

rtkore: R and STK++ Integration using Rcpp

Serge Iovleff

November 21, 2016

Abstract

This vignette gives some hints about the usage of the `rtkore` (successor of the `rtk`) package. It explains shortly how to wrap R vectors and matrices into `STK++` structures. It gives also an example of Makevars for linking an R package with `rtkore`.

1 Introduction

`rtkore` is a versatile, fast, reliable and elegant collection of C++ classes for statistics, clustering, linear algebra (using native methods or Lapack[1]), arrays (with an Eigen-like API [2]), regression, dimension reduction, etc. Some functionalities provided by the library are available in the R environment as R functions or distributed as R packages (`MixAll` [6] and `HDPenReg` [5] among others).

The `rtkore` package provides a subset of the `stk++` library and is only composed of templated classes and inlined functions. The `rtkore` package is also available and provides the header files composing the whole `stk++` library. These packages furnish implementations of `Rcpp::as` and `Rcpp::wrap` for the C++ classes defined in `rtkore`. In this sense it is similar to the `RcppEigen` [3, 2] and `RcppArmadillo` [4] packages.

The current version of the `stk++` library is given below

```
> .Call("stk_version", FALSE, PACKAGE="rtkore")

major minor patch
  0      9      6
```

2 Wrapping R data with `STK++` arrays

`Rcpp` facilitates conversion of objects from R to C++ through the templated functions `Rcpp::as`. The function `Rcpp::as` is implemented in `rtkore` but it is not strictly necessary to use it. You can rather use this kind of code `[style=customcpp] SEXP myFunction(SEXP data) { STK::RMatrix<double> mat(data); // if data is not a matrix, Rcpp will throw an exception // }` The templated class `STK::RMatrix` wraps a `Rcpp` matrix which itself wrap the R `SEXP` structure. You can access directly (and eventually modify) the R data in your application like if it were a usual `STK++` array.

The second templated class you can use is `STK::RVector` which allows to wrap `Rcpp::NumericVector`.

3 Converting `STK++` arrays and expressions to R data

`Rcpp` facilitates data conversion from C++ to R through `Rcpp::wrap`. This function is extended by `rtkore` for the `stk++` arrays and vectors.

The following example is taken from the `STK::ClusterLauncher` class `[style=customcpp] Array2D<Real> mean(K, nbVariable), sigma(K, nbVariable); // get estimated parameters // // and save them NumericVector m_mean = Rcpp::wrap(mean); NumericVector m_sigma = Rcpp::wrap(sigma);`

Note that the `Rcpp::wrap` is rather limited in its usage and if you need, for example, to convert expression rather than arrays then you can use the `STK::wrap` function (see example below).

4 An example

The package countMissings can be downloaded at the http://sourceforge.net/projects/stkpp/files/R/20packages/countMissings_1.0.tar.gz/download url. It is basically composed of one R-script file (countNA.R) and one C++ file (countNA.cpp).

Given a R matrix, you will get a list composed of two vectors constaining respectively the number of missing values in each rows and the number of missing values in each columns of the R matrix.

The R-script countNA.R is essentially `[style=customcpp] countNA <- function(data) if (!is.matrix(data)) stop("in countNA, data must be a matrix.") .Call("countNA", data, PACKAGE = "countMissings")` and the C++ files is `[style=customcpp] include "RTKpp.h" RcppExport SEXP countNA(SEXP r_matrix) BEGIN_RCPP STK :: RMa`

5 Using random number generators

All the random numbers of R are interfaced in rtkore. You can used them like STK++ random number generators like in the following example

```
[style=customcpp] RcppExport SEXP fastBetaRand( SEXP n, SEXP alpha, SEXP beta) BEGIN_RCPP; STK ::  
RVector < double > tab(Rcpp :: as < int > (n)); STK :: Law :: Betalaw(Rcpp :: as < double > (alpha), Rcpp ::  
as < double > (beta)); tab.rand(law); returntab.vector(); END_RCPP;
```

6 Linking with

At the R level, you have to add the `LinkingTo: rtkore, Rcpp` line in the DESCRIPTION file.

At the C++ level, the only thing to do is to include the header file `[style=customcpp] // Rcpp.h` will be include by rtkore include `<RTKpp.h>` in the C++ code.

When compiling the sources, you indicate the location of the stk++ library using `rtkore:::CxxFlags()`, `rtkore:::CppFlags()` and `rtkore:::LdFlags()` in the `src/Makevars` file.

If you are building a package with a lot of cpp files, you may find convenient to locate your sources in a separate directory. Hereafter we give an example of a Makevars you can modify at your convenience in order to handle this situation.

```
#-----  
# Purpose:  Makevars for the R packages using rtkore (stk++)  
#-----  
PKGNAME    = NAME_OF_YOUR_SRC  # for example MyPackage  
PKGDIR     = PATH_TO_YOUR_SRC  # for example ./MyPackage  
PKGLIBDIR  = $(PKGDIR)/lib      # ./MyPackage/lib  
PKGLIB     = $(PKGLIBDIR)/lib$(PKGNAME).a # ./MyPackage/lib/libMyPackage.a  
  
## Use the R_HOME indirection to support installations of multiple R version.  
PKG_CXXFLAGS = `${R_HOME}/bin/Rscript -e "rtkore:::CxxFlags()"`  
PKG_CPPFLAGS = `${R_HOME}/bin/Rscript -e "rtkore:::CppFlags()"` \  
              $(SHLIB_OPENMP_CXXFLAGS)  
  
## We link the source in the src/ directory with the stkpp library and libMyPackage.a  
## use $(SHLIB_OPENMP_CFLAGS) as stkpp use openMP  
## use $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS) if you want to use lapack and/or stk++  
## wrappers of lapack  
PKG_LIBS = `${R_HOME}/bin/Rscript -e "rtkore:::LdFlags()"` $(PKGLIB) \  
$(SHLIB_OPENMP_CFLAGS) \  
              $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS)  
  
## Define any flags you may need for compiling your sources and export them  
MY_CXXFLAGS = $(PKG_CXXFLAGS)  
MY_CPPFLAGS = $(PKG_CPPFLAGS)
```

```

export

.PHONY: all pkglib

## $(SHLIB) is the usual default target that is built automatically from all source
## files in this directory. pkglib is an additional target for the package
## that will be found in $(PKGDIR).
all: $(SHLIB)
$(SHLIB): pkglib

## build the PKGLIB (lib$(PKGNAME).a)
pkglib:
(cd $(PKGDIR) && $(MAKE) all)
(cd $(PKGDIR) && $(MAKE) clean)

```

References

- [1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.
- [2] Douglas Bates and Dirk Eddelbuettel. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5):1–24, 2013.
- [3] Douglas Bates, Romain François, and Dirk Eddelbuettel. *RcppEigen: Rcpp integration for the Eigen templated linear algebra library*, 2014. R package version 0.3.2.0.2.
- [4] Romain François, Dirk Eddelbuettel, and Douglas Bates. *RcppArmadillo: Rcpp integration for Armadillo templated linear algebra library*, 2014. R package version 0.4.000.2.
- [5] Quentin Grimonprez. *HDPenReg: High-Dimensional Penalized Regression*, 2015. R package version 0.91.
- [6] Serge Iovleff. *Clustering With MixAll*, 2015. R package version 1.0.2.