Estimation in the Time Domain

Al Nosedal
University of Toronto

March 18, 2019

The R function *ar.yw( )*

An R function is available for solving the Yule-Walker equations, *ar.yw( ). Pass any sequence of errors or residuals to the function and the function will solve a sequence of models, AR(1) . . . AR(m) and pick the best model using the criterion of minimizing AIC. It should be noted that the Yule-Walker estimates are not maximum likelihood estimates, so that AIC is only approximated.*

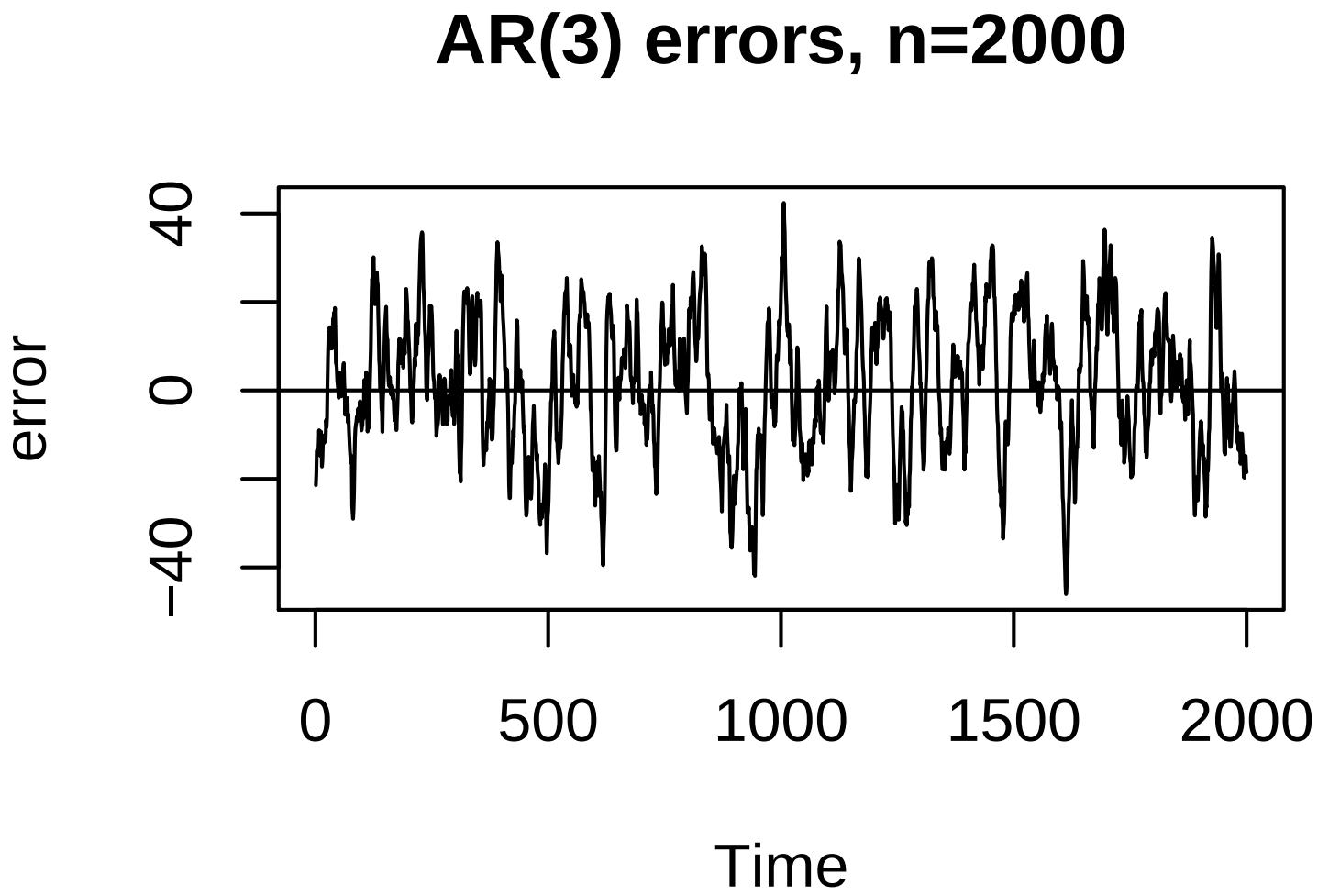
AR(3) Example

```
# Simulating 2000 AR(3) errors;

error<- arima.sim(n=2000,
list(ar=c(1.4,-0.31,-0.126) ),sd=3 );
```

AR(3) Example

```
# Simulating 2000 AR(3) errors;

plot.ts(error,main="AR(3) errors, n=2000");

abline(0,0);
```

AR(3) Example



ar.yw()

```
z<-ar.yw(error);

names(z);

##  [1] "order"         "ar"            "var.pred"      "x.mean"
##  [5] "aic"           "n.used"        "order.max"     "partiala
##  [9] "resid"         "method"        "series"        "frequenc
## [13] "call"          "asy.var.coef"
```

ar.yw()

```
z$aic[1:8];

##            0            1            2            3            4
## 6451.745399  478.121449   30.935561    0.000000    0.548598
##            6            7
##    3.856297    3.383999
```

This is a common format for AIC values. With AIC (or BIC), the values themselves are unique only up to a common constant, and it is the minimum value and differences that are important. Therefore, all values are frequently reported as differences from the minimum value. Using AIC, the AR(3) model is identified as the best.

ar.yw()

```
z$order;

## [1] 3

# The order, m, of the model
# chosen by AIC.
```

ar.yw()

```
z$ar;

## [1]  1.3542153 -0.2681125 -0.1278003

# The estimated values of:
# a_1, a_2, ..., a_m for
# the chosen order.
```

ar.yw()

```
z$var.pred;

## [1] 9.474848

# The estimate for
# variance of w.
```

In some cases, there is a desire to find $\hat{\phi}_1, ..., \hat{\phi}_m$ and/ or σ_w^2 for a different model. In that case, the command
ar.yw(error, aic = FALSE, order.max=m)
will ignore AIC and force the fitting of some specified order m.

