The best of two worlds: the optimal usage of the Rcpp package in R

Data Science Summit 2020

Jadwiga Słowik

About me

- Passionate programmer since 11 years old
- Strong algorithmic and software engineering background
- Polyglot programmer:
  R, Python, C++, Java, Kotlin, C#, C and Dart
- Data Science master degree student at
  Warsaw University of Technology

The comparison of R and C++ languages

	R	**C++**
Typing	Weak	Strong
Memory management	Automatic	Partially automatic
Threading	Single-threaded	Multi-threaded
Safety	Safe	Unsafe
Low-level optimization	No	Yes
High-level interface	Yes	No
Package management	Effortless	Inconvenient
Availability of packages	High	Low
	Convenient to use	Efficient

Choose one

Jadwiga Słowik jadwiga.slowik5@gmail.com Twitter: @slowikj5

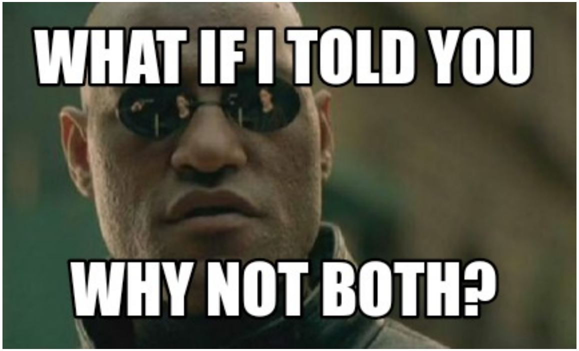
Situation no. 1

Situation no. 2

Which one to choose?

Why not both?

No single language or software system is likely to be ideal for all aspects. **Interfacing multiple systems is essential.**
Extending R, Chapter 4



Rescue

What is the Rcpp? I

- **an R package that facilitates the integration between R and C++**

What is the Rcpp? II

- an R package that facilitates the integration between R and C++
- **makes a C++ function accessible in R**

What is the Rcpp? III

- an R package that facilitates the integration between R and C++
- makes a C++ function accessible in R
- **an R API wrapper**

What is the Rcpp? IV

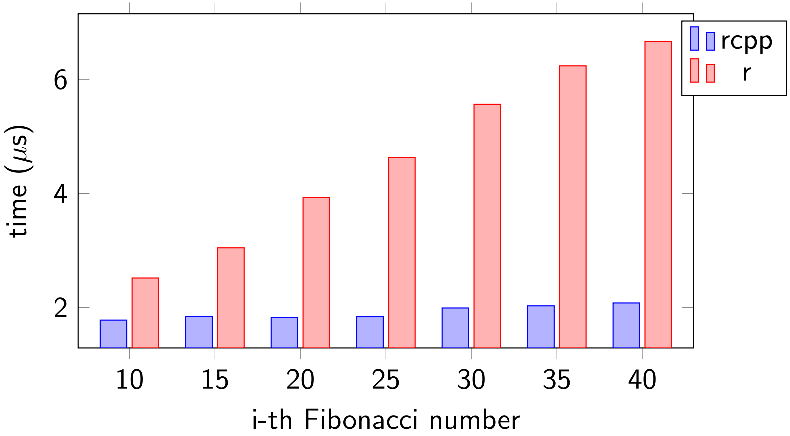
- an R package that facilitates the integration between R and C++
- makes an C++ function accessible in R
- a R API wrapper
- **provides C++ structures for R types (vectors, matrices, and more...)**

What is the Rcpp? V

- an R package that facilitates the integration between R and C++
- makes a C++ function accessible in R
- an R API wrapper
- provides C++ structures for R types (vectors, matrices, and more...)
- **it does not copy R objects during the invocation of a C++ function from R code**

