Extending R with C++

A Brief Introduction to Rcpp

Dirk Eddelbuettel

14 January 2017

Overview

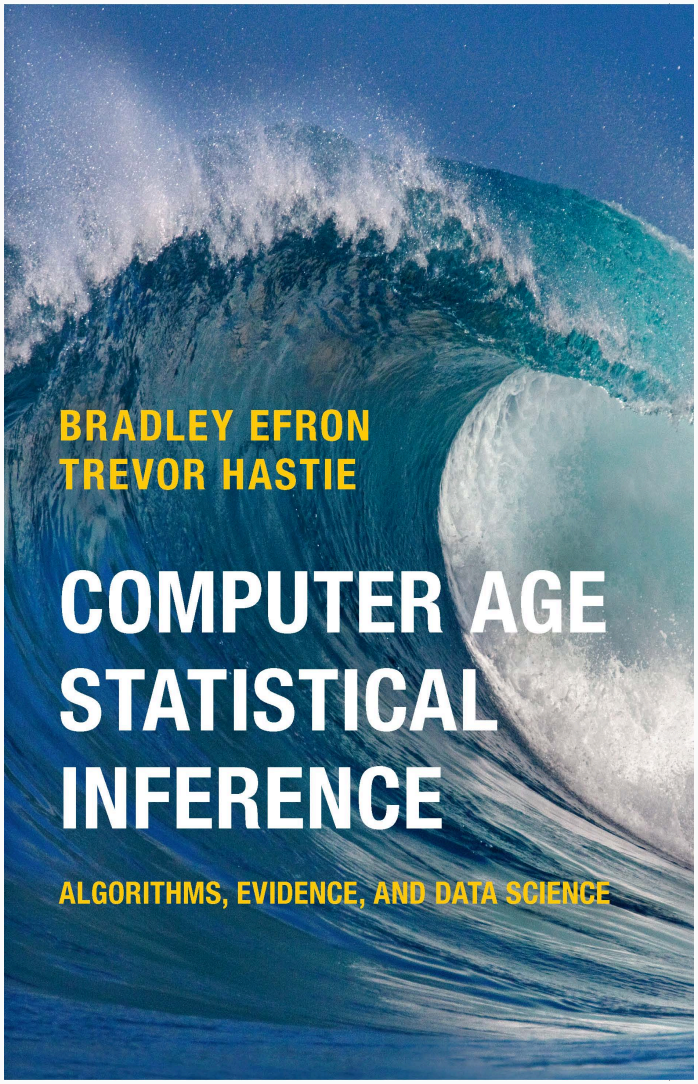
About Me: Not Quite

About Me

Brief Bio

- PhD, MA Econometrics; MSc Ind.Eng. (Comp.Sci./OR)
- Finance Quant for 20+ years
- Open Source for 22+ years
    - Debian developer
    - R package author / contributor
- R Foundation Board member, R Consortium ISC member
- JSS Associate Editor

Motivation



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.

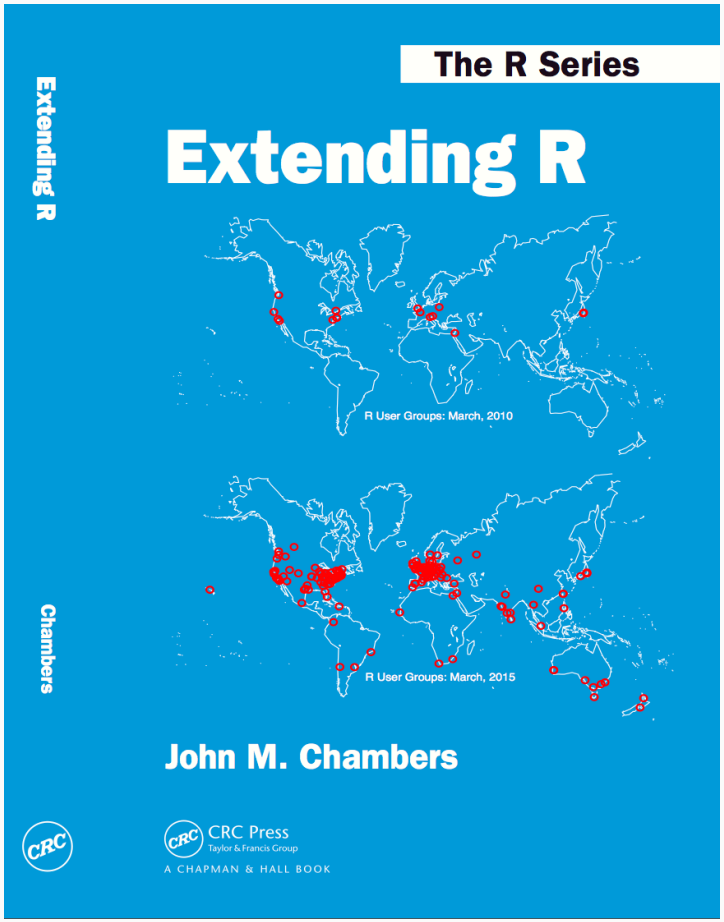
Efron and Hastie, 2016,
pages xv and 6 (footnote 3)

A View of the World

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)

John Chambers



Chambers (2016) Extending R

An entire book about this with *concrete* Python, Julia and C++ code and examples

John Chambers

Chambers 2016, Chapter 1

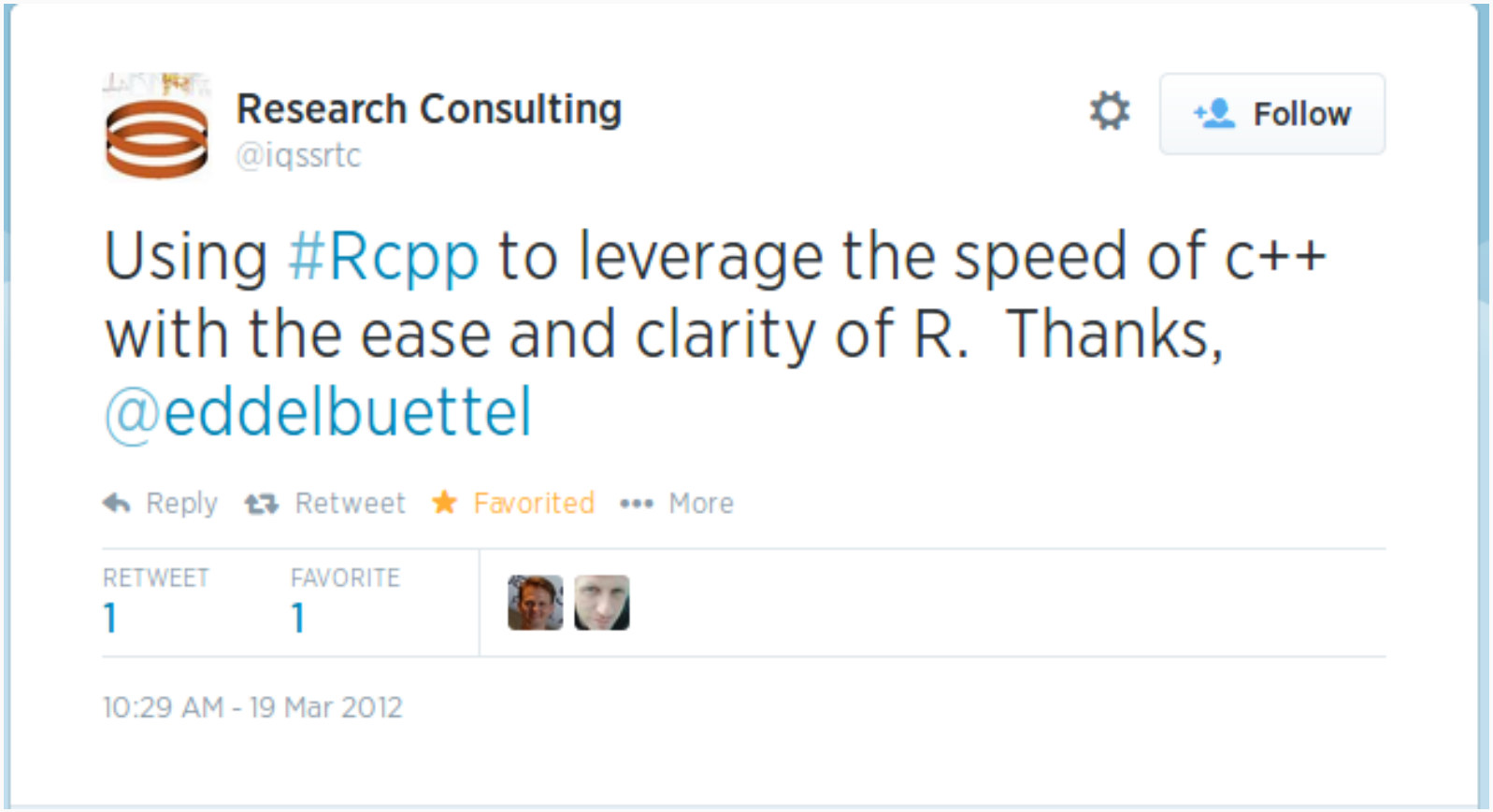
- *Everything that exists in R is an object*
- *Everything happens in R is a function call*
- *Interfaces to other software are part of R*

