Introduction to Financial Engineering (W4)

Time Series: Macroeconomic Studies

1 Context

As commented in the previous chapter, empirical observation suggest that $ARMA$ process might not be enough to model macroeconomic data as they lack the ability to model volatility clustering. Since $ARMA$ models are stationary, their unconditional mean and variance are constant, and hence to capture volatility clustering we will need to model the conditional variance of the process.

To gain some heuristic concept on the purpose of the following sections, let us take a general autoregression model as the ones we have presented in the previous chapter of the form

$$Y_t = f(Y_{t-1}, \dots, Y_{t-p}) + \epsilon_t$$

The conditional expectation and variance of the above process are

$$\begin{aligned} \mathbb{E}[Y_t | Y_{t-1}, \dots, Y_{t-p}] &= f(Y_{t-1}, \dots, Y_{t-p}) \\ \mathbb{V}(Y_t | Y_{t-1}, \dots, Y_{t-p}) &= \mathbb{V}(\epsilon_t | Y_{t-1}, \dots, Y_{t-p}) = \sigma_\epsilon^2 \end{aligned}$$

The above derivation shows that for stationary processes, even though the mean is constant, the conditional mean is not and the $ARMA$ processes have set different function for f in other to feature the observe data characteristics. Unfortunately, for the above processes both the variance and conditional variance is constant and hence unable to produce a process with volatility clustering.

The way to address this issue is to enhance the model with

$$Y_t = f(Y_{t-1}, \dots, Y_{t-p}) + \sigma(Y_{t-1}, \dots, Y_{t-p})\epsilon_t$$

for the modified process the conditional mean has not changed (since ϵ has mean 0) but the conditional variance is driven by

$$\mathbb{V}(Y_t | Y_{t-1}, \dots, Y_{t-p}) = \sigma^2(Y_{t-1}, \dots, Y_{t-p})\sigma_\epsilon^2$$

Finally, observe that the variance of a process Y_t is captured by the square of it, Y_t^2, therefore the idea behind $GARCH$ model is to reproduce the same sort of construction for the squared process Y^2 and hence formulas, ACF plots, ... will be referenced to the square process.

Before we deep dive into $GARCH$ models we start by a couple of useful tools to calibrate this kind of time series, the Unit Root Test and the Partial Autocorrelation Function.

Further reading can be found in Chapter 18 of [1].

2 Unit Root Test

We have seen the difficulty to tell whether a time series is best modeled as stationary or non-stationary. To help decide between both we use a unit root test.

Recall that an $ARMA(p, q)$ process can be written as

$$Y_t - \mu = \phi_1(Y_{t-1} - \mu) + \phi_2(Y_{t-2} - \mu) + \dots + \phi_p(Y_{t-p} - \mu) + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \dots + \theta_q\epsilon_{t-q}$$

It turns out that the condition for Y_t to be stationary is that all roots of the polynomial

$$1 - \phi_1 x - \phi_2 x^2 - \cdots - \phi_p x^p$$

have absolute value greater than one.

There are 2 main unit root test:

- **Dickey-Fuller**: The null hypothesis is that there is a unit root, and the alternative that the process is stationary.
- **KPPS**: The null hypothesis is that the process is sationary, and the alternative that there is a unit root.

The function `auto.arima` in R executes KPPS test recursively until a differentiated series cannot reject the null hypothesis and fits an $ARMA$ to it.

```
rm(list=ls())                                   # Removes all variables from workspace
cat("\f")                                       # Clear console

library("quantmod")                             # Library to get prices from Yahoo
library("forecast")
options("getSymbols.warning4.0"=FALSE)          # set off warnings
cat("\f")                                       # Clear console

getSymbols.FRED('INDPRO',env=globalenv())

fit <-auto.arima(x=INDPRO)
fit
```

```
Series: INDPRO
ARIMA(1,1,1) with drift

Coefficients:
         ar1      ma1    drift
      0.8491  -0.6185   0.0860
s.e.  0.0280   0.0399   0.0275

sigma^2 estimated as 0.1403:  log likelihood=-509.76
AIC=1027.52   AICc=1027.56   BIC=1047.78
```

3 Partial Autocorrelation Function

The Partial Autocorrelation Function (PACF) can be useful to identify the order of a AR process. The k-th partical autocorrelation function $\phi_{k,k}$ for a stationary process Y_t is the correlation of Y_t and Y_{t+k} condition on $Y_{t+1}, \ldots, Y_{t+k-1}$.

For an $AR(p)$ process, it turns out that $\phi_{k,k} = 0$ for $k > p$.

