weekly_update_02_feb_16

Christian Carmona

February 17, 2014

Part I

Weekly update 02: feb 10 - feb 16, 2014

I dedicated this week to analyze different models for the term structure of interest rates, identifying the market and academic standards for quoting interest rates, and coding two functions: 1) for obtaining spot (zero-coupon) rates from yield to maturity, and 2) for pricing bonds given a of zero-coupon yield curve.

Note on coding language: I decided to use R as the main coding language. The reason for this is that my research is widely applicable to the functions carried by the International Reserve Division in the Central Bank of Mexico, my current employer, and they use R as the main software for data analysis. However, If it is necessary to create equivalent functions in Python to cover course objectives, I am open to translate all my developed code.

The main inputs of portfolio selection models are the expected values and covariances of the assets under consideration. In the case of fixed-income portfolios, the prices of the instruments are function of the interest rates observed in the market. Therefore, a theoretical model for the evolution of bond prices over time is needed.

1 Term Structure of interest rate

The *term structure of interest rates* at a given time is the functional relationship between spot interest rates and term to maturity [Wilhelm, 1995].

For the term structure to be observable in the market, zero-coupon bonds of different maturities must trade. In reality, only zero-coupon bonds of very few maturities (typically smaller than a year) trade. But an abundance of traded coupon bonds exist in the fixed income markets. Coupon bonds can be thought of as portfolios of zero-coupon bonds. Hence, only specific portfolios of zero-coupon bonds trade.

The problem is then to extract individual zero-coupon prices from the prices of particular portfolios of zero-coupon bonds. From these (theoretical) zero-coupon bond prices the corresponding spot rates can then be calculated.

Common alternatives for extracting Yield Curves from Bond Prices [Munk, 2011]:

Bootstrapping

Cubic Spline

The Nelson-Siegel parametrization

Market quoting for bond prices

Consider that we can observe two types of bonds in the market: 1) zero-coupon bonds with maturity of at most ne year, 2) Coupon with semi-annual coupons payments with maturities ranging from 0 to 30 years.

Denote z_t: t-year continuously compounding interest rate. The price at time t, for a bond maturing at time T with a face value of \$1 is:

For zero-coupon bonds:

$$B_t = e^{-t*z_t}$$

For coupon bonds:

$$P_t = c\sum_{t_i} e^{-t_i*z_{t_i}} + e^{-t*z_t} = c\sum_{t_i} B_{t_i} + B_t$$

where: c: coupon-rate, and t_i: time of coupon payments; $0 < t_i \leq t$

However, the market standard for quoting prices of these instruments are not their prices, neither the zero-coupon rates.

In the following, we examine the case of US governments bonds. In practive, we will find the following:

For zero-coupon bonds (T-bills), the quotes yield is the following "adjusted discount rate":

$d_t = \frac{1-B_t}{B_t} * \frac{365}{t}$

For coupon bonds (T-notes), the quote is the yield to maturity of the bond, i.e. the rate y_t such that the current price of the bond is equal to present value of the cashflows discounted with that same rate:

y_t is such that: $P_t = c\sum_{t_i} e^{-t_i*y_t} + e^{-t*y_t}$

If we have the quotes for d_t and y_t, we can easily obtain the prices of the bonds: B_t and P_t, and then calculate the spot rates using one of the three aforementioned methods.

Bootstrapping

This method is an iterative procedure for getting spot rates from given prices of coupon bonds.

Suppose we have the following prices: $B_{0.5}$, B_1, $P_{1.5}$, P_2, $P_{2.5}$, P_3, ...

It is easy to get spot rates from zero-coupon bonds by:

$z_t = -t * log(B_t)$

for $t = 0.5$ and $t = 1$, we can directly obtain $z_{0.5}$ and z_1.

