MUNI
ECON

Applied Financial Econometrics

Class 4: ARDL models and introduction to volatility modelling

Lecturer: Axel A. Araneda, Ph.D.

Reviewing the previous class

Stationarity arises when Module of the roots from the characteristic polynomial should be greater than 1.

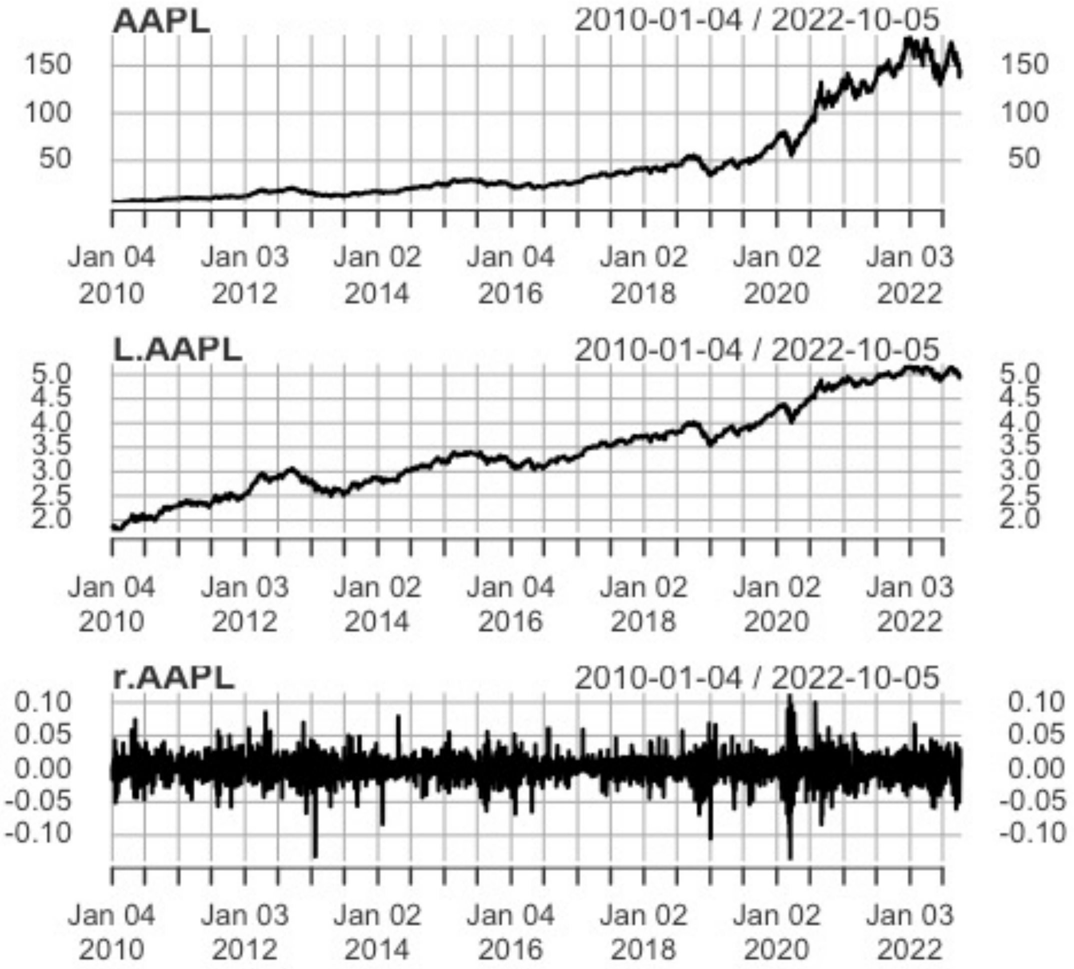
If the process has unit root (with multiplicity d) we just need to differentiate the process d times to get a stationary process.

An Integrated process is a non-stationary process, which can be transformed to a stationary process by differentiating.

$$y_t \sim I(d) \Rightarrow \Delta^d y_t \sim I(0)$$

MUNI ECON

Reviewing the previous class




```
###### Dickey-Fuller test ###########
#library(tseries)
adf.test(AAPL,k=1) # k: number of lags
adf.test(L.AAPL,k=1)
adf.test(r.AAPL[2:length(r.AAPL)],k=1)
```

```
        Augmented Dickey-Fuller Test

data:  AAPL
Dickey-Fuller = -1.4038, Lag order = 1, p-value = 0.8307
alternative hypothesis: stationary
```

```
data:  L.AAPL
Dickey-Fuller = -2.3349, Lag order = 1, p-value = 0.4365
alternative hypothesis: stationary
```

```
data:  r.AAPL[2:length(r.AAPL)]
Dickey-Fuller = -40.737, Lag order = 1, p-value = 0.01
alternative hypothesis: stationary
```


MUNI
ECON

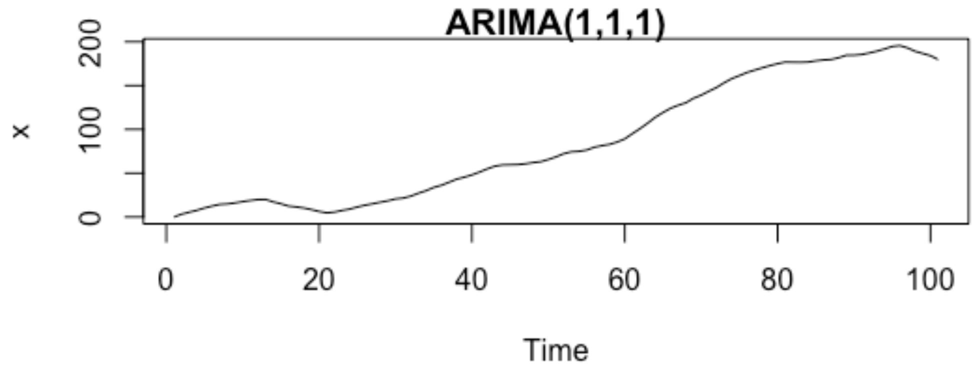
Reviewing the previous class

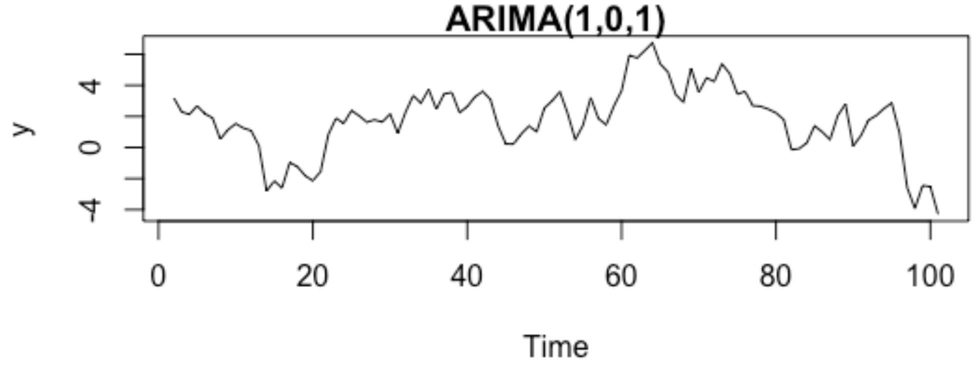
ARIMA(p,d,q):

$$x_t \sim I(d), \quad y_t = \Delta^d x_t, \quad \varepsilon_t \sim N(0,1)$$

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-i} + \varepsilon_t$$

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) y_t = \left(1 + \sum_{j=1}^{q} \theta_i L^i\right) \varepsilon_t$$






```
par(mar=c(5,4,1,1),mfrow=c(2,1))
x<-arima.sim(list(order=c(1,1,1),ar=.9,ma=.2),n=100)
plot(x,ylab='x',main='ARIMA(1,1,1)')
plot(diff(x),ylab='y',main='ARIMA(1,0,1)')
```


