QuantLib, R and Rcpp

Dr. Dirk Eddelbuettel
dirk.eddelbuettel@R-Project.org
edd@debian.org
@eddelbuettel

QuantLib User Meeting 2014
IKB Deutsche Industriebank AG
Düsseldorf, Germany
4 December 2014
Outline

1 QuantLib
QuantLib: *The Open Source Quant Library*

Accomplishments well know to all of us

In a nutshell:

- Fifteen years of blood, sweat and tears
- 700k lines of code, examples and unit tests\(^1\)
- Ten of thousands of (svn and now git) commits
- Hundreds of modules, pricers, classes, functions ...
- Mostly Fernando & Luigi, plus a small core team\(^2\)

---

\(^1\)Well, Ohloh says so...

\(^2\)My unscientific guess..
In another nutshell:

- A language and an environment
- Has forever altered the way people analyze, visualize and manipulate data
- A vibrant community and ecosystem: CRAN + BioConductor provide > 6k packages that “just work”
- Reliably cross-platform + cross-operating system
- The lingua franca of (applied) statistical research

---

3 R FAQ, Question 2.1
4 1999 ACM citation for John Chambers
R: History by the Books
John Chambers, with a few co-authors


Thanks to John Chambers for sending me high-resolution scans of the covers of his books.
What makes R so useful?
Succinct and expressive

```r
> xx <- faithful[, "eruptions"]
> fit <- density(xx)
> plot(fit)
```
What makes R so useful?
Succinct and expressive

The example was posted by Greg Snow on r-help a few years ago.
What makes R so useful?
Interactive

R enables us to

- work interactively
- explore and visualize data
- access, retrieve and/or generate data
- summarize and report into pdf, html, ...
- *dynamic documents* in markdown, Shiny dashboards and more

making it a preferred environment for many data analysts.
What makes R so useful?
Extensible

R has always been extensible via

C via a bare-bones interface described in *Writing R Extensions*

Fortran which is also used internally by R

Java via *rJava* by S Urbanek

C++ but essentially at the bare-bones level of C

So ’in theory’ this always worked – yet tedious ’in practice’.
RQuantLib
Obvious Idea: Take R, add QuantLib

Best of both worlds:

- Obvious appeal: R rocks for *Programming with Data*
- Obvious appeal: QuantLib rocks for Finance (pricing, risk management, ...)
- So see if we can join them together
- The first steps were very manual.\(^5\)

---

\(^5\)The very impressive parallel work by Joe Wang on the QL R / SWIG bindings is of course also acknowledged.
// simple helper function to insert "labelled" element into list
static inline
void insertListElement(SEXP &list, SEXP &names,
                        const int pos, const double value,
                        const char *label) {

    SEXP vec = PROTECT(allocVector(REALSXP, 1));
    REAL(vec)[0] = value;
    SET_VECTOR_ELT(list, pos, vec);
    SET_STRING_ELT(names, pos, mkChar(label));
    UNPROTECT(1);
}

// get the list element named str, or return NULL
// courtesy of the R Exts manual, and the nls package
static inline SEXP getListElement(SEXP list, char *str) {
  SEXP elmt = R_NilValue,
     names = getAttrib(list, R_NamesSymbol);
  int i;

  for (i = 0; i < length(list); i++)
    if (strcmp(CHAR(STRING_ELT(names, i)), str) == 0) {
      elmt = VECTOR_ELT(list, i);
      break;
    }
  return elmt;
}
Underlying C API for R somewhat bare
Everything evolves around `.Call` and `SEXP`

At the C++ level:

```
SEXP foo(SEXP a, SEXP b, SEXP C, ...)
```

and at the R level:

```
> res <- .Call("foo", a, b, c, ..., 
+ PACKAGE="mypkg")
```
From RQuantLib (RQL) to Rcpp and back