4 $ARCH$

Let us introduce the autoregressive conditional heteroscedasticity process of order 1, $ARCH(1)$

: $ARCH(1)$

Let $\{\epsilon_n\}_n$ be a Gaussian $WN(0, \sigma^2)$. We say that $\{Y_n\}_n$ is an $ARCH(1)$ process if for some constant parameter $\alpha \geq 0$ and $\omega > 0$, the following equation holds:

$$Y_t = \sqrt{\omega + \alpha Y_{t-1}^2}\epsilon_t$$

For Y to be stationary we require $\alpha < 1$. The equation driving the $ARCH(1)$ process is very similar to an $AR(1)$ process, only we have now a multiplicative noise and the equation drives the process Y^2. Since ϵ_t is independent from Y_{t-1} we have

$$\mathbb{E}[Y_t | Y_{t-1}, \ldots] = 0$$

and

$$\sigma_t = \mathbb{V}(Y_t|Y_{t-1},\ldots) = \omega + \alpha Y_{t-1}^2 \,.$$

From the last equation above we can see the possibility that the process exhibits volatility clustering. This is a straight forward example of an uncorrelated process with dependence.

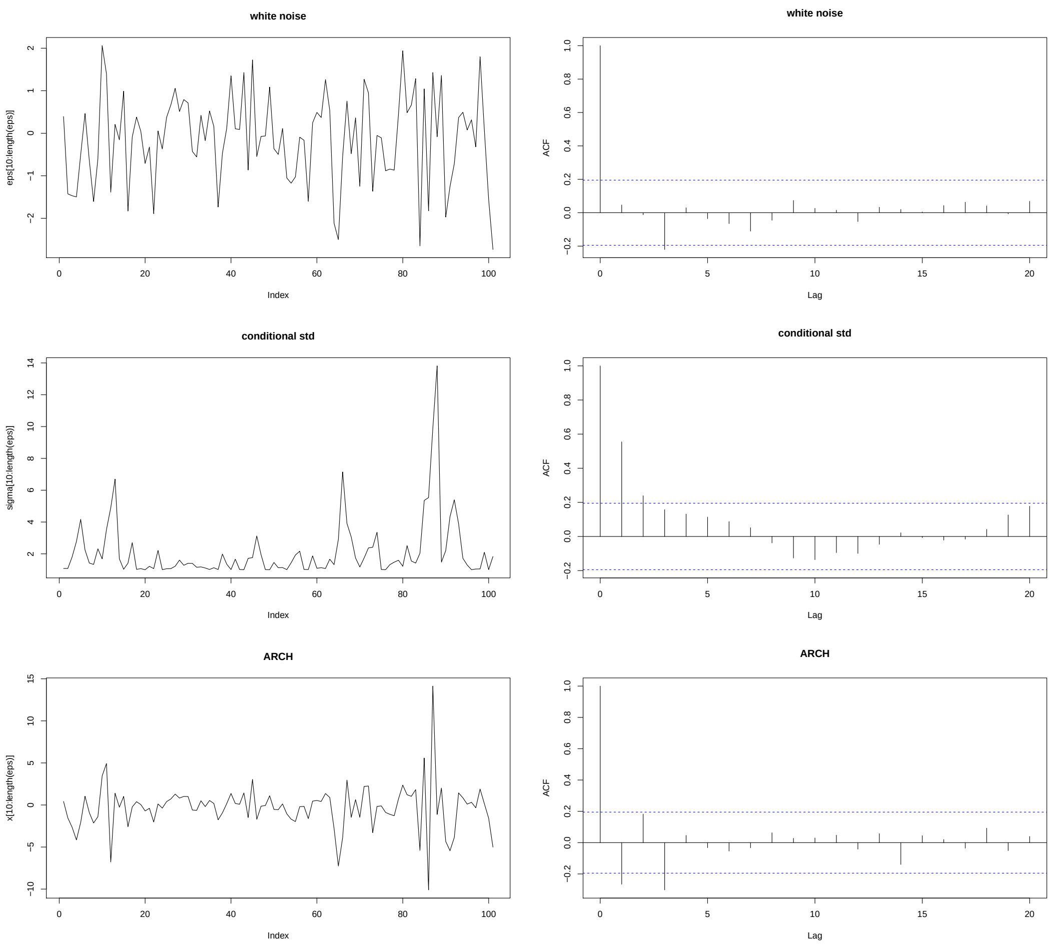
Although the process Y_t is uncorrelated and hence has a trivial autocorrelation function, the process Y_t^2 has a more interesting ACF function, in fact the autocorrelation function is of the form:

$$\rho(h) = \alpha_1^{|h|}$$

for $\alpha < 1$, otherwise the process is nonstationary and has infinite variance.

The following code shows an IDD white noise, the conditional standard deviation of an $ARCH(1)$ process and the original $ARCH(1)$ series:

```
eps=rnorm(110)
x <- vector(mode="numeric", length=110)
sigma <- vector(mode="numeric", length=110)
x[1]<-0
sigma[1]<-0

for (i in 2:110){
  x[i]<-sqrt(1+0.95*(x[i-1])^2)*eps[i]
  sigma[i]<-sqrt(1+0.95*(x[i-1])^2)
}

plot(eps[10:length(eps)],main="white noise",type="l")
acf(eps[10:length(eps)],main="white noise")
plot(sigma[10:length(eps)],main="conditional std",type="l")
acf(sigma[10:length(eps)],main="conditional std")
plot(x[10:length(eps)],main="ARCH",type="l")
acf(x[10:length(eps)],main="ARCH")
```



As you can see the ACF plots of the first and third process are similar, but the processes are completely different. While $AR(1)$ has a nonconstant conditional mean but a constant conditional variance, $ARCH(1)$ process is completely the opposite.

If both the conditional mean and variance depend on the past, we can combine both process:

: $AR(1)/ARCH(1)$

Let $\{a_t\}_t$ be an $ARCH(1)$ process. We say that $\{Y_t\}_t$ is an $AR(1)/ARCH(1)$ process if for some constant parameter μ and ϕ, the following equation holds:

$$Y_t - \mu = \phi(Y_{t-1} - \mu) + a_t$$

The noise process is a weak white noise, and because it is uncorrelated, the ACF plot of an $AR(1)$ looks the same as the ACF plot for an $AR(1)/ARCH(1)$.

The above construction can be generalized to higher orders:

: $ARCH(p)$

Let $\{\epsilon_n\}_n$ be a Gaussian $WN(0, \sigma^2)$. We say that $\{Y_n\}_n$ is an $ARCH(p)$ process if for some constant parameter $\{\alpha_i\}_1^p \geq 0$ and $\omega > 0$, the following equation holds:

$$Y_t = \sqrt{\omega + \sum_{i=1}^{p} \alpha_i Y_{t-i}^2} \epsilon_t$$

Like an $ARCH(1)$ process, the above is uncorrelated and has constant mean (both conditional and unconditional). For the process to be stationary we require $\sum_{i=1}^{p} \alpha_i < 1$.

The $ARCH(p)$ process has nonconstant conditional variance and in fact the squared process has the same ACF as for the $AR(p)$.

5 $GARCH$

The drawback on $ARCH(p)$ process is that the conditional volatiltiy behaves with shocks. To keep the volatility smother but still exhibiting clusters, we introduce the $GARCH(P, Q)$ model, which feeds the past volatility terms in an $ARCH$ process.

: $GARCH(P, Q)$

Let $\{\epsilon_t\}_t$ be a Gaussian $WN(0, \sigma^2)$. We say that $\{Y_t\}_t$ is a $GARCH(P, Q)$ process if for some constant parameters $\{\alpha_i\}_1^P \geq 0$, $\{\beta_i\}_1^Q \geq 0$ and $\omega > 0$, the following equation holds:

$$Y_t = \sigma_t \epsilon_t$$

$$\sigma_t = \sqrt{\omega + \sum_{i=1}^{P} \alpha_i Y_{t-i}^2 + \sum_{j=1}^{Q} \beta_j \sigma_{t-j}^2}$$

For the process to be stationary we require $\sum_{i=1}^{P}\alpha_i + \sum_{j=1}^{Q}\beta_j < 1$.

```
rm(list=ls())                         # Removes all variables from workspace
library(tseries)
cat("\f")

eps=rnorm(110)

x <- vector(mode="numeric", length=110)
sigma <- vector(mode="numeric", length=110)
x[1]<-0
sigma[1]<-0
for (i in 2:110){
  sigma[i]<-sqrt(1+0.8*(x[i-1])^2+0.09*(sigma[i-1])^2)
  x[i]<-sigma[i]*eps[i]
}

plot(eps[10:length(eps)],main="white noise",type="l")
acf(eps[10:length(eps)],main="white noise")
plot(sigma[10:length(eps)],main="conditional std",type="l")
acf(sigma[10:length(eps)],main="conditional std")
plot(x[10:length(eps)],main="GARCH",type="l")
acf(x[10:length(eps)],main="GARCH")
```