MUNI ECON

Reviewing the previous class

```
library(lmtest)
arima_1_1_1=arima(L.AAPL, order=c(1,1,1))
coeftest(arima_1_1_1)
```

```
z test of coefficients:

     Estimate Std. Error z value Pr(>|z|)
ar1 -0.36729    0.31992 -1.1481   0.2509
ma1  0.32451    0.32331  1.0037   0.3155
```

```
arima_1_1_0=arima(L.AAPL, order=c(1,1,0))
coeftest(arima_1_1_0)
```

```
     Estimate Std. Error z value Pr(>|z|)
ar1 -0.043228   0.017622 -2.4531  0.01416 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’
```

```
arima_0_1_1=arima(L.AAPL, order=c(1,1,0))
coeftest(arima_0_1_1)
```

```
     Estimate Std. Error z value Pr(>|z|)
ma1 -0.042605   0.017506 -2.4337  0.01494 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’
```

```
arima_0_0_0=arima(L.AAPL, order=c(0,1,0))
coeftest(arima_0_0_0)
```

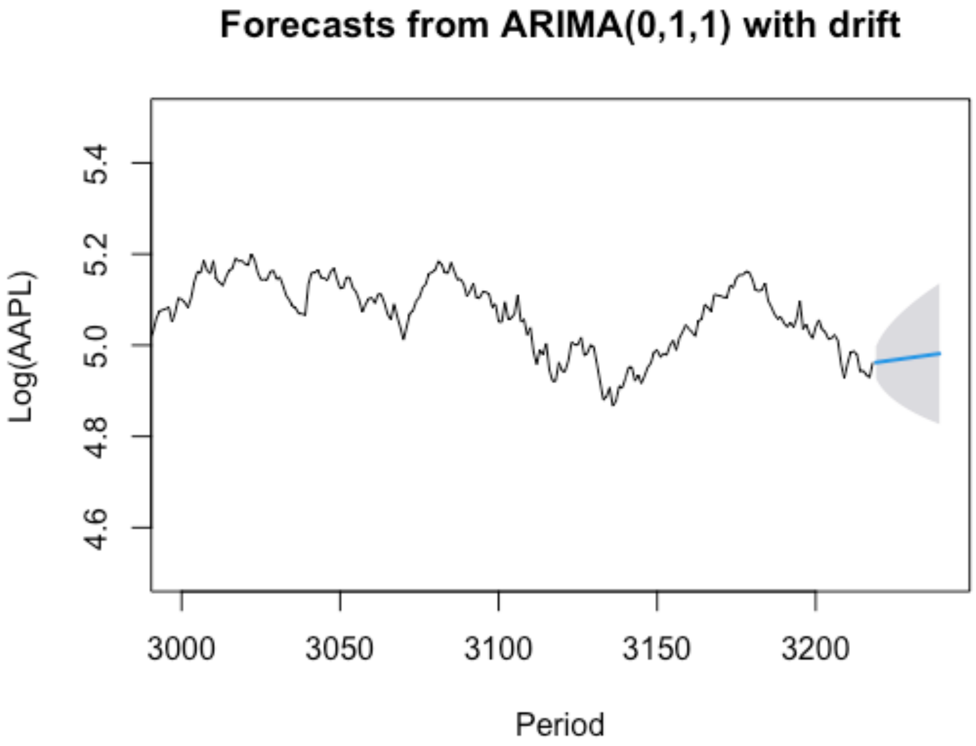
Reviewing the previous class

```
##### Model selection #####

#comparing AIC (lower wins)
arima_1_1_1$aic
arima_1_1_0$aic
arima_0_1_1$aic
arima_0_1_0$aic

#comparing BIC (lower wins)
BIC(arima_1_1_1)
BIC(arima_1_1_0)
BIC(arima_0_1_1)
BIC(arima_0_1_0)
```

```
auto_arima <- auto.arima(L.AAPL)
```




```
##################Forecasting ####################
forecasting<-forecast(auto_arima,21,level=95)
# Forecast future 21 values and 95% CI
plot(autoarima_forecasting, xlim=c(3000,3239),
ylim=c(4.5,5.5),ylab=('Log(AAPL)'), xlab=('Period'))
```

Lecture 4
ARDL models and introduction to volatility modeling

- ARDL, Heteroskedasticity, ARCH, GARCH, EWMA.

Distributed lagged models

- If the model includes one or more lagged values of the dependent variable among its explanatory variables, it is called an autorregressive model.

$$Y_t = \alpha + \beta X_t + \gamma Y_{t-1} + u_t$$

Distributed lagged models

- In regression analysis with time series data, when the regression model includes not only current values but also lagged (past) values of the explanatory variables (the X's), it is called a distributed lagged model.

$$Y_t = \alpha + \beta_0 X_t + \beta_1 X_{t-1} + \beta_2 X_{t-2} + \cdots + \beta_k X_{t-k} + u_t$$

- The coecient β_0 is known as the short-run or impact multiplier because it gives the change in the mean value of Y that follows a unit change in X in the same period.

Distributed lagged models

- If the change in X remains the same from the beginning, then $(\beta_0 + \beta_1)$ gives the change in (the mean value of) Y in the next period; $(\beta_0 + \beta_1 + \beta_2)$ in the one that follows, and so on. These partial sums are denoted as interim, or intermediate, multipliers X. Finally, after k periods we obtain the long-run or total distributed lag multiplier:

$$\sum \beta i = \beta_0 + \beta_1 + \beta_2 + \ldots + \beta_k = \beta$$

- The partial sums of the standardized β_i ($\dot{\beta}_i = \beta_i / \beta$) give the proportion of the long-run, or total, impact felt during a certain period.

Distributed lagged models

```
getSymbols('^GSPC',src='yahoo', from="2010-01-01",periodicity = 'daily')
getSymbols('^RUT',src='yahoo',from="2010-01-01",periodicity = 'daily')
SP<-GSPC[,6]
RUT<-RUT[,6]
dSP<-diff(SP)
dRUT<-diff(RUT)
adf.test(as.numeric(na.omit(dRUT)),k=1)
reg1<-lm(dRUT~dSP,na.action=na.exclude)
reg1<-lm(dRUT~dSP,na.action=na.exclude)
reg2<-lm(dRUT~dSP +lag(dSP),na.action=na.exclude)
coeftest(reg1)
coeftest(reg2)
```

Distributed lagged models

```
getSymbols('^GSPC',src='yahoo', from="2010-01-01",periodicity = 'daily')
getSymbols('^RUT',src='yahoo',from="2010-01-01",periodicity = 'daily')
SP<-GSPC[,6]
RUT<-RUT[,6]
dSP<-diff(SP)
dRUT<-diff(RUT)
adf.test(as.numeric(na.omit(dRUT)),k=1)
reg1<-lm(dRUT~dSP,na.action=na.exclude)
reg1<-lm(dRUT~dSP,na.action=na.exclude)
reg2<-lm(dRUT~dSP +lag(dSP),na.action=na.exclude)
coeftest(reg1)
coeftest(reg2)
```

```
#install.packages('dynlm')
library(dynlm)
reg1_b<-dynlm(d(RUT, 1) ~ d(SP))
reg2_b<-dynlm(d(RUT)~L(d(SP), 0:1)) #1lag
```


MUNI
ECON

Distributed lagged models

```r
getSymbols('^GSPC',src='yahoo', from="2010-01-01",periodicity = 'daily')
getSymbols('^RUT',src='yahoo',from="2010-01-01",periodicity = 'daily')
SP<-GSPC[,6]
RUT<-RUT[,6]
dSP<-diff(SP)
dRUT<-diff(RUT)
adf.test(as.numeric(na.omit(dRUT)),k=1)
reg1<-lm(dRUT~dSP,na.action=na.exclude)
reg1<-lm(dRUT~dSP,na.action=na.exclude)
reg2<-lm(dRUT~dSP +lag(dSP),na.action=na.exclude)
coeftest(reg1)
coeftest(reg2)
```

```r
#install.packages('dynlm')
library(dynlm)
reg1_b<-dynlm(d(RUT, 1) ~ d(SP))
reg2_b<-dynlm(d(RUT)~L(d(SP), 0:1)) #1lag
```

Autoregresive distributed lagged models

- An autoregressive distributed lag model (ARDL) is a model that contains both independent variables and their lagged values as well as the lagged values of the dependent variable

