C++ for R Programmers

Dr. Dirk Eddelbuettel
edd@debian.org
dirk.eddelbuettel@R-Project.org

Invited Session: *What other languages should R users know about ?*

useR! 2012
Vanderbilt University
June 14, 2012


Here be
Dragons
ALBION
DRACONES

So "Why C++" ?

So "Why C++" ?

- Asking Google leads to 51,600,000 hits.

So "Why C++" ?

- Asking Google leads to 51,600,000 hits.
- No, I didn't read all of those.

So "Why C++" ?

- Asking Google leads to 51,600,000 hits.
- No, I didn't read all of those.
- Wikipedia starts with *C++ (pronounced "cee plus plus") is a statically typed, free-form, multi-paradigm, compiled, general-purpose, powerful programming language.*

So "Why C++" ?

- Asking Google leads to 51,600,000 hits.
- No, I didn't read all of those.
- Wikipedia starts with *C++ (pronounced "cee plus plus") is a statically typed, free-form, multi-paradigm, compiled, general-purpose, powerful programming language.*
- We could spend this session discussing just that sentence.

So "Why C++" ?

- Asking Google leads to 51,600,000 hits.
- No, I didn't read all of those.
- Wikipedia starts with *C++ (pronounced "cee plus plus") is a statically typed, free-form, multi-paradigm, compiled, general-purpose, powerful programming language.*
- We could spend this session discussing just that sentence.
- C++ is industrial-strength, widely-used, vendor-independent and *still evolving*.

So "Why C++" ?

- Asking Google leads to 51,600,000 hits.
- No, I didn't read all of those.
- Wikipedia starts with *C++ (pronounced "cee plus plus") is a statically typed, free-form, multi-paradigm, compiled, general-purpose, powerful programming language.*
- We could spend this session discussing just that sentence.
- C++ is industrial-strength, widely-used, vendor-independent and *still evolving*.
- In science and research, it is one of the most widely-used languages. If there is something you want to use or connect to, it probably has a C/C++ API.

So "Why C++" ?

- Asking Google leads to 51,600,000 hits.
- No, I didn't read all of those.
- Wikipedia starts with *C++ (pronounced "cee plus plus") is a statically typed, free-form, multi-paradigm, compiled, general-purpose, powerful programming language.*
- We could spend this session discussing just that sentence.
- C++ is industrial-strength, widely-used, vendor-independent and *still evolving*.
- In science and research, it is one of the most widely-used languages. If there is something you want to use or connect to, it probably has a C/C++ API.
- As a widely used language it also has good tool support (debuggers, [memory] profilers, code analysis).

A popular view on "What C++ is"

From Scott Meyers highly-regarded "Effective C++"

A popular view on "What C++ is"

From Scott Meyers highly-regarded "Effective C++"

Item 1: *"View C++ as a federation of languages"*

A popular view on "What C++ is"

From Scott Meyers highly-regarded "Effective C++"

Item 1: *"View C++ as a federation of languages"*

1. *C* provides a rich inheritance and interoperability as Unix, Windows, ... are all build on C.

A popular view on "What C++ is"

From Scott Meyers highly-regarded "Effective C++"

Item 1: *"View C++ as a federation of languages"*

1. *C* provides a rich inheritance and interoperability as Unix, Windows, ... are all build on C.
2. *Object-Oriented C++* just to provide endless discussions about exactly what OO is or should be (and R really helps here having three different ones to offer :-/).

A popular view on "What C++ is"

From Scott Meyers highly-regarded "Effective C++"

Item 1: *"View C++ as a federation of languages"*

1. *C* provides a rich inheritance and interoperability as Unix, Windows, ... are all build on C.
2. *Object-Oriented C++* just to provide endless discussions about exactly what OO is or should be (and R really helps here having three different ones to offer :-/).
3. *Templated C++* which is mighty powerful; template meta programming unequalled in other languages.

A popular view on "What C++ is"

From Scott Meyers highly-regarded "Effective C++"

Item 1: *"View C++ as a federation of languages"*

1. *C* provides a rich inheritance and interoperability as Unix, Windows, ... are all build on C.
2. *Object-Oriented C++* just to provide endless discussions about exactly what OO is or should be (and R really helps here having three different ones to offer :-/).
3. *Templated C++* which is mighty powerful; template meta programming unequalled in other languages.
4. *The STL* which is a specific template library which is powerful but has its own conventions.

A popular view on "What C++ is"

From Scott Meyers highly-regarded "Effective C++"

Item 1: *"View C++ as a federation of languages"*

1. *C* provides a rich inheritance and interoperability as Unix, Windows, ... are all build on C.
2. *Object-Oriented C++* just to provide endless discussions about exactly what OO is or should be (and R really helps here having three different ones to offer :-/).
3. *Templated C++* which is mighty powerful; template meta programming unequalled in other languages.
4. *The STL* which is a specific template library which is powerful but has its own conventions.

