ARMA MODELING AND ESTIMATION IN R

1. INTRODUCTION

Import data into R, create a time series object, and construct time series plots

```
mydatats1<- read.table("…./datats.txt")
y <- mydatats1$V1
j=ts(y, frequency=4, start = c(1960,1))
```

Below are presented the time series plot of the J&J data, of the log of J&J and of the first differences of the log of J&J data, together with their corresponding histograms. The log of J&J time series and the first differences of the log of J&J data are computed by:

```
lj=log(j)      # compute the logartithm of the J&J
dlj=diff(lj)   # compute the first differences of the log of J&J
```

A nice graph can be produced by using the command par(mfrow=c(rows,col)), which splits the graph into (rows x col) subplots:

```
par(mfrow=c(3,2))      # set up the graphics
plot(j,type="l", col='red', lwd=1,main="Time Series plot of Johnson & Johnson", ylab="Quarterly earnings per share")
hist(j, nclass=15, main="Histogram of Johnson & Johnson")
plot(lj,type="l", col='red', lwd=1,main="Log of Johnson & Johnson", ylab="Log of Quarterly earnings per share")
hist(lj, nclass=15, main="Histogram of log of Johnson & Johnson")
plot(dlj,type="l", col='red', lwd=1,main="Differences of log of Johnson & Johnson")
hist(dlj, nclass=15, main="Histogram of differences of log of J&J")
```

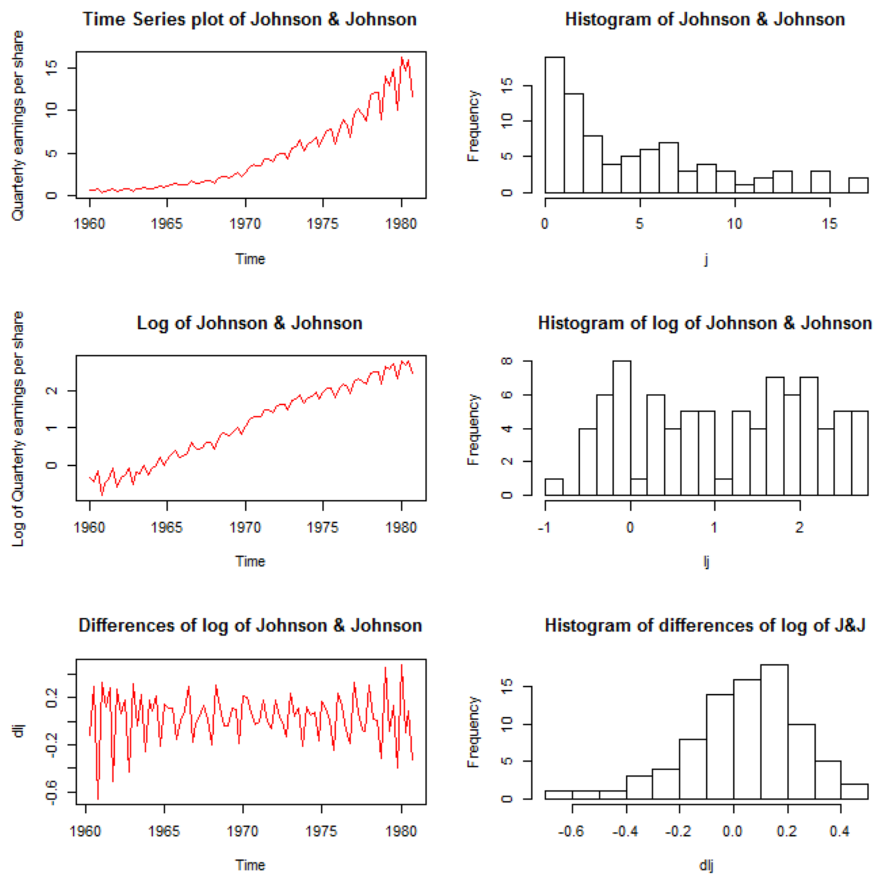


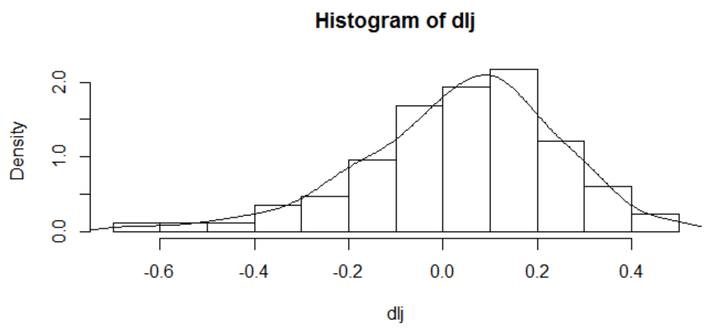
Figure 1: Time series plots and histograms for the J&J, log of J&J and the differences of log(J&J)