- ARDL(p,q):

$$Y_t = \delta + \theta_1 Y_{t-1} + ... + \theta_p Y_{t-p} + \delta_1 X_{t-1} + ... + \delta_q X_{t-q} + v_t$$

MUNI
ECON

Selection model criteria: AIC

– Akaike Information Criteria

$$AIC = log\hat{\sigma}_k^2 + \frac{n+2k}{n}$$

where $\hat{\sigma}_k^2 = \frac{SSE_k}{n}$, and k is the number of model parameters, n the sample size, and SSE_k is equal to the sum of the squared residuals under the model k ($SSE_k = \sum_{t=1}^{n}(x_t - \bar{x})^2$).

– The value of k that produces the minimum AIC represents the best model. parameters.

MUNI ECON

Selection model criteria: BIC

- Bayesian Information Criteria

$$\boxed{AICc = log\hat{\sigma}_k^2 + \frac{klogn}{n}}$$

- Simulation studies have verified that BIC is adequate to obtain the correct order in large samples, while AICc tends to be superior in smaller samples where the relative number of parameters is large.

MUNI ECON

Forecasting performance

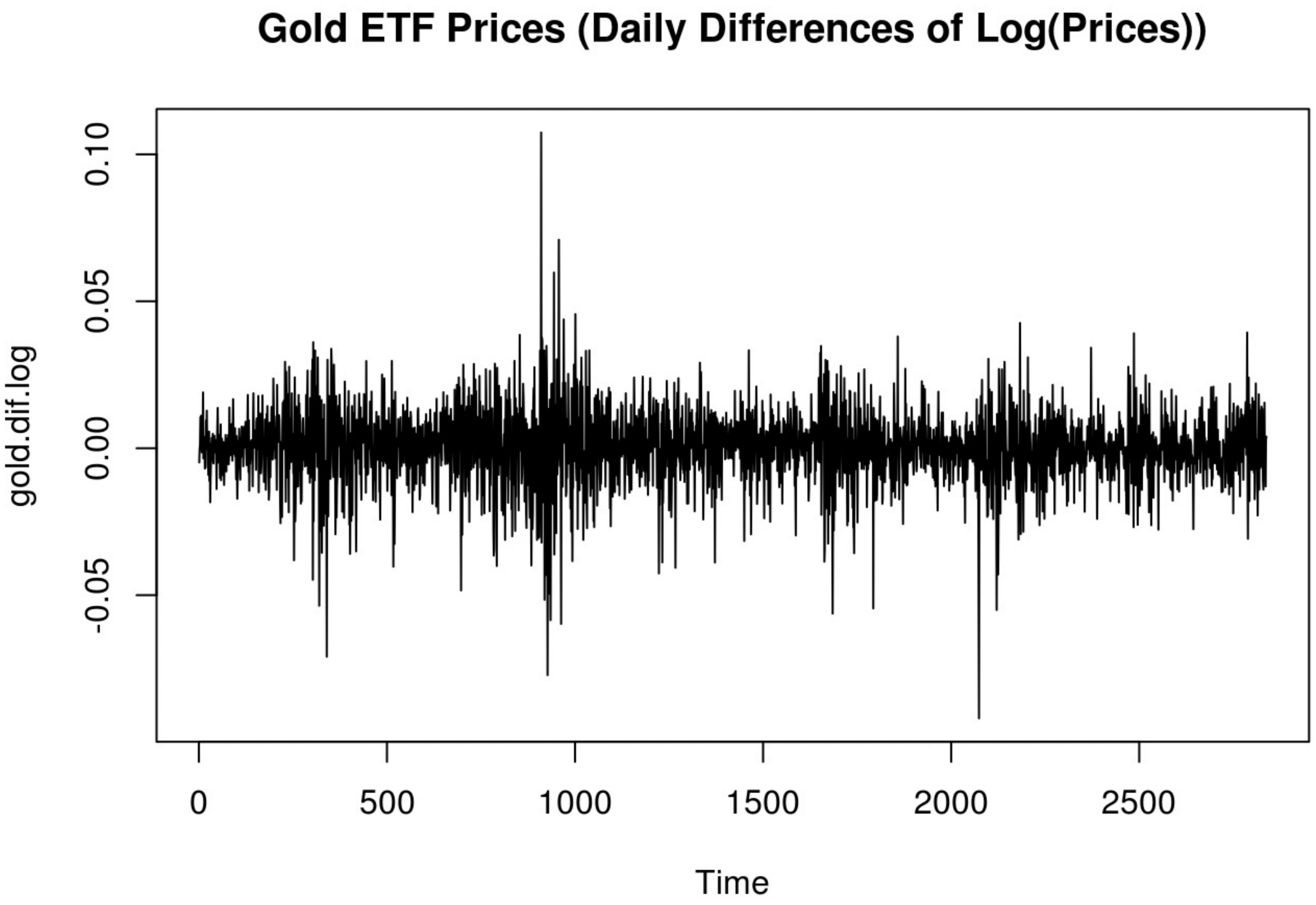
1. Mean Error,
2. MAE - Mean Absolute Error,
3. MSE - Mean Squared Error,
4. MIS - Mean Interval Score
5. MPE - Mean Percentage Error
6. MAPE - Mean Absolute Percentage Error