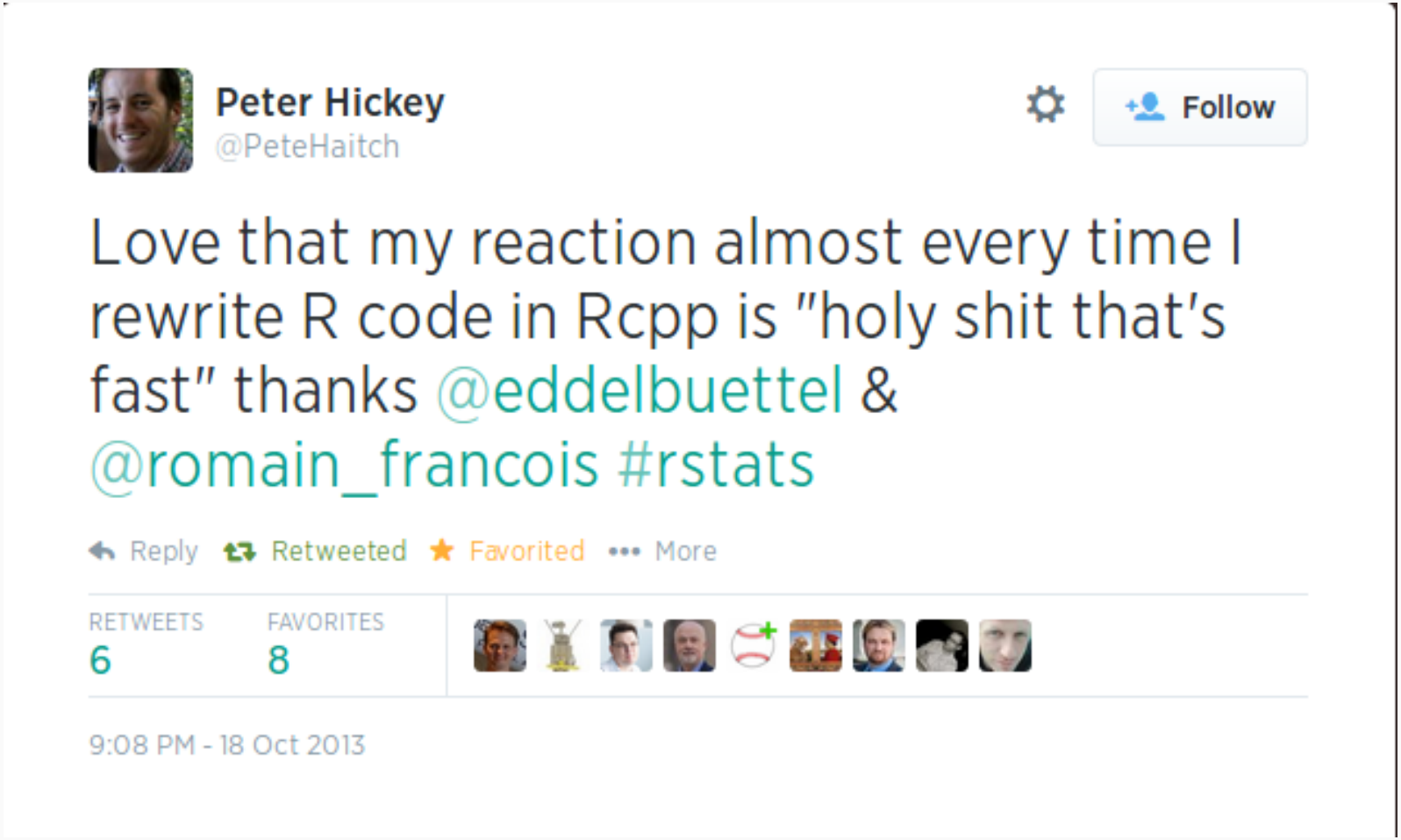
John Chambers

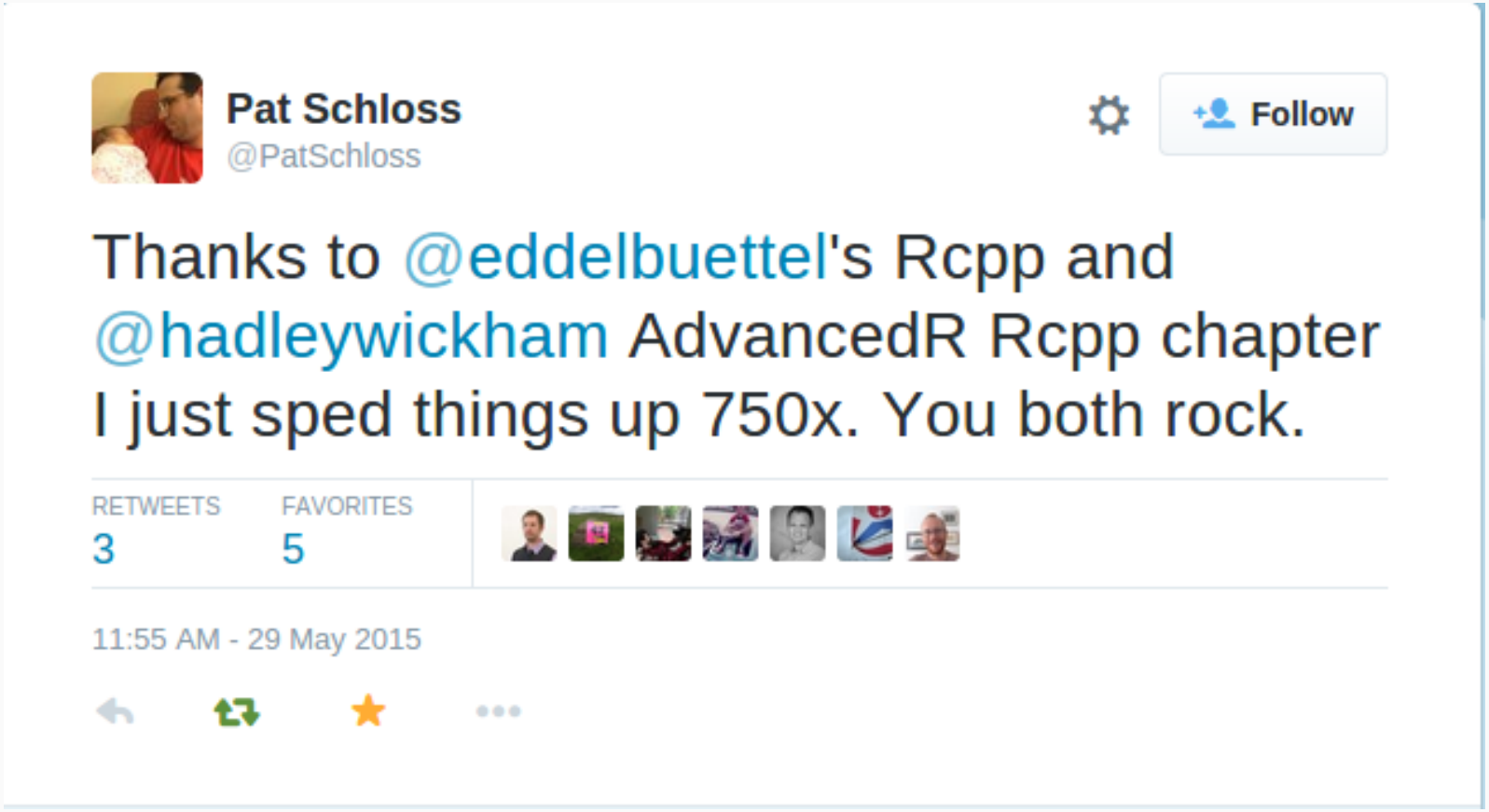
Chambers 2016, Chapter 4

> *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.*

Rcpp: Introduction via Tweets


Research Consulting
@iqssrtc
Follow
Using #Rcpp to leverage the speed of c++ with the ease and clarity of R. Thanks, @eddelbuettel
Reply
