Extending R with C++

Motivation, Examples, and Context

Dirk Eddelbuettel

Invited Keynote, ICORS-LACSC 2019
Escuela Superior Politécnica del Litoral (ESPOL)
Guayaquil, Ecuador
May 30, 2019

Who Am I?

Brief Bio

- Finance Quantitative Research Professional for 20+ years
- (Adjunct) Clinical Professor at U of Illinois for 2 years
- Open Source for around 25 years
  - Debian developer (~ 170 packages)
  - R package author (~ 50 packages)
  - Rcpp, Rocker, ...
- R and Statistics
  - JSS Associate Editor
  - R Foundation Board member
- PhD, MA Econometrics; MSc Ind.Eng. (Comp.Sci./OR)

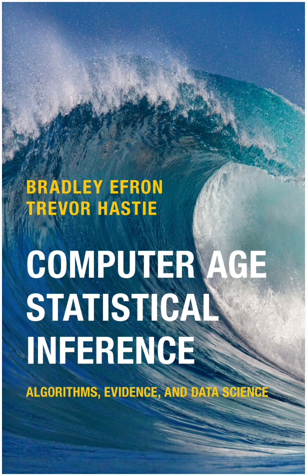
Overview

Outline

Extending R with C++

- Why R ?
- Why Extending R ?
- Why C++ and Rcpp ?
- (Briefly) How ?
- Context
- Outlook

Why R: View from Academia



Almost all topics in twenty-first-century statistics are now computer-dependent [...]

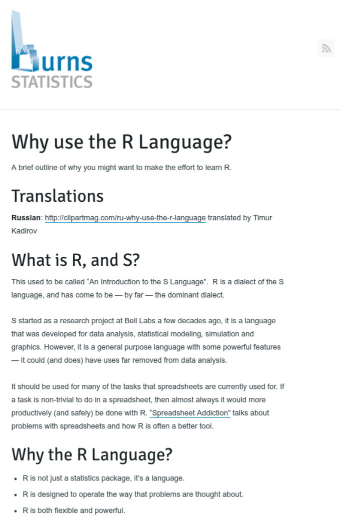
Here and in all our examples we are employing the language R, itself one of the key developments in computer-based statistical methodology.

Efron and Hastie, 2016
pages xv and 6 (footnote 3)

Computational Statistics in Practice

- Statistics is now computational (Efron & Hastie, 2016)
- Within (computational) statistics, reigning tool is R
- Given R, Rcpp key for two angles:
  - *Performance* always matters, ease of use a sweetspot
  - *"Extending R"* (Chambers, 2016)

Why R: View from Practitioners



Burns STATISTICS

Why use the R Language?

A brief outline of why you might want to make the effort to learn R.

Translations

Russian: http://clipartmag.com/ru-why-use-the-r-language translated by Timur Kadirov

What is R, and S?

This used to be called "An Introduction to the S Language". R is a dialect of the S language, and has come to be — by far — the dominant dialect.

S started as a research project at Bell Labs a few decades ago, it is a language that was developed for data analysis, statistical modeling, simulation and graphics. However, it is a general purpose language with some powerful features — it could (and does) have uses far removed from data analysis.

It should be used for many of the tasks that spreadsheets are currently used for. If a task is non-trivial to do in a spreadsheet, then almost always it would more productively (and safely) be done with R. "Spreadsheet Addiction" talks about problems with spreadsheets and how R is often a better tool.

Why the R Language?

- R is not just a statistics package, it's a language.
- R is designed to operate the way that problems are thought about.
- R is both flexible and powerful.

Why the R Language?

Screen shot on the left part of short essay at Burns-Stat

His site has more truly excellent (and free) writings.

The (much longer) R Inferno (free pdf, also paperback) is highly recommended.

Why R? Pat Burn's View

Why the R Language?

- R is not just a statistics package, it's a language.
- R is designed to operate the way that problems are thought about.
- R is both flexible and powerful.

Source: https://www.burns-stat.com/documents/tutorials/why-use-the-r-language/

Why R? Pat Burn's View

Why R for data analysis?

R is not the only language that can be used for data analysis. Why R rather than another? Here is a list:

- interactive language
- data structures
- graphics
- missing values
- functions as first class objects
- packages
- community

Source: https://www.burns-stat.com/documents/tutorials/why-use-the-r-language/

Why R: Programming with Data

R as an Extensible Environment

- As R users we know that R can
  - ingest data in many formats from many sources
  - aggregate, slice, dice, summarize, ...
  - visualize in many forms, ...
  - model in just about any way
  - report in many useful and scriptable forms
- It has become central for programming with data
- Sometimes we want to extend it further than R code goes

R as central point

R as central point

From any one of

- csv
- txt
- xlsx
- xml, json, ...
- web scraping, ...
- hdf5, netcdf, ...
- sas, stata, spss, ...
- various SQL + NOSQL DBs
- various binary protocols