For each candidate model, AR(0) to AR(m), the command
ar.yw(error, aic = FALSE, order.max =m)
can be used to fit a specified order. Because information criteria assumes maximum likelihood estimates, a slightly different command can be used to get just those values
ar.mle(error, aic = FALSE, order.max =m)
(we use it to obtain $\hat{\sigma}_w^2$).

BIC estimate

A reasonable estimate would be

$$BIC \approx nln(\hat{\sigma}_w^2) + mln(n),$$

using *ar.mle( ) to estimate* σ_w^2.
($\hat{\sigma}_w^2$ *= variance estimate,* n *= sample size or number of observations in your series,* m *= specified order).*

ar.mle()

```
z.0<-ar.mle(error,aic=FALSE,order.max=0);

z.1<-ar.mle(error,aic=FALSE,order.max=1);

z.2<-ar.mle(error,aic=FALSE,order.max=2);

z.3<-ar.mle(error,aic=FALSE,order.max=3);

z.4<-ar.mle(error,aic=FALSE,order.max=4);

z.5<-ar.mle(error,aic=FALSE,order.max=5);
```

Model	σ_w^2	BIC	Δ BIC
0	238.7725521	1.0951023×10^4	6599.8910059
1	11.6015368	4909.8760483	513.1387714
2	9.0618556	4423.3496369	26.61236
3	8.9081562	4396.7372769	0
4	8.9052815	4403.6926708	6.9553939
5	8.9039331	4410.9907193	14.2534424

The choice of *ar.yw( ) versus* ar.mle( ) for estimation of σ_w^2 was of no impact. However, *ar.mle( ) may run slow or even fail to converge for large sets of data. The results from our table correctly identify the AR(3) model as the model that generated that data.*

A sequence of hypothesis tests

Let $\hat{\phi}_m^m$ be the estimate of ϕ_m for the model AR(m). It is known that $\hat{\phi}_m^m$ has a $N(0, 1/n)$ distribution if the true model has order less than m. Therefore $z \approx \sqrt{n}\hat{\phi}_m^m$ is a standard Normal if the order is less than m and can be treated as a test statistic for

H_0: the order is less than m

vs

H_a: the order is at least m.

A sequence of tests for the order

Model	$\hat{\phi}_m^m$ (mle)	z	p-value
1	0.9758358	43.640703	0
2	-0.4679047	-20.9253332	0
3	-0.1302117	-5.823244	5.771619×10^{-9}
4	-0.0177032	-0.7917105	0.4285295
5	-0.0121136	-0.5417373	0.5879995

It seems clear that the correct order is 3 from these hypothesis tests. Put more precisely, the data suggests that an order of at least 3 is required, but there is no evidence that an order higher than 3 is required.

