Where Rcpp wins and where it fails - the light and the dark side of R and Rcpp integration

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

The comparison of R and C++ languages

	R	**C++**
Typing	Weak	Strong
Memory management	Automatic	Partially automatic
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

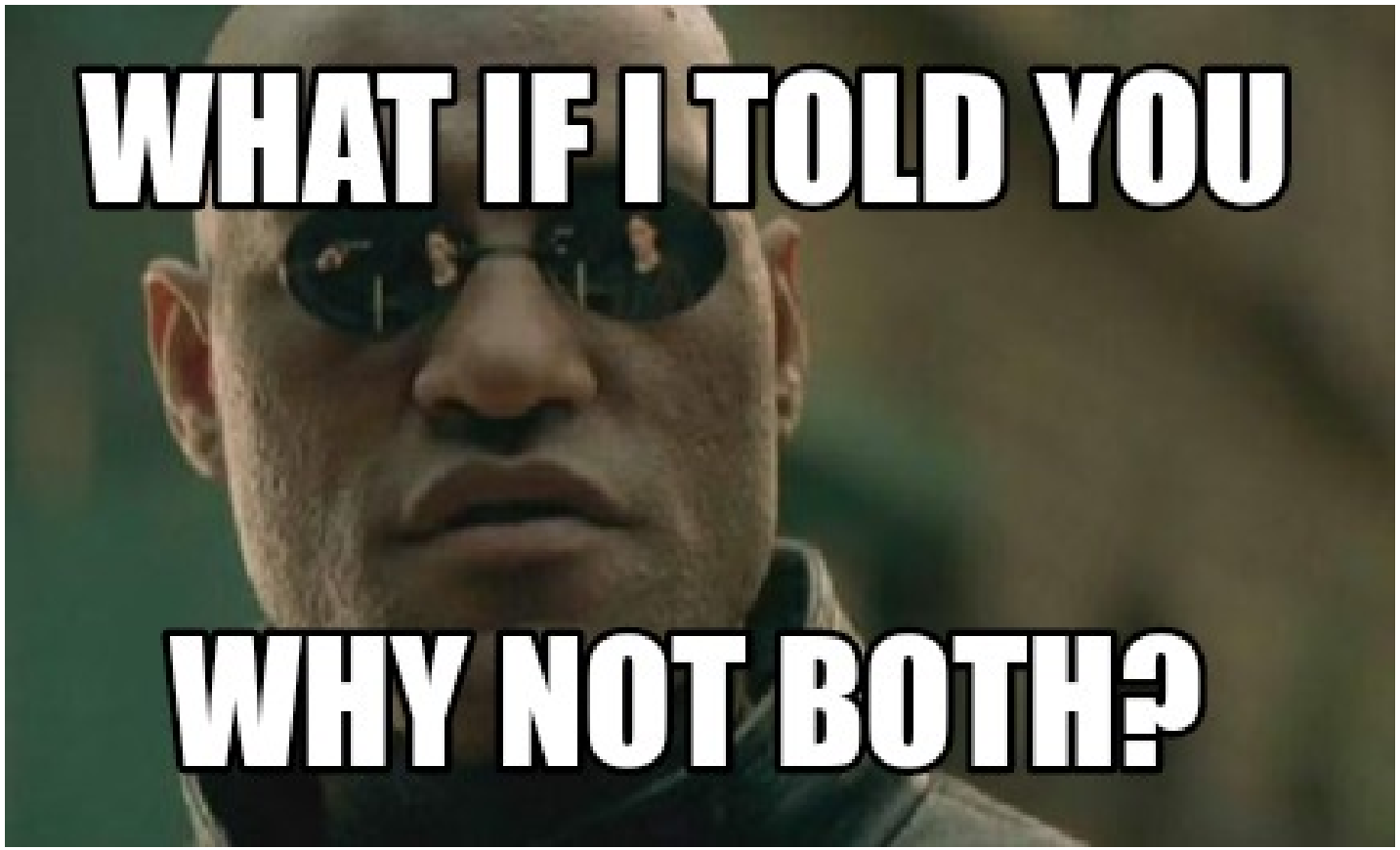
Situation no. 1

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



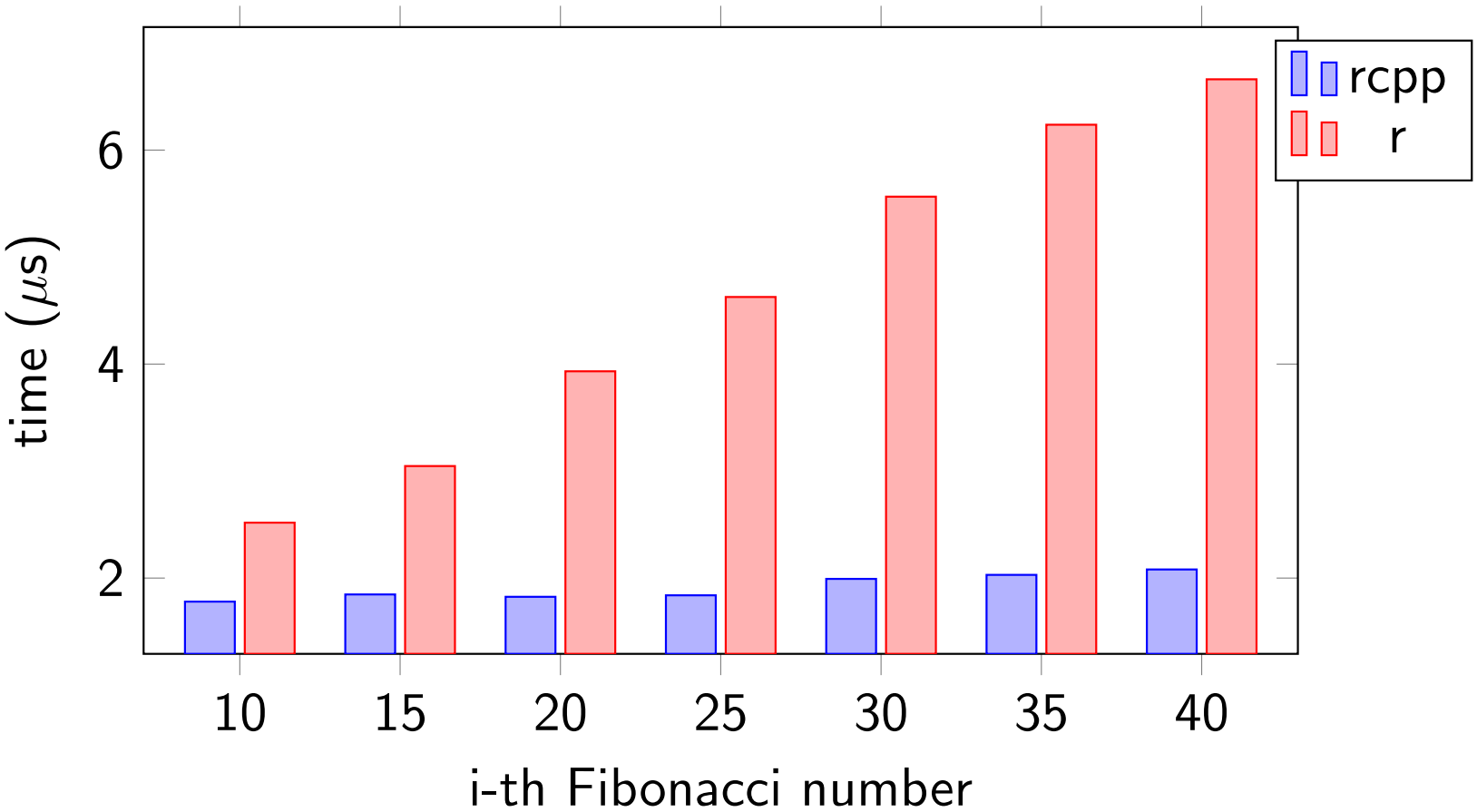
Rescue

What is the Rcpp?