A popular view on "What C++ is"

From Scott Meyers highly-regarded "Effective C++"

Item 1: *"View C++ as a federation of languages"*

1. *C* provides a rich inheritance and interoperability as Unix, Windows, ... are all build on C.
2. *Object-Oriented C++* just to provide endless discussions about exactly what OO is or should be (and R really helps here having three different ones to offer :-/).
3. *Templated C++* which is mighty powerful; template meta programming unequalled in other languages.
4. *The STL* which is a specific template library which is powerful but has its own conventions.

And C++11 adds many more goodies that could be called a fifth language.

Barbells not bullets

- "Barbell" portfolios mean those comprised of both long and short duration bonds – as opposed to "bullet" portfolio concentrated at one (middle) duration.

Barbells not bullets

- "Barbell" portfolios mean those comprised of both long and short duration bonds – as opposed to "bullet" portfolio concentrated at one (middle) duration.
- I feel language choice is similar: It is rare to have one single solution for all problems. Python may be close; Julia may get there too.

Barbells not bullets

- "Barbell" portfolios mean those comprised of both long and short duration bonds – as opposed to "bullet" portfolio concentrated at one (middle) duration.
- I feel language choice is similar: It is rare to have one single solution for all problems. Python may be close; Julia may get there too.
- But I am a realist, and I have *never* been on a project or team that was single-language, single-solution.

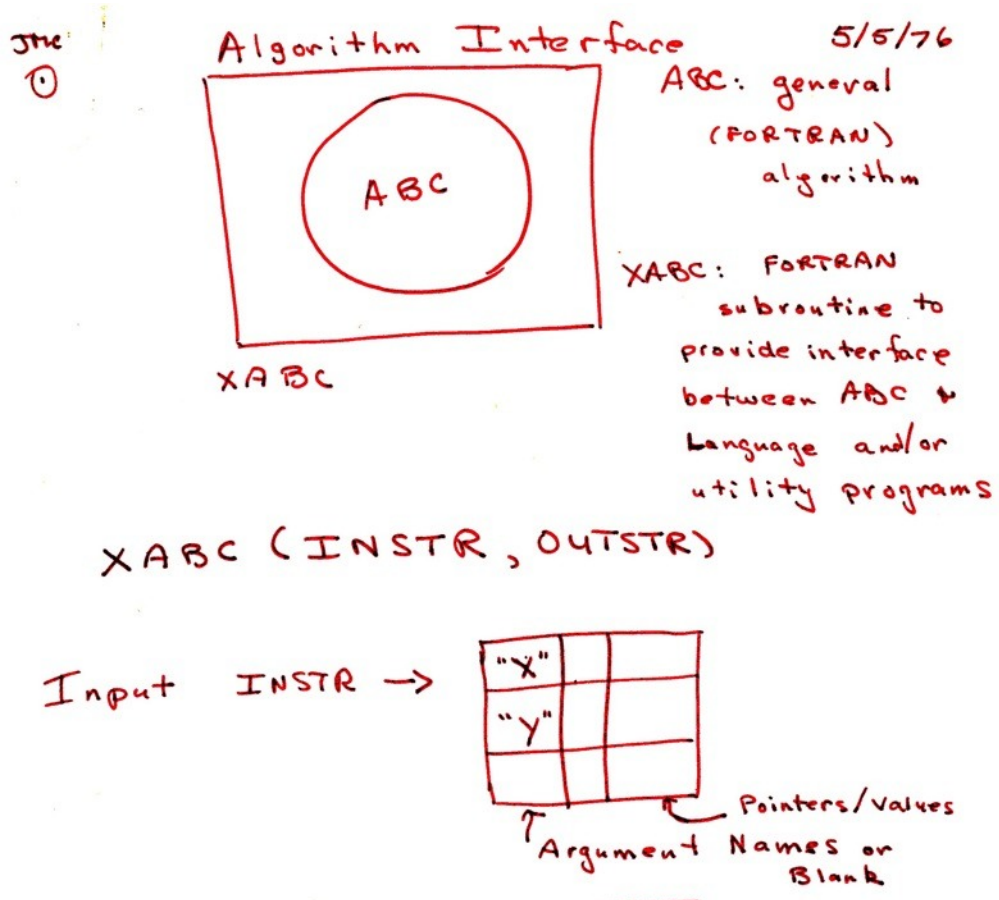
Barbells not bullets

- "Barbell" portfolios mean those comprised of both long and short duration bonds – as opposed to "bullet" portfolio concentrated at one (middle) duration.
- I feel language choice is similar: It is rare to have one single solution for all problems. Python may be close; Julia may get there too.
- But I am a realist, and I have *never* been on a project or team that was single-language, single-solution.
- In practice, people will always mix. So let's face this head-on and pick tools *which mix well*.