Rewriting R code in C++

Figure 1: Fibonacci sequence computation

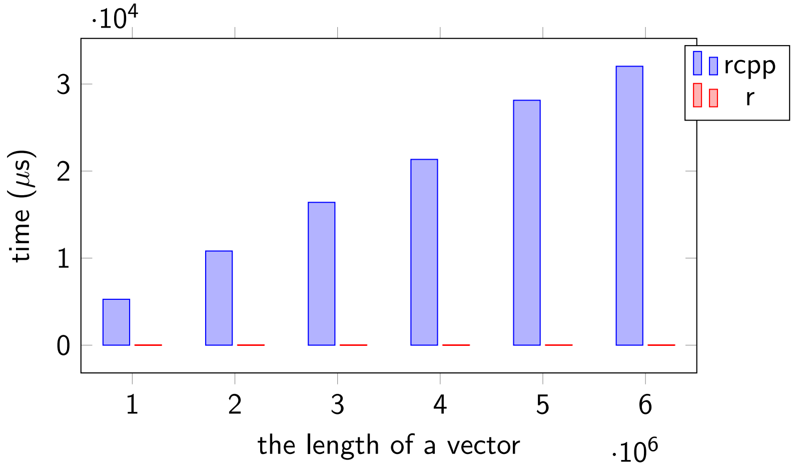


Jadwiga Słowik jadwiga.slowik5@gmail.com Twitter: @slowikj5

Is it just enough to write C++ code and export it to R?

Not exactly.

Figure 2: Linear search in Rcpp and binary search in R



Where is it worth using Rcpp I

Where is it worth using Rcpp II

- **For-loops that cannot be vectorized**

Where is it worth using Rcpp III

- For-loops that cannot be vectorized
- **The usage of advanced data structures: trees, queues, sets...**

Jadwiga Słowik jadwiga.slowik5@gmail.com Twitter: @slowikj5

Where is it worth using Rcpp IV

- For-loops that cannot be vectorized
- **The usage of advanced data structures: trees, queues, sets...**
    - In particular, you can use STL, Boost, ...

Where is it worth using Rcpp V

- For-loops that cannot be vectorized
- The usage of advanced data structures: trees, queues, sets...
- **Recursive functions or calling functions many times**

Where is it worth using Rcpp VI

- For-loops that cannot be vectorized
- The usage of advanced data structures: trees, queues, sets...
- Recursive functions or calling functions many times
- **Algorithms that require an imperative approach**

The dark side of Rcpp

Allocation of pure C++ objects vs Rcpp objects I

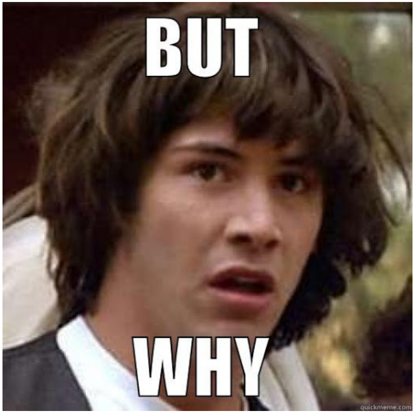
```
for(int i = 0; i < n; ++i) {
    std::vector<int> v(30);
}
```

Figure 3: C++ object creation in a loop

```
for(int i = 0; i < n; ++i) {
    Rcpp::IntegerVector v(30);
}
```

Figure 4: Rcpp object creation in a loop

Allocation of pure C++ objects vs Rcpp objects II



Allocation of pure C++ objects vs Rcpp objects III

- **In both cases in each iteration the destructor of a current object is invoked**

Allocation of pure C++ objects vs Rcpp objects IV

- In both cases in each iteration the destructor of a current object is invoked
- **However, the behavior of the destructor of an Rcpp is totally different**

Allocation of pure C++ objects vs Rcpp objects V

- In both cases in each iteration the destructor of a current object is invoked
- However, the behavior of the destructor of an Rcpp is totally different
- **Actually, it does not release the inner memory immediately (as opposed to the standard C++ destructor)**

Allocation of pure C++ objects vs Rcpp objects VI

- In both cases in each iteration the destructor of a current object is invoked
- However, the behavior of the destructor of an Rcpp is totally different
- Actually, it does not release the inner memory immediately (as opposed to the standard C++ destructor)
- **The Rcpp destructor just calls R memory management**