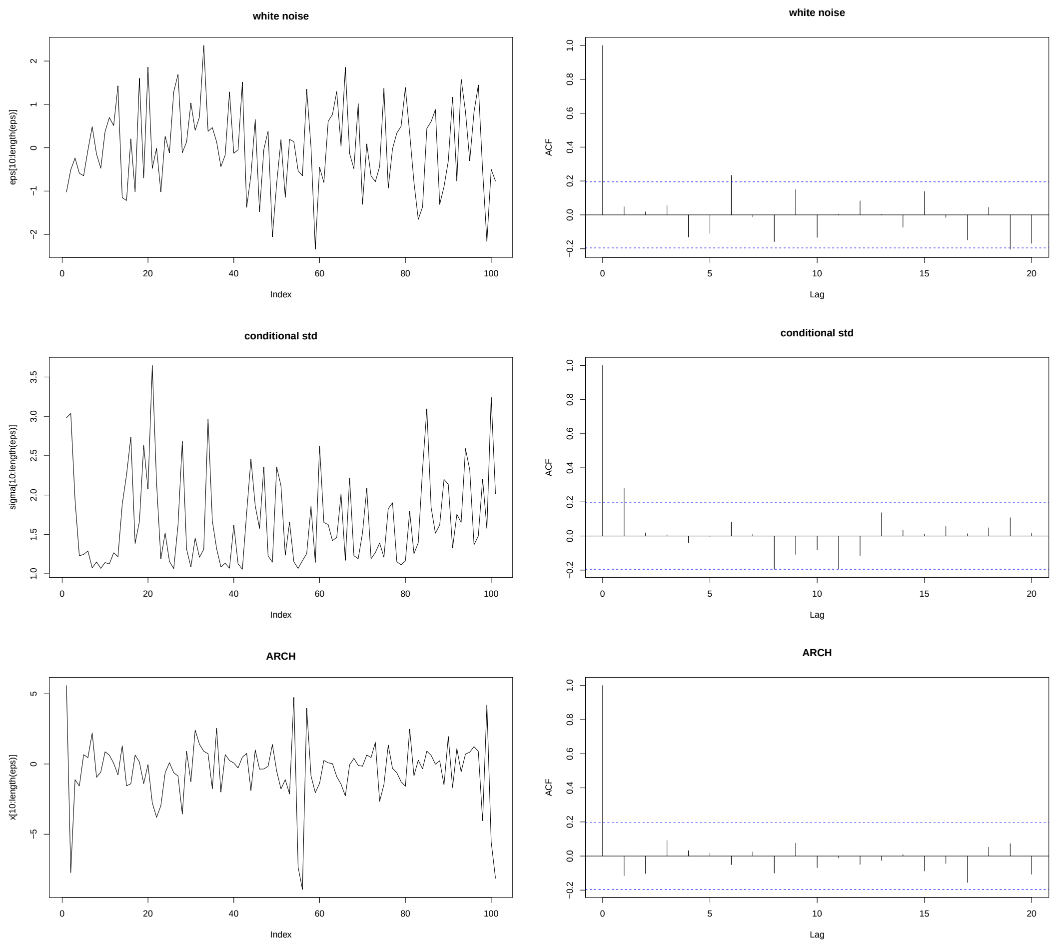


Figure 1. The process Y^2 has similar ACF plot to an $ARMA$ process and exhibits more persistent periods of volatility clustering.

6 $ARIMA(p, d, q)/GARCH(P, Q)$

Now, we just need to combine all the previous process we know to built up a complete and comprehensive family of models for financial data

: $ARIMA(p,d,q)/GARCH(P,Q)$

Let $\{\epsilon_t\}_t$ be a Gaussian $WN(0,\sigma^2)$. We say that $\{Y_t\}_t$ is a $ARIMA(p,d,q)/GARCH(P,Q)$ process if for some constant parameters μ, $\{\phi_i\}_{i=1}^p$, $\{\theta_j\}_{j=1}^q$, $\{\alpha_i\}_1^P \geq 0$, $\{\beta_i\}_1^Q \geq 0$ and $\omega > 0$, the following equation holds:

$$a_t = \sigma_t \epsilon_t$$

$$\sigma_t = \sqrt{\omega + \sum_{i=1}^{P} \alpha_i a_{t-i}^2 + \sum_{j=1}^{Q} \beta_j \sigma_{t-j}^2}$$

$$Y_t = \mu + \sum_{i=1}^{p} \phi_i (Y_{t-i} - \mu) + a_t + \sum_{j=1}^{q} \theta_j a_{t-j}$$

The formula above essentially describes the $ARIMA(p,d,q)/GARCH(P,Q)$ process as an $ARIMA(p,d,q)$ with a $GARCH(P,Q)$ noise.

When fitting $ARIMA(p,d,q)/GARCH(P,Q)$ we usually find out that most software will compute two kind of errors:

- Ordinary residuals, $\hat{a}$, are the difference between Y and its conditional expectation, ie $\mathbb{E}[Y_t|Y_{t-1},\ldots]$.
- Standardized residuals, $\hat{\epsilon}$, are computed by dividing the ordinary residuals by its standard deviation, $\hat{\sigma}$.

Clearly, when fitting $ARIMA(p,d,q)/GARCH(P,Q)$, one should check that neither the standard residuals, $\hat{\epsilon}$, or its squared process, $\hat{\epsilon}^2$, exhibit correlation.