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

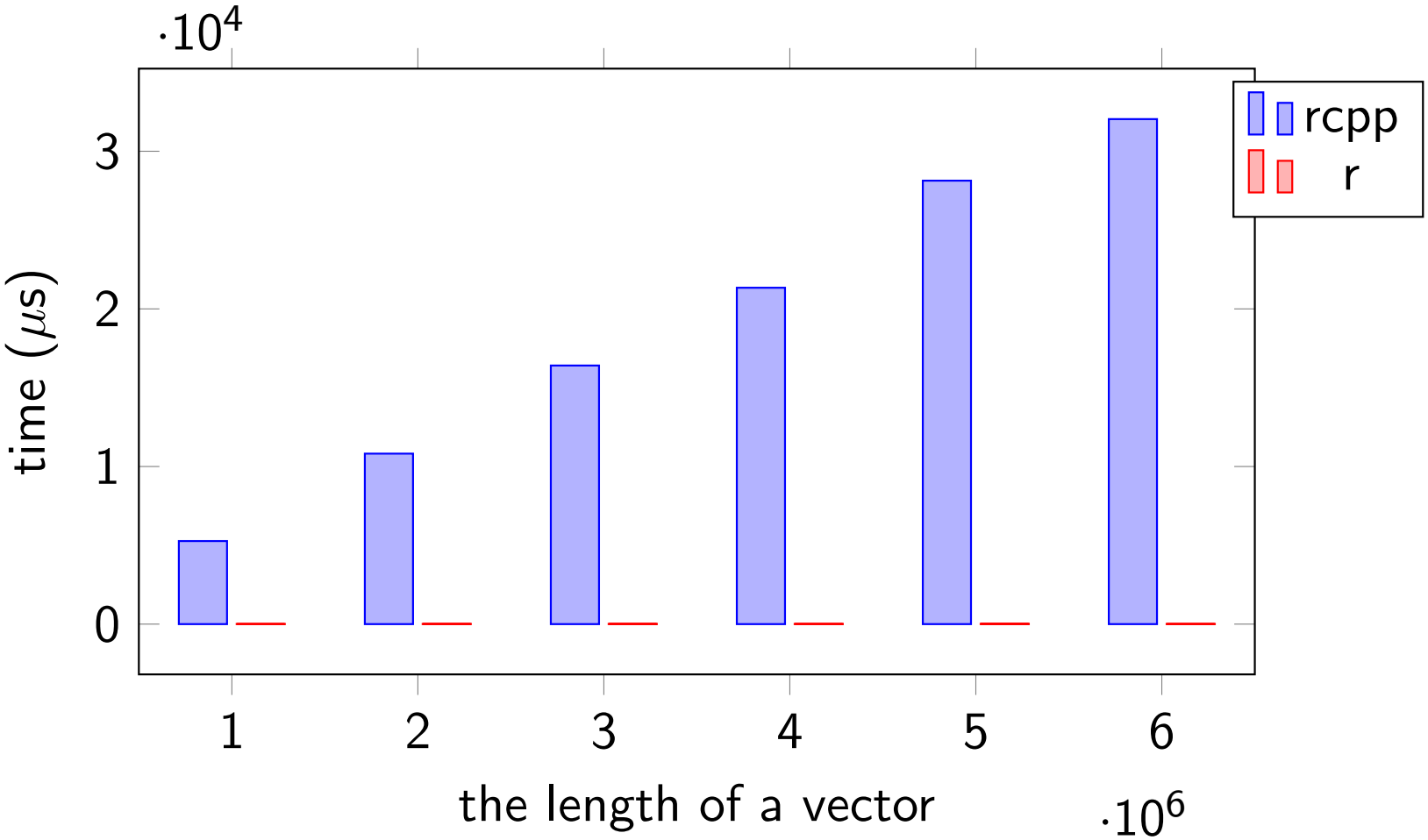
Figure 1: Fibonacci sequence computation