Retweet
Favorited
More
RETWEET
1
FAVORITE
1
10:29 AM - 19 Mar 2012

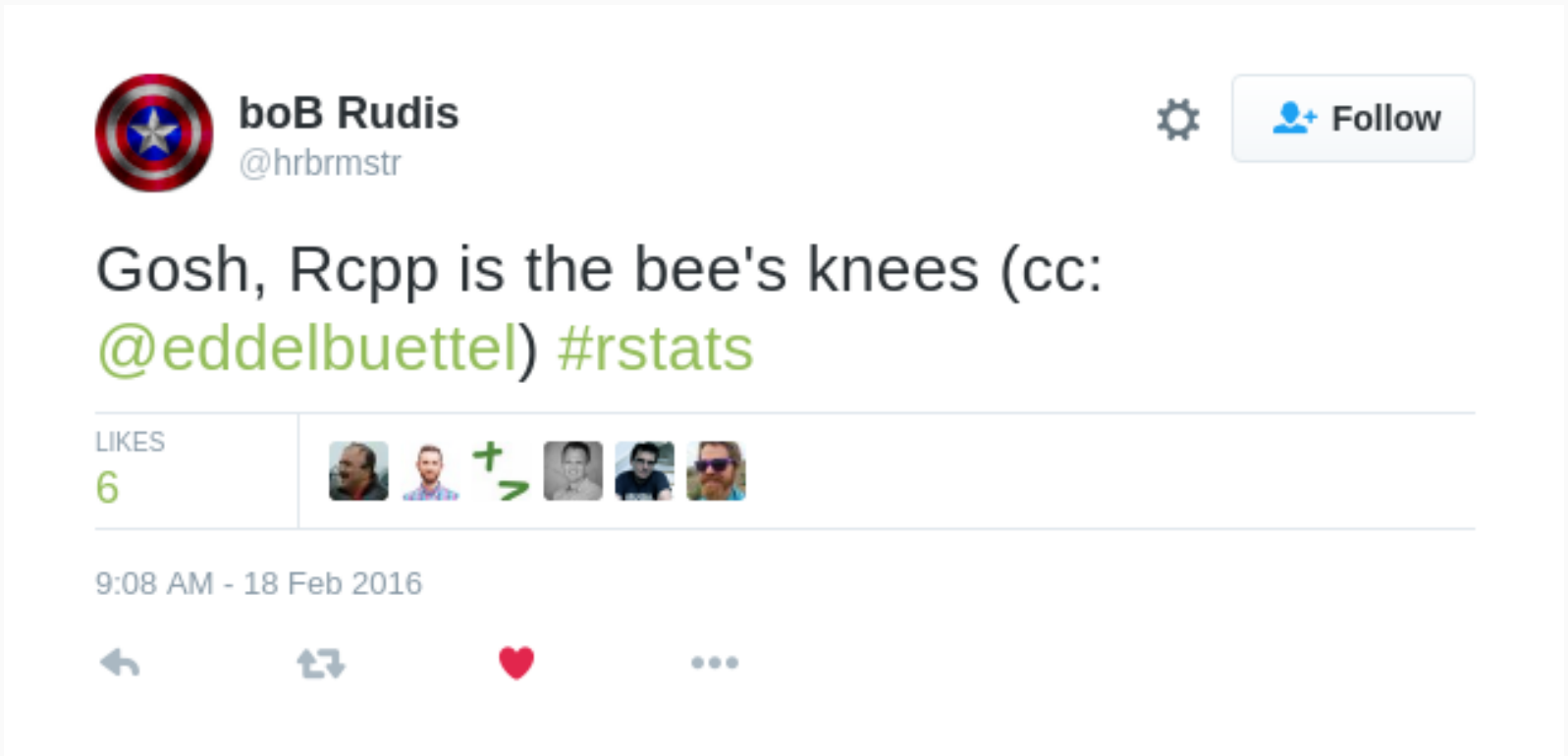

Peter Hickey
@PeteHaitch
Follow
Love that my reaction almost every time I rewrite R code in Rcpp is "holy shit that's fast" thanks @eddelbuettel & @romain_francois #rstats
Reply
Retweeted
Favorited
More
RETWEETS
6
FAVORITES
8
9:08 PM - 18 Oct 2013

