

Package ‘Rcpp’

January 2, 2012

Title Seamless R and C++ Integration

Version 0.9.9

Date \$Date: 2011-12-25 14:14:33 -0600 (Sun, 25 Dec 2011) \$

Author

Dirk Eddelbuettel and Romain Francois, with contributions by Douglas Bates and John Chambers

Maintainer Dirk Eddelbuettel and Romain Francois <RomainAndDirk@r-enthusiasts.com>

Description The Rcpp package provides a C++ library which facilitates the integration of R and C++.

R data types (SEXP) are matched to C++ objects in a class hierarchy. All R types are supported (vectors, functions, environment, etc ...) and each type is mapped to a dedicated class. For example, numeric vectors are represented as instances of the Rcpp::NumericVector class, environments are represented as instances of Rcpp::Environment, functions are represented as Rcpp::Function, etc ... The ‘‘Rcpp-introduction’’ vignette provides a good entry point to Rcpp.

Conversion from C++ to R and back is driven by the templates Rcpp::wrap and Rcpp::as which are highly flexible and extensible, as documented in the ‘‘Rcpp-extending’’ vignette.

Rcpp also provides Rcpp modules, a framework that allows exposing C++ functions and classes to the R level. The ‘‘Rcpp-modules’’ vignette details the current set of features of Rcpp-modules.

Rcpp includes a concept called Rcpp sugar that brings many R functions into C++. Sugar takes advantage of lazy evaluation and expression templates to achieve great performance while exposing a syntax that is much nicer to use than the equivalent low-level loop code. The ‘‘Rcpp-sugar’’ vignette gives an overview of the feature.

Several examples are included, and more than 750 unit tests in over 330 unit test functions provide additional usage examples.

An earlier version of Rcpp, containing what we now call the ‘classic Rcpp API’ was written during 2005 and 2006 by Dominick Samperi. This code has been factored out of Rcpp into the package RcppClassic, and it is still available for code relying on the older interface. New development, however, should use the Rcpp package instead.

Depends R (>= 2.12.0), methods

Suggests RUnit, inline, rbenchmark

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

License GPL (>= 2)

BugReports http://r-forge.r-project.org/tracker/?atid=637&group_id=155&func=browse

MailingList Please send questions and comments regarding Rcpp to
rcpp-devel@lists.r-forge.r-project.org

Repository CRAN

Date/Publication 2011-12-27 10:05:58

R topics documented:

Rcpp-package	3
.DollarNames-methods	4
C++Class-class	4
C++ClassRepresentation-class	5
C++Constructor-class	6
C++Field-class	6
C++Function-class	7
C++Object-class	7
C++ObjectS3-class	8
C++OverloadedMethods-class	8
formals<-methods	9
loadRcppModules	9
Module	10
Module-class	10
populate	11
Rcpp.package.skeleton	11
RcppUnitTests	13
setRcppClass	14
Index	16

Rcpp-package

R / C++ interface

Description

The **Rcpp** package provides C++ classes that greatly facilitate interfacing C or C++ code in R packages using the `.Call` interface provided by R.

Introduction

Rcpp provides C++ classes to facilitate manipulation of a large number of R data structures : vectors, functions, environments, ...

The "Rcpp-introduction" vignette gives an introduction on the package

Usage for package building

The "Rcpp-package" vignette documents how to use Rcpp in client packages.

History

The initial versions of Rcpp were written by Dominick Samperi 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.

Author(s)

Dirk Eddelbuettel and Romain Francois

References

Dirk Eddelbuettel and Romain Francois (2011). **Rcpp**: Seamless R and C++ Integration. *Journal of Statistical Software*, **40(8)**, 1-18. URL <http://www.jstatsoft.org/v40/i08/> and available as vignette("Rcpp-introduction").

