# Approaches to making R faster

Philip Barrett

University of Chicago

2 May 2014

#### Installation



If not, do so while I'm rambling on (next  $\sim 5$  minutes).

Extra: you may want to install microbenchmark:

install.packages('microbenchmark')
library(microbenchmark)

## The limitations of R

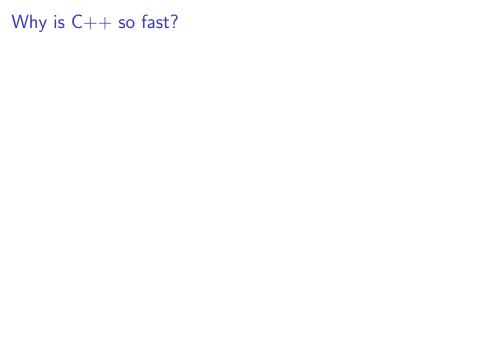
- ▶ R is great in many ways: flexible, easy to write, mature.
- ▶ But it isn't the fastest.
- ▶ So what if your problem has large computational demands?
- ► Two approaches:
  - 1. Parallelize
  - 2. Integrate other languages
- ▶ Today, focus on 2 (esp. C++), but discuss 1 briefly.

#### **Parallelism**

- Pros:
  - Easy: implementation and learning
  - ▶ Time cost reduced by factor of  $\sim$  50 on small(ish) systems
  - No (software) cost of extra nodes on v large systems
- Cons:
  - At best,  $t \propto 1/N$
  - Communication time
  - Sequential operations
- Resources:
  - "Parallel R", McCallum & Weston (O'Reilly)
  - Steven Mohr: smohr@uchicago.edu

## C++

- Pros:
  - Fast (esp. for loops & Linear algebra)
  - ▶ Universal: The "Latin" of CS
  - Mature
- Cons:
  - Harder to learn
  - Slower to write
  - Compilers
- Our approach:
  - Use the Rcpp and RcppArmadillo packages
  - ▶ Integrates beautifully to R.
  - ▶ Call functions in R, but written in C++.
  - No painful set up or compilation headaches.



# Why is C++ so fast?

Blah defining types blah machine code blah compilers blah but basically, it just is.

#### The motherlode

Combining C++ and paralellism can give you *massive* speed gains.

- Strategy:
  - 1. Write R code that parallelizes a repeated operation
  - 2. Write the code for that operation in C++
- Conservative numbers:
  - Parallelism gives a ×50 speed gain
  - ightharpoonup C++ operation is  $\sim 100$  times faster
  - ▶ Total speed gain  $\sim 5000$
- Excluding cluster initialization, this means:
  - 1. Before: 1 hour, after:  $\sim 0.7$  seconds
  - 2. Before: 1 day, after:  $\sim$  17 seconds
  - 3. Before: 1 week, after:  $\sim$  2 mins
  - 4. Before: 1 year, after:  $\sim 1.5$  hours

In practice, speed gains from C++ part be much greater.

NB: Parallelizing in C++ alone is not fun.

## Outline

- Example
- ▶ Basics of coding in C++
  - 1. Hello world
  - 2. Types
  - 3. Returns
  - 4. If..else
  - 5. Loops
  - 6. R-style vectors (with zero-based counting!)
  - 7. R-style matrices and lists
- ► Matrix algebra with RcppArmadillo.
  - 1. Vectors and matrices
  - 2. Simple operations: Addition, multiplication, matrix metadata
  - 3. Harder operations: Accessing submatrices, inversion
- ▶ Rewrite John Eric's MCMC example