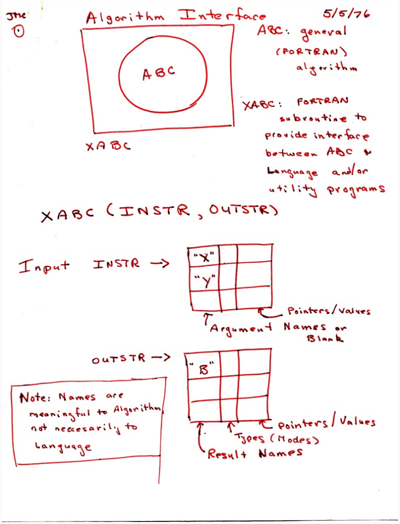
via

into any one of

- txt
- html
- latex and pdf
- html and js
- word
- shiny
- most graphics formats
- other dashboards
- web frontends

Why R: Historical Perspective

R as 'The Interface'



A design sketch called 'The Interface'

AT&T Research lab meeting notes

Describes an outer 'user interface' layer to core Fortran algorithms

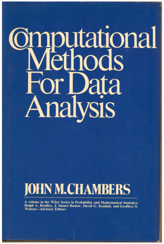
Key idea of abstracting away inner details giving higher-level more accessible view for user / analyst

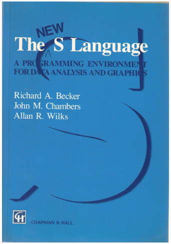
Lead to "The Interface"

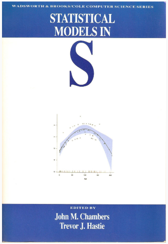
Which became S which lead to R

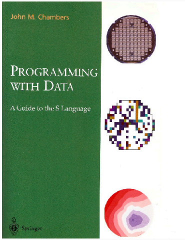
Source: John Chambers, personal communication

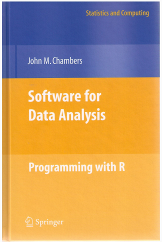
Why R? : *Programming with Data* from 1977 to 2016

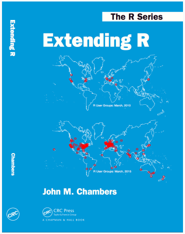






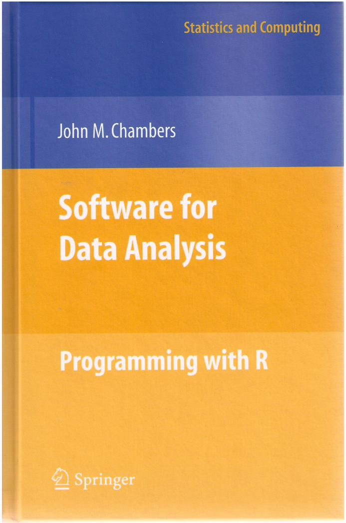






Thanks to John Chambers for high-resolution cover images. The publication years are, respectively, 1977, 1988, 1992, 1998, 2008 and 2016.

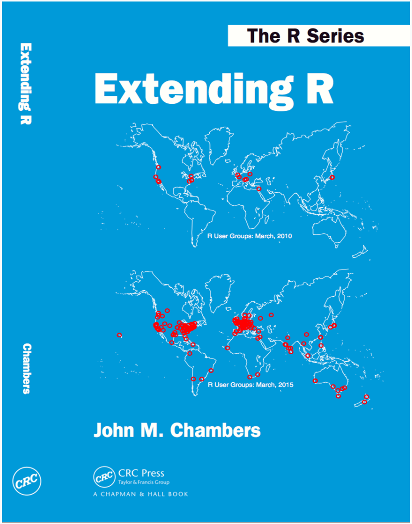
CHAMBERS (2008)



Software For Data Analysis

Chapters 10 and 11 devoted to *Interfaces I: C and Fortran* and *Interfaces II: Other Systems*.

Chambers (2016)

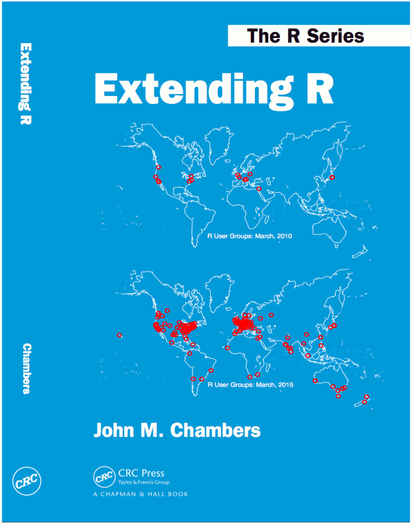


Extending R

Object: Everything that exists in R is an object

Function: Everything happens in R is a function call

Interface: Interfaces to other software are part of R



Extending R, Chapter 4

The fundamental lesson about programming in the large is that requires a correspondingly broad and flexible response. In particular, no single language or software system os likely to be ideal for all aspects. Interfacing multiple systems is the essence. Part IV explores the design of of interfaces from R.