Allocation of pure C++ objects vs Rcpp objects VII

- In both cases in each iteration the destructor of a current object is invoked
- However, the behavior of the destructor of an Rcpp is totally different
- Actually, it does not release the inner memory immediately (as opposed to the standard C++ destructor)
- The Rcpp destructor just calls R memory management
- **R garbage collector can be called only after the invocation of an R API method**

Allocation of pure C++ objects vs Rcpp objects VIII

A solution:
If you're not going to return something to R, there is no reason to create an Rcpp object to contain it.
(look at the issue 482 in Rcpp github repo)

Inconsistency in the behavior of assignment operators

```
std::vector<int> v {1, 2, 3};
// copy of the vector v:
std::vector<int> w = v;
v[1] = 100; // w is still c(1, 2, 3)
```

Figure 5: Pure C++

```
Rcpp::IntegerVector v {1, 2, 3};
// no copy of vector v:
Rcpp::IntegerVector w = v;
v[1] = 100; // w was modified: c(1, 100, 3)
```

Figure 6: Rcpp

Inconsistency in attributes names I

Let us suppose that **a variable `x` has one attribute** `velocity` and does not have more attributes whose names starts with `vel`.

Jadwiga Słowik jadwiga.slowik5@gmail.com Twitter: @slowikj5

Inconsistency in attributes names II

Let us suppose that a variable `x` has one attribute `velocity` and does not have more attributes whose names starts with `vel`.
In R it is valid to use:

```
attr(x, "vel")
```

Inconsistency in attributes names III

Let us suppose that a variable `x` has one attribute `velocity` and does not have more attributes whose names starts with `vel`.
In R it is valid to use:

```
attr(x, "vel")
```

However, in Rcpp we will get `NULL`:

```
x.attr("vel")
```

Inconsistency in attributes names IV

Let us suppose that a variable `x` has one attribute `velocity` and does not have more attributes whose names starts with `vel`.
In R it is valid to use:

```
attr(x, "vel")
```

However, in Rcpp we will get `NULL`:

```
x.attr("vel")
```

Therefore, in Rcpp we have to provide full names

What exactly does the following code return? I

```
// [[Rcpp::export]]
Rcpp::IntegerVector getVector() {
    return 5;
}
/*** R
getVector()
*/
```

What exactly does the following code return? II

```
// [[Rcpp::export]]
Rcpp::IntegerVector getVector() {
    return 5;
}
/*** R
getVector()
*/
```

Answer: `c(0, 0, 0, 0, 0)`

What exactly does the following code return? III

```
// [[Rcpp::export]]
Rcpp::IntegerVector getVector() {
    return 5;
}
/*** R
getVector()
*/
```

Answer: `c(0, 0, 0, 0, 0)`
Why: In C++, the constructor `IntegerVector(int n)` is not explicit, so implicit conversions are allowed!

Beware of NULLs! I

```
// [[Rcpp::export]]
void f(Rcpp::IntegerVector v)
{
    // some code...
}

/*** R
f(c())
*/
```

Beware of NULLs! II

```
// [[Rcpp::export]]
void f(Rcpp::IntegerVector v)
{
    // some code...
}

/*** R
f(c())
*/
```

An error will be raised!

Beware of NULLs! III

Use `Rcpp::Nullable` **as a wrapper!**

```
// [[Rcpp::export]]
void f(Rcpp::Nullable<Rcpp::IntegerVector> v)
{
    // some code...
}

/*** R
f(c())
*/
```

Multithreaded environment (RcppParallel) I

Multithreaded environment (RcppParallel) II

- **Calling any R function is potentially unsafe in multithreaded environment**

Multithreaded environment (RcppParallel) III

- Calling any R function is potentially unsafe in multithreaded environment
- **Particularly, the creation of a new Rcpp object is not allowed**

Multithreaded environment (RcppParallel) IV

```
// v is an Rcpp::IntegerVector
int elem = v[i];
```

Multithreaded environment (RcppParallel) V

```
// v is an Rcpp::IntegerVector
int elem = v[i];
```

Multithreaded environment (RcppParallel) VI

```cpp
// v is an Rcpp::IntegerVector
int elem = v[i];
```