- RQL 0.1.13 (2002 - Aug 2005): Two macros
- RQL 0.2.0 (Oct 2005): First minimal Rcpp (by Dominick): very different from what we use today
- RQL 0.2.9 (Aug 2008): Last with embedded old Rcpp
- Rcpp 0.6.0 (Nov 2008): New start following the Rcpp/RcppTemplate withdrawal
- RQL 0.2.10 (Dec 2008): Uses external Rcpp
- RQL 0.3.0 (Sep 2009): With Khanh’s GSoC contributions
- Rcpp 0.7.0 (Dec 2009) Romain joined, leading to rapid Rcpp changes over next few years
- Rcpp 0.10.0 (Dec 2012): JJ added Attributes
- Rcpp 0.11.0 (Feb 2013): Easier build, no more linking
- RQL 0.4.0 (Dec 2014): Now with proper use of modern Rcpp
Invoking (some) QuantLib functions for use from R can be as simple as this:

```r
> s <- "QuantLib::Date calDemo(QuantLib::Date d, int dt) {
  return QuantLib::Argentina().advance(d, dt, QuantLib::Days); }"
> Rcpp::cppFunction(s, depends="RQuantLib")
> calDemo(Sys.Date(), 3)
```

```
## [1] "2014-12-03"
```
Fast-forward to today

Or written in a short C++ file ...

```cpp
#include <RQuantLib.h>

// [[Rcpp::depends(RQuantLib)]]

// [[Rcpp::export]]
QuantLib::Date calDemo(QuantLib::Date day, int delta) {
    // or any other calendar
    QuantLib::Calendar cal = QuantLib::Argentina();
    QuantLib::Date newDate =
        cal.advance(day, delta, QuantLib::Days);
    return newDate;
}
```
Fast-forward to today

... which is sourced:

```r
> Rcpp::sourceCpp("code/calDemo.cpp")
> calDemo(Sys.Date(), 3)
```

```
## [1] "2014-12-03"
```
Fast-forward to today
Actual Code Example from Package [slightly compacted, flat namespaces]

```cpp
#include <rquantlib.h>
// [[Rcpp::interfaces(r, cpp)]]
// [[Rcpp::export]]
List europeanOptionEngine(std::string type, double underlying, double strike,
                          double dividendYield, double riskFreeRate, double maturity, double volatility)
{
    int length = int(maturity*360 + 0.5); // FIXME: this could be better
    Option::Type optionType = getOptionType(type);
    Date today = Date::todaysDate();
    Settings::instance().evaluationDate() = today;
    DayCounter dc = Actual360();
    shared_ptr<SimpleQuote> spot(new SimpleQuote(underlying));
    shared_ptr<SimpleQuote> vol(new SimpleQuote(volatility));
    shared_ptr<BlackVolTermStructure> volTS = flatVol(today, vol, dc);
    shared_ptr<SimpleQuote> qRate(new SimpleQuote(dividendYield));
    shared_ptr<YieldTermStructure> qTS = flatRate(today, qRate, dc);
    shared_ptr<SimpleQuote> rRate(new SimpleQuote(riskFreeRate));
    shared_ptr<YieldTermStructure> rTS = flatRate(today, rRate, dc);
    Date exDate = today + length;
    shared_ptr<Exercise> exercise(new EuropeanExercise(exDate));
    shared_ptr<StrikedTypePayoff> payoff(new PlainVanillaPayoff(optionType, strike));
    shared_ptr<VanillaOption> opt = makeOption(payoff, exercise, spot, qTS, rTS, volTS);
    return List::create(Named("value") = opt->NPV(),
                          Named("delta") = opt->delta(),
                          Named("gamma") = opt->gamma(),
                          Named("vega") = opt->vega(),
                          Named("theta") = opt->theta(),
                          Named("rho") = opt->rho(),
                          Named("divRho") = opt->dividendRho());
}
```

Dirk Eddelbuettel
Rcpp
- How
- Example: Recursion
- Example: VAR(1)
- Growth
How do we use Rcpp?

Rcpp Attributes: evalCpp, cppFunction, sourceCpp

> ### evaluate a C++ expression, retrieve result
> evalCpp("2 + 2")

## [1] 4

> ### create ad-hoc R function 'square'
> cppFunction('int square(int x) { return x*x;}')
> square(7L)

## [1] 49

> ### or source an entire file (including R code)
> #sourceCpp("code/squareWithRCall.cpp")
Consider a function defined as

\[
\begin{align*}
    f(n) \quad & \text{such that} \quad \\
    & \begin{cases} 
        n & \text{when } n < 2 \\
        f(n - 1) + f(n - 2) & \text{when } n \geq 2
    \end{cases}
\end{align*}
\]
When do we use Rcpp?

Easy speedup: Simple R Implementation

```r
> fibR <- function(n) {
+   if (n < 2) return(n)
+   return(fibR(n-1) + fibR(n-2))
+ }
> ## Using it on first 11 arguments
> sapply(0:10, fibR)

## [1]  0  1  1  2  3  5  8 13 21 34 55
```
### When do we use Rcpp?

Easy speedup: Timing R Implementation

```r
> benchmark(fibR(10), fibR(15), fibR(20))[,1:4]

<table>
<thead>
<tr>
<th></th>
<th>test</th>
<th>replications</th>
<th>elapsed</th>
<th>relative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fibR(10)</td>
<td>100</td>
<td>0.017</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>fibR(15)</td>
<td>100</td>
<td>0.201</td>
<td>11.824</td>
</tr>
<tr>
<td>3</td>
<td>fibR(20)</td>
<td>100</td>
<td>2.132</td>
<td>125.412</td>
</tr>
</tbody>
</table>
```
When do we use Rcpp?
Easy speedup: C++ Implementation

```r
> cppFunction(" int fibCcpp(int n) {
    if (n < 2) return(n);
    return(fibCcpp(n-1) + fibCcpp(n-2));
}")
> ## Using it on first 11 arguments
> sapply(0:10, fibCcpp)
```

```
## [1] 0 1 1 2 3 5 8 13 21 34 55
```
When do we use Rcpp?
Easy speedup: Putting it all together

```r
> fibR <- function(n) {
+     if (n<2) return(n)
+     return(fibR(n-1) + fibR(n-2))
+ }
> cppFunction('int fibCpp(int n) {
+     if (n<2) return n;
+     return fibCpp(n-2) + fibCpp(n-1);
+ }
> benchmark(fibR(25), fibCpp(25), order="relative")[,1:4]
```

```
## test replications elapsed relative
## 2 fibCpp(25)     100   0.058   1.0  
## 1 fibR(25)      100  24.157  416.5
```
Let’s consider a simple possible VAR(1) system of $k$ variables.

For $k = 2$:

$$X_t = X_{t-1} B + E_t$$

where $X_t$ is a row vector of length 2, $B$ is a 2 by 2 matrix and $E_t$ is a row of the error matrix of 2 columns.
When do we use Rcpp?

