$T_{install}$	T_{check}	T_{total}	Status	Flags
r-devel-linux-x86_64-debian-clang	1.2.7	201.65	80.11	281.76	OK	
r-devel-linux-x86_64-debian-gcc	1.2.7	113.81	57.30	171.11	OK	
r-devel-linux-x86_64-fedora-clang	1.2.7	288.00	117.06	405.06	OK	
r-devel-linux-x86_64-fedora-gcc	1.2.7	344.00	117.51	461.51	OK	
r-devel-macos-arm64	1.2.7	33.00	-4.00	29.00	OK	
r-devel-windows-x86_64	1.2.7	225.00	123.00	348.00	OK	
r-patched-linux-x86_64	1.2.7	166.54	74.81	241.35	OK	
r-release-linux-x86_64	1.2.7	157.80	72.08	229.88	OK	
r-release-macos-arm64	1.2.7	21.00	-12.00	9.00	ERROR	
r-release-macos-x86_64	1.2.7	72.00	-15.00	57.00	ERROR	
r-release-windows-x86_64	1.2.7	222.00	0.00	222.00	OK	
r-oldrel-macos-arm64	1.2.7	17.00	-4.00	13.00	ERROR	
r-oldrel-macos-x86_64	1.2.7	62.00	-18.00	44.00	ERROR	
r-oldrel-windows-x86_64	1.2.7	262.00	146.00	408.00	OK	

MacOS

- **Implicit conversion not supported**
- std::jthread is not supported

Decision: revert

Lessons

- Version control
- Not bother with CRAN?
- Test on different OS (or compilers and standard libraries)
- C++23

Sherman Lo
Research Software Engineer

Metadata

June 15, 2026

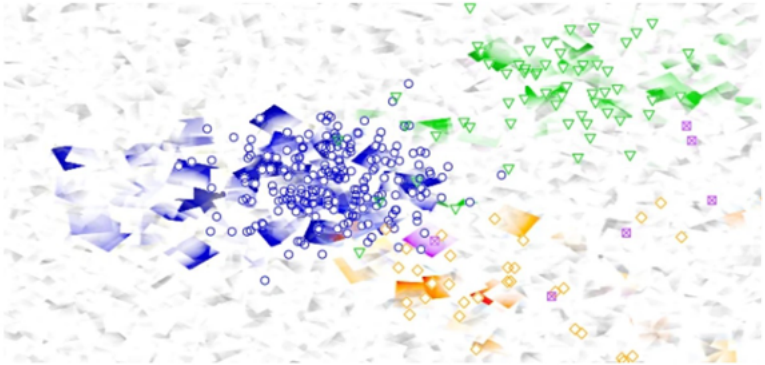
in rse

21 min read

C++ R optimisation

Modernising Rcpp and C++ Code With `std::span`

Table of contents



The programming language C++ is still widely used today, especially for high-perform computing. Out-of-date practices from the 80s and 90s should still work now becaus language is designed to be highly backwards compatible. As C++ evolved, there were efforts to modernise the language and update programming practices with new featu improve the C++ experience, such as memory-safety features.

As new practices are introduced, there are multiple ways to do the same thing. Entrer programmers may stick with older practices, whereas modern programmers may kee practices up to date. Newer programmers have a clean slate to learn best practices a habit, but they may also pick up an old textbook or tutorial and learn out-of-date practices.