The following code fits an $ARIMA(p,d,q)/GARCH(P,Q)$ to the BMW time series

```
library(fGarch)
data(bmw, package="evir")
fit<-garchFit(formula=~arma(1,0)+garch(1,1),data=bmw,cond.dist="norm")
```

```
Call:
 garchFit(formula = ~arma(1, 0) + garch(1, 1), data = bmw, cond.dist = "norm")

Mean and Variance Equation:
 data ~ arma(1, 0) + garch(1, 1)
<environment: 0x10fd753b8>
 [data = bmw]

Conditional Distribution:
 norm

Coefficient(s):
        mu         ar1        omega       alpha1        beta1
4.0092e-04   9.8596e-02   8.9043e-06   1.0210e-01   8.5944e-01

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     4.009e-04   1.579e-04    2.539   0.0111 *
ar1    9.860e-02   1.431e-02    6.888 5.65e-12 ***
omega  8.904e-06   1.449e-06    6.145 7.97e-10 ***
alpha1 1.021e-01   1.135e-02    8.994  < 2e-16 ***
beta1  8.594e-01   1.581e-02   54.348  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Log Likelihood:
 17757.16    normalized:  2.889222
```

```
Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  11377.99  0
 Shapiro-Wilk Test  R    W      NA        NA
 Ljung-Box Test     R    Q(10)  15.15693  0.1264445
 Ljung-Box Test     R    Q(15)  20.09345  0.168377
 Ljung-Box Test     R    Q(20)  30.54788  0.06144917
 Ljung-Box Test     R^2  Q(10)  5.032717  0.8889818
 Ljung-Box Test     R^2  Q(15)  7.539272  0.9409245
 Ljung-Box Test     R^2  Q(20)  9.277229  0.9794681
```

```
LM Arch Test         R    TR^2    6.03254    0.9144329
```

```
library(fGarch)
data(bmw, package="evir")
fit<-garchFit(formula=~arma(1,1)+garch(1,1),data=bmw,cond.dist="std")
```

```
Standardised Residuals Tests:
                                  Statistic  p-Value
 Jarque-Bera Test    R    Chi^2   13355.05   0
 Shapiro-Wilk Test   R    W       NA         NA
 Ljung-Box Test      R    Q(10)   21.93247   0.01545233
 Ljung-Box Test      R    Q(15)   26.50076   0.03307727
 Ljung-Box Test      R    Q(20)   36.7898    0.01239977
 Ljung-Box Test      R^2  Q(10)   5.828522   0.8294584
 Ljung-Box Test      R^2  Q(15)   8.09067    0.9200962
 Ljung-Box Test      R^2  Q(20)   10.73302   0.9528543
LM Arch Test         R    TR^2    7.009039   0.8570157
```

Notice that all parameters are statistically significant and the large $\hat{\beta}_1$ parameter demonstrate persistent volatility clusters. The final set of hypothesis test are applied to the standardized residuals and its squared. There is also a test to show whether the white noise comes from a Gaussian distribution, but this feature is rejected. One could test the function `garchFit` with distribution to be other than normal to check for a better fit.

7 Rule of thumb for calibrating

The following section aims to give some rules in order to identify the order of the different components of a stationary process. As you have seen sometimes several models fits the same data an in that case we will need to impose some sort of expert criteria.

7.1 Order of differentiation

The first objective in calibrating a model is to identify the order of differentiation. You have seen already the unit root test, but what follows are some further useful observations:

- If the ACF shows meaningful lags of high order (10 or more) the series may need differentiation
- If the first lag of ACF is very negative the series might be overdifferentiated

7.2 $ARIMA$ models

- The lag beyond which the PACF cuts off is the indicative order of the AR term
- The lag beyond which the ACF cuts off is the indicative order of the MA term

7.3 $GARCH$ models

In order to capture the $GARCH$ effect in your model, one would fit the $ARMA$ model and then apply the ACF and PACF functions to the squared residuals. We follow the same rules as before for the regressive and moving average part of the $GARCH$.

7.4 Check residuals

A good calibration is one where residuals are compatible with a white noise, otherwise the proposed model is not capturing all characteristics of the original time series.

8 Example: fitting $ARCH$ models

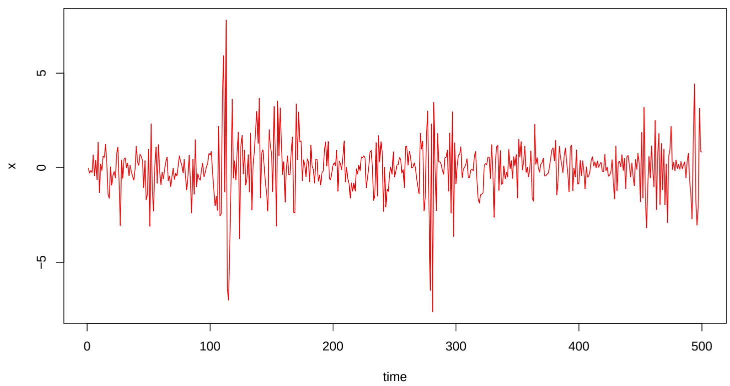
Let us assume you are given a time series and are asked to fit an $ARCH$ model to it. In order to start from a known time series we will simulate the $ARCH$ process and pretend we do not know the parameters in order to try to estimate them.

The procedure describes a step-by-step process not only applicable to $ARCH$ models but in general to any kind of model fitting process.

```
#Fitting and Diagnostic Checking for ARCH Models

library("forecast")
library("rugarch")
library("tseries")
require(fBasics)

rm(list=ls())
cat("\f")

n <- 600
arch2.spec = ugarchspec(variance.model = list(garchOrder=c(2,0)),
                        mean.model = list(armaOrder=c(0,0)),
                        fixed.pars=list(mu = 0, omega=0.25, alpha1=0.6, alpha2=0.35))
arch2.sim <- ugarchpath(arch2.spec, n.sim=n)
x <- drop(arch2.sim@path$seriesSim)[101:600]
time <- 1:length(x)
```

We will assume that the time series x is given and from unknown model. The first step is to plot the time series and analyze its qualitative behavior:

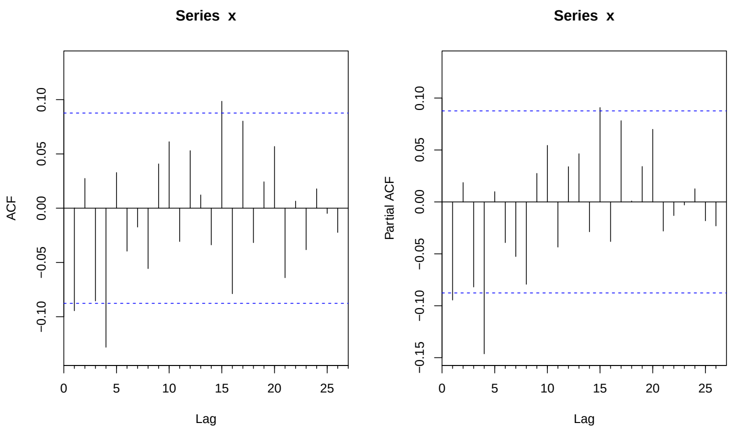
```
plot(time, x, type="l", col=2)
```



The above plot exhibits a time series with a mean-reverting behavior, while we do knot know if it is stationary or not it does not seem to need differentiation, nevertheless one can use a unit root test to check it. We can also appreciate volatility clustering and can anticipate a $GARCH$ component in the model.

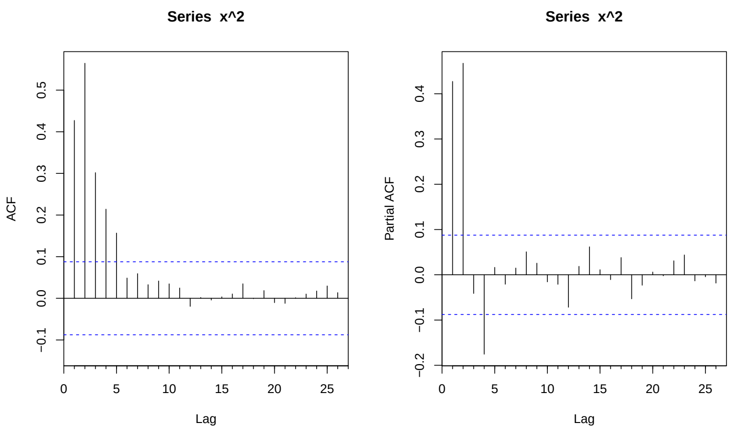
Next step would be plot the ACF and PACF for the original series:

```
par(mfrow=c(1,2))
Acf(x)
Pacf(x)
```



The above figures show a process which is uncorrelated, since the ACF function is trivial, and hence one would guess that the $ARMA$ component is null. Since we have no need to fit the $ARMA$ terms we can proceed to compute the ACF and $PACF$ plots for the square process (in case we have an $ARMA$ component, we first fit this part and apply the following procedure to the residuals):

```
par(mfrow=c(1,2))
Acf(x^2)
Pacf(x^2)
```



We can see a sharp decline in the PACF plot meaning that parameter P could be around 2. On the other hand, the ACF plot shows a slow decay with no sharp cut, this is compatible with a under differentiated series if it where testes on the original time series and not the squared, in this case is showing no need for a Q term in the $GARCH$ component.

After the qualitative guess of the parameters so far, $p = d = q = P = 0$ and $Q \sim 2$ we can use AIC or BIC criteria to choose between a set of similar models around the guesses parameters. One could even try different fitting process

```
fit1 <- garch(x, c(0,1), trace=FALSE) # ARCH(1)
fit2 <- garch(x, c(0,2), trace=FALSE) # ARCH(2)
fit3 <- garch(x, c(0,3), trace=FALSE) # ARCH(3)
fit4 <- garch(x, c(0,4), trace=FALSE) # ARCH(4)
fit5 <- garch(x, c(0,5), trace=FALSE) # ARCH(5)

N <- length(x)
loglik1 <- logLik(fit1)
loglik2 <- logLik(fit2)
loglik3 <- logLik(fit3)
loglik4 <- logLik(fit4)
loglik5 <- logLik(fit5)

```

```
loglik <- c(loglik1, loglik2, loglik3, loglik4, loglik5)
q <- c(1, 2, 3, 4, 5)
k <- q + 1
aicc <- -2 * loglik  + 2 * k * N / (N - k - 1)

print(data.frame(q, loglik, aicc))
```

```
  q    loglik     aicc
1 1 -705.7442 1415.513
2 2 -666.3854 1338.819
3 3 -729.4100 1466.901
4 4 -734.6751 1479.472
5 5 -729.9601 1472.091
```

With `tseries` package one would choose model $ARCH(1)$ or $ARCH(2)$.

```
spec1 <- ugarchspec(variance.model = list(garchOrder=c(1,0)),
                    mean.model = list(armaOrder=c(0,0), include.mean=FALSE))
fit1 <- ugarchfit(spec1, x)
loglik1 <- fit1@fit$LLH

q <- c(1, 2, 3, 4, 5)
loglik <- rep(NA, length(q))

for (i in 1:length(q)) {
    spec <- ugarchspec(variance.model = list(garchOrder=c(q[i],0)),
                       mean.model = list(armaOrder=c(0,0), include.mean=FALSE))
    fit <- ugarchfit(spec, x)
    loglik[i] <- likelihood(fit)
}

k <- q + 1
aicc <- -2 * loglik  + 2 * k * N / (N - k - 1)

print(data.frame(q, loglik, aicc))
```

```
  q    loglik     aicc
1 1 -707.5218 1419.068
2 2 -669.4327 1344.914
3 3 -669.2573 1346.595
4 4 -669.1381 1348.398
5 5 -668.1957 1348.562
```

With `rugarch` package one would choose model $ARCH(2)$ or $ARCH(3)$. Hence, by consensus between both calibration models, we would choose to fit an $ARCH(2)$.

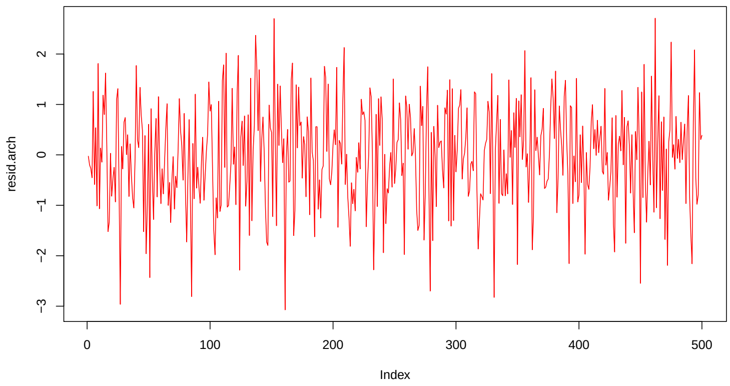
```
fit.tseries <- garch(x, c(0, 2), trace=FALSE)
summary(fit.tseries)
```

```
Call:
garch(x = x, order = c(0, 2), trace = FALSE)

Model:
GARCH(0,2)

Residuals:
     Min       1Q   Median       3Q      Max
-2.96390 -0.69896 -0.06408  0.66584  2.75127

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0   0.23304     0.04186    5.567 2.59e-08 ***
a1   0.57147     0.10637    5.372 7.77e-08 ***
a2   0.37627     0.08302    4.532 5.84e-06 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Diagnostic Tests:
     Jarque Bera Test

data:  Residuals
X-squared = 2.6731, df = 2, p-value = 0.2627


     Box-Ljung test

data:  Squared.Residuals
X-squared = 0.21612, df = 1, p-value = 0.642
```

We first notice that all model parameters are statistically significant. The last step would be to check the residuals and start its analysis by plotting them

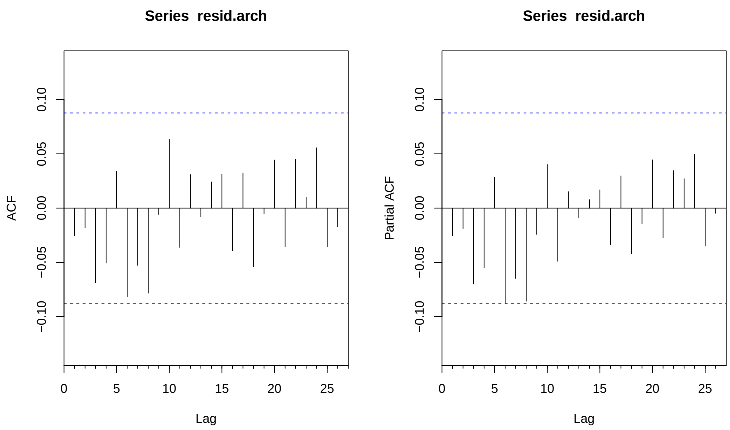
```
# Using tseries package:
resid.tseries <- residuals(fit.tseries)

```

```
# Using rugarch package:
resid.rugarch <- as.numeric(residuals(fit.rugarch, standardize=TRUE))

# Note: resid.tseries is equal to (x / sqrt(h.tseries))
#       resid.rugarch is equal to (x / sqrt(h.rugarch))

par(mfrow=c(1,1))
resid.arch <- resid.rugarch
plot(resid.arch, type="l", col=2)
```



The above plot seems compatible with a white noise, at least it does not exhibit apparent volatility dynamics. We check the ACF and PACF plots for the residuals

```
par(mfrow=c(1,2))
Acf(resid.arch)
Pacf(resid.arch)
```



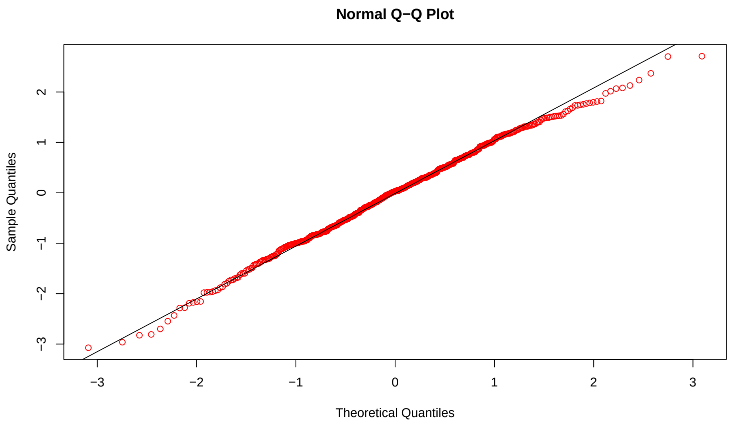
The above plots are now compatible with a white noise.We can also perform some tests

```
Box.test(resid.arch, lag=1, type="Ljung-Box", fitdf=0)
Box.test(resid.arch, lag=12, type="Ljung-Box", fitdf=0)
Box.test(resid.arch, lag=24, type="Ljung-Box", fitdf=0)
Box.test(resid.arch, lag=36, type="Ljung-Box", fitdf=0)
```

```
> Box.test(resid.arch, lag=1, type="Ljung-Box", fitdf=0)

      Box-Ljung test

data:  resid.arch
X-squared = 0.0036762, df = 1, p-value = 0.9517

> Box.test(resid.arch, lag=12, type="Ljung-Box", fitdf=0)

      Box-Ljung test

data:  resid.arch
X-squared = 4.7582, df = 12, p-value = 0.9656

> Box.test(resid.arch, lag=24, type="Ljung-Box", fitdf=0)

      Box-Ljung test

data:  resid.arch
X-squared = 13.96, df = 24, p-value = 0.9476

> Box.test(resid.arch, lag=36, type="Ljung-Box", fitdf=0)

      Box-Ljung test

data:  resid.arch
X-squared = 23.563, df = 36, p-value = 0.945
```

Finally we can even check whether the white noise of the model has a particular distribution, for example a normal distribution

```
qqnorm(resid.arch, col=2)
qqline(resid.arch, col=1)
```




```
normalTest(resid.arch,method="jb")
```

```
Title:
 Jarque - Bera Normalality Test

Test Results:
  STATISTIC:
    X-squared: 2.5738
  P VALUE:
    Asymptotic p Value: 0.2761

Description:
 Wed Sep 28 22:33:13 2016 by user:
```

9 Recap

In this chapter we have completed our model for stationary time series by including a family of models that are able to capture volatility clustering.

We also introduce the process for model calibration which can be summarize as

1. **Plot the Data**: Identify unusual data observations
2. **Transform and Differentiate**: Transform your data or differentiate it to obtain a stationary time series
   (a) **Unit Root Test**: Use a test to ensure that the series is stationary
3. **Calibrate** $ARMA$: Plot ACF and PACF functions for your transformed series and estimate the order p and q
   (a) **Chose model**: Perform a test to show how many parameters from the ACF and PACF are relevant
   (b) **Chose model**: Use the AIC or BIC criteria to get a final estimate for p and q
4. **Estimate coefficients of the** $ARMA$: Make sure that all coefficients are statistically meaningful
5. **Plot the residuals and squared residuals**: If the residuals do not seem like a white noise we might have some $GARCH$ effect on the model
   (a) **Plot ACF and PACF for the square residuals**: Plot ACF and PACF functions and estimate the order P and Q
   (b) **Estimate coefficients of the** $ARMA/GARCH$: Make sure that all coefficients are statistically meaningful
6. **Plot the residuals**: Show that the residuals are compatible with a white noise

References

[1] David Ruppert (2010) *Statistics and Data Analysis for Financial Engineering*, Springer Texts in Statistics.