Integrando C++ e R: Uma introdução ao pacote Rcpp

Carlos Trucíos Maza

Universidade Estadual de Campinas
R Campinas User Group

26 de agosto, 2015

R is not a fast language. This is not an accident. R was purposely designed to make data analysis and statistics easier for you to do. It was not designed to make life easier for your computer.

Hadley Wickham

Doing your R code run faster

```
library(Rcpp)
```

```
## Warning: package 'Rcpp' was built under R version 3.1.3
```

```
library(microbenchmark)
x = runif(100)
microbenchmark(sqrt(x),x^0.5,x^(1/2),(x)^(1/2))
```

```
## Unit: nanoseconds
##         expr  min     lq mean median     uq   max neval
##     sqrt(x)  555  606.5 1009  655.5  769.5  3112   100
##       x^0.5 3457 3527.5 4012 3598.0 3829.0 12089   100
##     x^(1/2) 3601 3724.5 4555 3826.5 4105.0 33972   100
##   (x)^(1/2) 3636 3770.0 4620 3854.0 4022.0 42540   100
```

Doing your R code run faster

```
f0 = function() NULL
f1 = function(a = 1) NULL
f2 = function(a = 1, b = 2) NULL
f3 = function(a = 1, b = 2, c = 3) NULL
f4 = function(a = 1, b = 2, c = 3, d = 4) NULL
microbenchmark(f0(), f1(), f2(), f3(), f4(), times = 2000)
```

```
## Unit: nanoseconds
##  expr min  lq  mean median  uq   max neval
##  f0() 119 127 179.6    130 134 40160  2000
##  f1() 136 152 188.3    156 199  7233  2000
##  f2() 154 169 226.7    173 246  7529  2000
##  f3() 173 189 278.8    203 293  8059  2000
##  f4() 192 208 305.1    239 333  9281  2000
```

Doing your R code run faster

```
microbenchmark(
  "[30,11]" = mtcars[30,11],
  "$carb[30]" = mtcars$carb[30],
  "[[c(11,30)]]" = mtcars[[c(11,30)]],
  "[[11]][30]" = mtcars[[11]][30],
  ".subset2" = .subset2(mtcars,11)[30])
```

```
## Unit: nanoseconds
##           expr   min      lq     mean median      uq
##       [30,11] 20055 21071.0 370279.0  22886 26819.5 346
##     $carb[30] 10585 11817.5  13982.7  13316 15350.5
##  [[c(11,30)]]  8456  9517.0  11566.8  11047 12520.5
##    [[11]][30]  8092  9045.5  11503.8  10164 11591.5
##      .subset2   313   519.5    650.7    608   759.5
```

Doing your R code run faster

```r
# Função 1
f_ife = function(x,a,b){
  ifelse(x<=a, a, ifelse(x>=b, b, x))
}
# Função 2
f_pmm = function(x,a,b){
  pmax(pmin(x,b),a)
}
# Função 3
f_place = function(x,a,b){
  x[x<=a] = a
  x[x>=b] = b
  x
}
```

Doing your R code run faster

```
x = runif(1000,-2,2)
microbenchmark(f_ife(x,-1,1),f_pmm(x,-1,1),f_place(x,-1,1)
```

```
## Unit: microseconds
##                 expr    min      lq   mean median      uq
##    f_ife(x, -1, 1) 486.45 492.32 647.10 500.54 607.68 37
##    f_pmm(x, -1, 1)  58.49  61.79  71.95  64.55  70.43
##  f_place(x, -1, 1)  50.50  52.95  66.97  54.37  61.11
```

Doing your R code run faster

```
microbenchmark(f_ife(x,-1,1),f_pmm(x,-1,1),f_place(x,-1,1),
               f_ife_Rcpp(x,-1,1),times=1000)
```

```
## Unit: microseconds
##                   expr     min       lq   mean  median
##       f_ife(x, -1, 1) 486.817 494.164 606.84 507.123 610
##       f_pmm(x, -1, 1)  58.647  62.347  78.28  65.148  73
##     f_place(x, -1, 1)  50.350  53.440  66.16  55.219  63
##  f_ife_Rcpp(x, -1, 1)   4.749   6.918  12.20   8.282   9
```

