R Schnittstellen - Hochperfomanter Code

Jan-Philipp Kolb

9 Mai 2017

C++ Integration - Überblick über das Paket `rcpp`

Warum die Integration von c++

Robert Gentleman, in R Programming for Bioinformatics, 2008, about R's built-in C interfaces:

> Since R is not compiled, in some situations its performance can be substantially improved by writing code in a compiled language. There are also reasons not to write code in other languages, and in particular we caution against premature optimization, prototyping in R is often cost effective. And in our experience very few routines need to be implemented in other languages for effiiency reasons. Another substantial reason not to use an implementation in some other language is increased complexity. The use of another language almost always results in higher maintenance costs and less stability. In addition, any extensions or enhancements of the code will require someone that is proficient in both R and the other language.

- Rcpp does make some of the above caution statements slightly less critical.

Warum und wann?

- Warum? - R wird langsam oder hat Probleme bei der Speicherverwaltung: zum Beispiel bei Schleifen, die nicht vektorisiert werden können.
- Wann? - wenn man es mit Rcode nicht besser hinbekommt und man den langsamen Code identifiziert hat.

Voraussetzung Compiler

Für Windows, Rtools

1. http://cran.r-project.org/bin/windows/Rtools/
2. http://cran.r-project.org/doc/manuals/R-admin.html#The-Windows-toolset

Für Mac, Xcode

3. http://cran.r-project.org/doc/manuals/R-admin.html#Installing-R-under-_0028Mac_0029-OS-X
4. http://cran.r-project.org/doc/manuals/R-admin.html#Mac-OS-X

Was wir nutzen werden

Wir werden die folgenden beiden Pakete nutzen:

- `inline` und die `cfunction` um Inline C code zu schreiben, der on-the-fly kompiliert wird (Es gibt auch eine `cxxfunction` für C++ Code).
- `Rcpp`, und die Nutzung der Funktion `cppFunction`

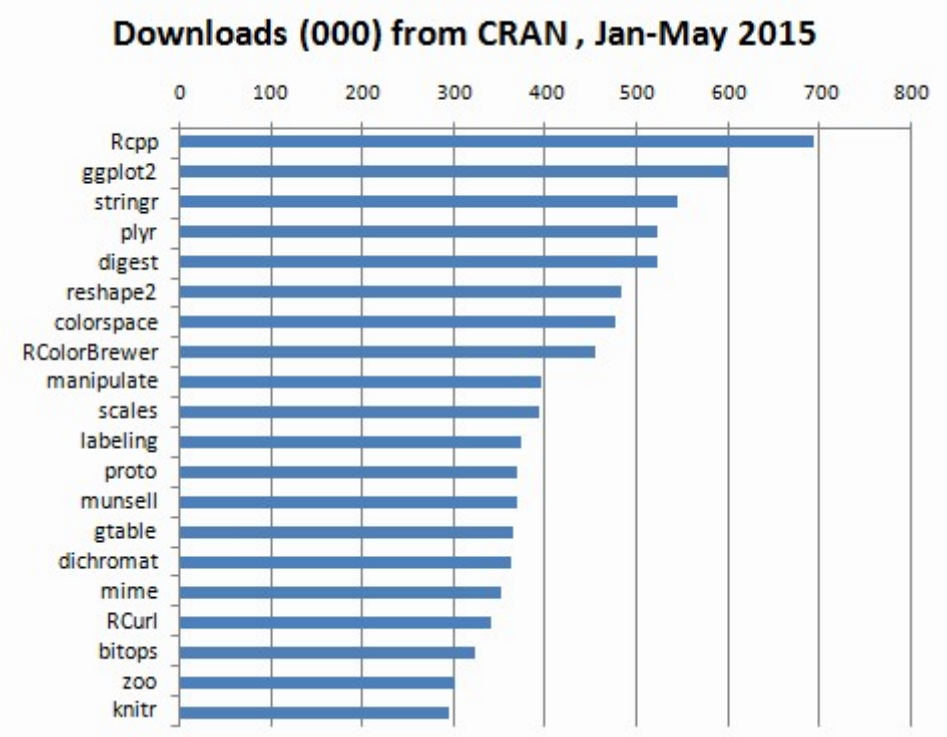


Figure 1:

`Rcpp` ist das am meisten heruntergeladene Paket

Das Paket `inline`

```
install.packages("inline")
```

```
library(inline)
citation("inline")
```

```
##
## To cite package 'inline' in publications use:
##
##   Oleg Sklyar, Duncan Murdoch, Mike Smith, Dirk Eddelbuettel,
##   Romain Francois and Karline Soetaert (2015). inline: Functions
##   to Inline C, C++, Fortran Function Calls from R. R package
##   version 0.3.14. https://CRAN.R-project.org/package=inline
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {inline: Functions to Inline C, C++, Fortran Function Calls from R},
##     author = {Oleg Sklyar and Duncan Murdoch and Mike Smith and Dirk Eddelbuettel and Romain Francois
##     year = {2015},
##     note = {R package version 0.3.14},
##     url = {https://CRAN.R-project.org/package=inline},
##   }
##
```

```
## ATTENTION: This citation information has been auto-generated from
## the package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

Das Rcpp Paket

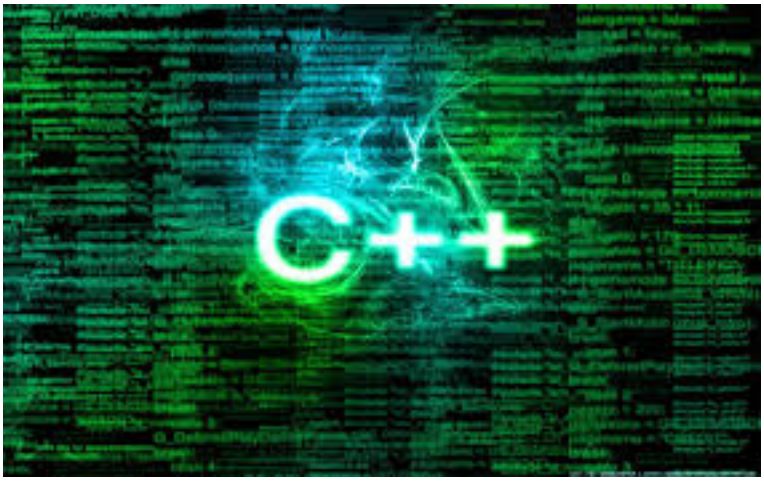


Figure 2:

- Dirk Eddelbuettel und Romain Francois, unter Mitwirkung von Douglas Bates, John Chambers und JJ Allaire.
- Eine flexible Umgebung das die Integration von R und C/C++ ermöglicht
- http://www.rcpp.org/
- Mit dabei ist die Dokumentation und Beispiele:

```
vignette(package = "Rcpp")
```

- Alle Basis R Typen sind als C++ Klassen integriert.
- Man muss sich keine Sorgen über garbage collection machen.

Was ist garbage collection

- Mit dem Befehl `gc()` werden Variablen gelöscht, die nicht mehr benötigt werden/auf die man keinen Zugriff mehr hat.
- Keine Variablen werden gelöscht, die man noch benötigt

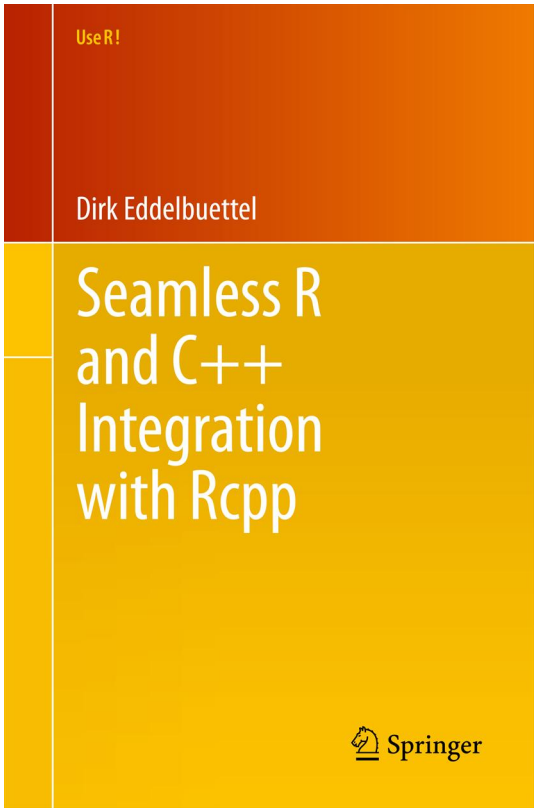
Verwandte Pakete

- `RcppArmadillo` - Armadillo basiert auf einem C++ Paket für lineare Algebra.
- `RcppEigen` - hoch-performante Eigen lineare Algebra Bibliothek.
- `RInside` - R aus C++ heraus nutzen.

Einleitung

- R ist in C geschrieben

- Die Nutzung der Schnittstelle zu C liegt nahe



Das R-Paket CPP

- R Simulationsmodelle bis zu 20 mal schneller
- Hohe Performanz mit Rcpp

```
install.packages("Rcpp")
```

```
library(Rcpp)
cppFunction('int add(int x, int y, int z) {
  int sum = x + y + z;
  return sum;
}')
# add works like a regular R function
add
```

```
add(1, 2, 3)
```

Rcpp

Tutorial on Rcpp by Hadley Wickham

```
library(Rcpp)
```

```
##
## Attaching package: 'Rcpp'
```

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

```
##     registerPlugin
```

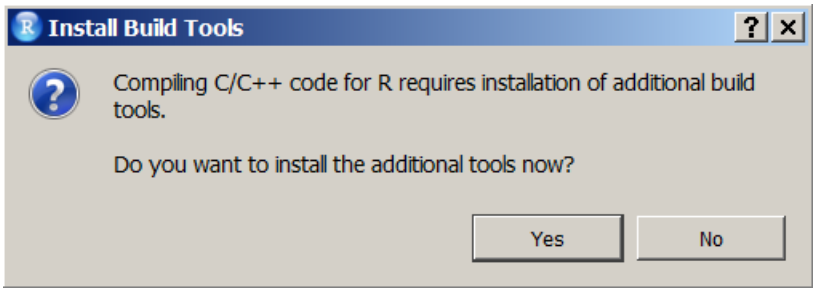


Figure 3:

```
cppFunction('int add(int x, int y, int z) {
  int sum = x + y + z;
  return sum;
}')
```

```
add(1, 2, 3)
```

Benchmarking

```
install.packages("microbenchmark")
```

```
library(microbenchmark)
```

- R-bloggers Artikel zu dem Paket `microbenchmark`

Das Paket `rbenchmark`

```
install.packages("rbenchmark")
```

```
library(rbenchmark)
```

Eine cpp Funktion zum aufsummieren

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

Ein erster Vergleich

```
y <- rnorm(10000000)
microbenchmark(
  sumC(y),
  sum(y)
)
```

```
> library(Rcpp)
> y <- rnorm(10000000)
> microbenchmark(
+   sumC(y),
+   sum(y)
+ )
Unit: milliseconds
    expr      min       lq     mean   median       uq      max neval cld
 sumC(y) 17.85742 18.34534 20.36847 18.98247 22.67203 28.78102   100   a
  sum(y) 17.88387 19.35946 21.19280 19.61960 23.38337 29.48921   100   a
```

Figure 4:

Resourcen

Youtube Videos

- ... mit Dirk Edelbuettel
- ... R mit C++: Rcpp, RInside, und RProtobuf
- Oliver Heidmann - Programmieren in R - Rcpp
- Das Paket `cxxfunplus`

```
install.packages("cxxfunplus")
```

- Rcpp: Seamless R and C++ Integration
- Rcpp Honig
- Advanced R on Memory
- C++ sugar

Überblick über Möglichkeiten des Parallel Computings - Paket parallel

Wann kann man das parallele Rechnen einsetzen

- Anwendbar bei der Wiederholung unabhängiger Berechnungen;
- wenn die Ergebnisse nur kombiniert werden müssen, nachdem parallele Berechnungen durchgeführt wurden.
- Ein Cluster von Knoten: es werden mehrere Arbeiter generiert, die dem Master zuhören; Diese Arbeiter sind neue Prozesse, die auf der aktuellen Maschine oder einer ähnlichen mit einer identischen R-Installation laufen können. Sollte auf allen R-Plattformen funktionieren (Beispiel Paket `snow`).

Zur Verfügung stehende Pakete

- Das Paket `parallel`, das in R Version 2.14.0 zuerst integriert wurde baut auf den CRAN Paketen `multicore` und `snow`.
- Das Paket `foreach` fürht ein neues Schleifenkonzept ein das die parallele Ausführung erlaubt. Es ist die natürliche Wahl, wenn es darum geht eine Schleife zu parallelisieren.

Das Paket `parallel`

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

Ersetzungen für *apply Funktionen

- `mclapply(X, FUN, ...)` (angepasst basierend auf `multicore`).
- `parLapply(cl, X, FUN, ...)` (angepasst basierend auf `snow`).