Using the R/Reuters SFC Plugin

Rory Winston

December 8, 2008

1 Introduction

This short article is a practical introduction to the R/Reuters time series interface, in which we will explore some financial data relevant to an exposition of some of the aspects of the global financial crisis.

In order to initialize the underlying market data feed provider, we can call the `init` function, passing in the required parameters. The signature of the init method is:

```
init(serviceName, configPath, appId, serverName)
```

Where the parameters are as follows (see the SFC documentation for information on the actual parameters):

- **serviceName**: the underlying market feed service (defaults to "IDN_SELECTFEED");
- **configPath**: a pointer to a config file containing the required SFC configuration (defaults to sslapi.cnf);
- **appId**: a textual identifier for the application (defaults to "R.reuters-client");
- **serverName**: a service list (for example "server1 server2").

In the following snippet of initialization code, we load the interface DLL (bear in mind that as the SFC APIs run on Windows and Linux, this DLL could theoretically be compiled to a .so file, although I have not had the opportunity to test this yet), and initialize the subsystem with an appropriate set of parameters:

```
> options(digits.secs = 6)
> dyn.load("reuters_ts.dll")
> init <- function(serviceName = "IDN_SELECTFEED", configPath = "sslapi.cnf",
+     appId = "R.reuters-client", serverName = "lonmdddist1 lonmdddist2") {
+     .Call("init", serviceName, configPath, appId, serverName)
+ }
```

In order to retrieve data, a function, `fetch()` is provided. This function has the following signature:

```
fetch(code, n, t, sd, ed, rev)
```

Where the parameters are as follows:

- **code**: The Reuters Information Code (RIC) for the item of daa being requested.
- **n**: An optional parameter for the number of items to receive.
- **t**: An optional parameter specifying the periodicity of items to retrieve. The default is `"w"`, for weekly observations. Note that the limited time series data available on TS1 does not include intraday data.
- **sd**: The start date for data to be retrieved (mm/dd/yy format).
- **ed**: The end date for data to be retrieved (mm/dd/yy format).
- **rev**: Determines the ordering of the data frame. Default is `TRUE` (most recent items last).
- **debug**: If `TRUE`, prints some extra debugging information.

The number of items to be retrieved can be specified in the following ways:

- Specify the number of items using `n`; or
- Specify a start date (end date will default to today); or
- Specify a start date and end date.

Specifying a start or end date that lie outside of the available points does not normally cause an error - for instance, specifying an end date later than the last date available for a particular instrument will result in the last available date being returned as the endpoint. Similarly, specifying a start date earlier than the earliest start date for that item should result in the earliest available observation being returned.

The `fetch` function is defined as follows:

```
> fetch <- function(code = "EUR=", n = 0, t = "d", sd = NULL, ed = NULL,
+     rev = TRUE, debug = FALSE) {
+     if (!is.null(sd)) {
+         if (is.null(ed)) {
+             ed <- format((Sys.Date()), format = "%m/%d/%Y")
+         }
+     }
+     if (debug) {
+         print(paste("fetch(code=", code, ",n=", n, ",sd=", sd,
+             ",ed=", ed, ")"))
+     }
+     frame <- data.frame(.Call("fetch", code, n, t, sd, ed, debug))
+     frame$Date <- as.Date(strptime(frame$Date, "%m/%d/%Y"))
+     if (rev) {
+         frame <- frame[order(frame$Date), ]
+     }
+     frame
+ }
> init()
```