Why C++ and Rcpp

R and C/C++

A good fit, it turns out

- A good part of R is written in C (besides R and Fortran code)
- The principle interface to external code is a function `.Call()`
- It takes one or more of the high-level data structures R uses
- ... and returns one. Formally:

```
SEXP .Call(SEXP a, SEXP b, ...)
```

A good fit, it turns out (cont.)

- An SEXP (or S-Expression Pointer) is used for *everything*
- (An older C trick approximating object-oriented programming)
- We can ignore the details but retain that
  - everything in R is a SEXP
  - the SEXP is self-describing
  - can matrix, vector, list, function, ...
  - 27 types in total
- The key thing for Rcpp is that via C++ features we can map
  - each of the (limited number of) SEXP types
  - to a specific C++ class representing that type
  - and the conversion is automated back and forth

R and C/C++

Other good reasons

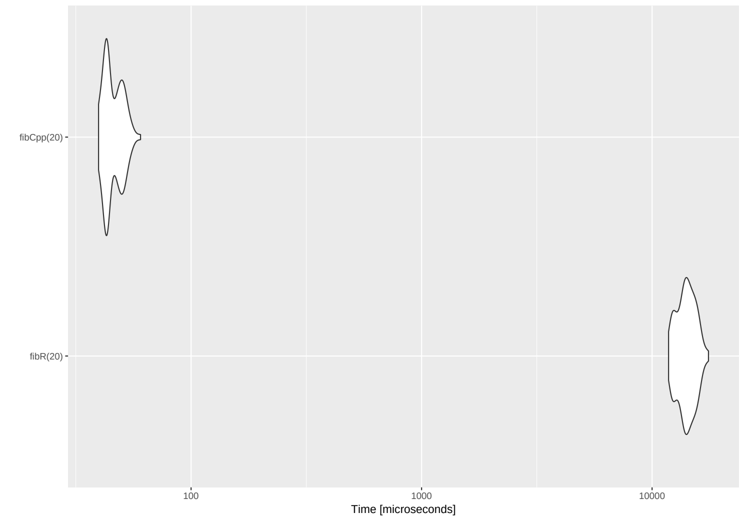
- It is *fast* – compiled C++ is hard to beat in other languages
    - (That said, you can *of course* write bad and slow code….)
- It is *very general* and widely used
    - *many libraries*
    - many tools
- It is fairly universal:
    - just about anything will have C interface so C++ can play
    - just about any platform / OS will have it

Key Features

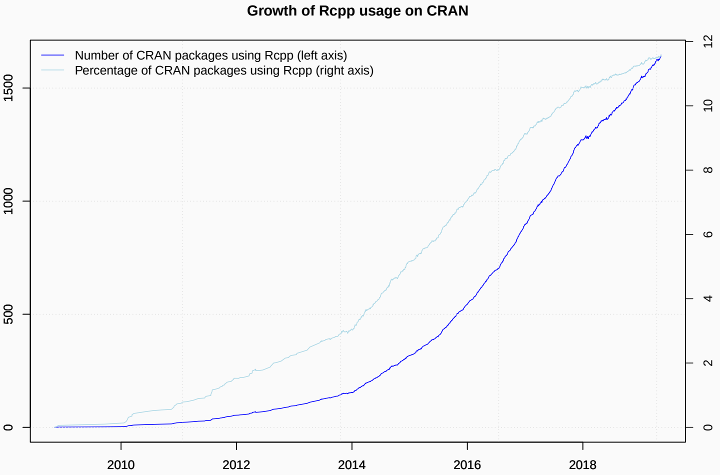
- (Fairly) Easy to learn as it really does not have to be that complicated – there are numerous examples
- Easy to use as it avoids build and OS system complexities thanks to the R infrastrucure
- Expressive as it allows for *vectorised* C++ using *Rcpp Sugar*
- Seamless access to all R objects: vector, matrix, list, S3/S4/RefClass, Environment, Function, ...
- Speed gains for a variety of tasks Rcpp excels precisely where R struggles: loops, function calls, ...
- Extensions greatly facilitates access to external libraries directly or via eg *Rcpp modules*

Rcpp Speed Illustration

Benchmark on Fibonacci(20) between C++ and R – note the log scale!



Growth



Data current as of May 12, 2019.

