Interpolation of zero curves with R

Introduction

This work provides tools to implement a term structure of interest rates with R. The first part displays tools on how to interpolate market zero rates between data points using both linear and cubic spline interpolation. The second part consists on using results from Part1 to calculate forward rates and forward discount factors.

Part1 : Interpolation with Dates using R

Interpolation consists on using two main packages {xts} and {lubridate}. The latter helps to add days, months, or years to a given date, and the {xts} package provides both linear and cubic spline interpolation.

Assume that the market yield curve data on 2017-05-14 appears on a trader's desk as follows:

maturity	symbol	rates
Overnight	0N	0.08
One week	1W	0.125
One month	1M	0.15
Two months	2M	0.2
Three months	3M	0.255
Six months	6M	0.35
Nine months	9M	0.55
One year	1Y	1.65
Two years	2Y	2.25
Three years	3Y	2.85
Five years	5Y	3.1
Seven years	7Y	3.35
Ten years	10Y	3.65
Fifteen years	15Y	3.95
Twenty years	20Y	4.65
TwentyFive years	25Y	5.15
Thirty years	30Y	5.85

Then, we will use the {lubridate} package to replicate the above yield curve. We set 2017-05-14 as our anchor date.

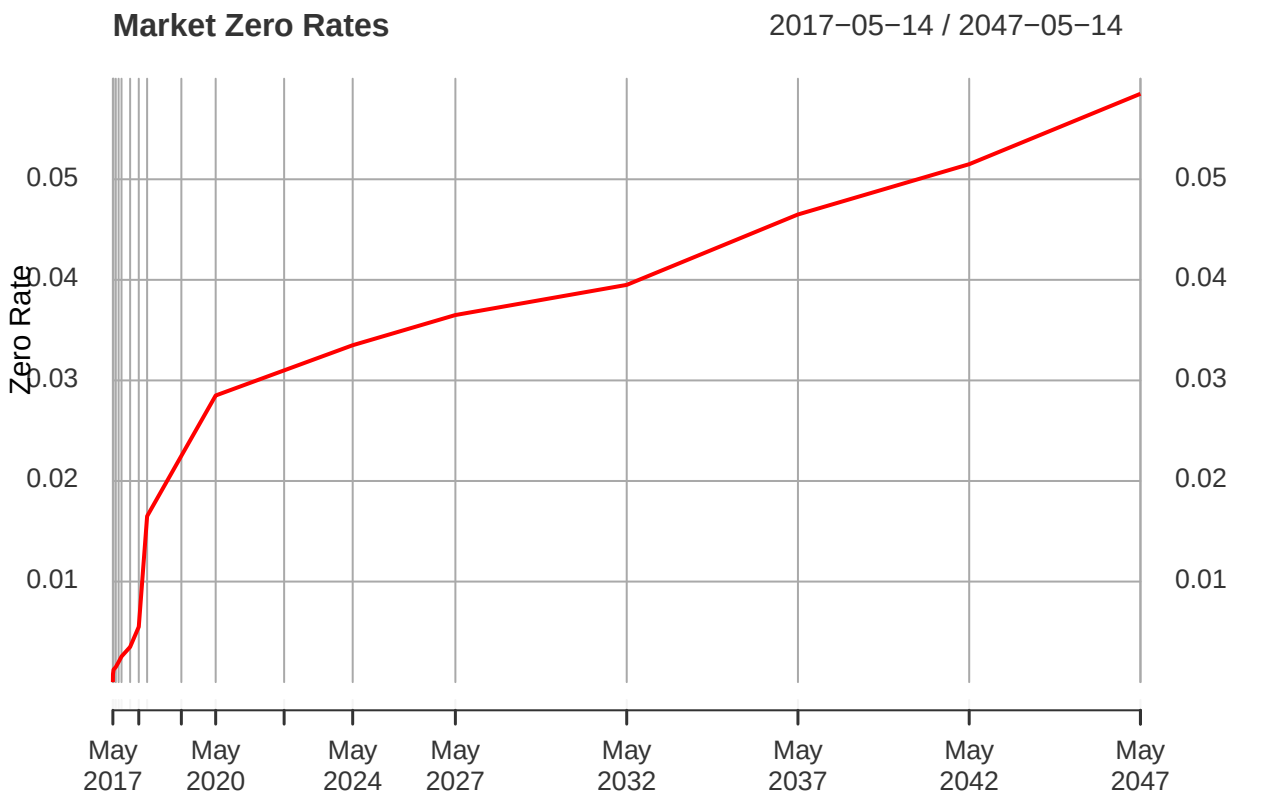
```
#AnchorDate = anchor date, tz = time zone

AnchorDate<- ymd(20170514, tz = "US/Pacific")
Dates <- c(AnchorDate, AnchorDate + days(1), AnchorDate + weeks(1), AnchorDate + months(1), AnchorDate +

AnchorDate + months(3), AnchorDate + months(6), AnchorDate + months(9), AnchorDate + years(1), AnchorDa

AnchorDate + years(20), AnchorDate + years(25), AnchorDate + years(30))

#get rid of "UTC"/time zone after the dates by using substring(.)
```

```
Dates <- as.Date(substring(Dates, 1, 10))

#Convert Rates to decimales

Rates <- c(0.0, 0.08, 0.125, 0.15, 0.20, 0.255, 0.35, 0.55, 1.65,
                          2.25, 2.85, 3.10, 3.35, 3.65, 3.95, 4.65, 5.15, 5.85) * 0.01

#Convert to xts

Data.xts <- as.xts(Rates, order.by = Dates)

head(Data.xts)
```

```
##                [,1]
## 2017-05-14 0.00000
## 2017-05-15 0.00080
## 2017-05-21 0.00125
## 2017-06-14 0.00150
## 2017-07-14 0.00200
## 2017-08-14 0.00255
```