2 A Whirlwind Tour of the R/Reuters Plugin

2.1 Using the Periodicity Parameter

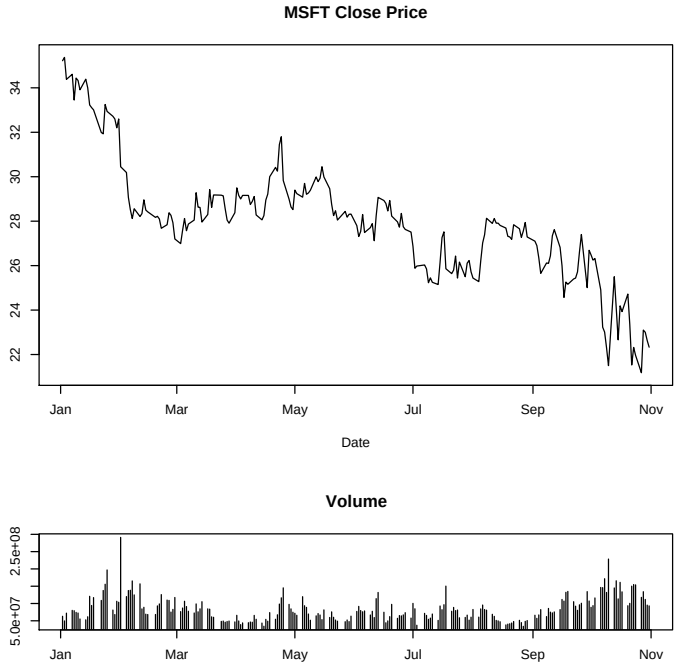
The following example uses the periodicity parameter to retrieve the value of the GBP/USD contributed price at the monthly, weekly, and daily levels:

```
> gbp.m <- fetch("GBP=", n = 1000, t = "m")
> gbp.w <- fetch("GBP=", n = 1000, t = "w")
> gbp.d <- fetch("GBP=", n = 1000, t = "d")
> head(gbp.m)

          Date    BID    OPN     HI     LO    ASK
123 1998-09-30 1.6981 1.6810 1.7145 1.6510 1.6988
122 1998-10-31 1.6747 1.6998 1.7366 1.6610 1.6752
121 1998-11-30 1.6479 1.6765 1.6799 1.6444 1.6484
120 1998-12-31 1.6539 1.6479 1.6940 1.6462 1.6544
119 1999-01-31 1.6455 1.6580 1.6718 1.6228 1.6460
118 1999-02-28 1.6025 1.6473 1.6479 1.5949 1.6035

> head(gbp.w)

          Date    BID    OPN     HI     LO    ASK
281 2003-07-25 1.6195 1.5791 1.6236 1.5776 1.6203
280 2003-08-01 1.6107 1.6199 1.6310 1.6000 1.6112
279 2003-08-08 1.6026 1.6111 1.6188 1.6006 1.6034
278 2003-08-15 1.5968 1.6032 1.6138 1.5922 1.5973
277 2003-08-22 1.5745 1.5973 1.5985 1.5699 1.5750
276 2003-08-29 1.5772 1.5742 1.5828 1.5616 1.5777

> head(gbp.d)

          Date    BID    OPN     HI     LO    ASK
589 2006-09-05 1.8941 1.9059 1.9061 1.8911 1.8943
588 2006-09-06 1.8840 1.8934 1.8957 1.8790 1.8845
587 2006-09-07 1.8755 1.8853 1.8870 1.8705 1.8760
586 2006-09-08 1.8656 1.8762 1.8775 1.8626 1.8658
585 2006-09-11 1.8652 1.8652 1.8706 1.8599 1.8657
584 2006-09-12 1.8739 1.8645 1.8772 1.8623 1.8744
```

2.2 Creating a Price/Volume Chart

This example creates a price/volume chart for MSFT:

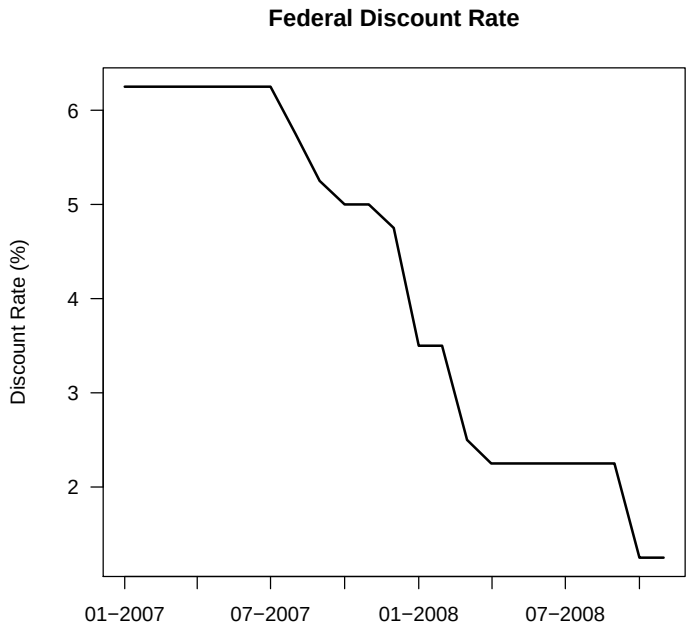
```
> msft.vol <- fetch("MSFT.O", sd = "1/1/2008", ed = "10/31/2008")
> layout(matrix(c(1, 1, 1, 1, 2, 2), 3, 2, T))
> plot(msft.vol$Date, msft.vol$CLS, main = "MSFT Close Price",
+     type = "l", ylab = "", xlab = "Date")
> plot(msft.vol$Date, msft.vol$VOL, main = "Volume", type = "h",
+     ylab = "", xlab = "")
```



2.3 Central Bank Interest Rates

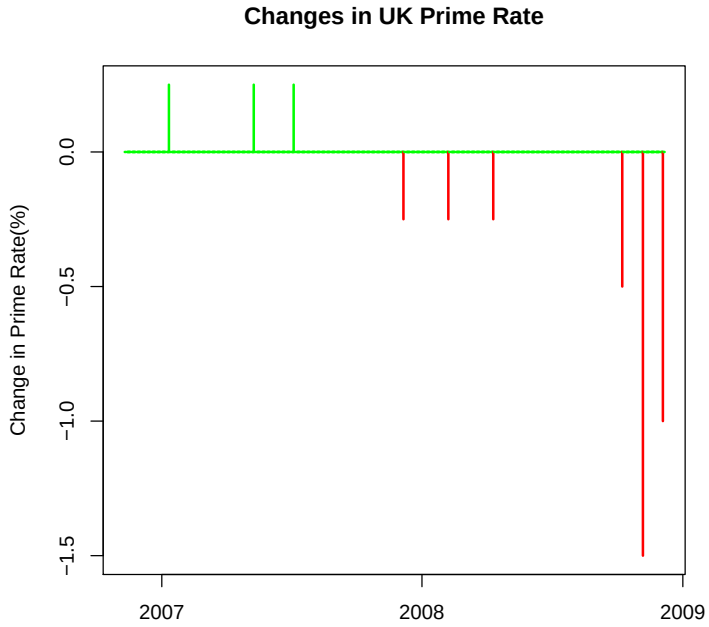
The Fed discount rate is the interest rate at which eligible financial institutions may borrow directly from the Federal Reserve acting in its capacity as lender of last resort. The discount rate has plummeted more recently as the Fed has made it cheaper for banks to borrow from it to fulfill their reserve requirements.

```
> fed.discount <- fetch("USDISC=", sd = "1/1/2007", t = "m")
> plot(fed.discount$Date, fed.discount$CLS, main = "Federal Discount Rate",
+     type = "l", lwd = 2, xaxt = "n", ylab = "Discount Rate (%)",
+     las = 2)
> axis.Date(1, at = seq(fed.discount$Date[1], fed.discount$Date[length(fed.discount$Date)]
+     "3 months"), format = "%m-%Y")
```



Similarly, the UK's central bank slashed its prime rate in an unprecedented 1.5% reduction in November 2008, the magnitude of which can be more fully appreciated by lookin at it in its historical context. Note in this example that we are using the fact that the `col` argument is vectorized, to color the points differently based on a boolean condition (in this case, whether the interest rate change is positive or negative).

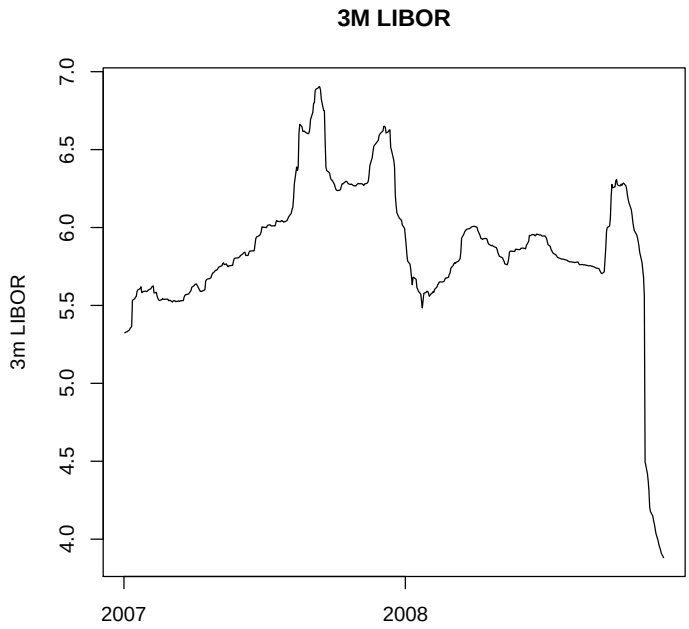
```
> gbprime <- fetch("GBPRIME=", sd = "1/1/2006", ed = "12/5/2008",
+     t = "d")
> plot(gbprime$Date[-1], diff(gbprime$CLS), type = "h", lwd = 2,
+     ylab = "Change in Prime Rate(%)", col = ifelse(diff(gbprime$CLS) <
+         0, "red", "green"), main = "Changes in UK Prime Rate")
```



2.4 LIBOR Rates

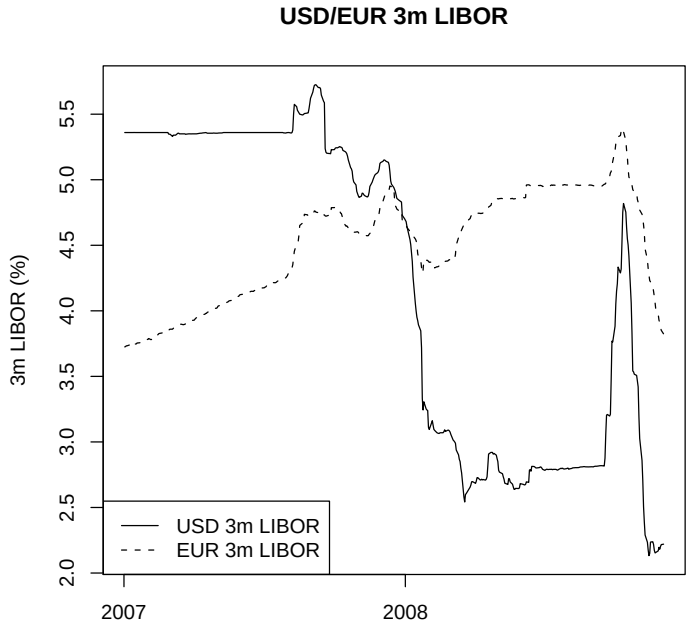
But if central banks have become willing to lend to member banks at reduced rates, what about the willingness of banks to lend to each other? One standard liquidity benchmark is the 3m LIBOR rate. Here we can see that banks' unwillingness (or inability) to lend to each other reached a peak in mid-2007, and has since declined sharply.

```
> libor.3m <- fetch("GBP3MFSR=", sd = "1/1/2007", ed = "12/1/2008",
+     t = "d")
> plot(libor.3m$Date, libor.3m$CLS, type = "l", main = "3M LIBOR",
+     ylab = "3m LIBOR")
```