Multithreaded environment (RcppParallel) VII

```
Warning:  stack imbalance in '.Call', 33 then 32
Warning:  stack imbalance in '<-', 31 then 30
```

Figure 7: An example warning message related to RcppParallel multithreading issues

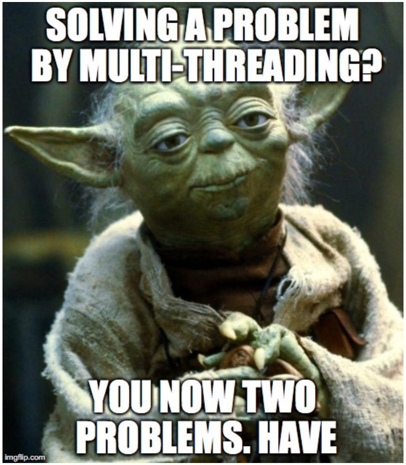
```
Error:  C stack usage is too close to the limit
```

Figure 8: An example error message related to RcppParallel multithreading issues

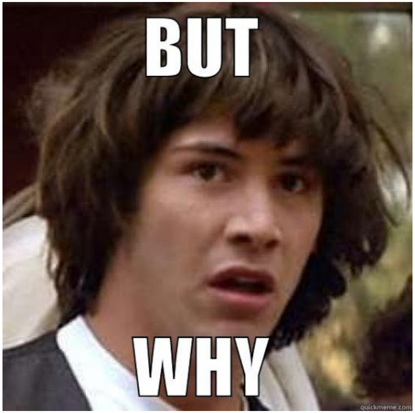
```
Error:  stack smashing detected
```

Figure 9: Another error message related to RcppParallel multithreading issues

Multithreaded environment (RcppParallel) VIII


SOLVING A PROBLEM BY MULTI-THREADING?
YOU NOW TWO PROBLEMS. HAVE
imgflip.com

Multithreaded environment (RcppParallel) IX



Multithreaded environment (RcppParallel) X

- **An Rcpp indexing entails the creation of a new (proxy) object**

Multithreaded environment (RcppParallel) XI

- An Rcpp indexing entails the creation of a new (proxy) object
- **Therefore, R API is invoked**

Multithreaded environment (RcppParallel) XII

- An Rcpp indexing entails the creation of a new (proxy) object
- Therefore, R API is invoked
- **Solution: use dedicated classes RVector and RMatrix**

Multithreaded environment (RcppParallel) XIII

- An Rcpp indexing entails the creation of a new (proxy) object
- Therefore, R API is invoked
- Solution: use dedicated classes *RVector* and *RMatrix*
- **Unfortunately, StringVector and StringMatrix is not supported (because of more complex memory representation)**

Multithreaded environment (RcppParallel) XIV

- An Rcpp indexing entails the creation of a new (proxy) object
- Therefore, R API is invoked
- Solution: use dedicated classes *RVector* and *RMatrix*
- Unfortunately, *StringVector* and *StringMatrix* is not supported (because of more complex memory representation)
- **Thus, the additional cost of the conversion to pure C++ is required or remapping string elements to integers**

The `Rcpp::Fast<VECTOR>` wrapper I

The `Rcpp::Fast<VECTOR>` wrapper II

- **It is an Rcpp vector wrapper**

The `Rcpp::Fast<VECTOR>` wrapper III

- It is an Rcpp vector wrapper
- **It provides indexing (read and write) and getting the size of a vector**

The `Rcpp::Fast<VECTOR>` wrapper IV

- It is an Rcpp vector wrapper
- It provides indexing (read and write) and getting the size of a vector
- **Operations are faster than on a raw vector and thread-safe**

The `Rcpp::Fast<VECTOR>` wrapper V

- It is an Rcpp vector wrapper
- It provides indexing (read and write) and getting the size of a vector
- Operations are faster than on a raw vector and thread-safe
- **It operates on a raw internal vector's memory**