And we can plot our market zero rates as follows:

```
colnames(Data.xts) <- "ZeroRate"
plot(x = Data.xts[, "ZeroRate"], xlab = "Time", ylab = "Zero Rate",
            main = "Market Zero Rates ", ylim = c(0.0, 0.06),
            major.ticks= "years", minor.ticks = FALSE, col = "red")
```



The following next steps will demonstrate how one might take market zero rates, place them into an xts object, and then interpolate the rates in between the data points using the aforementioned interpolation

functions.

First, we create an empty xts object with daily dates spanning the anchor date out to 30 years:

```
createEmptyTermStructureXtsLub <- function(anchorDate, plusYears)
{
# anchorDate is a lubridate here:
endDate <- anchorDate + years(plusYears)
numDays <- endDate - anchorDate
# We need to convert anchorDate to a standard R date to use
# the "+ 0:numDays" operation.
# Also, note that we need a total of numDays + 1 in order to capture both end points.
xts.termStruct <- xts(rep(NA, numDays + 1), as.Date(anchorDate) + 0:numDays)
return(xts.termStruct)
}
```

Then, using our anchor date ad (2017-05-14), we generate an empty xts object going out daily for 30 years:

```
termStruct <- createEmptyTermStructureXtsLub(AnchorDate, 30)
```

Next, we substitute in the known rates from our market yield curve:

```
numRates <- length(Rates)
for(i in (1:numRates)) termStruct[Dates[i]] <- Data.xts[Dates[i]]
```

Results are as follows: Note that we capture the market ratesat ON, 1W, and 30Y:

```
head(termStruct)
```

```
##             [,1]
## 2017-05-14 0e+00
## 2017-05-15 8e-04
## 2017-05-16    NA
## 2017-05-17    NA
## 2017-05-18    NA
## 2017-05-19    NA
```

```
tail(termStruct)
```

```
##              [,1]
## 2047-05-09     NA
## 2047-05-10     NA
## 2047-05-11     NA
## 2047-05-12     NA
## 2047-05-13     NA
## 2047-05-14 0.0585
```

Finally, we use the interpolation functions provided by the {xts} package to fill in the rates in between.

```
termStruct.lin.interpolate <- na.approx(termStruct)
termStruct.spline.interpolate <- na.spline(termStruct, method = "hyman")
colnames(termStruct.lin.interpolate)=colnames(termStruct.spline.interpolate)="ZeroRate"
head(termStruct.lin.interpolate)
```