For $t = 1.5$, we can use $P_{1.5}$ and get $B_{1.5}$ from the expression:

$$P_{1.5} = c * (B_{0.5} + B_1 + B_{1.5}) + B_{1.5}$$

then,

$$B_{1.5} = \frac{1}{1+c} * (P_{1.5} - c * (B_{0.5} + B_1))$$

which is a zero-coupon bond, from which we can get $z_{1.5}$

this algorithm is repeated for t={ 1.5, 2, 2.5, 3, ... }

Cubic Spline [McCulloch, 1975]

This method adjust a fuction $\bar{B}(t)$ to obtain the values of the discount factors for all times t,

Divide the time domain into intervals defined by the "knot points" $0 = \tau_0 \leq \tau_1 \leq ... \leq \tau_k$

The spline approximation of the discount function is given by:

$$\bar{B}(t) = \sum_{j=0}^{k-1} G_j(t) I_j(t)$$

where the $G_j(t)$'s are basis functions and the $I_j(t)$'s are step functions:

$$I_j(t) = \begin{cases} 0 & \text{if } t \geq \tau_j \\ 1 & \text{otherwise} \end{cases}$$

Hence:

$$\bar{B}(t) = \begin{cases} G_0(t) & \text{for } t \in [\tau_0, \tau_1) \\ G_0(t) + G_1(t) & \text{for } t \in [\tau_1, \tau_2) \\ ... & ... \\ G_0(t) + G_1(t) + ... + G_{k-1}(t) & \text{for } t \geq \tau_{k-1} \end{cases}$$

We demand that the $G_j(t)$'s are continuous, differentiable and ensure a smooth transition in the knot points.

This approach use a cubic polinomial spline:

$$G_j(t) = \alpha_j + \beta_j(t - \tau_j) + \gamma_j(t - \tau_j)^2 + \delta_j(t - \tau_j)^3$$

where $\alpha_j, \beta_j, \gamma_j, \delta_j$ are constants.

The smooth transition demand continuity in the function and its two derivatives:

$$\bar{B}(\tau_1-) = \bar{B}(\tau_1+)$$

$$\bar{B}'(\tau_1-) = \bar{B}'(\tau_1+)$$

$$\bar{B}''(\tau_1-) = \bar{B}''(\tau_1+)$$

with these assumptions, we can derive some of the constants, reducing the expression to:

$$\bar{B}(t) = 1 + \beta_0 t + \gamma_0 t^2 + \delta_0 t^3 + \sum_{j=1}^{k-1} \delta_j (t - \tau_j)^3 I_j(t)$$

To obtain the remaining constants, we adjust a linear model using the observed prices of bonds B_t and P_t

$$P_t = c \sum_{0 < t_i \leq t} \bar{B}(t_i) + \bar{B}(t) + \epsilon_t$$

The Nelson-Siegel parametrization [Nelson and Siegel, 1987]

This approach is based on a parametrization on the structure of the forward rates:

$$\bar{f}(t) = \beta_0 + \beta_1 e^{-t/\theta} + \beta_2 \frac{t}{\theta} e^{-t/\theta}$$

where $\beta_0, \beta_1, \beta_2, \theta$ are constants to be estimated and apply for all maturities.

Then, the term structure of zero-coupon rates is given by:

$$\bar{z}_t = \frac{1}{t} \int_0^t \bar{f}(u) = \beta_0 + \beta_1 \frac{1 - e^{-t/\theta}}{t/\theta} + \beta_2 (\frac{1 - e^{-t/\theta}}{t/\theta} - e^{-t/\theta})$$

which we can rewrite as:

$$\bar{z}_t = a + b\frac{1 - e^{-t/\theta}}{t/\theta} + ce^{-t/\theta}$$

the parameters a, b, c are obtained using the observed prices B_t for short terms using fitting a linear model:

$$z_{t_i} = a + b\frac{1 - e^{-t_i/\theta}}{t_i/\theta} + ce^{-t_i/\theta} + \epsilon_i$$

and for the coupon bonds, the expression is slightly more complicated and requires the use of Generalized Linear Models, because it is not linear in the unknown parameters.