The `Rcpp::Fast<VECTOR>` wrapper VI

- It is an Rcpp vector wrapper
- It provides indexing (read and write) and getting the size of a vector
- Operations are faster than on a raw vector and thread-safe
- It operates on a raw internal vector's memory
- **Rcpp::StringVector is not supported**

Disadvantages of a strongly typed language I

Disadvantages of a strongly typed language II

- **More discipline is required**

Disadvantages of a strongly typed language III

- More discipline is required
- **If the algorithm is the same, but matrices/vectors of several types are supported, we need to support all of the types separately**

Disadvantages of a strongly typed language IV

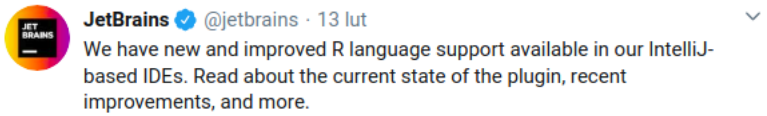
- More discipline is required
- If the algorithm is the same, but matrices/vectors of several types are supported, we need to support all of the types separately
- **You can make some abstractions of common code, for example using C++ templates**

Looking for the perfect IDE I

Looking for the perfect IDE II

Looking for the perfect IDE III

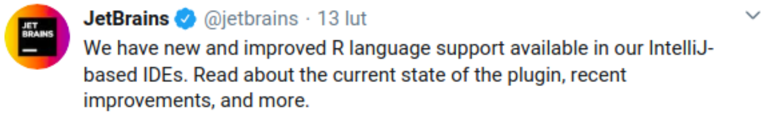
Jetbrains team starts supporting R in their tools!


JetBrains @jetbrains · 13 lut

We have new and improved R language support available in our IntelliJ-based IDEs. Read about the current state of the plugin, recent improvements, and more.

Looking for the perfect IDE IV

Jetbrains team starts supporting R in their tools!



JetBrains @jetbrains · 13 lut

We have new and improved R language support available in our IntelliJ-based IDEs. Read about the current state of the plugin, recent improvements, and more.

The state of the art:

- **CLion** supports building an R package
- You need to install a plugin **R language for IntelliJ**
- There is still the problem with debugging

Looking for the perfect IDE V

A solution:

1. Apply object oriented design principles in order to make abstractions (wrappers) for Rcpp structures

Looking for the perfect IDE VI

A solution:

1. Apply object oriented design principles in order to make abstractions (wrappers) for Rcpp structures

```
class IntegerVectorWrapper {
    // ...
public:
    virtual std::size_t size() const = 0;
    virtual int getElem(int index) const = 0;
};
```

Figure 10: An example contract for the IntegerVector class

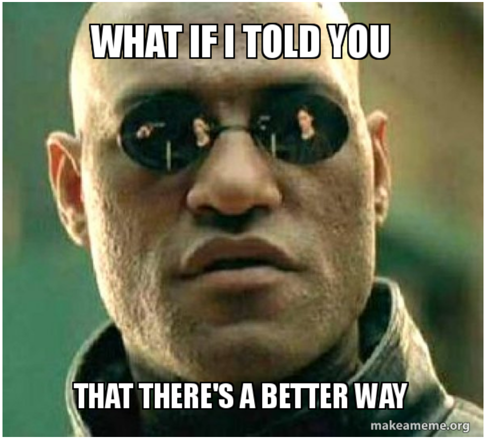
Looking for the perfect IDE VII

1. Apply object oriented design principles in order to make abstractions (wrappers) for Rcpp structures
2. **In order to debug C++ code: in CLion put a concrete pure C++ counterparts for Rcpp structures**