Examples

```
## Not run:
# introduction to Rcpp
vignette( "Rcpp-introduction" )

# information on how to build a package that uses Rcpp
vignette( "Rcpp-package" )

## End(Not run)
```

`.DollarNames-methods` *completion*

Description

completion

Methods

signature(x = "ANY")

signature(x = "C++Object") completes fields and methods of C++ objects

signature(x = "Module") completes functions and classes of modules

C++Class-class *Reflection information for an internal c++ class*

Description

Information about an internal c++ class.

Objects from the Class

Objects are usually extracted from a [Module](#) using the dollar extractor.

Slots

.Data: mangled name of the class

pointer: external pointer to the internal information

module: external pointer to the module

fields: list of [C++Field](#) objects

constructors: list of [C++Constructor](#) objects

methods: list of [C++OverloadedMethods](#) objects

generator the generator object for the class

docstring description of the class

typeid unmangled typeid of the class

Methods

show signature(object = "C++Class"): prints the class.

\$ signature(object = "C++Class"): ...

C++ClassRepresentation-class
Class "C++ClassRepresentation"

Description

Representation of a C++ class

Slots

pointer: Object of class "externalptr" ~~
generator: Object of class "refObjectGenerator" ~~
cpp_fields: Object of class "list" ~~
cpp_methods: Object of class "list" ~~
cpp_constructor: Object of class "list" ~~
slots: Object of class "list" ~~
contains: Object of class "list" ~~
virtual: Object of class "logical" ~~
prototype: Object of class "ANY" ~~
validity: Object of class "OptionalFunction" ~~
access: Object of class "list" ~~
className: Object of class "character" ~~
package: Object of class "character" ~~
subclasses: Object of class "list" ~~
versionKey: Object of class "externalptr" ~~
sealed: Object of class "logical" ~~

Extends

Class "[classRepresentation](#)", directly.

Methods

referenceMethods signature(classDef = "C++ClassRepresentation"): ...

Examples

```
showClass("C++ClassRepresentation")
```

C++Constructor-class *Class "C++Constructor"*

Description

Representation of a C++ constructor

Extends

Class "[envRefClass](#)", directly. Class "[.environment](#)", by class "envRefClass", distance 2. Class "[refClass](#)", by class "envRefClass", distance 2. Class "[environment](#)", by class "envRefClass", distance 3, with explicit coerce. Class "[refObject](#)", by class "envRefClass", distance 3.

Fields

pointer: pointer to the internal structure that represent the constructor
class_pointer: pointer to the internal structure that represent the associated C++ class
nargs: Number of arguments the constructor expects
signature: C++ signature of the constructor
docstring: Short description of the constructor

C++Field-class *Class "C++Field"*

Description

Metadata associated with a field of a class exposed through Rcpp modules

Fields

pointer: external pointer to the internal (C++) object that represents fields
cpp_class: (demangled) name of the C++ class of the field
read_only: Is this field read only
class_pointer: external pointer to the class this field is from.

Methods

No methods defined with class "C++Field" in the signature.

See Also

The fields slot of the [C++Class](#) class is a list of C++Field objects

Examples

```
showClass("C++Field")
```

C++Function-class *Class "C++Function"*

Description

Internal C++ function

Objects from the Class

Objects can be created by the Rcpp: :InternalFunction class from the Rcpp library

Slots

.Data: R function that calls back to the internal function

pointer: External pointer to a C++ object pointing to the function

docstring: Short documentation for the function

signature: C++ signature

Extends

Class "[function](#)", from data part. Class "[OptionalFunction](#)", by class "function", distance 2.

Class "[PossibleMethod](#)", by class "function", distance 2.

Methods

show signature(object = "C++Function"): print the object

Examples

```
showClass("C++Function")
```

C++Object-class *c++ internal objects*

Description

C++ internal objects instantiated from a class exposed in an Rcpp module

Objects from the Class

This is a virtual class. Actual C++ classes are subclasses.