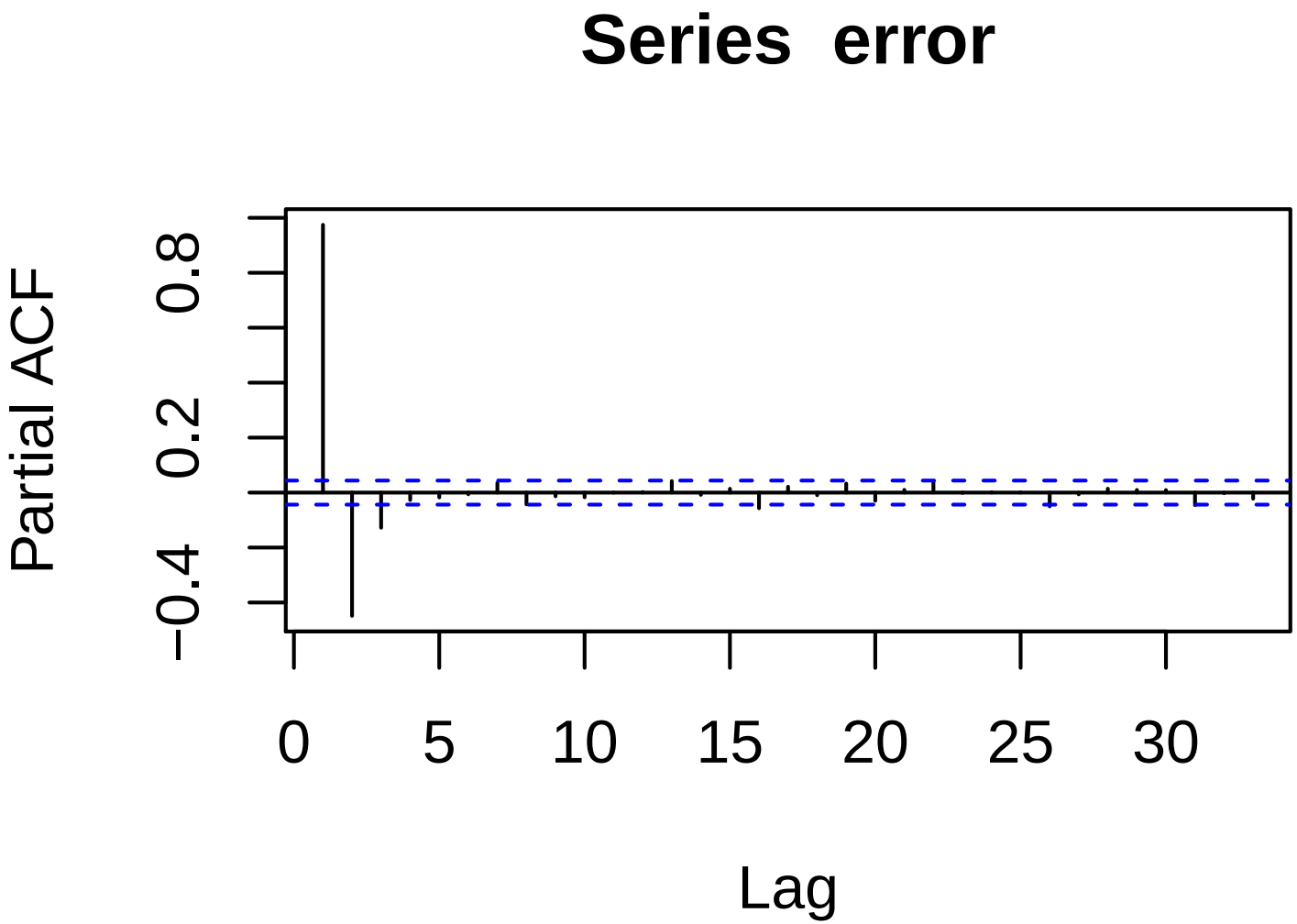
Hypothesis tests presented in a plot

If the values $\hat{\phi}_m^m$ for $m = 1, 2, \ldots$ are plotted versus m, with lines at $\pm 2/\sqrt{n}$, the display becomes a visual representation of the above hypothesis tests. When $\hat{\phi}_m^m$ falls far outside the lines $\pm 2/\sqrt{n}$, there is evidence that the model is at least of order m; when the values begin to fall far inside the lines, the indications are that the order is no higher than the last large spike.

The Partial Autocorrelation Plot

```
pacf(error);
```

The Partial Autocorrelation Plot



Example. The lake data

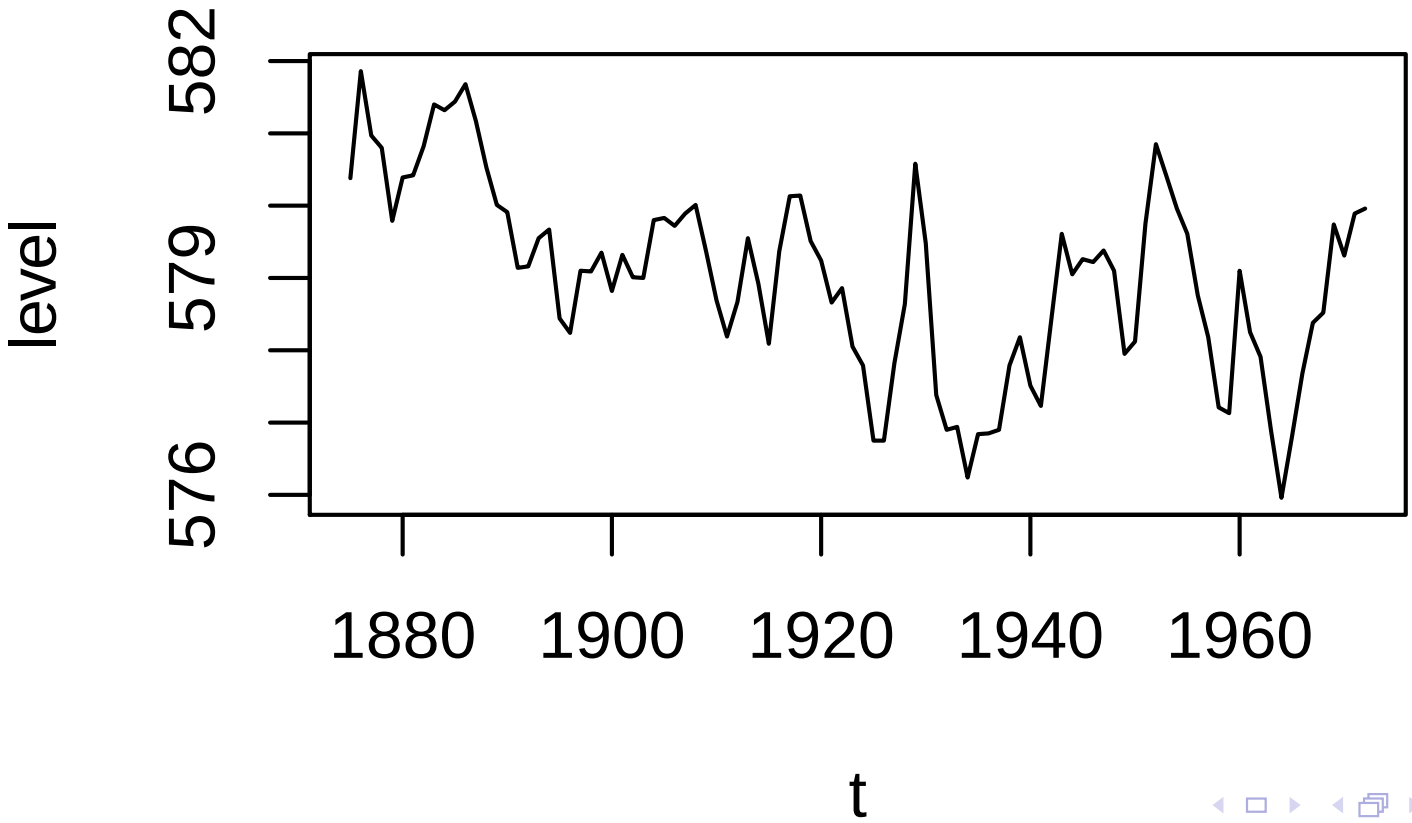
This series $\{Y_t,\ t = 1, 2, ..., 98\}$ has already been studied (see assignment 1). In this example we shall consider the problem of fitting an AR process to these data.

Example. The lake data

```
# Just type LakeHuron;

LakeHuron;

## Time Series:
## Start = 1875
## End = 1972
## Frequency = 1
##  [1] 580.38 581.86 580.97 580.80 579.79 580.39 580.42 580.8
## [11] 581.44 581.68 581.17 580.53 580.01 579.91 579.14 579.1
## [21] 578.44 578.24 579.10 579.09 579.35 578.82 579.32 579.0
## [31] 579.83 579.72 579.89 580.01 579.37 578.69 578.19 578.6
## [41] 578.09 579.37 580.13 580.14 579.51 579.24 578.66 578.8
## [51] 576.75 576.75 577.82 578.64 580.58 579.48 577.38 576.9
## [61] 576.84 576.85 576.90 577.79 578.18 577.51 577.23 578.4
## [71] 579.26 579.22 579.38 579.10 577.95 578.12 579.75 580.8
## [81] 579.61 578.76 578.18 577.21 577.13 579.10 578.25 577.9
```

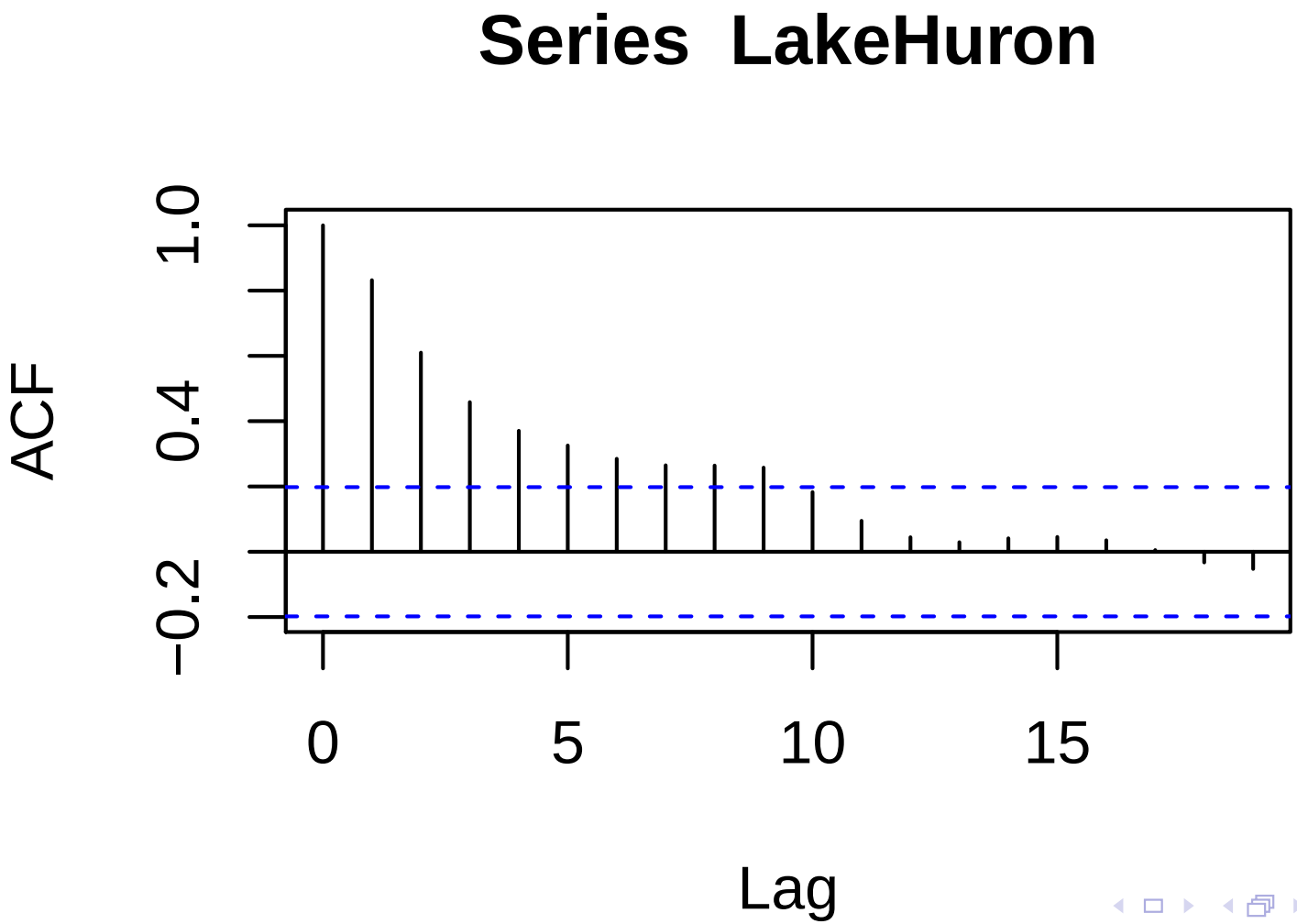
Example. The lake data

```
plot.ts(LakeHuron,xlab="t",ylab="level");
```



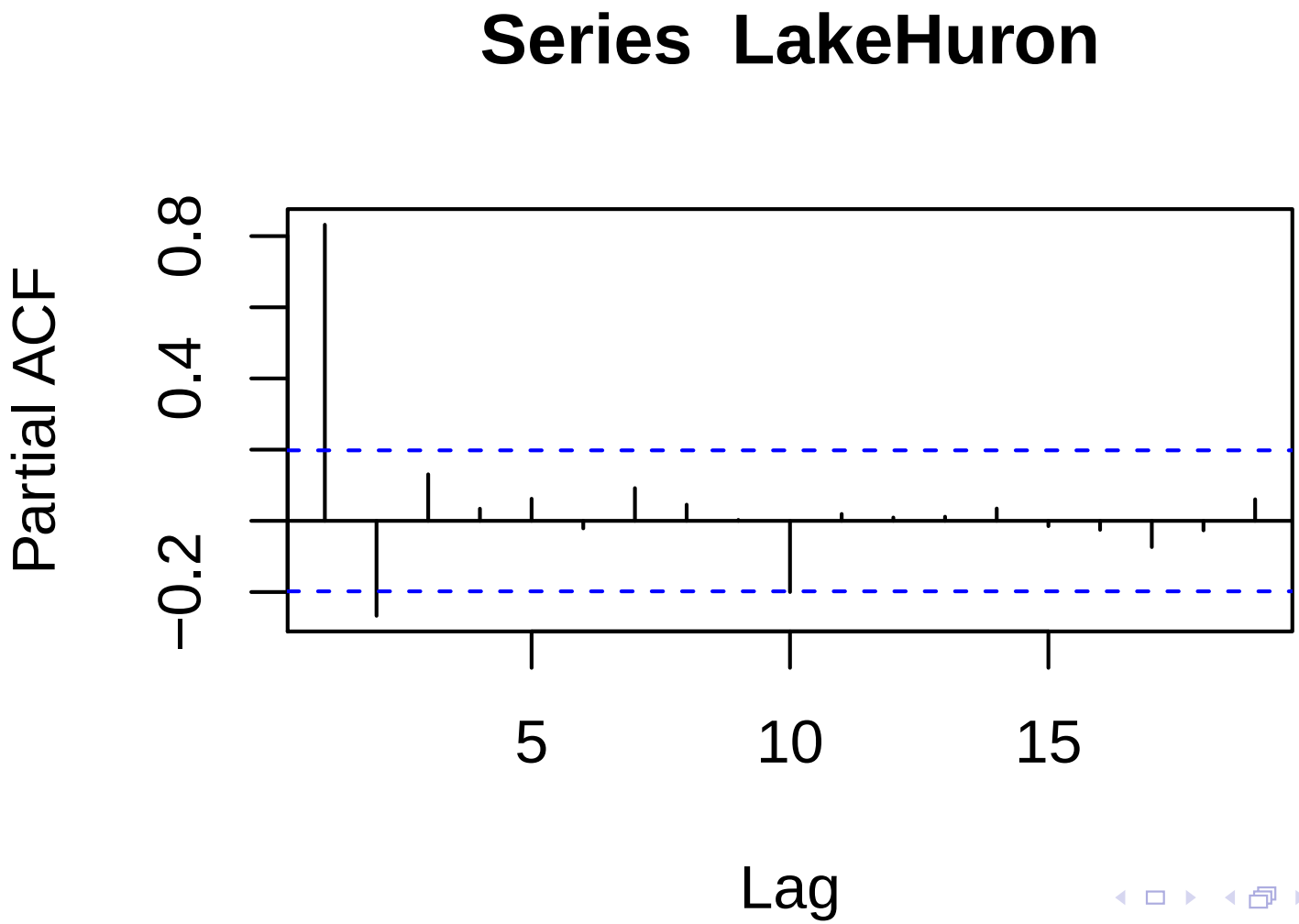
Example. The lake data

```
acf(LakeHuron)
```



Example. The lake data

```
pacf(LakeHuron)
```



Example. The lake data

```
mod1.yw<-ar.yw(LakeHuron);
```

Example. The lake data

```
mod1.yw$aic[1:11];

##           0           1           2           3           4
## 118.6683709   5.2338642   0.0000000   0.3100411   2.1963067
##           6           7           8           9          10
##   5.7739630   6.9415933   8.7386819  10.7379711   8.7361256
```

Example. The lake data

```
mod1.yw$order;

## [1] 2
```

Example. The lake data

```
mod1.yw$ar;

## [1]  1.0538249 -0.2667516
```

Example. The lake data

```
mod1.yw$var.pred;

## [1] 0.5075296
```

Example. The lake data

```
mod1.yw$asy.var.coef;

##               [,1]          [,2]
## [1,]  0.009777301 -0.008133846
## [2,] -0.008133846  0.009777301
```

Example. The lake data

95% Confidence Bounds for $\hat{\phi}_1$.

```
mod1.yw$ar[1]-1.96*sqrt(mod1.yw$asy.var.coef[1,1]);
```

```
## [1] 0.8600196
```

```
mod1.yw$ar[1]+1.96*sqrt(mod1.yw$asy.var.coef[1,1])
```

```
## [1] 1.24763
```

Example. The lake data

95% Confidence Bounds for $\hat{\phi}_2$.

```
mod1.yw$ar[2]-1.96*sqrt(mod1.yw$asy.var.coef[2,2]);
```

```
## [1] -0.4605569
```

```
mod1.yw$ar[2]+1.96*sqrt(mod1.yw$asy.var.coef[2,2])
```

```
## [1] -0.07294637
```

ar.mle()

```
m.0<-ar.mle(LakeHuron,aic=FALSE,order.max=0);

m.1<-ar.mle(LakeHuron,aic=FALSE,order.max=1);

m.2<-ar.mle(LakeHuron,aic=FALSE,order.max=2);

m.3<-ar.mle(LakeHuron,aic=FALSE,order.max=3);

m.4<-ar.mle(LakeHuron,aic=FALSE,order.max=4);

m.5<-ar.mle(LakeHuron,aic=FALSE,order.max=5);
```

Model	σ_w^2	BIC	Δ BIC
0	1.7201772	53.1578773	116.1579372
1	0.5092867	-61.5399601	1.4600998
2	0.478821	-63.0000599	$-3.5527137 \times 10^{-15}$
3	0.4720128	-59.8185224	3.1815375
4	0.4710446	-55.4347754	134.4978071
5	0.4703691	-50.9904536	139.0827746

Example. Quarterly

The file quarterly.txt contains a number of series including the U.S. index of industrial production (indprod), unemployment rate (urate), and producer price index for finished goods (finished). All the series run from 1960Q1 to 2008Q1 (Note: The first row contains the name of each series).

Example. Quarterly

```
#Step 1. Importing data;

# url of Dow Jones;
quarter_url=
"https://mcs.utm.utoronto.ca/~nosedal/data/quarterly.txt"

quarter=read.table(quarter_url,header=TRUE);

names(quarter);

head(quarter);
```

Example. Quarterly

```
##  [1] "Date"     "ffr"      "Tbill"    "r3"         "r10"
##  [7] "ConsNoFd" "Finished" "CapEq"    "PPINSA"     "M1NSA"
## [13] "CPINSA"   "CPICore"  "urate"    "indpro"
##      Date  ffr Tbill   r3  r10 Consumer ConsNoFd Finished Ca
## 1 1960Q1 3.93  3.94 4.67 4.49    33.13    33.37    33.03 32
## 2 1960Q2 3.70  3.09 4.25 4.26    33.47    33.33    33.30 32
## 3 1960Q3 2.94  2.39 3.49 3.83    33.63    33.47    33.43 32
## 4 1960Q4 2.30  2.36 3.52 3.89    33.77    33.53    33.50 32
## 5 1961Q1 2.00  2.38 3.40 3.79    33.87    33.53    33.63 32
## 6 1961Q2 1.73  2.33 3.40 3.79    33.80    33.50    33.57 32
##    M1NSA  M2NSA CPINSA CPICore urate indpro
## 1 140.53 299.40  30.57   30.57  5.13  26.78
## 2 138.40 300.03  30.60   30.63  5.23  26.20
## 3 139.60 305.50  30.60   30.60  5.53  25.77
## 4 142.67 312.30  30.80   30.77  6.27  25.15
## 5 142.23 317.10  30.80   30.83  6.80  24.77
```

Exercises with urate.

a) Plot and examine the ACF of the series.
b) Plot and examine the PACF of the series.
c) Estimate urate as an AR(2) process including a constant (i.e. $y_t = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + w_t$). Provide estimated coefficients.
d) Make a time series plot where you show urate and your predictions from part c), on the same plot.

Exercises with finished.

a) Construct the growth rate of the series as
$y_t = log(finished_t) - log(finished_{t-1})$. Make the time series plot of y_t.
b) Make and examine the ACF of the series.
c) Make and examine the PACF of the series.
d) Since the first few autocorrelations suggest an AR process, fit a series of AR processes using *ar.yw( )*. Find the AR process that minimizes AIC and provide: order, estimated coefficients, and σ_w^2.

Example. Dow Jones

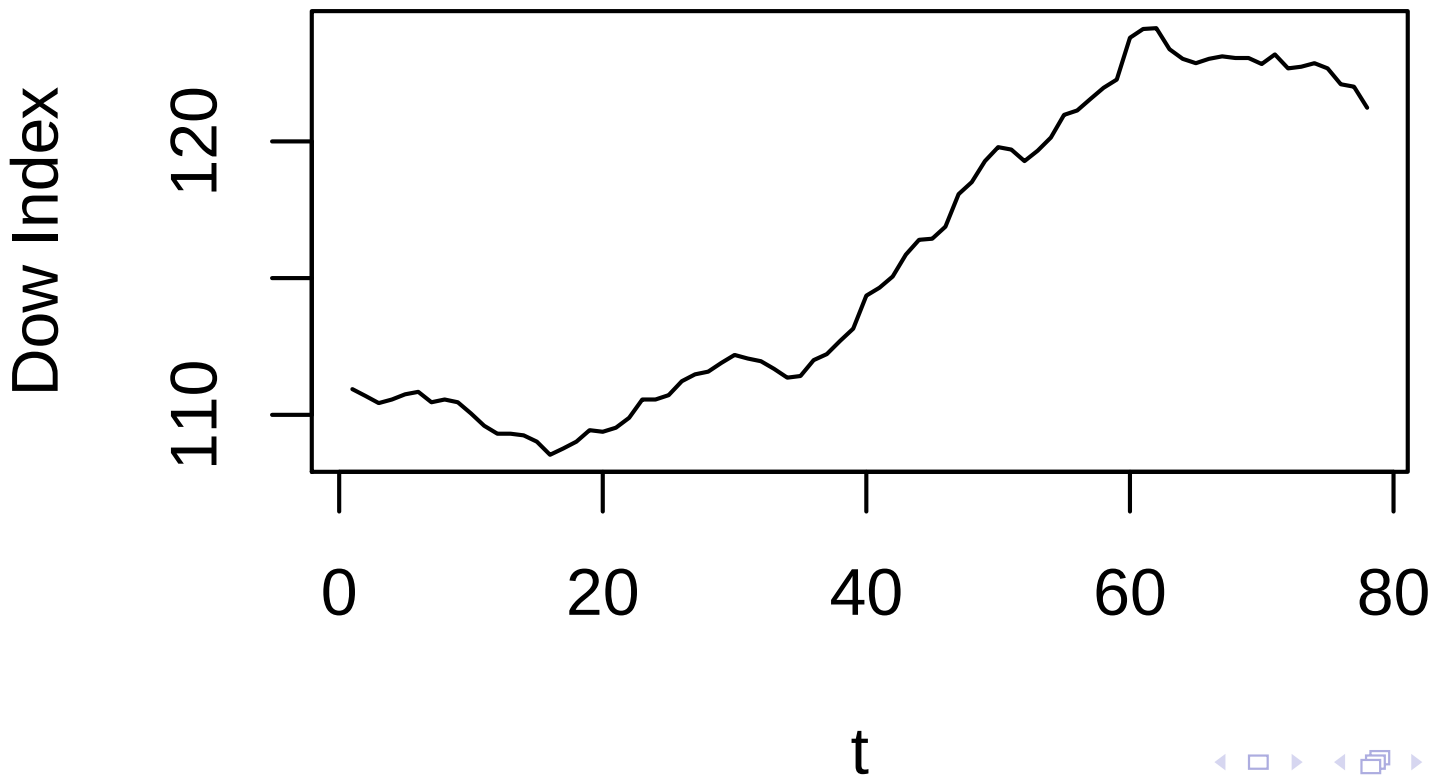
```
#Step 1. Importing data;

# url of Dow Jones;
dowj_url=
"https://mcs.utm.utoronto.ca/~nosedal/data/dowj.txt"

dow=read.table(dowj_url,header=FALSE);

names(dow);

head(dow);
```

Example. Dow Jones

```
## [1] "V1"
##       V1
## 1 110.94
## 2 110.69
## 3 110.43
## 4 110.56
## 5 110.75
## 6 110.84
```

Example. Dow Jones

```
plot.ts(dow,xlab="t",ylab="Dow Index")
```



Example. Dow Jones

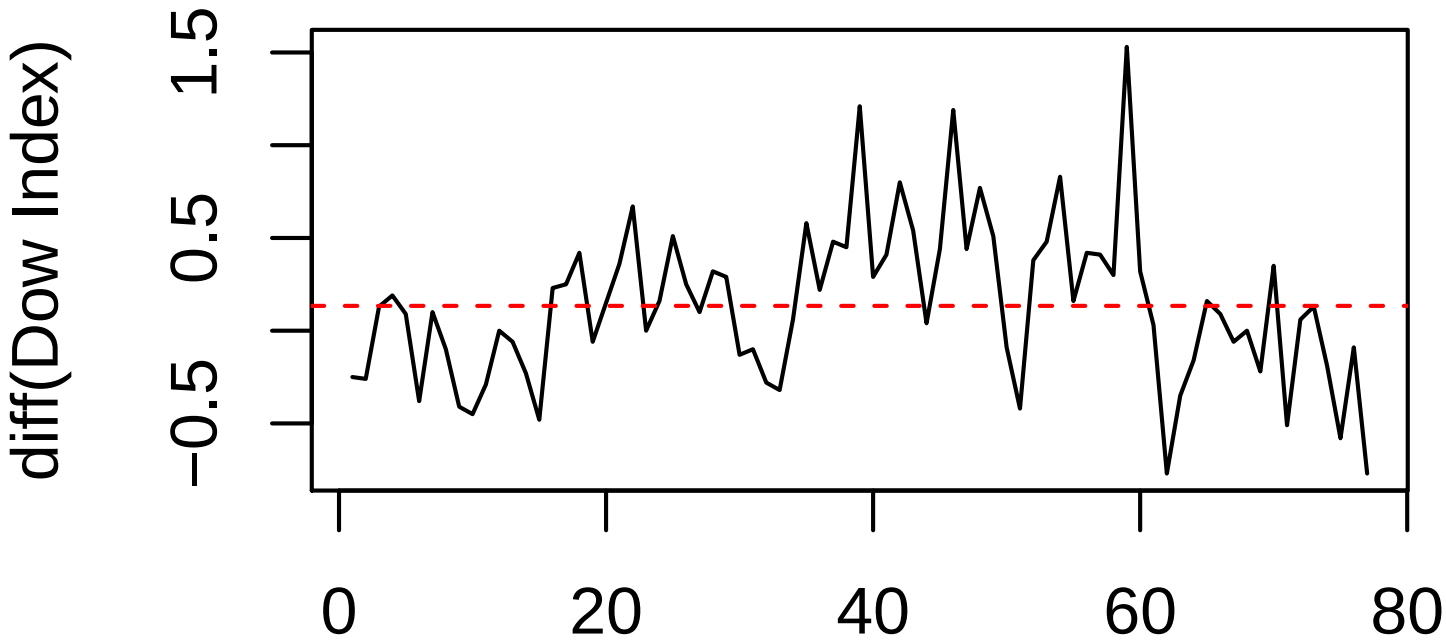
```
mode(dow);

## [1] "list"
```

Example. Dow Jones

```
# Computing first differences;
diff.dow<-diff(unlist(dow));
```

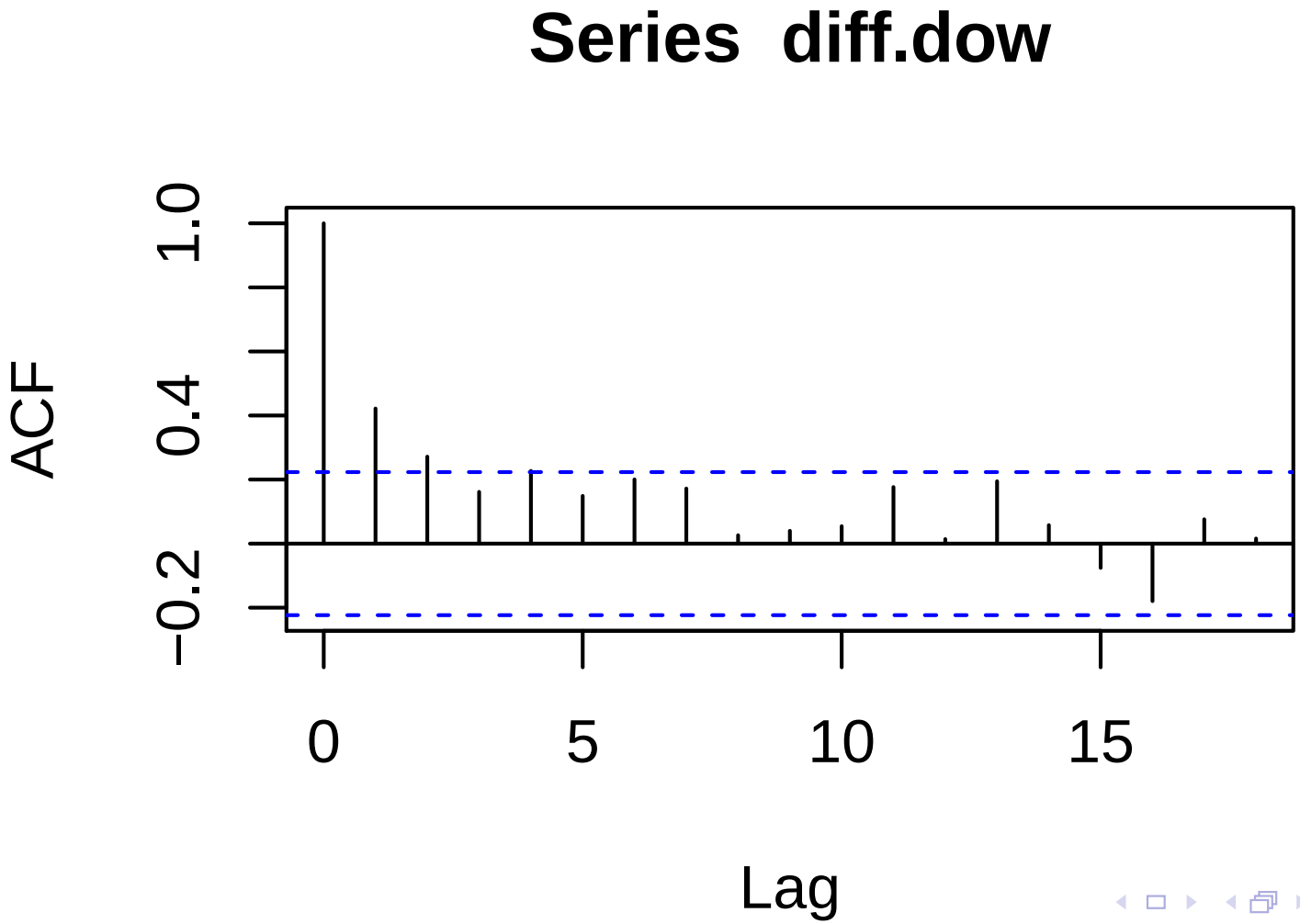
Example. Dow Jones

```
plot.ts(diff.dow,xlab="t",ylab="diff(Dow Index)");
abline(h=mean(diff.dow), lty=2, col="red");
```



