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 18 and 19, 2010
Chicago, IL, USA

Overview
Presentation details

- Brief overview of QuantLib
  - History, about to release 1.0 after eight long years
  - Luigi's design document draft, mention rigorous design, unit tests, boost, 'grown up C++'
  - Maybe mention different language bindings
  - Maybe mention liberal QL license; R / RQuantLib with GPL somewhat tighter but in spirit of R community
- RQuantLib maybe chronologically
  - Equity options part
  - Simple calendaring
  - Mention the older fixed income / curve stuff without dwelling on it
- Fixed Income / GSoC 2009
  - Khanh ....
  - More Khanh ...
- Total of somewhere between 20 and 30 pages
- Finish with Outlook / Agenda / Areas not yet covered

We can do code

Thanks to lstlisting

```
#include <Rcpp.hpp>

RcppExport SEXP dd_rcpp(SEXP v) {
  SEXP  rl = R_NilValue;             // Use this when nothing is returned

  RcppVector<int> vec(v);            // vec parameter viewed as vector of doubles
  int n = vec.size(), i = 0;

  for (int a = 0; a < 9; a++)
    for (int b = 0; b < 9; b++)
      for (int c = 0; c < 9; c++)
        for (int d = 0; d < 9; d++)
          vec(i++) = a*b - c*d;

  RcppResultSet rs;                  // Build result set returned as list to R
  rs.add("vec", vec);                // vec as named element with name 'vec'
  rl = rs.getReturnList();           // Get the list to be returned to R.

  return rl;
}
```

Fixed Income in RQuantLib

Quick overview

- Fixed Income functions are added during the summer of 2009 as part of the Google Summer of Code program.
- RQuantLib offeres strong support for fixed income pricing whereas several other packages (e.g. termstrc, YieldCurve, fBonds) focus on modelling term structure.
- The functions aim to support two primary tasks: pricing and curve fitting.

Fixed Income in RQuantLib
Primary tasks: Curve fitting

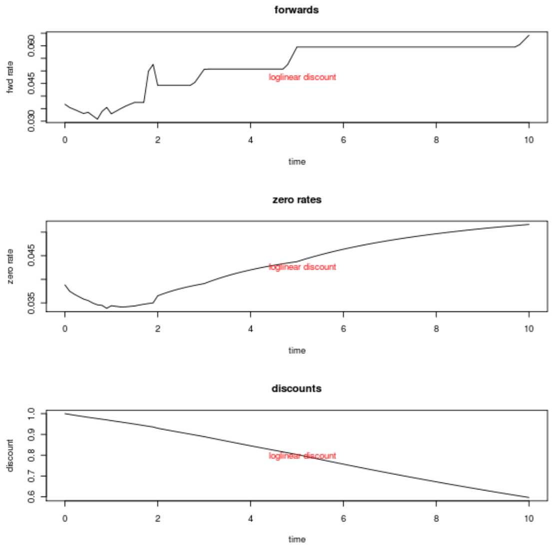
- Curve fitting functions
  - Curve fitting functions return a DiscountCurve object that contains a two column dates/zeroRates data frame.
  - The returned DiscountCurve object are used as inputs for pricing functions.
  - Currently, there are two curve fitting functions
    - DiscountCurve - constructs the spot term structure of interest rates based on input market data including the settltment date, deposit rates, future prices, FRA rates or swap rates in various combination.
    - FittedBondCurve - fits a term structure to a set of bonds using three different fitting methods (ExponentialSplinesFitting, SimplePolynomialFitting, NelsonSiegelFitting).

Fixed Income in RQuantLib
Primary tasks: Bond pricing

- Bond pricing functions return clean price, dirty price, NPV and cash flow of a bond
- Currently, the following bonds are supported
  - Zero Coupon Bond
  - Fixed Rate Bond
  - Floating Rate Bond
  - Convertible Zero Coupon Bond
  - Convertible Fixed Rate Bond
  - Convertible Floating Rate Bond
  - Callable Bond
- The bonds available in QuantLib that yet are implemented are AmortizingCmsRateBond, AmortizingFixedRateBond, AmortizingFloatingRateBond, CallableFixedRateBond, CmsRateBond.

