Introduction to Financial Engineering (W3)

Time Series: Macroeconomic Studies

1 Context

The objective of this Chapter is to introduce the applications of ARMA and GARCH models in finance. The most common application of such models is to produce predictions of macroeconomic time series such as GDP, inflation, unemployment rates, house prices, ... As you may guess, such data is very important for long term forecast in financial entities. Such kind of entities usually have dedicated department of macroeconomic studies to monitor the evolution of such variables and estimate future behavior. For instance, unemployment rates might be an indicator that delinquencies in loan payments may rise in the near future, and hence the credit entity needs to be prepare for that event. Those macroeconomic department employ a mixture of economists and statisticians/mathematicians to produce their models, and recently data science profiles has been increasingly demanded.

The aim of this Chapter is to present a quick overview of the mathematical models and use built-in packages in R to calibrate real data. We do not aim to provide the full derivation of the models used or elaborate on the best estimates for the parameters of a particular model, on the other hand we will present this models as plausible models to be used in such macroeconomic time series and use what we have available in R to built them, calibrate them and forecast them.

Further reading can be found in Chapter 9 of [1].

2 Stochastic Processes and Stationary Series

A time series is a sequence of observations in chronological order, for example daily stock returns. A **Stochastic Process** is a sequence of random variables, and hence a time series can be viewed as a realization (sample) from a stochastic process.

Whenever you are faced with the task of building up a model for real data you will face the problem of determining how many parameters are needed to fit the model properly. Over fitting a model is a serious problem since you may create a model with very limited capacity to predict, and on the other hand using to few variables may give rise to a useless model. One need to use as many parameters as needed and not more, and if possible, give a context meaning to all of them. This means that the parameters used have a practical/qualitative interpretation with respect to the data we are calibrating. This is known as parsimony of the model.

One of the most useful methods for obtaining parsimony in time series is to assume stationarity. This property is observed in a time series when the fluctuations appear random but often with the same type of stochastic properties along the sample. For example, stock prices appear random from one period to the other, but often statistics as the mean and variance of its returns seem to behave less randomly.

A stationary process is a probability model with time invariant behavior, in other words all aspects of its behavior are invariant under a time shift. The time invariant property will allow us to build up models with less parameters.

The following example depicts stock price prices and returns for Nintendo share:

: Returns

The goal of investing is, of course, to make profit. The revenue depends upon both the change in prices and the amounts of asset being held. Investors are interested in revenues that are high relative to the size of the initial investment. Returns measure this because returns on an asset are changes in price expressed as a fraction of the initial price. Returns are scale-free, meaning they do not depend on units.

- **Net Returns**: Let P_t be the price of an asset at time t, the net return is

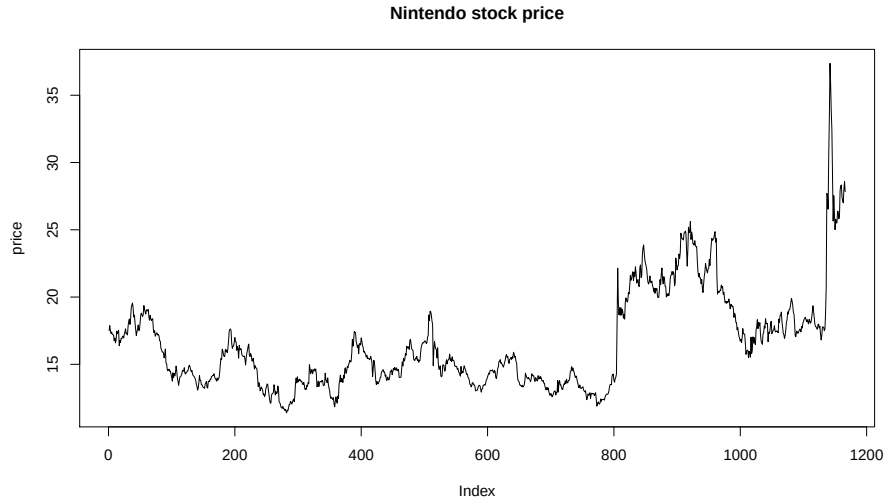
$$R_t = \frac{P_t}{P_{t-1}} - 1 = \frac{P_t - P_{t-1}}{P_{t-1}} .$$

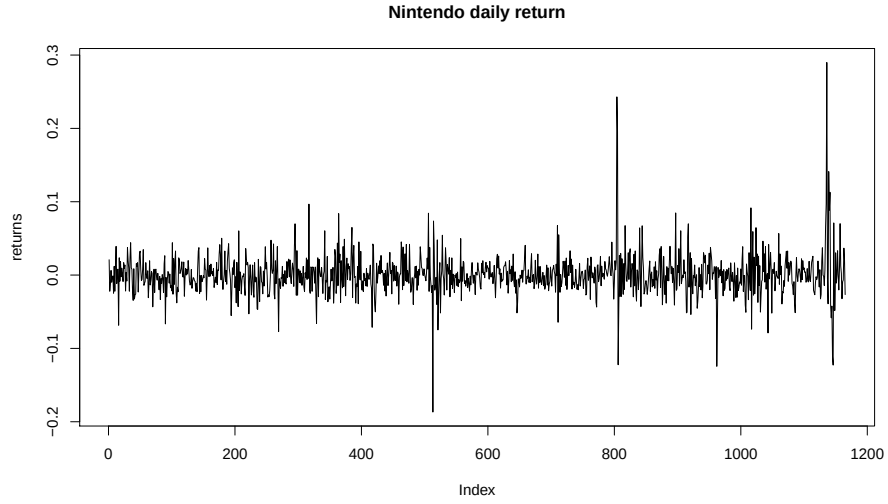
  Therefore the revenue of an investment is simply the initial investment times the net return.
- **Log Returns**: Called continuously compounded returns are defined as:

$$r_t = \log(1 + R_t) = \log\left(\frac{P_t}{P_{t-1}}\right) = p_t - p_{t-1}$$

  where $p_t = \log(P_t)$ is called the log price.

```
# Stock evolution of Nintendo

rm(list=ls())                              # Removes all variables from workspace
library("quantmod")                        # Library to get prices from Yahoo
options("getSymbols.warning4.0"=FALSE)     # set off warnings
cat("\f")                                  # Clear console


getSymbols('NTDOY',scr='yahoo',from="2012-01-01",to="2016-08-20")
nint_close=NTDOY$NTDOY.Close

price=as.numeric(nint_close)
returns=log(price[2:length(price)]/price[1:length(price)-1])


pdf("nintendo01.pdf",width=10,height=6)
plot(price, type="l",main="Nintendo stock price")
dev.off()

pdf("nintendo02.pdf",width=10,height=6)
plot(returns,type="l", main="Nintendo daily return")
dev.off()
```





Let us properly define this time-invariant property

: Strong Stationary Process

Let $\{Y_n\}_n$ be a stochastic process and $F(Y_{n_1}, Y_{n_2}, \ldots, Y_{n_k})$ be the cumulative distribution function of $\{Y_n\}_n$ at times $n_1, n_2, \ldots, n_k$. Then $\{Y_n\}_n$ is said to be strongly stationary if for any n and m

$$F(Y_{n_1}, Y_{n_2}, \ldots, Y_{n_k}) = F(Y_{n_1+m}, Y_{n_2+m}, \ldots, Y_{n_k+m})$$

The above property is indeed very restrictive and often we are comfortable enough to state a weak version

: Weak Stationary Process

Let $\{Y_n\}_n$ be a stochastic process, then $\{Y_n\}_n$ is said to be weak stationary if

$$\begin{aligned} \mathbb{E}[Y_i] &= \mu \\ \mathbb{V}[Y_i] &= \sigma^2 \\ \text{Corr}(Y_i, Y_j) &= \rho(|i-j|) \end{aligned}$$

for all i and j and some function ρ.

The function $\rho(h)$ is called the autocorrelation function. Another related and relevant function would be the autocavariance function

$$\text{Cov}(Y_i, Y_j) = \gamma(|i-j|)$$

which is a simple transformation of the autocorrelation function as $\gamma(h) = \sigma^2\rho(h)$.

A stationary time series should show oscillations around the same fixed level, called mean-reversion. If the series wonders around without repeatably returning to the same fixed level, then the series should not be modeled as stationary. As seen before, many financial time series are not stationary, but often transformations on them are, for instance transforming prices in log-returns on Nintendo stock.

: White Noise

It is the simplest example of stationary process. The sequence $\{Y_n\}_n$ is a weak white noise, $WN(\mu, \sigma^2)$, if

$$\begin{aligned} \mathbb{E}[Y_i] &= \mu \\ \mathbb{V}[Y_i] &= \sigma^2 \\ \text{Corr}(Y_i, Y_j) &= \delta_{i,j} \end{aligned}$$

By definition the autocorrelation function is:

$$\rho(h) = \begin{cases} \rho(0) = 1 \\ \rho(h) = 0 \text{ for } h \neq 0 \end{cases}$$

Note that an i.i.d. white noise (independent and identically distributed) is strongly stationary, since the joint multivariate distribution is the product of their marginal distributions which are all the same.

3 Estimating parameters

Suppose we observe $Y_1, Y_2, \ldots, Y_n$ from a stationary process. Sample mean, $\widehat{\mu} = \overline{Y}$, and sample variance, $\widehat{\sigma}^2 = s^2$, can be used to estimate mean and variance. To estimate the autocorrelation function we can use

$$\widehat{\gamma}(h) = \frac{\sum_{j=1}^{n-h}(Y_{j+h} - \overline{Y})(Y_j - \overline{Y})}{n}$$

and from here we can estimate the autocorrelation function as

$$\widehat{\rho}(h) = \frac{\widehat{\gamma}(h)}{\widehat{\gamma}(0)}$$

ACF plots and the **Ljung-Box test** are built-in tools in R to plot the autocorrelation function and test it. The ACF plot is pictured with test bounds, which are used to test whether a particular autocorrelation coefficient

is null. The usual level of confidence is 0.05 and hence we can expect 1 out of 20 sample autocorrelation coefficients out of the bandwidth simply by chance. The alternative is to use a simultaneous test where the null hypothesis is

$$H_0 := \rho(1) = \rho(2) = \cdots = \rho(k)$$

for some k. The above test is called Ljung-Box test.

```
...

pdf("nintendo01_ACF.pdf",width=10,height=6)
acf(price, main="ACF Nintendo stock price")
dev.off()

pdf("nintendo02_ACF.pdf",width=10,height=6)
acf(returns, main="ACF Nintendo daily return")
dev.off()

Box.test (returns, lag = 20, type = "Ljung")
```

```
        Box-Ljung test

data:  returns
X-squared = 29.1, df = 20, p-value = 0.08581
```

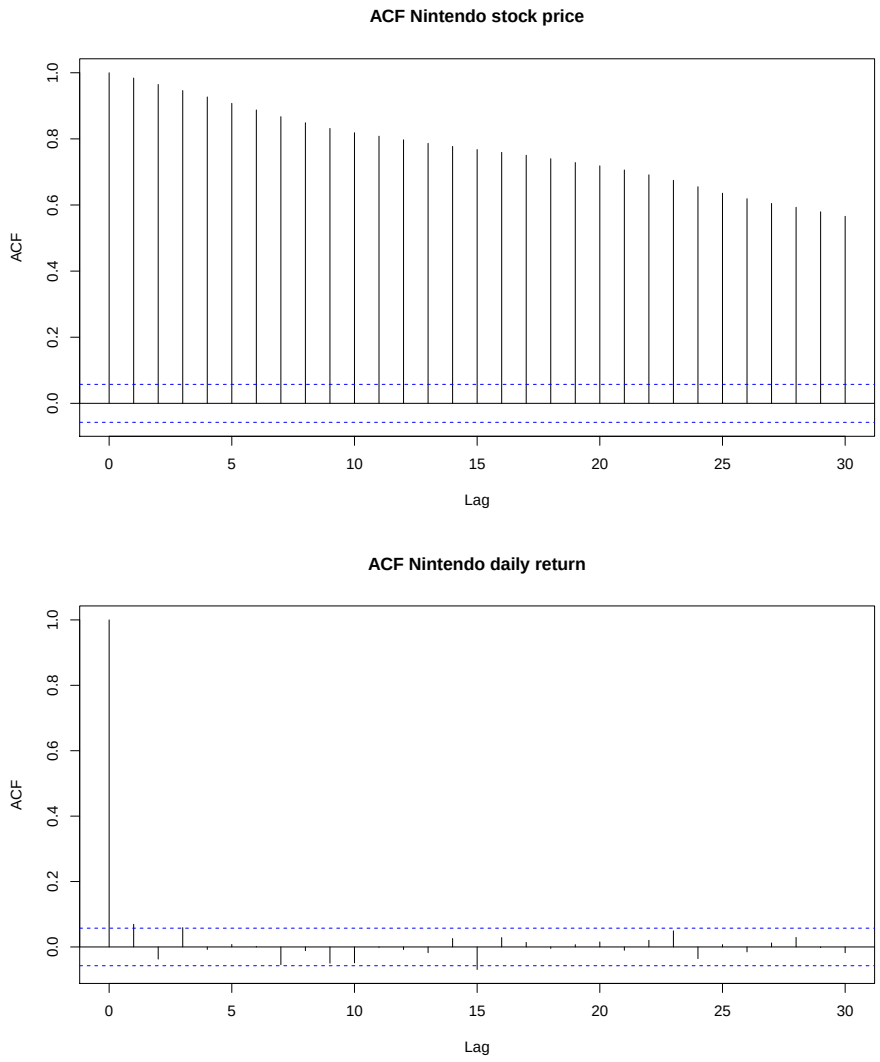


Figure 1. The first ACF plot shows a slow decay to zero which is a sign of long dependence memory or non-stationary process. Quick decay towards zero is consistent with stationary and matches the observation made in the previous pictures.

4 $AR(p)$ Autoregressive Processes

Although stationary processes are somehow parsimonious with parameters, they still have an infinite number of them, namely $\{\rho(i)\}_i$. We need is a class of stationary process with a finite number of parameters but a wide range of behaviors so they can be easily calibrate to real data and are meaningful models. These are $ARIMA$ models. In order to built up the $ARIMA$ models we will first look at the smaller family of AR processes, which are known by autoregressive models.

: $AR(1)$

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

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

Parameter μ is the mean of the process and term $\phi(Y_{t-1} - \mu)$ represents the memory, in particular the constant ϕ determines the velocity of the mean-reversion of the process (the greater in absolute value the slowest the process reverts to its mean). Finally $\{\epsilon_n\}_n$, is interpreted as information shocks in finance time series. Later in the course we will show that in some price formation theories the news and information shocks are the drivers of price evolution. In this case new information cannot be anticipated so effects of today's information should be independent from yesterday's news and hence the use of a WN.

Observe that for Y to be a weakly stationary process we need $|\phi| < 1$. To derive this observation, let us name σ_Y the variance of Y_t, which by independence, follows

$$\sigma_Y^2 = \phi^2\sigma_Y^2 + \sigma_\epsilon^2 \, .$$

We can rewrite the definition of $AR(1)$ as

$$\begin{aligned} Y_t &= (1-\phi)\mu + \phi Y_{t-1} + \epsilon_t \\ &= (1-\phi)\mu + \phi((1-\phi)\mu + \phi Y_{t-2} + \epsilon_{t-1}) + \epsilon_t \\ &= \mu + \phi^2\mu + \epsilon_t + \phi\epsilon_{t-1} + \phi Y_{t-2} \\ &= \mu + \epsilon_t + \phi\epsilon_{t-1} + \phi^2\epsilon_{t-2} + \cdots = \mu + \sum_{h=0}^{\infty} \phi^h \epsilon_{t-h} \end{aligned}$$

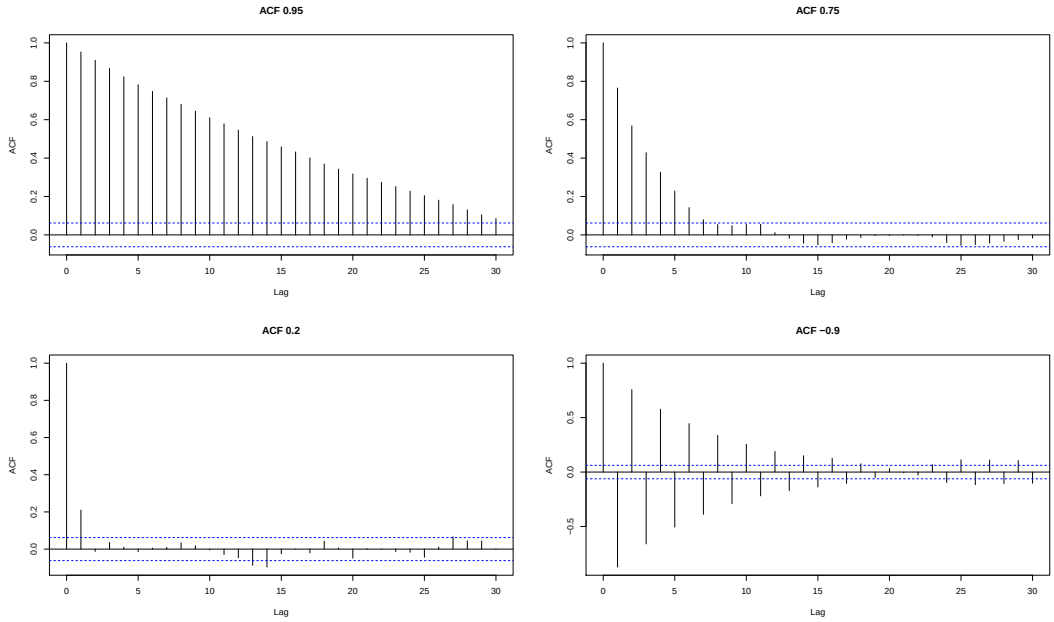
which is a way of looking at an $AR(1)$ as a weighted average of all past white noise shocks (or the price of a financial asset as the weighted average of all past related information).

For a stationary $AR(1)$ process, that is $|\phi| < 1$, we can easily derive the following

$$\begin{aligned} \mathbb{E}[Y_t] &= \mu \\ \gamma(0) &= \mathbb{V}(Y_i) = \mathbb{V}\left(\sum_{h=0}^{\infty} \phi^h \epsilon_{t-h}\right) = \frac{\sigma_\epsilon^2}{1-\phi^2} \\ \gamma(h) &= \text{Cov}\left(\sum_{i=0}^{\infty} \phi^i \epsilon_{t-i}, \sum_{j=0}^{\infty} \phi^j \epsilon_{t+h-j}\right) = \frac{\sigma_\epsilon^2}{1-\phi^2}\phi^{|h|} \\ \rho(h) &= \phi^{|h|} \end{aligned}$$

The ACF plot for an $AR(1)$ depends upon only one parameter, i.e. ϕ. This is a remarkable level of parsimony but with high price. Only a limited amount of different ACF plots are achieved with an $AR(1)$.

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

y.95 <- arima.sim(model=list(ar=.95), n=1000)

pdf("AR_ACF_1.pdf",width=10,height=6)
acf(y.95, main="ACF 0.95")
dev.off()

y.75 <- arima.sim(model=list(ar=.75), n=1000)

pdf("AR_ACF_2.pdf",width=10,height=6)
acf(y.75, main="ACF 0.75")
dev.off()

y.2 <- arima.sim(model=list(ar=.2), n=1000)

pdf("AR_ACF_3.pdf",width=10,height=6)
acf(y.2, main="ACF 0.2")
dev.off()

y.99 <- arima.sim(model=list(ar=-.9), n=1000)

pdf("AR_ACF_4.pdf",width=10,height=6)
acf(y.99, main="ACF -0.9")
dev.off()
```



If $|\phi| \geq 1$ the $AR(1)$ is not a stationary process, and the variance is not constant. There is a special case when $\phi = 1$ named Random Walk:

: Random Walk

Let $\{\epsilon_n\}_n$ be a $WN(0, \sigma^2)$. We say that $\{Y_n\}_n$ is an Random Walk if the following equation holds:

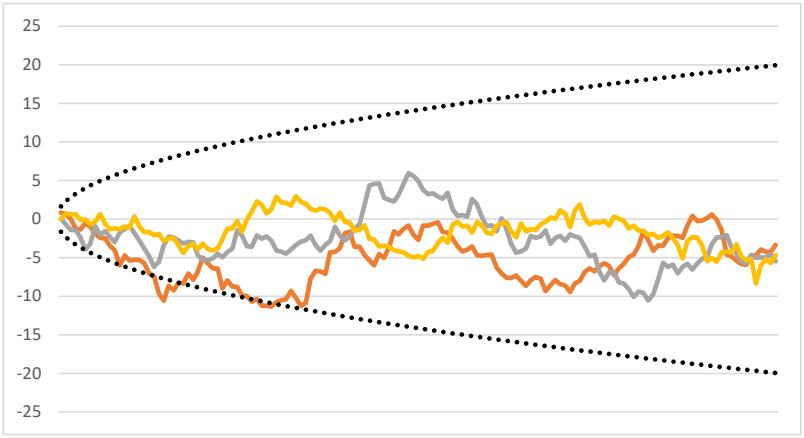
$$Y_t = Y_{t-1} + \epsilon_t$$

One can easily rewrite the above as:

$$Y_t = Y_0 + \epsilon_1 + \epsilon_2 + \cdots + \epsilon_t$$

And hence the process has a mean reversion behavior but it also has long excursions from the mean

$$\begin{aligned}\mathbb{E}[Y_t|Y_0] &= Y_0\\ \mathbb{V}(Y_t) &= t\sigma_\epsilon^2\end{aligned}$$



When $|\phi| > 1$ the process has explosive behavior.

R has a built-in package to estimate AR processes and other time series. The method uses the maximum likelihood estimation with lest-square estimates as the starting point. In the process of calibrating the model, one will usually analyze the autocorrelation function of the residuals, as any positive correlation in the residuals is an evidence against the $AR(1)$.

The following example fits a time series of BMW prices.

```
rm(list=ls())                                  # Removes all variables from workspace
cat("\f")                                      # Clear console
library("evir")
data(bmw,package="evir")                       # Load BMW log prices

pdf("BMW_1.pdf",width=10,height=6)             # ACF for log prices
acf(bmw, main="BMW log reutrns")
dev.off()
Box.test (bmw, lag = 5, type = "Ljung")        # Evidence of not being a white noise
fit<-arima(x=bmw,order=c(1,0,0))               # Fit of AR(1)
fit                                            # Showing the parameters
                                               # Final analysis of the residuals
pdf("BMW_2.pdf",width=10,height=6)
acf(fit$resid, main="BMW AR residuals")
Box.test (fit$resid, lag = 5, type = "Ljung",fitdf = 1)
dev.off()
```

```
        Box-Ljung test
data:  bmw
X-squared = 44.987, df = 5, p-value = 1.46e-08
```

```
Call:
arima(x = bmw, order = c(1, 0, 0))
Coefficients:
         ar1  intercept
      0.0811       3e-04
s.e.  0.0127       2e-04

sigma^2 estimated as 0.0002163:  log likelihood = 17212.34,  aic = -34418.68
```

```
        Box-Ljung test
data:  fit$resid
X-squared = 6.8669, df = 4, p-value = 0.1431
```

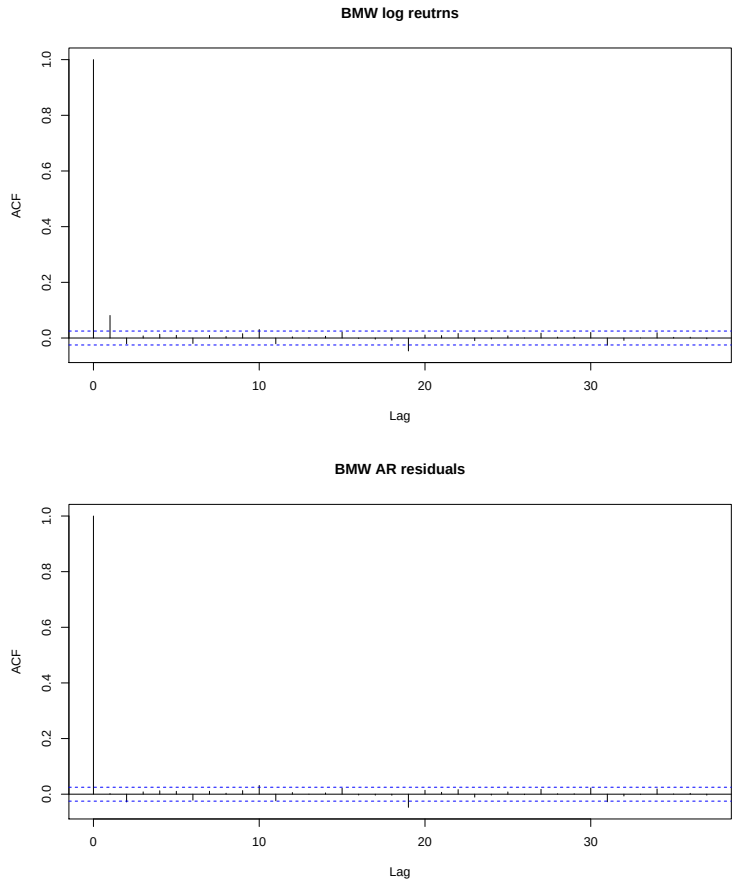


Figure 2. The example fits the following parameters $\hat{\phi} = 0.0811$ and $\hat{\mu} = 0.00034$. The first test shows a p-value below threshold meaning the original process is not a White Noise, second output shows calibration and significance of coefficients, while the last test perform over the residuals shows compatibility with a White Noise. Overall the analysis shows a decent calibration of the data to an $AR(1)$ model.

We have seen that the ACF for an $AR(1)$ process decays geometrically to zero and alternate signs if $\phi < 0$, but this is a limited range of behavior for the ACF function. To get an extended class of autocorrelation function we can generalize $AR(1)$ processes to regress to older values of the time series:

: $AR(p)$

Let $\{\epsilon_n\}_n$ be a $WN(0, \sigma^2)$. We say that $\{Y_n\}_n$ is an $AR(p)$ process if for some constant parameter μ and $\{\phi_i\}_{i=1}^p$, the following equation holds:

$$Y_t - \mu = \phi_1(Y_{t-1} - \mu) + \phi_2(Y_{t-2} - \mu) + \cdots + \phi_p(Y_{t-p} - \mu) + \epsilon_t$$

The above formula can be rewritten as

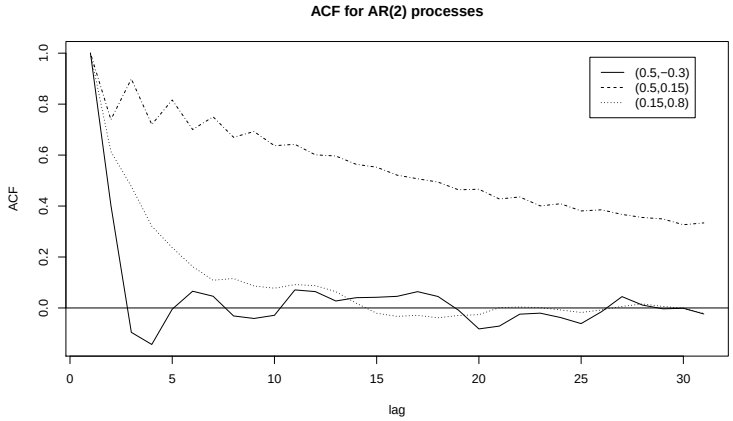
$$Y_t = \beta_0 + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \epsilon_t,$$

where $\beta_0 = [1 - (\phi_1 + \cdots + \phi_p)]\mu$. It can be showed that if Y is a stationary process, then

$$\frac{\beta_0}{\mu} > 0$$

Even though the formulas get more involved, most of the concepts we have discussed for $AR(1)$ generalize for $AR(p)$. Let us plot some ACF functions for $AR(2)$:

```
y <- arima.sim(model=list(order=c(2,0,0), ar=c(0.5,-0.3)), n=1000)
z <- arima.sim(model=list(order=c(2,0,0), ar=c(0.5,0.15)), n=1000)
v <- arima.sim(model=list(order=c(2,0,0), ar=c(0.15,0.8)), n=1000)

yy<-acf(y)
zz<-acf(z)
vv<-acf(v)

pdf("AR2_ACF.pdf",width=10,height=6)
plot(yy$acf, type="l",main="ACF for AR(2) processes",xlab="lag",ylab="ACF")
lines(zz$acf,lty=3)
lines(vv$acf,lty=4)
legend("topright", inset=.05, cex = 1, c("(0.5,-0.3)","(0.5,0.15)","(0.15,0.8)"), horiz=FALSE, lty=c(1,2,3))
abline(h=0,lty=1,lwd=1)
dev.off()
```



5 $MA(q)$ Moving Average Processes

There is a potential need for large values of p when fitting an $AR(p)$ model, the remedy is to introduce a moving average component to construct an $ARMA$ process.

Before we introduce $ARMA$ process, let's look at the MA processes in more detail.

: $MA(q)$

Let $\{\epsilon_n\}_n$ be a $WN(0,\sigma^2)$. We say that $\{Y_n\}_n$ is an $MA(q)$ process if for some constant parameter μ and $\{\theta_j\}_{j=1}^q$, the following equation holds:

$$Y_t = \mu + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q}$$

For the $MA(1)$ process defined as

$$Y_t = \mu + \epsilon_t + \theta_1\epsilon_{t-1}$$

we can easily derive

$$\begin{aligned}
\mathbb{E}[Y_t] &= \mu \\
\mathbb{V}(Y_t) &= \sigma_\epsilon^2(1+\theta_1^2) \\
\gamma(1) &= \theta_1\sigma_\epsilon^2 \\
\gamma(h) &= 0 \text{ for all } |h|>1
\end{aligned}$$

In a general framework, it can be showed that for $MA(q)$ process, $\gamma(h) = 0$ for $|h| > q$.

6 $ARMA$ Processes

Stationary time series with complex autocorrelation behavior are often more parsimoniously modeled by mixed autoregressive and moving average ($ARMA$) process than a pure AR or MA process

: $ARMA(p,q)$

Let $\{\epsilon_n\}_n$ be a $WN(0,\sigma^2)$. We say that $\{Y_n\}_n$ is an $ARMA(p,q)$ process if for some constant parameter μ and $\{\phi_i\}_{i=1}^p$ and $\{\theta_j\}_{j=1}^q$, the following equation holds:

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

The $ARMA(1,1)$ is used in practice and yet simply enough to study theoretically, among other expressions one can derive:

$$\rho(k) = \begin{cases} \frac{(1+\theta\phi)(\theta+\phi)}{1+\theta^2+2\theta\phi} \text{ for } k=1 \\ \phi\rho(k-1) \text{ for } k \geq 2 \end{cases}$$

In particular, one observes that after one lag, the autocorrelation function decays as an $AR(1)$. This means that the $ARMA(1,1)$ features the same characteristics as a $AR(1)$ and a $MA(1)$.

The following code fits several $ARMA$ processes to the same time series and extracts its AIC and BIC criteria. AIC (Akaike information criterion) and BIC (Bayesian information criterion) are two similar criteria for model selection. They both balance how good the model fit the data and how many parameters are in the model. Generally, it is possible to increase likelihood by adding parameters, but this might generate over fitting. In both cases the lower the criteria is the better the model. Bear in mind that even when using AIC or BIC criteria to select a particular model among a finite set, one needs to evaluate the behavior of such model by means of AFC plots and further analyses that we will introduce in the next sections.

Let us recap the example on the BMW price time series and fit a set of $ARMA$ models to it

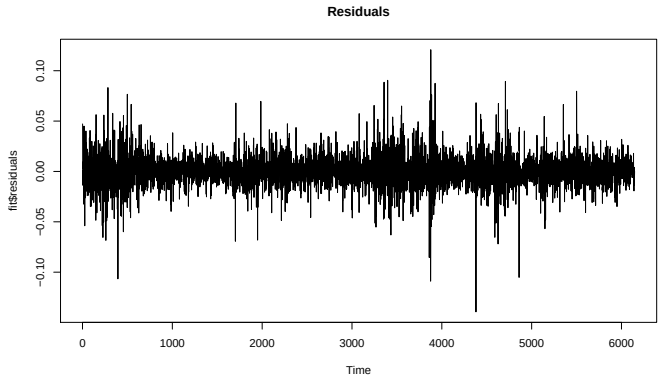
```
rm(list=ls())                                   # Removes all variables from workspace
cat("\f")                                       # Clear console

library("evir")
data(bmw, package="evir")                       # Load BMW log prices

aic_bic <-array(dim=c(9,4))
k=1
for (i in 0:2) {
  for (j in 0:2){
    fit <-arima(x=bmw, order=c(i,0,j))
    aic_bic[k,1]=i
    aic_bic[k,2]=j
    aic_bic[k,3]=AIC(fit)+34500
    aic_bic[k,4]=AIC(fit,k=log(length(bmw)))+34500
    k=k+1
  }
}
colnames(aic_bic) <- c("p","q","AIC","BIC")
aic_bic
```

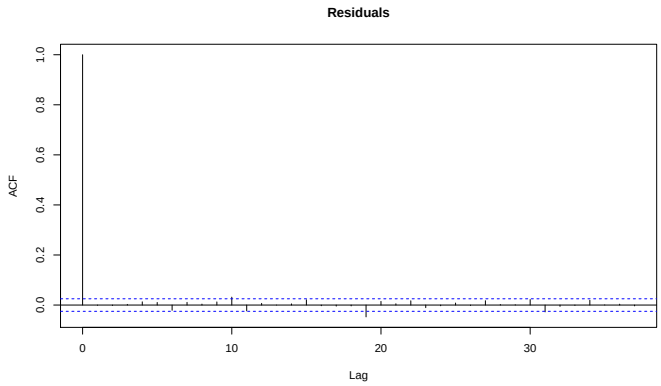
```
  aic_bic
       p q       AIC       BIC
 [1,] 0 0 119.82867 133.27578
 [2,] 0 1  79.21393  99.38461
 [3,] 0 2  78.36773 105.26195
 [4,] 1 0  81.31575 101.48642
 [5,] 1 1  78.25941 105.15364
 [6,] 1 2  80.44557 114.06336
 [7,] 2 0  78.73415 105.62838
 [8,] 2 1  80.58248 114.20026
 [9,] 2 2  82.22671 122.56805
```

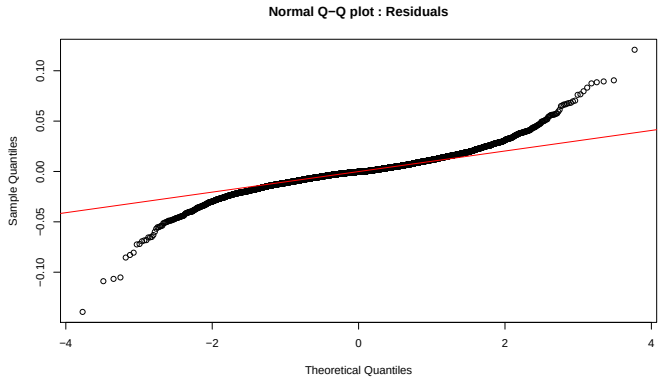
$ARMA(0,1)$ and $ARMA(1,1)$ seem the best fits to the model. Let us check the $ARMA(1,1)$ model:

```
rm(list=ls())                                   # Removes all variables from workspace
```

```
cat("\f")                                   # Clear console

library("evir")
data(bmw, package="evir")                   # Load BMW log prices

fit <- arima(x=bmw, order=c(1,0,1))

pdf("BMW_res_plot.pdf", width=10, height=6)
plot(fit$residuals, type="l", main="Residuals")
dev.off()

pdf("BMW_res_acf.pdf", width=10, height=6)
acf(fit$residuals, main="Residuals")
dev.off()

pdf("BMW_res_qqplot.pdf", width=10, height=6)
qqnorm(fit$residuals, main="Normal Q-Q plot : Residuals")
qqline(fit$residuals, col=2)
dev.off()
```







Although the above analysis shows a better fit than a $AR(1)$ we can see that the QQplot of the residuals exihibit havy tails and there are signs of clustering on the residuals. This shows us that $ARMA$ might not be an enough complex model to fit the time series. In fact clustering effects cannot be produced in $ARMA$ models and we

need to introduce volatility modeling in the next chapter.

7 $ARIMA$ Processes

Often the first or second difference of a nonstationary time series are stationary. Autoregressive Integrated Moving Average (ARIMA) processes are introduced easily with the help of the following to operators:

: Backward Operator

It is defined for $k \geq 0$ by

$$B^k Y_t = Y_{t-k}$$

: Differentiating Operator

It is defined as $\Delta = 1 - B$, hence

$$\begin{aligned} \Delta Y_t &= (1-B)Y_t = Y_t - Y_{t-1} \\ \Delta^k Y_t &= (1-B)^k Y_t = \sum_{i=1}^{k} \binom{k}{i} Y_{t-i} \end{aligned}$$

Then

: $ARIMA(p,d,q)$

We say that $\{Y_n\}_n$ is an $ARIMA(p,d,q)$ process if $\Delta^d Y_t$ is a $ARMA(p,q)$.

By definition we set $ARIMA(p,0,q)$ as $ARMA(p,q)$. For instance, if log-returns are $ARMA(p,q)$, then log-prices are $ARIMA(p,1,q)$. Note that if $ARIMA(p,d,q)$ is stationary, then $d = 0$.

The inverse of differentiation is integration of a process. The integration of Y_t is the process X_t defined by

$$X_t = X_0 + Y_{t_0} + Y_{t_1} + \cdots + Y_t$$

indeed $\Delta X_t = Y_t$. Qualitative speaking, integrating smooths the process and differentiating make it more rough. In the following example you can appreciate how an integrating process looks like:

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

y <- arima.sim(model=list(order=c(1,0,0), ar=c(0.5)), n=1000)

pdf("AR1.pdf",width=10,height=6)
plot(y, type="l",main="AR(1)")
dev.off()

pdf("AR1_I1.pdf",width=10,height=6)
plot(cumsum(y), type="l",main="AR(1) I1")
dev.off()

pdf("AR1_I2.pdf",width=10,height=6)
plot(cumsum(cumsum(y)), type="l",main="AR(1) I2")
dev.off()
```

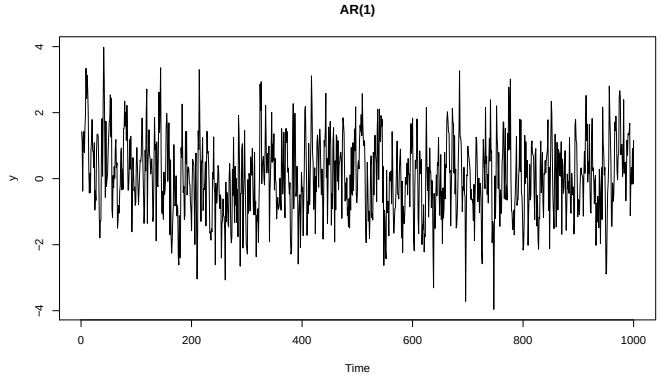


Figure 3. The original time series shows mean-reversion behavior.

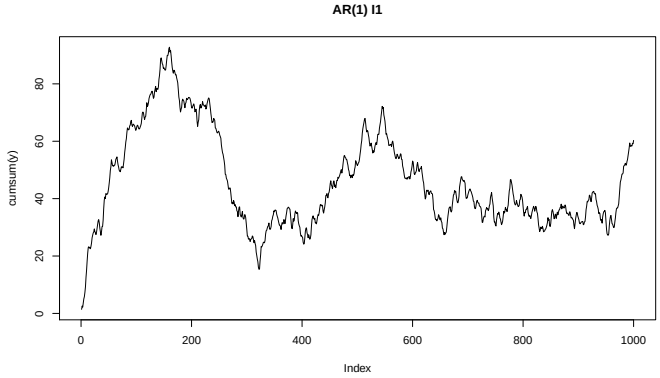


Figure 4. The integrated time series behaves like a random walk.

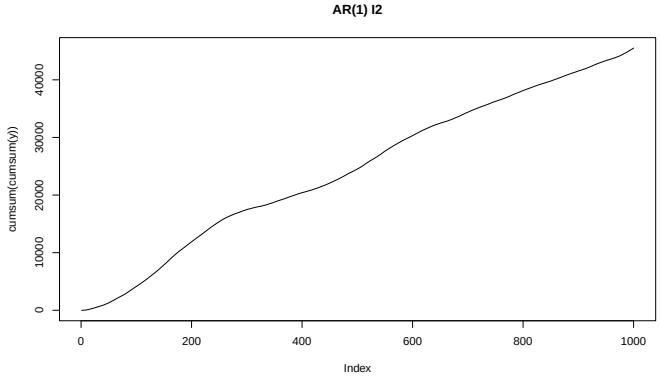


Figure 5. The twice integrated time series shows what is known as momentum, the tendency to a particular direction. This is a clear sign of second order integration.