Users on Core Repositories

Rcpp is currently used by

- 1655 CRAN packages
- 176 BioConductor packages (with 38 added since last year)
- an unknown (but “large”) number of GitHub projects

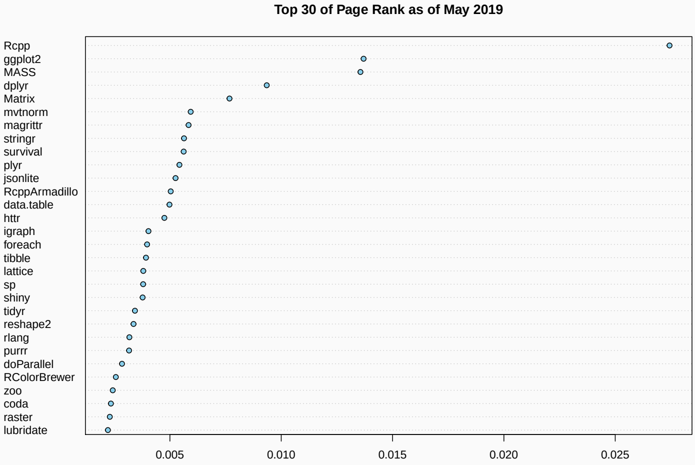
PageRank

```r
suppressMessages(library(utils))
library(pagerank)   # cf github.com/andrie/pagerank

cran <- "http://cloud.r-project.org"
pr <- compute_pagerank(cran)
round(100*pr[1:5], 3)
```

```
##    Rcpp ggplot2    MASS   dplyr  Matrix
##   2.744   1.370   1.356   0.935   0.768
```

Pagerank


Top 30 of Page Rank as of May 2019
Rcpp
ggplot2
MASS
dplyr
Matrix
mvtnorm
magrittr
stringr
survival
plyr
jsonlite
RcppArmadillo
data.table
httr
igraph
foreach
tibble
lattice
sp
shiny
tidyr
reshape2
rlang
purrr
doParallel
RColorBrewer
zoo
coda
raster
lubridate
0.005
0.010
0.015
0.020
0.025

Percentage of Compiled Packages

```
db <- tools::CRAN_package_db()   # added in R 3.4.0
## rows: number of pkgs, cols: different attributes
nTot <- nrow(db)
## all direct Rcpp reverse depends, ie packages using Rcpp
nRcpp <- length(tools::dependsOnPkgs("Rcpp", recursive=FALSE,
                                      installed=db))
nCompiled <- table(db[, "NeedsCompilation"])[["yes"]]
propRcpp <- nRcpp / nCompiled * 100
data.frame(tot=nTot, totRcpp = nRcpp, totCompiled = nCompiled,
           RcppPctOfCompiled = propRcpp)
```

```
##     tot totRcpp totCompiled RcppPctOfCompiled
## 1 14312    1655        3591          46.08744
```

How: Brief Rcpp Intro

First Steps: evalCpp()

```r
library(Rcpp)
evalCpp("2 + 2")
```

```
## [1] 4
```

This function can validate your installation.

It takes the supplied expression, wraps enough code around it to make a compilable function, compiles, links and loads it – to evaluate the C++ expression.

Here we skip all details about Rcpp installations. It just works *e.g.* on the (free) RStudio Cloud and in most normal system – see the documentation for more. As always, Windows may be hardest as you may have to install another R toolchain: `Rtools`.

First Steps: cppFunction()

```r
library(Rcpp)
cppFunction("double fib(double n) { \
   if (n < 2) return(n); \
   return(fib(n-1) + fib(n-2)); \
}")
fib(30)
```

```
## [1] 832040
```

Creates R-callable function from a C++ function.

Finds function identifier in supplied string, creates R function of same name.

Useful for quick tests.

Can use additional headers and library (see documentation).

`sourceCpp()`

- ‘sources’ a file and compiles, links, loads
- file can contain multiple functions
- functions that are ‘tagged’ with `// [[Rcpp::export]]` become callable
- can contain non-exported helper functions
- use:

```
sourceCpp("someFile.cpp")  # with or without path
```

First Steps: sourceCpp()

```cpp
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
NumericVector timesTwo(NumericVector x) {
  return x * 2;
}

/*** R
timesTwo(42)
*/
```

This is a shortened (comments-removed) version of the file currently included when you say ‘File -> New File -> C++’ in RStudio.

First Steps: sourceCpp()

```cpp
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
NumericVector timesTwo(NumericVector x) {
  return x * 2;
}

/*** R
timesTwo(42)
*/
```

Key features:

Rcpp header and `namespace`

One exported function `timesTwo()`

An automatically executed (!!) R call for tests and demos

Try it!