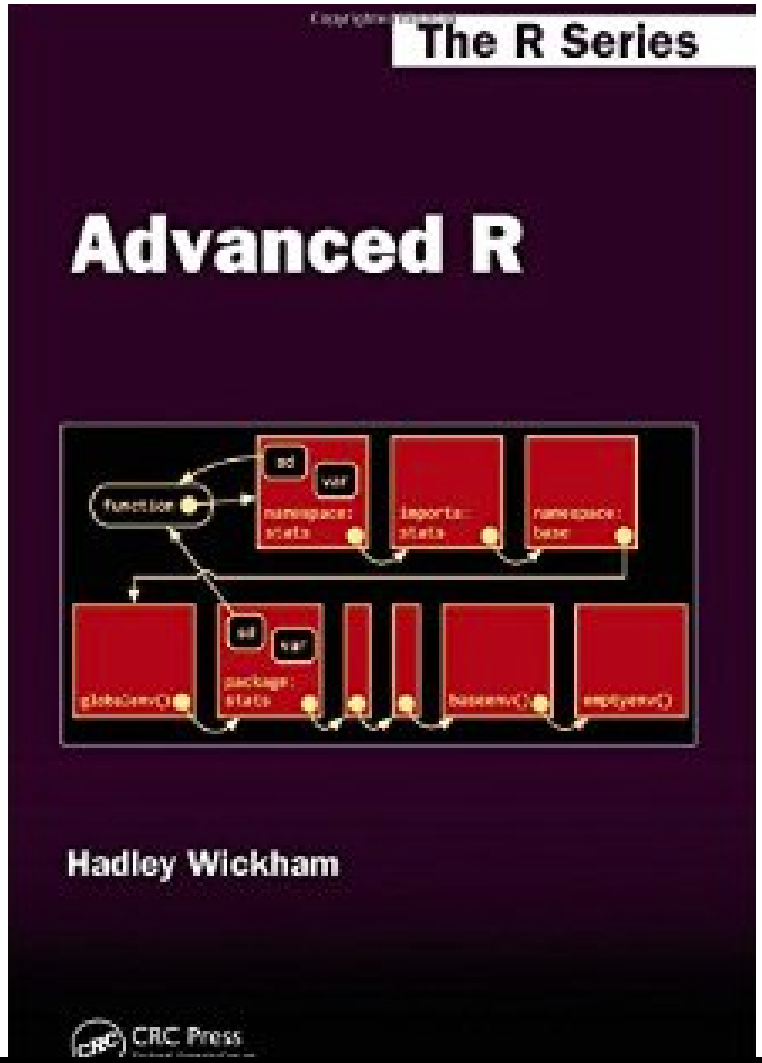
Doing your R code run faster

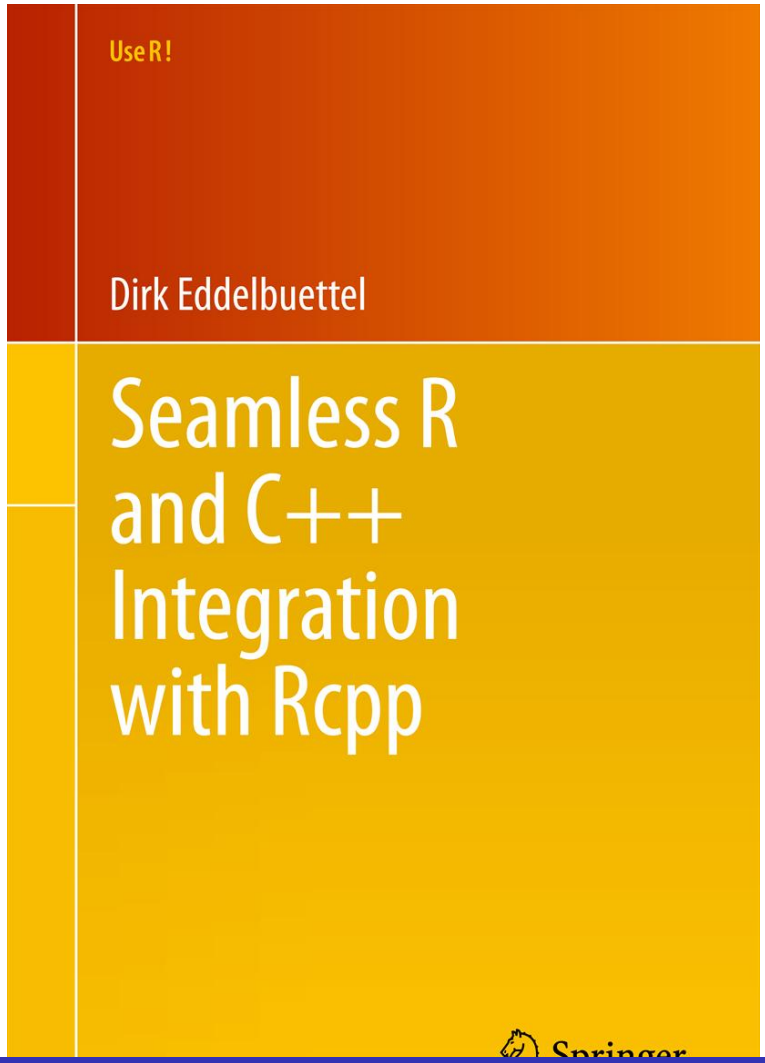
```
x = runif(1e6)
y = runif(1e6)
z = sample(c(T,F),1e6, rep=TRUE)
microbenchmark(sum((x+y)[z]),cond_sum_Rcpp(x,y,z),times=10)
```

```
## Unit: milliseconds
##                       expr   min      lq  mean median      uq
##           sum((x + y)[z]) 30.66 31.185 36.01 33.569 34.077
##  cond_sum_Rcpp(x, y, z)  7.78  8.045 15.05  8.264  9.274
```

What is Rcpp?

- Rcpp é um pacote desenvolvido por Dirk Eddelbuettel
- Rcpp integra R com C++





First example

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

// [[Rcpp::export]]
NumericVector f_place_Rcpp(NumericVector x,
                           double a, double b){
  int n= x.length();  NumericVector out(n);
  for (int i=0; i<n; i++){
    double xi = x[i];
    if (xi<a){  out[i]=a;
    } else if (xi>b){ out[i] = b;
    } else { out[i] = xi;}
  }
  return out;
}
```

Second example

```
// [[Rcpp::export]]
double cond_sum_Rcpp(NumericVector x, NumericVector y,
                     NumericVector z){
  double sum = 0;
  int n = x.length();
  for (int i=0; i<n; i++){
    if(!z[i]) continue;
    sum += x[i]+y[i];
  }
  return sum;
}
```

Rcpp

- sourceCpp()
- IntegerVector: Para vetores de tipo inteiro
- **NumericVector:** Para vetores de tipo numerico