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

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

Where is it worth using Rcpp II

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

Where is it worth using Rcpp III

- 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 IV

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

The dark side of Rcpp

Allocation of pure C++ objects vs Rcpp objects I

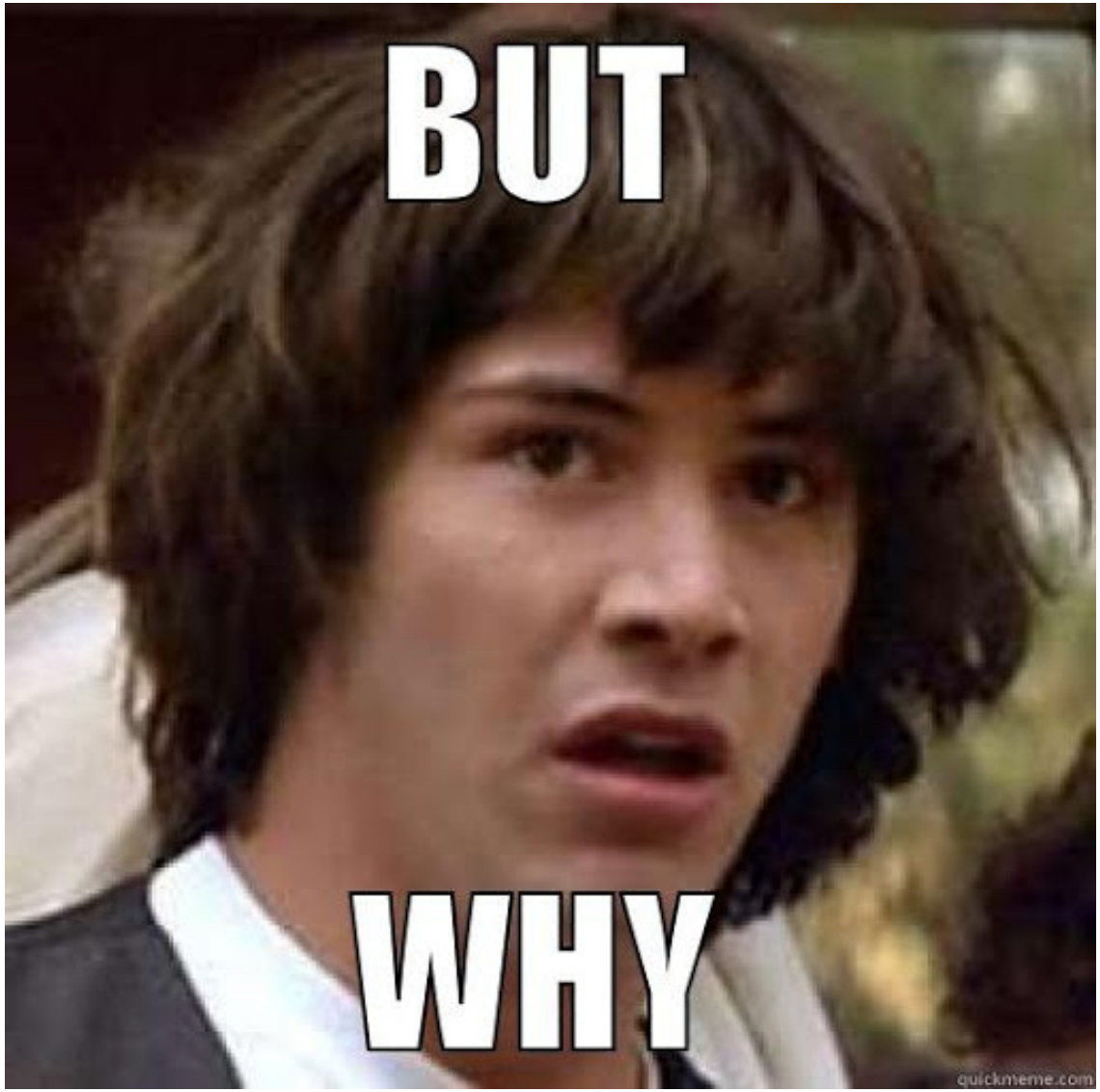
```
for(int i = 0; i < n; ++i) {
    std::vector<int> v(30);
}
```

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 do not have more attributes whose names starts with `vel`.

Inconsistency in attributes names II

Let us suppose that a variable `x` has one attribute `velocity` and do 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 do 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

Disadvantages of a strongly typed language I

- **More discipline is required**

Disadvantages of a strongly typed language II

- 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 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
- **You can make some abstractions of common code, for example using C++ templates**

