RQuantLib: Interfacing QuantLib from R

Dirk Eddelbuettel\textsuperscript{1} \hspace{1cm} Khanh Nguyen\textsuperscript{2}

\textsuperscript{1} Debian Project
\textsuperscript{2} UMASS at Boston

Presentation on 23 July 2010 at \textit{useR! 2010}
National Institute of Standards and Technology (NIST)
Gaithersburg, Maryland, USA
Outline

1. QuantLib
   - Overview
   - Timeline
   - Architecture
   - Examples

2. RQuantLib

3. Fixed Income

4. Summary
The initial **QuantLib** release was 0.1.1 in Nov 2000.

The first Debian **QuantLib** package was prepared in May 2001.

Boost has been a **QuantLib** requirement since July 2004.

The long awaited **QuantLib** 1.0.0 release appeared in Feb 2010.
A few key points about QuantLib

QuantLib ...

- is a C++ library for financial quantitative analysts and developers.
- was started in 2000 and is hosted on Sourceforge.Net
- is a free software project under a very liberal license allowing for inclusion in commercial projects.
- is primarily the work of Ferdinando Ametrano and Luigi Ballabio.
- is sponsored by the Italian consultancy StatPro which derives consulting income from it.
QuantLib is written in C++ and fairly rigourously designed.
Luigi Ballabio has draft chapters on the QuantLib design and implementation at http://sites.google.com/site/luigiballabio/qlbook.
QuantLib use the Boost testing framework and employs hundreds of detailed unit tests.
QuantLib makes extensive use of Swig and bindings for Java, Perl, Python, Ruby, C#, Guile ... exist.
QuantLibAddin exports a procedural interface to a number of platforms including Excel and Oo Calc.
Several manual (non-SWIG) extension such as RQuantLib exist as well.
Key Modules
A rough guide, slight re-arranged from the QuantLib documentation

- Pricing engines (Asian, Barrier, Basket, Cap/Floor, Cliquet, Forward, Quanto, Swaption, Vanilla)
- Finite-differences framework
- Fixed-Income (Short-rate modelling, Term structures)
- Currencies and FX rates
- Financial instruments
- Math tools (Lattice method, Monte Carlo Framework, Stochastic Process)
- Date and time calculations (Calendars, Day Counters)
- Utilities (Numeric types, Design patterns, Output manipulators)
- QuantLib macros (Numeric limits, Debugging)
$ EquityOption

Option type = Put
Maturity = May 17th, 1999
Underlying price = 36
Strike = 40
Risk-free interest rate = 6.000000 %
Dividend yield = 0.000000 %
Volatility = 20.000000 %

<table>
<thead>
<tr>
<th>Method</th>
<th>European</th>
<th>Bermudan</th>
<th>American</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black-Scholes</td>
<td>3.844308</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Barone-Adesi/Whaley</td>
<td>N/A</td>
<td>N/A</td>
<td>4.459628</td>
</tr>
<tr>
<td>Bjerksund/Stensland</td>
<td>N/A</td>
<td>N/A</td>
<td>4.453064</td>
</tr>
<tr>
<td>Integral</td>
<td>3.844309</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Finite differences</td>
<td>3.844342</td>
<td>4.360807</td>
<td>4.486118</td>
</tr>
<tr>
<td>Binomial Jarrow-Rudd</td>
<td>3.844132</td>
<td>4.361174</td>
<td>4.486552</td>
</tr>
<tr>
<td>Binomial Cox-Ross-Rubinstein</td>
<td>3.843504</td>
<td>4.360861</td>
<td>4.486415</td>
</tr>
<tr>
<td>Additive equiprobabilities</td>
<td>3.836911</td>
<td>4.354455</td>
<td>4.480097</td>
</tr>
<tr>
<td>Binomial Trigeorgis</td>
<td>3.843557</td>
<td>4.360909</td>
<td>4.486461</td>
</tr>
<tr>
<td>Binomial Tian</td>
<td>3.844171</td>
<td>4.361176</td>
<td>4.486413</td>
</tr>
<tr>
<td>Binomial Leisen-Reimer</td>
<td>3.844308</td>
<td>4.360713</td>
<td>4.486076</td>
</tr>
<tr>
<td>Binomial Joshi</td>
<td>3.844308</td>
<td>4.360713</td>
<td>4.486076</td>
</tr>
<tr>
<td>MC (crude)</td>
<td>3.834522</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>QMC (Sobol)</td>
<td>3.844613</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>MC (Longstaff Schwartz)</td>
<td>N/A</td>
<td>N/A</td>
<td>4.481675</td>
</tr>
</tbody>
</table>

Run completed in 5 s
Errors from discrete hedging (Derman and Kamal)

```
$ DiscreteHedging

Option value: 2.51207

<table>
<thead>
<tr>
<th>samples</th>
<th>trades</th>
<th>P&amp;L</th>
<th>mean</th>
<th>std.dev.</th>
<th>Derman&amp;Kamal</th>
<th>skewness</th>
<th>kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>50000</td>
<td>21</td>
<td>-0.001</td>
<td>0.43</td>
<td>0.44</td>
<td>-0.33</td>
<td>1.56</td>
<td></td>
</tr>
<tr>
<td>50000</td>
<td>84</td>
<td>0.000</td>
<td>0.22</td>
<td>0.22</td>
<td>-0.20</td>
<td>1.68</td>
<td></td>
</tr>
</tbody>
</table>

Run completed in 16 s
```

Other examples include SwapValuation, Repo, Replication, FRA, FittedBondCurve, Bonds, BermudanSwaption, CDS, ConvertibleBonds, CallableBonds and MarketModels.

Also available are quantlib-benchmark (running 85 tests) and quantlib-test-suite (running 446 tests cases).
Overview

- Initial implementation: Standard equity option pricing:
  - pricers and greeks for European and American options
  - first set of exotics using barrier and binaries
  - also implied volatility calculations where available
- First external contribution: Curves and Swaption pricing.
- Second external contribution (as Google Summer of Code): Fixed Income Functionality (more on this below)
- Other small extensions on date and holiday calculations.
### Option Valuation and Greeks

Analytical results where available

```r
R> example(EuropeanOption)

ErpnOpR> # simple call with unnamed parameters
ErpnOpR> EuropeanOption("call", 100, 100, 0.01, 0.03, 0.5, 0.4)
Concise summary of valuation for EuropeanOption
  value delta gamma vega theta rho divRho
  11.6365  0.5673  0.0138 27.6336 -11.8390 22.5475 -28.3657

ErpnOpR> # simple call with some explicit parameters, and slightly increased vol:
ErpnOpR> EuropeanOption(type="call", underlying=100, strike=100, dividendYield=0.01,
ErpnOpR>  riskFreeRate=0.03, maturity=0.5, volatility=0.5)
Concise summary of valuation for EuropeanOption
  value delta gamma vega theta rho divRho
  14.3927  0.5783  0.0110 27.4848 -14.4673 21.7206 -28.9169
R> example(BinaryOption)

BnryOpR> BinaryOption(binType="asset", type="call", excType="european",
BnryOpR>  underlying=100, strike=100, dividendYield=0.02,
BnryOpR>  riskFreeRate=0.03, maturity=0.5, volatility=0.4, cashPayoff=10)
Concise summary of valuation for BinaryOption
  value delta gamma vega theta rho divRho
  55.760  1.937  0.006 12.065 -5.090 68.944 -96.824
R> example(BarrierOption)

BrrrOpR> BarrierOption(barrType="downin", type="call", underlying=100,
BrrrOpR>  strike=100, dividendYield=0.02, riskFreeRate=0.03,
BrrrOpR>  maturity=0.5, volatility=0.4, barrier=90)
Concise summary of valuation for BarrierOption
  value delta gamma vega theta rho divRho
  3.738 NaN NaN NaN NaN NaN NaN
```
Option Valuation and Greeks

The demo(OptionSurfaces) provides some animation
Outline

1. QuantLib
2. RQuantLib
3. Fixed Income
   - Overview and development
   - Examples
4. Summary
Fixed Income Development

RQuantLib before GSOC 2009...

Option Pricing

Curve Fitting

DiscountCurve
GSOC started. April 2009...
Fixed Income Development

Making curve fitting and bond pricing work together...
And recently, we have started to add **GUIs**
In summary

**Fixed-Income**

**Bond Pricing**
- Zero Coupon Bond
- Fixed Rate Bond
- Floating Rate Bond
- Convertible Zero Coupon Bond
- Convertible Fixed Rate Bond
- Convertible Floating Rate Bond
- Callable Bond

**Curve Fitting**
- DiscountCurve
- FittedBondCurve
Examples

- Curve fitting: DiscountCurve
- Curve fitting: FittedBondCurve
- Bond pricing: QuantLib’s bond examples
- Bond pricing: Matlab’s convertible bond
- Graphical User Interface screenshots
We construct a bond discounting term structure and then use it to price a zero coupon bond and a fixed rate bond.

All the input data and dates are taken from the bond pricing example shipped with QuantLib.

```r
#we start with date parameters
fixingDays <- 3
settlementDays <- 3
settlementDate <- as.Date('2008-09-18')
todaysDate <- settlementDate - fixingDays
```
#set up bond discounting term structure

lengths <- c(5, 6, 7, 16, 48)
coupons <- c(0.02375, 0.04625, 0.03125, 0.04000, 0.04500)
marketQuotes <- c(100.390625, 106.21875, 100.59375, 101.6875, 102.140625)
dateparams <- list(settlementDays=settlementDays,
                   period=2, dayCounter="ActualActual",
                   businessDayConvention ="Unadjusted")
curveparams <- list(method="ExponentialSplinesFitting",
                    origDate=todaysDate)
bondDsctTsr <- FittedBondCurve(curveparams, lengths,
coupons, marketQuotes, dateparams)
### Set up a Zero-Coupon Bond

```r
zc.bond.param <- list(
maturityDate = as.Date('2013-08-15'),
issueDate = as.Date('2003-08-15'),
redemption = 116.92)
```

```r
zc.bond.dateparam <- list(
refDate = todaysDate,
settlementDays = settlementDays,
businessDayConvention = 'Following')
```

### Call the pricing function

```r
ZeroCouponBond(zc.bond.param,
bondDsctTsr,
zc.bond.dateparam)
```
# Set up a Fixed-Coupon Bond

```r
defined.bond.param <- list(
maturityDate = as.Date('2017-05-15'),
issueDate = as.Date('2007-05-15'),
redemption = 100,
effectiveDate = as.Date('2007-05-15'))
defined.bond.dateparam <- list(
settlementDays = settlementDays,
dayCounter = 'ActualActual',
period = 'Semiannual',
businessDayConvention = 'Unadjusted',
terminationDateConvention = 'Unadjusted',
dateGeneration = 'Backward',
endOfMonth = 0)
defined.bond.coupon <- c(0.045)
```

# Call the pricing function

```r
FixedRateBond(fixed.bond.param, defined.bond.coupon,
               bondDscTsr, defined.bond.dateparam)
```
Perform a spread effect analysis of a 4%-coupon convertible bond callable at 110 at the end of the second year, maturing at par in 5 years, with yield to maturity of 5% and spread (of YTM versus 5-year treasury) of 0, 50, 100, and 150 basis points. The underlying stock pays no dividend.

```
1 RiskFreeRate = 0.05; Sigma = 0.3;
2 ConvRatio = 1; NumSteps = 200;
3 IssueDate = datenum('2−Jan−2002');
4 Settle = datenum('2−Jan−2002');
5 Maturity = datenum('2−Jan−2007');
6 CouponRate = 0.04; Period = 2; Basis = 1; EndMonthRule = 1;
7 DividendType = 0; DividendInfo = [];
8 CallInfo = [datenum('2−Jan−2004'), 110];
9 CallType = 1; TreeType = 1;
10 % Nested loop across prices and static spread dimensions to compute convertible prices.
11 for j = 0:0.005:0.015;
12 StaticSpread = j;
13 for i = 0:10:100
14 Price = 40+i;
15 [CbMatrix, UndMatrix, DebtMatrix, EqtyMatrix] = cbprice(RiskFreeRate, StaticSpread, Sigma, Price, ConvRatio, NumSteps, IssueDate, Settle, Maturity, CouponRate, Period, Basis, EndMonthRule, DividendType, DividendInfo, CallType, CallInfo, TreeType);
16 convprice(i/10+1,j*200+1) = CbMatrix(1,1);
17 stock(i/10+1,j*200+1) = Price;
18 end
end
```
Fixed Income in RQuantLib
Examples: Convertible Bond from Matlab's Fixed Income Toolbox


```r
plot(stock, convprice);
legend( c('+0 bp'; '+50 bp'; '+100 bp'; '+150 bp' ) );
title( 'Effect of Spread using Trinomial tree − 200 steps' )
xlabel('Stock Price');
ylabel('Convertible Price');
text(50, 150, c('Coupon 4% semiannual.',
                 sprintf('\n'), ...
                 '110 Call-on-clean after two years. 'sprintf('\n'), ...
                 'Maturing at par in five years. '],
                 'fontweight', 'Bold')
```
Doing it in R using RQuantLib....