Fixed Income in RQuantLib
Examples: Curve fitting

```
params <- list(tradeDate=as.Date('2004-09-20'),
               settleDate=as.Date('2004-09-22'),
               interpWhat="discount",
               interpHow="loglinear")
tsQuotes <- list(d1w=0.0382, d1m=0.0372,
                 d3m=0.0363, d6m=0.0353,
                 d9m=0.0348, d1y=0.0345,
                 fut2=96.7875, fut3=96.9875,
                 fut4=96.6875, fut5=96.4875,
                 fut7=96.2875, s2y=0.037125,
                 s3y=0.0398, s5y=0.0443,
                 s10y=0.05165, s15y=0.055175)
curves <- DiscountCurve(params, tsQuotes)
```

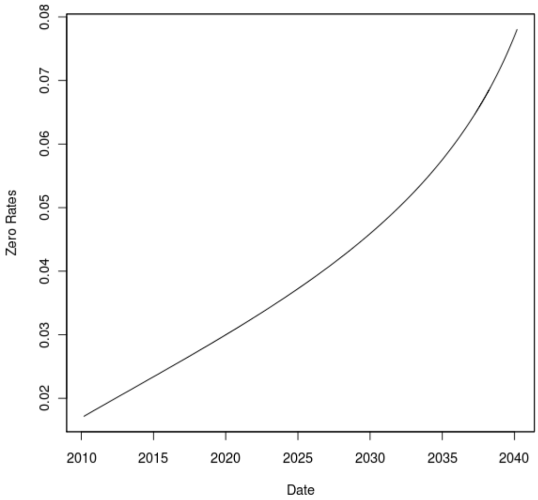

forwards
fwd rate
0.030
0.045
0.060
loglinear discount
0
2
4
6
8
10
time
zero rates
zero rate
0.035
0.045
loglinear discount
0
2
4
6
8
10
time
discounts
discount
0.6
0.7
0.8
0.9
1.0
loglinear discount
0
2
4
6
8
10
time

Fixed Income in RQuantLib

Examples: Curve fitting

```
lengths <- c(2,4,6,8,10,12,14,16,18,
             20,22,24,26,28,30)
coupons <- c(0.0200, 0.0225, 0.0250, 0.0275,
             0.0300, 0.0325, 0.0350, 0.0375,
             0.0400, 0.0425, 0.0450, 0.0475,
             0.0500, 0.0525, 0.0550 )
marketQuotes <- rep(100, length(lengths))
dateparams <- list(settlementDays=0,
                   period="Annual",
                   dayCounter="ActualActual",
                   businessDayConvention="Unadjusted")
curveparams <- list(method="ExponentialSplinesFitting",
                    origDate = Sys.Date())
curve <- FittedBondCurve(curveparams, lengths,
                         coupons, marketQuotes,
                         dateparams)
```

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



Fixed Income in RQuantLib
Examples: Bond pricing

```
// the only header you need to use QuantLib
#include <ql/quantlib.hpp>

#include <boost/timer.hpp>
#include <iostream>
#include <iomanip>

using namespace QuantLib;

#if defined(QL_ENABLE_SESSIONS)
namespace QuantLib {

Integer sessionId() { return 0; }

}
#endif


int main(int, char* []) {

    try {

        boost::timer timer;
        std::cout << std::endl;

        /*********************
         ***  MARKET DATA  ***
         *********************/

        Calendar calendar = TARGET();

        Date settlementDate(18, September, 2008);
        // must be a business day
        settlementDate = calendar.adjust(settlementDate);

        Integer fixingDays = 3;
        Natural settlementDays = 3;
```

Fixed Income in RQuantLib

Examples: Bond pricing

```
fixingDays <- 3
settlementDays <- 3
settlementDate <- as.Date('2008-09-18')
todaysDate <- settlementDate - fixingDays
#begin to 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)
```

Fixed Income in RQuantLib

Examples: Bond pricing

```
#begin to set up swap term structure
swp.tsr.params <- list(tradeDate=todaysDate,
                       settleDate=todaysDate+2,
                       dt=0.25,
                       interpWhat="discount",
                       interpHow="loglinear")
market.quotes <- list(d1w=0.043375, d1m=0.031875,
                      d3m=0.0320375, d6m=0.03385,
                      d9m=0.0338125, d1y=0.0335125,
                      s2y=0.0295, s3y=0.0323,
                      s5y=0.0359, s10y=0.0412,
                      s15y=0.0433)
depoSwpTsr <- DiscountCurve(swp.tsr.params,
market.quotes)
```