2 Model Selection for my project

I decided to adopt **Bootstrapping method** for modeling the term structure in my project in view of the following:

1. Bootstrapping is a widely used methodology among practitioners
2. The underlying assumptions to get zero-coupon rates are consistent with the non-arbitrage criteria
3. For cubic spline, the value of the discount function for some maturities can often be determined by pure no-arbitrage arguments as utilized in the bootstrapping approach. The discount function estimated with cubic splines will not necessarily match those values so applications of the estimated function will not respect the fundamental no-arbitrage pricing principle.
4. For cubic spline, there is no guarantee whatsoever that the discount function estimated using cubic splines has an economically credible form. In particular, the discount function should be positive and decreasing (which will ensure positive forward rates), but there is nothing in the approach ensuring that.
5. For cubic and Nelson-Siegel, small variations in the input bond prices may have a substantial effect on the estimated discount function and yield curve. In particular, a change in the input price of a short-maturity bond may even affect the long-maturity end of the estimated curves.
6. For Nelson-Siegel, the necessity of incorporate estimation by Generalized Linear Models is way more complicated and computationally expensive.
7. For Nelson-Siegel, the slope and the curvature factors decay rapidly to zero as the maturity increases. Hence, it is difficult to fit medium- and long-term yields. An additional curvature term can be added to reduce this problem, but the complexity in the model is increased even more.