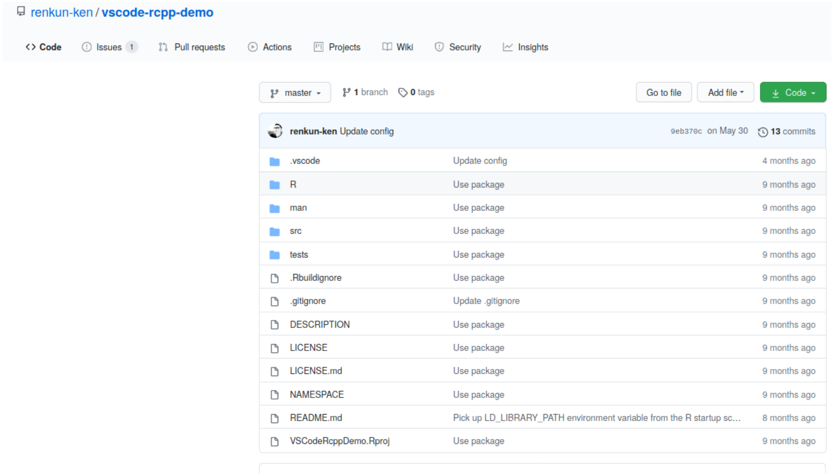
Looking for the perfect IDE VIII

1. Apply object oriented design principles in order to make abstractions (wrappers) for Rcpp structures
2. In order to debug C++ code: in CLion put a concrete pure C++ counterparts for Rcpp structures
3. **In order to build an R package: replace the aforementioned structures with Rcpp counterparts**

Looking for the perfect IDE IX



Looking for the perfect IDE X



renkun-ken Update config 9eb370c on May 30 13 commits

.vscode	Update config	4 months ago
R	Use package	9 months ago
man	Use package	9 months ago
src	Use package	9 months ago
tests	Use package	9 months ago
.Rbuildignore	Use package	9 months ago
.gitignore	Update .gitignore	9 months ago
DESCRIPTION	Use package	9 months ago
LICENSE	Use package	9 months ago
LICENSE.md	Use package	9 months ago
NAMESPACE	Use package	9 months ago
README.md	Pick up LD_LIBRARY_PATH environment variable from the R startup sc…	8 months ago
VSCodeRcppDemo.Rproj	Use package	9 months ago

Testing a package that uses Rcpp I

Testing a package that uses Rcpp II

- **Write R tests only for Rcpp functions that should be provided for R end users**

Testing a package that uses Rcpp III

- Write R tests only for Rcpp functions that should be provided for R end users
- **If you want to write tests for other (not exported) C++ functions, write C++ tests**

Testing a package that uses Rcpp IV

- Write R tests only for Rcpp functions that should be provided for R end users
- If you want to write tests for other (not exported) C++ functions, write C++ tests
- **An example package that uses C++ tests:** `mi2-warsaw/FSelectorRcpp`

Key takeaways I

Key takeaways II

1. **Profile your code and rewrite only bottlenecks in C++**

Premature optimization is the root of all evil
Donald Knuth

Key takeaways III

1. Profile your code and rewrite only bottlenecks in C++
2. **Do not call R API code in the multithreaded environment**

Key takeaways IV

1. Profile your code and rewrite only bottlenecks in C++
2. Do not call R API code in the multithreaded environment
3. **Do not create Rcpp objects that will not be returned to R**

Key takeaways V

1. Profile your code and rewrite only bottlenecks in C++
2. Do not call R API code in the multithreaded environment
3. Do not create Rcpp objects that will not be returned to R
4. **Try another IDE (CLion, Visual Studio Code, ...) to develop C++ code**

Key takeaways VI

1. Profile your code and rewrite only bottlenecks in C++
2. Do not call R API code in the multithreaded environment
3. Do not create Rcpp objects that will not be returned to R
4. Try another IDE (CLion, Visual Studio Code, ...) to develop C++ code
5. **Be aware that Rcpp objects behavior is different than for pure C++ objects (Reference vs value semantics)**

Key takeaways VII

1. Profile your code and rewrite only bottlenecks in C++
2. Do not call R API code in the multithreaded environment
3. Do not create Rcpp objects that will not be returned to R
4. Try another IDE (CLion, Visual Studio Code, ...) to develop C++ code
5. Be aware that Rcpp objects behavior is different than for pure C++ objects
6. **Write C++ tests (for example, using `testthat`) for the C++ functions that you do not want to export to R**

Thank you for your attention!

And special thanks to the research group (*Biogenies*) whom I have the pleasure to work with: