rtkpp: R and stk++ integration using Rcpp

Serge Iovleff

1 Introduction

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

The rtkpp package provides the header files composing the stk++ C++ library (thus users do not need to install stk++ itself in order to use rtkpp), along with implementations of Rcpp::as and Rcpp::wrap for the C++ classes defined in stk++. In this sense it is similar to the RcppEigen [2, 1] and RcppArmadillo [3] packages.

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

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

```
major minor patch
    0     8     6
```

2 Wrapping R data by stk++ arrays

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

```
Rcpp::NumericMatrix m_data = data; // data is SEXP representing a R-matrix
STK::RMatrix<double> mat(m_data); // STK::RMatrix<double> mat(data) will work
```

The templated class STK::RMatrix wrap the Rcpp matrix (which itself wrap the R SEXP structure). You can access directly (and eventually modify) the R data in your application.

The second templated class you can use is the STK::RVector which allow 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 rtkpp for the stk++ arrays and vectors.

The following example is taken from the STK::ClusterLauncher class

```
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 www.stkpp.org url. It is basically composed of one R-script file (countNA.R) and one C++ file (countNA.cpp).

Given a R matrix, it is possible to 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 looks

```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 looks

```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
}
```

5 Linking with `rtkpp`

The only thing to do is to include the header file

```cpp
// Rcpp.h will be include by rtkpp
#include <RTKpp.h>
```

in the code. When compiling the sources, you indicate the location of the stk++ library using `rtkpp:::CxxFlags()`, `rtkpp:::CppFlags()` and `rtkpp:::LdFlags()` in package `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 rtkpp (stk++)
#-------------------------------------------------------------------------
PKGNAME = NAME_OF_YOUR_PACKAGE

PKGDIR    = FULL_PATH_TO_YOUR_PACKAGE
PKGLIBDIR = $(PKGDIR)/lib
PKGLIB    = $(PKGLIBDIR)/lib$(PKGNAME).a

## Use the R_HOME indirection to support installations of multiple R version.
## use $(SHLIB_OPENMP_CXXFLAGS) if you want openmp.
## It is not necessary to use Rcpp:::CxxFlags() if there is already
## LinkingTo: Rcpp in your DESCRIPTION file
PKG_CXXFLAGS = `${R_HOME}/bin/Rscript -e "Rcpp:::CxxFlags()"` \
               `${R_HOME}/bin/Rscript -e "rtkpp:::CxxFlags()"`

PKG_CPPFLAGS = `${R_HOME}/bin/Rscript -e "rtkpp:::CppFlags()"` \
                $(SHLIB_OPENMP_CXXFLAGS)

## use $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS) if you want lapack
PKG_LIBS = `${R_HOME}/bin/Rscript -e "Rcpp:::LdFlags()"` \
           `$(R_HOME)/bin/Rscript -e "rtkpp:::LdFlags()"` \
            $(SHLIB_OPENMP_CFLAGS) $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS) $(PKGLIB)

## Define any flags you need for compiling your sources
PKGNAME_CXXFLAGS = $(PKG_CXXFLAGS)
PKGNAME_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] Douglas Bates and Dirk Eddelbuettel. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5):1–24, 2013.

[2] 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.

[3] Romain François, Dirk Eddelbuettel, and Douglas Bates. *RcppArmadillo: Rcpp integration for Armadillo templated linear algebra library*, 2014. R package version 0.4.000.2.