First Steps: sourceCpp()

Quick Demo

```
Rcpp::sourceCpp("code/timestwo.cpp")  # runs demo too
```

```
##
## > timesTwo(42)
## [1] 84
```

```
timesTwo(c(5,10,20)) # vectorized like R
```

```
## [1] 10 20 40
```

Example: Column Sums

```cpp
#include <Rcpp.h>

// [[Rcpp::export]]
Rcpp::NumericVector colSums(Rcpp::NumericMatrix mat) {
  size_t cols = mat.cols();
  Rcpp::NumericVector res(cols);
  for (size_t i=0; i<cols; i++) {
    res[i] = sum(mat.column(i));
  }
  return(res);
}
```

Column Sums

Key Elements

- `NumericMatrix` and `NumericVector` go-to types for matrix and vector operations on floating point variables
- We prefix with `Rcpp::` to make the namespace explicit
- Accessor functions `.rows()` and `.cols()` for dimensions
- Result vector allocated based on number of columns column
- Function `column(i)` extracts a column, gets a vector, and `sum()` operates on it
- That last `sum()` was internally vectorised, no need to loop over all elements

Column Sums

```r
Rcpp::sourceCpp("code/colSums.cpp")


# test it
colSums(matrix(1:16, 4, 4))
```

```
## [1] 10 26 42 58
```

```r
# base R for comparison
apply(matrix(1:16, 4, 4), 2, sum)
```

```
## [1] 10 26 42 58
```

First Steps: Packages

Package are *the* standard unit of R code organization.

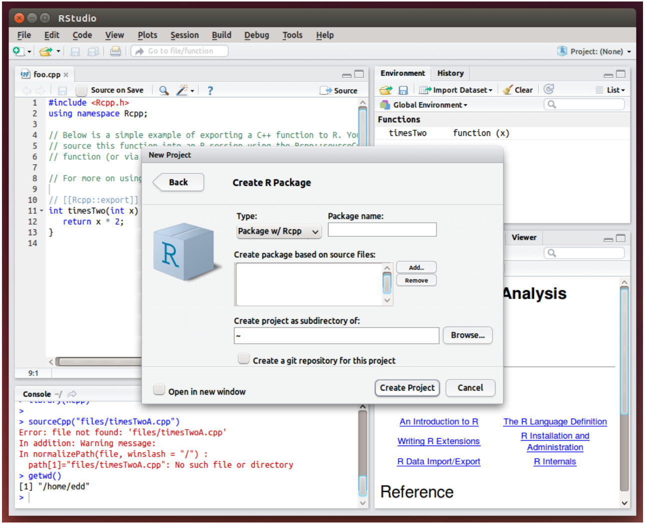
Creating packages with Rcpp is easy; an empty one to work from can be created by `Rcpp.package.skeleton()`

The vignette Rcpp-packages has fuller details.

As of May 2019, there are 1655 CRAN and 176 BioConductor packages which use Rcpp all offering working, tested, and reviewed examples.

Packages and Rcpp

Best way to organize R code with Rcpp is via a package:



Packages and Rcpp

`Rcpp.package.skeleton()` and its derivatives. e.g. `RcppArmadillo.package.skeleton()` create working packages.

```
// another simple example: outer product of a vector,
// returning a matrix
//
// [[Rcpp::export]]
arma::mat rcpparma_outerproduct(const arma::colvec & x) {
    arma::mat m = x * x.t();
    return m;
}

// and the inner product returns a scalar
//
// [[Rcpp::export]]
double rcpparma_innerproduct(const arma::colvec & x) {
    double v = arma::as_scalar(x.t() * x);
    return v;
}
```

Two (or three) ways to link to external libraries

- *Full copies:* Do what several packages (*e.g.* RcppMLPACK (v1), RVowpalWabbit) do and embed a full copy; larger build time, harder to update, self-contained
- *With linking of libraries:* Do what RcppGSL or RcppMLPACK (v2) do and use hooks in the package startup to store compiler and linker flags which are passed to environment variables
- *With C++ template headers only:* Do what RcppArmadillo and other do and just point to the headers
- More details in extra vignettes.

Key Extension Package RcppArmadillo

Armadillo
C++ library for linear algebra & scientific computing

About | Questions | License | Documentation | Speed | Contact | Download

- Armadillo is a high quality linear algebra library (matrix maths) for the C++ language, aiming towards a good balance between speed and ease of use
- Provides high-level syntax and functionality deliberately similar to Matlab
- Useful for algorithm development directly in C++, or quick conversion of research code into production environments (eg. software & hardware products)
- Provides efficient classes for vectors, matrices and cubes (1st, 2nd and 3rd order tensors); dense and sparse matrices are supported
- Integer, floating point and complex numbers are supported
- Various matrix decompositions are provided through integration with LAPACK, or one of its high performance drop-in replacements (eg. multi-threaded Intel MKL, or OpenBLAS)
- A sophisticated expression evaluator (based on template meta-programming) automatically combines several operations to increase speed and efficiency
- Can automatically use OpenMP multi-threading (parallelisation) to speed up computationally expensive operations
- Available under a permissive license, useful for both open-source and proprietary (closed-source) software
- Can be used for machine learning, pattern recognition, computer vision, signal processing, bioinformatics, statistics, finance, etc
- **download latest version** | **GitLab repo** | **browse documentation**

Supported by:

Source: `http://arma.sf.net`

Armadillo

What is Armadillo?

