#Transform the time series using (a) “classical decomposition” (decompose to trend and/or seasonal components)

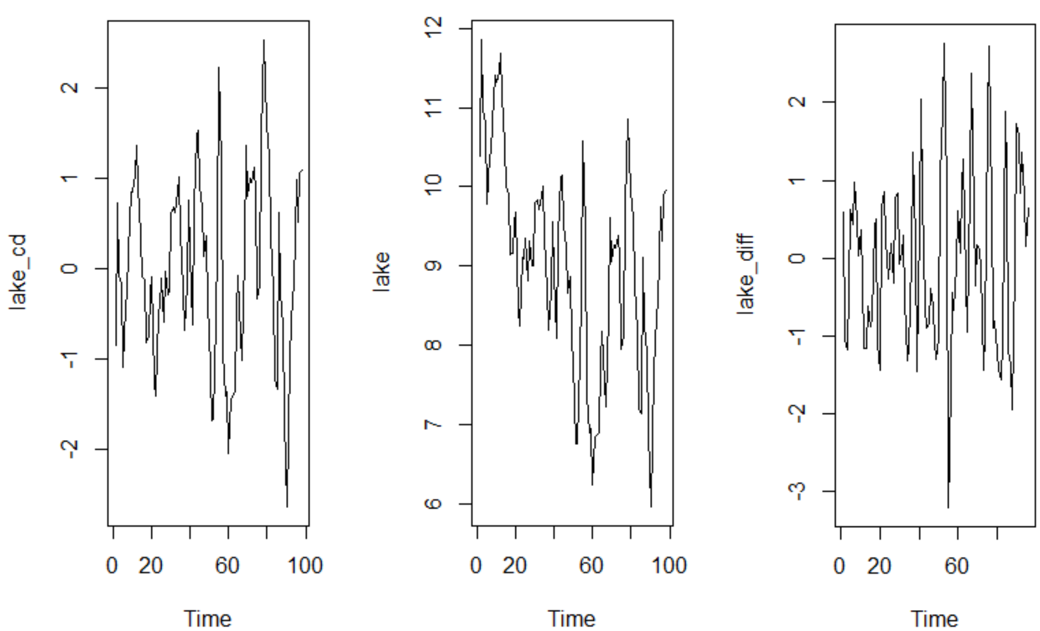
```
> library(astsa)
> library(itsmr)
> #Transform the time series using (a) “classical decomposition” (decompose to tre
nd and/or seasonal components)
> tre = trend(lake,2)
> lake_cd = ts(lake - tre)
>
```

#Transform the time series using (b) “differencing”

```
> #Transform the time series using (b) “differencing”
> lake_diff = diff(lake, lag = 2)
```

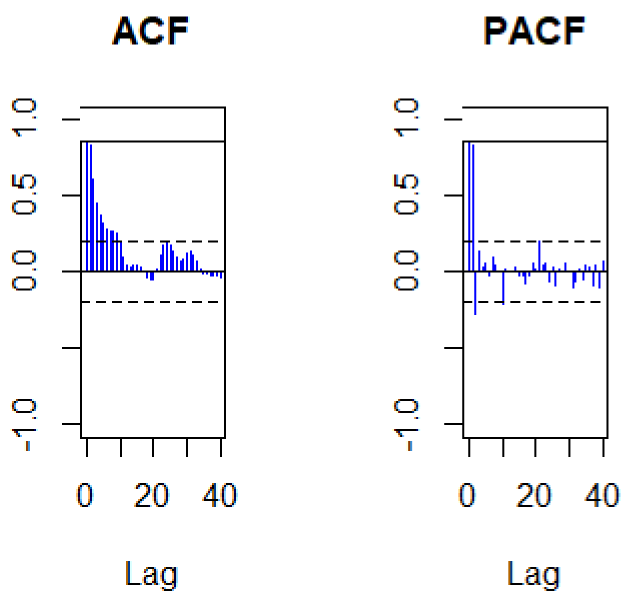
#Plot the original time series and the transformed time series (obtained by methods (a) and (b) above).

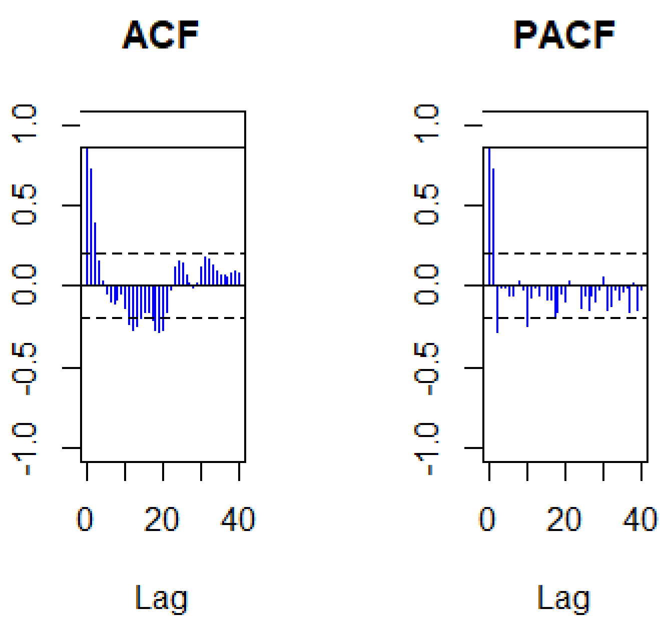
```
> #Plot the original time series and the transformed time series (obtained by meth
ods (a) and (b) above).
> plot.ts(lake)
> plot.ts(lake_cd)
> plot.ts(lake_diff)
>
```

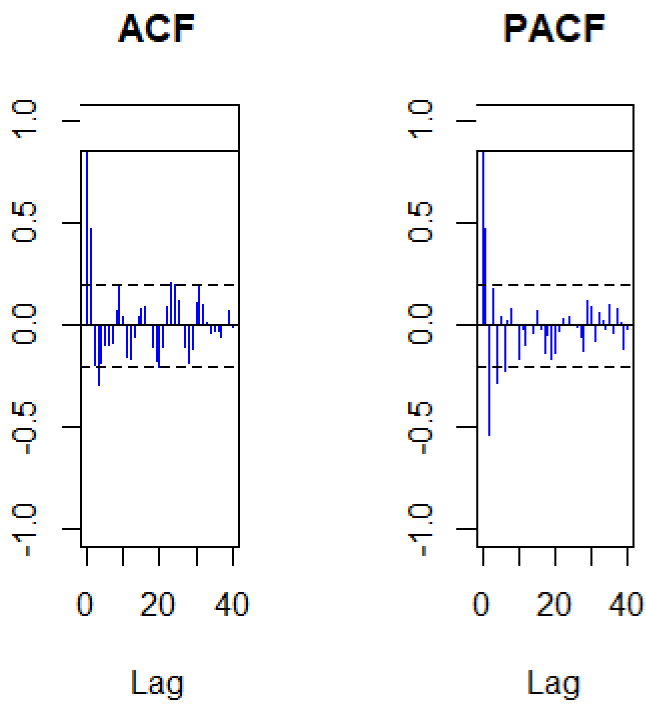



#Plot the autocorrelations (ACF) and the partial autocorrelations (PACF) of the transformed time series (obtained by methods (a) and (b) above).

```
> #Plot the autocorrelations (ACF) and the partial autocorrelations (PACF) of the
 transformed time series (obtained by methods (a) and (b) above).
> plota(lake)
> plota(lake_cd)
> plota(lake_diff)
>
```




ACF
PACF
1.0
0.5
0.0
-0.5
-1.0
0
20
40
Lag

```
> lake_burg = burg(lake, 3)
> lake_burg
$phi
[1]  1.0726245 -0.3634421  0.1127770

$theta
[1] 0

$sigma2
[1] 0.4727809

$aicc
[1] 214.5074

$se.phi
[1] 0.09902332 0.14141246 0.09902332

$se.theta
[1] 0

> lakecd_burg = burg(lake_cd, 2)
> lakecd_burg
$phi
[1]  0.9497421 -0.3044418

$theta
[1] 0

$sigma2
[1] 0.4339304

$aicc
[1] 203.4997

$se.phi
[1] 0.09521511 0.09521511

$se.theta
[1] 0

> lakediff_burg = burg(lake_diff, 3)
> lakediff_burg
$phi
[1]  0.8540729 -0.7024455  0.2064804

$theta
[1] 0

$sigma2
[1] 0.6429137

$aicc
[1] 239.5773

$se.phi
[1] 0.09876044 0.11031833 0.09876044

$se.theta
```

```
$se.theta
[1] 0

> lake_yw = yw(lake, 2)
> lake_yw
$phi
[1]  1.0538249 -0.2667516

$theta
[1] 0

$sigma2
[1] 0.4790562

$aicc
[1] 213.5709

$se.phi
[1] 0.097355 0.097355

$se.theta
[1] 0

> lakecd_yw = yw(lake_cd, 2)
> lakecd_yw
$phi
[1]  0.9206804 -0.2765911

$theta
[1] 0

$sigma2
[1] 0.4347255

$aicc
[1] 203.6227

$se.phi
[1] 0.09707442 0.09707442

$se.theta
[1] 0
```

```
> lakediff_yw = yw(lake_diff, 3)
> lakediff_yw
$phi
[1]  0.8297486 -0.6700653  0.1825892

$theta
[1] 0

$sigma2
[1] 0.6439204

$aicc
[1] 239.6562

$se.phi
[1] 0.1003463 0.1120898 0.1003463

$se.theta
[1] 0

> lake_arma = arma(lake,1,1)
> lake_arma
$phi
[1] 0.7448993

$theta
[1] 0.3205891

$sigma2
[1] 0.4750447

$aicc
[1] 212.7675

$se.phi
[1] 0.07765066

$se.theta
[1] 0.1135295

> lakecd_arma = arma(lake_cd,2,0)
> lakecd_arma
$phi
[1]  0.9541393 -0.3074418

$theta
[1] 0

$sigma2
[1] 0.4338805

$aicc
[1] 203.4977

$se.phi
[1] 0.09754420 0.09796247

$se.theta
```

```
$se.theta
[1] 0

> lakediff_arma = arma(lake_diff,2,1)
> lakediff_arma
$phi
[1]  0.1795393 -0.2273004

$theta
[1] 0.9638015

$sigma2
[1] 0.5312353

$aicc
[1] 223.5719

$se.phi
[1] 0.1040764 0.1033121

$se.theta
[1] 0.04509966

> lake_auto = autofit(lake)
> lake_auto
$phi
[1] 0.7448993

$theta
[1] 0.3205891

$sigma2
[1] 0.4750447

$aicc
[1] 212.7675

$se.phi
[1] 0.07765066

$se.theta
[1] 0.1135295

> lakecd_auto = autofit(lake_cd)
Warning messages:
1: In sqrt(v[1:p]) : NaNs produced
2: In sqrt(v[(p + 1):(p + q)]) : NaNs produced
3: In sqrt(v[1:p]) : NaNs produced
4: In sqrt(v[(p + 1):(p + q)]) : NaNs produced
5: In sqrt(v[(p + 1):(p + q)]) : NaNs produced
> lakecd_auto
$phi
[1]  0.9541393 -0.3074418

$theta
[1] 0
```

```
$sigma2
[1] 0.4338805

$aicc
[1] 203.4977

$se.phi
[1] 0.09754420 0.09796247

$se.theta
[1] 0

> lakediff_auto = autofit(lake_diff)
Warning messages:
1: In sqrt(v[1:p]) : NaNs produced
2: In sqrt(v[(p + 1):(p + q)]) : NaNs produced
3: In sqrt(v[1:p]) : NaNs produced
4: In sqrt(v[(p + 1):(p + q)]) : NaNs produced
5: In sqrt(v[1:p]) : NaNs produced
6: In sqrt(v[(p + 1):(p + q)]) : NaNs produced
> lakediff_auto
$phi
[1] 0.6764564

$theta
[1]  0.3189545 -0.9810337 -0.3379075

$sigma2
[1] 0.4763428

$aicc
[1] 218.1815

$se.phi
[1] 0.0955887

$se.theta
[1] 0.12356409 0.06048707 0.11430098

> |
```

# Identify the optimal model (e.g. by using the AICC criterion).

We identify the optimal model with the help of AICC, the smaller the AIC value, the better the model fit. Here the optimal model is lacecd_arma and lakecd_burg  with AICC 203.4977 .

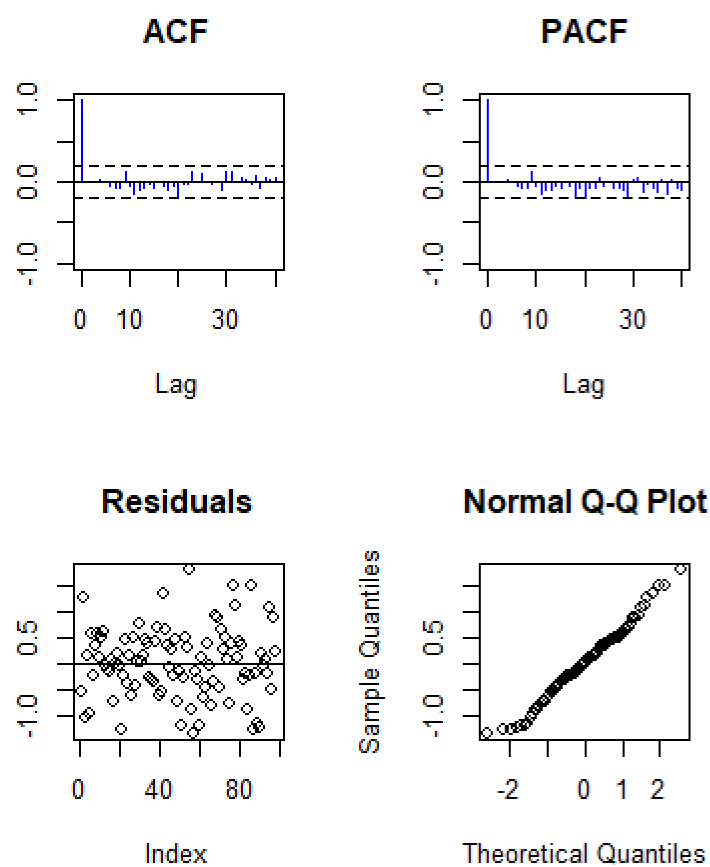
# equation :

$X_t - \phi X_{t-1} = Z_t + \theta Z_{t-1}$ (1)

where $\{Z_t\} \sim WN(0, \sigma^2)$ and $\phi + \theta \neq 0$.

# Check for stationarity of the residuals of the optimal model (by using test() in itsmr).

```
> # Check for stationarity of the residuals of the optimal model (by using test
() in itsmr).
> M = c("trend",2)
> ee = Resid(lake, M, lakecd_arma)
> test(ee)
Null hypothesis: Residuals are iid noise.
Test                          Distribution Statistic   p-value
Ljung-Box Q                   Q ~ chisq(20)     13.17      0.87
McLeod-Li Q                   Q ~ chisq(20)     21.09    0.3918
Turning points T       (T-64)/4.1 ~ N(0,1)        67    0.4682
Diff signs S         (S-48.5)/2.9 ~ N(0,1)        47    0.6015
Rank P           (P-2376.5)/162.9 ~ N(0,1)      2349    0.8659
>
```




# Use “forecast()” (in itsmr) to forecast the future 10 values of the time series.

```
> forecast(lake, M, lakecd_arma)
 Step      Prediction        sqrt(MSE)      Lower Bound      Upper Bound
    1         9.62898        0.6586961         8.337959            10.92
    2        9.307392        0.9104271         7.522988          11.0918
    3        9.118572        0.9932819         7.171775         11.06537
    4        9.054008         1.010495         7.073475         11.03454
    5        9.067661         1.011996         7.084185         11.05114
    6        9.118218         1.012006         7.134723         11.10171
    7        9.180419         1.012229         7.196487         11.16435
    8         9.24286         1.012405         7.258582         11.22714
    9        9.302428         1.012471         7.318022         11.28683
   10        9.359658         1.012484         7.375227         11.34409
```