The next plot shows USD 3m LIBOR and EUR 3m LIBOR:

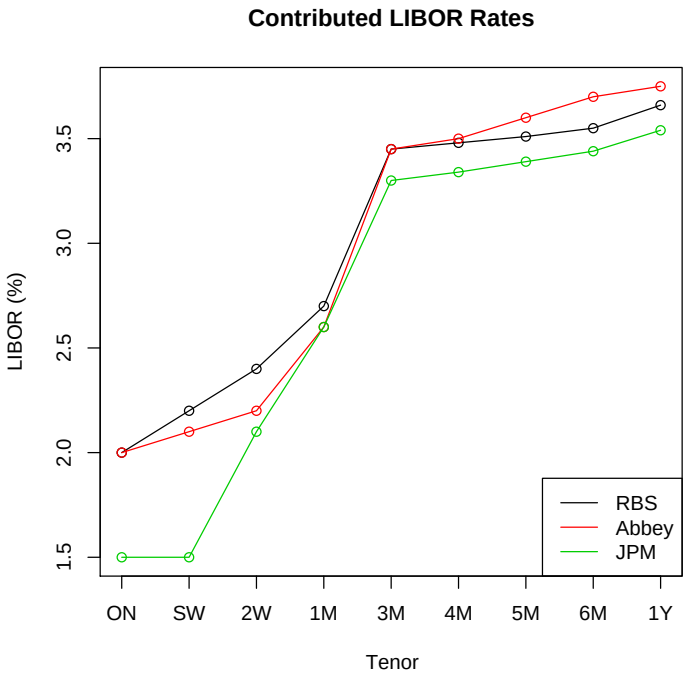
```
> usd.libor3m <- fetch("USD3MFSR=", sd = "1/1/2007", ed = "12/1/2008",
+     t = "d")
> eur.libor3m <- fetch("EUR3MFSR=", sd = "1/1/2007", ed = "12/1/2008",
+     t = "d")
> plot(usd.libor3m$Date, usd.libor3m$CLS, type = "l", ylim = range(usd.libor3m$CLS,
+     eur.libor3m$CLS), main = "USD/EUR 3m LIBOR", ylab = "3m LIBOR (%)")
> lines(eur.libor3m$Date, eur.libor3m$CLS, lty = 2)
> legend("bottomleft", legend = c("USD 3m LIBOR", "EUR 3m LIBOR"),
+     lty = c(1, 2))
```



LIBOR is a trimmed average calcuation based on the current bid rates from certain banks. We can construct a forward LIBOR curve for 1 year using the individual contributions from various banks (using the Reuters contributor code for each individual bank):

```
> codes <- c("RBSL", "ABBL", "JPML")
> tenors <- c("GBPONFSR=", "GBPSWFSR=", "GBP2WFSR=", "GBP1MFSR=",
+     "GBP3MFSR=", "GBP4MFSR=", "GBP5MFSR=", "GBP6MFSR=", "GBP1YFSR=")
> rics <- apply(expand.grid(tenors, codes), 1, function(x) paste(x,
+     collapse = ""))
> rbs.libor <- sapply(rics[grep("RBSL", rics)], function(x) fetch(x,
+     n = 1))
> jpm.libor <- sapply(rics[grep("JPML", rics)], function(x) fetch(x,
+     n = 1))
> abby.libor <- sapply(rics[grep("ABBL", rics)], function(x) fetch(x,
+     n = 1))
> yrange <- range(unlist(rbs.libor[2, ]), unlist(jpm.libor[2, ]),
+     unlist(abby.libor[2, ]))
> plot(1:length(rbs.libor[2, ]), unlist(rbs.libor[2, ]), xaxt = "n",
+     type = "o", main = "Contributed LIBOR Rates", ylab = "LIBOR (%)",
+     ylim = yrange, xlab = "Tenor")
> lines(1:length(abby.libor[2, ]), unlist(abby.libor[2, ]), type = "o",
+     col = 2)
> lines(1:length(jpm.libor[2, ]), unlist(jpm.libor[2, ]), type = "o",
+     col = 3)
> axis(1, labels = unique(sub("GBP(\\w\\w).*", "\\1", rics)), at = 1:length(rbs.libor[2,
+     ]))
```

```
> legend("bottomright", legend = c("RBS", "Abbey", "JPM"), col = 1:3,
+     lty = 1)
```



