SMF Solutions 6. 29.11.2013

Q1. *ARMA fitting and the Box-Jenkins methodology*
(i) We first start by loading the Dow Jones data and create the log-return series.

```
 dj<-read.csv(file="DJ07.csv",header=T)[,2]
logreturns<-log(dj[length(dj):2]/dj[(length(dj)-1):1])
```

(ii) The ACF and PACF do not, in this case really give a clear answer for what model would fit. (Be **very** careful with the ACF function in R, the first bar that is displayed is with lag zero, and hence always have amplitude one: the series is obviously perfectly correlated with itself. It is one of the few "R traps" for whoever does not pay close attention to what is displayed).
(iii) It is good practice before starters to check for the existence of unit roots, e.g. by using the augmented Dickey-Fuller test, called by

```
library(tseries)
adf.test(logreturns)
```

returns

```
        Augmented Dickey-Fuller Test

data:  logreturns
Dickey-Fuller = -11.6831, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(logreturns) : p-value smaller than printed p-value
```

Hence, it is reasonable to conclude that the series is stationary enough and that there is no unit root. Had it not been, we would have had to differentiate the series, which would have brought us to ARIMA models rather than ARMA.

Now, we use the **forecast** package to fit an ARMA according to the Akaike information criterion.

```
library(forecast)
armafit<-auto.arima(logreturns,stepwise=T,ic="aic")
```

When typing $armafit$ in command line, one can see the model retained.

```
 Series: logreturns
ARIMA(0,0,2) with zero mean

Coefficients:
          ma1      ma2
```

```
      -0.1277  -0.0411
s.e.   0.0248   0.0254

sigma^2 estimated as 0.0001845:  log likelihood=4754.88
AIC=-9503.76   AICc=-9503.74   BIC=-9487.53
```

So the most likely model is an MA(2) model.

Here, we are not quite done yet: one still needs to check that the model assumptions are satisfactory. In fact, assuming uncorrelated errors (homoskedasticity) is a strong assumption that need to be checked. That is the purpose of the Ljung-Box test applied to the residuals, which we do not discuss in detail here. It is essentially a hypothesis test where the null hypothesis is that there is no correlation among the residuals. The test takes one parameter l as an input corresponding to the number of lags one wants to investigate the absence on correlation on. It is an asymptotic test, so one should not take the parameter too small, but if the parameter is too big the power of the test decreases to zero. $l = 20$ or $l = log(n)$ where n is the sample size are commonly found in textbooks. When looking at several lag values, notably 20, by using

```
Box.test(armafit$residuals, type="Ljung-Box",lag=20)
```

the p-value ends up being smaller than 10^{-3}, which casts some doubts on the assumption of homoskedasticity.
(iv) Now, running the same procedure for Coca-Cola returns interesting results: the best model is an ARMA(0,0) with zero mean, and the Ljung-Box test on residuals gives p-values close to one, so the data is very close to a white noise.

As a conclusion, this exercises presents two extreme cases where the data behaves either badly (which is fairly common for financial data, homoskedasticity being a strong hypothesis) or very well (a company such as Coca-Cola whose returns are unaffected by their past values).
Q2.(GARCH fitting)

The GARCH(p,q) model, in particular the GARCH(1,1) model is widely used to model volatility clustering in financial data. Essentially, there are times where the assumption that the error in the ARMA model has constant variance is inappropriate (see above!). This is the case for some financial time series where one can indeed see that big variations are followed by other high variations; this is the reason why ARCH and GARCH models were introduced. The purpose of this exercise is to fit a GARCH(1,1) model to the S&P500 volatility time series. The data is available on my webpage, under the SMF section. One of the requirements of GARCH is to have a fairly big data sample (typically at least 500-700 observations), hence it is better used for intra-day or daily data than, say, monthly data. As to why the parameters are set to (1,1), as is commonly done in practice; we quote R. Engle: "The GARCH (1,1) model can be generalized to a GARCH(p,q) model; that is, a model with additional lag terms. Such higher-order models are often useful when a long span of data is used, like several decades of daily data or a year of hourly data. With additional lags, such models allow both fast and slow

decay of information." The full article can be downloaded here:
http://www.cmat.edu.uy/~mordecki/hk/engle.pdf

(i) and (ii) The fitting of a Garch model is somehow very straightforward, with only 4 lines of code (we divide the series by 100 to work with the real values rather than percentages):

```
>library(fGarch)
>vix<-read.csv(file="CBOEVIX_data.csv",header=T)[,2]/100
>garchmodel<-garchFit(~garch(1,1),data=vix)
>coef(garchmodel)
```

One gets

```
           mu         omega        alpha1         beta1
1.838575e-01 9.702114e-05 7.967775e-01 1.970972e-01
```

Let us recall what the above parameters mean: the series is modeled as

$$x_t = \mu + \epsilon_t$$

with

$$\begin{aligned} \epsilon_t &= \sigma_t z_t \\ \sigma_t^2 &= \omega + \alpha_1 \epsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \end{aligned}$$

and z_t a white noise.

Using the plot command on the GARCH type object yields the following:

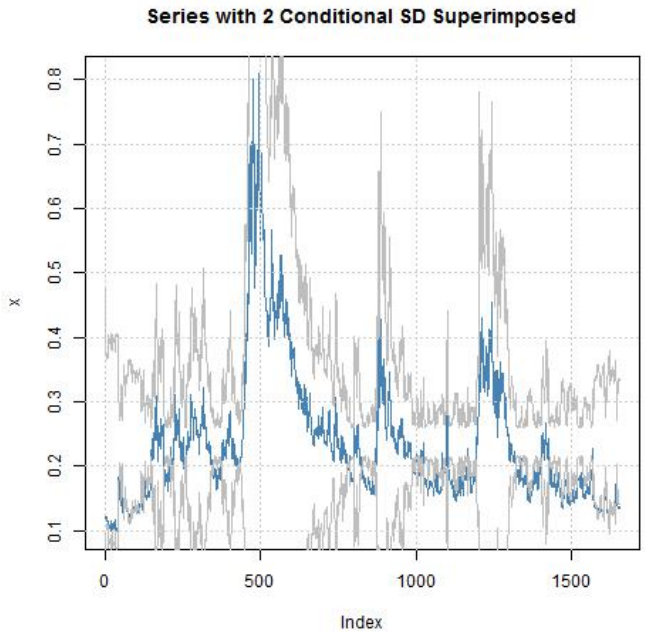

Figure 1: Original time series in blue, standard deviation envelope generated by the GARCH modelling in grey

Complement

There are essentially two ways of fitting a GARCH model in R. The first one is the **garchFit** function in the **fGarch** package, the second one is by using the **garch** command. Once again, note that the GARCH model really only models the error parameter. If one wants to model, say the S&P500 itself, it could be done by fitting an ARMA(p,q)+GARCH(1,1). In that sense, the **garchFit** function is more flexible and allows to do this kind of thing. For example, if one wants wants to fit the S&P500 itself with and ARMA(1,1)+GARCH(1,1) (available as a "bonus" on my webpage), it could be done using the following line.

```
sp<-garchFit(~arma(1,1)+garch(1,1),data = datasp)
```

NHB/PMBF