Package 'RcppClassicExamples'

November 30, 2023

Title Examples using 'RcppClassic' to Interface R and C++

Version 0.1.3

Date 2024-11-30

Author Dirk Eddelbuettel and Romain Francois, based on code written during 2005 and 2006 by Dominick Samperi

Maintainer Dirk Eddelbuettel <edd@debian.org>

Description The 'Rcpp' package contains a C++ library that facilitates the integration of R and C++ in various ways via a rich API. This API was preceded by an earlier version which has been deprecated since 2010 (but is still supported to provide backwards compatibility in the package 'RcppClassic'). This package 'RcppClassicExamples' provides usage examples for the older, deprecated API. There is also a corresponding package 'RcppExamples' with examples for the newer, current API which we strongly recommend as the basis for all new development.

Depends R (>= 2.15.2)

Imports Rcpp (>= 0.10.2), RcppClassic (>= 0.9.3)

LinkingTo Rcpp, RcppClassic

Suggests RUnit

URL https://github.com/eddelbuettel/rcppclassicexamples, https://dirk.eddelbuettel.com/code/rcpp.classic.html

BugReports https://github.com/eddelbuettel/rcppclassicexamples/issues

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-11-30 14:10:02 UTC

R topics documented:

`RcppClassicExamples-package`
Examples for the deprecated Rcpp R/C++ Interface library API

Description

This package shows some simple examples for the use of the deprecated *classic* API from the first implementation of **Rcpp**.

Note that the **RcppClassic** package has been deprecated since 2010, all new development should use the **Rcpp** package instead.

Details

The **Rcpp** package provides a number of C++ classes that ease access to C++ from R. This comprises both passing parameters to functions, as well as returning results back from C++ to R.

Two APIs are supported. The first is an older API which was first introduced mostly in 2006 and extended in 2008. This interface is used by a few other packages and will be supported going forward in the RcppClassic-package package.

A second and newer API that was started in 2009 offers more functionality, see the Rcpp-package package.

The **RcppExamples** package provides some simple examples for use of **Rcpp**.

Author(s)

Dominick Samperi wrote the initial versions of Rcpp (and RcppTemplate) during 2005 and 2006. Dirk Eddelbuettel made some additions, and became maintainer in 2008. Dirk Eddelbuettel and Romain Francois have been extending Rcpp since 2009.

See Also

See the RcppExamples-package for examples of the recommended **Rcpp** API and Rcpp-package for documentation on the recommended API to extend R with C++ code, while the deprecated RcppClassic-package documents the older, deprecated API.

RcppDataFrame *Rcpp::DataFrame example for Rcpp – deprecated API*

Description

A DataFrame can be passed C++ and can be instantiated as a corresponding C++ object using the Rcpp API.

This example shows (in the corresponding C++ code) how to access, modify and create a data frame.

Note that the **RcppClassic** package has been deprecated since 2010, all new development should use the **Rcpp** package instead.

Details

Usage of Rcpp::DataFrame is fully defined in the respective header file.

The C++ source file corresponding to the this function does the following (inside of a try/catch block):

```
      // construct the data.frame object
      Rcpp::DataFrame DF = Rcpp::DataFrame(Dsexp);

      // and access each column by name
      Rcpp::IntegerVector a = DF["a"];
      Rcpp::CharacterVector b = DF["b"];
      Rcpp::DateVector c = DF["c"];

      // do something
      a[2] = 42;
      b[1] = "foo";
      c[0] = c[0] + 7;                      // move up a week

      // create a new data frame
      Rcpp::DataFrame NDF =
  Rcpp::DataFrame::create(Rcpp::Named("a")=a,
  Rcpp::Named("b")=b,
  Rcpp::Named("c")=c);

      // and return old and new in list
      return(Rcpp::List::create(Rcpp::Named("origDataFrame")=DF,
Rcpp::Named("newDataFrame")=NDF));
```

Author(s)

Dirk Eddelbuettel and Romain Francois

See Also

See the RcppExamples-package for examples of the recommended **Rcpp** API and Rcpp-package for documentation on the recommended API to extend R with C++ code, while the deprecated RcppClassic-package documents the older, deprecated API.

Examples

```
  ## Not run:
  RcppDataFrame()

## End(Not run)
```

`RcppDate` *C++ classes for interfacing date and datetime R objects – deprecated API*

Description

`RcppDate`, `RcppDatetime`, `RcppDateVector` and `RcppDatetimeVector` are C++ classes defined in their respective headers files. They are part of the 'classic' Rcpp API. These classes pass scalars and vectors of R objects of types `Date` and `POSIXct`, respectively, to C++ via the `.Call()` function interface.

Member functions are provided to query the dimension of the vector or matrix object, convert it in a corresponding `C` representation.

R objects of type `Date`, and hence the `RcppDate` and `RcppDateVector` objects, are internally represented as an integer counting days since the epoch, i.e. January 1, 1970. Similarly, R objects of type `POSIXct` and the `RcppDatetime` and `RcppDatetimeVector` objects, are internally represented as seconds since the epoch. However, R extends the POSIX standard by using a double leading to microsecond precision in timestamps. This is fully supported by `Rcpp` as well.

The new API currently has the classes `Rcpp::Date`, `Rcpp::Datetime`, `Rcpp::DateVector` and `Rcpp::DatetimeVector` which are preferred for new developments, as is the rest of the new API in the **Rcpp** package while the **RcppClassic** package has been deprecated since 2010.

Details

Usage of the `RcppDate`, `RcppDatetime` (and their vector extensions) in `C++` is fully defined in the respective header files `RcppDate.h` and `RcppDatetime.h`.

As example, consider a call from R to `C++` such as

```
# an R example passing one type of each class to a function
# someFunction in package somePackage
val <- .Call("someFunction",
             Sys.Date(),      # current date
         Sys.time(),      # current timestamp
         as.Date("2000-02-25")
           + 0:5,         # date vector
```

```
            ISOdatetime(1999,12,31,23,59,0)
              + (0:5)*0.250, # datetime vector
                PACKAGE="somePackage")
```

At the C++ level, the corresponding code to assign these parameter to C++ objects is can be as follows::

```
SEXP someFunction(SEXP ds, SEXP dts,
                  SEXP dvs, SEXP dtvs) {

  RcppDate           d(ds);
  RcppDatetime       dt(dts);
  RcppDateVector     dv(dvs);
  RcppDatetimeVector dtv(dtvs);
}
```

Standard accessor functions are defined, see `RcppDate.h` and `RcppDatetime.h` for details.

Objects of these types can also be returned via `RcppResultSet`.

Author(s)

Dominick Samperi wrote the initial versions of Rcpp (and RcppTemplate) during 2005 and 2006. Dirk Eddelbuettel made some additions, and became maintainer in 2008. Dirk Eddelbuettel and Romain Francois have been extending Rcpp since 2009.

References

Writing R Extensions, available at https://www.r-project.org.

See Also

`RcppResultSet`.

See the RcppExamples-package for examples of the recommended **Rcpp** API and Rcpp-package for documentation on the recommended API to extend R with C++ code, while the deprecated RcppClassic-package documents the older, deprecated API.

Examples

```
# set up date and datetime vectors
dvec <- Sys.Date() + -2:2
dtvec <- Sys.time() + (-2:2)*0.5

# call the underlying  C++ function
result <- RcppDateExample(dvec, dtvec)

# inspect returned object
result
```

RcppExample *Rcpp R / C++ interface example – deprecated API*

Description

RcppExample illustrates how the older Rcpp R/C++ interface class library is used. It provides fairly complete coverage for the older 'classic' API.

Note that the **RcppClassic** package has been deprecated since 2010, all new development should use the **Rcpp** package instead.

Usage

```
RcppExample(params, nlist, numvec, nummat, df, datevec, stringvec, fnvec, fnlist)
## S3 method for class 'RcppExample'
print(x,...)
```

Arguments

params	A heterogeneous list specifying method (string), tolerance (double), maxIter (int).
nlist	a list of named numeric values (double or int).
numvec	a numeric 1D vector (double or int).
nummat	a numeric 2D matrix (double or int).
df	a data frame.
datevec	a vector of Date's.
stringvec	a vector of strings.
fnvec	an R function with numeric vector argument.
fnlist	an R function with list argument.
x	Object of type RcppExample.
...	Extra named parameters.

Details

The C++ represention of data frames are not passed back to R in a form that R recognizes as a data frame, but it is a simple matter to do the conversion. For example, the return value named PreDF (see return values below) is not seen as a data frame on the R side (thus the name "pre-data frame"), but it can be converted to a data frame using `df <- data.frame(result$PreDF)`.

The print.RcppExample() function is defined so that we can control what gets printed when a variable assigned the return value is entered on a line by itself. It is defined to simply list the names of the fields returned (see RcppExample.R).

Value

RcppExample returns a list containing:

method	string input paramter
tolerance	double input paramter
maxIter	int input parameter
nlFirstName	first name in nlist
nlFirstValue	first value in nlist
matD	R matrix from an RcppMatrix<double> object
stlvec	R vector from a vector<double> object
stlmat	R matrix from a vector<vector<double> > object
a	R matrix from C/C++ matrix
v	R vector from C/C++ vector
strings	R vector of strings from vector<string> object
InputDF	a data frame passed in from R
PreDF	a data frame created on C++ side to be passed back to R
params	input parameter list (this is redundant because we returned the input parameters above)

Author(s)

Dominick Samperi wrote the initial versions of Rcpp (and RcppTemplate) during 2005 and 2006. Dirk Eddelbuettel made some additions, and became maintainer in 2008. Dirk Eddelbuettel and Romain Francois have been extending Rcpp since 2009.

References

Writing R Extensions, available at https://www.r-project.org.

See Also

See the RcppExamples-package for examples of the recommended **Rcpp** API and Rcpp-package for documentation on the recommended API to extend R with C++ code, while the deprecated RcppClassic-package documents the older, deprecated API.

Examples

```
params <- list(method='BFGS',
               tolerance=1.0e-8,
               maxIter=1000,
               startDate=as.Date('2006-7-15'))

nlist <- list(ibm = 80.50, hp = 53.64, c = 45.41)

numvec <- seq(1,5) # numerical vector
```

```
nummat <- matrix(seq(1,20),4,5) # numerical matrix

stringvec <- c("hello", "world", "fractal") # string vector

datestr <- c('2006-6-10', '2006-7-12', '2006-8-10')
datevec <- as.Date(datestr, "%Y-%m-%d") # date vector

df <- data.frame(a=c(TRUE, TRUE, FALSE), b=I(c('a','b','c')),
c=c('beta', 'beta', 'gamma'), dates=datevec)

fnvec <- function(x) { sum(x) } # Add up components of vector

fnlist <- function(l) { # Return vector with 1 added to each component
  vec <- c(l$alpha + 1, l$beta + 1, l$gamma + 1)
  vec
}

result <- RcppExample(params, nlist, numvec, nummat, df, datevec,
                      stringvec, fnvec, fnlist)

result
```

RcppParams *C++ class for receiving (scalar) parameters from R – deprecated API*

Description

RcppParams is a C++ class defined in Rcpp.h that receive any number of scalar parameters of types in a single named list object from R through the .Call() function.

The parameters can be of different types that are limited to the R types numeric, integer, character, logical or Date. These types are mapped into, respectively, the corresponding C++ types double, int, string, bool and Date (a custom class defined by Rcpp.

RcppParams is part of the old deprecated Rcpp API, and should be replaces by Rcpp::List which is more flexible and can be used for both inputs and outputs. RcppParams is retained for backwards compatibility, but should be avoided in new projects and replaced in old projects.

Note that the **RcppClassic** package has been deprecated since 2010, all new development should use the **Rcpp** package instead.

Arguments

params	A heterogeneous list specifying method (string), tolerance (double), maxIter (int) and startDate (Date in R, RcppDate in C++).

Details

Usage of RcppParams from R via .Call() is as follows:

```
# an R example passing one type of each class to a function
# someFunction in package somePackage
val <- .Call("someFunction",
             list(pie=3.1415, magicanswer=42, sometext="foo",
                  yesno=true, today=Sys.date()),
  PACKAGE="somePackage")
```

At the C++ level, the corresponding code to assign these parameter to C++ objects is

```
SEXP someFunction(SEXP params) {
  RcppParams par(params);
  double p   = par.getDoubleValue("pie");
  int magic  = par.getIntValue("magicanswer");
  string txt = par.getStringValue("sometext");
  bool yn    = par.getBoolValue("yesno");
  RcppDate d = par.getDateValue("today");
  // some calculations ...
  // some return values ...
}
```

As the lookup is driven by the names givem at the R level, order is not important. It is however important that the types match. Errors are typically caught and an exception is thrown.

The class member function checkNames can be used to verify that the SEXP object passed to the function contains a given set of named object.

Value

RcppExample returns a list containing:

method	string input paramter
tolerance	double input paramter
maxIter	int input parameter
startDate	Date type with starting date
params	input parameter list (this is redundant because we returned the input parameters above)

Author(s)

Dominick Samperi wrote the initial versions of Rcpp (and RcppTemplate) during 2005 and 2006. Dirk Eddelbuettel made some additions, and became maintainer in 2008. Dirk Eddelbuettel and Romain Francois have been extending Rcpp since 2009.

References

Writing R Extensions, available at https://www.r-project.org.

See Also

RcppExample.

See the RcppExamples-package for examples of the recommended **Rcpp** API and Rcpp-package for documentation on the recommended API to extend R with C++ code, while the deprecated RcppClassic-package documents the older, deprecated API.

Examples

```
# set up some value
params <- list(method='BFGS',
               tolerance=1.0e-5,
               maxIter=100,
               startDate=as.Date('2006-7-15'))

# call the underlying  C++ function
result <- RcppParamsExample(params)

# inspect returned object
result
```

RcppResultSet *C++ class for sending C++ objects back to R – deprecated API*

Description

RcppResultSet is a C++ class defined in RcppResultSet.h that can assign any number of C++ objects to R in a single named list object as the SEXP return value of a .Call() function call. It is part of the classic API.

The C++ objects can be of different types that are limited to types double, int, string, vectors of double or int (with explicit dimensions), matrices of double or int (with explicit dimensions), STL vectors of double, int or string, STL 'vector of vectors' of types double or int (all with implicit dimensions), the internal types RcppDate, RcppDateVector, RcppStringVector, RcppVector of types double or int, RcppMatrix of types double or int as well RcppFrame, a type that can be converted into a data.frame, and the R type SEXP.

Where applicable, the C++ types are automatically converted to the corresponding R types structures around types numeric, integer, or character. The C++ code can all be retrieved in R as elements of a named list object.

The new API has more generic templated functions.

Note that the **RcppClassic** package has been deprecated since 2010, all new development should use the **Rcpp** package instead.

Details

Usage of RcppResultSet from C++ is fully defined in RcppResultSet.h. An example for returning data to R at the end of a .Call() call follows.

At the C++ level, the corresponding code to assign these parameter to C++ objects is can be as follows (taken from the C++ source of RcppExample):

```
  SEXP rl;
  RcppResultSet rs;

  rs.add("date", aDate); // RcppDate
  rs.add("dateVec", dateVec); // RcppDateVec
  rs.add("method", method); // string
  rs.add("tolerance", tol); // numeric
  rs.add("maxIter", maxIter); // int
  rs.add("matD", matD); // RcppMatrix
  rs.add("stlvec", stlvec); // vector<double> or <int>
  rs.add("stlmat", stlmat); // vector< vector <double> >
   //   or <int>
  rs.add("a", a, nrows, ncols); // double** (or int**) with
//   two dimension
  rs.add("v", v, len); // double* (or int*) with
//   one dimension
  rs.add("stringVec", strVec); // RcppStringVector
  rs.add("strings", svec); // vector<string>
  rs.add("InputDF", inframe); // RcppFrame
  rs.add("PreDF", frame); // RcppFrame

  rl = rs.getReturnList();
  return(rl);
```

As the R level, we assign the returned object a list variables from which we select each list element by its name. lookup is driven by the names givem at the R level, order is not important. It is however important that the types match. Errors are typically caught and an exception is thrown.