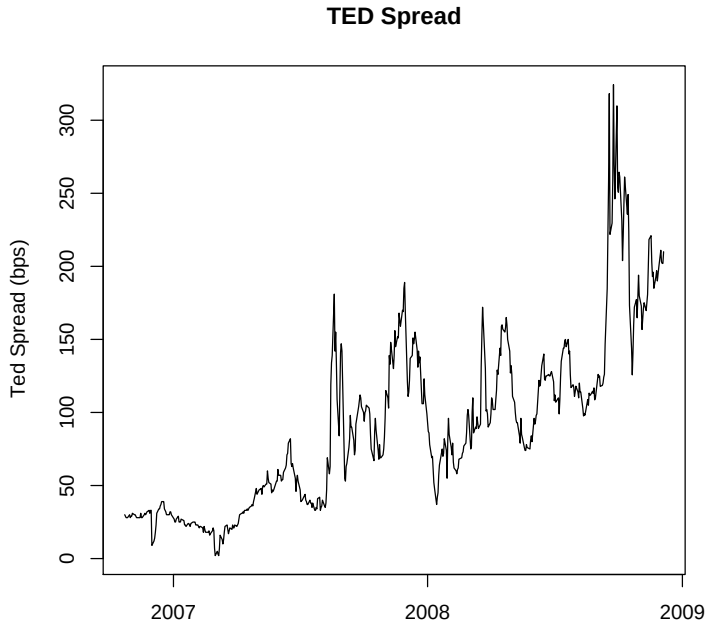
2.5 The TED Spread

An indicator sometimes used to gauge the level of credit risk in the economy is the TED (T-Bill / Eurodollar futures) spread. This is the spread between the current liquid lending rate (as measured by the 3m Eurodollar futures) and the riskless 3m T-Bill rate. A spike in the TED spread implies an increase in credit risk and a lack of liquidity.

```
> ted <- fetch("TED", sd = "1/1/2006", t = "d")
> max(ted$CLS)
```

```
[1] 324.32
```

```
> plot(ted$Date, ted$CLS, main = "TED Spread", type = "l", ylab = "Ted Spread (bps)")
```