Barbells not bullets

- "Barbell" portfolios mean those comprised of both long and short duration bonds – as opposed to "bullet" portfolio concentrated at one (middle) duration.
- I feel language choice is similar: It is rare to have one single solution for all problems. Python may be close; Julia may get there too.
- But I am a realist, and I have *never* been on a project or team that was single-language, single-solution.
- In practice, people will always mix. So let's face this head-on and pick tools *which mix well*.
- It so happens that I think R and C++ mix well via Rcpp and RInside.

John Chambers agrees – and shares a "vision" from 1976



Source: John Chambers' talk at Google, 2010.

Sending R objects back and forth

Possible with R's API, easier with Rcpp

- Essentially, any R object is represented internally as a SEXP.

Sending R objects back and forth

Possible with R's API, easier with Rcpp

- Essentially, any R object is represented internally as a SEXP.
- The `.Call` interface lets you send SEXPs back and forth.

Sending R objects back and forth

Possible with R's API, easier with Rcpp

- Essentially, any R object is represented internally as a SEXP.
- The `.Call` interface lets you send SEXPs back and forth.
- SEXP can be nested just like R objects: lists of lists of ...

Sending R objects back and forth

Possible with R's API, easier with Rcpp

- Essentially, any R object is represented internally as a SEXP.
- The `.Call` interface lets you send SEXPs back and forth.
- SEXP can be nested just like R objects: lists of lists of ...
- Rcpp makes the interchange of R objects a little easier than the plain C API for R.

Sending R objects back and forth

Possible with R's API, easier with Rcpp

- Essentially, any R object is represented internally as a SEXP.
- The `.Call` interface lets you send SEXPs back and forth.
- SEXP can be nested just like R objects: lists of lists of ...
- Rcpp makes the interchange of R objects a little easier than the plain C API for R.
- "Empirically speaking", 68 CRAN packages (as of 3 June 2012) using Rcpp seem to agree.

Sending R objects back and forth

Possible with R's API, easier with Rcpp

- Essentially, any R object is represented internally as a SEXP.
- The `.Call` interface lets you send SEXPs back and forth.
- SEXP can be nested just like R objects: lists of lists of ...
- Rcpp makes the interchange of R objects a little easier than the plain C API for R.
- "Empirically speaking", 68 CRAN packages (as of 3 June 2012) using Rcpp seem to agree.
- This makes Rcpp the most widely used foreign-language interface package for R (as it overtook rJava recently).

Python

Python

Python

Trust me, Mogli.... Seriously?

Julia

Julia

My Julia – and until I find a matching language named after my other daughter Anna ...

Shootout and C++ / Rcpp based solutions

1. Gibbs Sampler example: I don't actually have much to add here which wasn't in the earlier blog posts.

Shootout and C++ / Rcpp based solutions

1. Gibbs Sampler example: I don't actually have much to add here which wasn't in the earlier blog posts.
2. Metropolis example: Whit Armstrong covered that in his recent rcppbugs package.

Shootout and C++ / Rcpp based solutions

1. Gibbs Sampler example: I don't actually have much to add here which wasn't in the earlier blog posts.
2. Metropolis example: Whit Armstrong covered that in his recent rcppbugs package.
3. Chris will cover both in the Python part.

C++11: The future

These are interesting times

- Arguably, new standard C++11 makes C++ a new language.

C++11: The future

These are interesting times

- Arguably, new standard C++11 makes C++ a new language.
- LLVM/Clang++ offers venues for static / dynamic code analysis and emergence of new tools.

C++11: The future

These are interesting times

- Arguably, new standard C++11 makes C++ a new language.
- LLVM/Clang++ offers venues for static / dynamic code analysis and emergence of new tools.
- Lots of very promising changes—while maintaining full compatibility and

C++11: The future

These are interesting times

- Arguably, new standard C++11 makes C++ a new language.
- LLVM/Clang++ offers venues for static / dynamic code analysis and emergence of new tools.
- Lots of very promising changes—while maintaining full compatibility and
  - still being the *fastest possible language*

C++11: The future

These are interesting times

- Arguably, new standard C++11 makes C++ a new language.
- LLVM/Clang++ offers venues for static / dynamic code analysis and emergence of new tools.
- Lots of very promising changes—while maintaining full compatibility and
  - still being the *fastest possible language*
  - while having *zero overhead*.

C++ Resources

- The book / website C++ Annotations by Brokken is very good, current and frequently updated.
- StackOverflow has a very good curated definite C++ book list
- Wikipedia is thorough as usual on C++ as well as C++11
- R and C++: Rcpp page, CRAN page and JSS paper
- Lastly, thanks to JJ, two pointers to two very recent talks:
  - Herb Sutter on (Not your father's) C++
  - Chander Carruth on Clang Defending C++ from Murphy's Million Monkeys