Methods

\$ signature(x = "C++Object"): invokes a method on the object, or retrieves the value of a property

\$<- signature(x = "C++Object"): set the value of a property

show signature(object = "C++Object"): print the object

C++ObjectS3-class *Class "C++ObjectS3"*

Description

deprecated class

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "C++ObjectS3" in the signature.

Examples

```
showClass("C++ObjectS3")
```

C++OverloadedMethods-class
Class "C++OverloadedMethods"

Description

Set of C++ methods

Extends

Class "[envRefClass](#)", directly. Class "[.environment](#)", by class "envRefClass", distance 2. Class "[refClass](#)", by class "envRefClass", distance 2. Class "[environment](#)", by class "envRefClass", distance 3, with explicit coerce. Class "[refObject](#)", by class "envRefClass", distance 3.

Fields

pointer: Object of class externalptr pointer to the internal structure that represents the set of methods

class_pointer: Object of class externalptr pointer to the internal structure that models the related class

formals<--methods	<i>Set the formal arguments of a C++ function</i>
-------------------	---------------------------------------------------

Description

Set the formal arguments of a C++ function

Methods

signature(fun = "C++Function") Set the formal arguments of a C++ function

loadRcppModules	<i>Loads Rcpp modules on package startup</i>
-----------------	----------------------------------------------

Description

Function to simplify loading Rcpp modules contained in a package. This function must be called from the `.onLoad` function of a package. It uses the `RcppModules` field of the package DESCRIPTION file to query the names of the modules that the package should export, loads each module, and [populate](#) each module into the package NAMESPACE.

Usage

```
loadRcppModules(direct=TRUE)
```

Arguments

direct	if TRUE the content of the module is exposed in the namespace. Otherwise, the module is exposed.
--------	--------------------------------------------------------------------------------------------------

See Also

[populate](#)

Module	<i>Retrieves an Rcpp module</i>
--------	---------------------------------

Description

Retrieves an Rcpp module from a dynamic library, usually associated with a package.

Usage

```
Module(module, PACKAGE = getPackageName(where), where = toplev(parent.frame()), mustStart = FALSE )
```

Arguments

module	Name of the module, as declared in the RCPP_MODULE macro internally
PACKAGE	Passed to getNativeSymbolInfo
where	When the module is loaded, S4 classes are defined based on the internal classes. This argument is passed to setClass
mustStart	TODO

Value

An object of class [Module](#) collecting functions and classes declared in the module.

Module-class	<i>Rcpp modules</i>
--------------	---------------------

Description

Collection of internal c++ functions and classes exposed to R

Objects from the Class

modules are created by the `link{Module}` function

Methods

\$ `signature(x = "Module")`: extract a function or a class from the module.

prompt `signature(object = "Module")`: generates skeleton of a documentation for a Module.

show `signature(object = "Module")`: summary information about the module.

initialize `signature(.Object = "Module")`: ...

See Also

The [Module](#) function

populate	<i>Populates a namespace or an environment with the content of a module</i>
----------	-----------------------------------------------------------------------------

Description

Populates a namespace or an environment with the content of a module

Usage

```
populate(module, env)
```

Arguments

module	Rcpp module
env	environment or namespace

Rcpp.package.skeleton	<i>Create a skeleton for a new package depending on Rcpp</i>
-----------------------	--------------------------------------------------------------

Description

Rcpp.package.skeleton automates the creation of a new source package that intends to use features of Rcpp.

It is based on the [package.skeleton](#) function which it executes first.

Usage

```
Rcpp.package.skeleton(name = "anRpackage", list = character(),
  environment = .GlobalEnv, path = ".", force = FALSE,
  namespace = TRUE, code_files = character(),
  example_code = TRUE, module = FALSE,
  author = "Who wrote it",
  maintainer = if(missing( author)) "Who to complain to" else author,
  email = "yourfault@somewhere.net",
  license = "What Licence is it under ?"
)
```

Arguments

name	See package.skeleton
list	See package.skeleton
environment	See package.skeleton
path	See package.skeleton
force	See package.skeleton

namespace	See package.skeleton
code_files	See package.skeleton
example_code	If TRUE, example c++ code using Rcpp is added to the package.
module	If TRUE, an example Module is added to the skeleton.
author	Author of the package.
maintainer	Maintainer of the package.
email	Email of the package maintainer.
license	License of the package.