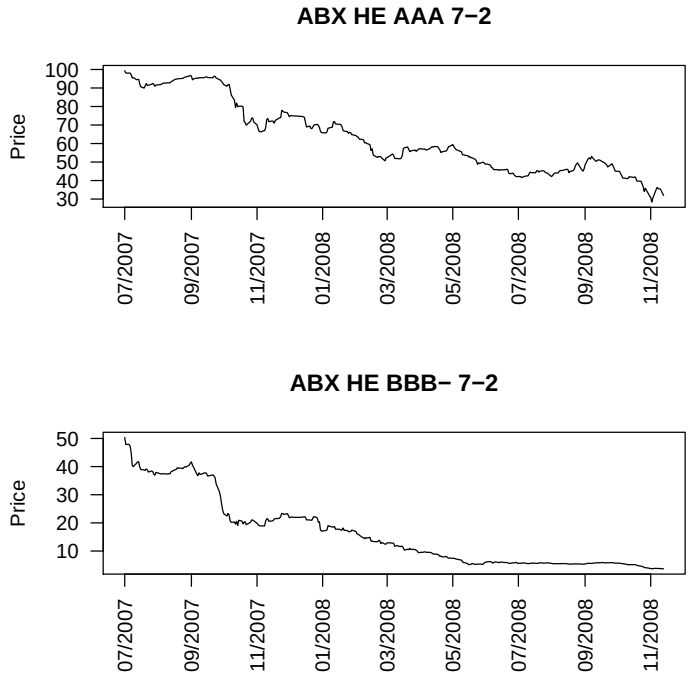
2.6 Subprime Loan Indices

One prominent indicator of the financial crisis has been the market for mortgage loan insurance and derivative products, and the most closely-watched range of products was the family of ABX indices, which were a family of indices based on underlying bonds backed by subprime mortgage loans. Many of these indices no longer have the required liquidity to function, but they serve as an effective reminder of the decline in subprime loans. A decline in the value of the index reflects a rise in the cost of insuring the underlying loans.

The following plots show the change in value of the highest and lowest-rated ABX index products. What the graphs show clearly is that even the highest-rated products suffered a catastrophic loss of value.

```
> abx.aaa <- fetch("ABXAA38FGB=MP", sd = "1/1/2007", ed = "12/1/2008",
+     t = "d")
> abx.bbb <- fetch("ABXBM38FGB=MP", sd = "1/1/2007", ed = "12/1/2008",
+     t = "d")
> op <- par(mfcol = c(2, 1), las = 2)
> plot(abx.aaa$Date, abx.aaa$BID, type = "l", main = "ABX HE AAA 7-2",
+     ylab = "Price", xaxt = "n")
> axis.Date(1, at = seq(abx.aaa$Date[1], abx.aaa$Date[length(abx.aaa$Date)],
+     "2 months"), format = "%m/%Y")
> plot(abx.bbb$Date, abx.bbb$BID, type = "l", main = "ABX HE BBB- 7-2",
+     ylab = "Price", xaxt = "n")
> axis.Date(1, at = seq(abx.bbb$Date[1], abx.bbb$Date[length(abx.aaa$Date)],
```

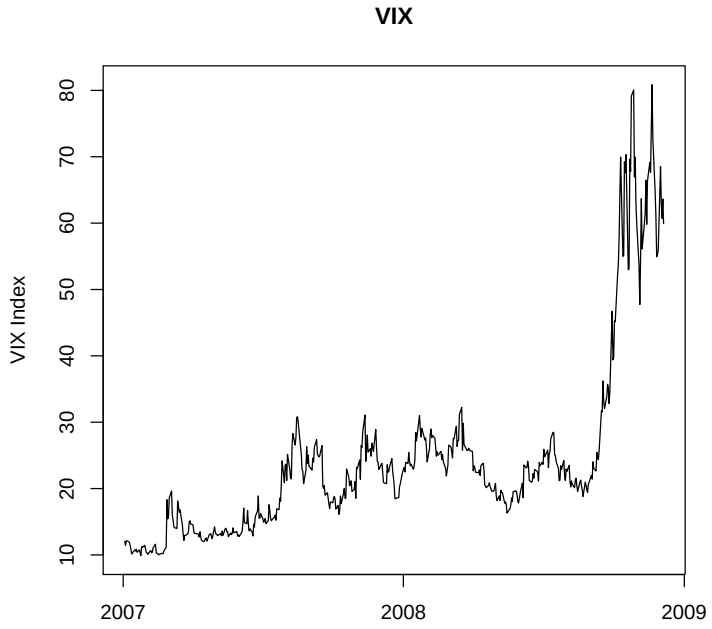
```
+     "2 months"), format = "%m/%Y")
> par(op)
```



2.7 The VIX Index

The Vix index is a common proxy for market volatility measurement. If the markets are driven by "fear and greed", the VIX is a good indicator of the prominence of the "fear" portion. Using a weighted combination of option prices on the S&P 500 index, the VIX is a measure of the current estimated implied volatility of the S&P 500 index over the next 30 days. Notice the huge spike in implied volatility in late 2008.

```
> vix <- fetch(".VIX", sd = "1/1/2007")
> plot(vix$Date, vix$CLS, type = "l", main = "VIX", ylab = "VIX Index")
```



At the time of writing, the annualized VIX level (as measured by $\frac{\text{VIX}_n}{\sqrt{12}}$) = 12.04 %.

