Ch5-ARMA

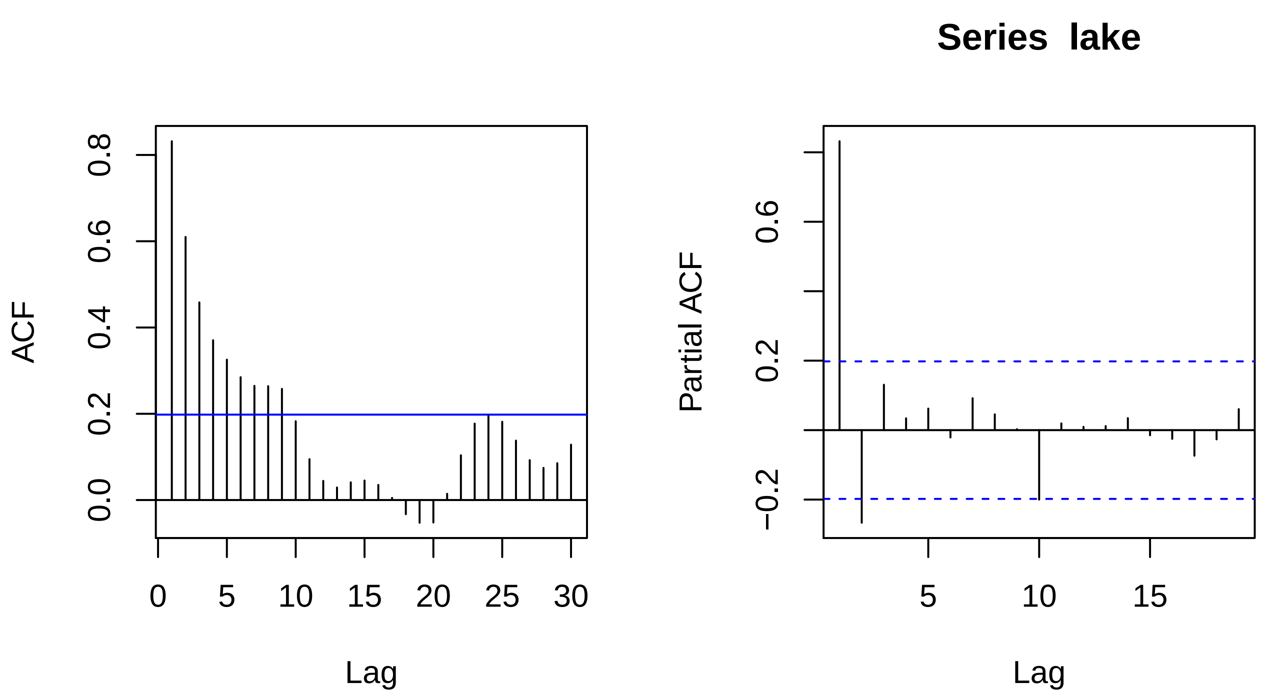
Fitting ARMA model to lake Huron data

This example shows how to estimate ARMA parameters using R. Recall lake Huron data given by

```
source("TS-library.R")
lake = c(10.38,11.86,10.97,10.8,9.79,
10.39,10.42,10.82,11.4,11.32,11.44,11.68,11.17,
10.53,10.01,9.91,9.14,9.16,9.55,9.67,8.44,
8.24,9.1,9.09,9.35,8.82,9.32,9.01,9,9.8,9.83,9.72,9.89,
10.01,9.37,8.69,8.19,8.67,9.55,8.92,8.09,
9.37,10.13,10.14,9.51,9.24,8.66,8.86,8.05,
7.79,6.75,6.75,7.82,8.64,10.58,9.48,7.38,
6.9,6.94,6.24,6.84,6.85,6.9,7.79,8.18,
7.51,7.23,8.42,9.61,9.05,9.26,9.22,9.38,
9.1,7.95,8.12,9.75,10.85,10.41,9.96,9.61,
8.76,8.18,7.21,7.13,9.1,8.25,7.91,6.89,
5.96,6.8,7.68,8.38,8.52,9.74,9.31,9.89,9.96)
```

From the ACF/PACF plots,

```
par(mfrow=c(1,2))
acf2(lake, lag=30);
pacf(lake)
```



AR(2) seems plausible.

We first consider Yule-Walker estimation of AR(2) model. Note by default, `ar.yw()` includes a constant (non-zero mean model).

```
ar.yw(lake, aic=FALSE, order.max=2)
```

```
##
## Call:
## ar.yw.default(x = lake, aic = FALSE, order.max = 2)
##
## Coefficients:
##       1        2
##  1.0538  -0.2668
##
## Order selected 2  sigma^2 estimated as  0.5075
```

```
ar.yw(lake, aic=FALSE, order.max=2, demean=FALSE)
```

```
##
## Call:
## ar.yw.default(x = lake, aic = FALSE, order.max = 2, demean = FALSE)
##
## Coefficients:
##       1        2
##  1.0747  -0.0923
##
## Order selected 2  sigma^2 estimated as  2.7
```

If you want to apply MLE to estimate parameters

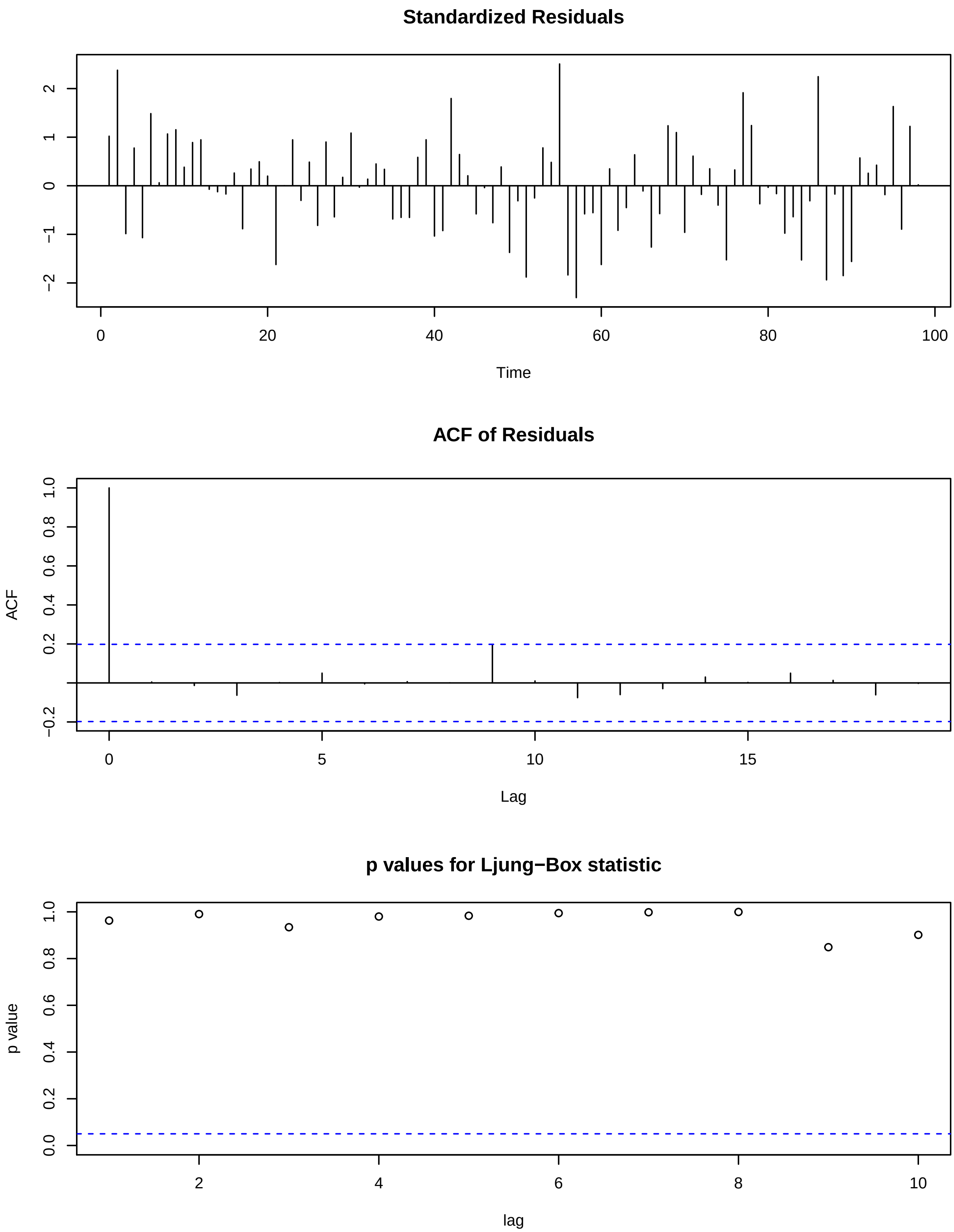
$$\Phi(B)(X_t - \mu) = \Theta(B)Z_t,$$

do the following

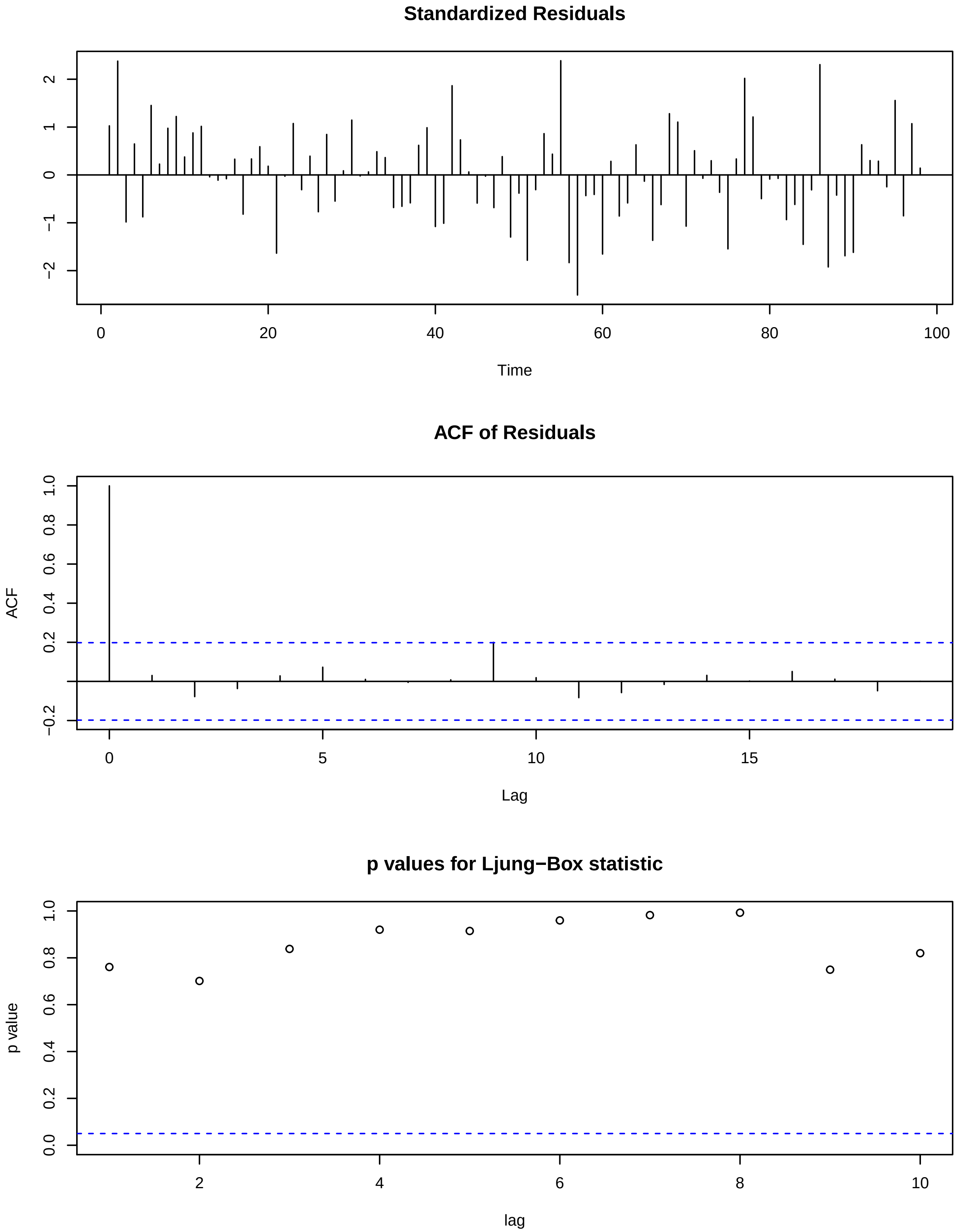
```
ar11.out = arima(lake, order=c(1,0,1), method = c("CSS-ML"), include.mean = TRUE)
ar1.out = arima(lake, order=c(1,0,0))
ar2.out = arima(lake, order=c(2,0,0))
ma1.out = arima(lake, order=c(0,0,1))
```

After fit ARMA, we need to check the gooness of fit using residuals.

```
tsdiag(ar11.out)
```



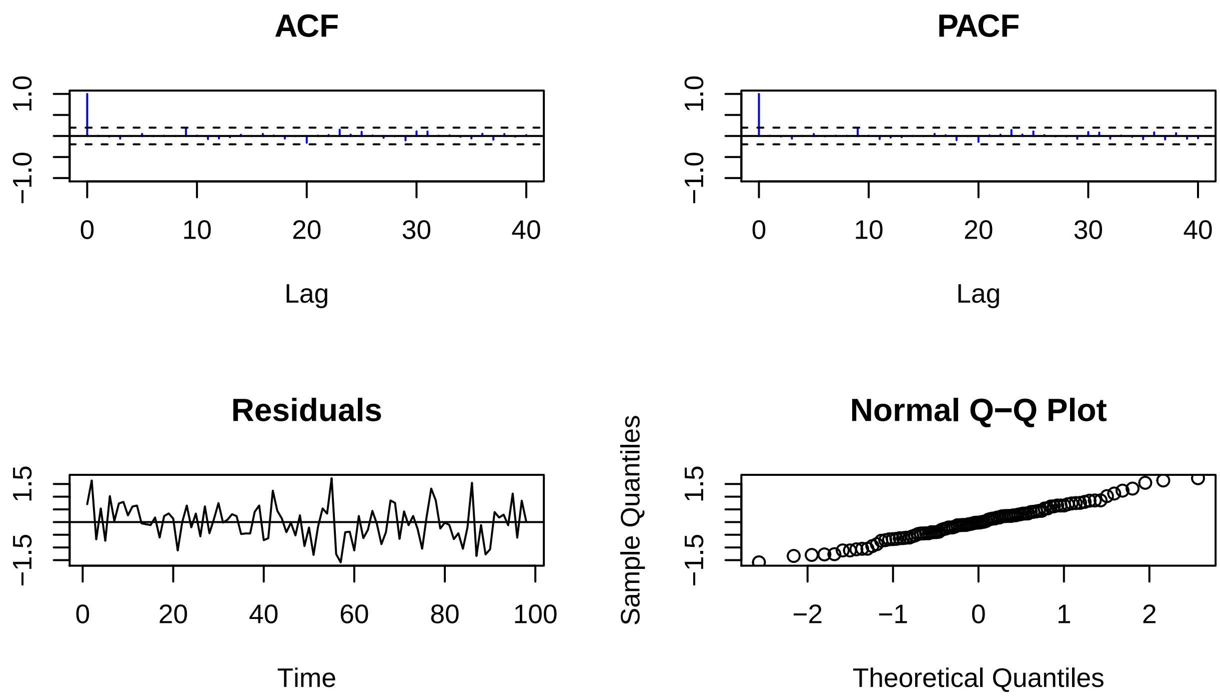

```r
tsdiag(ar2.out)
```



Test of randomness gives

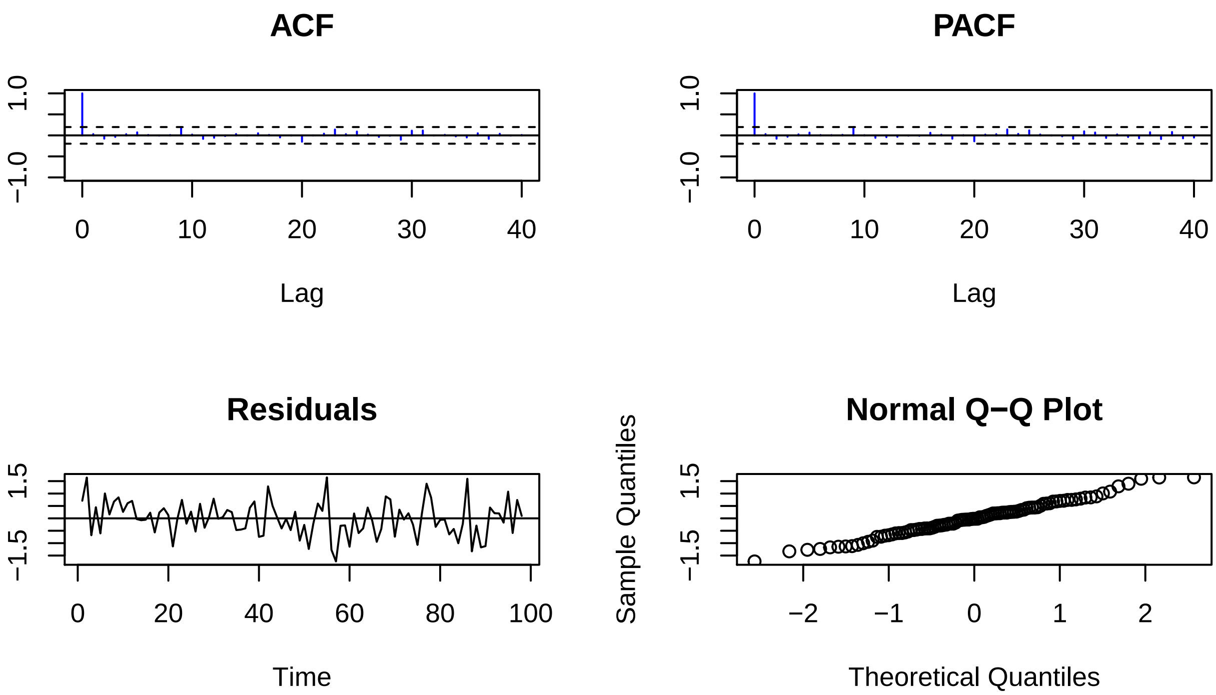
```r
library(itsmr)
test(resid(ar11.out))
```

```
## Null hypothesis: Residuals are iid noise.
## Test                        Distribution Statistic   p-value
## Ljung-Box Q                Q ~ chisq(20)     10.14    0.9656
## McLeod-Li Q                Q ~ chisq(20)     16.43    0.6899
## Turning points T      (T-64)/4.1 ~ N(0,1)        69    0.2266
## Diff signs S        (S-48.5)/2.9 ~ N(0,1)        50    0.6015
## Rank P          (P-2376.5)/162.9 ~ N(0,1)      2083    0.0716
```




```r
test(resid(ar2.out))
```

```
## Null hypothesis: Residuals are iid noise.
## Test                        Distribution Statistic   p-value
## Ljung-Box Q                Q ~ chisq(20)     10.67    0.9544
## McLeod-Li Q                Q ~ chisq(20)     17.49    0.6211
## Turning points T      (T-64)/4.1 ~ N(0,1)        65    0.8089
## Diff signs S        (S-48.5)/2.9 ~ N(0,1)        52     0.223
## Rank P          (P-2376.5)/162.9 ~ N(0,1)      2051    0.0457 *
```



Both model seems plasible. To fine the best model in terms of information criteria,

```
AICC = BIC = AIC = P = Q = NULL;
pmax=3; qmax=3;
n = length(lake);
for(p  in 0:qmax){
    for(q in 0:qmax){
    fit = arima(lake, order=c(p, 0, q), include.mean=TRUE);
    m = p+q+2;
    AIC = c(AIC, -2*fit$loglik + 2*m);
    AICC = c(AICC, -2*fit$loglik + 2*m*n/(n-m-1));
    BIC = c(BIC, -2*fit$loglik + m*log(n));
    P = c(P, p);
    Q = c(Q, q);
}
}
```

```
## Warning in arima(lake, order = c(p, 0, q), include.mean = TRUE): possible
## convergence problem: optim gave code = 1
```

```
id1 = which.min(AICC);
id2 = which.min(BIC);
c(P[id1], Q[id1])
```

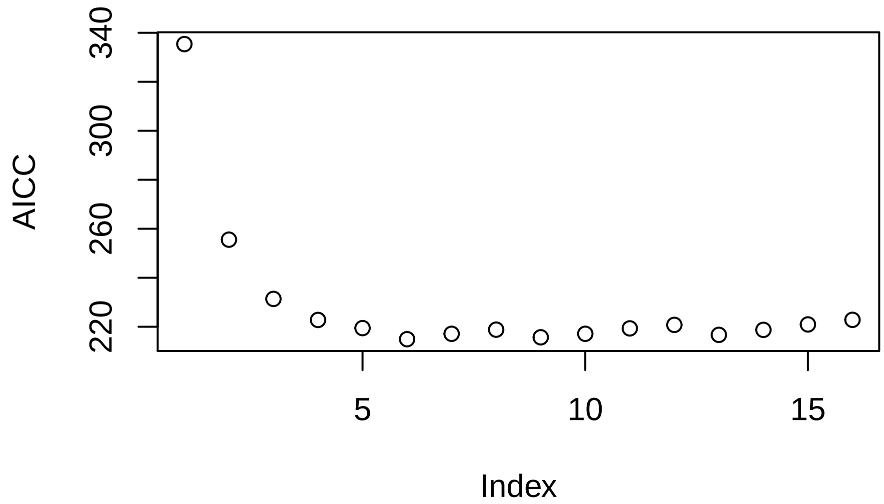
```
## [1] 1 1
```

```
c(P[id2], Q[id2])
```

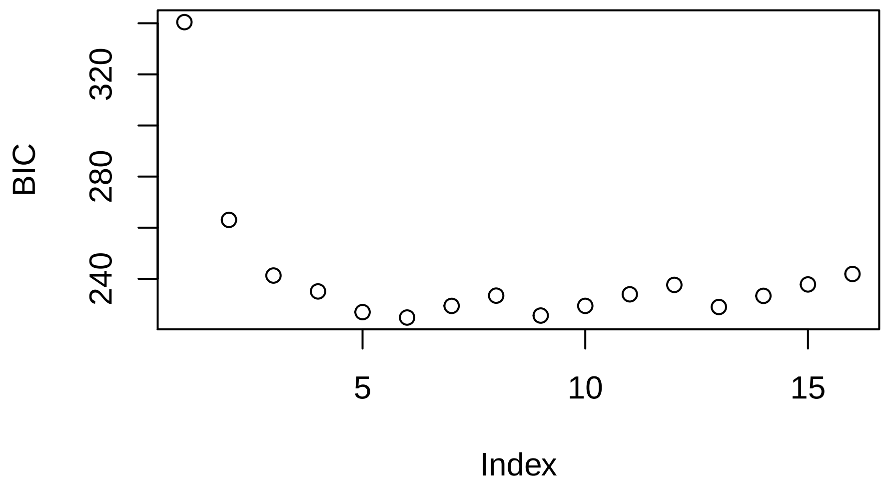
```
## [1] 1 1
```

```
# Both gives ARMA(1,1) as the best model

plot(AICC);
```




```
plot(BIC);
```



Or simply, you can apply

```
library(forecast)
```

```
##
## Attaching package: 'forecast'

## The following object is masked from 'package:itsmr':
##
##     forecast
```

```
fit=auto.arima(lake, d=0);
summary(fit)
```

```
## Series: lake
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ma1    mean
##       0.7449  0.3206  9.0555
## s.e.  0.0777  0.1135  0.3501
##
## sigma^2 estimated as 0.4899:  log likelihood=-103.25
## AIC=214.49   AICc=214.92   BIC=224.83
##
## Training set error measures:
##                        ME      RMSE       MAE       MPE     MAPE      MASE
## Training set -0.008976993 0.6891588 0.5495562 -0.741374 6.307826 0.9385026
##                     ACF1
## Training set 0.004670692
```

We can also perform the test whether all coefficients are away from zero

$$H_0 : \phi_i = 0 \quad vs \quad H_1 : \phi_i \neq 0$$

```
2*(1-pnorm(fit$coef/(sqrt(diag(fit$var.coef)))))
```

```
##         ar1         ma1   intercept
## 0.000000000 0.004745202 0.000000000
```

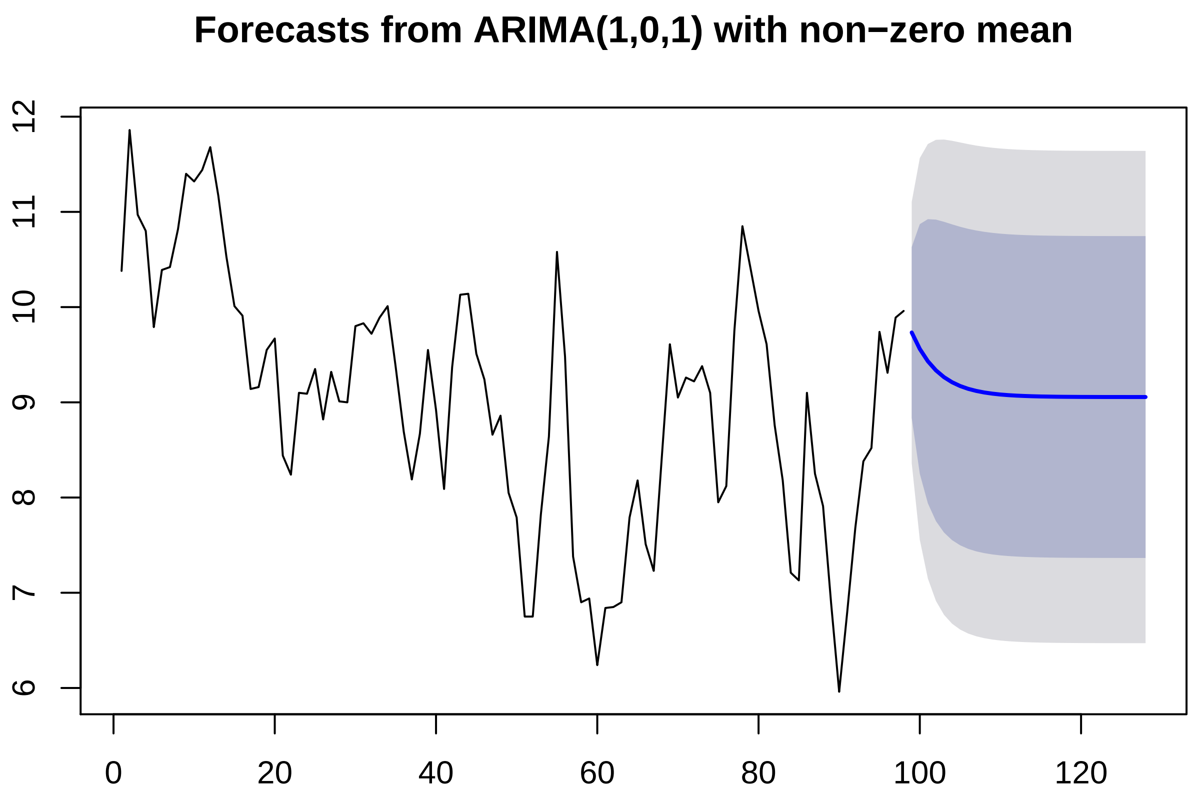
Therefore, the The best model is ARMA(1,1). Now we will do forecasting and provide 95% prediction interval.

```
detach("package:itsmr")
library(forecast)
forecast(fit, 30)
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
##  99       9.733373 8.836344 10.63040 8.361485 11.10526
## 100       9.560436 8.249647 10.87122 7.555758 11.56511
## 101       9.431615 7.939956 10.92327 7.150319 11.71291
## 102       9.335656 7.752526 10.91879 6.914467 11.75685
## 103       9.264177 7.632502 10.89585 6.768745 11.75961
## 104       9.210932 7.552934 10.86893 6.675242 11.74662
## 105       9.171270 7.498844 10.84370 6.613515 11.72902
## 106       9.141726 7.461348 10.82210 6.571810 11.71164
## 107       9.119718 7.434944 10.80449 6.543079 11.69636
## 108       9.103325 7.416116 10.79053 6.522962 11.68369
## 109       9.091113 7.402556 10.77967 6.508687 11.67354
```

```
## 110       9.082017 7.392711 10.77132 6.498447 11.66559
## 111       9.075241 7.385520 10.76496 6.491036 11.65945
## 112       9.070194 7.380243 10.76014 6.485637 11.65475
## 113       9.066434 7.376355 10.75651 6.481682 11.65119
## 114       9.063633 7.373484 10.75378 6.478773 11.64849
## 115       9.061547 7.371358 10.75174 6.476626 11.64647
## 116       9.059993 7.369783 10.75020 6.475039 11.64495
## 117       9.058836 7.368613 10.74906 6.473863 11.64381
## 118       9.057973 7.367744 10.74820 6.472990 11.64296
## 119       9.057331 7.367098 10.74756 6.472342 11.64232
## 120       9.056853 7.366617 10.74709 6.471861 11.64184
## 121       9.056496 7.366260 10.74673 6.471503 11.64149
## 122       9.056231 7.365994 10.74647 6.471236 11.64123
## 123       9.056033 7.365796 10.74627 6.471038 11.64103
## 124       9.055886 7.365648 10.74612 6.470890 11.64088
## 125       9.055776 7.365538 10.74601 6.470780 11.64077
## 126       9.055694 7.365456 10.74593 6.470699 11.64069
## 127       9.055633 7.365396 10.74587 6.470638 11.64063
## 128       9.055588 7.365350 10.74583 6.470592 11.64058
```

```r
plot(forecast(fit, 30))
```



Out-of-sample forecasting error

Other than information criteria, out-of-sample forecasting error is widely used to determine the best model. In short, our best model is the one shows the smallest forecasting error. For example, 1-step out-of-sample forecasting error is given by

$$E(X_{t+1} - \widehat{X}_{t+1})^2.$$

We will estimate this quantity by using cross-validation. In time series context, to preserve dependence structreu, we divde entire sample data of size $N + m$ into two:

$$\text{Training set} \quad X_1, \ldots, X_N$$

$$\text{Test set} \quad X_{N+1}, \ldots, X_{N+m}$$

Then, we can approximate 1-step MSPE as

$$\frac{1}{m} \sum_{t=N}^{N+m-1} (X_{t+1} - \widehat{X}_{t+1})^2.$$

Here are the procedure to calculate 1-step out-of-step forecasting error

1. Step1 : Find the best model using Training set $X_1, \ldots, X_N$
2. Step2 : For $t = N, \ldots, N + m - 1$ (corresponding to Test set): Find the 1-steap ahead estimate X_{t+1} based on observation $X_1, \ldots, X_t$ based on the model parameters estimated in Step1. Or alternatively,
3. Step2' : We can re-estimate the model parameters in Step 2 using sample $X_1, \ldots, X_t$.

In-class exercise

Implement and evalulate 1-step-out of sample forecating error for mysterious.txt in HW5 for the last 30 observations. Which model would you select for this dataset?

```
## Out of sample forecasting practice
myst = scan("mysterious.txt");
library(forecast)
auto.arima(myst, d=0);
```

```
## Series: myst
## ARIMA(3,0,1) with zero mean
##
## Coefficients:
##           ar1     ar2      ar3     ma1
##       -0.1305  0.3945  -0.3174  0.8219
## s.e.   0.1078  0.0740   0.0683  0.0967
##
## sigma^2 estimated as 0.133:  log likelihood=-80.37
## AIC=170.74   AICc=171.05   BIC=187.23
```

```
m=30; n = length(myst);
N = n-m;
testindex = (N+1):n;
p=3; q=1;

err = numeric(m);
for(i in 1:m){
```

```
  trainindex = 1:(N+i-1);
  fit = arima(myst[trainindex], order=c(p,0,q), include.mean=FALSE);
  Xhat =  forecast(fit, h=1)$mean;
  err[i] = (myst[N+i] - Xhat)^2;
}
mean(err)
```

```
## [1] 0.1104433
```

```
## In-terms of out-of-sample forecasting ARMA(1,1) is better...
p=1; q=1;
err = numeric(m);
for(i in 1:m){
  trainindex = 1:(N+i-1);
  fit = arima(myst[trainindex], order=c(p,0,q), include.mean=FALSE);
  Xhat =  forecast(fit, h=1)$mean;
  err[i] = (myst[N+i] - Xhat)^2;
}
mean(err)
```

```
## [1] 0.09931474
```