Computing Tools for Fixed Income

Bjørn Eraker

Wisconsin School of Business

December 12, 2014

- Bloomberg commands
- Excel quotes through Bloomberg
- Excel quotes through Interactive Brokers
- Visual Basic programing
- Combining VB with streaming data

Bloomberg is available in the ASAP office and Library:

- FIT - Basic fixed income screen. Sub-screens with detailed bond issue data.
- BTMM - Another fixed income overview screen. Contains REPO, EURO DOLLAR futures, FED FUNDS futures, stock mkt. data, etc.
- ECO - economic calendar. Lists market relevant macro releases.
- FED - overview of FED related news and data
- USSW - Swap market overview screen.

Figure: BTMM screen in Bloomberg

Figure: USSW screen in Bloomberg.

Bjørn Eraker   Computing Tools for Fixed Income

Right click on a price to get a number of options:

- DES gives a description of the security. For a bond, it will give you the issue date, coupon, etc.
- YA - yield analysis gives several measures of YTM and also Duration measures.
- GP - gives the price graph

Figure: Right-click menu.

Bjørn Eraker   Computing Tools for Fixed Income

To use Bloomberg's excel functionality, you need to open excel on a BB terminal/ computer.

The function BDP returns live market data.

It takes two arguments

1. A security identifier. Typically the CUSIP # followed by an identifier such as "govt cusip"
2. A field identifier. Examples: BID, ASK, DUR (duration), CPN (coupon), PX_DIRTY_BID, MATURITY

The next page shows the BDP function call in the spreadsheet "discountFNs2014."

- Row1 : identifiers
- Column A: CUSIP codes

Example:
CELL D7 (highlighted in green) is calling =BDP($A7 & "govt cusip",D$1).

The first argument is "912828DM9" which is the cusip code in A7. The statement "$A7 & "govt cusip" is equal to "912828DM9 govt cusip"

This is a text-string. The second argument is also a text-string. In this case, NXT_CPN.

The BDP function then returns the next coupon date, 2/15/2015 in this case.

Figure: BDP function.

Bjørn Eraker   Computing Tools for Fixed Income

We will discuss the most elementary concepts in computer programing: How to write functions, and then to use those functions in Excel.
A simple function declaration looks like this:

```
Function sq(x)
        sq = x * x
End Function
```

You can now call this function in a spreadsheet. For example "=sq(2)" returns the number 4 in the spreadsheet.

Programming statements are always executed right-to-left.
It is sometimes useful to be able to do intermediate
calculations. If so, we might want to declare a local variable to
store the result temporarily. Local variables are cannot be
accessed outside the function. Here 'y' is a local variable

```
Function sq(x)
        y = x*x
        sq = y
End Function
```

The result of the multiplication $x \times x$ is stored temporarily in the
new (local) variable y. That number is then assigned to the
output "sq"

Note that since programing statements are executed right to left, this does not work

```
Function sq(x)
        x*x = y
        sq = y
End Function
```

and will return an error.

Here's an example of an "if"

```
Function lessThanFive(x)
        If (x<5) Then
                sq = 0
        Else
                sq = 1
        End if
End Function
```

If called with an argument less than 5, this function returns 0, otherwise 1.

One of the most important concepts in numerical programming is loops. We use loops to compute sums.
An example is

```
i =1
Do While i < n
  i = i + 1
Loop
```

The statement increases the index *i* from 1 to *n* (a variable) in increments of 1.

- Functions can take multiple arguments $f(x, y, z, ..)$
- Functions can call other functions
- We can call built in mathematical functions (i.e, EXP())
- We can also call Excel functions from VB

A real example:

```
Function NelsonSiegel(timeToExp, b1, b2, b3, lambda)
'' gives a single point along the yeild curve using the NS yie
    temp = Exp(-lambda * timeToExp)
    If (timeToExp < 0) Then
        NelsonSiegel = 0.0
        Else
        NelsonSiegel = b1 + b2 * (1 - temp) / (lambda * timeTo
         + b3 * ((1 - temp) / (lambda * timeToExp) - temp)
    End If

End Function
```

- This function takes 5 arguments
- The first line starts with " which means that it is a comment - not executable code
- The second line defines the temporary variable "temp" which is subsequently used. This avoids having to compute this multiple times
- The if statement is actually not necessary if the function is always called with a positive timeToExp as it should