Package 'RcppExamples'

July 2, 2014

Title Examples using Rcpp to interface R and C++

Version 0.1.6

Date $Date: 2013-01-15 11:25:09 -0600 (Tue, 15 Jan 2013) $

Author Dirk Eddelbuettel and Romain Francois

Maintainer Dirk Eddelbuettel <edd@debian.org>

Description Examples for Seamless R and C++ integration The Rcpp package contains a C++ library that facilitates the integration of R and C++ in various ways. This package provides some usage examples.

Note that the documentation in this package currently does not cover all the features in the package. It is not even close. On the other hand, the site http://gallery.rcpp.org is regrouping a number of examples for Rcpp.

Depends R (>= 2.11.0), Rcpp (>= 0.9.9)

LinkingTo Rcpp

Suggests RUnit

URL http://dirk.eddelbuettel.com/code/rcpp.html,http://romainfrancois.blog.free.fr/index.php?category/R-package/Rcpp

License GPL (>= 2)

Repository CRAN

Date/Publication 2013-01-16 09:21:56

NeedsCompilation yes

R topics documented:

`RcppExamples-package` *Examples for the Rcpp R/C++ Interface library*

Description

This package shows some simple examples for the use of **Rcpp**.

It can also serve as a working template to create packages that use **Rcpp** to interface C++ code or libraries.

A sibbling package **RcppClassicExamples** provides some examples for the deprecated older API still being made available via the **RcppClassic** package. New development should use **Rcpp** 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, extended in 2008 and deprecated in 2009. This interface is used by a few other packages and will be supported going forward, but not extended. A second and newer API was started in 2009 offers more functionality: **Rcpp**

The **RcppExamples** package provides some simple examples for use of **Rcpp**. At this point the documentation is not complete in the sense of not covering all accessible classes. However, several basic use cases are illustrated,

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

The `http://gallery.rcpp.org` site regroups a number of examples.

RcppDataFrame *Rcpp::DataFrame example for Rcpp*

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.

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

Examples

```
  ## Not run:
  RcppDataFrame()

## End(Not run)
```

RcppDate *C++ classes for interfacing date and datetime R objects*

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, but currently less documented.

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 http:www.r-project.org.

See Also

`RcppResultSet`.

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
```

`RcppParams` *C++ class for receiving (scalar) parameters from R*

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 Rcpp API, and should be replace 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.

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 http:www.r-project.org.

See Also

RcppExample.

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*

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.

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.

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
```

RcppRNGs *Rcpp RNGs example*

Description

Rcpp sugar provides numerous p/q/d/r functions for numerous distributions.

This example shows (in the corresponding C++ code) how to draw from three different distributions and returns a data frame.

Details

The various header file, and the Rcpp sugar vignette, provide full documentation for Rcpp sugar.

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

```
Rcpp::RNGScope scope;          // needed when RNGs are drawn

int n = Rcpp::as<int>(ns);  // length vector
Rcpp::NumericVector rn = Rcpp::rnorm(n);
Rcpp::NumericVector rt = Rcpp::rt(n, 1.0);
Rcpp::NumericVector rp = Rcpp::rpois(n, 1.0);

// create a new data frame to return drawns
Rcpp::DataFrame NDF =
    Rcpp::DataFrame::create(Rcpp::Named("rnorm") =rn,
                            Rcpp::Named("rt")    =rt,
                            Rcpp::Named("rpois") =rp);

// and return old and new in list
return(NDF);
```

As shown in the example section, provided the seed is reset, the exact same draws can be obtained in R itself – which is important for reproducibility.

Author(s)

Dirk Eddelbuettel and Romain Francois

Examples

```
set.seed(42)
X <- RcppRNGs(10)
set.seed(42)
Y <- data.frame(rnorm=rnorm(10),rt=rt(10,1),rpois=rpois(10,1))
all.equal(X,Y)
```

RcppVector *C++ classes for receiving R object in C++*

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.

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`.

Examples

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

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

# inspect returned object
result
```

Index