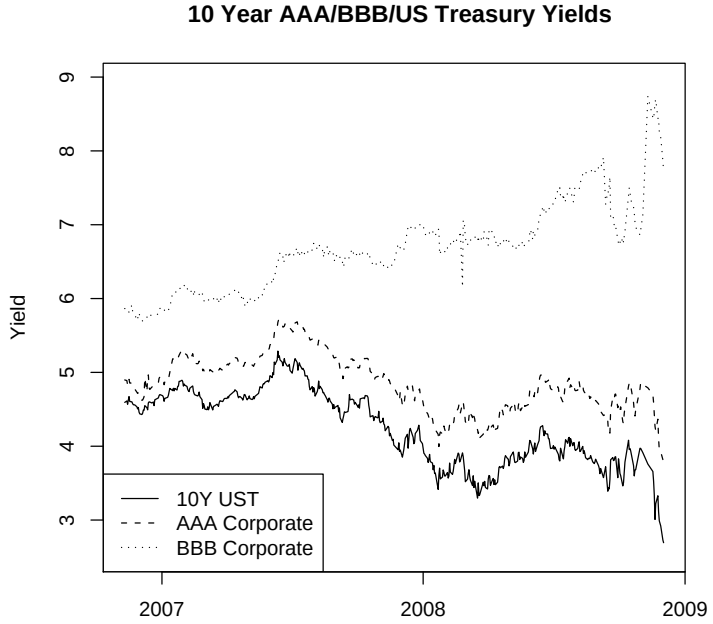
2.8 Corporate Credit Spreads

A widening of the credit spread charged to AAA-rated and lower investment grade corporations may indicate a lack of confidence amongst lenders in a deteriorating economic environment. The following plot shows the calculated yield on AAA and BBB corporate bonds, and the calculated yield on the US 10 year treasury bills.

In this example, we use the `zoo` library, which is invaluable when dealing with irregularly-spaced time series data. We use the `merge.zoo()` function to calculate the intersection (by calendar date) of three time series.

```
> library(zoo)
> aaa10y <- fetch("AAAUSD10Y=", n = 600, t = "d")
> bbb10y <- fetch("BBBUSD10Y=", n = 600, t = "d")
> ust10y <- fetch("US10YT=RR", n = 600, t = "d")
> aaa10y.series <- zoo(aaa10y, order.by = aaa10y$Date)
> bbb10y.series <- zoo(bbb10y, order.by = bbb10y$Date)
> ust10y.series <- zoo(ust10y, order.by = ust10y$Date)
> full.series <- merge(aaa10y.series, bbb10y.series, ust10y.series,
+     all = FALSE)
> ylims <- range(aaa10y$CYLD, bbb10y$CYLD, ust10y$CYLD)
> plot(index(full.series), full.series$CYLD.ust10y.series, type = "l",
+     ylim = ylims, main = "10 Year AAA/BBB/US Treasury Yields",
+     ylab = "Yield")
```

```
> lines(index(full.series), full.series$CYLD.aaa10y.series, lty = 2)
> lines(index(full.series), full.series$CYLD.bbb10y.series, lty = 3)
> legend("bottomleft", lty = c(1, 2, 3), legend = c("10Y UST",
+     "AAA Corporate", "BBB Corporate"))
```



3 Summary

I hope this gives you a quick flavour of what is possible using the Reuters/SFC time series extension. Note that the time series data that is obtainable via the TS1 service is quite limited compared to that in the DBU, however DBU access incurs an extra cost. The extension was compiled and tested using Visual Studio 2005, R 2.8.0, Reuters SFC 4.5.5, and Windows XP. I have not had a chance to compile the extension under Linux, though it should be cross-compilable using the Reuters APIs for Linux without too much difficulty.