Easy speedup:: VAR(1) Simulation

In R code, given both the coefficient and error matrices (revealing $k$ and $n$):

```r
> rSim <- function(B,E) {
+   n <- nrow(E); k <- ncol(E)
+   X <- matrix(0, n, k)
+   for (r in 2:n) {
+     X[r,] = X[r-1,] %*% B + E[r,]
+   }
+   return(X)
+ }
```
When do we use Rcpp?
Easy speed gain: VAR(1) Simulation

```cpp
cppFunction('arma::mat cppSim(const arma::mat& B, const arma::mat& E) {
    int n = E.n_rows; int k = E.n_cols;
    arma::mat X = arma::zeros<arma::mat>(n,k);
    for (int r=1; r < n; r++) {
        X.row(r) = X.row(r-1) * B + E.row(r);
    }
    return X;
}', depends="RcppArmadillo")
```
When do we use Rcpp?
Easy speed gain: VAR(1) Simulation

```cpp
#include <Rcpp.h>

// [[Rcpp::depends(RcppArmadillo)]]

// [[Rcpp::export]]
arma::mat cppSim(const arma::mat& B,
const arma::mat& E) {
  int n = E.n_rows; int k = E.n_cols;
  arma::mat X = arma::zeros<arma::mat>(n,k);
  for (int r=1; r < n; r++) {
    X.row(r) = X.row(r-1) * B + E.row(r);
  }
  return X;
}
```
When do we use Rcpp?
Easy speed gain: VAR(1) Simulation

```r
> a <- matrix(c(0.5, 0.1, 0.1, 0.5), nrow=2)
> e <- matrix(rnorm(10000), ncol=2)
> all.equal(cppSim(a, e), rSim(a, e))

## [1] TRUE

> benchmark(cppSim(a, e), rSim(a, e),
+ order="relative")[,1:4]

## test replications elapsed relative
## 1 cppSim(a, e) 100 0.024 1.000
## 2 rSim(a, e) 100 2.300 95.833
```

Dirk Eddelbuettel
QuantLib, R and Rcpp
Rcpp on CRAN
Used by 304 packages, or just under 5 per cent

Growth of Rcpp usage on CRAN

- Number of CRAN packages using Rcpp
- Percentage of CRAN packages using Rcpp

Dirk Eddelbuettel  QuantLib, R and Rcpp
Outline

5 R and QuantLib
- General
- Shiny
- RMarkdown
Basic just work: we convert standard C++ types, including STL containers seamlessly.

Custom converters can be added easily as shown for `QuantLib::Date`.

Should work out a proper R presentation of things like *curves* and *surfaces*.

R gives us a wealth of things for data creation, analysis and reporting.

Two recent R developments for reporting / communicating results highlighted in the next two sections.
Shiny for Dynamic Documents
After a decade of GUI attempts, web frameworks, ...

Shiny just works:

- Minimal coding:
- One file `ui.R` to declare the user interface
- One file `server.R` to declare the backend
- Well documented, many examples and add-ons.
Simple DiscountCurve Example from RQuantLib

Interpolation type:
- loglinear
- linear
- spline

Curve type:
- forwards
- zero rates
- discounts

spline zerorates

zerosrates

0.035
0.04
0.045
0.05

0 2 4 6 8 10

time in years
library(shiny)

shinyUI(fluidPage(  
  ## Application title
  titlePanel("Simple DiscountCurve Example from RQuantLib"),
  ## Sidebar with controls to select parameters
  sidebarLayout(    
    sidebarPanel(      
      radiosButtons("interpolation", "Interpolation type:",
                     c("loglinear" = "loglinear",
                        "linear" = "linear",
                        "spline" = "spline")),

      br(),
      radiosButtons("curve", "Curve type:",
                     c("forwards" = "forwards",
                        "zero rates" = "zerorates",
                        "discounts" = "discounts")),

    )
  ),
  ## Show a tabset that includes a plot, summary, and table view
  mainPanel(    
    tabsetPanel(type = "tabs",
                tabPanel("Plot", plotOutput("plot")),
                tabPanel("Summary", verbatimTextOutput("summary")),
                tabPanel("Table", tableOutput("table"))
    )
  )
))
library(shiny); library(RQuantLib)