Details

In addition to [package.skeleton](#) :

The ‘DESCRIPTION’ file gains a Depends line requesting that the package depends on Rcpp and a LinkingTo line so that the package finds Rcpp header files.

The ‘NAMESPACE’, if any, gains a useDynLib directive.

The ‘src’ directory is created if it does not exist and a ‘Makevars’ file is added setting the environment variables ‘PKG_LIBS’ to accommodate the necessary flags to link with the Rcpp library.

If the example_code argument is set to TRUE, example files ‘rcpp_hello_world.h’ and ‘rcpp_hello_world.cpp’ are also created in the ‘src’. An R file ‘rcpp_hello_world.R’ is expanded in the ‘R’ directory, the rcpp_hello_world function defined in this file makes use of the C++ function ‘rcpp_hello_world’ defined in the C++ file. These files are given as an example and should eventually be removed from the generated package.

If the module argument is TRUE, a sample Rcpp module will be generated as well.

Value

Nothing, used for its side effects

References

Read the *Writing R Extensions* manual for more details.

Once you have created a *source* package you need to install it: see the *R Installation and Administration* manual, [INSTALL](#) and [install.packages](#).

See Also

[package.skeleton](#)

Examples

```
## Not run:
# simple package
Rcpp.package.skeleton( "foobar" )

# package with a module
Rcpp.package.skeleton( "testmod", module = TRUE )
```

```
# the Rcpp-package vignette
vignette( "Rcpp-package" )

# the Rcpp-modules vignette for information about modules
vignette( "Rcpp-modules" )

## End(Not run)
```

RcppUnitTests *Rcpp : unit tests results*

Description

Unit tests results for package Rcpp.

Unit tests are run automatically at build time and reports are included in the ‘doc’ directory as html or text.

Details

See Also

Examples

```
# unit tests are in the unitTests directory of the package
list.files( system.file("unitTests", package = "Rcpp" ),
pattern = "^runit", full = TRUE )

# trigger the unit tests preparation, follow printed instructions
# on how to run them
## Not run:
source( system.file("unitTests", "runTests.R", package = "Rcpp" ) )

## End(Not run)
```

setRcppClass	<i>Define a Reference Class containing a C++ Class</i>
--------------	--------------------------------------------------------

Description

Function `setRcppClass` creates an extended reference class that contains an interface to a C++ class, typically as exposed by an Rcpp [Module](#). The Rcpp class can have R-based fields and methods in addition to those from the C++ class.

Usage

```
setRcppClass(Class, CppClass = , fields = list(), contains = character(), methods = list(), where = topo
```

Arguments

Class	The name of the class to be defined.
CppClass	The C++ class extended by this class, usually a class specified in an Rcpp Module .
fields, contains, methods, where, ...	Arguments to setRefClass , extended in the case of <code>contains=</code> to include the C++ class.

Value

A generator object for the reference class. The generator object is from class "rcppObjectGenerator", with some extra slots to hold information about the contained C++ class.

Author(s)