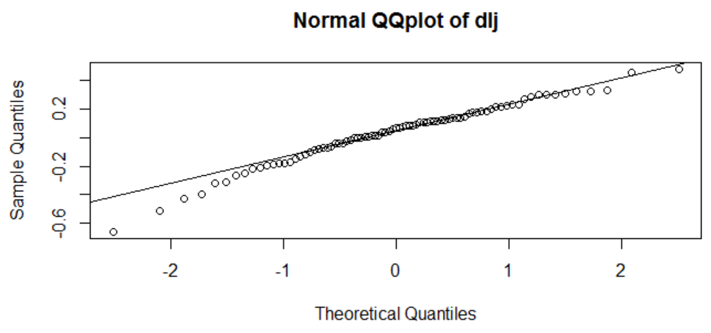
We focus on the stationary time series, i.e. the differences of the logarithms of the J&J series. We can test for normality of dlj, and create the plot of the histogram together with a density plot, and also the normal QQplot:

```
Shapiro.test(dlj)                          # Shapiro test of normality
```

```
Shapiro-Wilk normality test:   data:  dlj
W = 0.97251, p-value = 0.07211
```

```
par(mfrow=c(2,1))
hist(dlj, prob=TRUE, 15)    # histogram
lines(density(dlj))         # smooth it - ?density for details
qqnorm(dlj,main="Normal QQplot of dlj")     # normal Q-Q plot
qqline(dlj)                                 # add a line
```



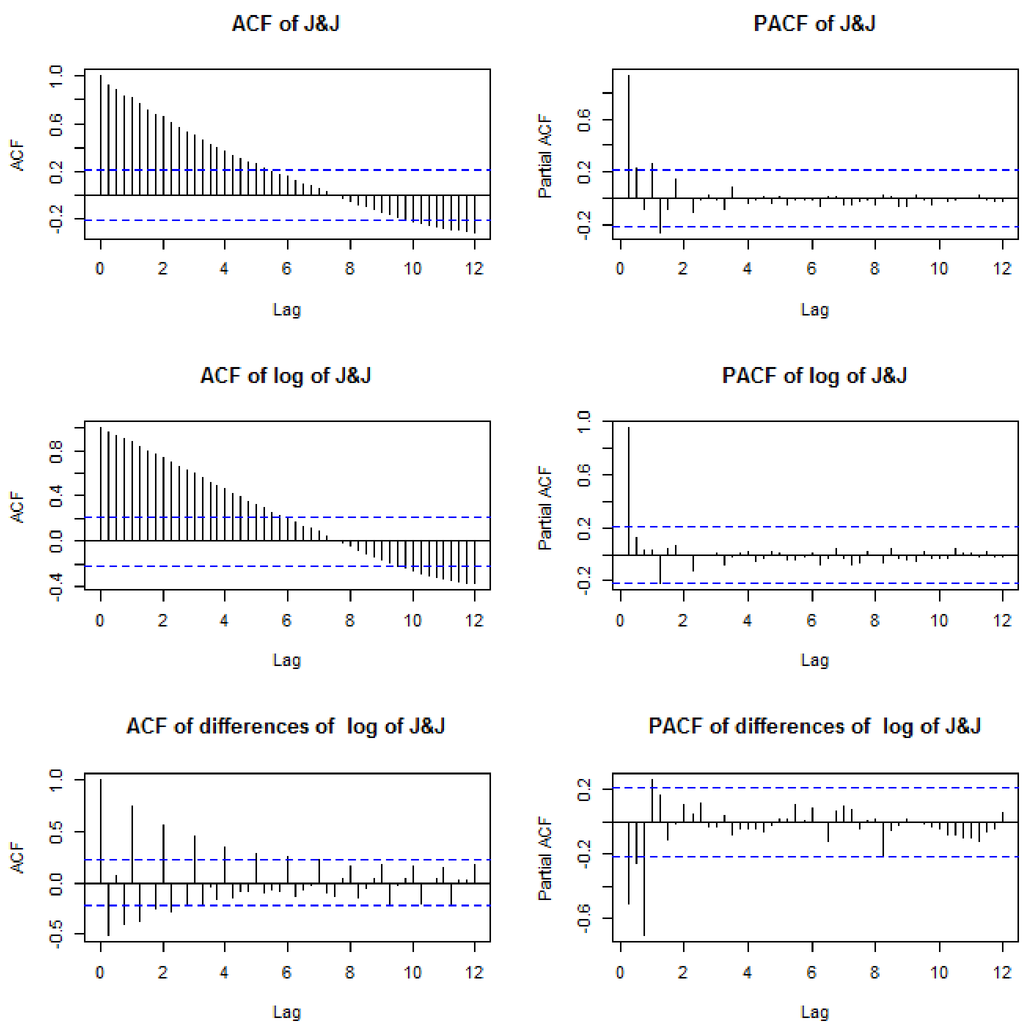


2. TIME SERIES ANALYSIS – BOX JENKINS METHODOLOGY

2.1. Identification step

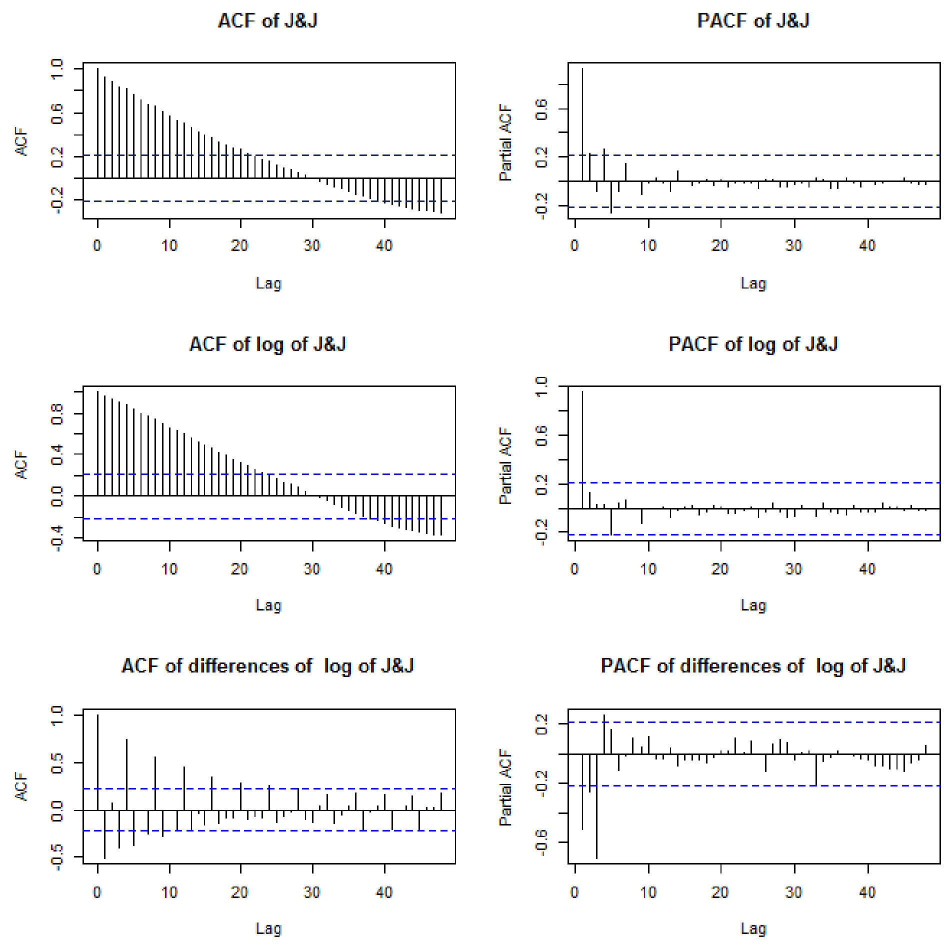
It is based on the autocorrelation and partial autocorrelation plot. The acf( ) command and the pacf( ) command create an autocorrelation and a partial autocorrelation plot, respectively. Below are produced the autocorrelation and partial autocorrelation plots of the J&J data, the log of J&J and of the first differences of the log of J&J data using the command par(mfrow=c(3,2)).

```
par(mfrow=c(3,2))       # set up the graphics
acf(j, 48, main="ACF of J&J")       # autocorrelation function plot
pacf(j, 48, main="PACF of J&J")     # partial autocorrelation function
acf(lj, 48, main="ACF of log of J&J")
pacf(lj, 48, main="PACF of log of J&J")
acf(dlj, 48, main="ACF of differences of  log of J&J")
pacf(dlj, 48, main="PACF of differences of  log of J&J")
```



Note that the lag values in the X axis are 1, 2, 3, 4, 5,… and correspond to lags 4, 8, 12, 16, 20,… because we have quarterly data, i.e. the frequency is 4. A better type of labeling can be produced by using the following set of commands:

```
par(mfrow=c(3,2))       # set up the graphics
acf(ts(j,freq=1), 48, main="ACF of J&J")       # autocorrelation function plot
pacf(ts(j,freq=1), 48, main="PACF of J&J")    # partial autocorrelation function plot
acf(ts(lj,freq=1), 48, main="ACF of log of J&J")
pacf(ts(lj,freq=1), 48, main="PACF of log of J&J")
acf(ts(dlj,freq=1), 48, main="ACF of differences of  log of J&J")
pacf(ts(dlj,freq=1), 48, main="PACF of differences of  log of J&J")
```



2.2. Estimation of ARMA models

The estimation of a specified ARMA model can be done by using the command arima() of R. A frequently used form of the command is:

```
arima(y, order = c(p,d,q), seasonal = list(order = c(ps,ds,qs), period = freq), include.mean = TRUE,
            fixed = NULL, method = c("CSS-ML", "ML", "CSS"), optim.method = "BFGS")
```

where

y: is the univariate time series under consideration.

order: specifies the non-seasonal part of the ARIMA model, i.e. p denotes the order of the Autoregressive part [AR(p)], q denotes the order of the Moving Average part [MA(q)] and d denotes the order of differencing I[(d)].

seasonal: specifies the seasonal part of the ARIMA model, i.e. ps denotes the order of seasonal AR part, qs denotes the order of seasonal MA part, ds denotes the order of seasonal differencing and period refers to the frequency of the analyzed series, i.e. period=4 for quarterly data, period=12 for minthly data.

include.mean: equals TRUE if the ARMA model includes the **mean** of the analyzed series, FALSE if the ARMA model has zero mean.

fixed: is a vector of same length as the number of model parameters to be estimated (the ARMA parameters and the mean). It takes the value of 0, if the corresponding parameter will not be estimated, and takes NA if the corresponding parameter will be estimated.

method: denotes the estimation method, i.e. maximum likelihood (ML), minimize conditional sum-of-squares (CSS), or first conditional-sum-of-squares to find starting values and then maximum likelihood (CSS-ML).

optim.method: denotes the optimization algorithm used.

The general form of the ARMA model, which is estimated by the command arima() if include.mean=FALSE is:

$$y_t = \varphi_1 y_{t-1} + \ldots + \varphi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t \,,$$

and

$$y_t - \mu = \varphi_1 (y_{t-1} - \mu) + \ldots + \varphi_p (y_{t-p} - \mu) + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t \,,$$

If include.mean=TRUE, which is the default choice of the command. Note that for ARIMA models, i.e. when differencing is used, the differenced series follows a zero mean ARMA model[1].

For illustration purposes, first, we will estimate some moving average (MA) models in R, using the differences of the logarithm of the J&J series (dlj), which is a stationary process (as shown in previous lectures). The command ma1fit=arima(dlj,order=c(0,0,1)) estimates a simple MA(1) model for the dlj series and returns the object/list ma1fit. Note that ma1fit is an object/list containing several results. For example, it contains the coefficients (ma1fit$coef), the estimated residual series (ma1fit$residuals), the Akaike Information Criterion AIC (ma1fit$aic).

```
ma1fit=arima(dlj,order=c(0,0,1))
ma1fit
```

[1] In R the commands arima(diff(y),order=c(1,0,1)) and arima(y,order=c(1,1,1)) provide different model estimates!.

```
Call: arima(x = dlj, order = c(0, 0, 1))
Coefficients:        ma1     intercept
                 -0.8246    0.0393
            s.e.  0.0582    0.0032
sigma^2 estimated as 0.0234:  log likelihood = 37.5,  aic = -69
```

The estimated model can be written in the form:

$$y_t - 0.0393 = -0.8246\varepsilon_{t-1} + \varepsilon_t \quad \text{or} \quad y_t = 0.0393 - 0.8246\varepsilon_{t-1} + \varepsilon_t$$

That is, in the case of MA(q) models the estimated intercept of the model is also the mean of the analyzed series. This holds only for the MA models.

We also estimate the MA(1) model in Eviews for comparison. The results are presented below:

```
Dependent Variable: DLJ
Method: Least Squares
Sample (adjusted): 1960Q2 1980Q4
Included observations: 83 after adjustments
Convergence achieved after 8 iterations
MA Backcast: 1960Q1

        Variable          Coefficient    Std. Error    t-Statistic     Prob.

           C                 0.040051      0.002714       14.75513    0.0000
         MA(1)              -0.851227      0.054221      -15.69910    0.0000

R-squared                    0.476610   Mean dependent var           0.033667
Adjusted R-squared           0.470148   S.D. dependent var           0.210213
S.E. of regression           0.153016   Akaike info criterion       -0.892751
Sum squared resid            1.896518   Schwarz criterion           -0.834466
Log likelihood               39.04917   Hannan-Quinn criter.        -0.869335
F-statistic                  73.76018   Durbin-Watson stat           2.196182
Prob(F-statistic)            0.000000

Inverted MA Roots               .85
```

Lets now fit a MA(4) model. The command is:

```
ma4fit=arima(dlj,order=c(0,0,4))
ma4fit
```

```
Call: arima(x = dlj, order = c(0, 0, 4))
Coefficients:
        ma1       ma2      ma3      ma4     intercept
     -0.6578   -0.1495  -0.4180   0.7598    0.0366
s.e.  0.0939    0.1296   0.1035   0.0732    0.0068
sigma^2 estimated as 0.01383:  log likelihood = 57.65,  aic = -103.31
```

The estimated model can be written in the form:

$$y_t - 0.0366 = -0.6578\varepsilon_{t-1} - 0.1495\varepsilon_{t-2} - 0.418\varepsilon_{t-3} + 0.7598\varepsilon_{t-4} + \varepsilon_t \text{ or}$$

$$y_t = 0.0366 - 0.6578\varepsilon_{t-1} - 0.1495\varepsilon_{t-2} - 0.418\varepsilon_{t-3} + 0.7598\varepsilon_{t-4} + \varepsilon_t$$

Note, that if minimization of the conditional sum of squares (CSS) is used to estimate the model parameters, we receive the following results

```
ma4afit=arima(dlj,order=c(0,0,4),method=c("CSS"))
ma4afit
```

```
Call: arima(x = dlj, order = c(0, 0, 4), method = c("CSS"))
Coefficients:
        ma1      ma2      ma3      ma4     intercept
     -0.6216  -0.1615  -0.3922   0.6940     0.0353
s.e.  0.0800   0.0977   0.0957   0.0778     0.0071
sigma^2 estimated as 0.01645:  log likelihood = 52.68,  aic = NA
```

Finally, if we want to estimate a restricted MA model (say with $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = 0$ and only the fourth parameter will be estimated , i.e. $\theta_4 \neq 0$) we use the option fixed() described above as follows:

```
ma4restricted=arima(dlj,order=c(0,0,4),fixed=c(0,0,0,NA,NA))
ma4restricted
```

```
Call: arima(x = dlj, order = c(0, 0, 4), fixed = c(0, 0, 0, NA, NA))
Coefficients:
       ma1  ma2  ma3    ma4     intercept
        0    0    0    0.7633    0.0314
s.e.    0    0    0    0.0815    0.0282
sigma^2 estimated as 0.02211:  log likelihood = 38.67,  aic = -71.34
```

Here, we will estimate some autoregressive (AR) models in R, using the differences of the logarithm of J&J series (dlj), which is a stationary process. The command ar1fit=arima(dlj,order=c(1,0,0)) estimates a simple AR(1) model for the dlj series and returns the object/list ar1fit.

```
ar1fit=arima(dlj,order=c(1,0,0))
ar1fit
```

```
Call: arima(x = dlj, order = c(1, 0, 0))
Coefficients:         ar1     intercept
                  -0.5226    0.0358
             s.e.  0.0950    0.0129
sigma^2 estimated as 0.03195:  log likelihood = 24.98,  aic = -43.95
```

The estimated AR(1) model can be written in the form:

$$y_t - \mu = \varphi_1(y_{t-1} - \mu) + \varepsilon_t \quad \text{or} \quad y_t = \mu - \mu\varphi_1 + \varphi_1 y_{t-1} + \varepsilon_t \quad \text{or} \quad y_t = \mu(1-\varphi_1) + \varphi_1 y_{t-1} + \varepsilon_t$$

i.e. $y_t - 0.0358 = -0.5226(y_{t-1} - 0.0358) + \varepsilon_t$ or $y_t = 0.0358(1-(-0.5226)) - 0.5226 y_{t-1} + \varepsilon_t$

or $y_t = 0.0545 - 0.5226 y_{t-1} + \varepsilon_t$

Lets now fit an AR(4) model. The command is:

```
ar4fit=arima(dlj,order=c(4,0,0))
ar4fit
```

```
Call: arima(x = dlj, order = c(4, 0, 0))
Coefficients:        ar1       ar2       ar3       ar4     intercept
                 -0.6834   -0.6104   -0.6226   0.2819    0.0384
            s.e.  0.1123    0.1181    0.1241   0.1183    0.0037
sigma^2 estimated as 0.007825:  log likelihood = 80.62,  aic = -149.25
```

The estimated AR(4) model can be written in the form:

$$y_t - \mu = \varphi_1(y_{t-1} - \mu) + \varphi_2(y_{t-2} - \mu) + \varphi_3(y_{t-3} - \mu) + \varphi_4(y_{t-4} - \mu) + \varepsilon_t \quad \text{or}$$

$$y_t - 0.038 = -0.683(y_{t-1} - 0.038) - 0.610(y_{t-2} - 0.038) - 0.622(y_{t-3} - 0.038) + 0.281(y_{t-4} - 0.038) + \varepsilon_t$$

Lets now fit an ARMA(4,1) model. The command is:

```
arma41fit=arima(dlj,order=c(4,0,1))
arma41fit
```

```
Call:arima(x = dlj, order = c(4, 0, 1))
Coefficients:
         ar1       ar2       ar3      ar4       ma1     intercept
     -0.4497   -0.3944   -0.4237   0.4910   -0.2465    0.0382
s.e.  0.2720    0.2535    0.2344   0.2423    0.2937    0.0042
sigma^2 estimated as 0.007751:  log likelihood = 80.99,  aic = -147.97
```

The estimated AR(4) model can be written in the form:

$$y_t - \mu = \varphi_1(y_{t-1} - \mu) + \varphi_2(y_{t-2} - \mu) + \varphi_3(y_{t-3} - \mu) + \varphi_4(y_{t-4} - \mu) + \theta_1\varepsilon_{t-1} + \varepsilon_t \quad \text{or}$$

$$y_t - 0.038 = -0.45(y_{t-1} - 0.038) - 0.39(y_{t-2} - 0.038) - 0.42(y_{t-3} - 0.038) + 0.49(y_{t-4} - 0.038) - 0.25\varepsilon_{t-1} + \varepsilon_t$$

Finally, if we want to estimate a restricted ARMA(4,1) model (say with $\varphi_1 = 0$, $\varphi_2 = 0$, $\varphi_3 = 0$ and only the fourth autoregressive parameter φ_4 will be estimated together with the moving average parameter and the mean), we use the option fixed() as follows:

```
arma41restricted=arima(dlj,order=c(4,0,1),fixed=c(0,0,0,NA,NA,NA))
arma41restricted
```

```
Call: arima(x = dlj, order = c(4, 0, 1), fixed = c(0, 0, 0, NA, NA, NA))
Coefficients:
        ar1   ar2  ar3   ar4        ma1     intercept
         0     0     0    0.8603   -0.8140    0.0337
s.e.     0     0     0    0.0599    0.0933    0.0113
sigma^2 estimated as 0.008292:  log likelihood = 78.35,  aic = -148.7
```

This restricted ARMA(4,1) model can be written in the form:

$$y_t - \mu = \varphi_4(y_{t-4} - \mu) + \theta_1\varepsilon_{t-1} + \varepsilon_t \quad \text{or} \quad y_t - 0.0337 = 0.8603(y_{t-4} - 0.0337) - 0.814\varepsilon_{t-1} + \varepsilon_t$$

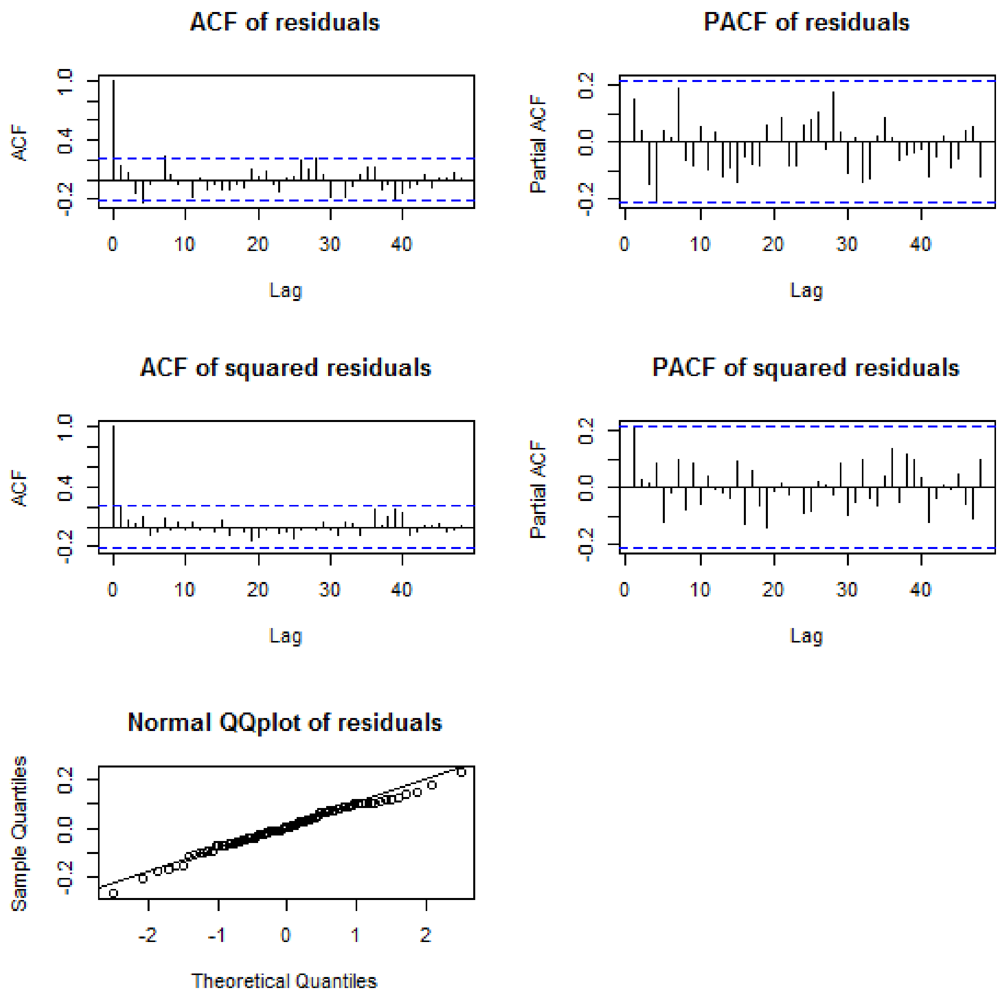
$$y_t = \mu(1-\varphi_4) + \varphi_4 y_{t-4} + \theta_1\varepsilon_{t-1} + \varepsilon_t \quad \text{or} \quad y_t = 0.0047 + 0.8603y_{t-4} - 0.814\varepsilon_{t-1} + \varepsilon_t .$$

2.3 Diagnostic plots

Now we will provide some diagnostic plots for the residuals of the above model. Based on the residuals of the restricted ARMA(4,1) model, we will present the autocorrelation plots and the partial autocorrelation plots of the estimated residuals (examine the assumption of autocorrelation of residuals), the autocorrelation plots and the partial autocorrelation plots of the squared residuals (a kind of plots to examine heteroskedasticity in the residual series), as well as some normality plots (examine the assumption of normality of residuals).

```
arma41residuals=arma41restricted$residuals
arma41residuals
residuals=ts(arma41residuals, frequency=4, start = c(1960,2))
residuals
```

```
par(mfrow=c(3,2))      # set up the graphics
acf(ts(residuals,freq=1), 48, main="ACF of residuals")
pacf(ts(residuals,freq=1), 48, main="PACF of residuals")
acf(ts(residuals^2,freq=1), 48, main="ACF of squared residuals")
pacf(ts(residuals^2,freq=1), 48, main="PACF of squared residuals")
qqnorm(residuals,main="Normal QQplot of residuals")
qqline(residuals)
```



Based on the residual plots presented above, it seems that the assumptions with respect to the residuals are satisfied, thus the restricted ARMA(4,1) models is a appropriate candidate model for modeling the stationary series of the differences of logarithm of J&J. Obviously, other alternative model specifications can be used.

2.4 Predictions

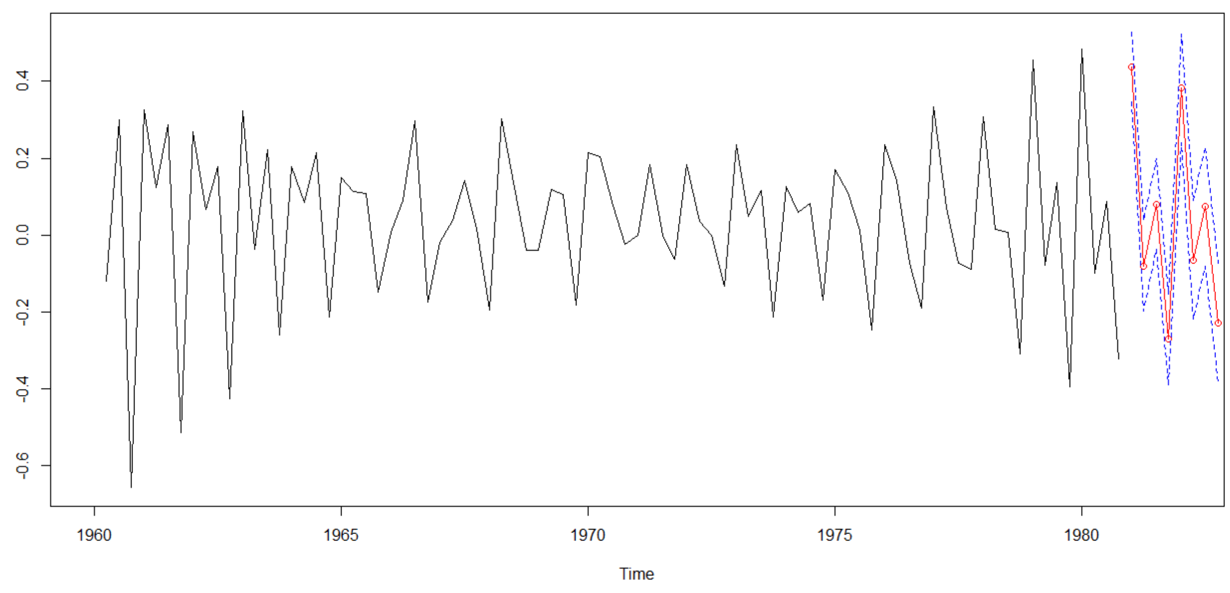
In this paragraph we compute predictions based on an estimated ARMA model using the command predict() of R. A frequently used form of the command is predict(modelfit, n.ahead), where modelfit is the estimated time series model, and n.ahead denotes the time steps ahead to predict. For our estimated restricted ARMA(4,1) model, predictions for 8 quarters ahead (i.e. two years) are obtained by using the following commands

```
forecast=predict(arma41restricted,8)
forecast
```

```
$pred
           Qtr1         Qtr2          Qtr3         Qtr4
1981  0.43859949   -0.08064274    0.08044484   -0.27229300
1982  0.38204628   - 0.06467171   0.07391625   -0.22955356
$se
            Qtr1         Qtr2         Qtr3         Qtr4
1981   0.09105964   0.11741583   0.11741583   0.11741583
1982   0.14115168   0.15488933   0.15488933   0.15488933
```

A plot of the analyzed time series together with the forecasts plus/minus one standard error is obtained by using the commands

```
# plot of forecasts with 1 s.e
UL=forecast$pred+forecast$se
LL=forecast$pred-forecast$se
minx = min(dlj,LL); maxx = max(dlj,UL)
ts.plot(dlj, forecast$pred, xlim=c(1960,1982), ylim=c(minx,maxx))
lines(forecast$pred, col="red", type="o")
lines(UL, col="blue", lty="dashed")
lines(LL, col="blue", lty="dashed")
```



2.5 Useful Comment

Note that in R, the results of the commands arima(dlj,order=c(1,0,0)) and arima(lj,order=c(1,1,0)) provide different results. See the estimation output below:

```
ar1=arima(dlj,order=c(1,0,0))

ar1

Call: arima(x = dlj, order = c(1, 0, 0))

Coefficients:        ar1   intercept

                  -0.5226     0.0358

             s.e.   0.0950     0.0129

sigma^2 estimated as 0.03195:  log likelihood = 24.98,  aic = -43.95

 dar1<-arima(lj, order=c(1,1,0))

 dar1

Call: arima(x = lj, order = c(1, 1, 0))

Coefficients:       ar1

                -0.4737

           s.e.   0.0974

sigma^2 estimated as 0.03482:  log likelihood = 21.44,  aic = -38.89
```