- Armadillo is a C++ linear algebra library (matrix maths) aiming towards a good balance between speed and ease of use.
- The syntax is deliberately similar to Matlab.
- Integer, floating point and complex numbers are supported.
- A delayed evaluation approach is employed (at compile-time) to combine several operations into one and reduce (or eliminate) the need for temporaries.
- Useful for conversion of research code into production environments, or if C++ has been decided as the language of choice, due to speed and/or integration capabilities.

Source: `http://arma.sf.net`

Armadillo Highlights

Key Points

- Provides integer, floating point and complex vectors, matrices, cubes and fields with all the common operations.
- Very good documentation and examples
  - website,
  - technical report (Sanderson, 2010),
  - CSDA paper (Sanderson and Eddelbuettel, 2014),
  - JOSS paper (Sanderson and Curtin, 2016),
  - ICMS paper (Sanderson and Curtin, 2018).
- Modern code, extending from earlier matrix libraries.
- Responsive and active maintainer, frequent updates.
- Used eg by MLPACK, see Curtin et al (JMLR 2013, JOSS 2018).

RcppArmadillo Highlights

Key Points

- Template-only builds—no linking, and available whereever R and a compiler work (but Rcpp is needed)
- Easy to use, just add `LinkingTo: RcppArmadillo, Rcpp` to `DESCRIPTION` (i.e. no added cost beyond Rcpp)
- Really easy from R via Rcpp and automatic converters
- Frequently updated, widely used – now over 600 CRAN packages

Example: Column Sums

```cpp
#include <RcppArmadillo.h>

// [[Rcpp::depends(RcppArmadillo)]]

// [[Rcpp::export]]
arma::rowvec colSums(arma::mat mat) {
    size_t cols = mat.n_cols;
    arma::rowvec res(cols);

    for (size_t i=0; i<cols; i++) {
        res[i] = sum(mat.col(i));
    }

    return(res);
}
```

Example: Column Sums

Key Features

- The `[[Rcpp::depends(RcppArmadillo)]]` tag tells R to tell `g++` about the need for Armadillo headers – needed for compilation
- Dimension accessor via member variables `n_rows` and `n_cols`; not function calls
- We return a `rowvec`; default `vec` is alias for `colvec`
- Column accessor is just `col(i)` here
- This is a normal example of how similar features may have slightly different names across libraries

Context / Examples

Example: Wordcloud

Background

- The wordcloud package by Ian Fellows computes wordclouds
- Initial release(s) had (only) an R version
- Basic algorithm:
  - 'trial and error' of placing words in a grid on page
  - constraint is to not overlap …
- Loops and trial and error are not fast
  - so an Rcpp version was added
- One of few packages that does 'before' and 'after'

Example: Wordcloud

Using Moby Dick with `min.freq=5` is a large enough task:

```
suppressMessages({library(wordcloud); library(tm)})
moby <- readLines("http://www.gutenberg.org/files/2701/2701-0.txt")
system.time(wordcloud(moby, min.freq = 5, random.order=FALSE,
                      use.r.layout = TRUE))
```

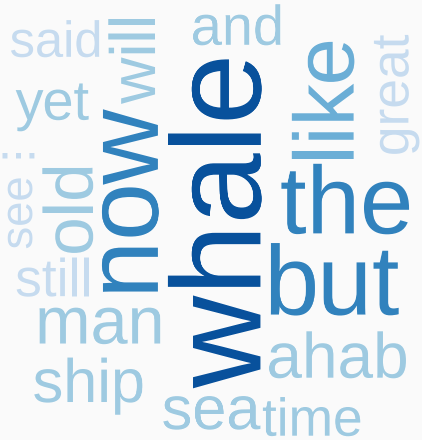
```
##    user  system elapsed
## 393.352   0.214 393.876
```

```
system.time(wordcloud(moby, min.freq = 5, random.order=FALSE,
                      use.r.layout = FALSE))
```

```
##    user  system elapsed
##  87.377   0.004  87.396
```

Decent speedup – but not as dramatic as *Fibonacci* above.

```r
wordcloud(moby, min.freq = 250, colors=brewer.pal(6,"Blues"),
          random.order=FALSE)
```



Example: winsorize

Background

- Excellent CRAN package robustHD by Andreas Alfons
    - Offers a variety of routines ...
    - .. but also pulls in a number of dependencies
- For a small (private) project I needed a subset of robustHD
- So we created winsorize (GitHub-only)
- (Partial) code examples follow showing
    - simple and clean C++
    - taking advantage of (Rcpp)Armadillo

WINSORIZE CODE (1 OF 2)

```
double corPearson(const vec& x, const vec& y) {
  // arma function cor() always returns matrix
  return as_scalar(cor(x, y));
}

double winsorize(const double& x, const double& cm,
                 const double& cp) {
  if(x < cm) {
    return cm;
  } else if(x > cp) {
    return cp;
  } else return x;
}
```