MUNI ECON

Volatility modelling

MUNI ECON

Volatility modelling



MUNI ECON

Volatility modelling

```
# Ljung-Box test
Box.test(gold.dif.log, lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  gold.dif.log
## X-squared = 19.676, df = 20, p-value = 0.4783
```

There is no well defined temporal structure in the transformed data (gold.dif.log).

The Auto Arima points to a random walk.

The Ljung-Box test applied to the transformed data does not reject the null hypothesis (inexistance of autocorrelations), with a p-value of 0.4783.

Given these assertions we will fit a linear model to the transformed data with only the intercept in order to impose a zero mean series.

MUNI ECON

ARCH type

A time series $\{\epsilon_t\}$ is given at each instance by:

$$\epsilon_t = \sigma_t w_t$$

Where $\{w_t\}$ is discrete white noise, with zero mean and unit variance, and σ_t^2 is given by:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2$$

Where α_0 and α_1 are parameters of the model.

We say that $\{\epsilon_t\}$ is an *autoregressive conditional heteroskedastic model of order unity*, denoted by ARCH(1). Substituting for σ_t^2, we receive:

$$\epsilon_t = w_t \sqrt{\alpha_0 + \alpha_1 \epsilon_{t-1}^2}$$

ARCH type

$$
\begin{aligned}
\mathrm{Var}(\epsilon_t) &= \mathrm{E}[\epsilon_t^2] - (\mathrm{E}[\epsilon_t])^2 \\
&= \mathrm{E}[\epsilon_t^2] \\
&= \mathrm{E}[w_t^2]\,\mathrm{E}[\alpha_0 + \alpha_1 \epsilon_{t-1}^2] \\
&= \mathrm{E}[\alpha_0 + \alpha_1 \epsilon_{t-1}^2] \\
&= \alpha_0 + \alpha_1 \mathrm{Var}(\epsilon_{t-1})
\end{aligned}
$$

It is straightforward to extend ARCH to higher order lags. An ARCH(p) process is given by:

$$
\epsilon_t = w_t \sqrt{\alpha_0 + \sum_{i=1}^{p} \alpha_p \epsilon_{t-i}^2}
$$

GARCH type

A time series $\{\epsilon_t\}$ is given at each instance by:

$$\epsilon_t = \sigma_t w_t$$

Where $\{w_t\}$ is discrete white noise, with zero mean and unit variance, and σ_t^2 is given by:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2$$

Where α_i and β_j are parameters of the model.

We say that $\{\epsilon_t\}$ is a *generalised autoregressive conditional heteroskedastic model of order p,q*, denoted by GARCH(p,q).

MUNI
ECON

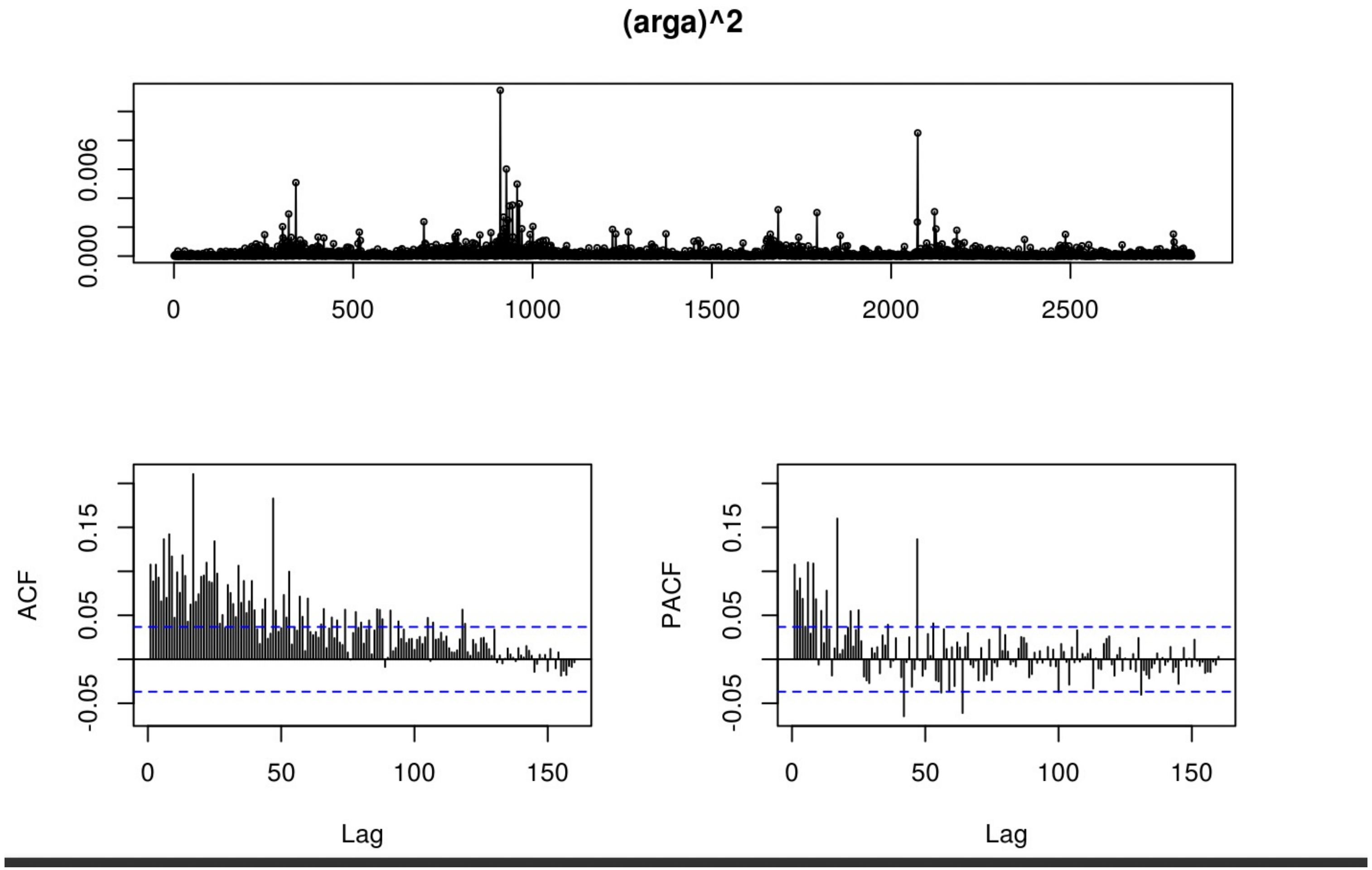
Example

```r
# fit of the linear model with just an intercept
fitlin <- Arima(gold.dif.log, order = c(0,0,0))
summary(fitlin)
```

```r
# analysis of the residuals
arga <- fitlin$residuals
tsdisplay(arga, lag.max = 160)
```

```r
tsdisplay((arga)**2, lag.max = 160)
```

Example



MUNI ECON

Example

```
gfit <- garchFit(~garch(1,1), arga, trace=F)
summary(gfit)
```


MUNI
ECON