

Using Rcpp* packages for easy and fast extension of R with C++



Ghislain VIEILLEDENT and Jeanne CLEMENT

Cirad, UMR AMAP, Montpellier, FRANCE





Outline

- 1 Short presentation of Rcpp* packages
 - Rcpp : extending R with C++
 - RcppGSL for fast random draws
 - RcppArmadillo for high-performance linear algebra

-
- 2 Application examples
 - Distance computation
 - Simple linear regression
 - Exercise to do yourself
- 3 Build of R package using Rcpp
 - Usefull functions
 - Example of package using Rcpp



Rcpp R package

- **Rcpp** is an R package to extend R with C++ code
- Main advantage : C++ is fast, it accelerates R (see next sections)
- Written by **Dirk EDDELBUETTEL** and **Romain FRANCOIS**
- <http://www.rcpp.org/>



Simple Rcpp example

C++ code (in file Code/addition.cpp)

```
#include <Rcpp.h>
using namespace Rcpp;
// [[Rcpp::export]]
int addition(int a, int b) {
  return a + b;
}
```

R code

```
Rcpp::sourceCpp("Code/addition.cpp")
addition(2, 2)

## [1] 4
```



Rcpp advantages

Thanks to `Rcpp::sourceCpp()`

- Compile the C++ code
- Export the function to the R session
- Direct interchange of R objects (including S3, S4) between R and C++
- ... (many more, see `vignette("Rcpp-package")`)



GSL and RcppGSL



GNU Scientific Library

- Numerical library for C and C++ programmers
- Reliable random number generator algorithms
- Thoroughly tested and fast random number distributions
- Linear algebra (matrices and vectors)
- <https://www.gnu.org/software/gsl/>

RcppGSL

- Interface between R and GSL
- Using Rcpp to interface R and C
- <http://dirk.eddelbuettel.com/code/rcpp.gsl.html>



GSL random number distributions

- GSL v2.6 includes **38 random number distributions** (see [GNU GSL](#))
- It's easy to implement additional random number distributions from the GSL base distributions (e.g. truncated normal distribution)
- For comparison, R API includes "only" 24 random number distributions (see [Writing R Extensions](#))
- Random draws are faster with GSL than with R (eg. `gsl_ran_gamma()` vs. `R::rgamma()`)



RcppGSL example

C++ code

```
#include <Rcpp.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>
using namespace Rcpp;
// [[Rcpp::depends(RcppGSL)]]
// [[Rcpp::export]]
Rcpp::NumericVector my_rnorm(int nsamp, double mu,
                             double sigma) {
  gsl_rng *s = gsl_rng_alloc(gsl_rng_mt19937); // Random seed
  Rcpp::NumericVector beta(nsamp);
  for (int i = 0; i < nsamp; i++) {
    beta[i] = mu + gsl_ran_gaussian(s, sigma); // Random draw
  }
  return beta;
}
```

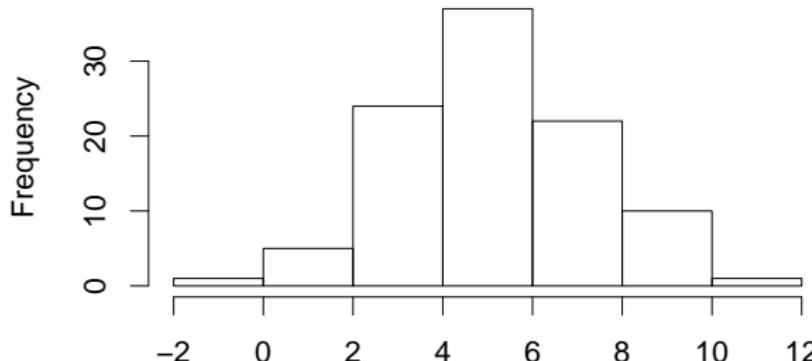


RcppGSL example

R code

```
library(Rcpp)
library(RcppGSL)
beta <- my_rnorm(100, 5, 2)
par(cex=2)
hist(beta)
```

Histogram of beta





Armadillo and RcppArmadillo



Armadillo

- C++ library for linear algebra and scientific computing
- Provides high-level syntax and functionality : speed and ease of use
- Classes for vectors, matrices and cubes
- Matrix operations, matrix decomposition, linear model solver, etc.
- <http://arma.sourceforge.net/>

RcppArmadillo

- Interface between R and Armadillo
- Using Rcpp to interface R and C++
- <http://dirk.eddelbuettel.com/code/rcpp.armadillo.html>



RcppArmadillo example

C++ code

```
#include <RcppArmadillo.h>
// [[Rcpp::depends(RcppArmadillo)]]
// [[Rcpp::export]]
Rcpp::List fastLm(const arma::mat& X, const arma::colvec& y) {
    int n = X.n_rows, k = X.n_cols;

    arma::colvec coef = arma::solve(X, y);      // fit model y ~ X
    arma::colvec res  = y - X*coef;              // residuals
    // std.errors of coefficients
    double s2 = std::inner_product(res.begin(),
                                    res.end(),
                                    res.begin(), 0.0)/(n - k);

    arma::colvec std_err = arma::sqrt(s2 *
        arma::diagvec(arma::pinv(arma::trans(X)*X)));
    return Rcpp::List::create(Rcpp::Named("coefficients") = coef,
                            Rcpp::Named("stderr")      = std_err,
                            Rcpp::Named("df.residual") = n - k);
}
```