WINSORIZE CODE (1 OF 2)

```
double corHuberUni(const vec& x, const vec& y,
                   const double& c) {
  // negative winsorization constant
  const double cm = -c;
  const uword n = x.n_elem;
  vec wx(n), wy(n);
  for(uword i = 0; i < n; i++) {
    wx(i) = winsorize(x(i), cm, c);
    wy(i) = winsorize(y(i), cm, c);
  }
  // call barebones function for Pearson correlation
  // with winsorized data
  return corPearson(wx, wy);
}
```

A Related Approach

R Interface to Python

The **reticulate** package provides a comprehensive set of tools for interopability between Python and R. The package includes facilites for:

- Translation between R and Python objects (for example, between R and Pandas data frames, or between R matrices and NumPy arrays).
- Calling Python from R in a variety of ways including R Markdown, sourcing Python scripts, importing Python modules, and using Python interactively within an R session.
- Flexible binding to different versions of Python including virtual environments and Conda environments.

Reticulate embeds a Python session within your R session, enabling seamless, high-performance interoperability. If you are an R developer that uses Python for some of your work or a member of data science team that uses both languages, reticulate can dramatically streamline your workflow!

Source: `https://rstudio.github.io/reticulate/`

Generic Python wrapping

reticulate

- Written to support `tensorflow` and `keras`
- Already used by several packages including
  - `greta`: think stan or bugs, but on tensorflow
  - `spacyr`: accesses the spaCy NLP engine
  - `h2o4gpu`: access to h2o.ai GPU-based ML solvers
- Also used by `XRPython`
- Uses Rcpp

The RcppCNPy package lets us load and save NumPy files (by wrapping the C library cnpy).

```r
library(RcppCNPy)
mat <- npyLoad("fmat.npy")
vec <- npyLoad("fvec.npy")

mat2 <- npyLoad("fmat.npy.gz")
```

Generic Python wrapping

But `reticulate` lets us load and save NumPy files directly!

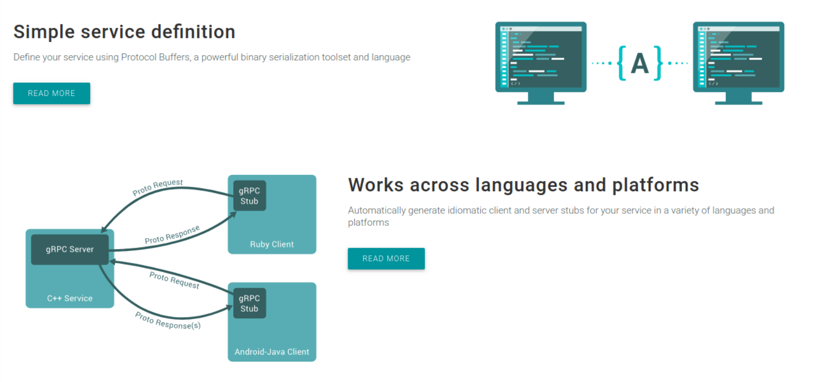
```r
library(reticulate)
np <- import("numpy")
mat <- np$load("fmat.npy"))
vec <- np$load("fvec.npy"))

## compressed data: import gzip
gz <- import("gzip")
## use it to create handle to uncompressed file
mat2 <- np$load(gz$GzipFile("fmat.npy.gz","r"))
```

See the vignettes in the RcppCNPy package for more.

Other Approaches

GRPC



Source: https://grpc.io

Different Approach

- define an *interface* (as Protocol Buffer)
- have code generated for both *server* and *client* side
- across OSs: Linux, Windows, Android, iOS, …
- across languages: C++, Python, Go, Javascript, Ruby, C#, PHP, …

Arrow

Apache Arrow

A cross-language development platform for in-memory data

Join Mailing List | Install (0.8.0 Release - 18 December 2017)

See Latest News

Apache Arrow is a cross-language development platform for in-memory data. It specifies a standardized language-independent columnar memory format for flat and hierarchical data, organized for efficient analytic operations on modern hardware. It also provides computational libraries and zero-copy streaming messaging and interprocess communication. Languages currently supported include C, C++, Java, JavaScript, Python, and Ruby.

Fast

Apache Arrow™ enables execution engines to take advantage of the latest SIMD (Single input multiple data) operations included in modern processors, for native vectorized optimization of analytical data processing. Columnar layout is optimized for data locality for better performance on modern hardware like CPUs and GPUs.

The Arrow memory format supports **zero-copy reads** for lightning-fast data access without serialization overhead.

Flexible

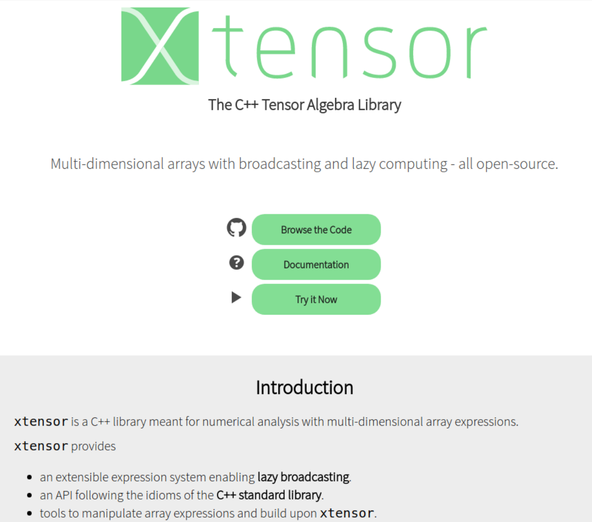
Arrow acts as a new high-performance interface between various systems. It is also focused on supporting a wide variety of industry-standard programming languages. Java, C, C++, Python, Ruby, and JavaScript implementations are in progress and more languages are welcome.

Standard

Apache Arrow is backed by key developers of 13 major open source projects, including Calcite, Cassandra, Drill, Hadoop, HBase, Ibis, Impala, Kudu, Pandas, Parquet, Phoenix, Spark, and Storm making it the de-facto standard for columnar in-memory analytics.

Learn more about projects that are Powered By Apache Arrow

Source: `https://arrow.apache.org/`



xtensor

The C++ Tensor Algebra Library

Multi-dimensional arrays with broadcasting and lazy computing - all open-source.

Browse the Code

Documentation

Try it Now

Introduction

`xtensor` is a C++ library meant for numerical analysis with multi-dimensional array expressions.

`xtensor` provides

- an extensible expression system enabling **lazy broadcasting**.
- an API following the idioms of the **C++ standard library**.
- tools to manipulate array expressions and build upon `xtensor`.

Source: `http://quantstack.net/xtensor`

SUMMARY

Key Points

- Statistics is now a computational discipline
- Within Statistics, the R language is the *lingua franca*
- Rcpp permits *extending R* in (relatively) easy ways
- Other approaches exist, some build upon Rcpp
- *Interfaces to other software are part of R.*

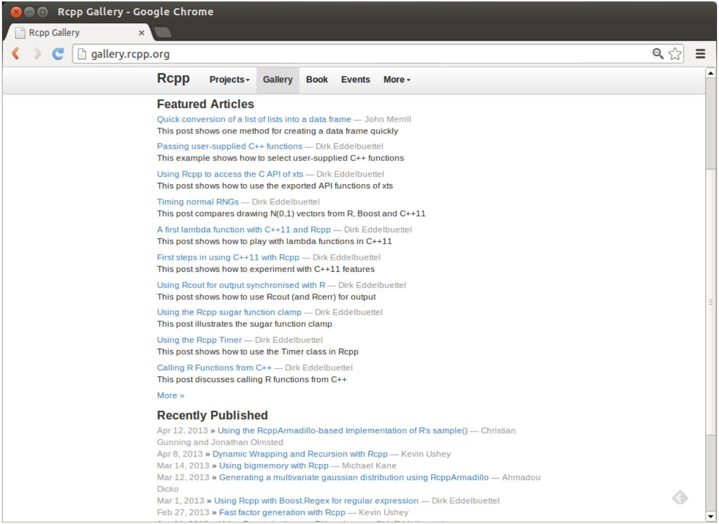
Appendix: More on Rcpp

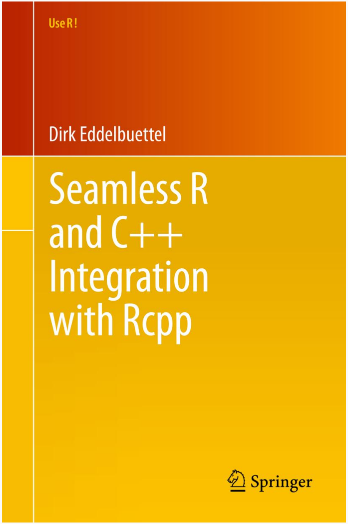
Resources

Documentation and Examples

- The package comes with nine pdf vignettes, and help pages.
- The introductory vignettes are now published (Rcpp and RcppEigen in *J Stat Software*, RcppArmadillo in *Comp Stat & Data Anlys*, Rcpp again in *TAS*)
- The rcpp-devel list is *the* recommended resource, generally very helpful, and fairly low volume.
- StackOverflow has a fair number of posts too.
- And a number of blog posts introduce/discuss features.

Rcpp Gallery





On sale since June 2013.

Thank you!

slides http://dirk.eddelbuettel.com/presentations/

web http://dirk.eddelbuettel.com/

mail dirk@eddelbuettel.com

github @eddelbuettel

twitter @eddelbuettel