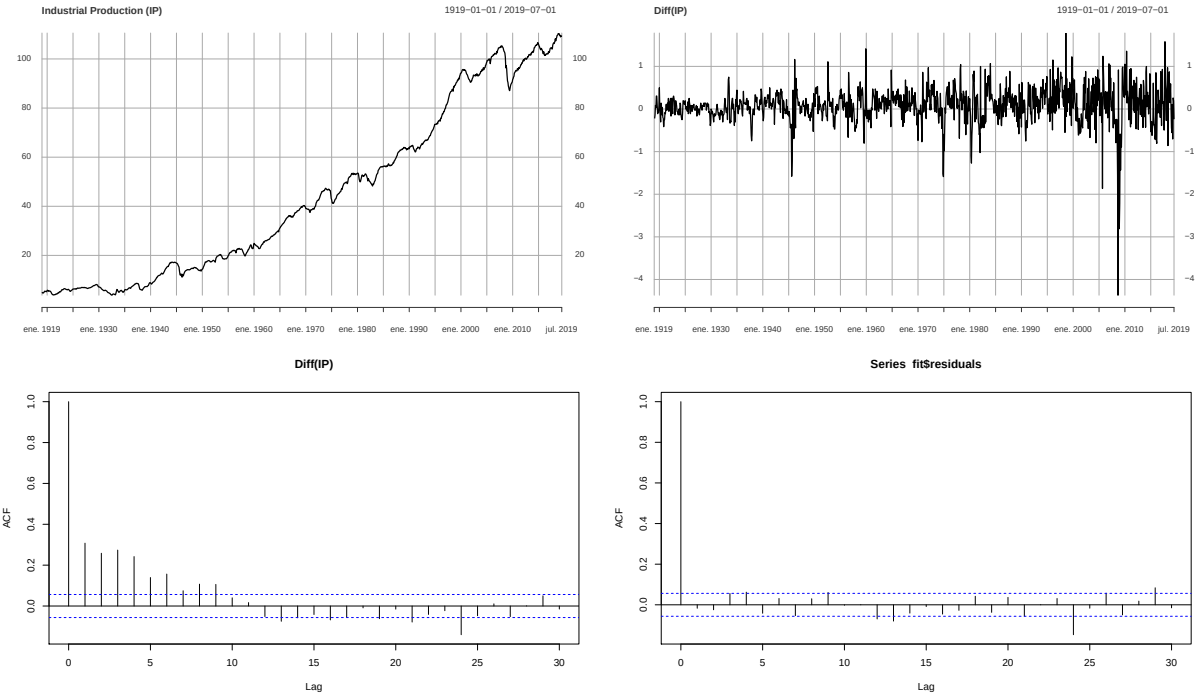
Fitting an $ARIMA$ model follows the same procedure as the one showed with $ARMA$ processes. The following example shows the fitting of an Industrial Production Index time series to an $ARIMA(1, 1, 1)$:

- The first two plots shows the original series and the differentiating one.
- Plotting the ACF for ΔY_t seems to fit an $ARMA$ process.

- $ARMA(1,1)$ is chosen based on BIC criteria.
- We finally check ACF plot for residuals, concluding a satisfactory fit.

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

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

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

pdf("IP.pdf",width=10,height=6)
plot(INDPRO,main="Industrial Production (IP)")
dev.off()

pdf("IP_diff.pdf",width=10,height=6)
plot(diff(INDPRO),main="Diff(IP)")
dev.off()

pdf("IP_diff_ACF.pdf",width=10,height=6)
acf(diff(INDPRO),na.action = na.pass,main="Diff(IP)")
dev.off()

fit<-arima(x=diff(INDPRO),order=c(1,0,1))
fit

pdf("IP_diff_ACF_res.pdf",width=10,height=6)
acf(fit$residuals,na.action = na.pass)
dev.off()
```

```
 Call:
arima(x = diff(INDPRO), order = c(1, 0, 1))

Coefficients:
         ar1      ma1  intercept
      0.8492  -0.6189     0.0857
s.e.  0.0279   0.0398     0.0275

sigma^2 estimated as 0.1399:  log likelihood = -509.76,  aic = 1027.52
```



If a nonstationary process has a constant mean, then the first difference of it has mean zero. The `arima` function in R makes this assumption. In order to address this issue use `auto.arima` instead.

8 Forecasting

The process of forecasting means using a calibrated model to make predictions of the future behavior of the time series. Consider forecasting $AR(1)$ and assume known $Y_1, \ldots, Y_n$ and estimates $\hat{\mu}$ and $\hat{\phi}$. We know that

$$Y_{n+1} = \mu + \phi(Y_n - \mu) + \epsilon_{n+1}$$

but since ϵ_{n+1} is independent of the past and Y_n, the best prediction of ϵ_{n+1} is its mean 0. Therefore our best guest is

$$\hat{Y}_{n+1} = \hat{\mu} + \hat{\phi}(Y_n - \hat{\mu}) \, .$$

Furthermore, the reasoning can be iterated to obtain the k-th estimator

$$\hat{Y}_{n+k} = \hat{\mu} + \hat{\phi}^k(Y_n - \hat{\mu}) \, .$$

Note that if $|\hat{\phi}| < 1$, as for stationary processes, then as k increases the process forecasts converge exponentially to the mean. The same derivation can be applied to $MA(1)$ process, since the process follows

$$Y_{n+1} = \mu + \epsilon_{n+1} + \theta\epsilon_n$$

our best estimate is

$$\hat{Y}_{n+1} = \hat{\mu} + \hat{\theta}\hat{\epsilon_n} \, ,$$

where $\hat{\epsilon_n}$ is the last observed error which can be computed recursively

$$\hat{\epsilon_n} = Y_n - \hat{\mu} - \hat{\theta}\epsilon_0$$

and ϵ_0 can be set to 0 for large n. Finally note that,

$$\hat{Y}_{n+k} = \hat{\mu}$$

for $k \geq 2$. The above derivations generalize easily to $ARMA$ processes, generally speaking the forecasting process consists in replacing future observations with their forecasts, future errors with zero, and past errors with the corresponding residuals in the corresponding governing equation of the model.

To obtain the uncertainty interval of the forecast we simply compute the standard deviation of the forecast and multiply by $z_{\alpha/2}$, assuming ϵ_i is a Gaussian white noise. This is the typical output by most statistical software and R. The R function `predict` receives a model (as the ones given by the function `arima`) and returns the prediction.

9 Recap

The idea of this section was to present a family of models to fit macroeconomic data. To this end we have seen empirically that such series, or transformations of them, could be considered stationary and hence look at stationary models. Weekly stationary models are described by their autocorrelation function, and therefore by a set of infinite parameters. To tackle this issue we have presented a set of parsimonious models with a finite number of parameters, the $ARMA$ models. Nevertheless we have seen in our examples that $ARMA$ models cannot capture completely the features we see in our macroeconomic data, specially the heterocedasticity or volatility clustering seen for example in the BMW price series.

One way to proceed is to introduce a second level of complexity to model the volatility component of the time series. Even though the unconditional volatility will remain constant, as the models will still be stationary, we can model the conditional volatility to capture periods of persistent high or low volatility, i.e. cluster periods.

References

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