- LogicalVector: Para vetores de tipo lógico
- CharacterVector: Para vetores de tipo carater
- ExpressionVector: Para vetores de expressões
- GenericVector: Listas
- etc

Rcpp: sourceCpp()

```
sourceCpp("Aula01.cpp")
sourceCpp(code='#include <Rcpp.h>
  using namespace Rcpp;

// [[Rcpp::export]]
double cond_sum_Rcpp(NumericVector x, NumericVector y,
NumericVector z){
  double sum = 0;
  int n = x.length();
  for (int i=0; i<n; i++){
    if(!z[i]) continue;
    sum += x[i]+y[i];
  }
  return sum;
}'
)
```

Rcpp: sourceCpp()

```
x = runif(1e6)
y = runif(1e6)
z = sample(c(T,F),1e6, rep=TRUE)
sum((x+y)[z])
```

```
## [1] 499715
```

```
cond_sum_Rcpp(x,y,z)
```

```
## [1] 499715
```

Rcpp: cppFunction()

```
sumR = function(x){
  total = 0
  for (i in seq_along(x)){
    total = total + x[i]
  }
  total
}
```

Rcpp: cppFunction()

```
code = 'double sumC(NumericVector x){
int n=x.size();
double total = 0;
for (int i=0;i<n;i++){
total += x[i];
}
return total;
}'

cppFunction(code)
```

Rcpp: cppFunction()

```
x = rpois(100,6)
microbenchmark(sumR(x),sumC(x),sum(x))
```

```
## Unit: nanoseconds
##      expr   min      lq     mean  median     uq     max neva
##  sumR(x) 36334 40249.5 43395.6 43552.5 44558 101482    10
##  sumC(x)  2010  2388.0  2740.0  2528.5  2798  19223    10
##   sum(x)   294   453.5   596.5   562.5   665   3903    10
```