3 Coding

I developed the following two fuctions in R for implementing Bootstrapping method using historical real data

```
################################################

interpolation <- function( x, fx, x_new, method=c("linear","exp","log","spline")[1] ) {
  # This function obtain the interpolated values in x_new
  # For a given a set of points (x,fx)


  x_new_bucket <- rep(1,length(x_new))
  for( i in seq(x) ) {
    x_new_bucket <- ifelse(x_new>x[i], i+1, x_new_bucket)
  }
  x_new_bucket <- ifelse(x_new==x[1], 2, x_new_bucket)
  temp <- !is.element(x_new_bucket,c(1,length(x)+1))
```

```
  fx_new <- as.numeric(NULL)
  if(method=="linear") {
    # Smoothing via "Linear interpolation"
    w <- rep(NA,length(x_new))
    w[temp] <- (x[x_new_bucket[temp]]-x_new[temp]) / (x[x_new_bucket[temp]] - x[x_new_bucke
    fx_new <- rep(NA,length(x_new))
    fx_new[temp] <- fx[x_new_bucket[temp]-1] * w[temp] + fx[x_new_bucket[temp]] * (1-w[temp
  }
  if(method=="exp") {
    # Smoothing via "exponential interpolation"
    w <- rep(NA,length(x_new))
    w[temp] <- (x[x_new_bucket[temp]]-x_new[temp]) / (x[x_new_bucket[temp]] - x[x_new_bucke
    fx_new[temp] <- log( exp(fx[x_new_bucket[temp]-1]) * w[temp] + exp(fx[x_new_bucket[temp
  }
  if(method=="log") {
    # Smoothing via "Logarithmic interpolation"
    w <- rep(NA,length(x_new))
    w[temp] <- (x[x_new_bucket[temp]]-x_new[temp]) / (x[x_new_bucket[temp]] - x[x_new_bucke
    #fx_new[temp] <- exp( log(fx[x_new_bucket[temp]-1]) * w[temp] + log(fx[x_new_bucket[ten
    fx_new[temp] <- fx[x_new_bucket[temp]-1]^w[temp] * fx[x_new_bucket[temp]]^(1-w[temp])
  }
  if(method=="spline") {
    # Smoothing via "Smoothing spline"
    smooth_model <- smooth.spline(cbind(x,fx))
    fx_new <- predict(smooth_model,x_new)$y
  }
  return(fx_new)
}

##################################################

##################################################

zero_from_yield_bootstrap <- function ( nodes, ytm_curve, smooth=c("linear","exp","log","sp
  # Function to calculate zero-coupon rates, z_t, using bootstrapping
  # using data acording to market quotes, y_t
  #   for t > 1 year: anualized yield to maturity for bonds paying semiannual coupons
  #   fot t <=1 year: adjusted discount rate, i.e. ((100-P)/P)*(365/days to maturity)
  # the output rate will be annual and continuosly compounded

  # Input:
  #   ytm_curve: a numeric vector with yields in the market standard
  #   nodes: a numeric  vector with the terms of ytm_curve in years
  # Output
  #   zero_curve_boot :a numeric vector with the corresponding zero-coupon yields

  ytm_curve <- as.numeric(ytm_curve)

  nodes_t <- as.numeric(substr( nodes , 1, nchar(nodes)-1 ))
  nodes_t[substr( nodes , nchar(nodes), nchar(nodes) )=="m"] <- nodes_t[substr( nodes , nch
  if(!all(nodes_t==sort(nodes_t,decreasing=F))) {
    stop("Introduced values have to be increasing with respect to 'nodes'")
  }
```

```
# Get rid of NA's
nodes_orig_t <- nodes_t
nodes <- nodes[!is.na(ytm_curve)]
nodes_t <- nodes_t[!is.na(ytm_curve)]
ytm_curve <- ytm_curve[!is.na(ytm_curve)]

# Validates that the vector has first, last and one additional node.
if(
  !( all( is.element( nodes_orig_t[c(1,length(nodes_orig_t))] , nodes_t[c(1,length(nodes_
) { return( rep(NA,length(nodes_orig_t)) ) }

# Interpolation of values in the input yields for terms that are not specified but needed
ytm_pred_t <- sort(unique( c( nodes_t, seq(0.5,max(nodes_t),0.5) ) ))
ytm_pred <- interpolation(x=nodes_t,fx=ytm_curve,x_new=ytm_pred_t,method=smooth)

# Validates that the interpolations does not give negative values
if(any(ytm_pred<0)) { stop("The interpolation method is calculating negatives yields") }

# Zero-coupon yield calculation via bootstrappping
# Assumptions:
# 1) Quotation according to market stndards:
#   1.1) for t<=1 instruments are zero-coupon bonds
zero_curve_boot <- ytm_pred
zero_curve_boot[ytm_pred_t<=1] <- ( 1 + ytm_pred[ytm_pred_t<=1] * ytm_pred_t[ytm_pred_t<=
zero_curve_boot[ytm_pred_t>1] <- NA
#   validation
if( any(round( ( (1+ytm_pred[ytm_pred_t<=1]*ytm_pred_t[ytm_pred_t<=1])^(-1) ) - ( (1+zero
  stop("Zero-coupon rates were not calculated correctly for terms t<=1")
}

#   1.2) for t>1 instruments are coupon bonds, with payments at: 0.5, 1, 1.5, 2, 2.5,...
# ytm are nominales semi-anually compounded
node_coupon <- is.element(ytm_pred_t,seq(0.5,max(ytm_pred_t),0.5))

for( node_i in ytm_pred_t[ytm_pred_t>1] ) {

  ytm_i <- ytm_pred[match(node_i,ytm_pred_t)]

  # cupon rate that makes bond price=100
  cpn_i <- ytm_i/2

  # zero-rate for node_i
  zero_curve_boot[match(node_i,ytm_pred_t)] <-
    ( ( 1-cpn_i*sum( ( 1 + zero_curve_boot[node_coupon & (ytm_pred_t<node_i)] )^(-ytm_pre

  # Validation
  if( 100 != round( sum(100*cpn_i*(1+zero_curve_boot[node_coupon & (ytm_pred_t<=node_i)])
    stop("Zero-coupon rates were not calculated correctly for terms t>1")
  }
  #plot(zero_curve_boot,pch=20,col=2)
  #points(ytm_pred,pch=20,col=4)
  #legend("top",c("zero","ytm"),col=c(2,4),pch=19)

}
```

```
  return( zero_curve_boot[match(nodes_orig_t,ytm_pred_t)] )
}

#####################################################
```

In []: