Moving average model estimated with the ML approach

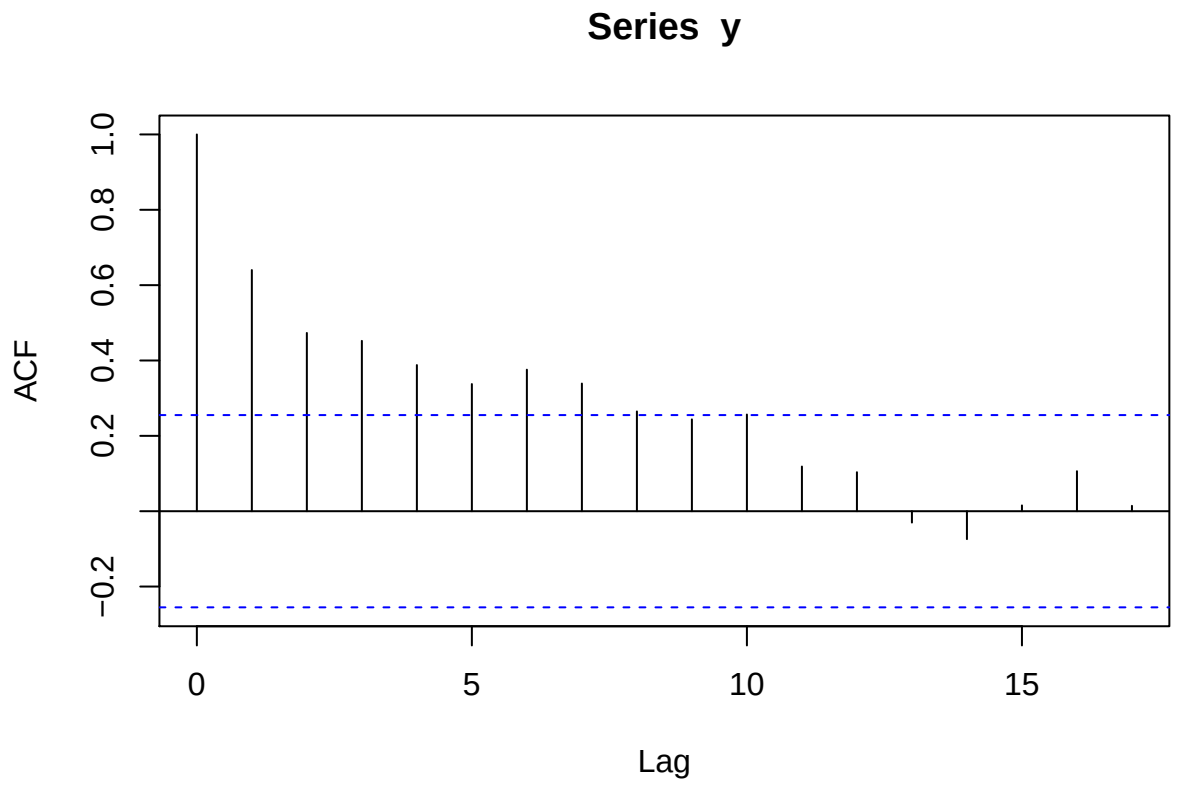
Estimation of MA(q) models and ARMA(p,q) models

In this short note, we show how to estimate MA(q) and ARMA(p,q) models using the maximum likelihood approach. In particular, we shall do the following:

1. Estimate an MA(q) model with the maximum likelihood approach.
2. Estimate an ARMA(p,q) model with the maximum likelihood approach.
3. Apply the Wald test and the likelihood ratio test.

In this document, we use the GDP growth rate of France to illustrate the estimation procedure. We first load the data. Then, we will estimate a MA(2) model and an ARMA(1,2) process.

```
url = "https://raw.githubusercontent.com/adufays/GDP_expectancy/main/france-gdp-growth-rate.csv"
data = read.csv(url, sep=",")
y = data[,2]
autocorr = acf(y)
```




```
(autocorr)
```

```
##
```

```
## Autocorrelations of series 'y', by lag
##
##     0     1     2     3     4     5     6     7     8     9    10
## 1.000 0.640 0.473 0.452 0.388 0.338 0.376 0.339 0.265 0.244 0.257
##    11    12    13    14    15    16    17
## 0.119 0.104 -0.030 -0.074 0.015 0.106 0.014
```

We observe that the series is autocorrelated. In fact, the first order autocorrelation is above 0.5 so the MA(1) model is inappropriate. In addition, a lot of lags seems to be significant and the ACF seems to decay geometrically. Therefore, an AR model should be preferred to model the series. We shall see that the ARMA model is able to eliminate the autocorrelation while the MA(q) process is not.

Estimation of a MA(q) model using the ML approach

To estimate a MA(q) process, we need to code the log-likelihood function of the model.

```
## Log likelihood function of a MA(q)
log_likelihood_MA = function(beta_and_log_sigma_sq,y){
  dim_param = length(beta_and_log_sigma_sq)
  sigma_sq = exp(beta_and_log_sigma_sq[dim_param]) # To avoid negative variance.
  mu = beta_and_log_sigma_sq[1]
  nb_q = dim_param-2
  theta_lags = beta_and_log_sigma_sq[2:(dim_param-1)]
  if(sigma_sq<0){
    LL = -10e8
  }else{
    LL = 0
    T = length(y)
    epsilon = numeric(T)


    #LL = -0.5*nb_q*log(2*pi*sigma_sq) - sum((y[1:nb_q]-mu)^2/(2*sigma_sq))
    epsilon[1:nb_q] = y[1:nb_q]-mu
    for(t in (nb_q+1):T){
      epsilon[t] = y[t] - mu - sum(theta_lags*epsilon[(t-1):(t-nb_q)])
      LL = LL -0.5*log(2*pi*sigma_sq) - (epsilon[t]^2)/(2*sigma_sq)
    }
  }
  return(-1*LL)
}
```

We can use an optimization function to maximize the log-likelihood function. The model is still of low dimension. We can randomly draw the starting values and the optimization function is likely to find the global maximum.

```
nb_q = 2
theta_MA = rnorm(nb_q+1)*0.4
beta_and_log_sigma_sq0=c(theta_MA,log(0.2))
MLE_MA = optim(beta_and_log_sigma_sq0,log_likelihood_MA,y=y,method=c('BFGS'),hessian=TRUE)

df_beta_all = data.frame(c(MLE_MA$par[1:(nb_q+1)],exp(MLE_MA$par[nb_q+2])))
rownames(df_beta_all) = c("mu","theta1","theta2","var")
colnames(df_beta_all) = c("MLE")
(df_beta_all)
```

```
##              MLE
```

```
## mu     2.5960691
## theta1 0.5498820
## theta2 0.2097645
## var    2.5588348
```

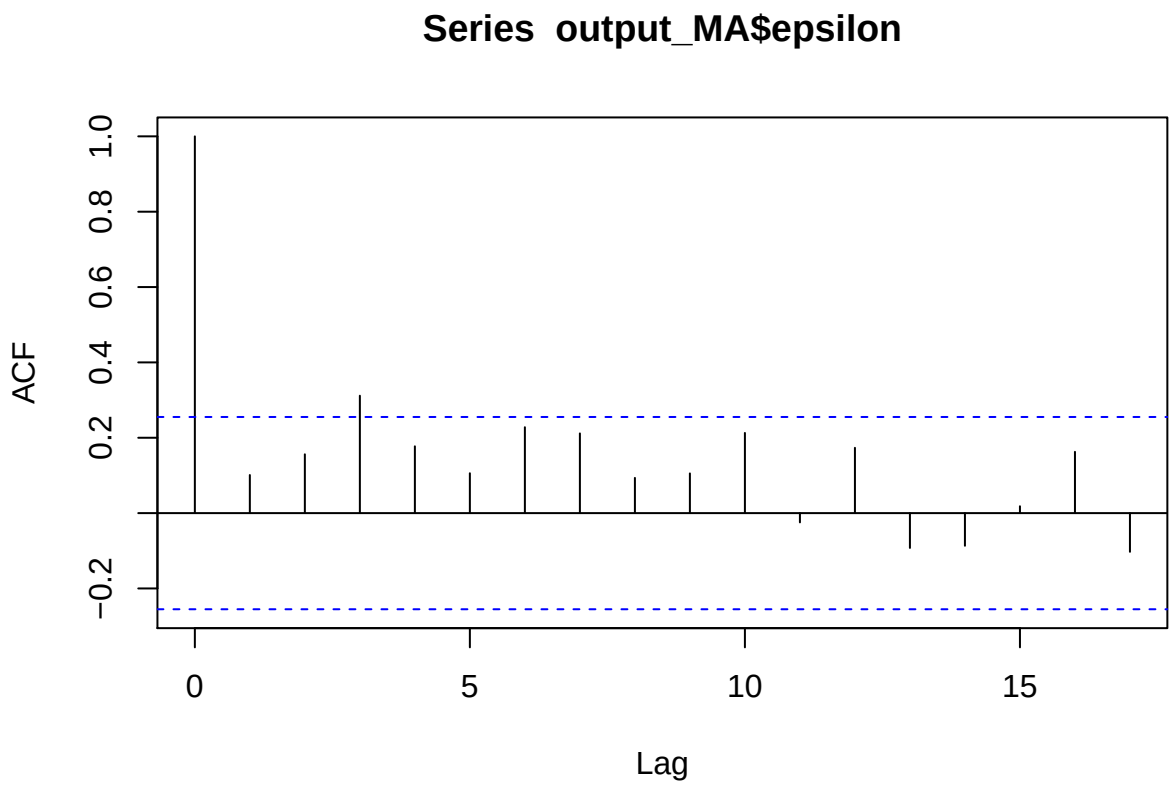
We shall test the autocorrelation of the residuals.

```
error_MA = function(beta_and_log_sigma_sq,y){ ## Function to get the residuals and other MA quantities.
  dim_param = length(beta_and_log_sigma_sq)
  sigma_sq = exp(beta_and_log_sigma_sq[dim_param]) # To avoid negative variance.
  mu = beta_and_log_sigma_sq[1]
  nb_q = dim_param-2
  theta_lags = beta_and_log_sigma_sq[2:(dim_param-1)]
  if(sigma_sq<0){
    LL = -10e8
  }else{
    LL = 0
    T = length(y)
    epsilon = numeric(T)


    #LL = -0.5*nb_q*log(2*pi*sigma_sq) - sum((y[1:nb_q]-mu)^2/(2*sigma_sq))
    epsilon[1:nb_q] = y[1:nb_q]-mu
    for(t in (nb_q+1):T){
      epsilon[t] = y[t] - mu - sum(theta_lags*epsilon[(t-1):(t-nb_q)])
      LL = LL -0.5*log(2*pi*sigma_sq) - (epsilon[t]^2)/(2*sigma_sq)
    }
  }
  output = list("epsilon"=epsilon,"LL"=LL,"param"=c(mu,theta_lags),"variance"=sigma_sq)
  return(output)
}

output_MA = error_MA(MLE_MA$par,y)

acf(output_MA$epsilon)
```




```
BP_test = Box.test(output_MA$epsilon,lag = 10) ## R function
```

We observe that the residuals exhibit some autocorrelation. Performing the BP test shows that we reject the NULL of no autocorrelation at a level of 5% since the p-value amounts to 0.0307903.

We end this section on the MA model by computing the standard errors of the parameters.

```
## Comparison of the standard errors of the two estimators
inv_hess = solve(MLE_MA$hessian)
std_beta = sqrt(diag(inv_hess))

df_std_all = data.frame(cbind(MLE_MA$par,std_beta))
rownames(df_std_all) = c("mu","theta1","theta2","log var")
colnames(df_std_all) = c("Estimates","std errors")
(df_std_all)
```

```
##         Estimates std errors
## mu      2.5960691 0.37486480
## theta1  0.5498820 0.11963317
## theta2  0.2097645 0.09976146
## log var 0.9395520 0.18731929
```

If we wanted to know the standard error of the variance (and not of the log of the variance), we could apply the delta method.

Estimation of an ARMA(p,q) model using the maximum likelihood approach

For estimating the ARMA model using the ML approach, we need to write down the conditional likelihood function (see function *log_likelihood_ARMA*).

```
log_likelihood_ARMA = function(beta_and_log_sigma_sq,y,nb_p=1,nb_q=2){
  dim_param = length(beta_and_log_sigma_sq)
  if(dim_param!=2+nb_p+nb_q){
    stop("Dimension problem")
  }
  sigma_sq = exp(beta_and_log_sigma_sq[dim_param])
  beta_0 = beta_and_log_sigma_sq[1]
  beta_AR = beta_and_log_sigma_sq[2:(nb_p+1)]
  beta_MA = beta_and_log_sigma_sq[(nb_p+2):(dim_param-1)]

  if(sigma_sq<0){
    LL = -10e8
  }else{
    LL = 0
    T = length(y)
    start = max(nb_p,nb_q)
    unc_mean = mean(y)
    epsilon = numeric(T)
    epsilon[1:start] = y[1:start] - unc_mean

    for(t in (start+1):T){
      epsilon[t] = y[t] - beta_0 - sum(beta_AR*y[(t-1):(t-nb_p)]) - sum(beta_MA*epsilon[(t-1):(t-nb_q)])
      LL = LL -0.5*log(2*pi*sigma_sq) - (epsilon[t]^2)/(2*sigma_sq)
    }
  }
  return(-1*LL)
}
```

We now maximize the log-likelihood function of the ARMA model. We choose an ARMA(1,2) model.

```
nb_p = 1
nb_q = 2
beta_and_log_sigma_sq=c(rnorm(1+nb_p+nb_q)*0.1,log(0.2))
MLE_ARMA = optim(beta_and_log_sigma_sq,log_likelihood_ARMA,y=y,nb_p=nb_p,nb_q=nb_q,method=c('BFGS'),hes

df_beta_all = data.frame(c(MLE_ARMA$par[1:(nb_p+nb_q+1)],exp(MLE_ARMA$par[nb_p+nb_q+2])))
rownames(df_beta_all) = c("beta0","beta1","theta1","theta2","var")
colnames(df_beta_all) = c("MLE")
(df_beta_all)
```

```
##               MLE
## beta0   0.1891393
## beta1   0.9092459
## theta1 -0.4148013
## theta2 -0.1432783
## var     2.2049756
```

We finally check the autocorrelation in the residuals. For that, we first create a function to get the residuals of the ARMA model and then we compute the ACF as well as the Box-Pierce test.

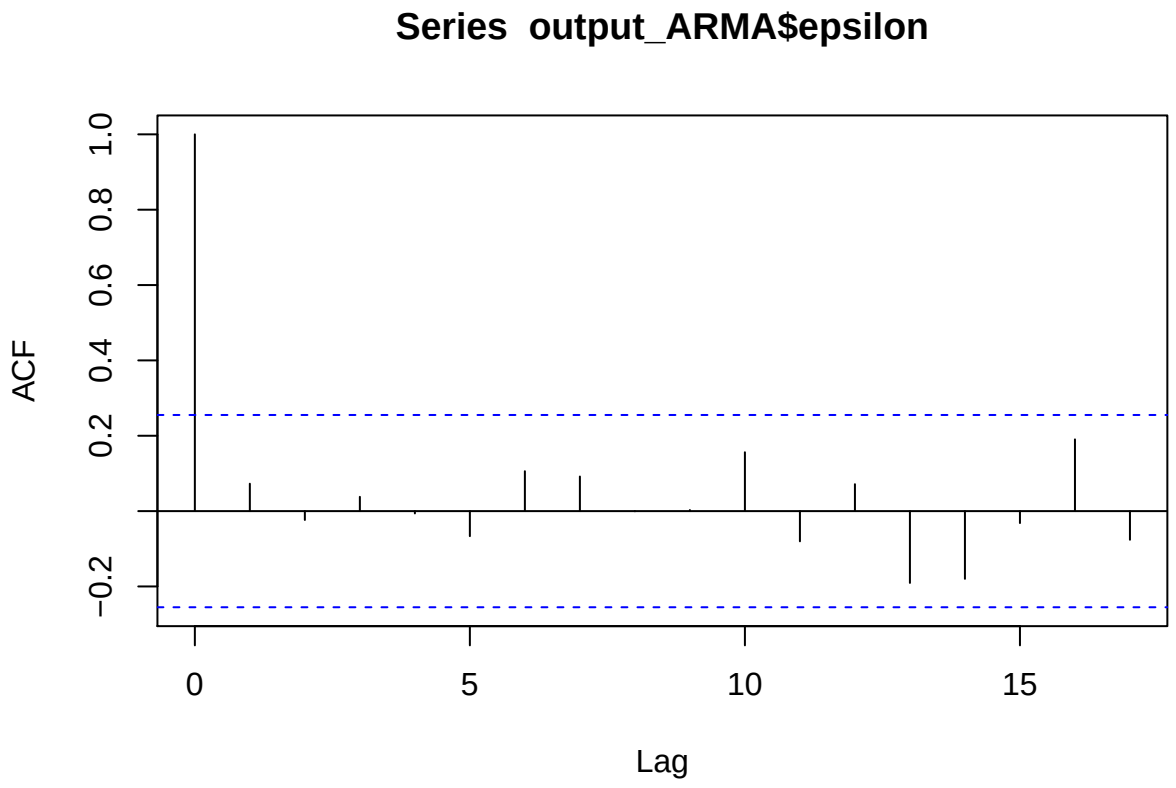
```
epsilon_ARMA = function(beta_and_log_sigma_sq,y,nb_p=1,nb_q=2){
  dim_param = length(beta_and_log_sigma_sq)
```

```
  if(dim_param!=2+nb_p+nb_q){
    stop("Dimension problem")
  }
  sigma_sq = exp(beta_and_log_sigma_sq[dim_param])
  beta_0 = beta_and_log_sigma_sq[1]
  beta_AR = beta_and_log_sigma_sq[2:(nb_p+1)]
  beta_MA = beta_and_log_sigma_sq[(nb_p+2):(dim_param-1)]

  if(sigma_sq<0){
    LL = -10e8
  }else{
    LL = 0
    T = length(y)
    start = max(nb_p,nb_q)
    unc_mean = mean(y)
    epsilon = numeric(T)
    epsilon[1:start] = y[1:start] - unc_mean

    for(t in (start+1):T){
      epsilon[t] = y[t] - beta_0 - sum(beta_AR*y[(t-1):(t-nb_p)]) - sum(beta_MA*epsilon[(t-1):(t-nb_q)])
      LL = LL -0.5*log(2*pi*sigma_sq) - (epsilon[t]^2)/(2*sigma_sq)
    }
  }
  output = list("epsilon"=epsilon,"LL"=LL,"param"=c(beta_0,beta_AR,beta_MA),"variance"=sigma_sq)
  return(output)
}

output_ARMA = epsilon_ARMA(MLE_ARMA$par,y,nb_p,nb_q)
acf(output_ARMA$epsilon)
```




```
BP_test = Box.test(output_ARMA$epsilon,lag = 10) ## R function
```

We observe that the we get rid of the autocorrelation. The Box-Pierce test confirms that result since the p-value of the test amounts to 0.9736572.

Likelihood ratio test

We end this exercise with a likelihood ratio test. In fact, the MA(2) process is a constrained version of the ARMA(1,2) model. In particular, if the AR coefficient is set to zero, the ARMA model boils down to the MA(2) process. Therefore, we can use the likelihood ratio test or the Wald test to test the following hypothesis:

$$H_0 : \beta_{\text{AR}} = 0 \qquad \text{versus} \qquad H_1 : \beta_{\text{AR}} \neq 0$$

Since there is only one linear constraint, the Wald test is equivalent to a standard significant test. We carry out the test by computing the standard error of the AR coefficient and by testing if it is significant.

```
## Z test and Wald test
inv_hess = solve(MLE_ARMA$hessian)
std_beta = sqrt(diag(inv_hess))

Z_test = abs(MLE_ARMA$par[2]/std_beta[2])
Wald_test = (MLE_ARMA$par[2]/std_beta[2])^2
Wald_crit = qchisq(0.95,1)
```

First, we observe that the standard Z test gives a statistic of Z_test=9.8563614 which is way above any standard threshold (i.e. 1.96 for a 5% significant test). As expected, the Wald test also rejects the Null hypothesis since the test statistic is very large: 97.1478599 compared to the chi-square threshold, 3.8414588.

We now perform a likelihood ratio test which compares both likelihood functions evaluated at the maximum likelihood estimates. The resulting statistic is distributed according to a chi-square distribution with degree of freedom equal to the number of constraints (i.e. =1 in this case).

```
## LR test
LL_ARMA = -MLE_ARMA$value
LL_MA = -MLE_MA$value
LR = 2*(LL_ARMA-LL_MA)
LR_crit = qchisq(0.95,1)
print(paste("The LR statistic amounts to ",LR, " which is above the 95% threshold: ",LR_crit))
```

```
## [1] "The LR statistic amounts to  8.48195277783955  which is above the 95% threshold:  3.84145882069
```