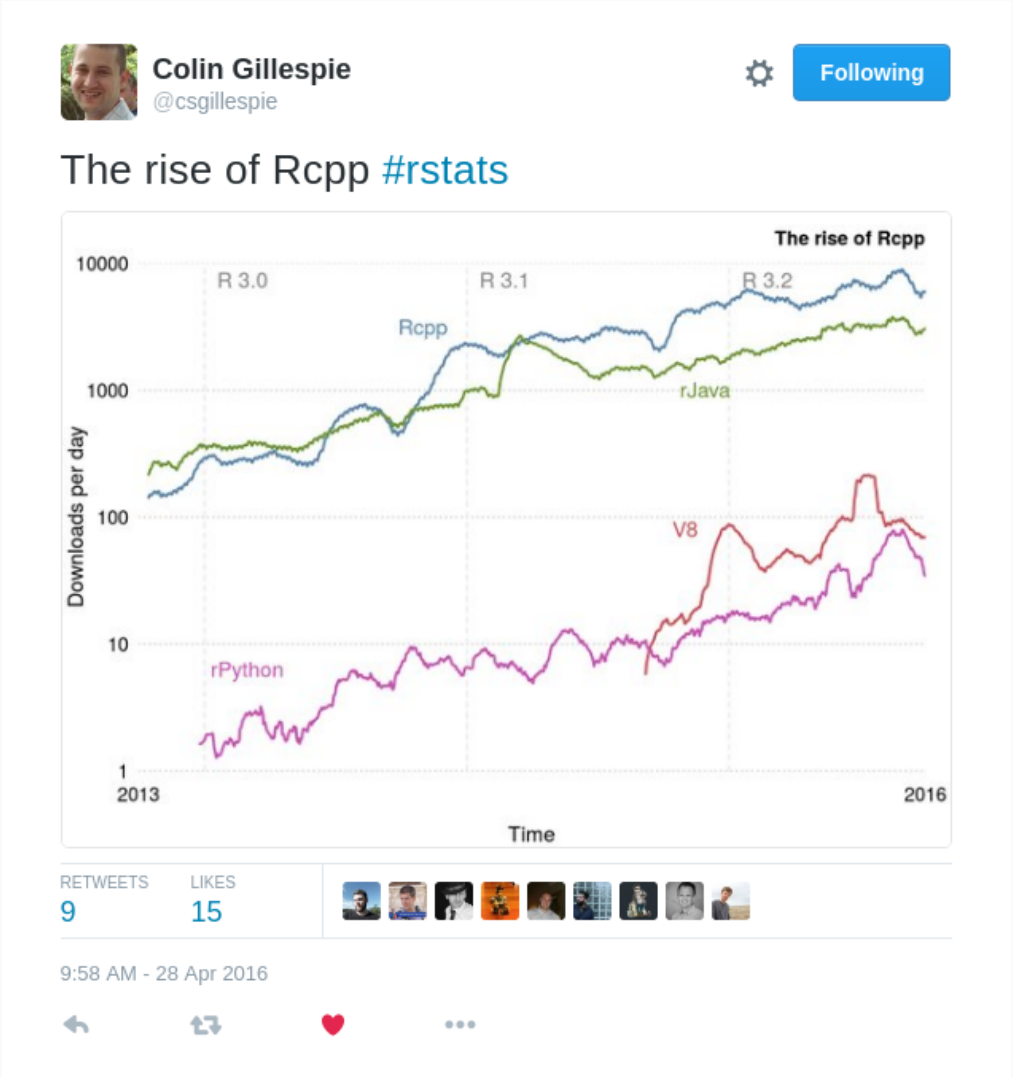

Pat Schloss
@PatSchloss
Follow
Thanks to @eddelbuettel's Rcpp and @hadleywickham AdvancedR Rcpp chapter I just sped things up 750x. You both rock.
RETWEETS
3
FAVORITES
5
11:55 AM - 29 May 2015


Rich FitzJohn
@rgfitzjohn
Follow
Writing some code using #rstats plain C API and realising/remembering quite how much work Rcpp saves - thanks @eddelbuettel
RETWEETS
5
FAVORITES
8
5:45 PM - 6 Mar 2015


Romain François
@romain_francois
Following
"Rcpp is one of the 3 things that changed how I write #rstats code". @hadleywickham at #EARL2014
RETWEETS
3
FAVORITES
7
EARL
3:19 AM - 16 Sep 2014


Karl Broman
@kwbroman
Following
@eddelbuettel @romain_francois Have I emphasized how much I ❤ #Rcpp?
LIKES
8
9:12 PM - 27 May 2016


boB Rudis
@hrbrmstr
Follow
Gosh, Rcpp is the bee's knees (cc: @eddelbuettel) #rstats
LIKES
6
9:08 AM - 18 Feb 2016


Colin Gillespie
@csgillespie
Following
The rise of Rcpp #rstats
The rise of Rcpp
10000
1000
100
10
1
Downloads per day
R 3.0
R 3.1
R 3.2
Rcpp
rJava
V8
rPython
2013
2016
Time
RETWEETS
9
LIKES
15
9:58 AM - 28 Apr 2016

Dirk Eddelbuettel @eddelbuettel · Oct 25

"It's easier to make an error if I am not using Rcpp"
-- @GaborCsardi , right now in the (wicked) R Hub presentation

11

Rcpp: A Better C API for R

The R API

In a nutshell:

- R is a C program, and C programs can be extended
- R exposes an API with C functions and MACROS
- R also supports C++ out of the box with `.cpp` extension
- R provides several calling conventions:
  - `.C()` provides the first interface, is fairly limited, and discouraged
  - `.Call()` provides access to R objects at the C level
  - `.External()` and `.Fortran()` exist but can be ignored
- We will use `.Call()` exclusively

The .Call Interface

At the C level, everything is a `SEXP`, and every `.Call()` access uses this interface pattern:

```
SEXP foo(SEXP x1, SEXP x2) {
...
}
```

which can be called from R via

```
.Call("foo", var1, var2)
```

Note that we need to compile, and link, and load, this manually in wasy which are OS-dependent.

Example: Convolution

```
#include <R.h>
#include <Rinternals.h>

SEXP convolve2(SEXP a, SEXP b) {
    int na, nb, nab;
    double *xa, *xb, *xab;
    SEXP ab;

    a = PROTECT(coerceVector(a, REALSXP));
    b = PROTECT(coerceVector(b, REALSXP));
    na = length(a);
    nb = length(b);
    nab = na + nb - 1;
    ab = PROTECT(allocVector(REALSXP, nab));
    xa = REAL(a);
    xb = REAL(b);
    xab = REAL(ab);
    for (int i = 0; i < nab; i++)
        xab[i] = 0.0;
    for (int i = 0; i < na; i++)
        for (int j = 0; j < nb; j++)
            xab[i + j] += xa[i] * xb[j];
    UNPROTECT(3);
    return ab;
}
```

Example: Convolution

```
#include <Rcpp.h>

// [[Rcpp::export]]
Rcpp::NumericVector
convolve2cpp(Rcpp::NumericVector a,
             Rcpp::NumericVector b) {
    int na = a.length(), nb = b.length();
    Rcpp::NumericVector ab(na + nb - 1);
    for (int i = 0; i < na; i++)
        for (int j = 0; j < nb; j++)
            ab[i + j] += a[i] * b[j];
    return(ab);
}
```

Basic Usage

Basic Usage: evalCpp()

`evalCpp()` evaluates a single C++ expression. Includes and dependencies can be declared.

This allows us to quickly check C++ constructs.

```
library(Rcpp)
evalCpp("2 + 2")      # simple test
```

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

```
evalCpp("std::numeric_limits<double>::max()")
```

```
## [1] 1.79769e+308
```

Basic Usage: cppFunction()

`cppFunction()` creates, compiles and links a C++ file, and creates an R function to access it.

```
cppFunction("
    int exampleCpp11() {
        auto x = 10;
        return x;
}", plugins=c("cpp11"))
exampleCpp11()  # same identifier as C++ function
```