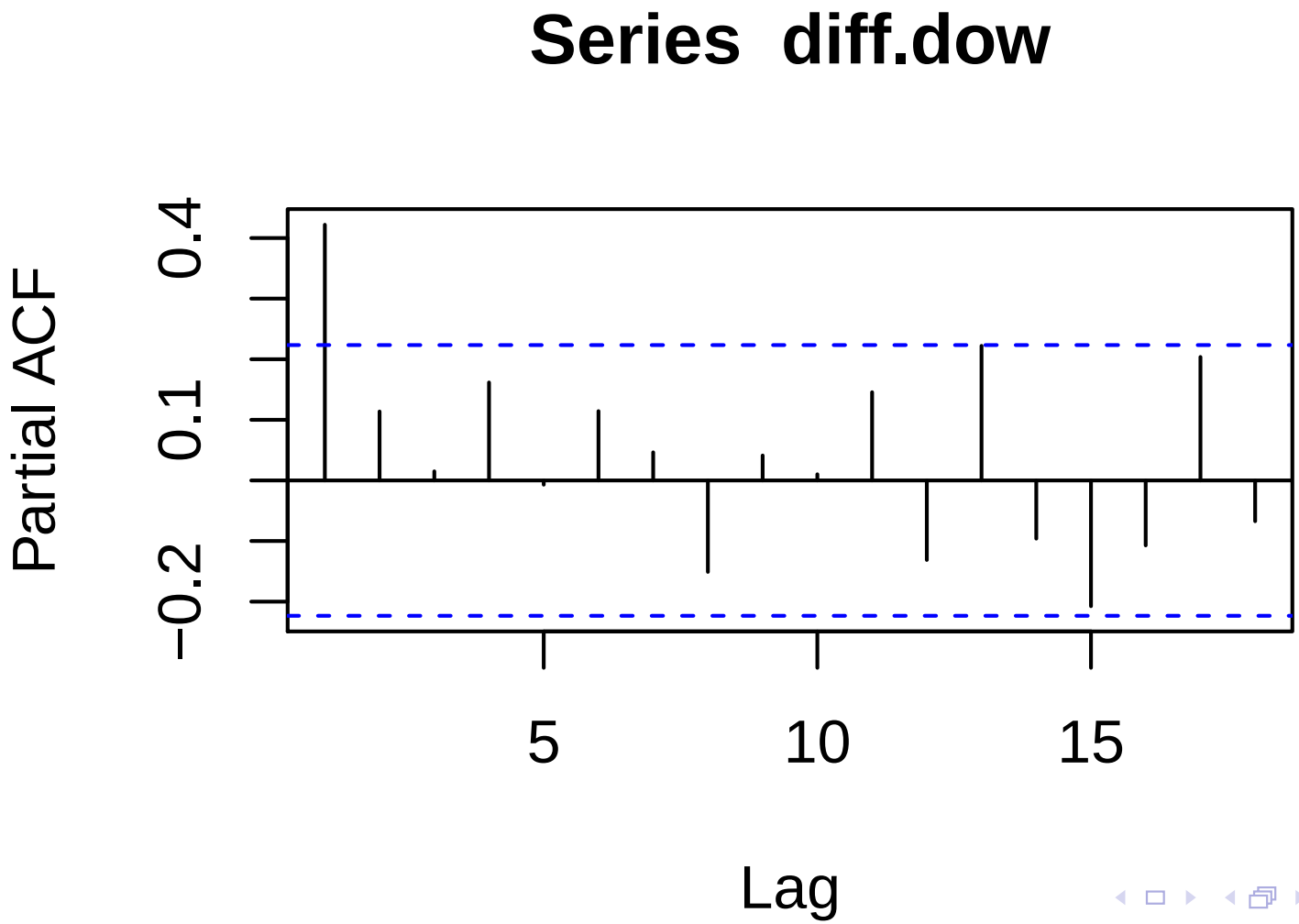
Example. Dow Jones

```
acf(diff.dow);
```



Example. Dow Jones

```
pacf(diff.dow);
```



Installing a package

```
install.packages("forecast", dependencies = TRUE)
```

Loading library

```
library(forecast);
```

Example. Dow Jones ... again

```
auto.arima(dow);

## Series: dow
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1
##       0.8510  -0.5263
## s.e.  0.1383   0.2548
##
## sigma^2 estimated as 0.1474:  log likelihood=-34.69
## AIC=75.38   AICc=75.71   BIC=82.41
```