RQuantLib: Interfacing QuantLib from R

R / Finance 2010 Presentation

Dirk Eddelbuettel\(^1\) Khanh Nguyen\(^2\)

\(^1\)Debian Project
\(^2\)UMASS at Boston

R / Finance 2010
April 16 and 17, 2010
Chicago, IL, USA

QuantLib releases
Showing the growth of QuantLib over time

- 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 Architecture

How is it put together and how do I use it?

- **QuantLib** is written in C++ and fairly rigorously designed.
- Luigi Ballabio has draft chapters on the QuantLib design and implementation at [http://sites.google.com/site/luigiballabio/qlbook](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)

Options: Fifteen solutions and three different exercises

$ 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>Bjerkund/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>3.843904</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)

$\text{DiscreteHedging}$

Option value: 2.51207

<table>
<thead>
<tr>
<th>samples</th>
<th>trades</th>
<th>P&amp;L</th>
<th>mean</th>
<th>Derman&amp;Kamal</th>
<th>P&amp;L</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).

- 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> 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,
ErpnOp+ 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",
BnryOp+ underlying=100, strike=100, dividendYield=0.02,
BnryOp+ 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,
BrrrOp+ strike=100, dividendYield=0.02, riskFreeRate=0.03,
BrrrOp+ 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

The demo (OptionSurfaces) provides some animation
Fixed Income Development

RQuantLib before GSOC 2009...

GSOC started. April 2009...
Making curve fitting and bond pricing work together...
And recently, we have started to add **GUIs**
In summary

- **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
Examples: Curve fitting with DiscountCurve function

plot(curves)
library(zoo)
z <- zoo(curve$table$zeroRates, order.by=curve$table$date)
plot(z, xlab='Date', ylab='Zero Rates')
Examples: Bond pricing

# 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=-3, period=2, dayCounter="ActualActual",
                     businessDayConvention ="Unadjusted"
                     )
curveparams <- list(method="ExponentialSplinesFitting",
                     origDate=todaysDate)
bondDsctTsr <- FittedBondCurve(curveparams, lengths,
coupons, marketQuotes, dateparams)
### Set up a Fixed-Coupon Bond

```r
fixed.bond.param <- list(
maturityDate=date('2017-05-15'),
issueDate=date('2007-05-15'),
redemption=100,
effectiveDate=date('2007-05-15'))
```

```r
fixed.bond.dateparam <- list(
settlementDays=settlementDays,
dayCounter='ActualActual',
period='Semiannual',
businessDayConvention='Unadjusted',
terminationDateConvention='Unadjusted',
dateGeneration='Backward',
endOfMonth=0)
```

```r
fixed.bond.coupon <- c(0.045)
```

### Call the pricing function

```r
FixedRateBond(fixed.bond.param, fixed.bond.coupon,
             bondDsctTsr, fixed.bond.dateparam)
```
Fixed Income in RQuantLib
Examples: Convertible Bond from Matlab's Fixed Income Toolbox


```r
plot(stock, convprice);
legend('+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, ['Coupon 4% semiannual.', sprintf('
'), ...
'110 Call-on-clean after two years', sprintf('
'), ...
'Maturing at par in five years.', 'fontweight', 'Bold')
```

Eddelbuettel and Nguyen  RQuantLib
#arguments to construct a BlackScholes process and set up the binomial pricing process

```r
#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))
}
```

Eddelbuettel and Nguyen

RQuantLib
RQuantLibGUI provides a graphical user interface via the 'traitr' package by John Verzani.
Fixed Income in RQuantLib

Graphical User Interface: Bonds
QuantLib represents a decade of work leading to the recent 1.0 release.

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

We are thinking about
- Conversion to the new Rcpp API
- 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!