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

Multithreaded environment (RcppParallel) I

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

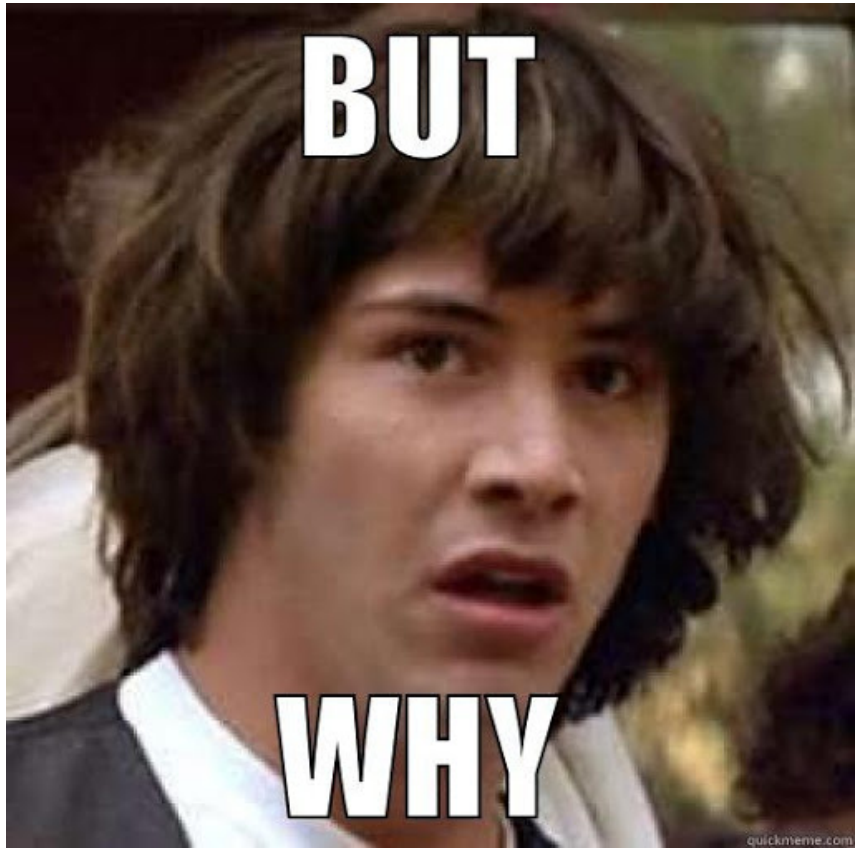
Multithreaded environment (RcppParallel) II

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

Multithreaded environment (RcppParallel) III

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

Multithreaded environment (RcppParallel) IV



Multithreaded environment (RcppParallel) V

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

Multithreaded environment (RcppParallel) VI

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

Multithreaded environment (RcppParallel) VII

- 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) VIII

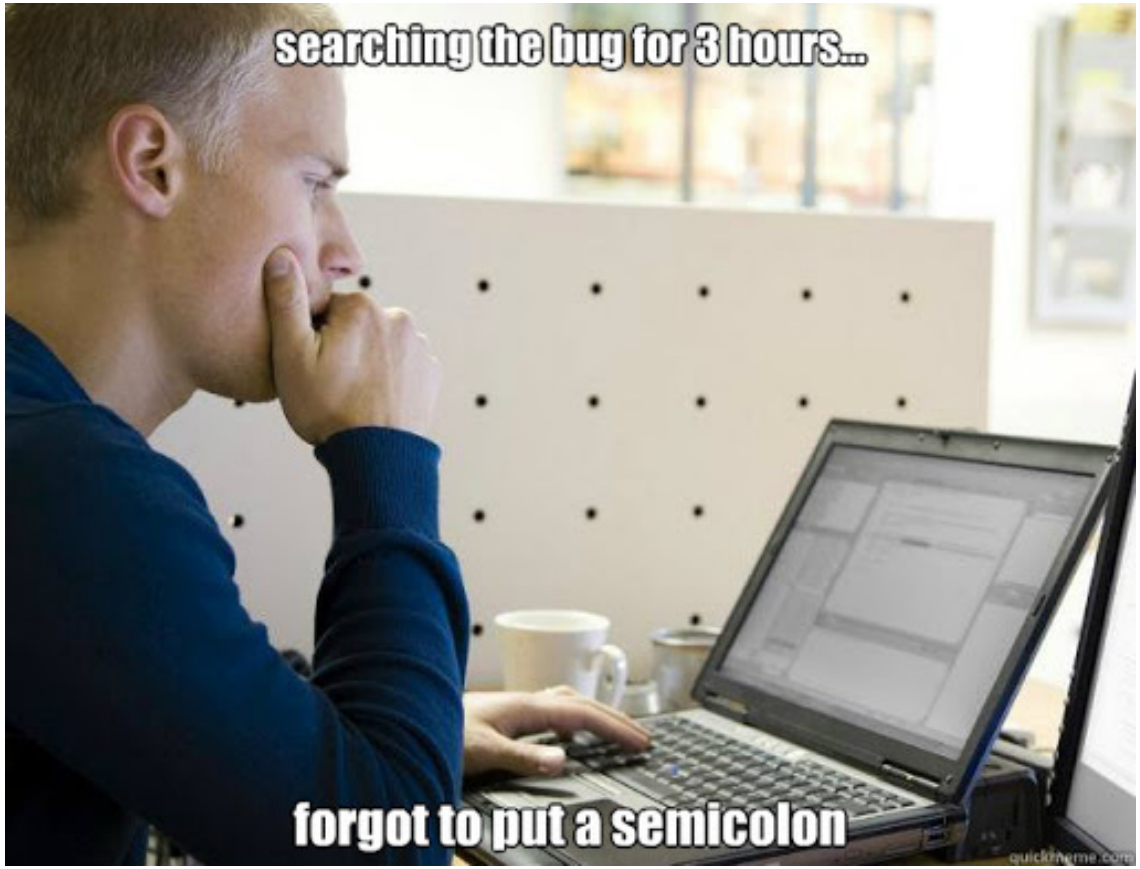
- An Rcpp indexing entails the creation of a new (proxy) object
- Therefore, R API is invoked
- Solution: use dedicated classes *RVector* and *RMatrix*
- **Unfortunately, StringVector and StringMatrix is not supported (because of more complex memory representation)**