shinyServer(function(input, output) {
  params <- list(tradeDate=as.Date('2004-09-20'), ...)
  setEvaluationDate(as.Date("2004-09-20"))
  tsQuotes <- list(d1w = 0.0382, d1m = 0.0372, ..., s15y = 0.055175)
  times <- seq(0,10,.1)
  data <- reactive({
    ## Reactive expression to generate the requested curves.
    params$interpHow <- input$interpolation
    curve <- DiscountCurve(params, tsQuotes, times)
  })
  output$plot <- renderPlot({
    ## Generate a plot of the data
    interp <- input$interpolation
    crv <- input$curve
    dat <- data()
    plot(dat[["times"]], dat[[crv]],
         type='l', main=paste(interp, crv), ylab=crv, xlab="time in years")
  })
  output$summary <- renderPrint({
    ## Generate a summary of the data
    dat <- data()
    cat("Return Object Structure\n")
    str(dat)
    cat("\n\nSummary of first four elements\n")
    summary(data.frame(dat[1:4]))
  })
  output$table <- renderTable({
    ## Generate an HTML table view of the data
    data.frame(x=data()[1:4])
  })
})
RMarkdown extends basic Markdown in multiple ways:

- Markdown can be mixed freely with R code expressions
- By relying on pandoc as the engine, conversion to html, latex/pdf, and even Word “just works”
- It complements the standard R + \LaTeX\ approach (used in these slides)
- The newest variant extends this for Dynamic Documents
This R Markdown document is made interactive using Shiny, following a standard examples -- try File -> New File -> R Markdown -> Shiny -> Shiny Document. Much more documentation is available on Interactive Documents (http://rmarkdown.rstudio.com/authoring_shiny.html).

### Standard Inputs and Outputs

You can embed Shiny inputs and outputs in your document. Outputs are automatically updated whenever inputs change. This demonstrates how a standard R plot can be made interactive by wrapping it in the Shiny `renderPlot` function. The `selectInput` and `sliderInput` functions create the input widgets used to drive the plot.

```r
echo=FALSE
inputPanel(
  selectInput("n_breaks", label = "Number of bins:",
              choices = c(10, 20, 35, 50), selected = 20),

  sliderInput("bw_adjust", label = "Bandwidth adjustment:",
              min = 0.2, max = 2, value = 1, step = 0.2)
)

renderPlot({
  hist(faithful$eruptions, probability = TRUE, breaks = as.numeric(input$n_breaks),
        xlab = "Duration (minutes)", main = "Geyser eruption duration")

  dens <- density(faithful$eruptions, adjust = input$bw_adjust)
  lines(dens, col = "blue")
})
```
This R Markdown document is made interactive using Shiny, following a standard example -- try File -> New File -> R Markdown -> Shiny -> Shiny Document. Much more documentation is [available on Interactive Documents](http://rmarkdown.rstudio.com/authoring_shiny.html).

### Embed Entire Application

It's also possible to embed an entire Shiny application within an R Markdown document using the `shinyAppDir` function. This example embeds a Shiny application located in another directory:

```r
{r, echo=FALSE}
shinyAppDir(
  "discountCurveShiny", options=list(width="100\%", height=550)
)
```
Outline

6 Issues
Open Issues / Challenges

- Statefulness etc is an issue for R interface: So far just simple calls and returns; one singleton.
- Eventually need a way to hang on to objects and revisit them.
- QL issue of 'time unit is a single day' is limiting
- Overall balance of featuritis and ease of use (still no simple Black/Scholes)
7 Successes
Achievements

- RQuantLib spawned Rcpp.
- RQuantLib, while incomplete, has a number of users.
- Automatic Windows builds from CRAN help a lot.
- (Mistyifies me as I see this a development framework rather than an appliance ...)
- The Future is so bright...
Postscriptum

Slides are available at my presentations page\textsuperscript{6}. Example code is in the samplecode\textsuperscript{7} github repo in the directory quantlib-2014-12. Rcpp repository\textsuperscript{8} RQuantLib repository\textsuperscript{9} Rcpp Gallery\textsuperscript{10}

\textsuperscript{6}http://dirk.eddelbuettel.com/presentations/
\textsuperscript{7}https://github.com/eddelbuettel/samplecode/
\textsuperscript{8}https://github.com/RcppCore/Rcpp
\textsuperscript{9}https://github.com/eddelbuettel/rquantlib
\textsuperscript{10}http://gallery.rcpp.org