RcppArmadillo example

R code

```
library(Rcpp)
library(RcppArmadillo)
# Trees data-set
y <- log(trees$Volume)
X <- cbind(1, log(trees$Girth))
# fastLm
mod <- fastLm(X, y)
mod$coef

##           [,1]
## [1,] -2.353325
## [2,]  2.199970
```



Licenses

- Licenses : GNU General Public License, Apache License 2.0 for Armadillo
- Free software licenses : we can use, modify and redistribute these softwares



Distance computation

C++ code (in file Code/arma_distmat.cpp)

```
#include <RcppArmadillo.h>
// [[Rcpp::depends(RcppArmadillo)]]
using namespace Rcpp;
using namespace arma;

// [[Rcpp::export]]
arma::mat arma_distmat(const arma::mat& x) {

    int np = x.n_rows;

    arma::mat distmat; distmat.zeros(np, np);
    for (int i = 0; i < np; i++) {
        arma::vec p0 = x.row(i).t();
        for (int j = i + 1; j < np; j++) {
            arma::vec p1 = x.row(j).t();
            arma::vec diff = p0 - p1;
            double squared_diff = as_scalar(diff.t() * diff);
            distmat(j, i) = distmat(i, j) = sqrt(squared_diff);
        }
    }
    return distmat;
}
```



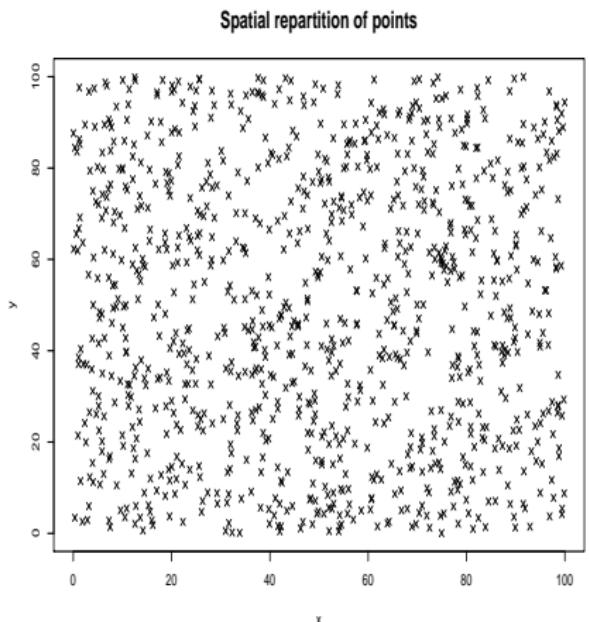
Distance computation

R code

```
R_distmat <- function(X){  
  np <- nrow(X)  
  distmat <- matrix(0,nrow=np,ncol=np)  
  for (i in 1:(np-1)) {  
    p0 <- X[i,]  
    for (j in (i+1):np){  
      p1 <- X[j,]  
      diff <- p0-p1  
      squared_diff <- t(diff)%*% diff  
      distmat[j,i] <- distmat[i,j] <- sqrt(squared_diff)  
    }  
  }  
  return(distmat)  
}
```

Distance computation

1000 points dispersed all over the space :



Distance matrix for 6 points :

0.0	36.0	48.6	61.4	57.3	95.8
36.0	0.0	25.8	68.2	76.8	100.2
48.6	25.8	0.0	50.7	66.5	78.8
61.4	68.2	50.7	0.0	24.9	34.5
57.3	76.8	66.5	24.9	0.0	47.9
95.8	100.2	78.8	34.5	47.9	0.0

Distance computation

Comparison of compilation times

```
Rcpp::sourceCpp("Code/arma_distmat.cpp")
library(rdist)
# Benchmark
library(rbenchmark)
Benchmark <- benchmark(
  "arma_distmat" = {arma_distmat(X)},
  "cdist" = {cdist(X,X)},
  "R_distmat" = {R_distmat(X)},
  replications=30,
  columns = c("test", "elapsed", "relative"))
knitr::kable(Benchmark,digits=1, booktabs=TRUE) %>%
  kableExtra::kable_styling(latex_options=c("HOLD_position","striped"),
                            full_width=FALSE)
```

test	elapsed	relative
arma_distmat	0.5	1.0
cdist	1.5	3.1
R_distmat	63.3	130.2



Simple linear regression

C++ code

```
#include <RcppArmadillo.h>
// [[Rcpp::depends(RcppArmadillo)]]
// [[Rcpp::export]]
Rcpp::List arma_fastLm(const arma::mat& X, const arma::colvec& y) {
    int n = X.n_rows, k = X.n_cols;

    arma::colvec coef = arma::solve(X, y);      // fit model  $y \sim X$ 
    arma::colvec res  = y - X*coef;              // residuals
    // std.errors of coefficients
    double s2 = arma::sum(res.t()*res);
    arma::colvec std_err = arma::sqrt(s2 *
        arma::diagvec(arma::pinv(arma::trans(X)*X)));
    return Rcpp::List::create(Rcpp::Named("coefficients") = coef,
                           Rcpp::Named("stderr") = std_err,
                           Rcpp::Named("residuals") = res,
                           Rcpp::Named("df.residual") = n - k);
}
```