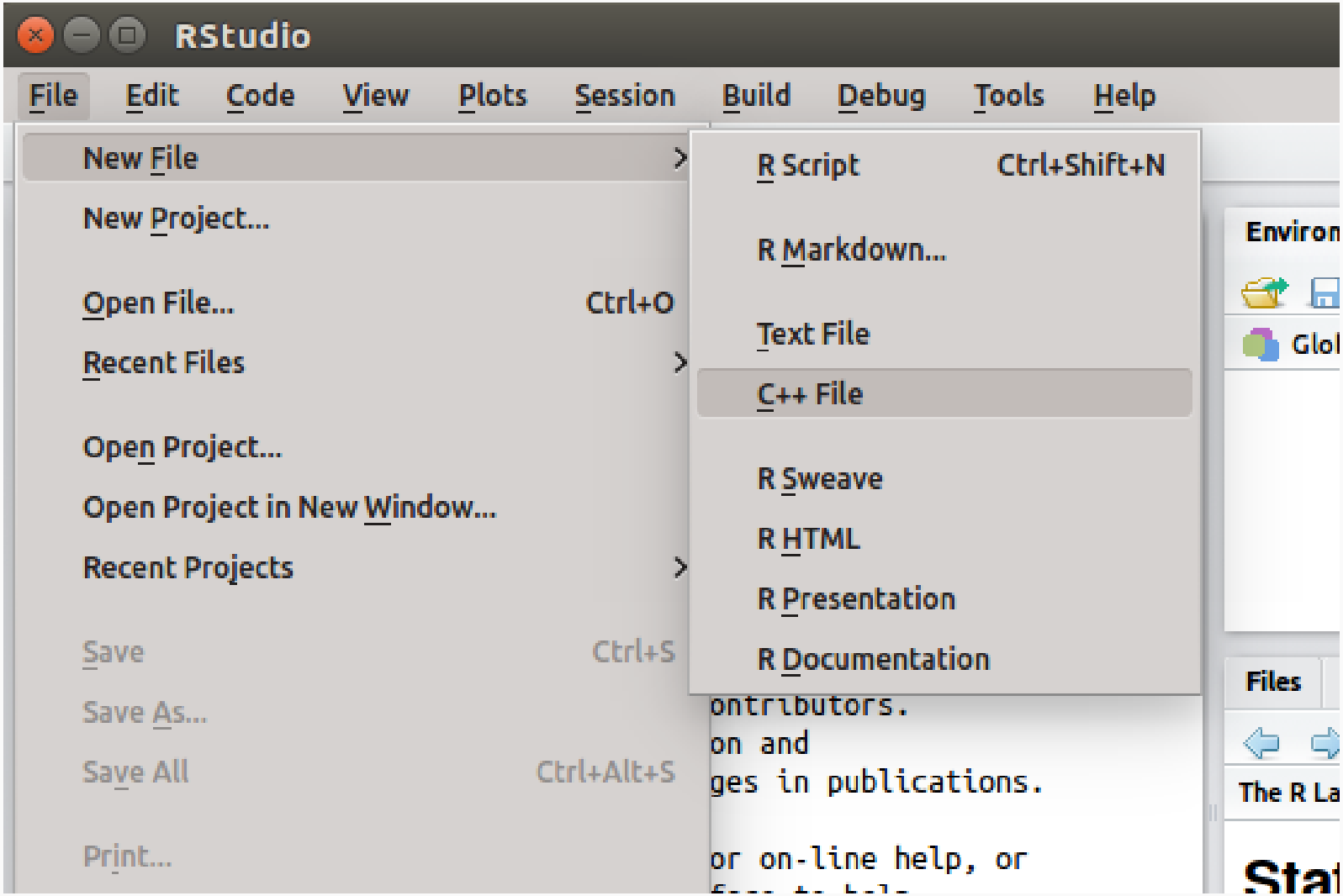
Basic Usage: sourceCpp()

`sourceCpp()` is the actual workhorse behind `evalCpp()` `and`cppFunction()‘. It is described in more detail in the package vignette Rcpp-attributes.

`sourceCpp()` builds on and extends `cxxfunction()` from package inline, but provides even more ease-of-use, control and helpers – freeing us from boilerplate scaffolding.

A key feature are the plugins and dependency options: other packages can provide a plugin to supply require compile-time parameters (cf RcppArmadillo, RcppEigen, RcppGSL).

Basic Uage: RStudio



Basic Uage: RStudio (Cont'ed)

The following file gets created:

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

// This is a simple example of exporting a C++ function to R. You can
// source this function into an R session using the Rcpp::sourceCpp
// function (or via the Source button on the editor toolbar). ...

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

// You can include R code blocks in C++ files processed with sourceCpp
// (useful for testing and development). The R code will be automatically
// run after the compilation.

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

Basic Uage: RStudio (Cont'ed)

So what just happened?

- We defined a simple C++ function
- It operates on a numeric vector argument
- We asked Rcpp to 'source it' for us
- Behind the scenes Rcpp creates a wrapper
- Rcpp then compiles, links, and loads the wrapper
- The function is available in R under its C++ name

Basic Usage: Packages

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

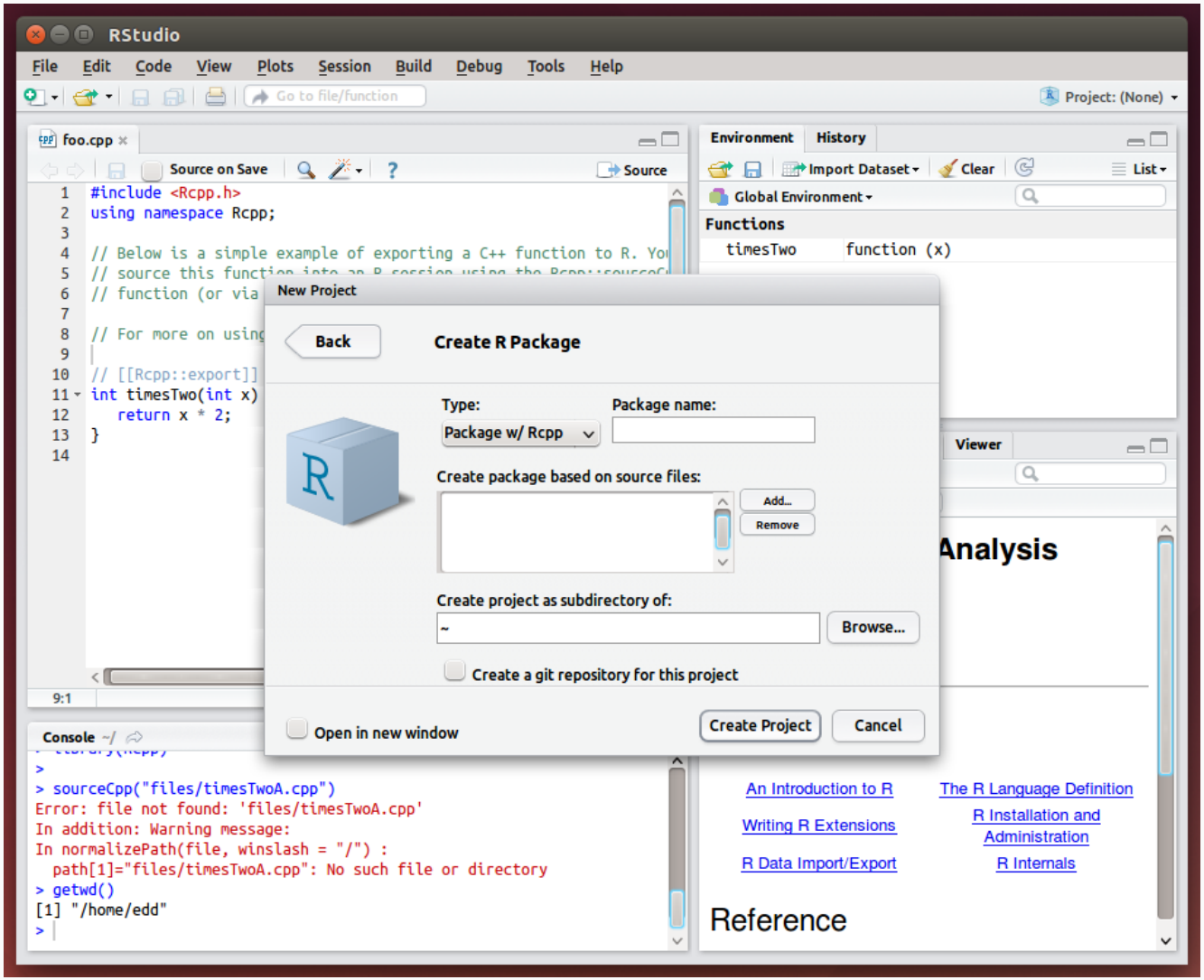
Creating packages with Rcpp is easy; an empty one to work from can be created by `Rcpp.package.skeleton()`

The vignette Rcpp-packages has fuller details.