Fixed Income in RQuantLib
Examples: Bond pricing

```
#Zero-Coupon Bond
zc.bond.param <- list(
              maturityDate=as.Date('2013-08-15'),
              issueDate=as.Date('2003-08-15'),
              redemption=116.92)
zc.bond.dateparam <- list(
              refDate=todaysDate,
              settlementDays=settlementDays,
              businessDayConvention='Following')
ZeroCouponBond(zc.bond.param,
               bondDsctTsr,
               zc.bond.dateparam)
```

Fixed Income in RQuantLib

Examples: Bond pricing

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

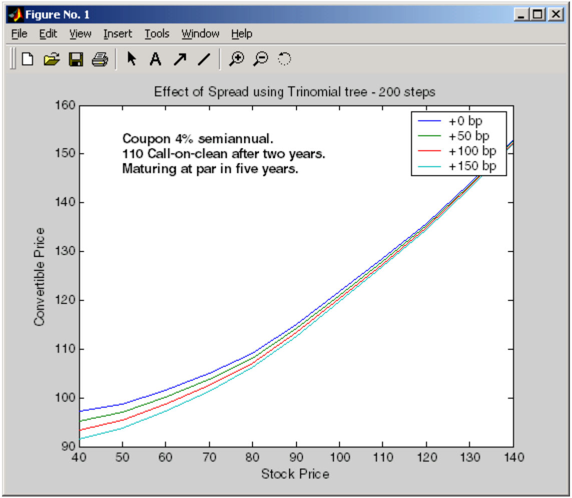
Fixed Income in RQuantLib

Examples: 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.

```
RiskFreeRate  = 0.05;
Sigma         = 0.3;
ConvRatio     = 1;
NumSteps      = 200;
IssueDate     = datenum('2-Jan-2002');
Settle        = datenum('2-Jan-2002');
Maturity      = datenum('2-Jan-2007');
CouponRate    = 0.04;
Period        = 2;
Basis         = 1;
EndMonthRule  = 1;
DividendType  = 0;
DividendInfo  = [];
CallInfo      = [datenum('2-Jan-2004'), 110];
CallType      = 1;
TreeType      = 1;
% Nested loop accross prices and static spread dimensions
% to compute convertible prices.
for j = 0:0.005:0.015;
StaticSpread = j;
      for i = 0:10:100
          Price = 40+i;
          [CbMatrix, UndMatrix, DebtMatrix, EqtyMatrix] = ...
           cbprice(RiskFreeRate, StaticSpread, Sigma, Price, ...
           ConvRatio, NumSteps, IssueDate, Settle, ...
           Maturity, CouponRate, Period, Basis, EndMonthRule, ...
           DividendType, DividendInfo, CallType, CallInfo, ...
           TreeType);

           convprice(i/10+1,j*200+1) =  CbMatrix(1,1);
           stock(i/10+1,j*200+1)     =  Price;
       end
end
```

```
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('\n'), ...
    '110 Call—on—clean after two years.' sprintf('\n'), ...
    'Maturing at par in five years.'],'fontweight','Bold')
```



```
params <- list(tradeDate=as.Date('2002-01-02'),
               settleDate=as.Date('2002-01-02'),
               dt=.25,
               interpWhat="discount",
               interpHow="loglinear")
times <- seq(0,10,.1)

RiskFreeRate <- DiscountCurve(params, list(flat=0.05),
                              times)
Sigma <- 0.3
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'))
process <- list(underlying=40, divYield=dividendYield,
                rff=RiskFreeRate, volatility=Sigma)

bondparams <- list(exercise="eu", faceAmount=100,
                   divSch=dividendSchedule,
                   callSch=callabilitySchedule,
                   redemption=100,
                   creditSpread=0.005,
                   conversionRatio=ConvRatio,
                   issueDate=issueDate,
                   maturityDate=maturityDate)
```

```r
dateparams <- list(settlementDays=3,
                   dayCounter="Thirty360",
                   period="Semiannual", calendar="us",
                   businessDayConvention="Following",
                   todayDate=issueDate)
coupon <- 0.04

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))
}
```

```
>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'
```