#set up a flat risk free curve
params <- list(tradeDate=as.Date("2002-01-02"), settleDate=as.Date("2002-01-02"),
               interpWhat="discount", interpHow="loglinear")
RiskFreeRate <- DiscountCurve(params, list(flat=0.05),times)

#parameters of the convertible bond
ConvRatio <- 1
issueDate <- as.Date("2002-01-02")
settleDate <- as.Date("2002-01-02")
maturityDate <- as.Date("2007-01-02")
dividendYield <- DiscountCurve(params, list(flat=0.01),times)
dividendSchedule <- data.frame(Type=character(0), Amount=numeric(0),
                               Rate=numeric(0), Date=as.Date(character(0)))
callabilitySchedule <- data.frame(Price=110, Type=0, Date=as.Date("2004-01-02"))
coupon <- 0.04
dateparams <- list(settlementDays=3, period="Semiannual", todayDate=issueDate)
bondparams <- list(exercise="eu", faceAmount=100,
                   divSch=dividendSchedule,
                   callSch=callabilitySchedule,
                   redemption=100,
                   creditSpread=0.005,
                   conversionRatio=ConvRatio,
                   issueDate=issueDate,
                   maturityDate=maturityDate)
#arguments to construct a BlackScholes process and set up the binomial pricing process
#engine for this bond.
Sigma <- 0.3
process <- list(underlying=40, divYield=dividendYield, 
                 rff=RiskFreeRate, volatility=Sigma)

#loop through underlying price and spread to produce similar analysis to Matlab
ret <- data.frame()
for (s in c(0, 0.005, 0.010, 0.015)) {
    x <- c()
    y <- c()
    i <- 1
    for (p in seq(0, 100, by = 10)) {
        process$underlying <- 40+p
        bondparams$creditSpread <- s
        t <- ConvertibleFixedCouponBond(bondparams, 
                                        coupon, 
                                        process, 
                                        dateparams)

        x[i] <- p + 40
        y[i] <- t$cleanPrice
        i <- i + 1
    }
    z <- rep(s, 11)
    ret <- rbind(ret, data.frame(Stock=x, ConvPrice=y, z))
}
#plot the result
>library(ggplot2)
>p <- ggplot(ret, aes(Stock, ConvPrice, colour=factor(z)))
>p + geom_line() + scale_colour_discrete("Spread")
+ opts(title='Effect of spread on a convertible bond')
RQuantLibGUI provides a graphical user interface via the 'traitr' package by John Verzani.
Fixed Income in RQuantLib
Graphical User Interface: Discount Curve
Fixed Income in RQuantLib
Graphical User Interface: Bonds

Zero Coupon Bond
Fixed Rate Bond
Floating Rate Bond

**Parameters**
- **Issue Date**: 2010-04-08
- **Maturity Date**: 2020-04-08
- **Rates**: 0.034
- **Face Amount**: 100
- **Redemption**: 100

**Date Parameters**
- **settlementDays**: 3
- **calendar**: US
- **dayCounter**: Thirty360
- **period**: Semiannual
- **businessDayConvention**: Following
- **terminationDateConvention**: Following
- **dateGeneration**: Backward

**Result**
- **NPV**: 87.0684451738347
- **Clean price**: 87.0808793008888
- **Dirty price**: 87.128101523111
- **Yield**: 0.0505189990997315

**Graph**
- X-axis: Date (2012-2020)
- Y-axis: Amount

Eddelbuettel and Nguyen
RQuantLib: Interfacing QuantLib from R
QuantLib represents a decade of work leading to the recent 1.0 release.

RQuantLib (still) exposes only a subset of the available functionality.

The conversion to the new Rcpp API (just completed, release pending) should make additions easier.

Next steps we are thinking about
- Expanding the GUIs to the option pricers
- And of course adding more products and QuantLib features

We welcome feedback as well as contributions – just register at the R-Forge project site.

Thank you!