As of January 14, 2017, there are 907 packages on CRAN which use Rcpp, and a further 89 on BioConductor — with working, tested, and reviewed examples.

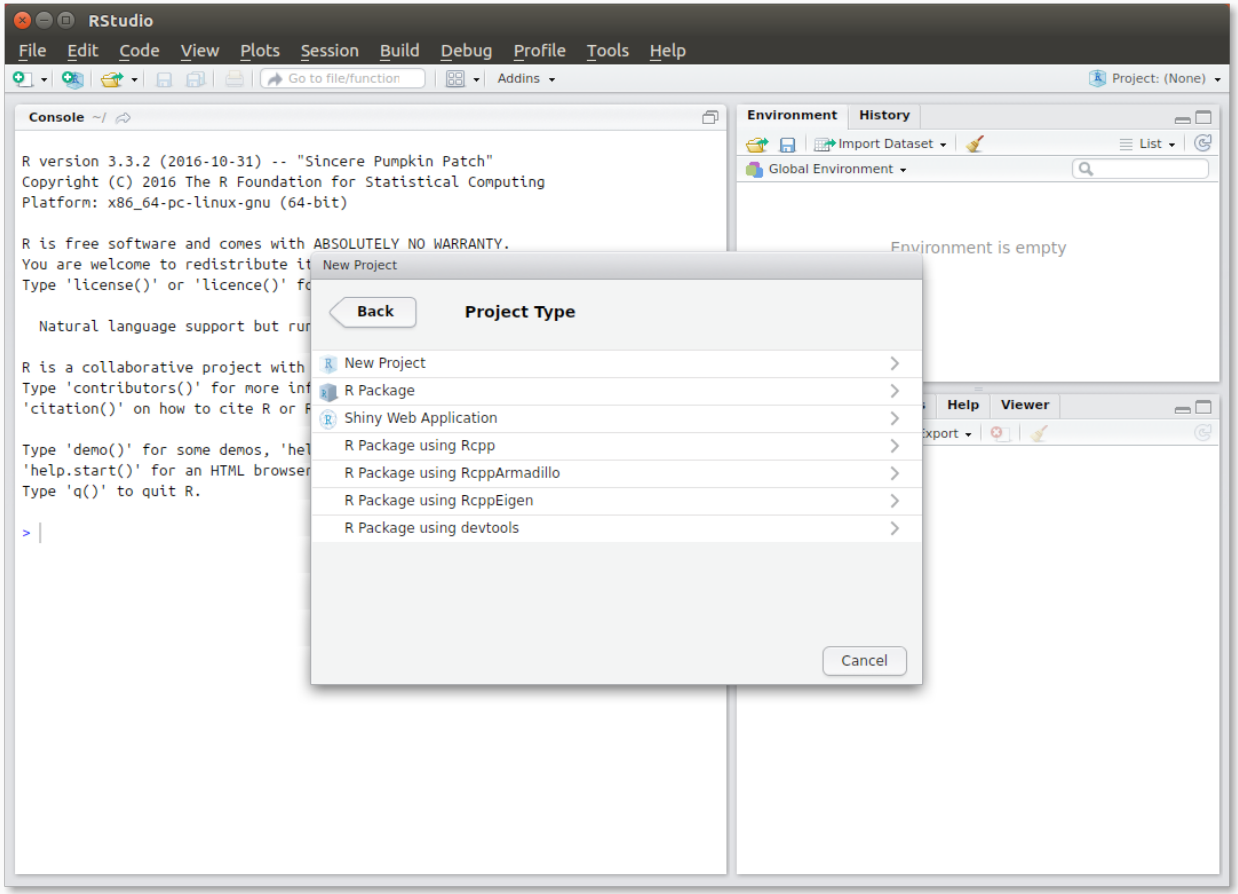
Packages and Rcpp

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



Packages and Rcpp – New!

Now supports RcppArmadillo, RcppEigen,... and package templates!



NB: This currently requires a *nightly development build* of RStudio.

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;
}
```

Packages and Rcpp

Three key ways to extend R using Rcpp

- *easiest*: just use types and classes offered by Rcpp
- *still easy*: use `LinkingTo` for other header-only packages: RcppArmadillo, RcppEigen, BH, ...
- *doable*: external libraries may take a little more work but entirely feasible

Two ways to link to external libraries

- *With linking of libraries:* Do what RcppGSL does and use hooks in the package startup to store compiler and linker flags, pass 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.

But generally still a hard(er) problem. Tooling helps.

RcppMLPACK: K-Means Example

```
#include "RcppMLPACK.h"

using namespace mlpack::kmeans;
using namespace Rcpp;

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

// [[Rcpp::export]]
List cppKmeans(const arma::mat& data, const int& clusters) {

    arma::Col<size_t> assignments;
    KMeans<> k;     // Initialize with the default arguments.
    k.Cluster(data, clusters, assignments);