Multithreaded environment (RcppParallel) IX

- 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**

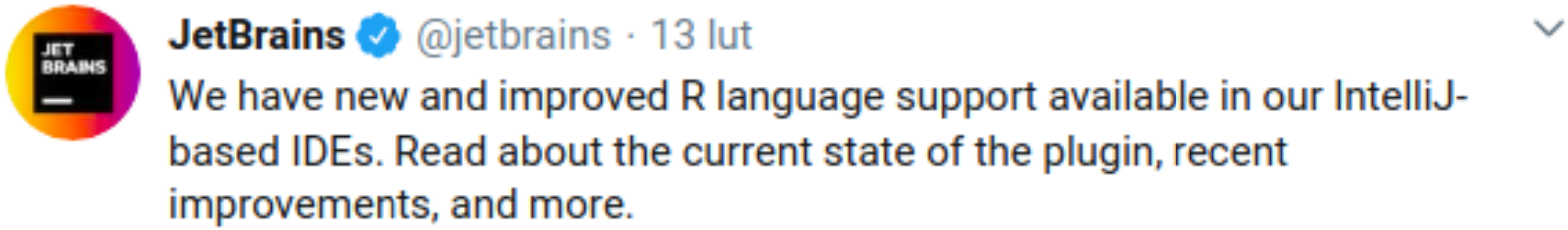
Looking for the perfect IDE I

Looking for the perfect IDE II



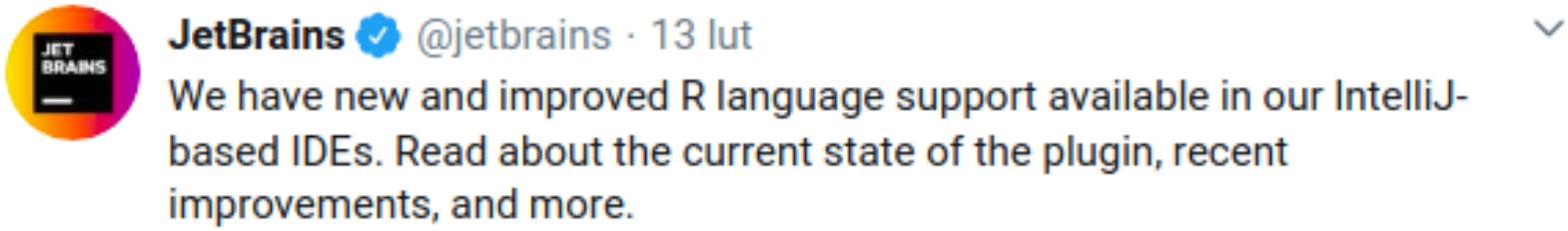
Looking for the perfect IDE III

Jetbrains team starts supporting R in their tools!