Simple linear regression

R code

```
R_fastLm <- function(X, y) {  
  n <- nrow(X)  
  k <- ncol(X)  
  
  # fit model  $y \sim X$   
  coef <- solve(t(X) %*% X) %*% t(X) %*% y  
  # residuals  
  res <- y - X %*% coef  
  # std.errors of coefficients  
  s2 = sum(t(res) %*% res)/(n - k);  
  std_err = sqrt(s2 %*% diag(solve(t(X)%*%X)))  
  return(list("coefficients" = coef,  
             "stderr" = std_err,  
             "residuals" = res,  
             "df.residual" = n - k))  
}
```



Simple linear regression

Comparison of compilation times

```
# Data simulation
X <- matrix(rnorm(500000), ncol=100)
y <- rnorm(5000)
# Call C++ function
Rcpp::sourceCpp("Code/arma_fastLm.cpp")
# Benchmark
library(rbenchmark)
Benchmark <- benchmark(
  "arma_fastLm" = {arma_fastLm(X,y)},
  "R_fastLm" = {R_fastLm(X,y)},
  replications=30,
  columns = c("test", "elapsed", "relative"))
knitr::kable(Benchmark,digits=1, booktabs=TRUE) %>%
  kableExtra::kable_styling(latex_options=c("HOLD_position","striped"),
                            full_width=FALSE, font_size=8)
```

test	elapsed	relative
arma_fastLm	3.6	1.0
R_fastLm	4.4	1.2



Representation of results



Exercise to do yourself

Implement a function in C++ to select in a matrix the lines identified by a vector of integer and return the corresponding matrix.

C++ code

```
#include <RcppArmadillo.h>
// [[Rcpp::depends(RcppArmadillo)]]
// [[Rcpp::export]]
arma::mat mat_select_lines(arma::mat& X, arma::uvec& rowId) {
    int nrows = rowId.n_elem;
    int ncols = X.n_cols;
    arma::mat R(nrows, ncols);
    for(int i = 0; i < nrows; i++) {
        for(int j = 0; j < ncols; j++) {
            R(i, j) = X(rowId(i), j);
        }
    }
    return R;
}
```



How to build an R package around C++ functions

- `Rcpp.package.skeleton()` to generate a new Rcpp package (modifying `DESCRIPTION` and `NAMESPACE`)
- `Rcpp::compileAttributes()` scans the C++ files and generates the `RcppExports.cpp` file to make the functions preceded by `// [[Rcpp::export]]` available in R.
- Implement R functions that checks the conformity of user-defined parameters, calls functions in C++ and returns the results in an easy-to-use format.

jSDM R package

jSDM 0.1.0

Get started Reference Articles ▾ Change log

[Twitter](#) [GitHub](#)

jSDM R Package

Package for fitting joint species distribution models (jSDM) in a hierarchical Bayesian framework (Warton *et al.* 2015). The Gibbs sampler is written in C++. It uses Rcpp, Armadillo and GSL to maximize computation efficiency.



System requirements

Make sure the GNU Scientific Library (GSL) is installed on your system.

Installation

Install the latest stable version of jSDM from CRAN with:

```
install.packages("jSDM")
```

Or install the development version of jSDM from GitHub with:

```
devtools::install_github("ghislainv/jSDM")
```

References

Warton, D.I., Blanchet, F.G., O'Hara, R.B., Ovaskainen, O., Taskinen, S., Walker, S.C. & Hui, F.K. (2015) So many variables: Joint modeling in community ecology. *Trends in Ecology & Evolution*, **30**, 766–779.

Links

[Download from CRAN at https://cloud.r-project.org/package=jSDM](#)
[Browse source code at https://github.com/ghislainv/jSDM](#)
[Report a bug at https://github.com/ghislainv/jSDM/issues](#)

License

GPL-3 | file LICENSE

Developers

Ghislain Vieilledent
 Author, maintainer 

Jeanne Clément
 Author 

 **cirad**
 Copyright holder, funder

Dev status

build 
 CRAN **0.1.0**
 DOI [10.5281/zenodo.3253460](#)
 downloads **231/month**

- <https://ecology.ghislainv.fr/jSDM>
- Made with Rcpp* packages



... Thank you for attention ...

 @ghislainv

<https://ecology.ghislainv.fr>

ghislain.vieilledent@cirad.fr | jeanne.clement16@laposte.net