    return List::create(Named("clusters") = clusters,
                        Named("result")   = assignments);
}
```

Timing

Table 1: Benchmarking result

test	replications	elapsed	relative	user.self	sys.self
mlKmeans(t(wine), 3)	100	0.028	1.000	0.028	0.000
kmeans(wine, 3)	100	0.947	33.821	0.484	0.424

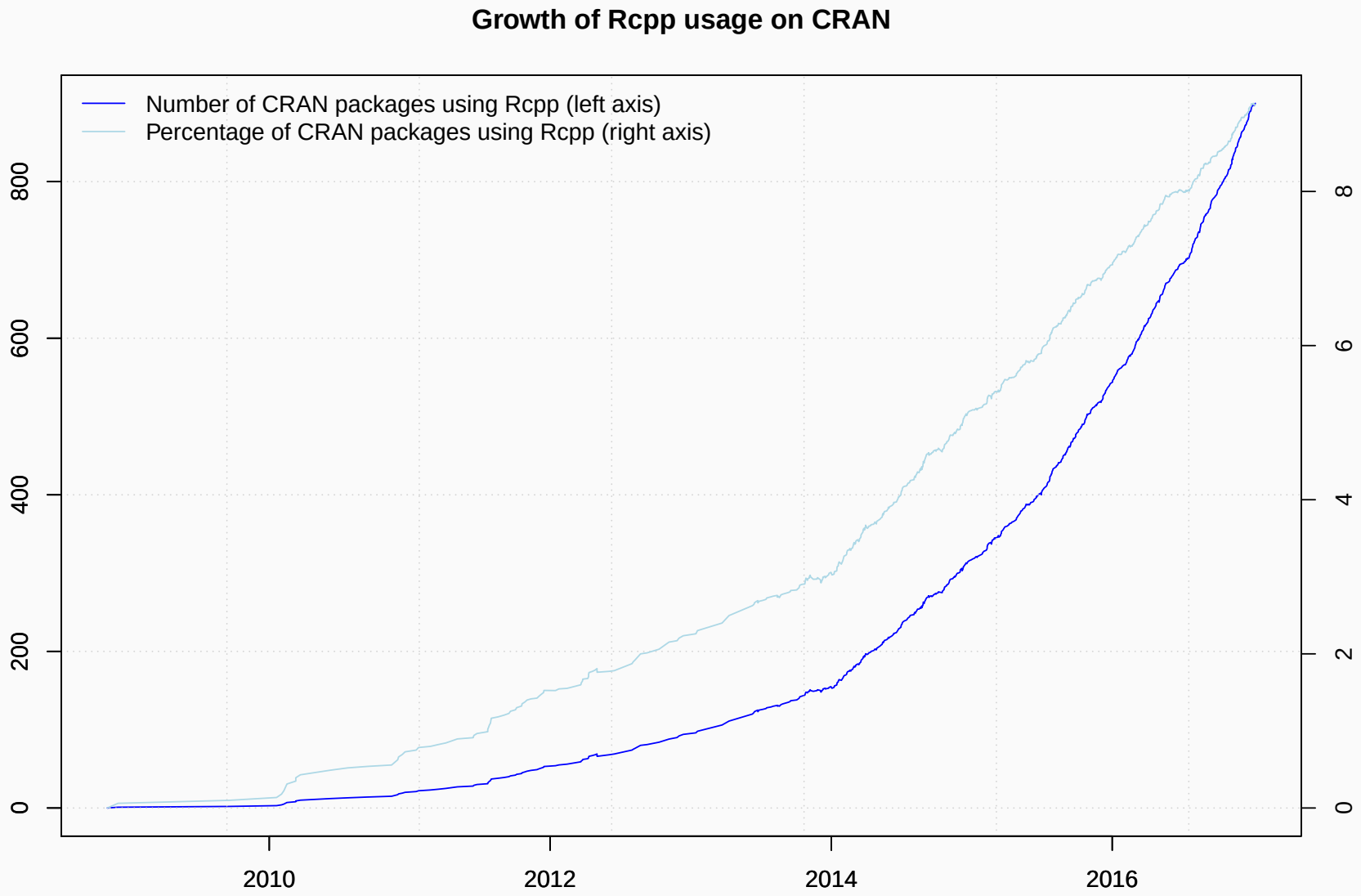
Table taken 'as is' from RcppMLPACK vignette.

RcppMLPACK: Nearest Neighbors Example

```
#include "RcppMLPACK.h"

using namespace Rcpp;
using namespace mlpack;           using namespace mlpack::neighbor;
using namespace mlpack::metric;   using namespace mlpack::tree;

// [[Rcpp::depends(RcppMLPACK)]]
// [[Rcpp::export]]
List nn(const arma::mat& data, const int k) {
    // using a test from MLPACK 1.0.10 file src/mlpack/tests/allknn_test.cpp
    CoverTree<LMetric<2>, FirstPointIsRoot,
              NeighborSearchStat<NearestNeighborSort> > tree =
        CoverTree<LMetric<2>, FirstPointIsRoot,
                  NeighborSearchStat<NearestNeighborSort> >(data);

    NeighborSearch<NearestNeighborSort, LMetric<2>,
                   CoverTree<LMetric<2>, FirstPointIsRoot,
                             NeighborSearchStat<NearestNeighborSort> > >
        coverTreeSearch(&tree, data, true);

    arma::Mat<size_t> coverTreeNeighbors;
    arma::mat coverTreeDistances;
    coverTreeSearch.Search(k, coverTreeNeighbors, coverTreeDistances);

    return List::create(Named("clusters") = coverTreeNeighbors,
                        Named("result")   = coverTreeDistances);

}
```

More

Growth

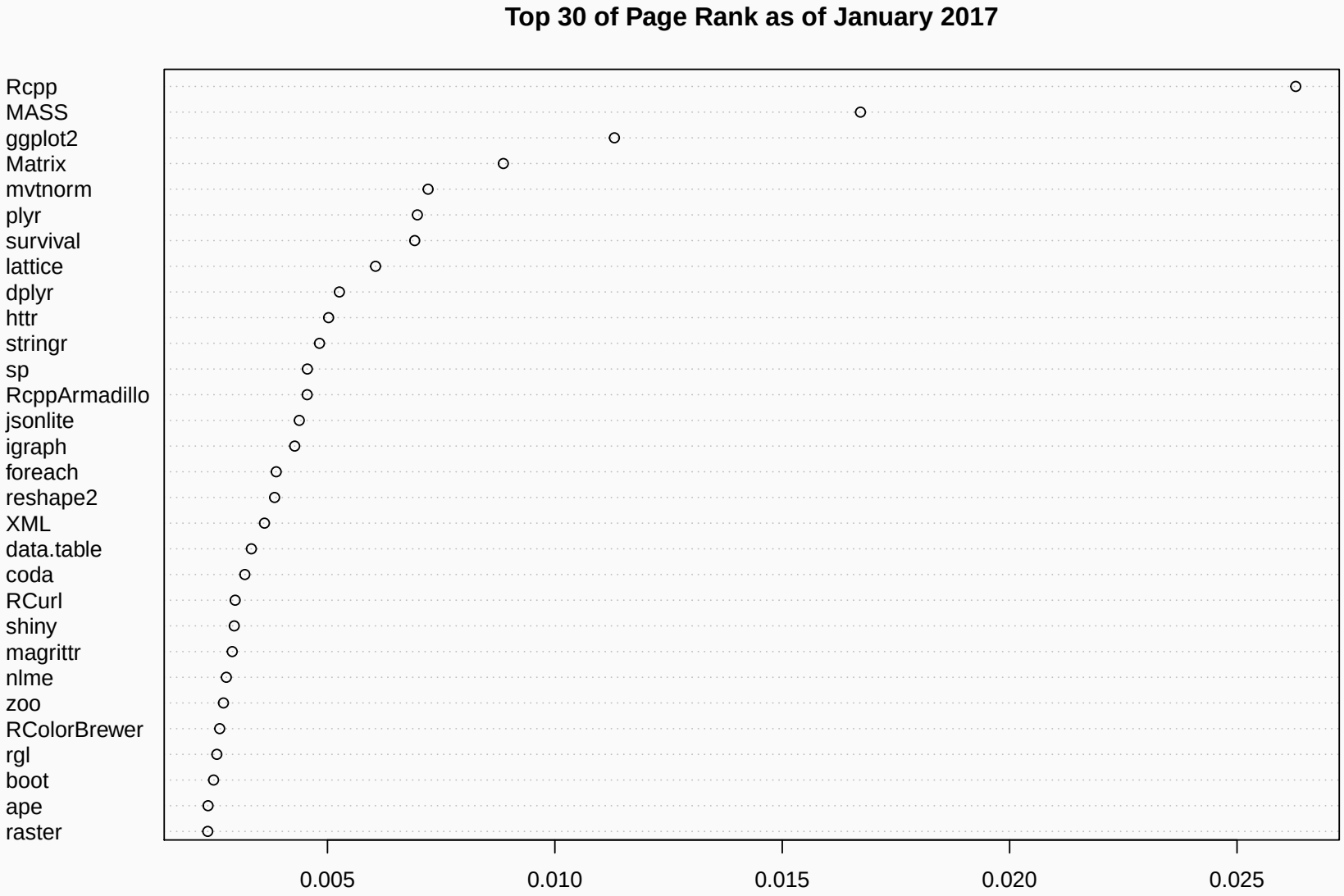

Growth of Rcpp usage on CRAN
Number of CRAN packages using Rcpp (left axis)
Percentage of CRAN packages using Rcpp (right axis)
0
200
400
600
800
0
2
4
6
8
2010
2012
2014
2016

Pagerank