Looking for the perfect IDE IV

Jetbrains team starts supporting R in their tools!



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

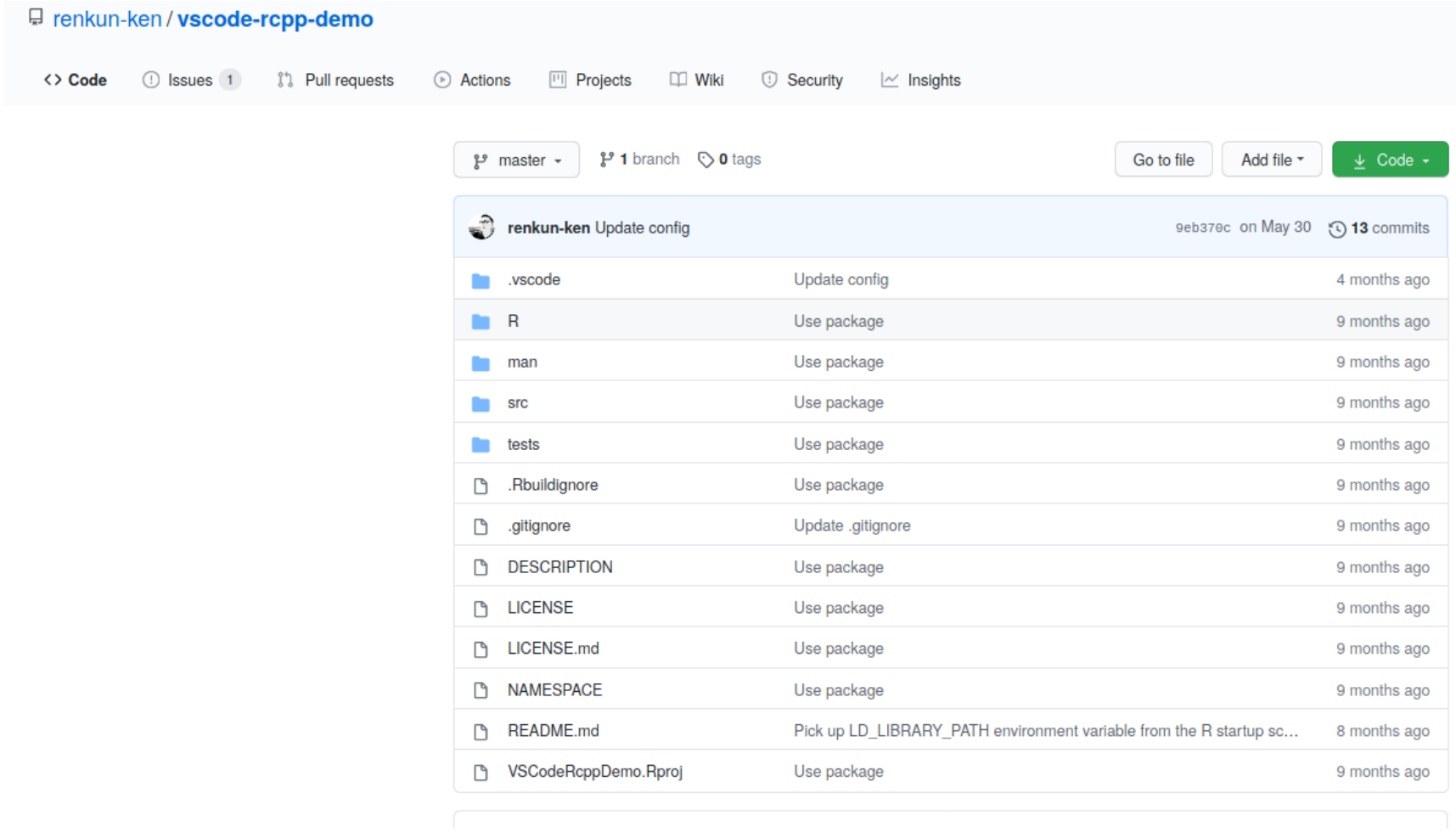
A solution:

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

Looking for the perfect IDE VI

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 VII



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**

Thank you for your attention!

Twitter: @slowikj5 Github: @slowikj

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