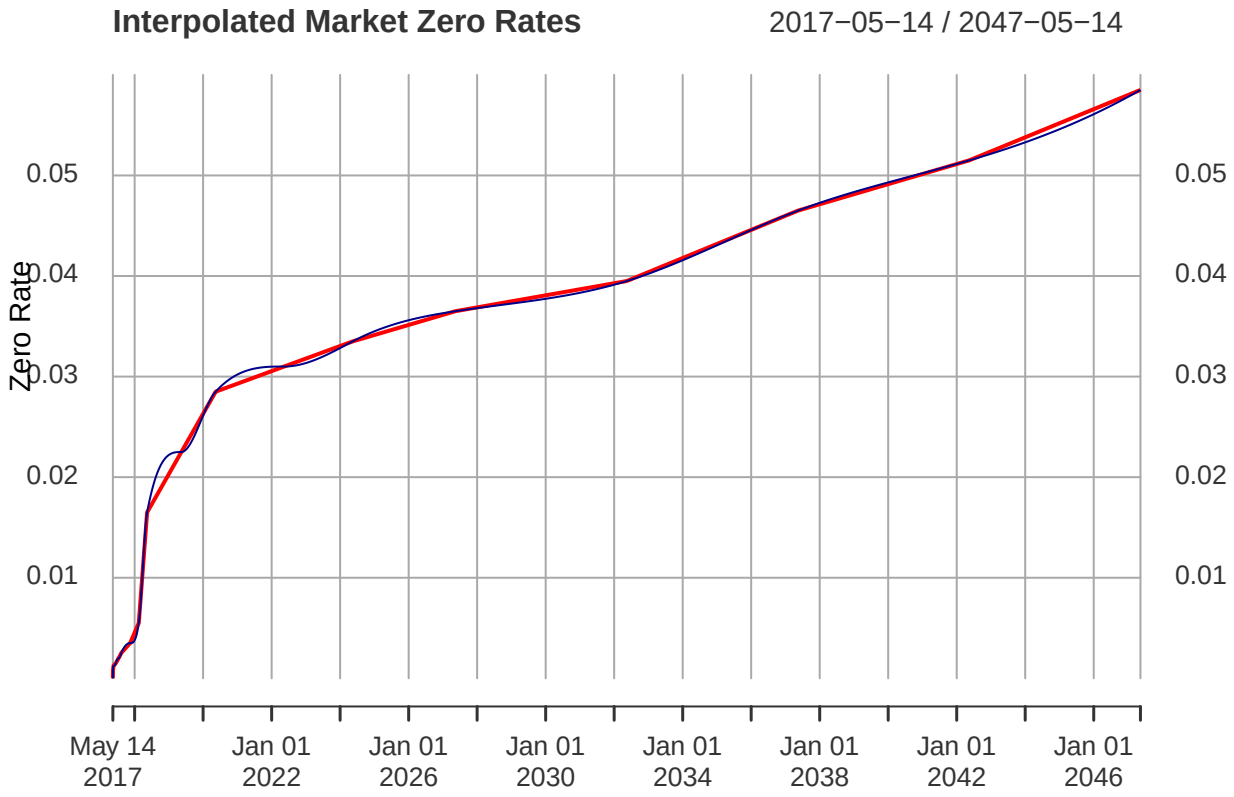
```
##            ZeroRate
## 2017-05-14 0.000000
## 2017-05-15 0.000800
## 2017-05-16 0.000875
## 2017-05-17 0.000950
## 2017-05-18 0.001025
```

```
## 2017-05-19 0.001100
```

```
head(termStruct.spline.interpolate)
```

```
##                ZeroRate
## 2017-05-14 0.0000000000
## 2017-05-15 0.0008000000
## 2017-05-16 0.0009895833
## 2017-05-17 0.0011166667
## 2017-05-18 0.0011937500
## 2017-05-19 0.0012333333
```

we can plot the interpolated curves as follows:



Part2 : forward rates and forward discount factors

The notation adopted is from chapter 1 of the book Interest Rate Models — Theory and Practice (2nd edition, Brigo and Mercurio, 2006).

A presentation by Damiano Brigo from 2007, which covers some of the essential background found in the book, is available here, from the Columbia University website.

First, we need to implement the Actual / 365 Fixed day count as a function:

```
#date1 and date2 are lubridate dates, so that we can easily carry out the subtraction of two dates
dayCountFcn_Act365F <- function(date1, date2)
{
    yearFraction <- as.numeric((date2 - date1)/365)
    return(yearFraction)
}
```

Next, we implement the forward discount function $P(t,T)$.

```
fwdDiscountFactor <- function(anchorDate, date1, date2, xtsMarketData, dayCountFunction)
{
    # Convert lubridate dates to base R dates in order to use as xts indices.
    xtsDate1 <- as.Date(date1)
    xtsDate2 <- as.Date(date2)
    if((xtsDate1 > xtsDate2) | xtsDate2 > max(index(xtsMarketData)) |
         xtsDate1 < min(index(xtsMarketData)))
         {
            stop("Error in date order or range")
         }

    # 1st, get the corresponding market zero rates from our
    # interpolated market rate curve:

    rate1 <- as.numeric(xtsMarketData[xtsDate1])      # R(0, T1)
    rate2 <- as.numeric(xtsMarketData[xtsDate2])      # R(0, T2)
    # P(0, T) = exp(-R(0, T) * (T - 0))   (A), with t = 0 <=> anchorDate
    discFactor1 <- exp(-rate1 * dayCountFunction(anchorDate, date1))
    discFactor2 <- exp(-rate2 * dayCountFunction(anchorDate, date2))

    # P(t, T) = P(0, T) / P(0, t)  (C), with t <=> date1 and T <=> date2
    fwdDF <- discFactor2/discFactor1

 return(fwdDF)
}
```

Now, we can compute the forward interest rate as follows:

```
fwdInterestRate <- function(anchorDate, date1, date2, xtsMarketData, dayCountFunction)
{
     if(date1 == date2) {
        fwdRate = 0.0 # the trivial case
     } else {
           fwdDF <- fwdDiscountFactor(anchorDate, date1, date2,
                                 xtsMarketData, dayCountFunction)

           # R(t, T) = -log(P(t, T)) / (T - t)    (B)
           fwdRate <- -log(fwdDF)/dayCountFunction(date1, date2)
            }
        return(fwdRate)
}
```

suppose we want to get the five year forward three-month discount factor and interest rates:

```
date1 <- AnchorDate + years(5)
date2 <- date1 + months(3)
fwdDiscountFactor(AnchorDate, date1, date2, termStruct.spline.interpolate,
                  dayCountFcn_Act365F)
```

```
## [1] 0.9919102
```

```
fwdInterestRate(AnchorDate, date1, date2, termStruct.spline.interpolate,
                dayCountFcn_Act365F)
```

```
## [1] 0.03222588
```