```
library(pagerank)  # github.com/andrie/pagerank

cran <- "http://cloud.r-project.org"

pr <- compute_pagerank(cran)
round(100*pr[1:5], 3)
```

```
##     Rcpp     MASS ggplot2   Matrix mvtnorm
##    2.629    1.672    1.131    0.887    0.721
```

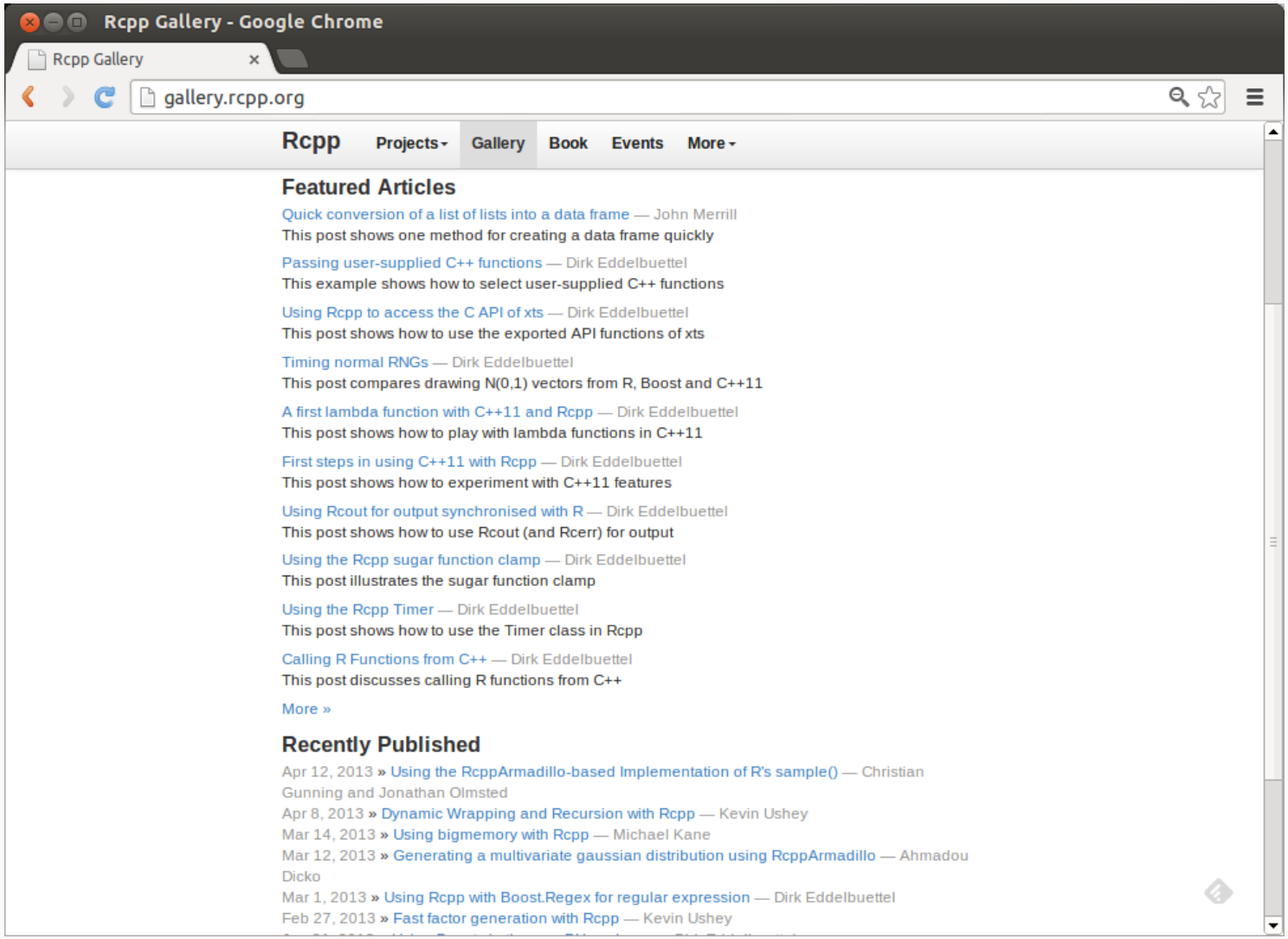
Pagerank


Top 30 of Page Rank as of January 2017
Rcpp
MASS
ggplot2
Matrix
mvtnorm
plyr
survival
lattice
dplyr
httr
stringr
sp
RcppArmadillo
jsonlite
igraph
foreach
reshape2
XML
data.table
coda
RCurl
shiny
magrittr
nlme
zoo
RColorBrewer
rgl
boot
ape
raster
0.005
0.010
0.015
0.020
0.025

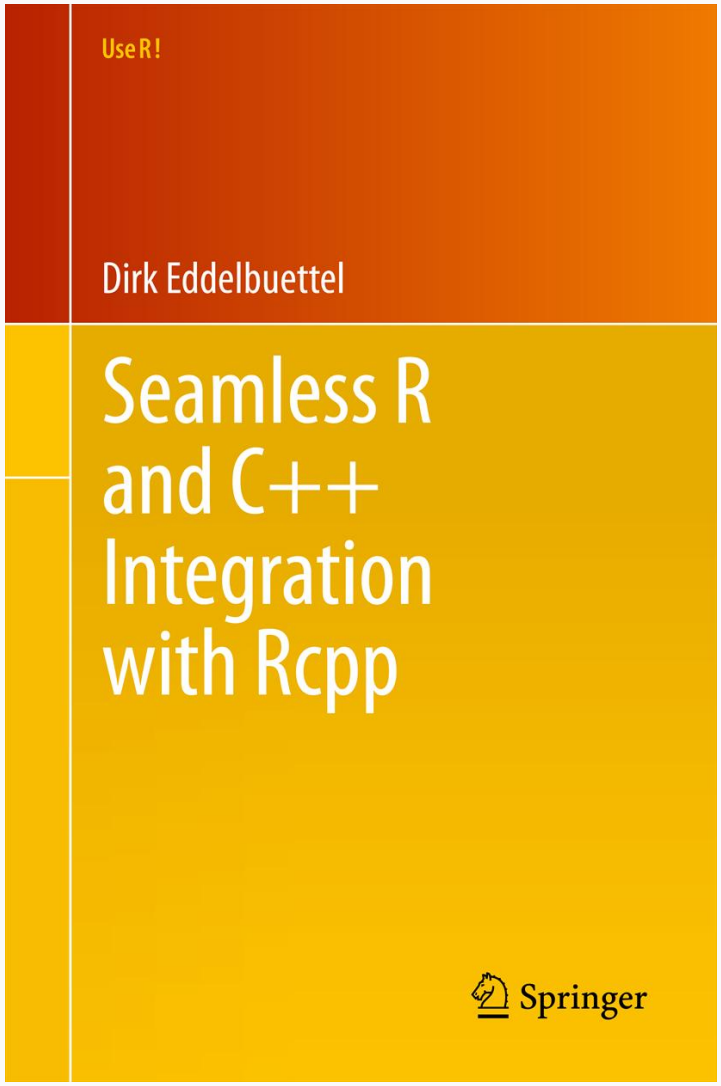
Documentation

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

Rcpp Gallery



The Rcpp book



On sale since June 2013.

Thank You!

http://dirk.eddelbuettel.com/

dirk@eddelbuettel.com

@eddelbuettel