rtkore: R and STK++ Integration using Rcpp

Serge Iovleff

May 2, 2025

Abstract

This vignette gives some hints about the usage of the **rtkore** (successor of the **rtkpp**) 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**. More informations can be found in the other vignettes coming with the package about the functionnalities furnished by the STK++ library.

1 Introduction

STK++ 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], blockcluster [7] 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. This package furnishes implementations of Rcpp::as and Rcpp::wrap for the C++ classes defined in STK++. 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    21
```

2 Wrapping R data with STK++ arrays

rtkore proposes two objects in order to facilitate data transfer

```
typename RVector<Type>;
typename RMatrix<Type>;
```

Rcpp facilitates conversion of objects from R to C++ through the templated functions Rcpp::as. The function Rcpp::as is re-implemented in STK++ but it is not strictly necessary to use it. You can rather use this kind of code

```
SEXP myFunction(SEXP data)
{
  // wrap a R SEXP struct with a STK++ RMatrix
  STK::RMatrix<double> mat(data); // if data is not a matrix, an exception is thrown
  // wrap a Rcpp matrix in a STK++ RMatrix
  Rcpp::NumericMatrix rmat(100,20);
  STK::RMatrix<double> mat(rmat);
  // Constructor with given dimension
  RMatrix<double> myData(100, 20);
}
```

The template 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 an usual STK++ array.

The second template class you can use is STK::RVector which allows to wrap SEXP struct.

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 STK++ arrays and vectors.

The following example is taken from the STK::ClusterLauncher class (in MixAll package)

```
  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 Using rtkore random number generators

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

```
RcppExport SEXP fastBetaRand( SEXP n, SEXP alpha, SEXP beta)
{
  BEGIN_RCPP;
  // create a STK++ RVector
  STK::RVector<double> tab(Rcpp::as<int>(n));
  // Create a Beta distribution function with alpha and beta as parameters
  STK::Law::Beta law(Rcpp::as<double>(alpha), Rcpp::as<double>(beta));
  // fill tab with random numbers
  tab.rand(law);
  // return the wrapped Rcpp vector
  return tab.vector();
  END_RCPP;
}
```

5 Linking with rtkore

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

```
// 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.

A minimal Makevars would look like

```
PKG_CXXFLAGS = `${R_HOME}/bin/Rscript -e "rtkore:::CxxFlags()"`
PKG_CPPFLAGS = `${R_HOME}/bin/Rscript -e "rtkore:::CppFlags()"` $(SHLIB_OPENMP_CXXFLAGS)
PKG_LIBS     = `$(R_HOME)/bin/Rscript -e "rtkore:::LdFlags()"` \
               $(SHLIB_OPENMP_CFLAGS) $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS)
```

6 Building huge package using rtkore

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)
```

7 An example

The package countMissings is basically composed of one R-script file (countNA.R) and one C++ file (countNA.cpp).

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

The R-script `countNA.R` is essentially

```r
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

```cpp
#include "RTKpp.h"
RcppExport SEXP countNA( SEXP r_matrix)
{
  BEGIN_RCPP
  STK::RMatrix<double> m_data(r_matrix);
  // use STK::wrap function (Rcpp::wrap function will not work)
  return Rcpp::List::create( Rcpp::Named("rows")= STK::wrap(STK::countByRow(m_data.isNA()))
                           , Rcpp::Named("cols")= STK::wrap(STK::count(m_data.isNA()))
                           );
  END_RCPP
}
```

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.

[7] Parmeet Singh Bhatia, Serge Iovleff, and Gérard Govaert. blockcluster: An R package for model-based co-clustering. *Journal of Statistical Software*, 76(9):1–24, 2017.