The class member function checkNames can be used to verify that the SEXP object passed to the function contains a given set of named object.

Author(s)

Dominick Samperi wrote the initial versions of Rcpp (and RcppTemplate) during 2005 and 2006. Dirk Eddelbuettel made some additions, and became maintainer in 2008. Dirk Eddelbuettel and Romain Francois have been extending Rcpp since 2009.

See Also

RcppExample.

See the RcppExamples-package for examples of the recommended **Rcpp** API and Rcpp-package for documentation on the recommended API to extend R with C++ code, while the deprecated RcppClassic-package documents the older, deprecated API.

Examples

```
# example from RcppDate
# set up date and datetime vectors
dvec <- Sys.Date() + -2:2
dtvec <- Sys.time() + (-2:2)*0.5

# call the underlying  C++ function
result <- RcppDateExample(dvec, dtvec)

# inspect returned object
result
```

RcppVector *C++ classes for receiving R object in C++ – deprecated API*

Description

RcppVector, RcppMatrix and RcppStringVector are C++ classes that can pass vectors (matrices) of R objects of appropriate types to C++ via the .Call() function interface. They are part of the 'classic' Rcpp API.

The vector and matrix types are templated and can operate on R types intger and numeric.

The RcppVectorView and RcppMatrixView are slighly more lightweight read-only variants.

Member functions are provided to query the dimension of the vector or matrix object, convert it in a corresponding C representation, and also to convert it into a corresponding STL object.

The new API has classes NumericVector, NumericMatrix, CharacterVector (and also an alias StringVector).

The files RcppVectorExample.cpp and RcppMatrixExample.cpp provide examples for both the classic and new APIs.

Note that the **RcppClassic** package has been deprecated since 2010, all new development should use the **Rcpp** package instead.

Details

Usage of RcppVector, RcppMatrix and RcppStringVector in C++ is fully defined in the respective header files.

As example, consider a call from R to C++ such as

```
# an R example passing one type of each class to a function
# someFunction in package somePackage
val <- .Call("someFunction",
```

```
        rnorm(100),        # numeric vector
    sample(1:10, 5, TRUE)  # int vector
    search(),          # character vector
    as.matrix(rnorm(100),10,10), # matrix
        PACKAGE="somePackage")
```

At the C++ level, the corresponding code to assign these parameter to C++ objects is can be as follows (taken from the C++ source of RcppExample):

```
SEXP someFunction(SEXP nvec, SEXP ivec,
                  SEXP svec, SEXP nmat) {

  RcppVector<double> nv(nvec);
  RcppVector<int>    iv(ivec);
  RcppStringVector   sv(svec);
  RcppMatrix<double> nm(nmat);
}
```

These C++ objects could then be queried via

```
int n = nv.size();
int d1 = nm.dim1(), d2 = nm.dim2();
```

to retrieve, respectively, vector length and matrix dimensions.

Moreover, the `stlVector()` and `stlMatrix()` member functions can be used to convert the objects into STL objects:

```
vector<int> ivstl = iv.stlVector();
vector< vector< double > > = nm.stlMatrix();
```

Author(s)

Dominick Samperi wrote the initial versions of Rcpp (and RcppTemplate) during 2005 and 2006. Dirk Eddelbuettel made some additions, and became maintainer in 2008. Dirk Eddelbuettel and Romain Francois have been extending Rcpp since 2009.

See Also

RcppExample.

See the RcppExamples-package for examples of the recommended **Rcpp** API and Rcpp-package for documentation on the recommended API to extend R with C++ code, while the deprecated RcppClassic-package documents the older, deprecated API.

Examples

```
# set up some value
vector <- (seq(1,9))^2

# call the underlying  C++ function
result <- RcppVectorExample(vector)

# inspect returned object
result
```

Index