John Chambers

Examples

```
## Not run:
### Following the vignette for Module exporting C++ class Uniform
### The class randU extends the C++ class to maintain seed for the RNG
### (An example of a reproducible simulation utility, protected
### from any other random number generation. Class "Rseed", not shown,
### mainly exists to print random seeds in a user-readable form.)
randU <- setRcppClass("randU", Uniform,
  fields = list(startSeed = "Rseed", currentSeed = "Rseed"),
  methods = list(
    set.seed = function(seed) {
      base::set.seed(seed)
      startSeed <<- as(.Random.seed, "Rseed")
      currentSeed <<- startSeed
      invisible(currentSeed)
    },
```

```
draw = function(...) {
  if(exists(".Random.seed",envir = .GlobalEnv)) {
    previous <- get(".Random.seed", envir= .GlobalEnv)
    on.exit(assign(".Random.seed", previous, envir = .GlobalEnv))
  }
  else
    on.exit(remove(".Random.seed", envir = .GlobalEnv))
  assign(".Random.seed", currentSeed, envir = .GlobalEnv)
  value <- callSuper(...)
  currentSeed <<- as(get(".Random.seed", envir = .GlobalEnv),
                    "Rseed")
  value
}
)
)
randU$lock("startSeed")

## End(Not run)
## Not run:

## This class can be used with mixed R/C++ methods:

ru <- randU$new(0,10)
ru$set.seed(429)
ru$draw(10)

## End(Not run)
```

Index

- *Topic **classes**
 - C++Class-class, 4
 - C++ClassRepresentation-class, 5
 - C++Constructor-class, 6
 - C++Field-class, 6
 - C++Function-class, 7
 - C++Object-class, 7
 - C++ObjectS3-class, 8
 - C++OverloadedMethods-class, 8
 - Module-class, 10
 - setRcppClass, 14
- *Topic **interface**
 - loadRcppModules, 9
 - Rcpp-package, 3
- *Topic **manip**
 - populate, 11
- *Topic **methods**
 - .DollarNames-methods, 4
 - formals<--methods, 9
- *Topic **programming**
 - Module, 10
 - Rcpp-package, 3
 - Rcpp.package.skeleton, 11
 - RcppUnitTests, 13
 - setRcppClass, 14
 - .DollarNames, ANY-method
 - (.DollarNames-methods), 4
 - .DollarNames, C++Object-method
 - (.DollarNames-methods), 4
 - .DollarNames, Module-method
 - (.DollarNames-methods), 4
 - .DollarNames-methods, 4
 - .environment, 6, 8
 - \$, C++Class-method (C++Class-class), 4
 - \$, C++Object-method (C++Object-class), 7
 - \$, Module-method (Module-class), 10
 - \$<-, C++Object-method (C++Object-class), 7
- C++Class, 6
- C++Class-class, 4
- C++ClassRepresentation-class, 5
- C++Constructor, 4
- C++Constructor-class, 6
- C++Field, 4
- C++Field-class, 6
- C++Function-class, 7
- C++Object-class, 7
- C++ObjectS3-class, 8
- C++OverloadedMethods, 4
- C++OverloadedMethods-class, 8
- classRepresentation, 5
- environment, 6, 8
- envRefClass, 6, 8
- formals<-, C++Function-method
 - (formals<--methods), 9
- formals<--methods, 9
- function, 7
- getNativeSymbolInfo, 10
- initialize, Module-method
 - (Module-class), 10
- INSTALL, 12
- install.packages, 12
- loadRcppModules, 9
- Module, 4, 10, 10, 12, 14
- Module-class, 10
- OptionalFunction, 7
- package.skeleton, 11, 12
- populate, 9, 11
- PossibleMethod, 7
- prompt, Module-method (Module-class), 10
- Rcpp (Rcpp-package), 3

- Rcpp-package, [3](#)
- Rcpp.package.skeleton, [11](#)
- RcppClass-class (setRcppClass), [14](#)
- RcppUnitTests, [13](#)
- refClass, [6](#), [8](#)
- referenceMethods, C++ClassRepresentation-method
(C++ClassRepresentation-class),
[5](#)
- refObject, [6](#), [8](#)

- setClass, [10](#)
- setRcppClass, [14](#)
- setRefClass, [14](#)
- show, C++Class-method (C++Class-class), [4](#)
- show, C++Function-method
(C++Function-class), [7](#)
- show, C++Object-method
(C++Object-class), [7](#)
- show, Module-method (Module-class), [10](#)