```
# More information about cppFuntion and sourceCpp
?cppFunction
?sourceCpp
```

Rcpp

```
sourceCpp(code='#include <Rcpp.h>
using namespace Rcpp ;

// [[Rcpp::export]]
int fibonacci(int x){
  if(x == 0) return(0);
  if(x == 1) return(1);
  return(fibonacci(x - 1) + fibonacci(x - 2));
}'
)
```

Rcpp

```
fibR <- function(n){
  if (n==0) return(0)
  if (n==1) return(1)
  return(fibR(n-1) + fibR(n-2))
}
microbenchmark(fibR(10),fibonacci(10))
```

```
## Unit: microseconds
##           expr     min       lq     mean  median       uq
##       fibR(10) 143.876 154.166 181.698 163.474 194.627 2
##  fibonacci(10)   2.046   2.327   3.253   2.556   2.922
```

Rcpp: inline

```
library(inline)
```

```
##
## Attaching package: 'inline'
##
## The following object is masked from 'package:Rcpp':
##
##     registerPlugin
```

Rcpp: inline

```
src <- '
Rcpp::NumericVector xa(a);
Rcpp::NumericVector xb(b);
int n_xa = xa.size(), n_xb = xb.size();

Rcpp::NumericMatrix xab(n_xa,n_xb);
for (int i=0; i < n_xa; i++)
for (int j=0; j < n_xb; j++)
xab(i, j) += xa[i] * xb[j];
return xab;
'
```

Rcpp: inline

```
fun <- cxxfunction(signature(a="numeric",b="numeric"),
                   src,plugin="Rcpp")
fun(1:4,2:5)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    3    4    5
## [2,]    4    6    8   10
## [3,]    6    9   12   15
## [4,]    8   12   16   20
```

Rcpp: IntegerVector

```
src <- '
Rcpp::IntegerVector epn(4);
epn[0] = 6;
epn[1] = 14;
epn[2] = 496;
epn[3] = 8182;
return epn;
'
fun <- cxxfunction(signature(),src,plugin="Rcpp")
fun()
```

```
## [1]    6   14  496 8182
```

Rcpp: IntegerVector

```
src<- '
Rcpp::IntegerVector vec(vx);
int prod = 1;
for (int i = 0; i < vec.size(); i++){
prod *= vec[i];
}
return Rcpp::wrap(prod);'
fun <- cxxfunction(signature(vx="integer"), src,
                   plugin="Rcpp")
fun(c(3.6,5))
```

```
## [1] 15
```

Rcpp: IntegerVector

```
src <-'
Rcpp::IntegerVector vec(vx);
int prod = std::accumulate(vec.begin(),
vec.end(),1,std::multiplies<int>());
return Rcpp::wrap(prod);'
fun <- cxxfunction(signature(vx = "integer"), src,
                   plugin = "Rcpp")
fun(1:10)
```

```
## [1] 3628800
```

Ver *Standard Template Library* (STL) para mais detalhes.

Rcpp: wrap and as

- Rcpp::wrap (from C++ to R)
- Rcpp::as (from R to C++)

Mais informação:
`http://dirk.eddelbuettel.com/code/rcpp.html`

Rcpp: IntegerVector

```
src<- '
Rcpp::NumericVector vec(vx);
double prod = 1;
for (int i = 0; i < vec.size(); i++){
prod *= vec[i];
}
return Rcpp::wrap(prod);'
fun <- cxxfunction(signature(vx="numeric"), src,
                   plugin="Rcpp")
fun(c(3.6,5))
```

```
## [1] 18
```

Rcpp: NumericVector

```
src<-'
Rcpp::NumericVector vec(vx);
double p = Rcpp::as<double>(dd);
double sum = 0.0;
for (int i = 0; i< vec.size(); i++){
sum += pow(vec[i],p);
}
return Rcpp::wrap(sum);
'
fun <- cxxfunction(signature(vx = "numeric",
                   dd = "numeric"), src, plugin = "Rcpp")
fun(1:4,2.2)
```

```
## [1] 37.92
```


TO BE
CONTINUED...