harinris_Homework4

Harin Rishabh

2024-04-01

Problem - 1

Chapter 6 Commands

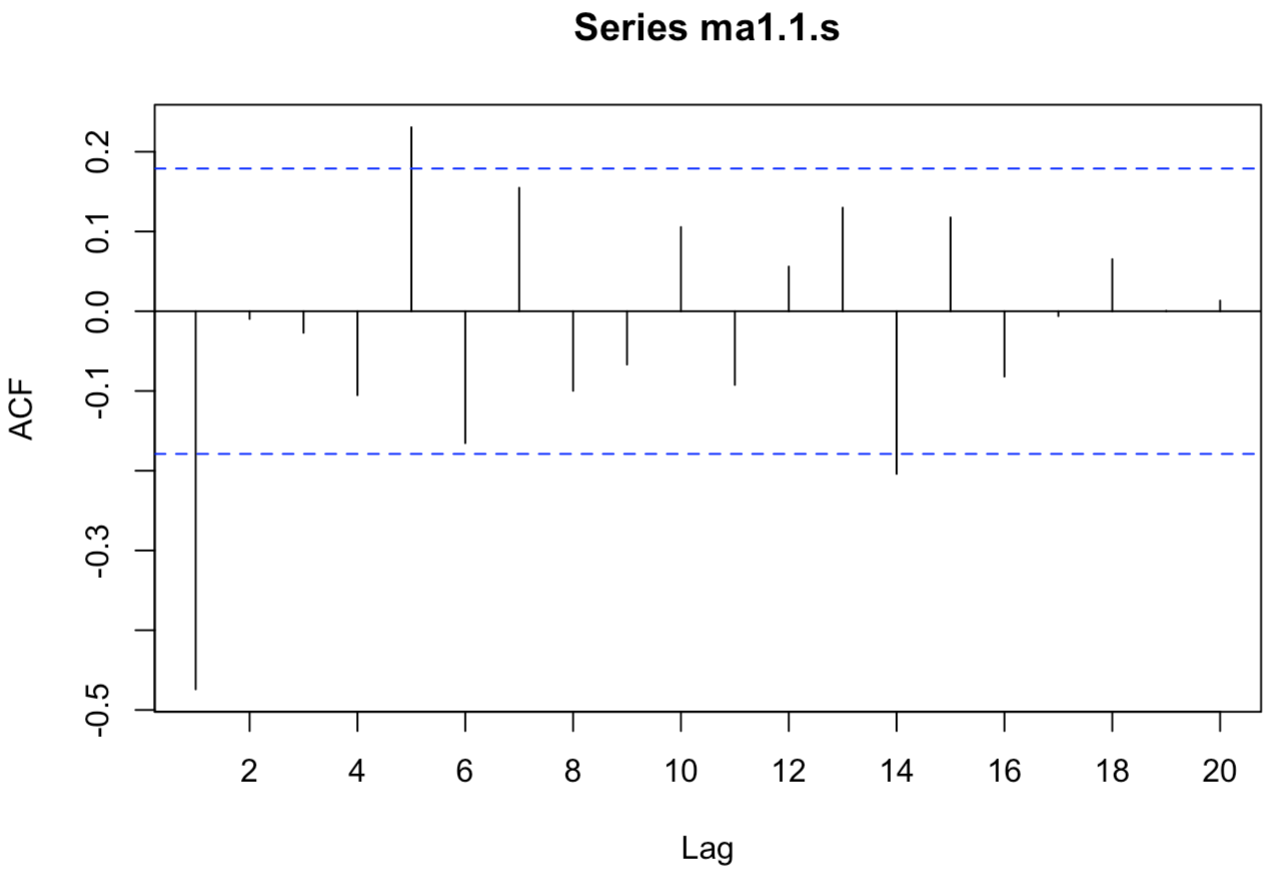
```
library(TSA) # Load the TSA package for time series analysis
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

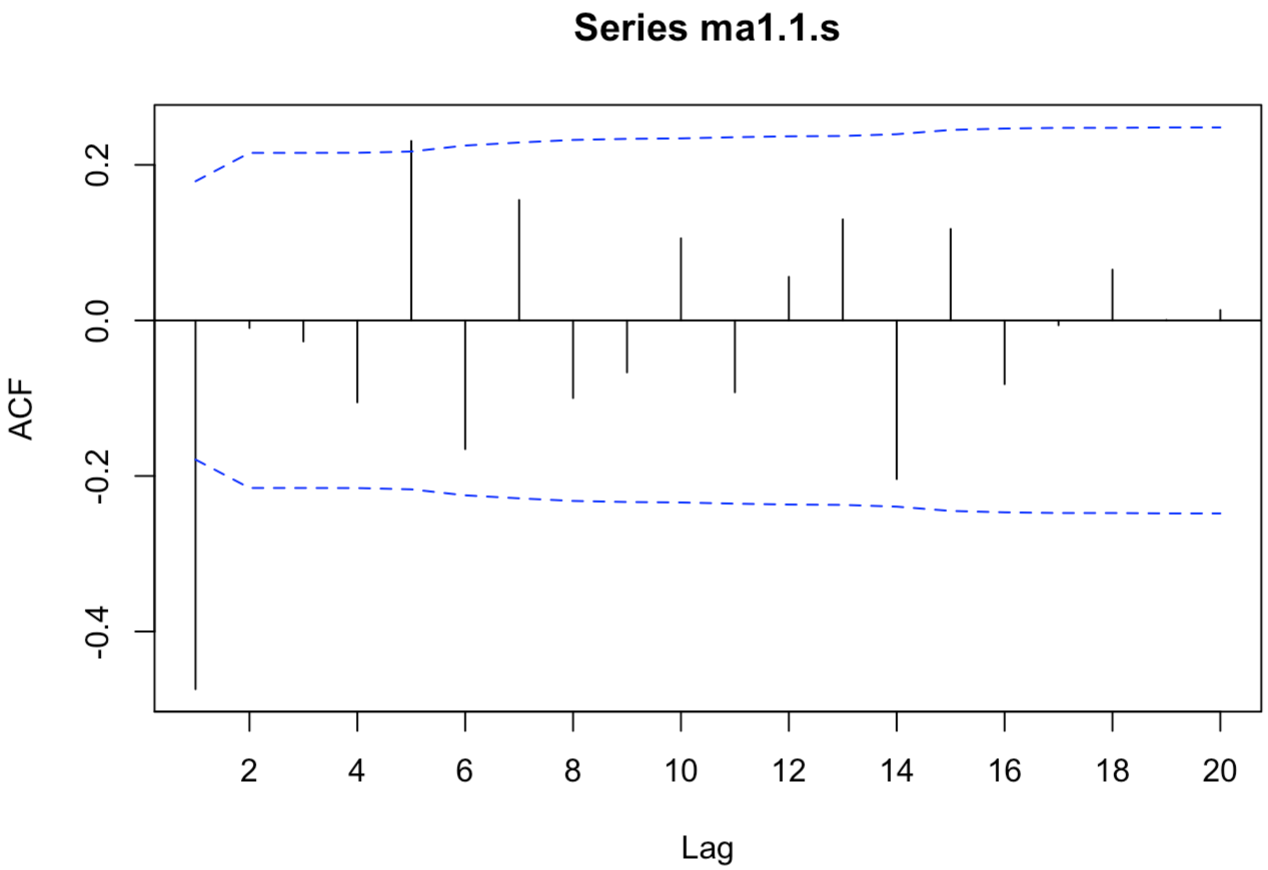
```
## The following object is masked from 'package:utils':
##
##     tar
```

```
data(ma1.1.s) # Load the dataset ma1.1.s
acf(ma1.1.s, xaxp=c(0,20,10)) # Compute and plot the autocorrelation function (ACF) o
f ma1.1.s
```

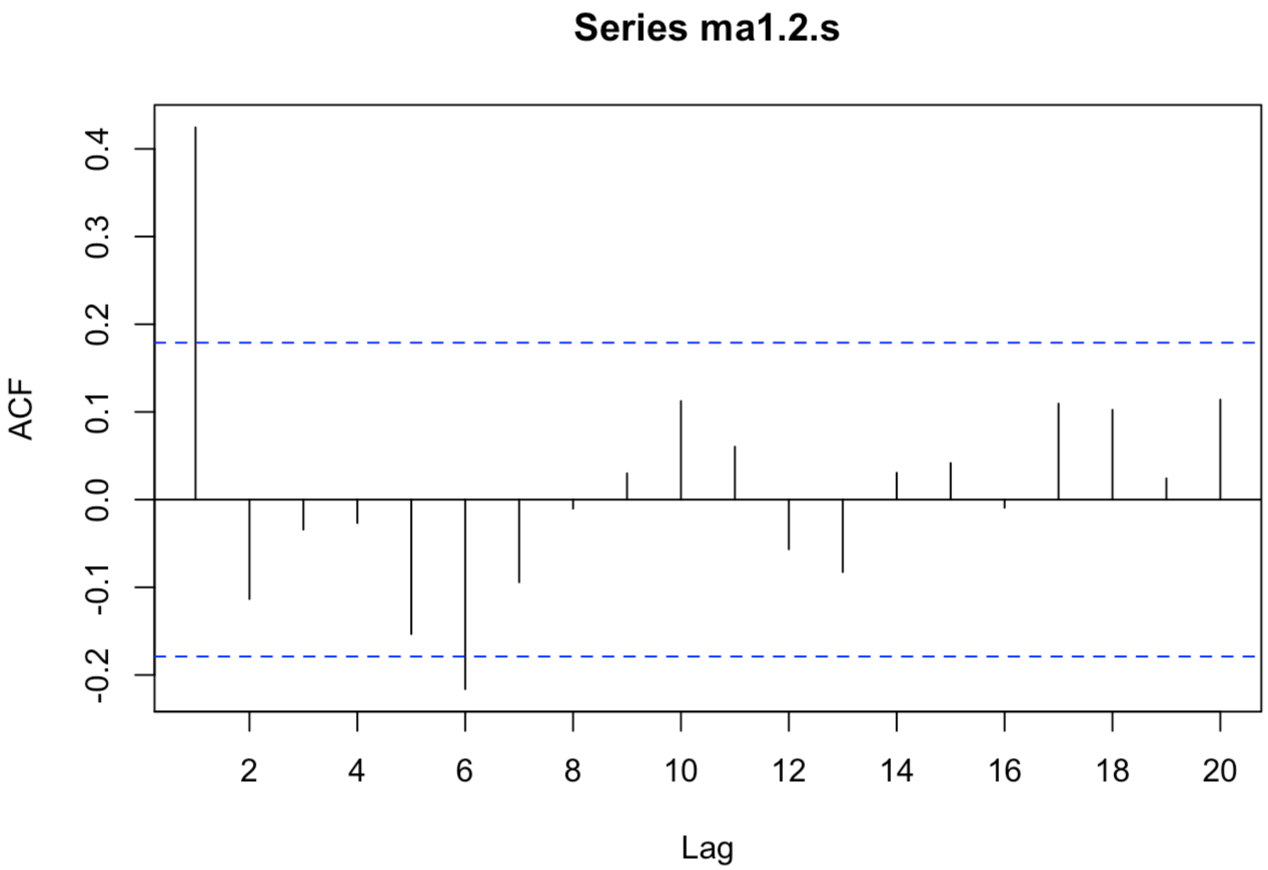



```
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
```

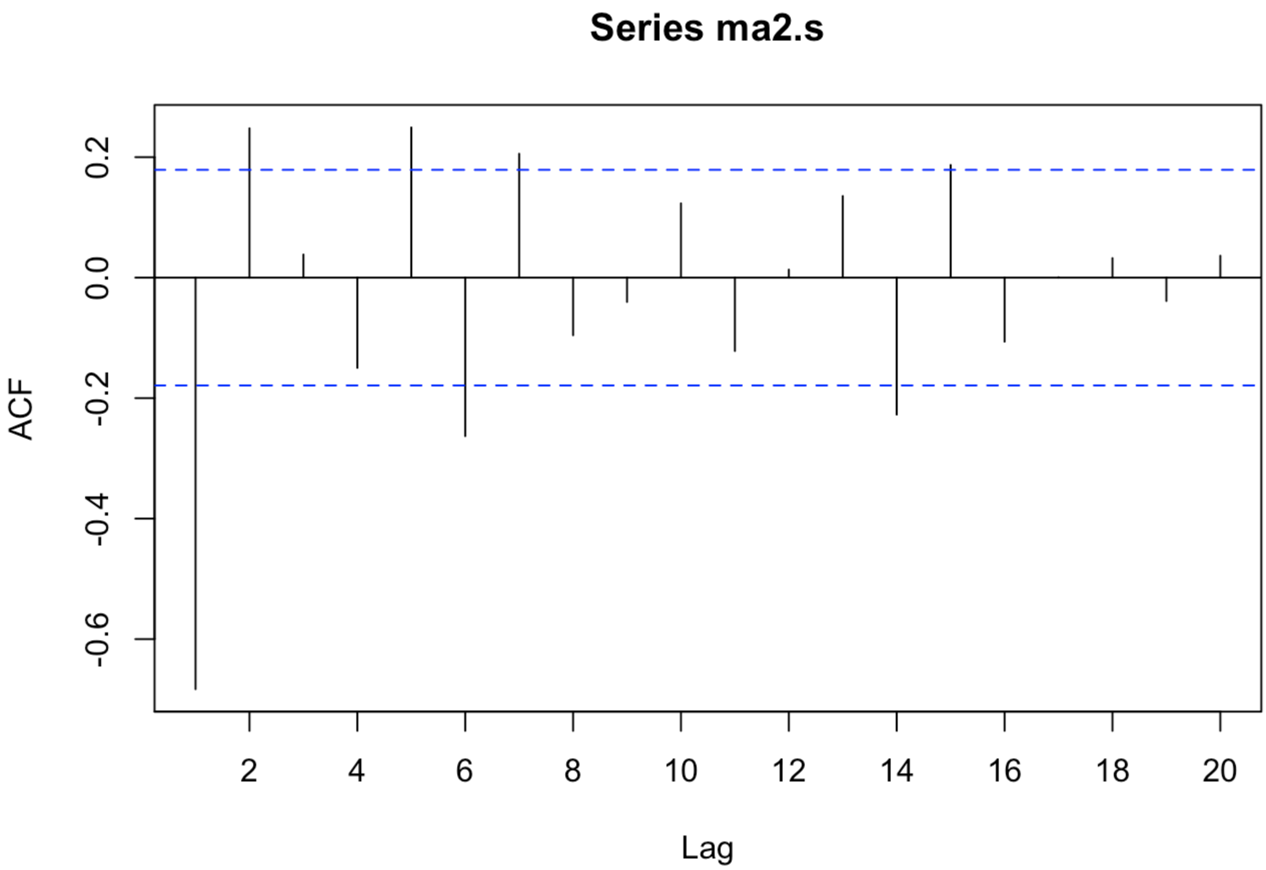
```
# Compute and plot the autocorrelation function (ACF) of ma1.1.s
# ci.type='ma' adds confidence intervals for a moving average (MA) process
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks

acf(ma1.1.s,ci.type='ma',xaxp=c(0,20,10))
```



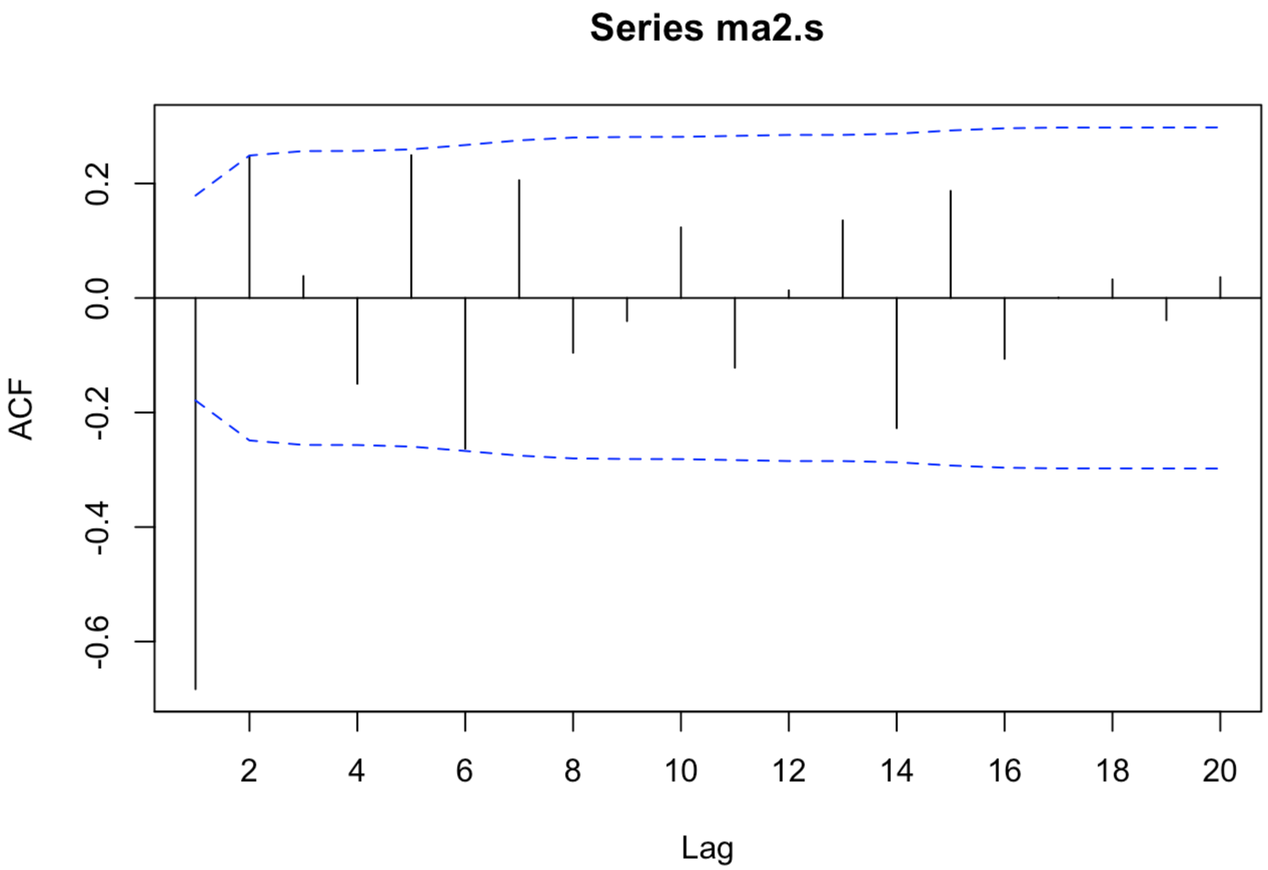

```
# Load the dataset ma1.2.s
data(ma1.2.s);
# Compute and plot the autocorrelation function (ACF) of ma1.2.s
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
acf(ma1.2.s,xaxp=c(0,20,10))
```



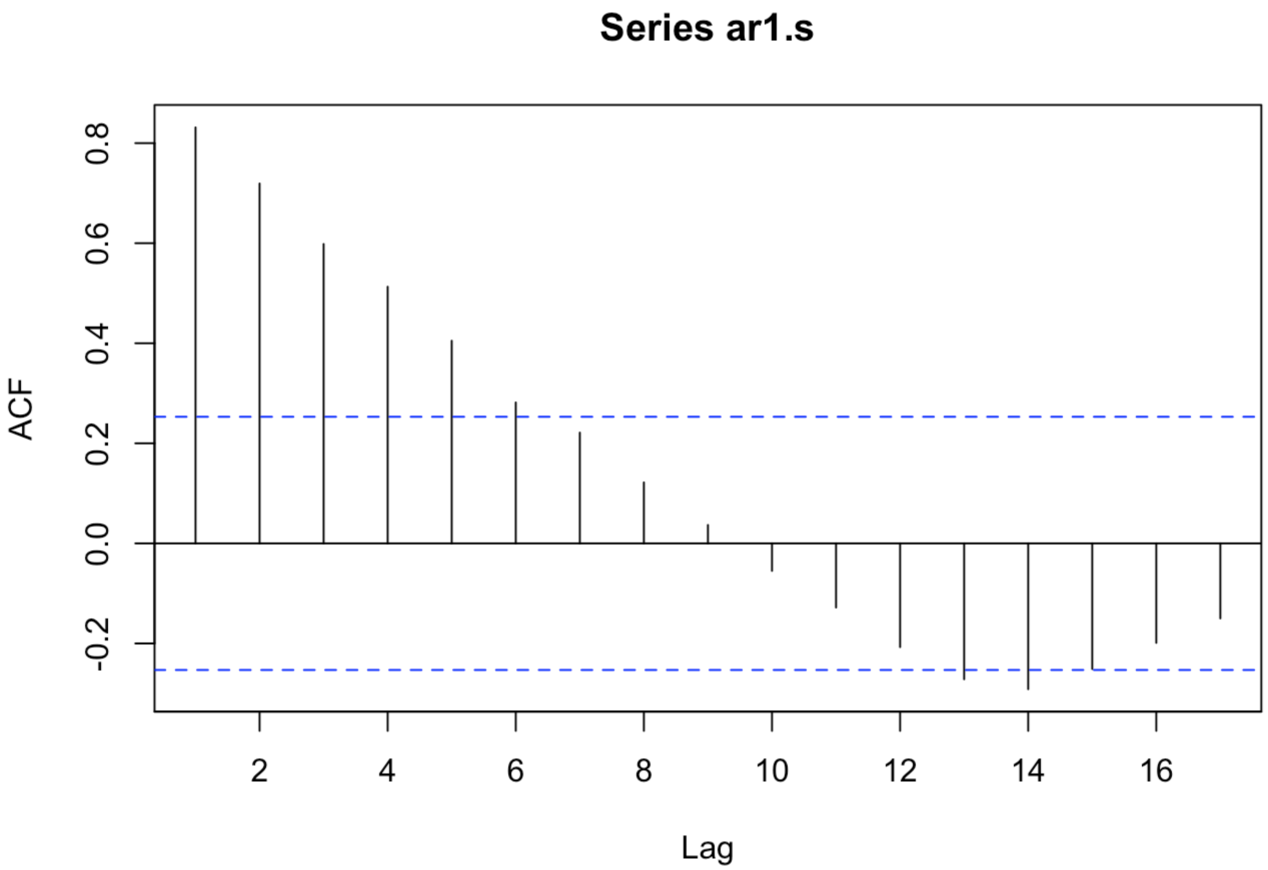

```
# Load the dataset ma2.s
data(ma2.s);
# Compute and plot the autocorrelation function (ACF) of ma2.s
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
acf(ma2.s,xaxp=c(0,20,10))
```



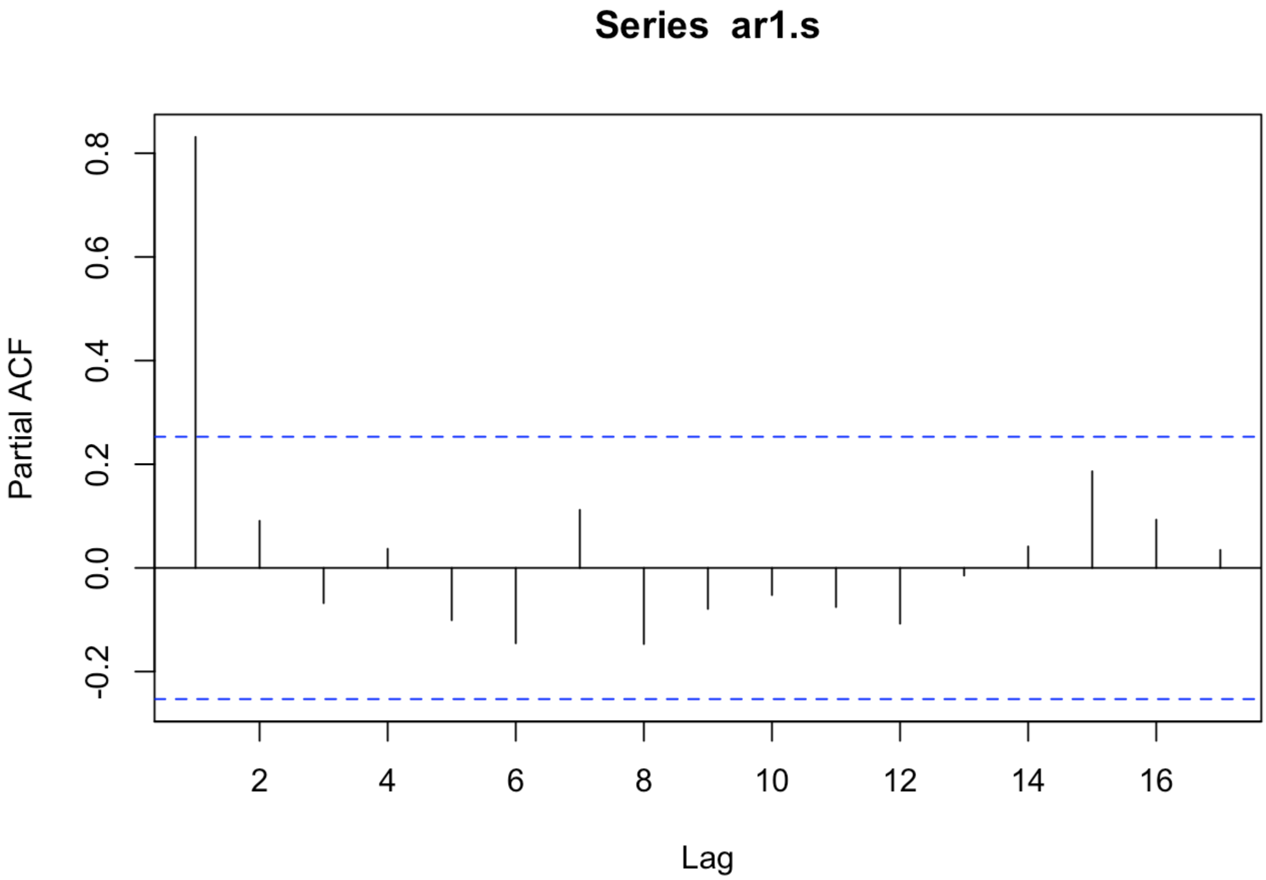

```
# Compute and plot the autocorrelation function (ACF) of ma2.s
# ci.type='ma' adds confidence intervals for a moving average (MA) process
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
acf(ma2.s,ci.type='ma',xaxp=c(0,20,10))
```



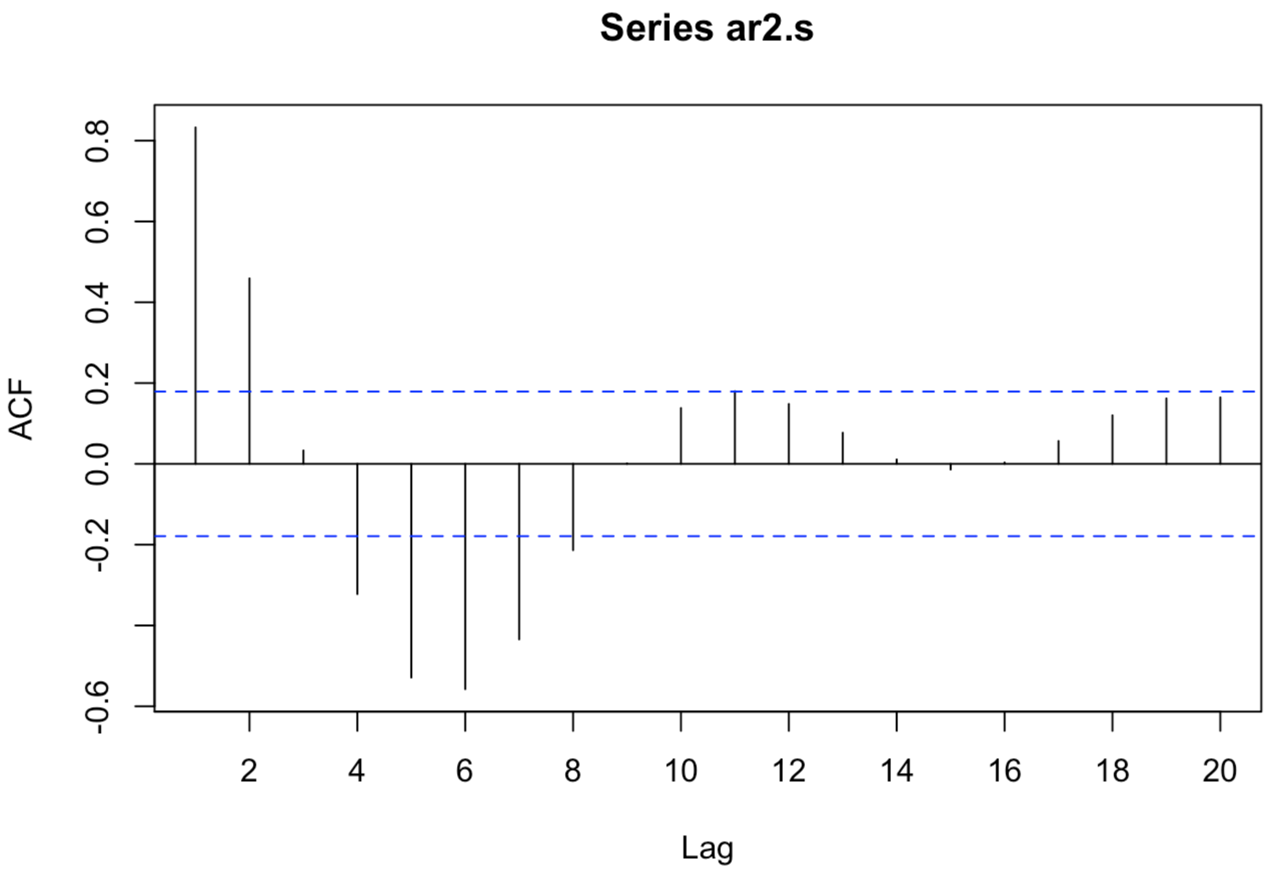

```
# Load the dataset ar1.s
data(ar1.s);
# Compute and plot the autocorrelation function (ACF) of ar1.s
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
acf(ar1.s,xaxp=c(0,20,10))
```



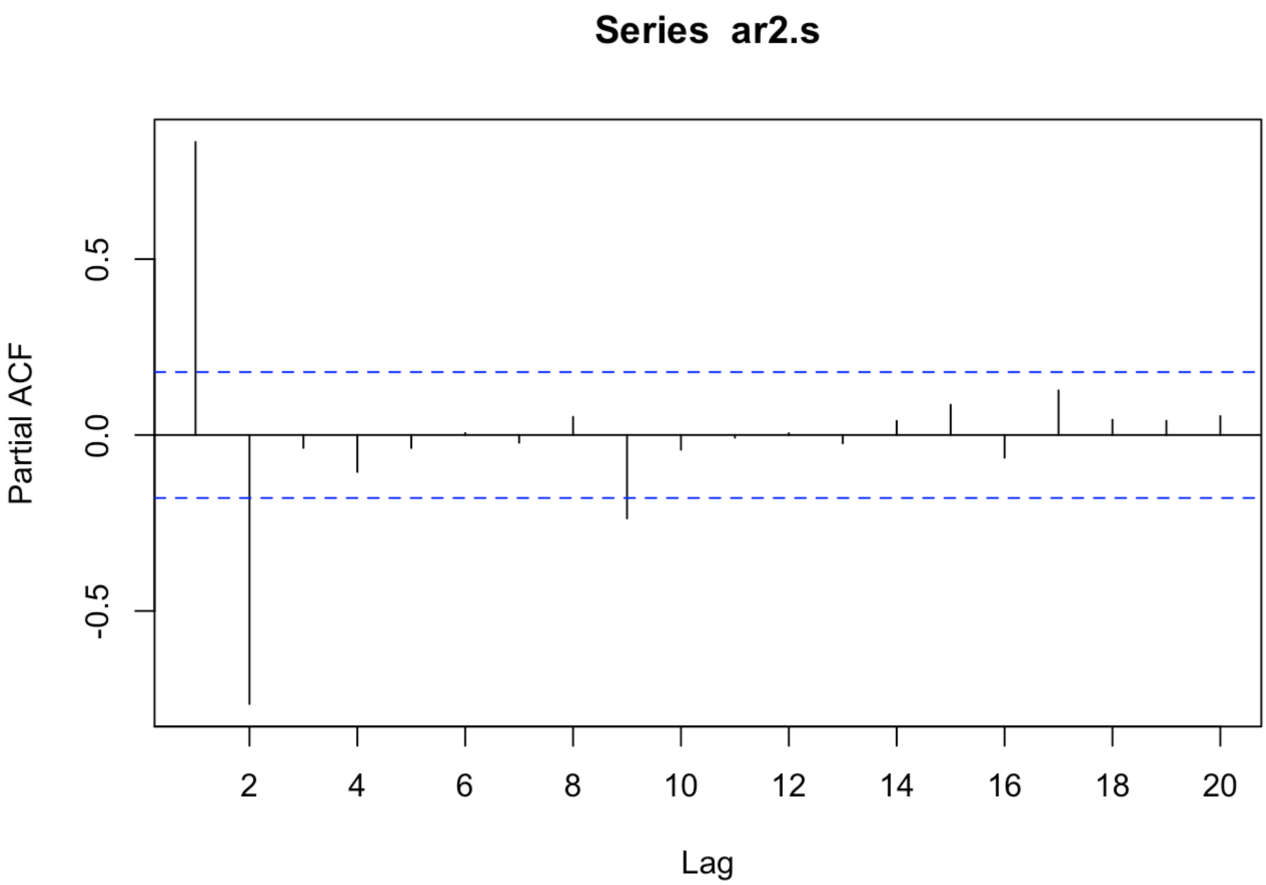

```
# Compute and plot the partial autocorrelation function (PACF) of ar1.s
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
pacf(ar1.s,xaxp=c(0,20,10))
```



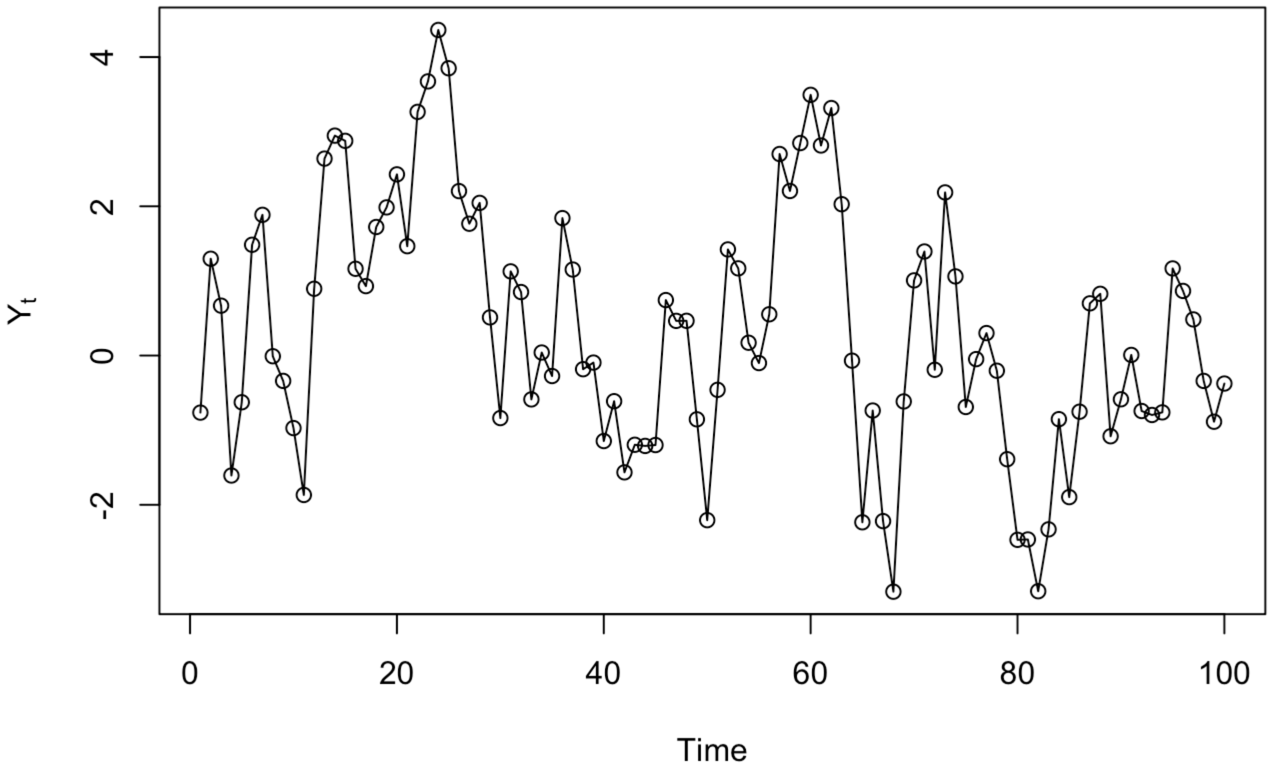

```
# Load the dataset ar2.s
data(ar2.s)
# Compute and plot the autocorrelation function (ACF) of ar2.s
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
acf(ar2.s,xaxp=c(0,20,10))
```



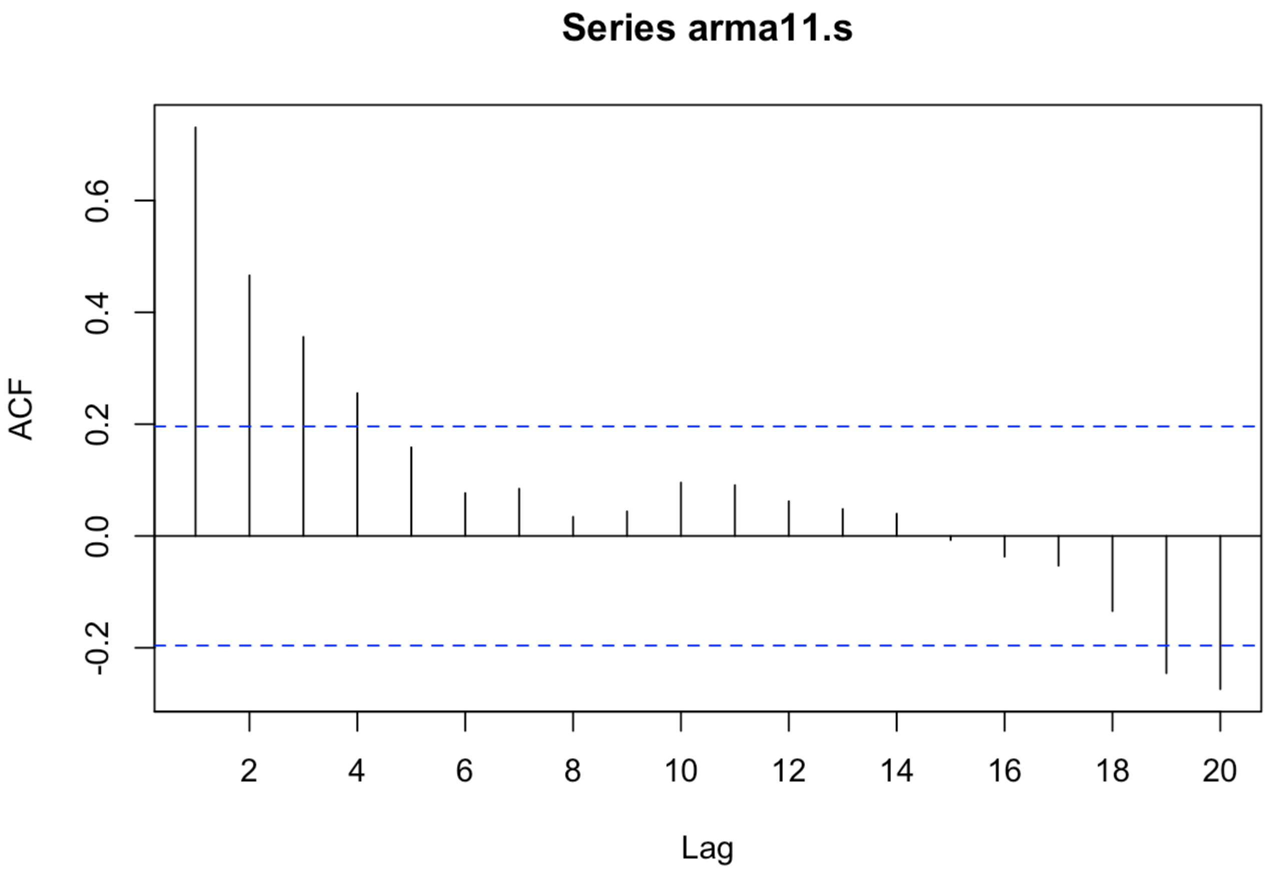

```
# Compute and plot the partial autocorrelation function (PACF) of ar2.s
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
pacf(ar2.s,xaxp=c(0,20,10))
```



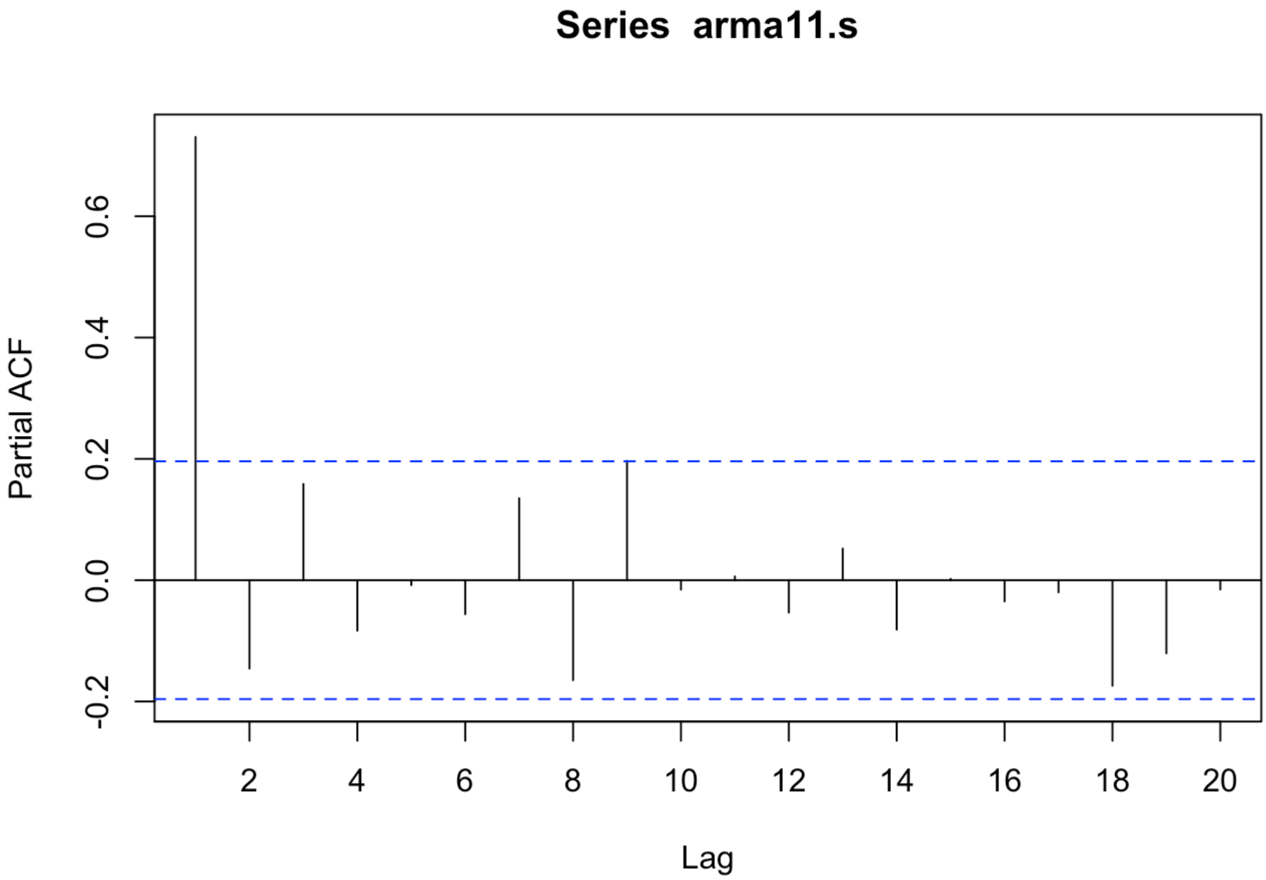

```
# Load the dataset arma11.s
data(arma11.s)

# Plot arma11.s as a time series with points connected by lines
# type='o' specifies both points and lines in the plot
# ylab=expression(Y[t]) sets the y-axis label using a mathematical expression
plot(arma11.s, type='o',ylab=expression(Y[t]))
```




```
# Compute and plot the autocorrelation function (ACF) of arma11.s
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
acf(arma11.s,xaxp=c(0,20,10))
```



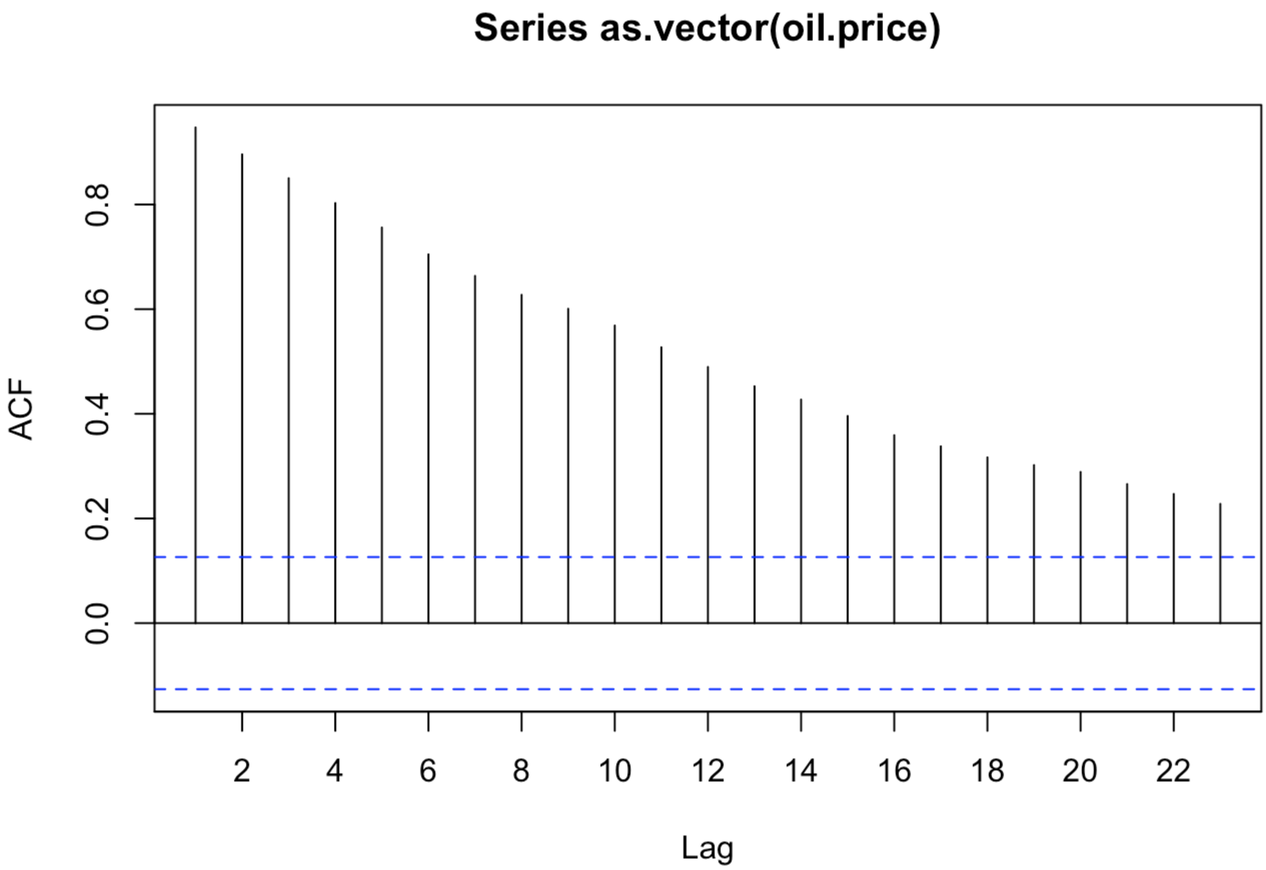

```
# Compute and plot the partial autocorrelation function (PACF) of arma11.s
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
pacf(arma11.s,xaxp=c(0,20,10))
```



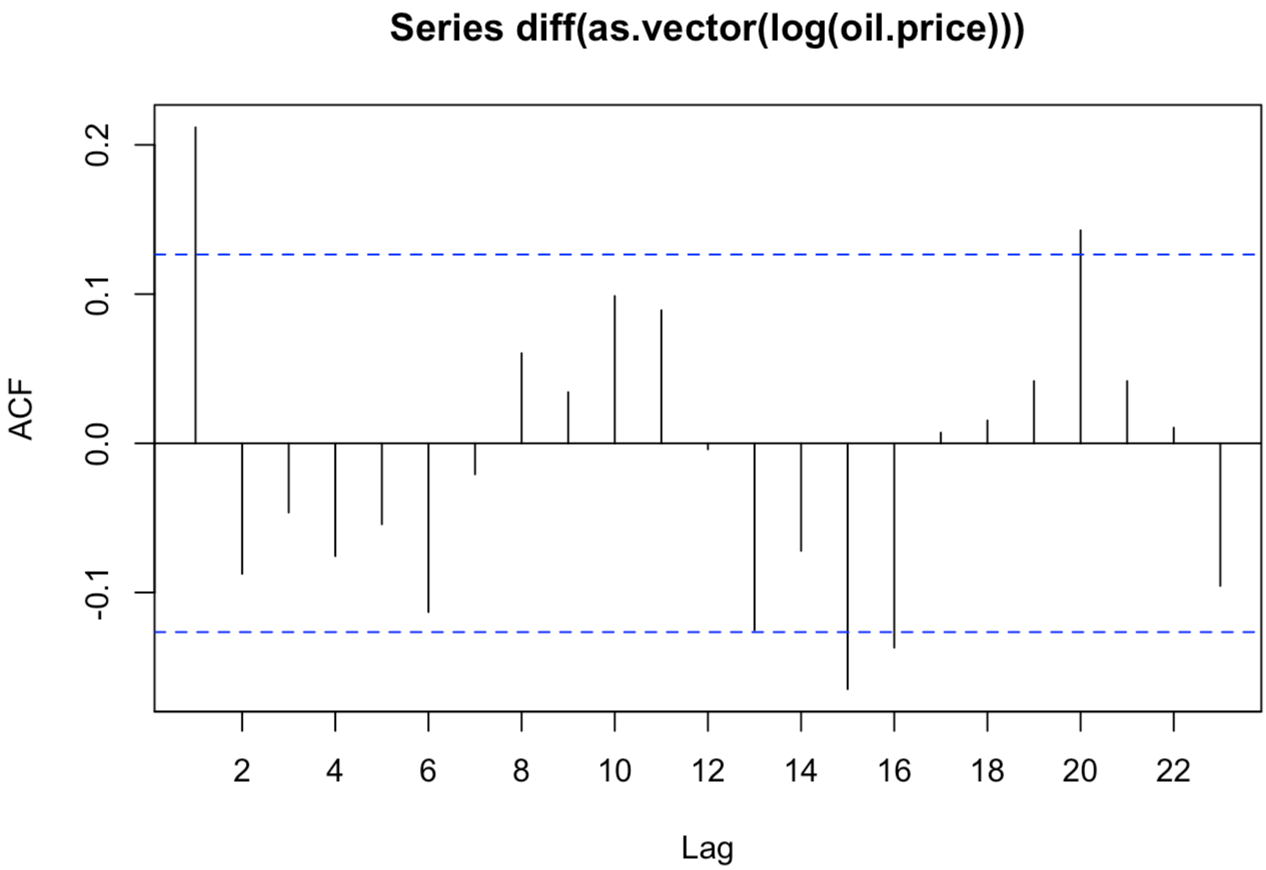

```
# Compute and plot the extended autocorrelation function (EACF) of arma11.s
eacf(arma11.s)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x o o o o o o o  o  o  o
## 1 x o o o o o o o o o o  o  o  o
## 2 x o o o o o o o o o o  o  o  o
## 3 x x o o o o o o o o o  o  o  o
## 4 x o x o o o o o o o o  o  o  o
## 5 x o o o o o o o o o o  o  o  o
## 6 x o o o x o o o o o o  o  o  o
## 7 x o o o x o o o o o o  o  o  o
```

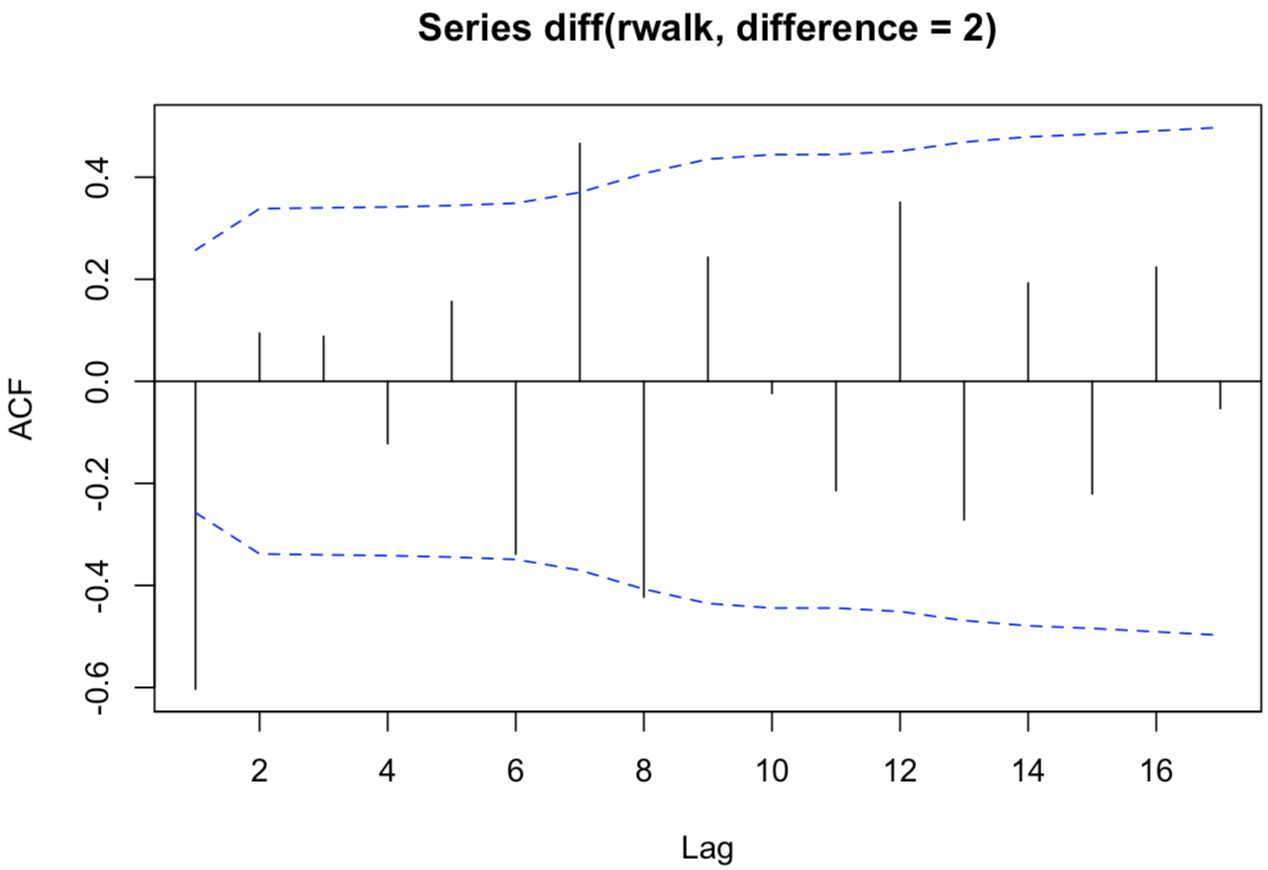
```
# Load the oil.price dataset
data(oil.price)

# Convert oil.price to a vector and compute the autocorrelation function (ACF)
# xaxp=c(0,24,12) sets the x-axis range from 0 to 24 with 12 tick marks
acf(as.vector(oil.price), xaxp=c(0,24,12))
```



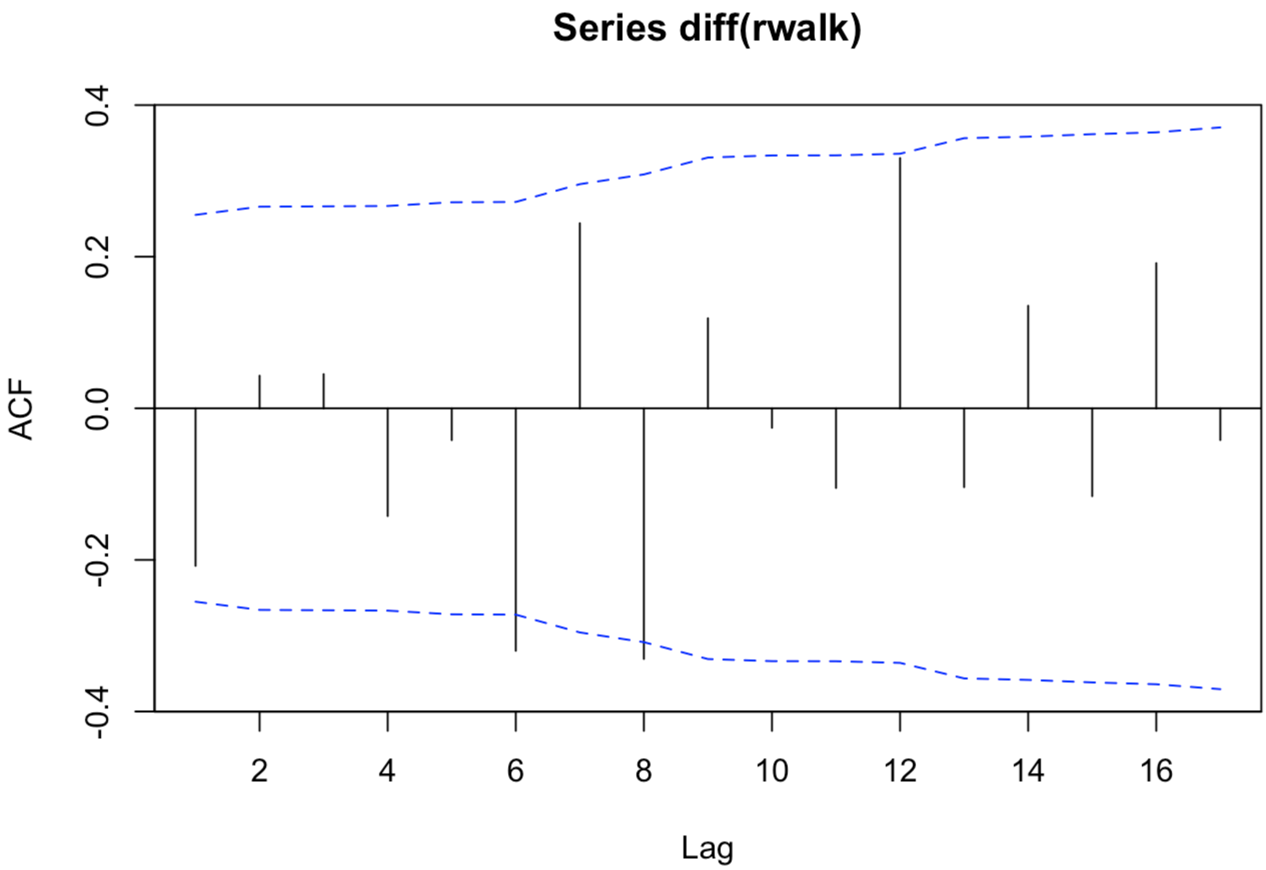

```
# Compute the logarithm of oil.price, convert it to a vector,
# take the difference of consecutive elements, and compute the autocorrelation functi
on (ACF)
# xaxp=c(0,24,12) sets the x-axis range from 0 to 24 with 12 tick marks
acf(diff(as.vector(log(oil.price))), xaxp=c(0,24,12))
```



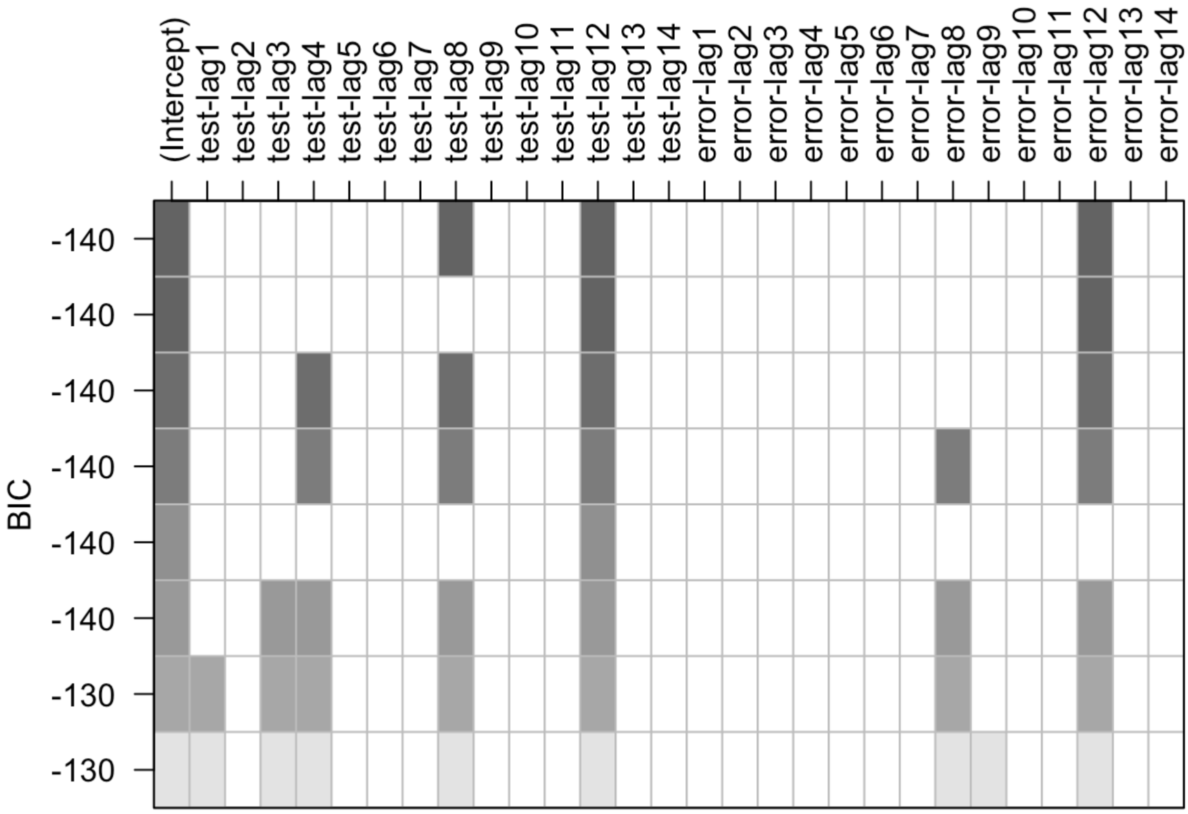

```
# Load the rwalk dataset
data(rwalk)

# Take the second order difference of the rwalk dataset,
# compute the autocorrelation function (ACF), and add confidence intervals for a movi
ng average (MA) process
# xaxp=c(0,18,9) sets the x-axis range from 0 to 18 with 9 tick marks
acf(diff(rwalk, difference=2), ci.type='ma', xaxp=c(0,18,9))
```



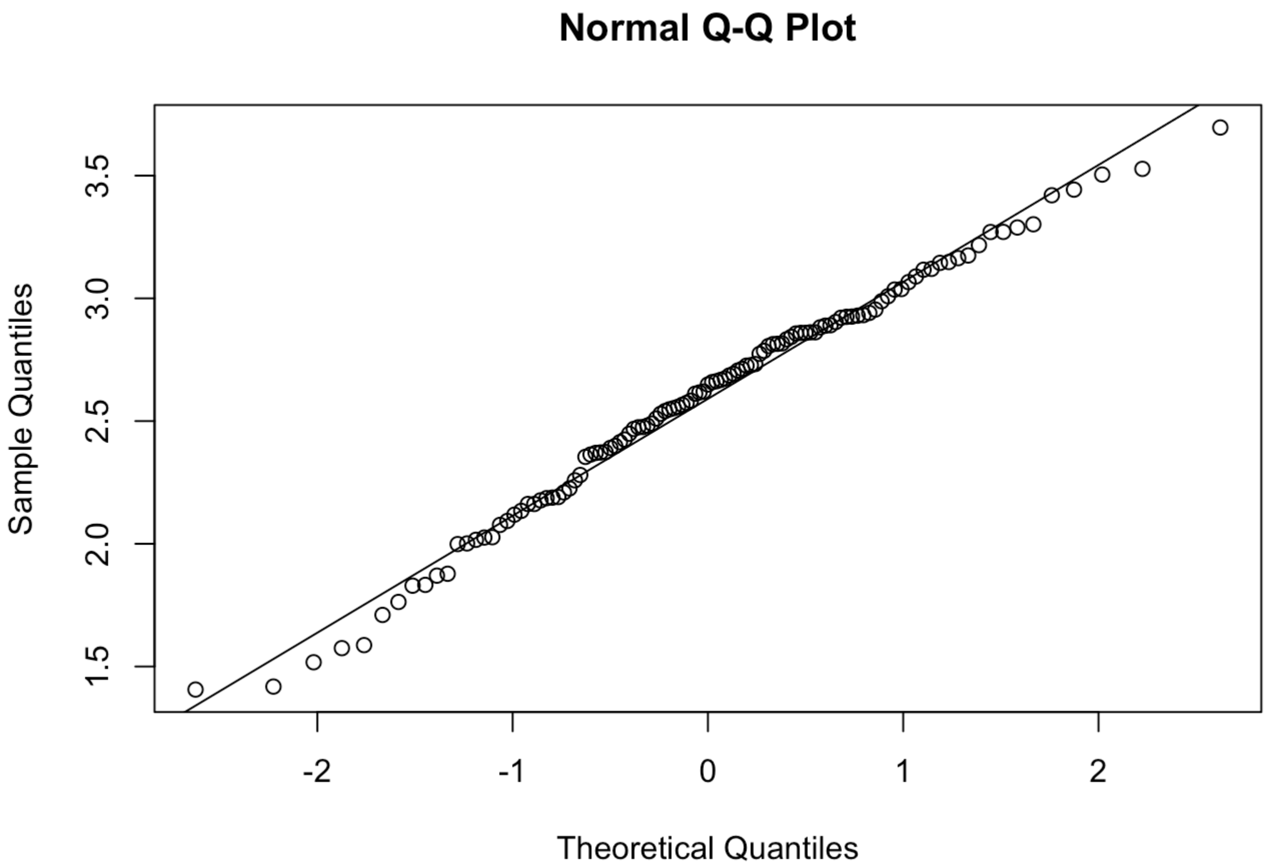

```
# Take the first order difference of the rwalk dataset,
# compute the autocorrelation function (ACF), and add confidence intervals for a movi
ng average (MA) process
# xaxp=c(0,18,9) sets the x-axis range from 0 to 18 with 9 tick marks
acf(diff(rwalk), ci.type='ma', xaxp=c(0,18,9))
```



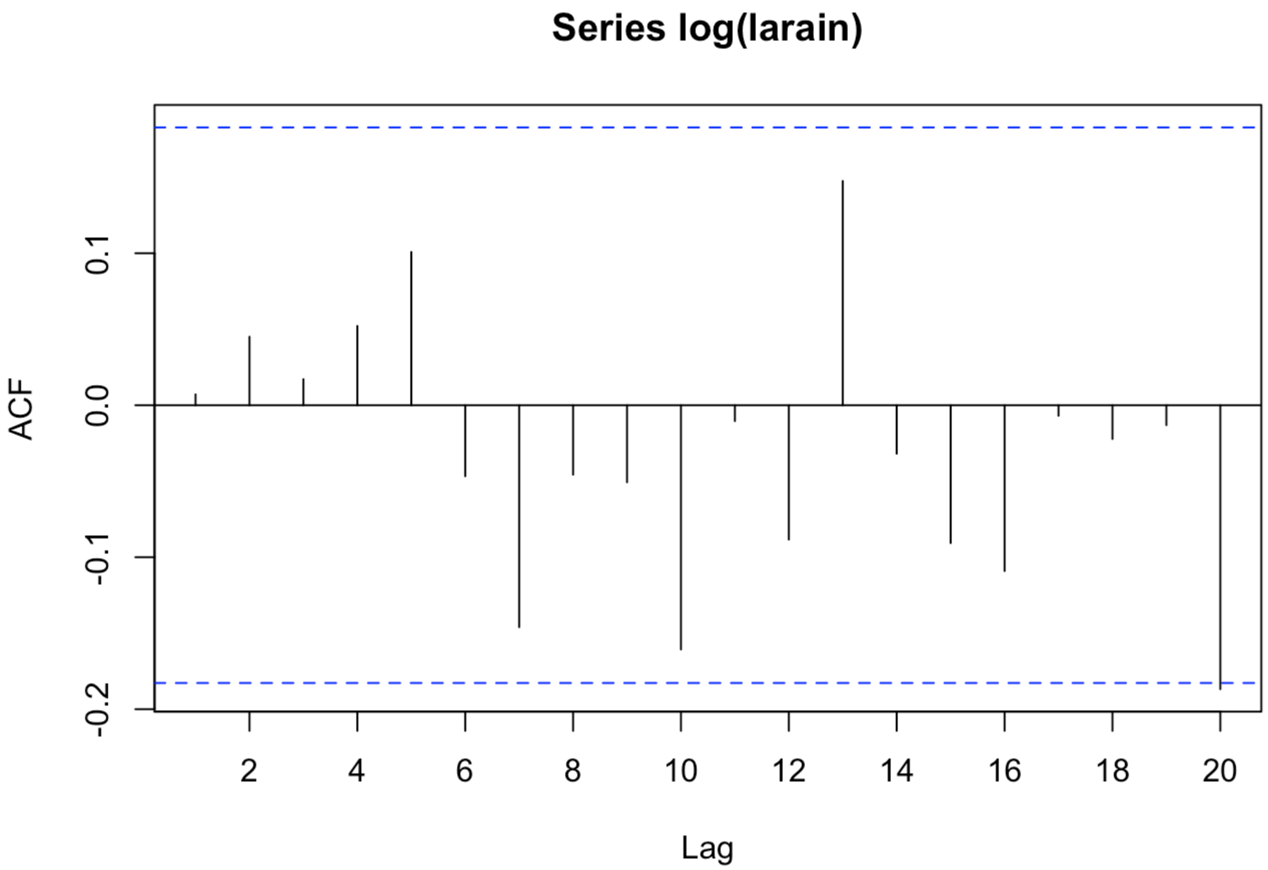

```
# Set the seed for reproducibility
set.seed(92397)

# Generate a time series using an ARIMA process
test = arima.sim(model=list(ar=c(rep(0,11),0.8), ma=c(rep(0,11),0.7)), n=120)

# Fit an ARMA model to the time series using subset selection
res = armasubsets(y=test, nar=14, nma=14, y.name='test', ar.method='ols')

# Plot the subset selection results
plot(res)
```



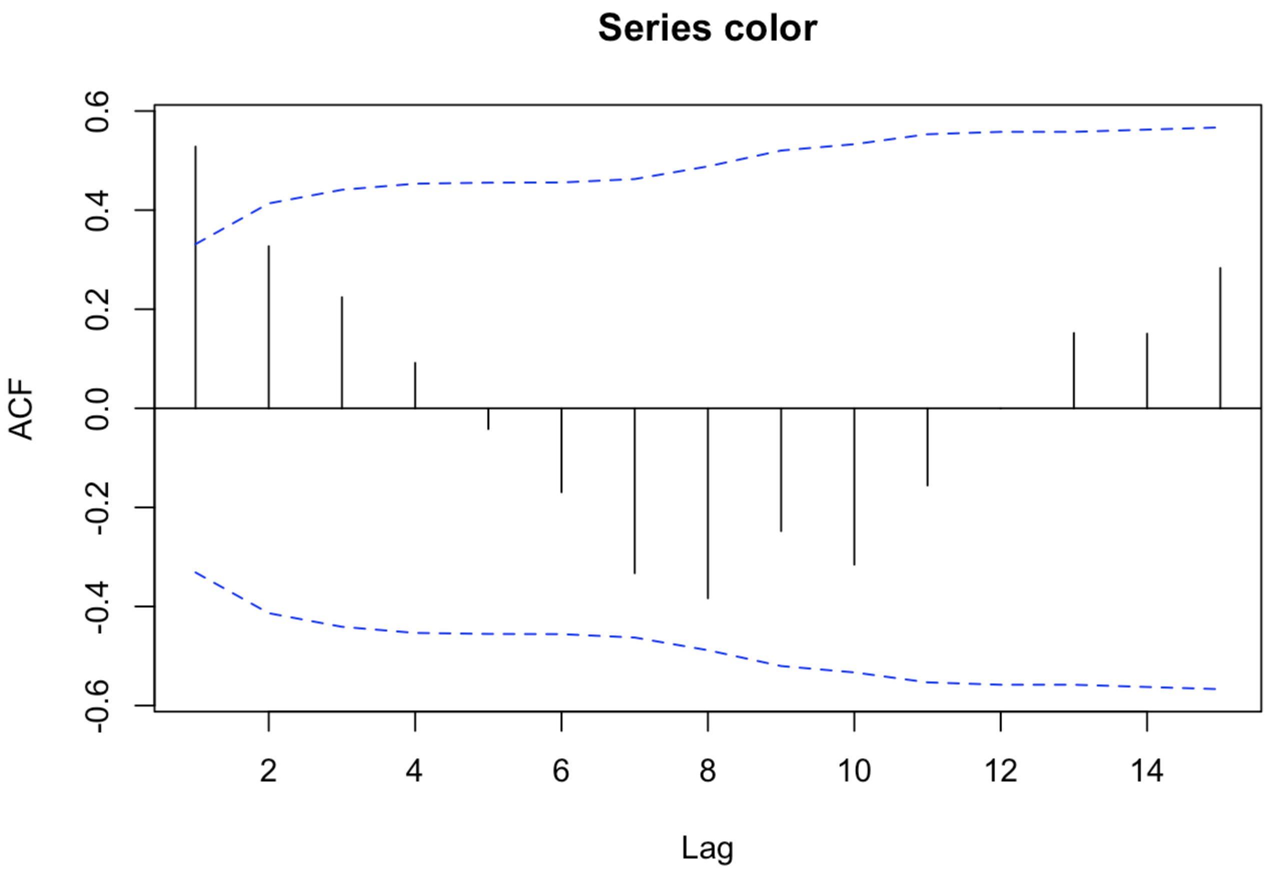

```
# Load the larain dataset
data(larain)

# Create a quantile-quantile (Q-Q) plot of the logarithm of larain
qqnorm(log(larain))

# Add a line to the Q-Q plot to indicate the expected values under normality
qqline(log(larain))
```



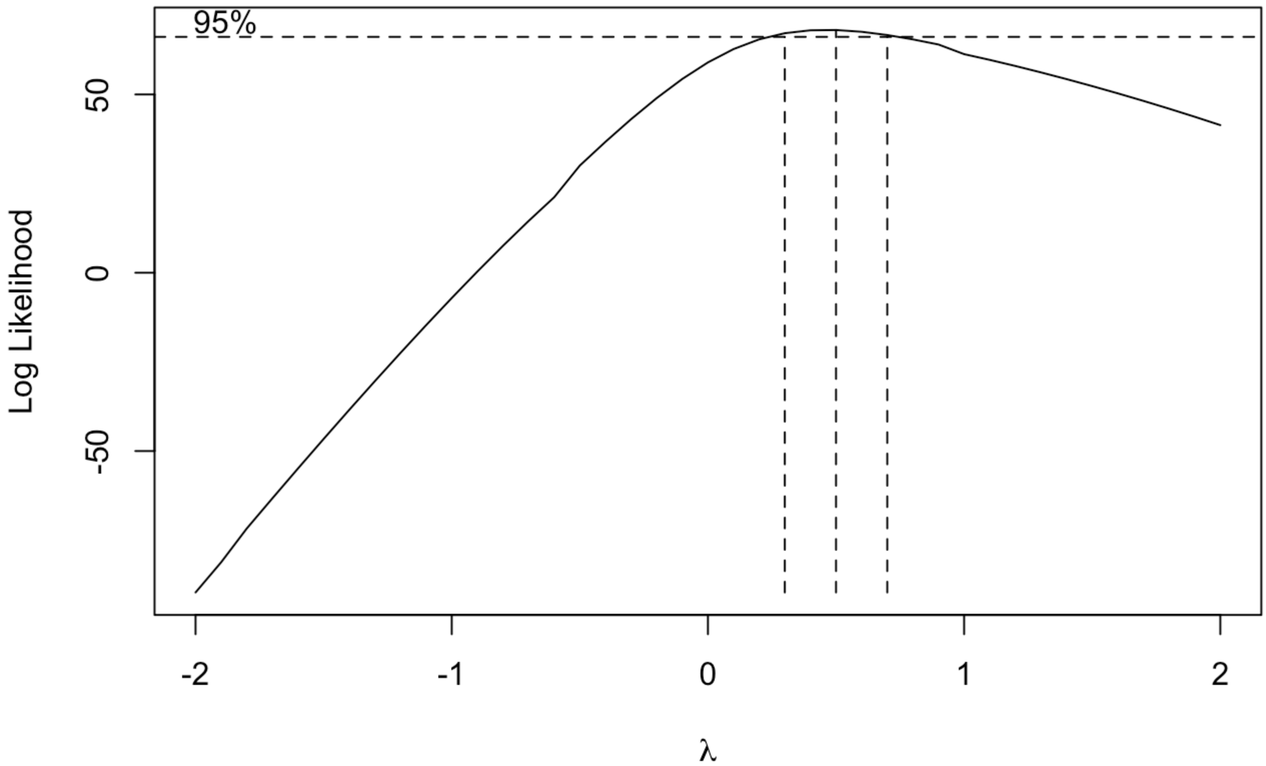

```
# Compute and plot the autocorrelation function (ACF) of the logarithm of larain
# xaxp=c(0,20,10) sets the x-axis range from 0 to 20 with 10 tick marks
acf(log(larain), xaxp=c(0,20,10))
```



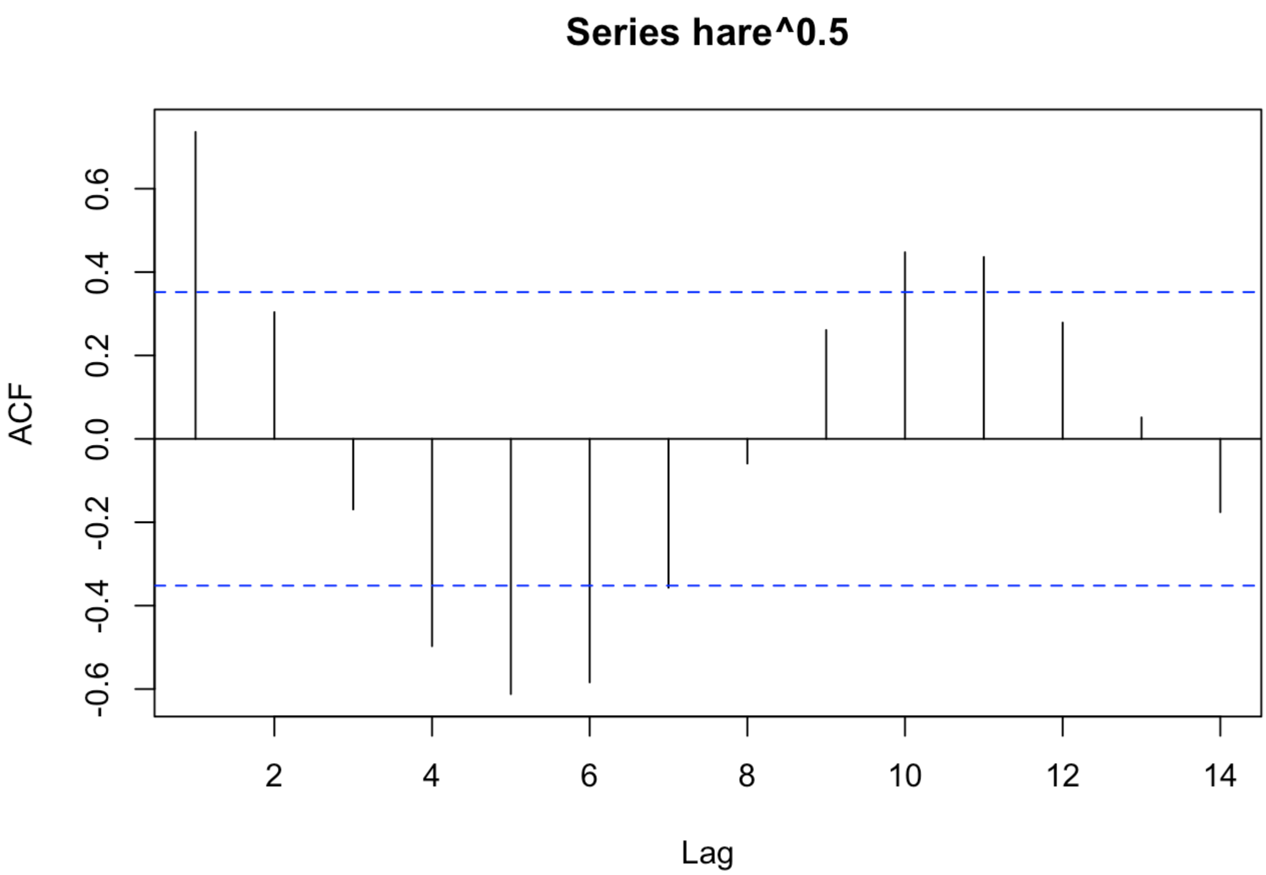

```
# Load the color dataset
data(color)

# Compute and plot the autocorrelation function (ACF) of the color dataset
# ci.type='ma' adds confidence intervals for a moving average (MA) process
acf(color, ci.type='ma')
```



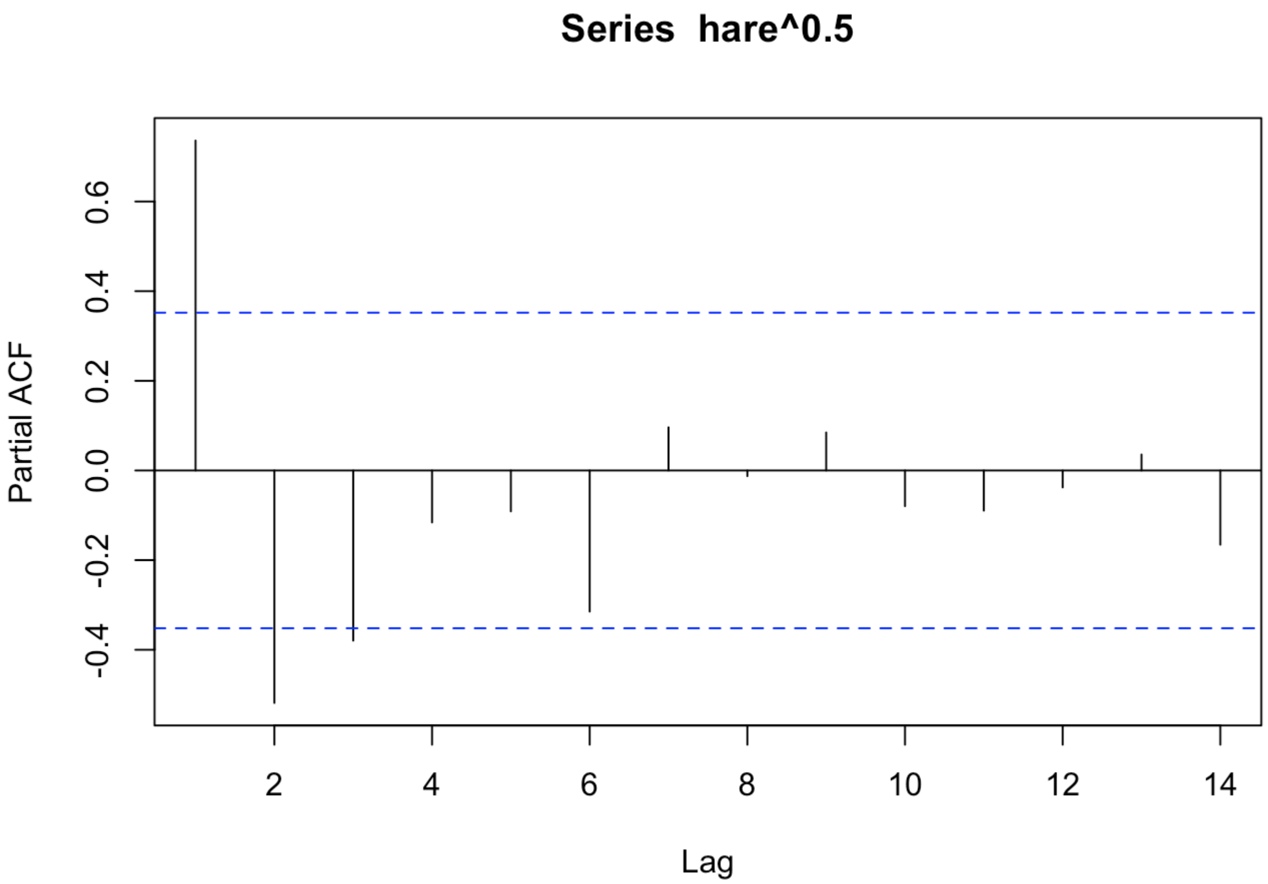

```
# Load the hare dataset
data(hare)

# Apply the Box-Cox transformation to the hare dataset
BoxCox.ar(hare)
```




```
# Compute and plot the autocorrelation function (ACF) of the square root of hare
acf(hare^.5)
```



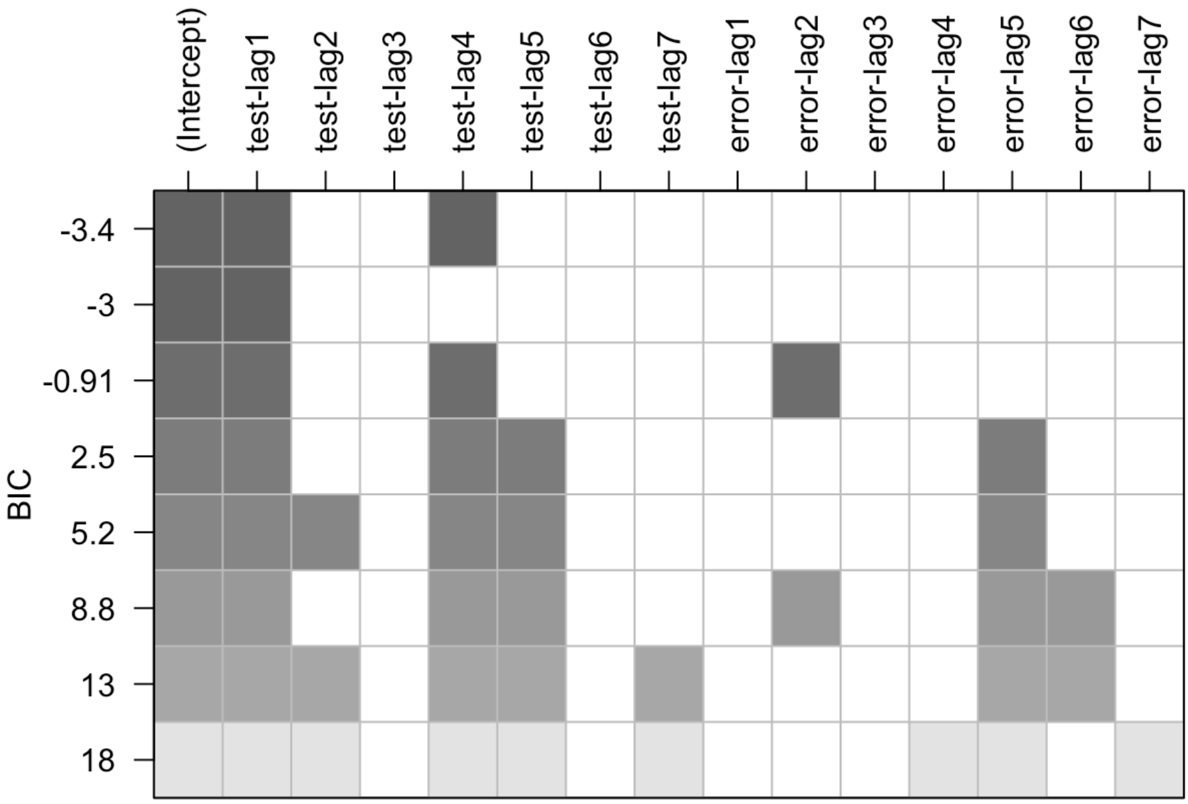
```
# Compute and plot the partial autocorrelation function (PACF) of the square root of
hare
pacf(hare^.5)
```



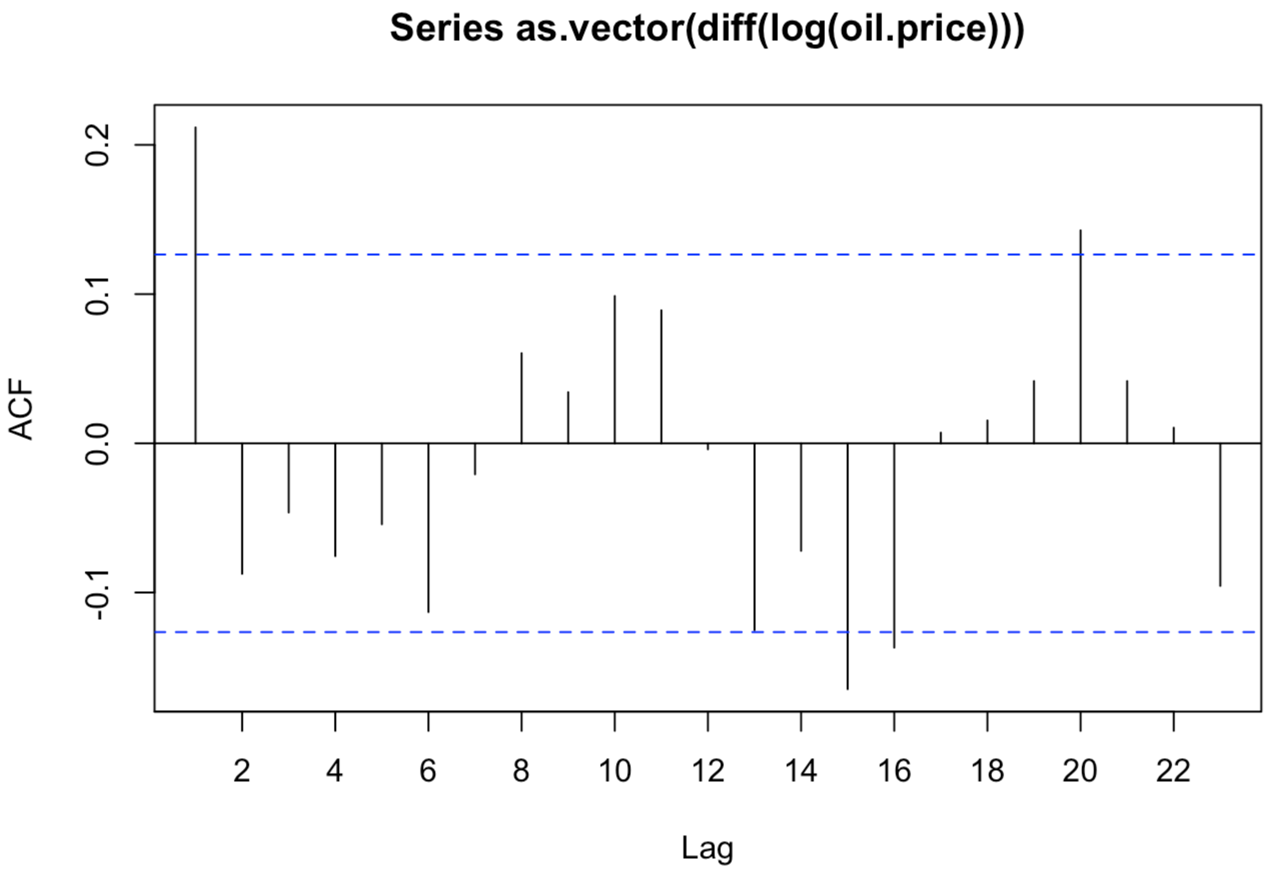

```
 # Take the first difference of the logarithm of the oil.price dataset,
# and compute and plot the extended autocorrelation function (EACF)
eacf(diff(log(oil.price)))
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o o o o o o o o o o  o  o  o
## 1 x x o o o o o o o o x  o  o  o
## 2 o x o o o o o o o o o  o  o  o
## 3 o x o o o o o o o o o  o  o  o
## 4 o x x o o o o o o o o  o  o  o
## 5 o x o x o o o o o o o  o  o  o
## 6 o x o x o o o o o o o  o  o  o
## 7 x x o x o o o o o o o  o  o  o
```

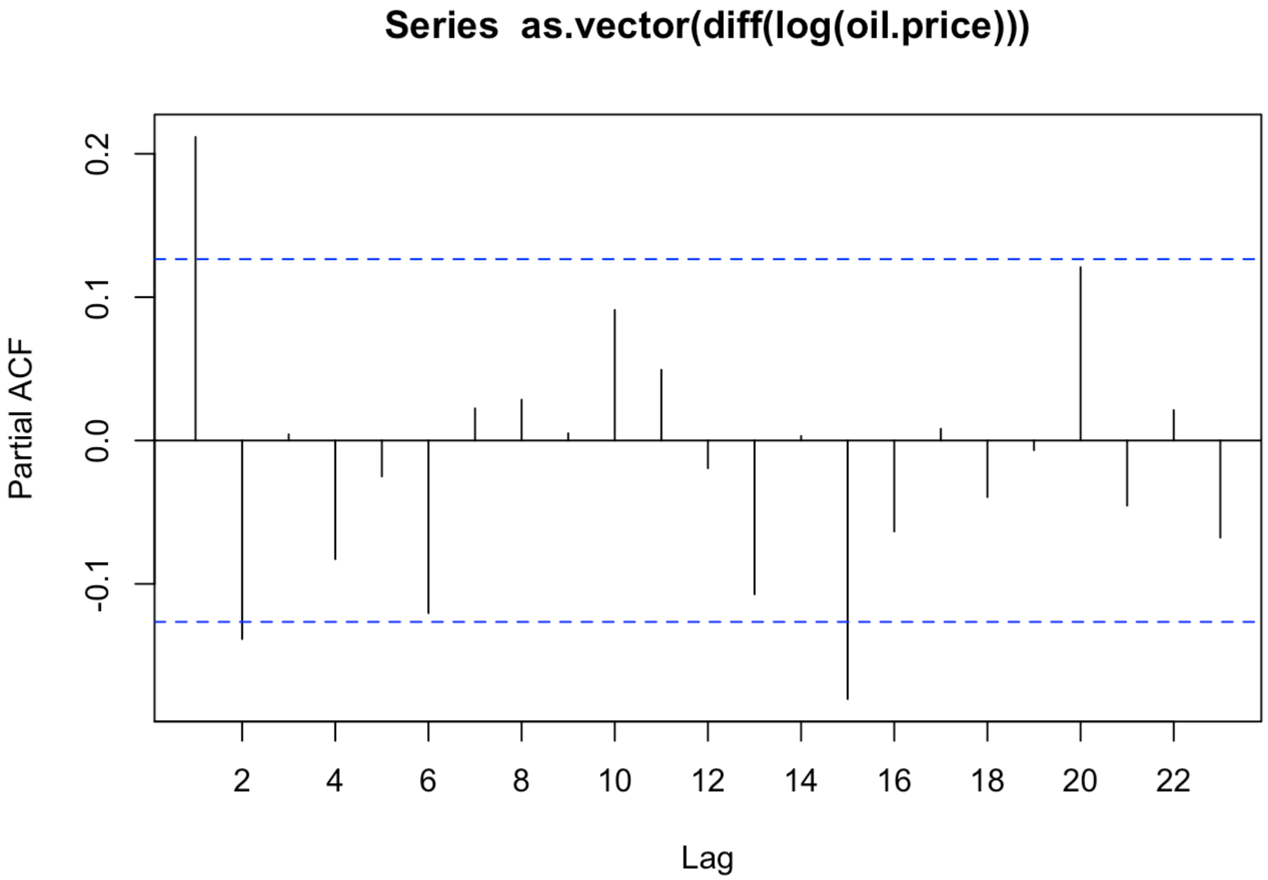
```
# Fit an ARMA model to the first difference of the logarithm of the oil.price dataset
# using subset selection
res = armasubsets(y=diff(log(oil.price)), nar=7, nma=7, y.name='test', ar.method='ol
s')

# Plot the subset selection results
plot(res)
```




```
# Compute and plot the autocorrelation function (ACF) of the first difference
# of the logarithm of the oil.price dataset
# xaxp=c(0,22,11) sets the x-axis range from 0 to 22 with 11 tick marks
acf(as.vector(diff(log(oil.price))), xaxp=c(0,22,11))
```




```
# Compute and plot the partial autocorrelation function (PACF) of the first differenc
e
# of the logarithm of the oil.price dataset
# xaxp=c(0,22,11) sets the x-axis range from 0 to 22 with 11 tick marks
pacf(as.vector(diff(log(oil.price))), xaxp=c(0,22,11))
```




```
# Fit an autoregressive (AR) model to the differenced rwalk series
# Specify the maximum order of the AR model using order.max=1 for an AR(1) model
ar(diff(rwalk), order.max=1)
```

```
##
## Call:
## ar(x = diff(rwalk), order.max = 1)
##
## Coefficients:
##       1
## -0.2078
##
## Order selected 1  sigma^2 estimated as  0.9995
```

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
# Perform Augmented Dickey-Fuller (ADF) tests on the 'rwalk' series with different la
g selection methods and differencing schemes

# ADF test with lag selection using modes 1 to 8 and Pmax=8, without seasonal differe
ncing
adf.test(rwalk, k = 8)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rwalk
## Dickey-Fuller = -2.2892, Lag order = 8, p-value = 0.4579
## alternative hypothesis: stationary
```

```
# ADF test with lag selection using modes 1 to 8 and Pmax=8, with first-order seasona
l differencing
adf.test(diff(rwalk, differences = 1), k = 8)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(rwalk, differences = 1)
## Dickey-Fuller = -3.6749, Lag order = 8, p-value = 0.0346
## alternative hypothesis: stationary
```

```
# ADF test with lag selection using Pmax=0 (no lag selection), with first-order seaso
nal differencing
adf.test(diff(rwalk, differences = 1), k = 0)
```

```
## Warning in adf.test(diff(rwalk, differences = 1), k = 0): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(rwalk, differences = 1)
## Dickey-Fuller = -9.1723, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

Chapter 7 Commands

```
# Below is a function that computes the method of moments estimator of
# the MA(1) coefficient of an MA(1) model.
estimate.ma1.mom=function(x){r=acf(x,plot=F)$acf[1]; if (abs(r)<0.5)
return((-1+sqrt(1-4*r^2))/(2*r)) else return(NA)}
```

```
# Load the TSA package for time series analysis
library(TSA)

# Load the ma1.2.s dataset
data(ma1.2.s)

# Estimate the parameters of a first-order moving average (MA(1)) model using the met
hod of moments
estimate.ma1.mom(ma1.2.s)
```

```
## [1] -0.5554273
```

```
# Load the TSA package for time series analysis
library(TSA)

# Load the ma1.1.s dataset
data(ma1.1.s)

# Estimate the parameters of a first-order moving average (MA(1)) model using CSS est
imation
fit <- arima(ma1.1.s, order=c(0,0,1), method="CSS")

# Display the estimated parameters
fit
```

```
##
## Call:
## arima(x = ma1.1.s, order = c(0, 0, 1), method = "CSS")
##
## Coefficients:
##           ma1  intercept
##        -0.958     0.0212
## s.e.   0.038     0.0051
##
## sigma^2 estimated as 1.2:  part log likelihood = -181.23
```

```
# Set the seed for reproducibility
set.seed(1234)

# Generate a time series using an MA(1) model
ma1.3.s <- arima.sim(list(ma=0.9), n=60)

# Estimate the parameters of a first-order moving average (MA(1)) model using CSS est
imation
fit <- arima(ma1.3.s, order=c(0,0,1), method="CSS")

# Display the estimated parameters
fit
```

```
##
## Call:
## arima(x = ma1.3.s, order = c(0, 0, 1), method = "CSS")
##
## Coefficients:
##          ma1  intercept
##        0.797    -0.8429
## s.e.  0.108     0.2238
##
## sigma^2 estimated as 0.9351:  part log likelihood = -83.12
```

```
# Set the seed for reproducibility
set.seed(1234)

# Generate a time series using an MA(1) model with ma=-0.5
ma1.4.s <- arima.sim(list(ma=-0.5), n=60)

# Estimate the parameters of a first-order moving average (MA(1)) model using CSS est
imation
fit <- arima(ma1.4.s, order=c(0,0,1), method="CSS")

# Display the estimated parameters
fit
```

```
##
## Call:
## arima(x = ma1.4.s, order = c(0, 0, 1), method = "CSS")
##
## Coefficients:
##           ma1  intercept
##       -0.4560    -0.1951
## s.e.   0.1084     0.0667
##
## sigma^2 estimated as 0.8788:  part log likelihood = -81.26
```

```
# Fit a first-order moving average (MA(1)) model to the ma1.4.s series without includ
ing a mean term
arima(ma1.4.s, order=c(0,0,1), method='CSS', include.mean=FALSE)
```

```
##
## Call:
## arima(x = ma1.4.s, order = c(0, 0, 1), include.mean = FALSE, method = "CSS")
##
## Coefficients:
##           ma1
##       -0.3356
## s.e.   0.1009
##
## sigma^2 estimated as 0.9776:  part log likelihood = -84.46
```

```
# Load the ar1.s dataset
data(ar1.s)

# Fit an autoregressive (AR(1)) model to the ar1.s dataset using the Yule-Walker meth
od
ar(ar1.s, order.max=1, AIC=FALSE, method='yw')
```

```
##
## Call:
## ar(x = ar1.s, order.max = 1, method = "yw", AIC = FALSE)
##
## Coefficients:
##      1
## 0.8314
##
## Order selected 1  sigma^2 estimated as  1.382
```

```
# Load the ar1.2.s dataset
data(ar1.2.s)

# Fit an autoregressive (AR(1)) model to the ar1.2.s dataset using the Yule-Walker me
thod
ar(ar1.2.s, order.max=1, AIC=FALSE, method='yw')
```

```
##
## Call:
## ar(x = ar1.2.s, order.max = 1, method = "yw", AIC = FALSE)
##
## Coefficients:
##      1
## 0.4699
##
## Order selected 1  sigma^2 estimated as  0.9198
```

```
# Load the ar2.s dataset
data(ar2.s)

# Fit an autoregressive (AR(2)) model to the ar2.s dataset using the Yule-Walker meth
od
ar(ar2.s, order.max=2, AIC=FALSE, method='yw')
```

```
##
## Call:
## ar(x = ar2.s, order.max = 2, method = "yw", AIC = FALSE)
##
## Coefficients:
##       1        2
##  1.4694  -0.7646
##
## Order selected 2  sigma^2 estimated as  1.051
```

```
data(ar1.s)
ar(ar1.s,order.max=1,AIC=F,method='yw') # method of moments
```

```
##
## Call:
## ar(x = ar1.s, order.max = 1, method = "yw", AIC = F)
##
## Coefficients:
##      1
## 0.8314
##
## Order selected 1  sigma^2 estimated as  1.382
```

```
ar(ar1.s,order.max=1,AIC=F,method='ols') # conditional sum of squares
```

```
##
## Call:
## ar(x = ar1.s, order.max = 1, method = "ols", AIC = F)
##
## Coefficients:
##     1
## 0.857
##
## Intercept: 0.02499 (0.1308)
##
## Order selected 1  sigma^2 estimated as  1.008
```

```
ar(ar1.s,order.max=1,AIC=F,method='mle') # maximum likelihood
```

```
##
## Call:
## ar(x = ar1.s, order.max = 1, method = "mle", AIC = F)
##
## Coefficients:
##      1
## 0.8924
##
## Order selected 1  sigma^2 estimated as  1.041
```

```
# The AIC option is set to be False otherwise the function will choose
# the AR order by minimizing AIC, so that zero order might be chosen.
```

```
data(ar1.2.s)
ar(ar1.2.s,order.max=1,AIC=F,method='yw') # method of moments
```

```
##
## Call:
## ar(x = ar1.2.s, order.max = 1, method = "yw", AIC = F)
##
## Coefficients:
##      1
## 0.4699
##
## Order selected 1  sigma^2 estimated as  0.9198
```

```
ar(ar1.2.s,order.max=1,AIC=F,method='ols') # conditional sum of squares
```

```
##
## Call:
## ar(x = ar1.2.s, order.max = 1, method = "ols", AIC = F)
##
## Coefficients:
##      1
## 0.4731
##
## Intercept: -0.006084 (0.1237)
##
## Order selected 1  sigma^2 estimated as  0.9024
```

```
ar(ar1.2.s,order.max=1,AIC=F,method='mle') # maximum likelihood
```

```
##
## Call:
## ar(x = ar1.2.s, order.max = 1, method = "mle", AIC = F)
##
## Coefficients:
##      1
## 0.4654
##
## Order selected 1  sigma^2 estimated as  0.8875
```

```
data(ar2.s)
ar(ar2.s,order.max=2,AIC=F,method='yw') # method of moments
```

```
##
## Call:
## ar(x = ar2.s, order.max = 2, method = "yw", AIC = F)
##
## Coefficients:
##       1        2
##  1.4694  -0.7646
##
## Order selected 2  sigma^2 estimated as  1.051
```

```
ar(ar2.s,order.max=2,AIC=F,method='ols') # conditional sum of squares
```

```
##
## Call:
## ar(x = ar2.s, order.max = 2, method = "ols", AIC = F)
##
## Coefficients:
##       1        2
##  1.5137  -0.8050
##
## Intercept: 0.02043 (0.08594)
##
## Order selected 2  sigma^2 estimated as  0.8713
```

```
ar(ar2.s,order.max=2,AIC=F,method='mle') # maximum likelihood
```

```
##
## Call:
## ar(x = ar2.s, order.max = 2, method = "mle", AIC = F)
##
## Coefficients:
##       1        2
##  1.5061  -0.7964
##
## Order selected 2  sigma^2 estimated as  0.862
```

```
data(arma11.s)
arima(arma11.s, order=c(1,0,1),method='CSS') # conditional sum of squares
```

```
##
## Call:
## arima(x = arma11.s, order = c(1, 0, 1), method = "CSS")
##
## Coefficients:
##          ar1     ma1  intercept
##       0.5586  0.3669     0.3928
## s.e.  0.1219  0.1564     0.3380
##
## sigma^2 estimated as 1.199:  part log likelihood = -150.98
```

```
arima(arma11.s, order=c(1,0,1),method='ML') # maximum likelihood
```

```
##
## Call:
## arima(x = arma11.s, order = c(1, 0, 1), method = "ML")
##
## Coefficients:
##          ar1     ma1  intercept
##       0.5647  0.3557     0.3216
## s.e.  0.1205  0.1585     0.3358
##
## sigma^2 estimated as 1.197:  log likelihood = -151.33,  aic = 308.65
```

```
data(color)
ar(color,order.max=1,AIC=F,method='yw') # method of moments
```

```
##
## Call:
## ar(x = color, order.max = 1, method = "yw", AIC = F)
##
## Coefficients:
##      1
## 0.5282
##
## Order selected 1  sigma^2 estimated as  27.56
```

```
ar(color,order.max=1,AIC=F,method='ols') # conditional sum of squares
```

```
##
## Call:
## ar(x = color, order.max = 1, method = "ols", AIC = F)
##
## Coefficients:
##      1
## 0.5549
##
## Intercept: 0.1032 (0.8474)
##
## Order selected 1  sigma^2 estimated as  24.38
```

```
ar(color,order.max=1,AIC=F,method='mle') # maximum likelihood
```

```
##
## Call:
## ar(x = color, order.max = 1, method = "mle", AIC = F)
##
## Coefficients:
##      1
## 0.5703
##
## Order selected 1  sigma^2 estimated as  24.83
```

```
# Load the hare dataset
data(hare)

# Fit an autoregressive (AR) model of order 3 to the square root of the hare dataset
# The ARIMA model is specified as ARIMA(p,d,q) where p is the order of the autoregres
sive (AR) part,
# d is the order of differencing, and q is the order of the moving average (MA) part
arima(sqrt(hare), order=c(3,0,0))
```

```
##
## Call:
## arima(x = sqrt(hare), order = c(3, 0, 0))
##
## Coefficients:
##          ar1      ar2      ar3  intercept
##       1.0519  -0.2292  -0.3931     5.6923
## s.e.  0.1877   0.2942   0.1915     0.3371
##
## sigma^2 estimated as 1.066:  log likelihood = -46.54,  aic = 101.08
```

```
data(oil.price)
arima(log(oil.price),order=c(0,1,1),method='CSS') # conditional sum of squares
```

```
##
## Call:
## arima(x = log(oil.price), order = c(0, 1, 1), method = "CSS")
##
## Coefficients:
##          ma1
##       0.2731
## s.e.  0.0681
##
## sigma^2 estimated as 0.006731:  part log likelihood = 259.58
```

```
arima(log(oil.price),order=c(0,1,1),method='ML') # maximum likelihood
```

```
##
## Call:
## arima(x = log(oil.price), order = c(0, 1, 1), method = "ML")
##
## Coefficients:
##          ma1
##       0.2956
## s.e.  0.0693
##
## sigma^2 estimated as 0.006689:  log likelihood = 260.29,  aic = -518.58
```

```
# Fit an ARIMA(3,0,0) model to the square root of the hare dataset, including the mea
n term
res <- arima(sqrt(hare), order=c(3,0,0), include.mean=TRUE)

# Set the seed for reproducibility
set.seed(12345)

# Method I: Conditional bootstrap with normal distribution assumption
coefm.cond.norm <- arima.boot(res, cond.boot=TRUE, is.normal=TRUE, B=1000, init=sqrt
(hare))
signif(apply(coefm.cond.norm, 2, function(x) quantile(x, c(0.025, 0.975), na.rm=TRU
E)), 3)
```

```
##         ar1    ar2    ar3 noise var
## 2.5%  0.714 -0.505 -0.790     0.916
## 97.5% 1.290  0.390 -0.153     2.040
```

```
# Method II: Conditional bootstrap without normal distribution assumption
coefm.cond.replace <- arima.boot(res, cond.boot=TRUE, is.normal=FALSE, B=1000, init=s
qrt(hare))
signif(apply(coefm.cond.replace, 2, function(x) quantile(x, c(0.025, 0.975), na.rm=TR
UE)), 3)
```

```
##         ar1    ar2    ar3 noise var
## 2.5%  0.715 -0.549 -0.792     0.867
## 97.5% 1.320  0.382 -0.141     2.000
```

```
# Method III: Non-conditional bootstrap with normal distribution assumption
coefm.norm <- arima.boot(res, cond.boot=FALSE, is.normal=TRUE, ntrans=100, B=1000, in
it=sqrt(hare))
signif(apply(coefm.norm, 2, function(x) quantile(x, c(0.025, 0.975), na.rm=TRUE)), 3)
```

```
##         ar1    ar2     ar3 noise var
## 2.5%  0.718 -0.736 -0.6570     0.506
## 97.5% 1.370  0.181 -0.0125     1.550
```

```
# Method IV: Non-conditional bootstrap without normal distribution assumption
coefm.replace <- arima.boot(res, cond.boot=FALSE, is.normal=FALSE, ntrans=100, B=100
0, init=sqrt(hare))
signif(apply(coefm.replace, 2, function(x) quantile(x, c(0.025, 0.975), na.rm=TRUE)),
3)
```

```
##         ar1    ar2     ar3 noise var
## 2.5%  0.709 -0.840 -0.6340      0.52
## 97.5% 1.440  0.165  0.0255      1.60
```

```
#retrieving the dimensions
dim(coefm.replace)
```

```
## [1] 1000    4
```

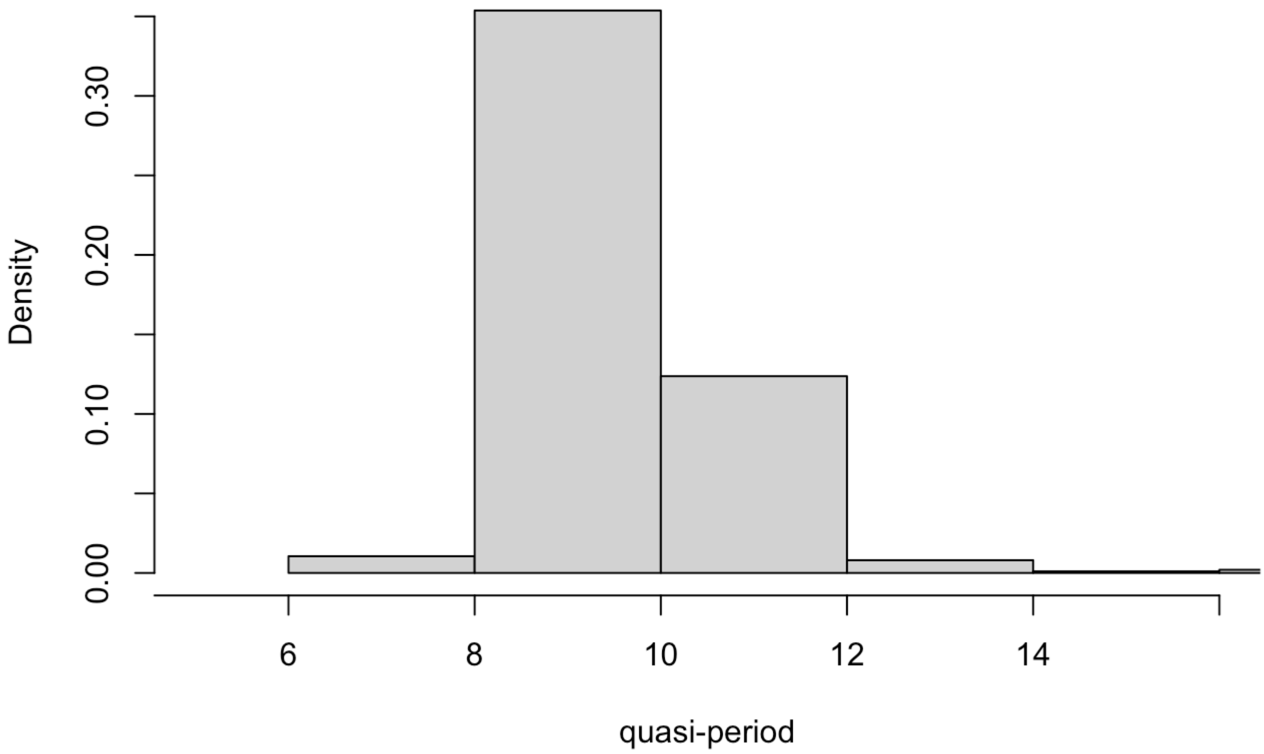
```
# Calculate the period for each bootstrap series in coefm.replace
period.replace <- apply(coefm.replace, 1, function(x) {
  roots <- polyroot(c(1, -x[1:3]))
  # Find the complex root with the smallest magnitude
  min1 <- 1e+9
  rootc <- NA
  for (root in roots) {
    if (abs(Im(root)) < 1e-10) next
    if (Mod(root) < min1) {min1 <- Mod(root); rootc <- root}
  }
  if (is.na(rootc)) period <- NA else period <- 2*pi/abs(Arg(rootc))
  period
})

# Count the number of bootstrap series that do not admit a well-defined quasi-period
sum(is.na(period.replace))
```

```
## [1] 2
```

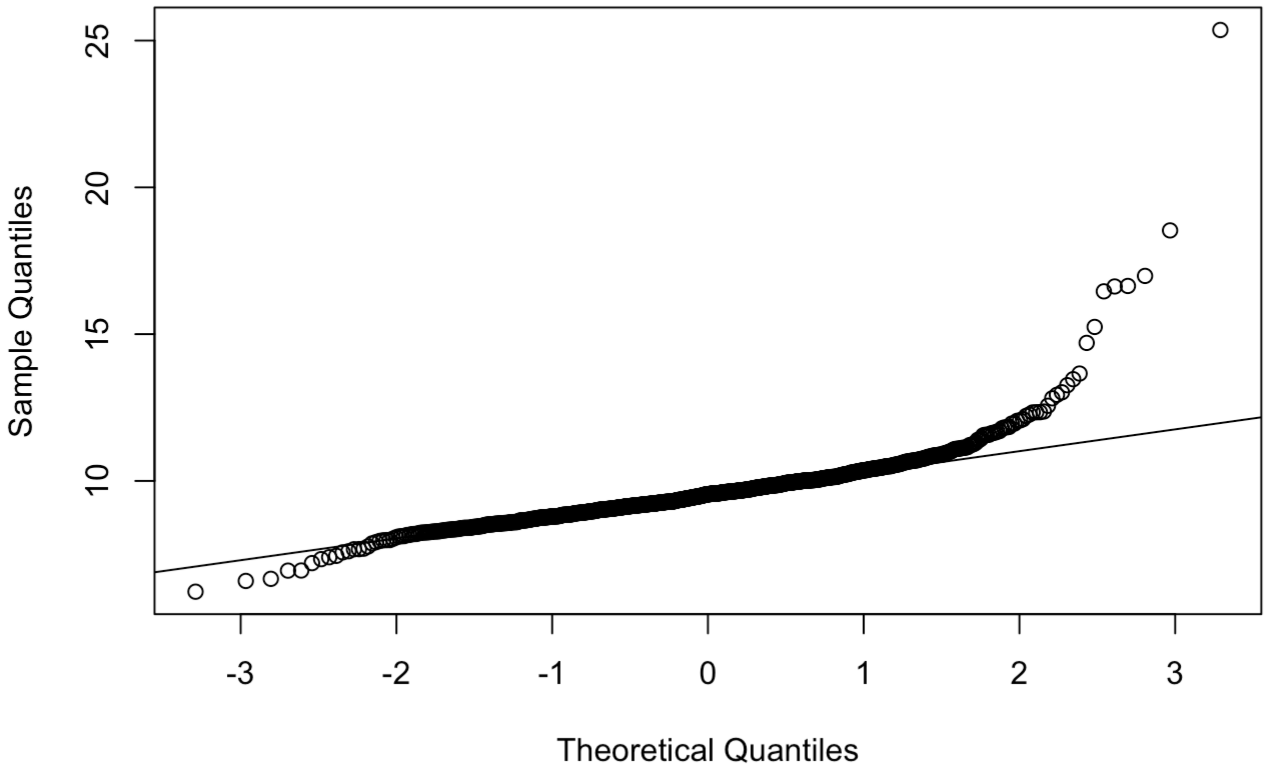
```
# Compute and display the 95% confidence interval for the period
quantile(period.replace, c(0.025, 0.975), na.rm=TRUE)
```

```
##      2.5%     97.5%
##  8.121142 11.953447
```

```
# Create a histogram of the quasi-periods with customized plot window size and point
size
hist(period.replace, prob=TRUE, main="", xlab="quasi-period", axes=FALSE, xlim=c(5,1
6))
axis(2)
axis(1, c(4,6,8,10,12,14,16), c(4,6,8,10,12,14,NA))
```




```
qqnorm(period.replace,main="") #Normal Q-Q Plot for the Bootstrap Quasi-period Estima
tes")
qqline(period.replace)
```

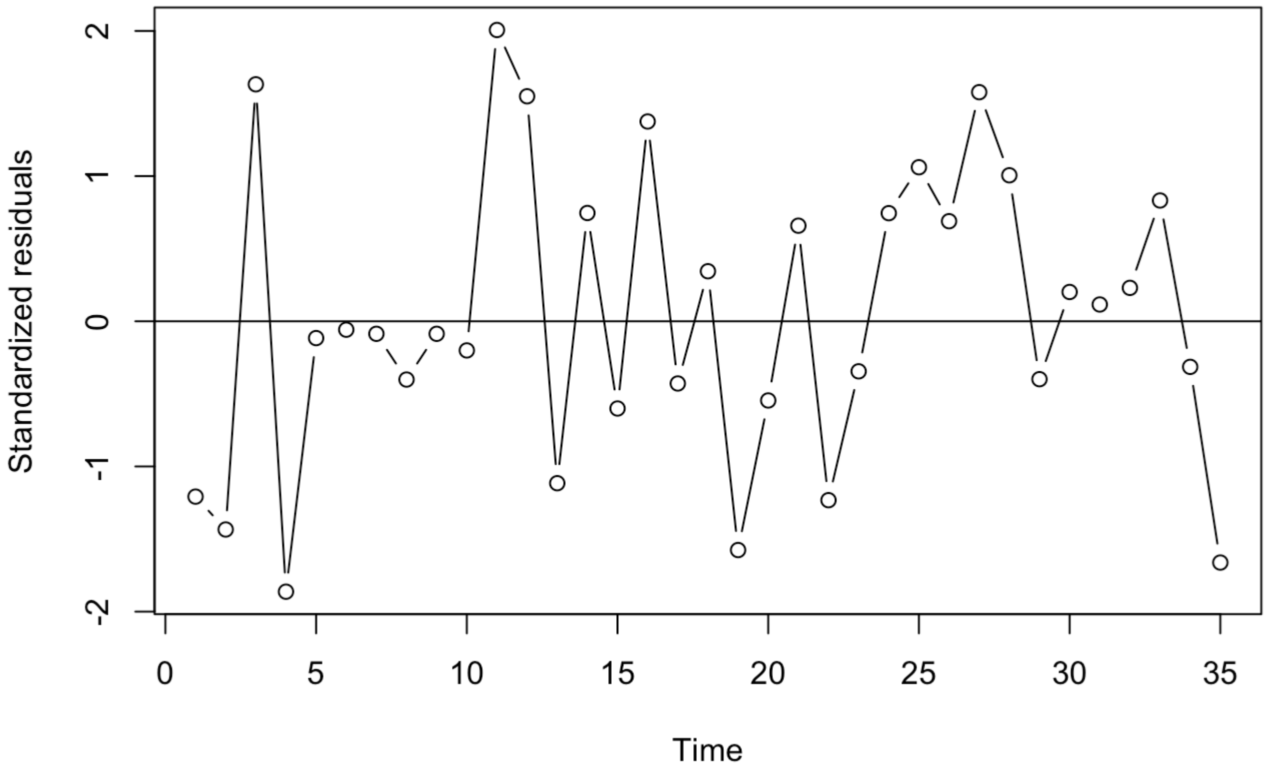


Chapter 8 Commands

```
# Create a plot with a customized plot window size and point size
data(color)
# Fit an ARIMA(1,0,0) model to the color dataset
m1.color <- arima(color, order=c(1,0,0))
# Display the model summary
m1.color
```

```
## 
## Call:
## arima(x = color, order = c(1, 0, 0))
## 
## Coefficients:
##          ar1  intercept
##       0.5705    74.3293
## s.e.  0.1435     1.9151
## 
## sigma^2 estimated as 24.83:  log likelihood = -106.07,  aic = 216.15
```

```
# Plot the standardized residuals
plot(rstandard(m1.color), ylab='Standardized residuals', type='b')
# Add a horizontal line at y=0 for reference
abline(h=0)
```




```
data(hare)
# Fit an ARIMA(3,0,0) model to the square root of the hare dataset
m1.hare <- arima(sqrt(hare), order=c(3,0,0))
# Display the model summary
m1.hare
```

```
##
## Call:
## arima(x = sqrt(hare), order = c(3, 0, 0))
##
## Coefficients:
##          ar1      ar2      ar3  intercept
##       1.0519  -0.2292  -0.3931     5.6923
## s.e.  0.1877   0.2942   0.1915     0.3371
##
## sigma^2 estimated as 1.066:  log likelihood = -46.54,  aic = 101.08
```

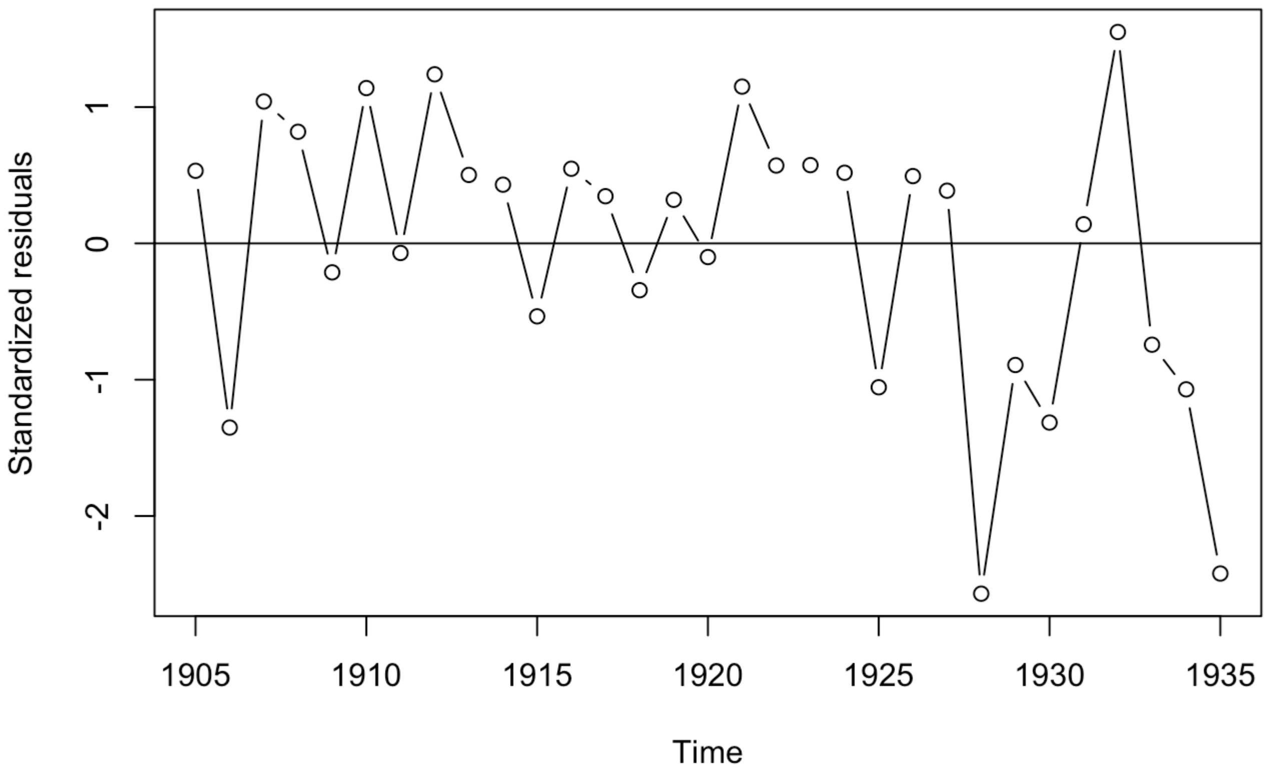
```
# Fit an ARIMA(3,0,0) model to the square root of the hare dataset with fixed paramet
ers
m2.hare <- arima(sqrt(hare), order=c(3,0,0), fixed=c(NA,0,NA,NA))
```

```
## Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg = xreg,
## : some AR parameters were fixed: setting transform.pars = FALSE
```

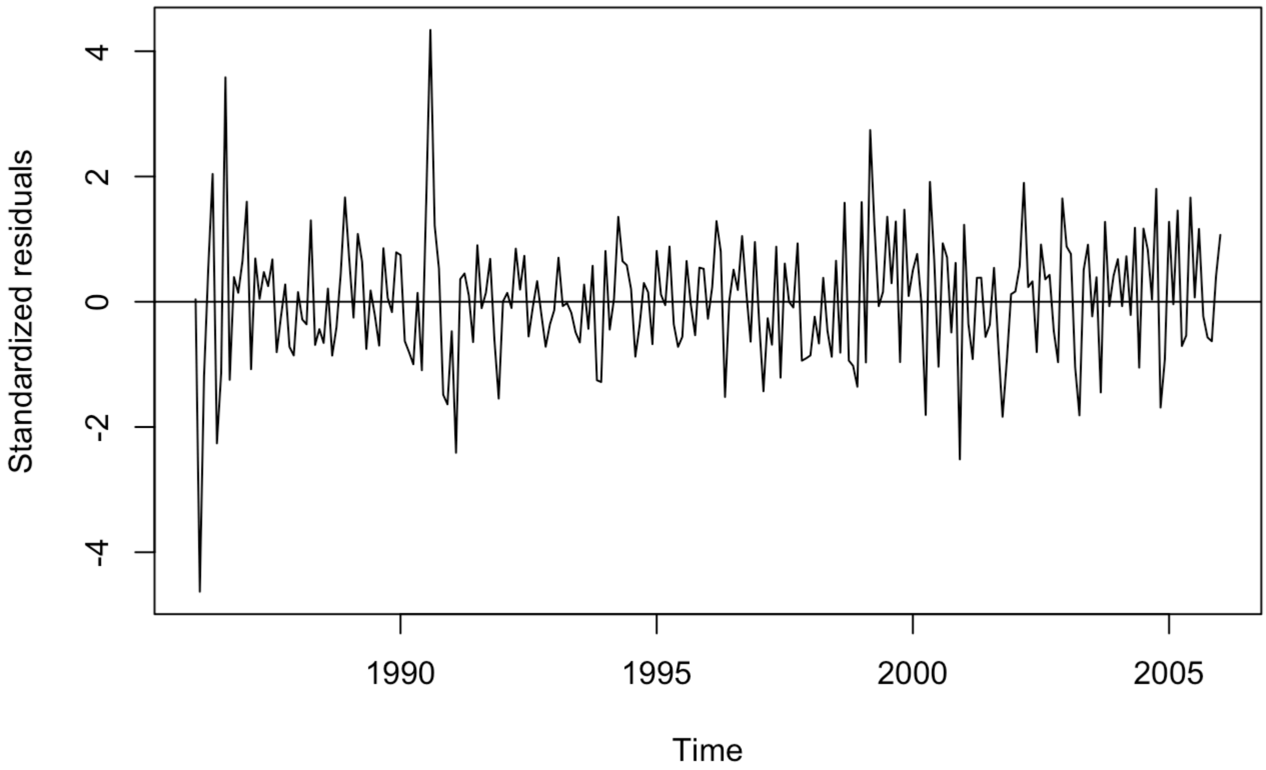
```
# Display the model summary
m2.hare
```

```
##
## Call:
## arima(x = sqrt(hare), order = c(3, 0, 0), fixed = c(NA, 0, NA, NA))
##
## Coefficients:
##          ar1  ar2      ar3  intercept
##       0.9190    0  -0.5313     5.6889
## s.e.  0.0791    0   0.0697     0.3179
##
## sigma^2 estimated as 1.088:  log likelihood = -46.85,  aic = 99.69
```

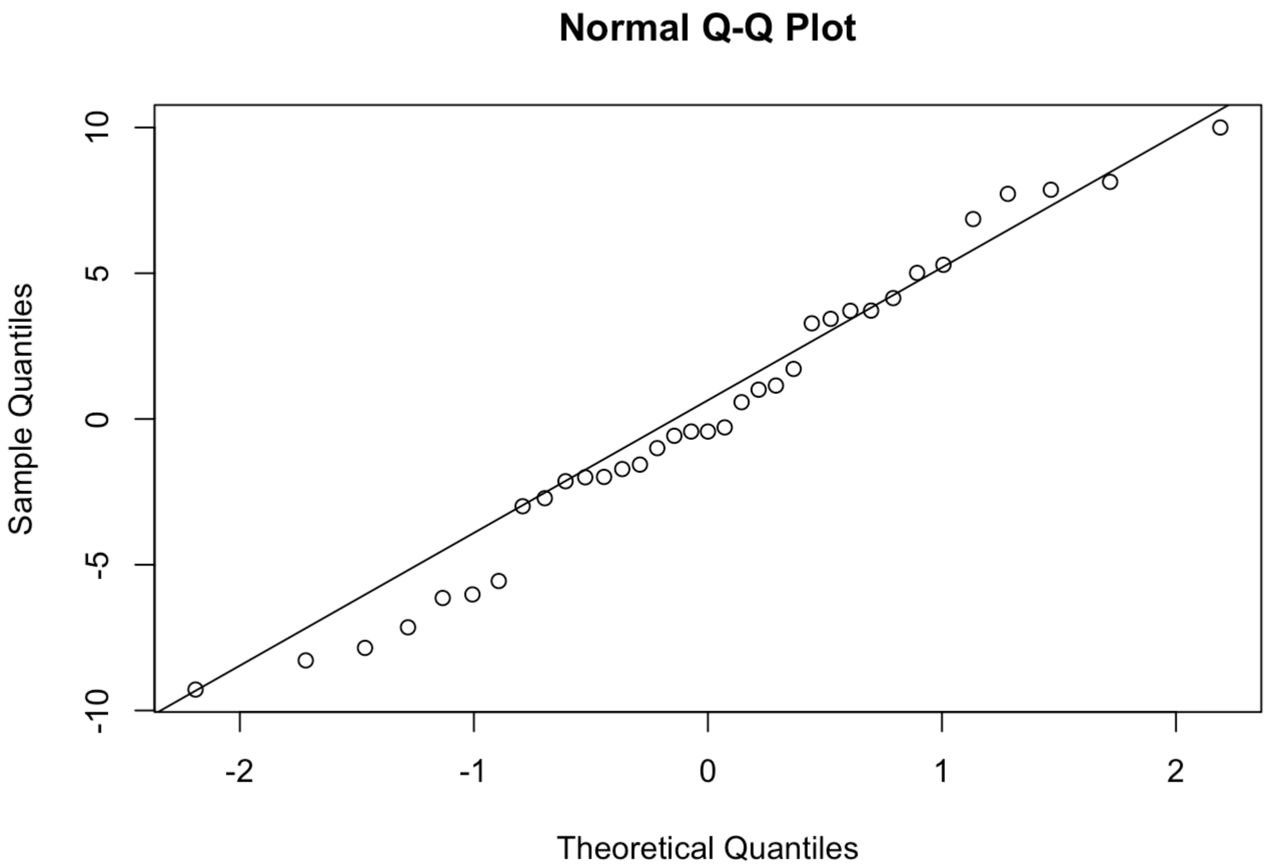
```
# Note that the intercept term is actually the mean in the centered form of the ARMA model,
# i.e. if y(t) = sqrt(hare) - intercept, then the model is y(t) = 0.919*y(t-1) - 0.5313*y(t-3) + e(t)
# Plot the standardized residuals of the second model
plot(rstandard(m2.hare), ylab='Standardized residuals', type='b')
# Add a horizontal line at y=0 for reference
abline(h=0)
```



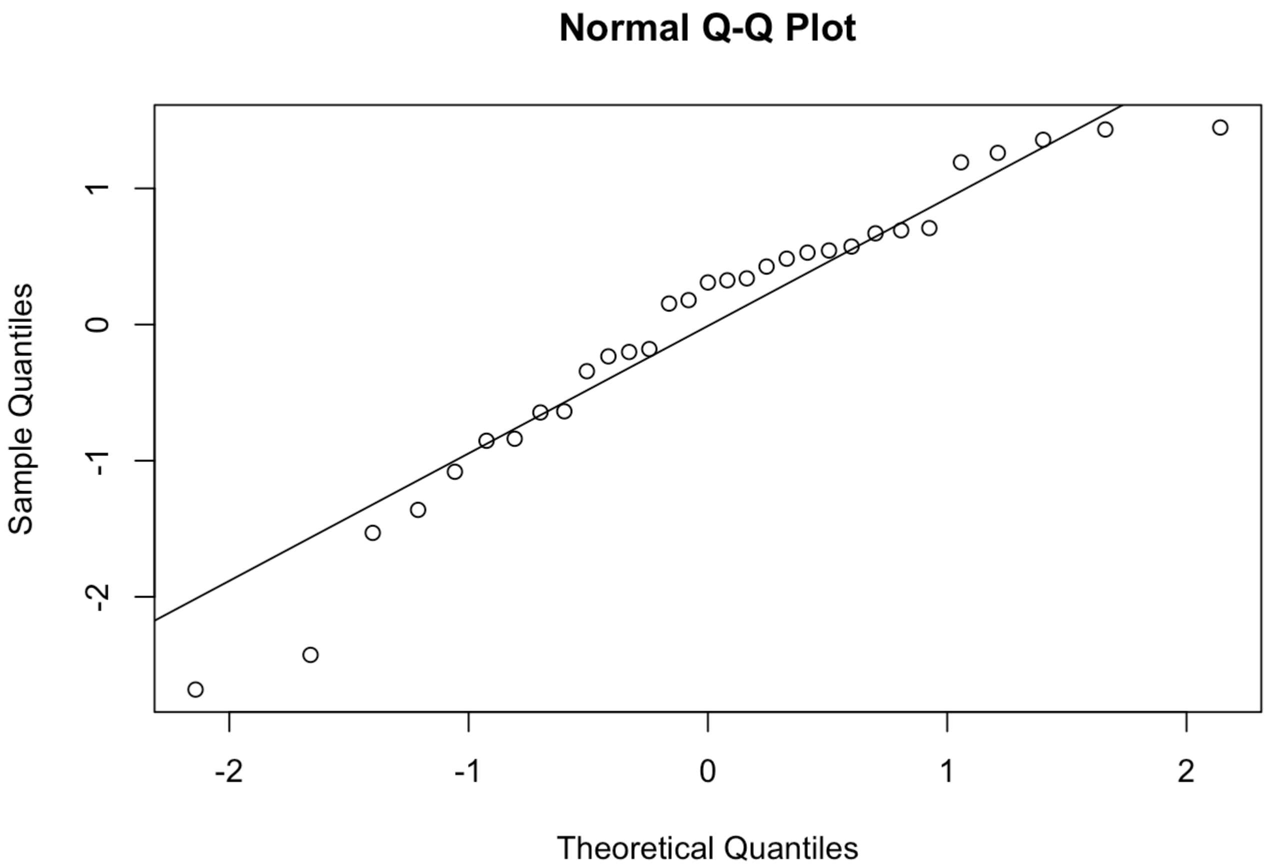

```
data(oil.price)
# Fit an ARIMA(0,1,1) model to the log-transformed oil price dataset
m1.oil <- arima(log(oil.price), order=c(0,1,1))
# Plot the standardized residuals of the model
plot(rstandard(m1.oil), ylab='Standardized residuals', type='l')
# Add a horizontal line at y=0 for reference
abline(h=0)
```



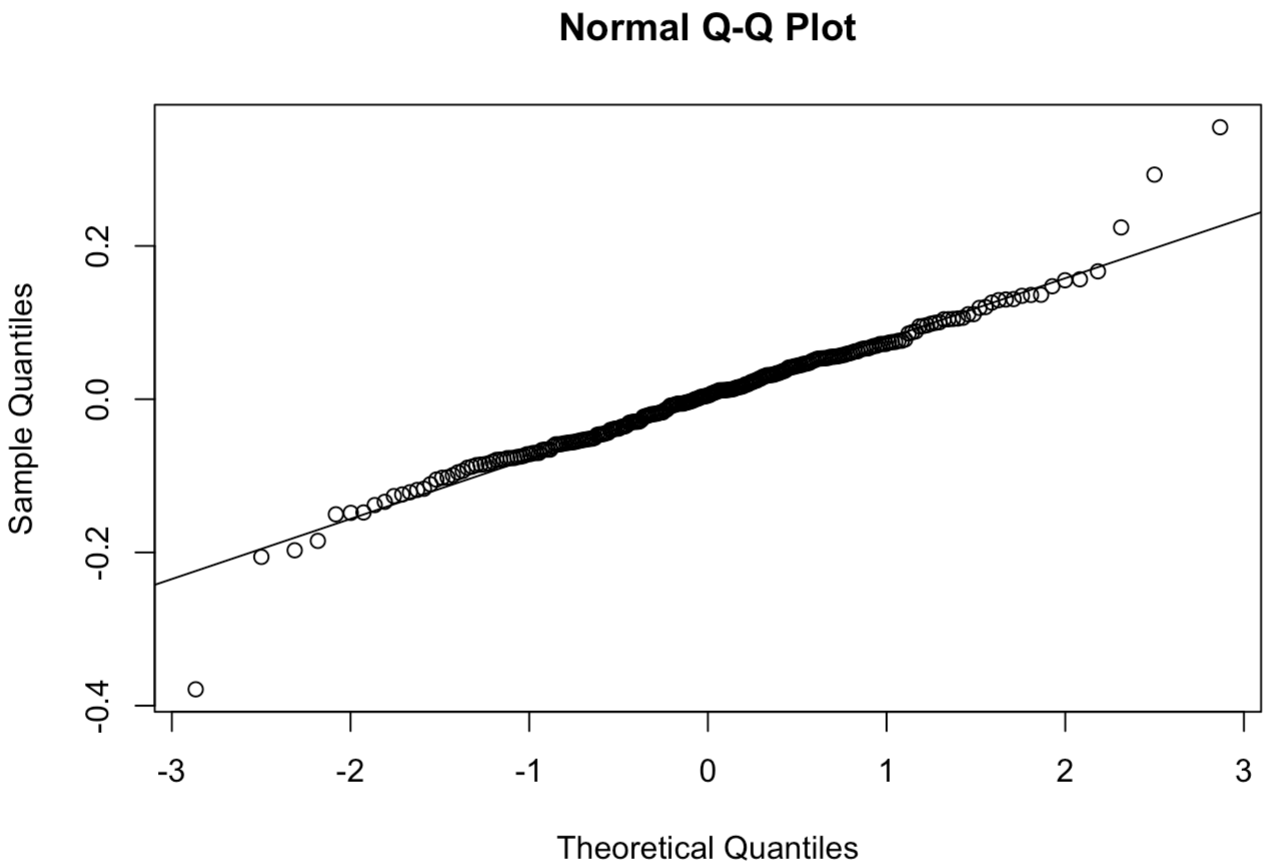

```
# Create a QQ plot of the residuals from the m1.color model
qqnorm(residuals(m1.color))
# Add a line to the QQ plot to show the expected distribution of the residuals
qqline(residuals(m1.color))
```



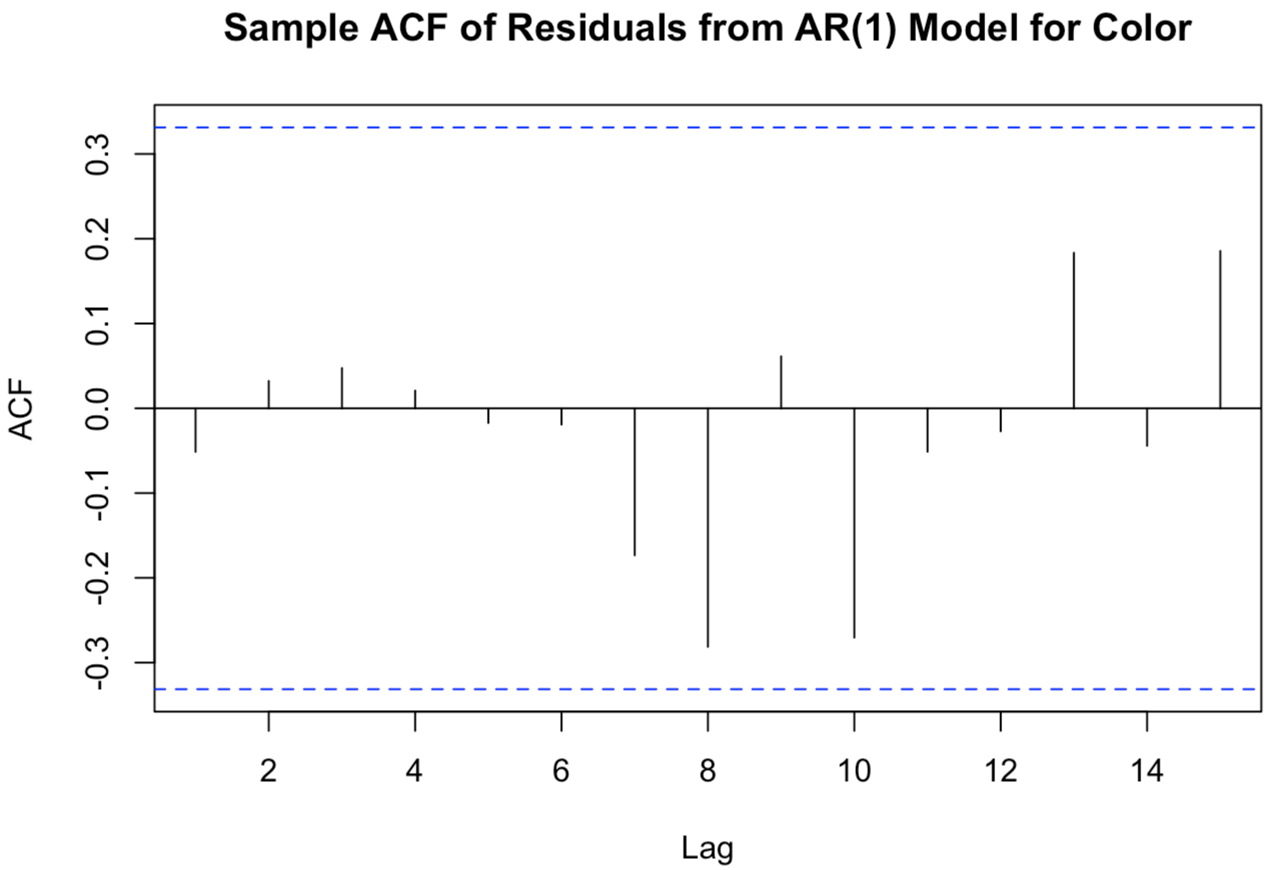

```
# Create a QQ plot of the residuals from the m1.hare model
qqnorm(residuals(m1.hare))
# Add a line to the QQ plot to show the expected distribution of the residuals
qqline(residuals(m1.hare))
```



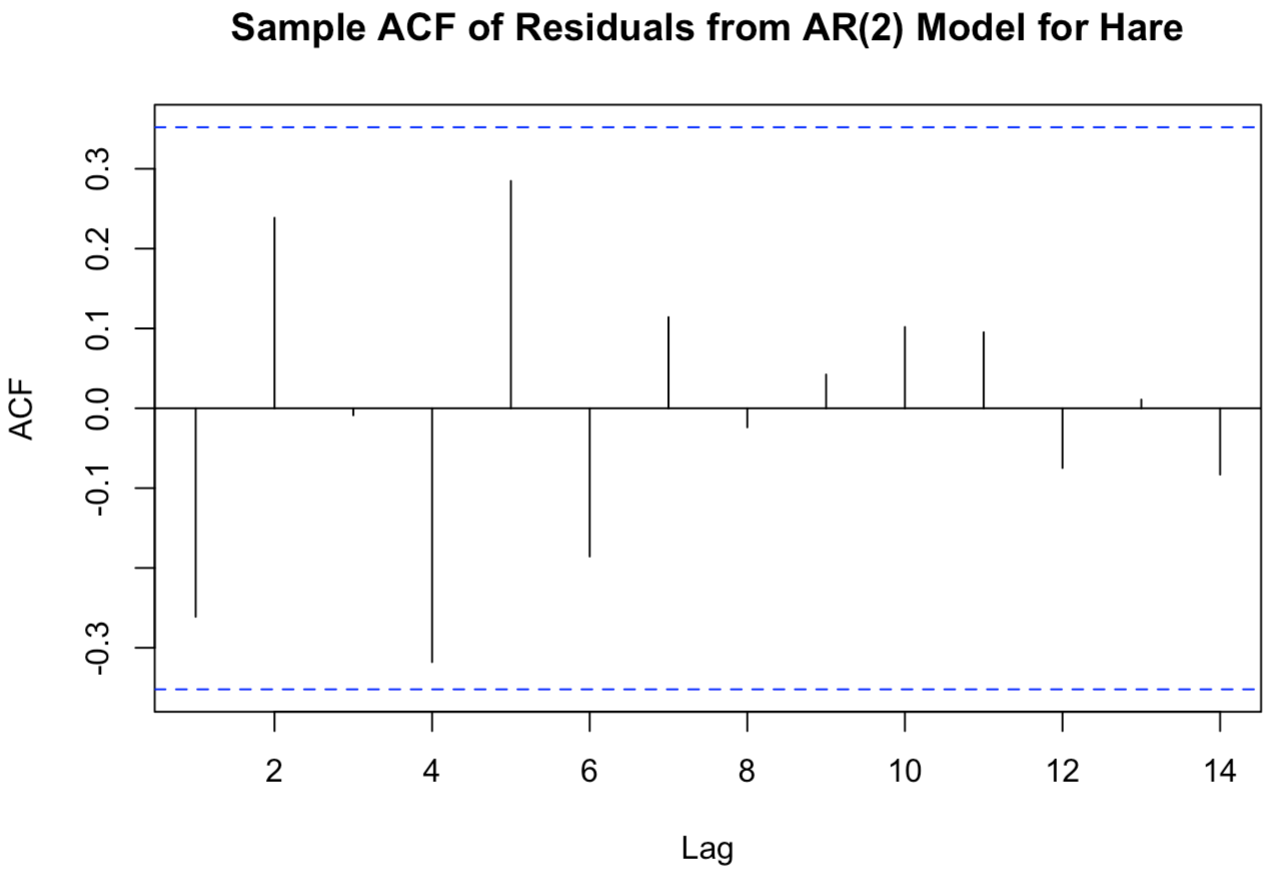

```r
# Create a QQ plot of the residuals from the m1.oil model
qqnorm(residuals(m1.oil))
# Add a line to the QQ plot to show the expected distribution of the residuals
qqline(residuals(m1.oil))
```




```
# Create a plot with a customized plot window size and point size
# Plot the autocorrelation function (ACF) of the residuals from the m1.color model
acf(residuals(m1.color), main='Sample ACF of Residuals from AR(1) Model for Color')
```




```
# Plot the autocorrelation function (ACF) of the residuals from the ARIMA(2,0,0) mode
l for the square root of the hare dataset
acf(residuals(arima(sqrt(hare), order=c(2,0,0))), main='Sample ACF of Residuals from
AR(2) Model for Hare')
```



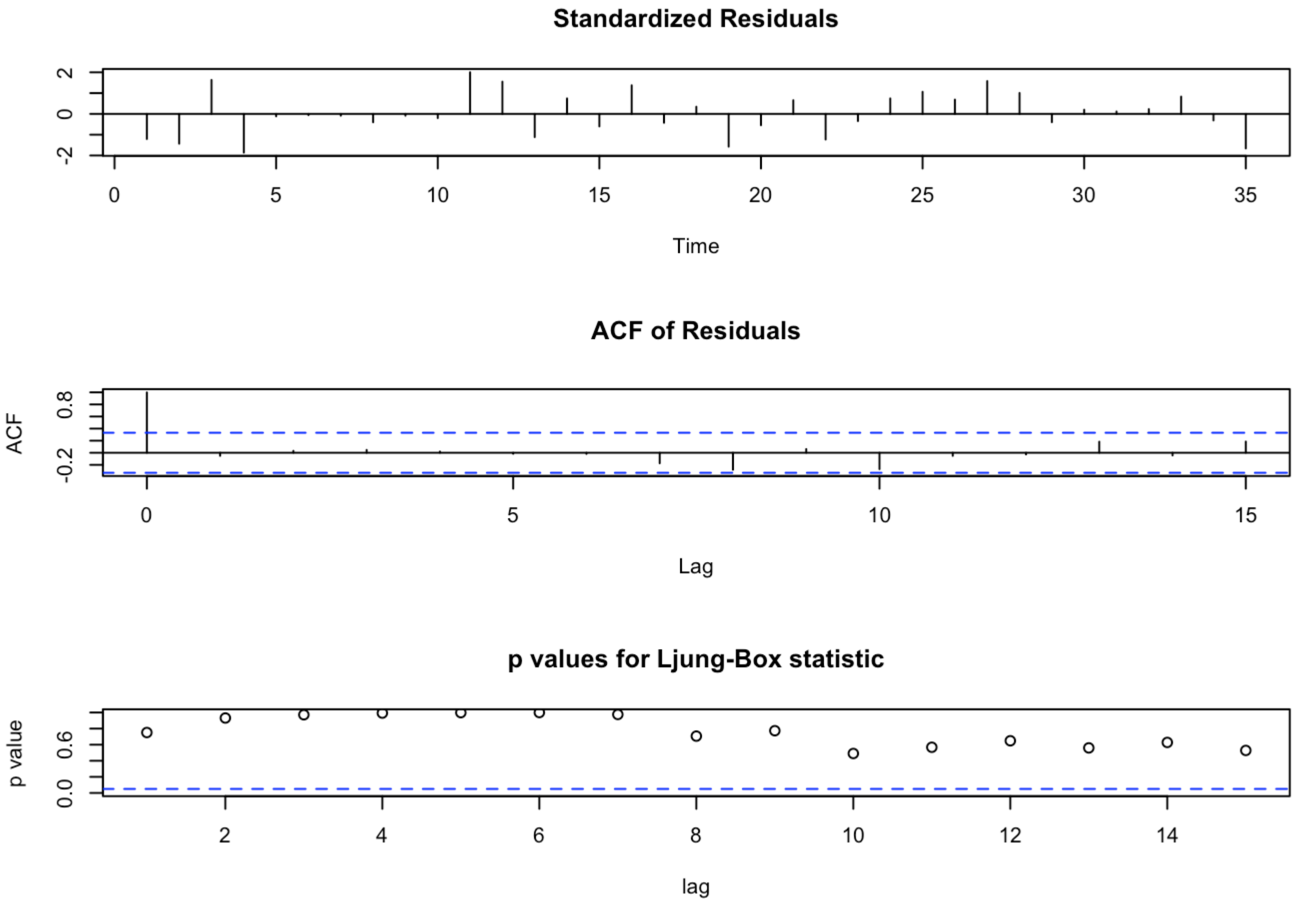

```
# Calculate the autocorrelation function (ACF) of the residuals from the m1.color mod
el
acf_res <- acf(residuals(m1.color), plot=FALSE)$acf
# Display the ACF values
acf_res
```

```
## , , 1
##
##               [,1]
##  [1,] -0.05138241
##  [2,]  0.03224346
##  [3,]  0.04749703
##  [4,]  0.02088157
##  [5,] -0.01729829
##  [6,] -0.01924314
##  [7,] -0.17341651
##  [8,] -0.28136877
##  [9,]  0.06127493
## [10,] -0.27045217
## [11,] -0.05123367
## [12,] -0.02712861
## [13,]  0.18344079
## [14,] -0.04423987
## [15,]  0.18564229
```

```
# Display the first 6 ACF values rounded to 2 significant digits
signif(acf_res[1:6], 2)
```

```
## [1] -0.051  0.032  0.047  0.021 -0.017 -0.019
```

```
# Create a plot with a customized plot window size
# Plot time series diagnostics for the m1.color model with a goodness-of-fit test of
15 lags
tsdiag(m1.color, gof=15, omit.initial=FALSE)
```




```
m1.color
```

```
##
## Call:
## arima(x = color, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.5705    74.3293
## s.e.  0.1435     1.9151
##
## sigma^2 estimated as 24.83:  log likelihood = -106.07,  aic = 216.15
```

```
# Fit an ARIMA(2,0,0) model to the color dataset
m2.color <- arima(color, order=c(2,0,0))
# Display the model summary
m2.color
```

```
##
## Call:
## arima(x = color, order = c(2, 0, 0))
##
## Coefficients:
##          ar1     ar2  intercept
##       0.5173  0.1005    74.1551
## s.e.  0.1717  0.1815     2.1463
##
## sigma^2 estimated as 24.6:  log likelihood = -105.92,  aic = 217.84
```

```
# Fit an ARIMA(1,0,1) model to the color dataset
m3.color <- arima(color, order=c(1,0,1))
# Display the model summary
m3.color
```

```
##
## Call:
## arima(x = color, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.6721  -0.1467    74.1730
## s.e.  0.2147   0.2742     2.1357
##
## sigma^2 estimated as 24.63:  log likelihood = -105.94,  aic = 217.88
```

```
# Fit an ARIMA(2,0,1) model to the color dataset
m4.color <- arima(color, order=c(2,0,1))
# Display the model summary
m4.color
```

```
##
## Call:
## arima(x = color, order = c(2, 0, 1))
##
## Coefficients:
##          ar1     ar2     ma1  intercept
##       0.2189  0.2735  0.3036    74.1653
## s.e.  2.0056  1.1376  2.0650     2.1121
##
## sigma^2 estimated as 24.58:  log likelihood = -105.91,  aic = 219.82
```

Exercise Questions

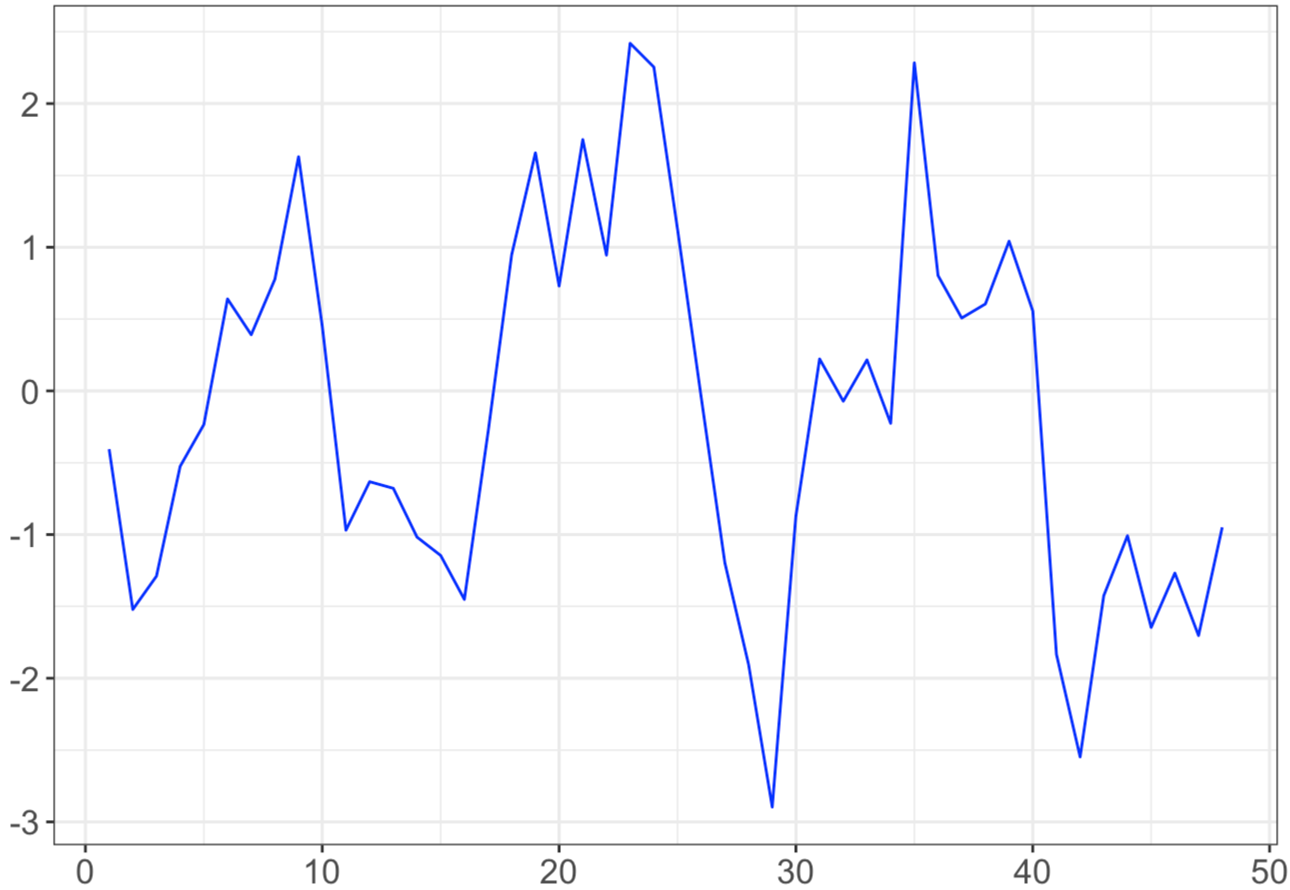
Exercise 6.20

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
set.seed(0)
Y = arima.sim(model=list(ar=0.7), n=48)
```

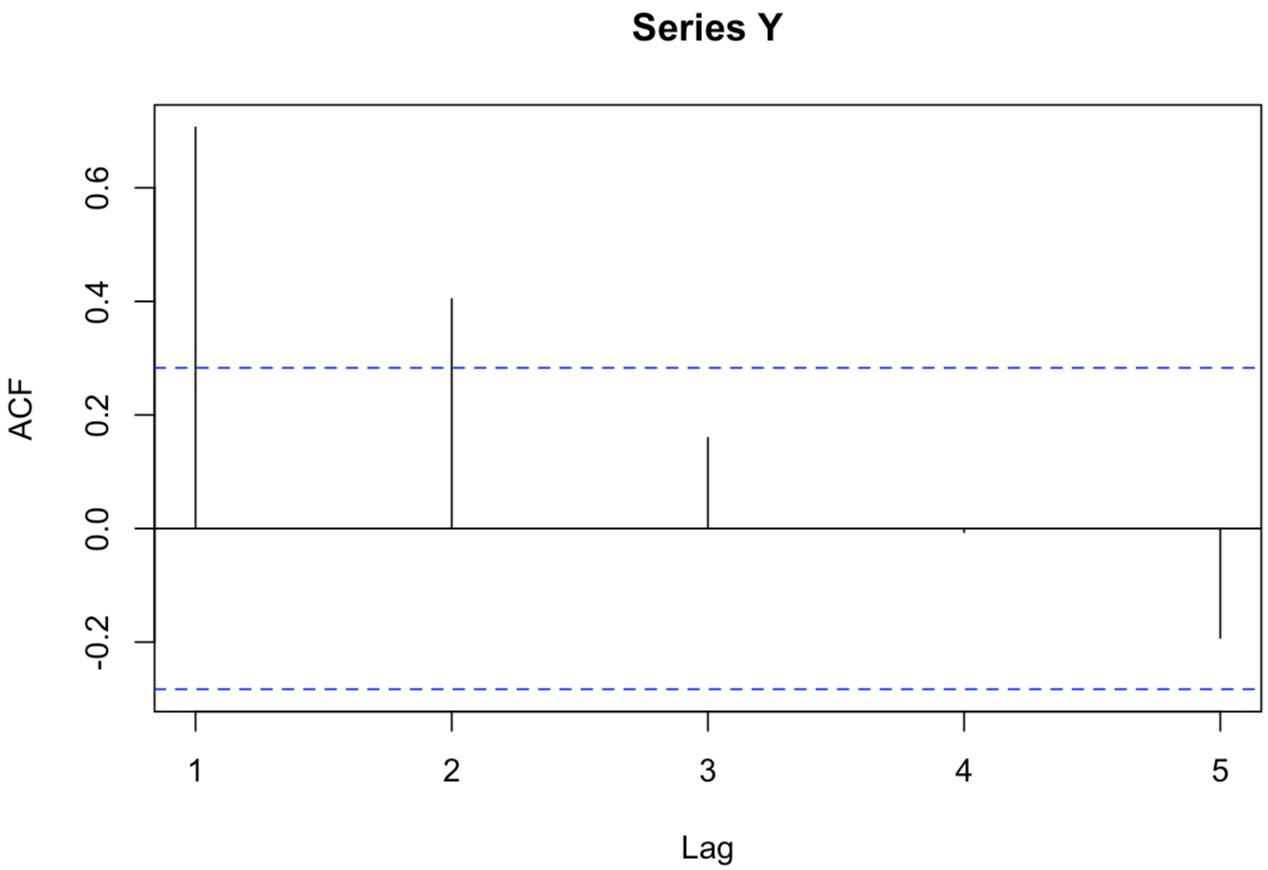
```
options(repr.plot.width=12, repr.plot.height=4)

ggplot() +
    geom_line(aes(x=1:48, y=c(Y)), color='blue') +
    xlab('') + ylab('') +
    theme_bw() + theme(text = element_text(size=16), plot.title = element_text(hjust
= 0.5))
```



a. The theoretical autocorrelations for an AR(1) process follow $\rho k=\phi k$, so $\rho 1=0.7$ and $\rho 5=(0.7)5=0.16807$

b.

```
lags = acf(Y, lag.max=5)
```



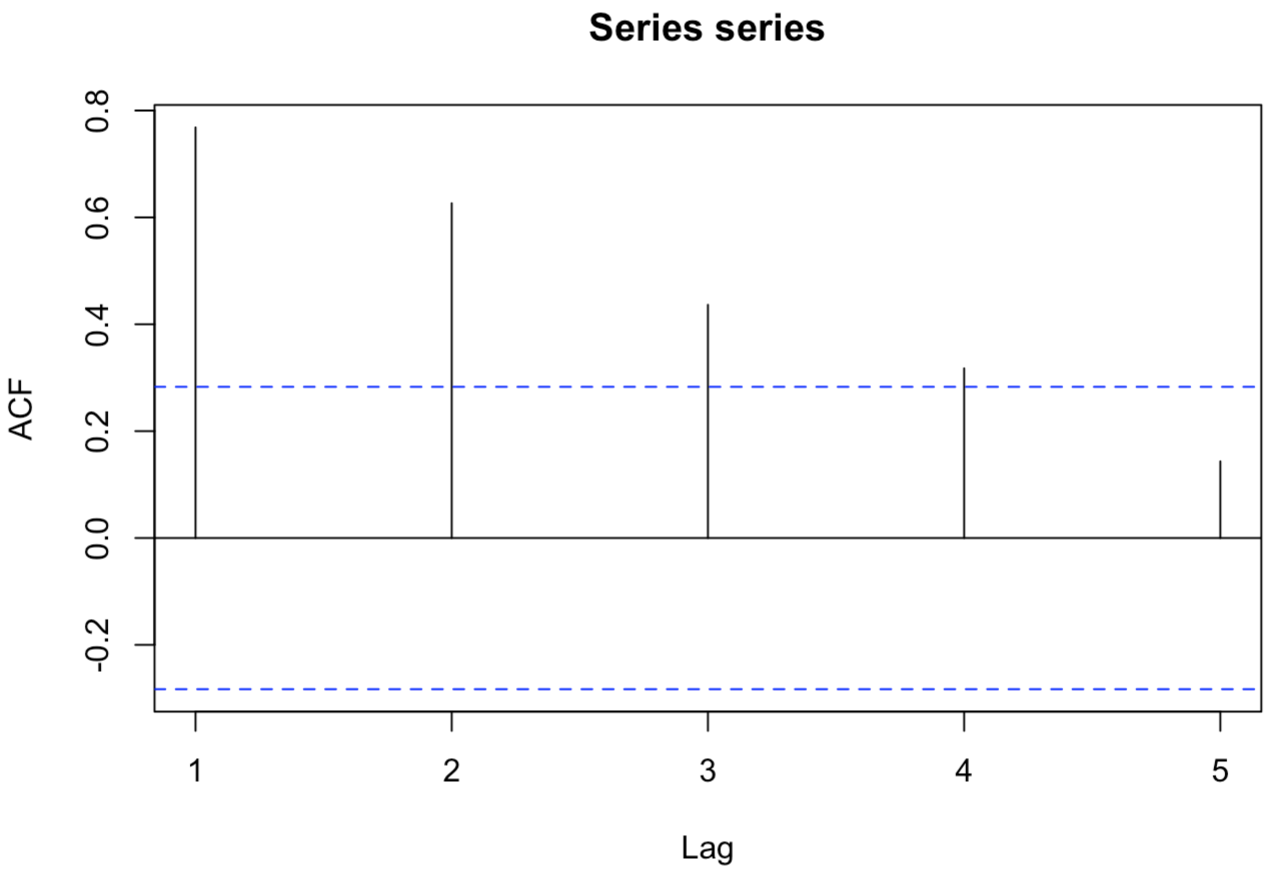

```
lags
```

```
##
## Autocorrelations of series 'Y', by lag
##
##      1      2      3      4      5
##  0.706  0.404  0.160 -0.007 -0.193
```

c.

```
set.seed(241357)
series=arima.sim(n=48,list(ar=0.7))
```

```
lags =acf(series,lag.max=5)[1:5]
```




```
lags
```

```
##
## Autocorrelations of series 'series', by lag
##
##     1     2     3     4     5
## 0.768 0.626 0.436 0.318 0.143
```

Exercise 6.21

```
require(ggplot2)
require(latex2exp)
```

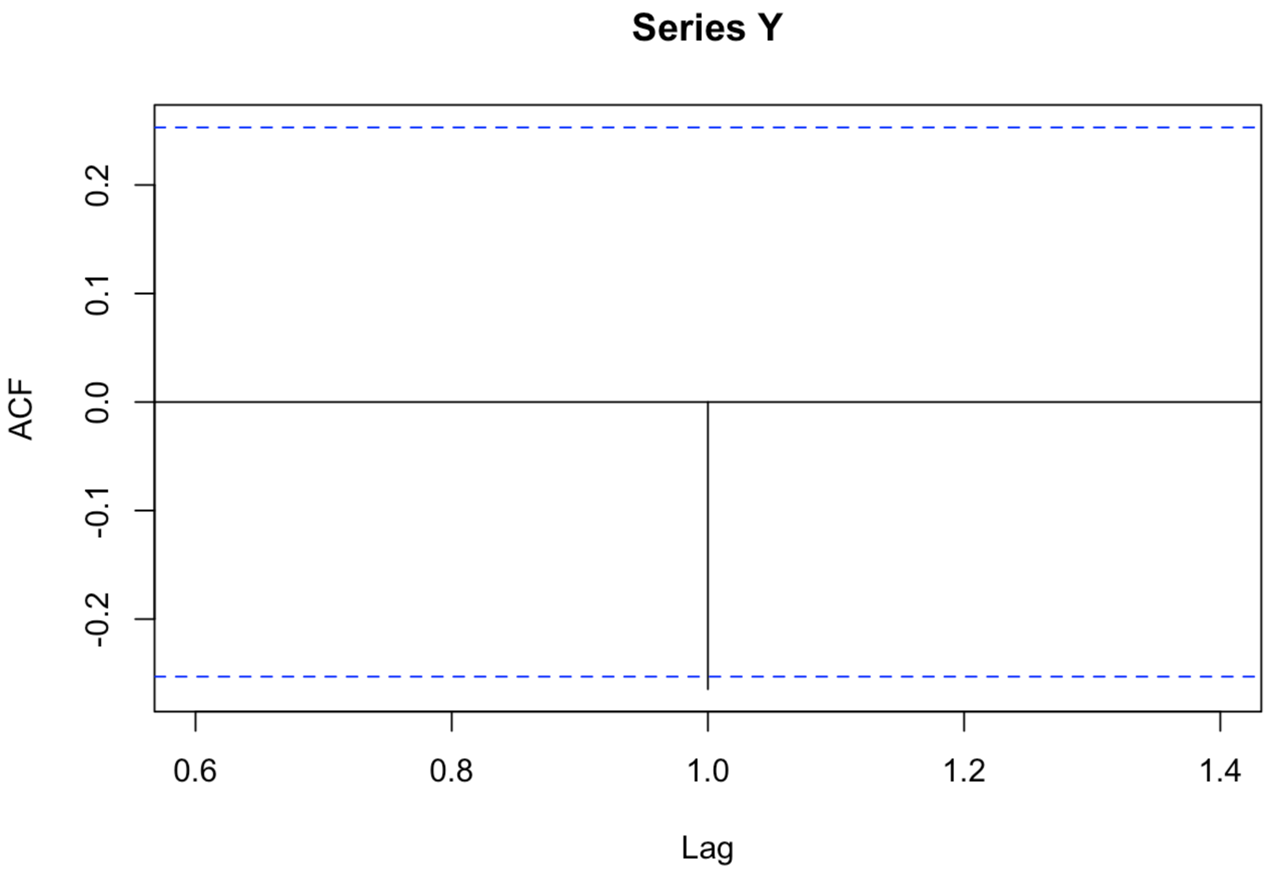
```
## Loading required package: latex2exp
```

```
set.seed(10)
Y = arima.sim(model=list(ma=-0.5), n=60)
```

a. The theoretical autocorrelation is $\rho 1 = -\theta/(1+\theta 2) = -0.4$.

b.

```
lags = acf(Y, lag.max=1)
```



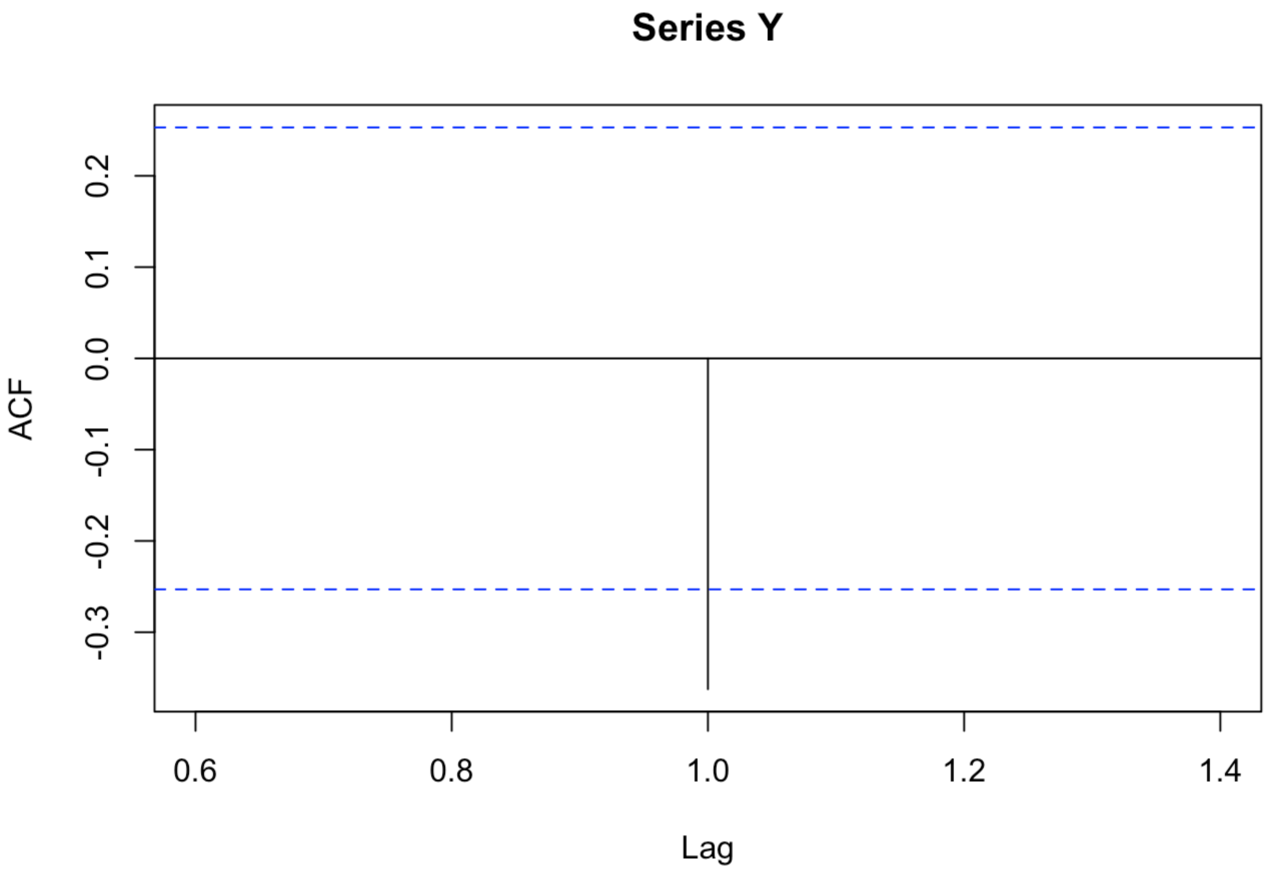

```
lags
```

```
##
## Autocorrelations of series 'Y', by lag
##
##      1
## -0.265
```

c.

```
set.seed(6453421)
Y = arima.sim(model=list(ma=-0.5), n=60)
```

```
lags = acf(Y, lag.max=1)[1]
```



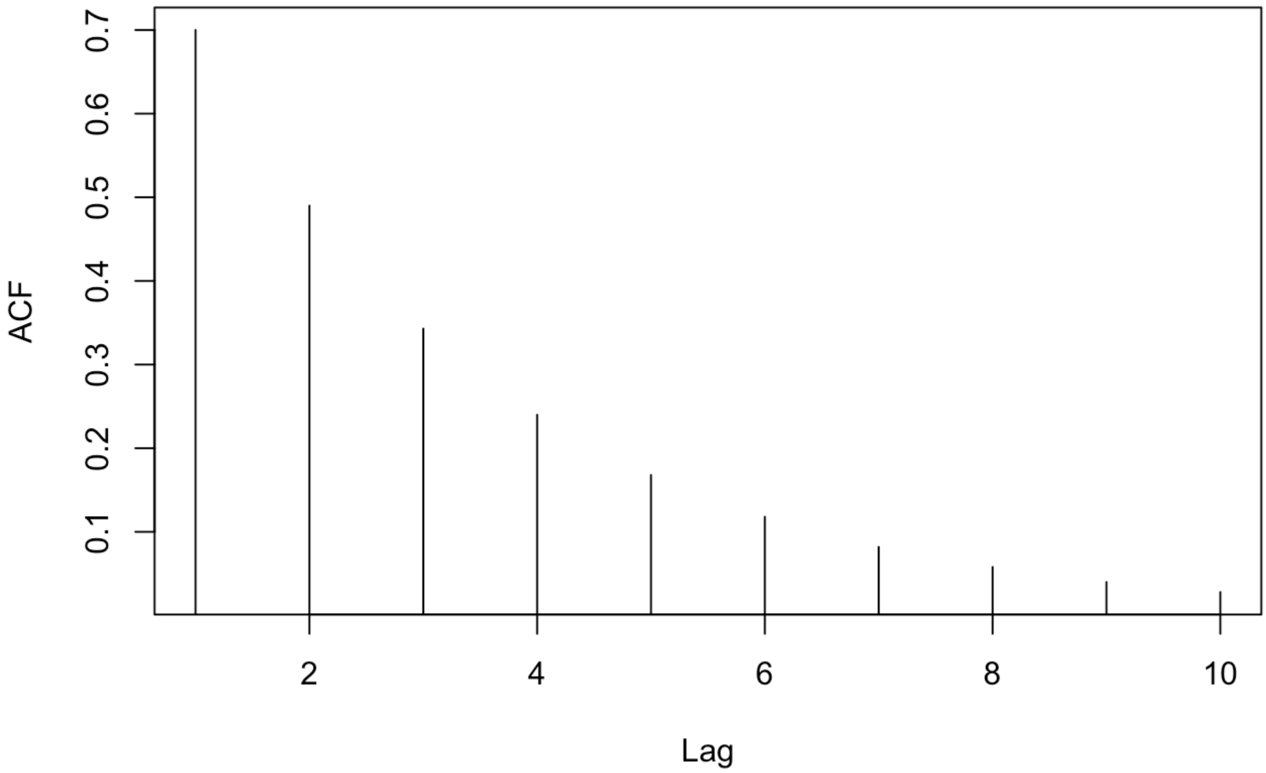

```
lags
```

```
##
## Autocorrelations of series 'Y', by lag
##
##      1
## -0.362
```

Exercise 6.25

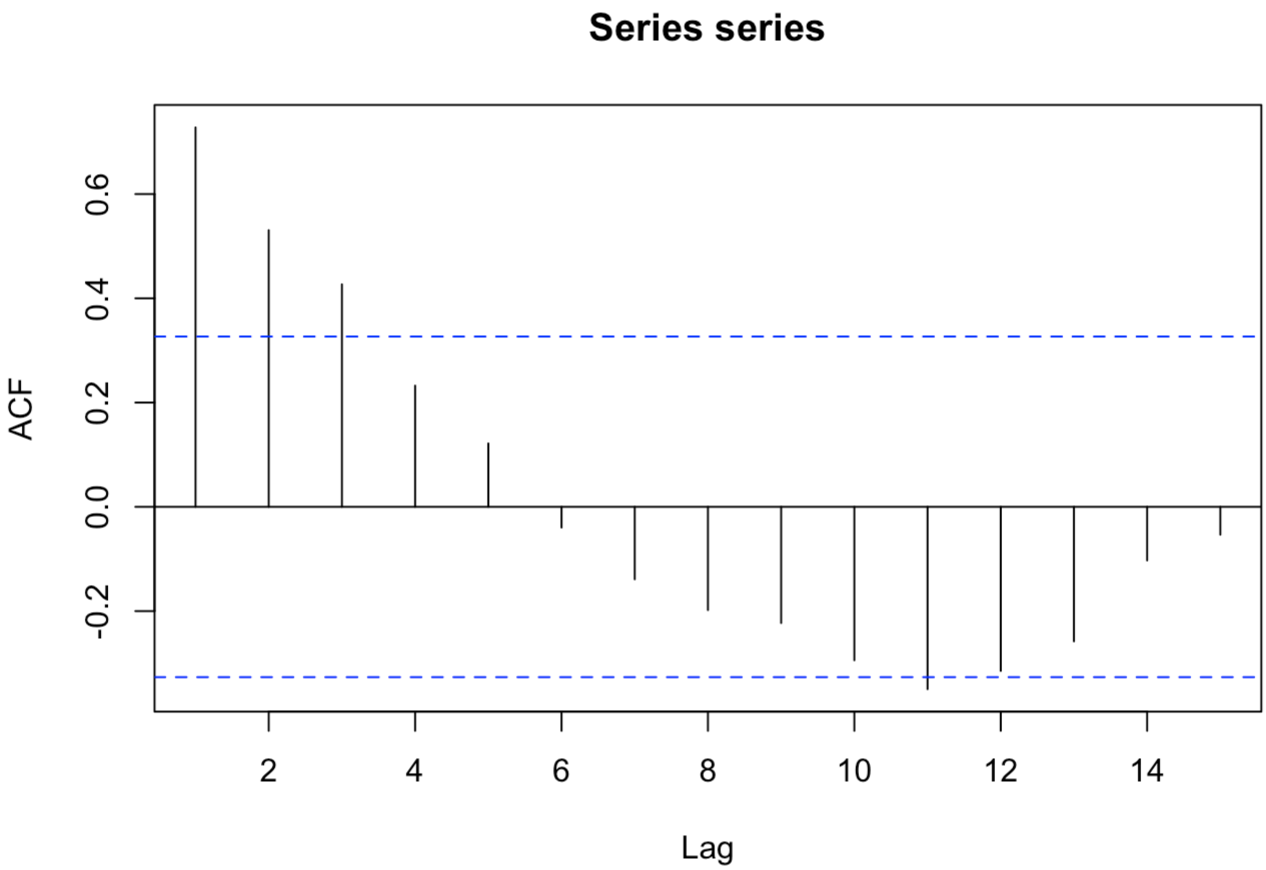
a.

```
ACF <- round(ARMAacf(ar=0.7,lag.max=10),digits=3)
plot(y=ACF[-1],x=1:10,xlab='Lag',ylab='ACF',type='h'); abline(h=0)
```



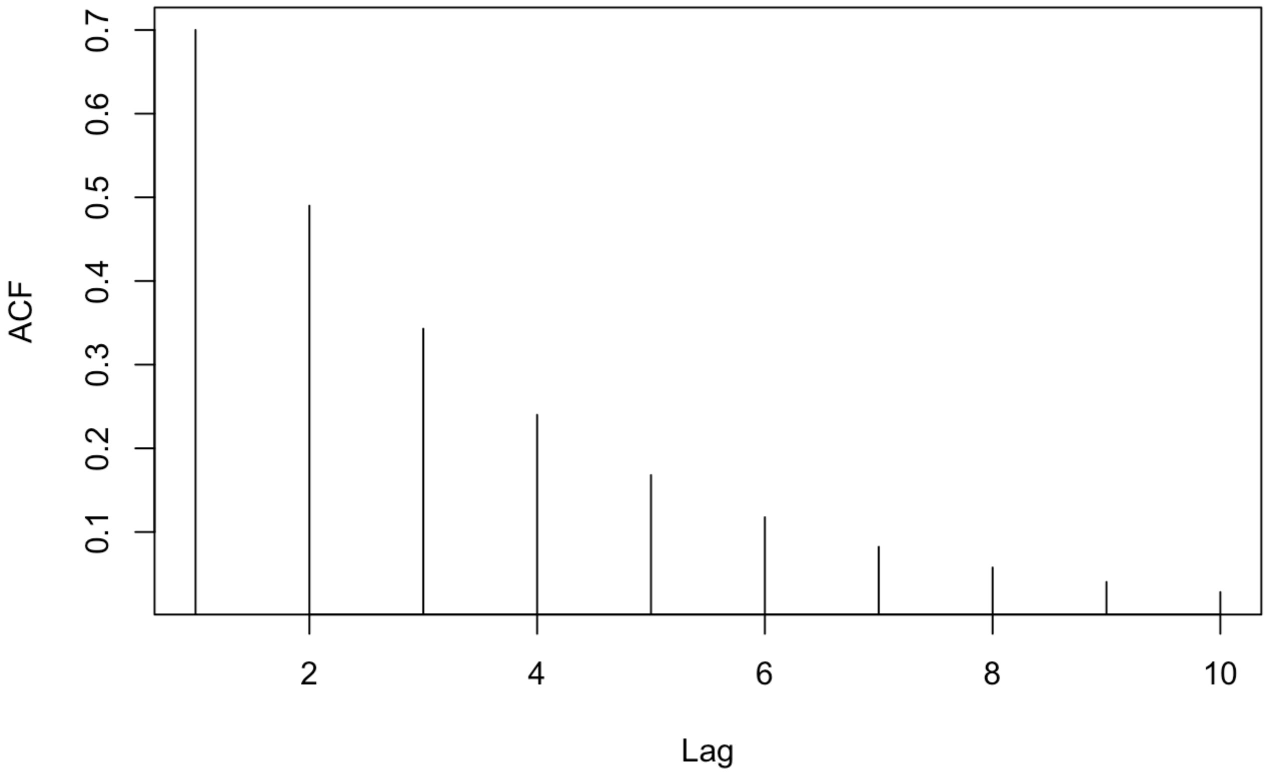
b.

```
set.seed(123)
series=arima.sim(n=36,list(ar=0.7))
acf(series)
```



The pattern match is not that good but we know that n = 36.

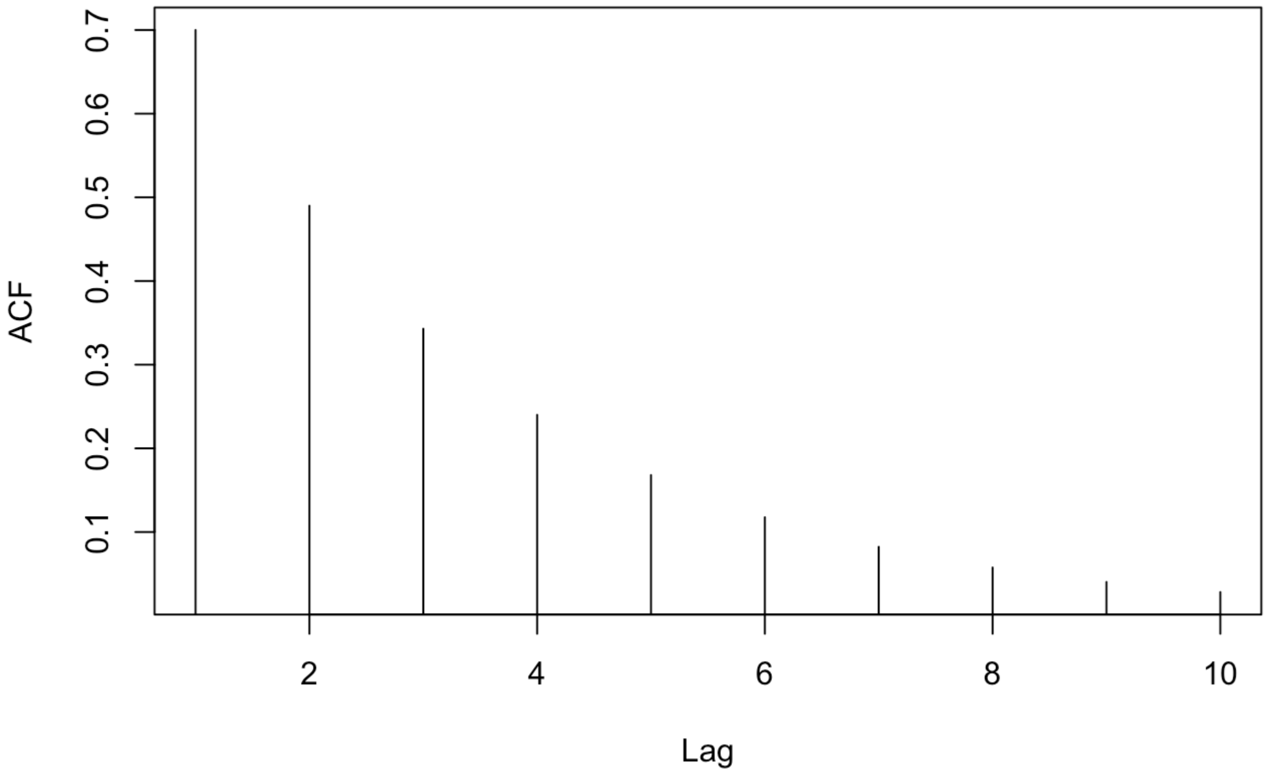
```
ACF=ARMAacf(ar=0.7,lag.max=10)
plot(y=ACF[-1],x=1:10,xlab='Lag',ylab='ACF',type='h'); abline(h=0)
```



c. For the AR(1) model, $\phi 11 = \phi = 0.7$ and $\phi kk = 0$ for $k > 1$. Theoretical auto-correlation for phi_11 = 0.7 and for phi_kk=0

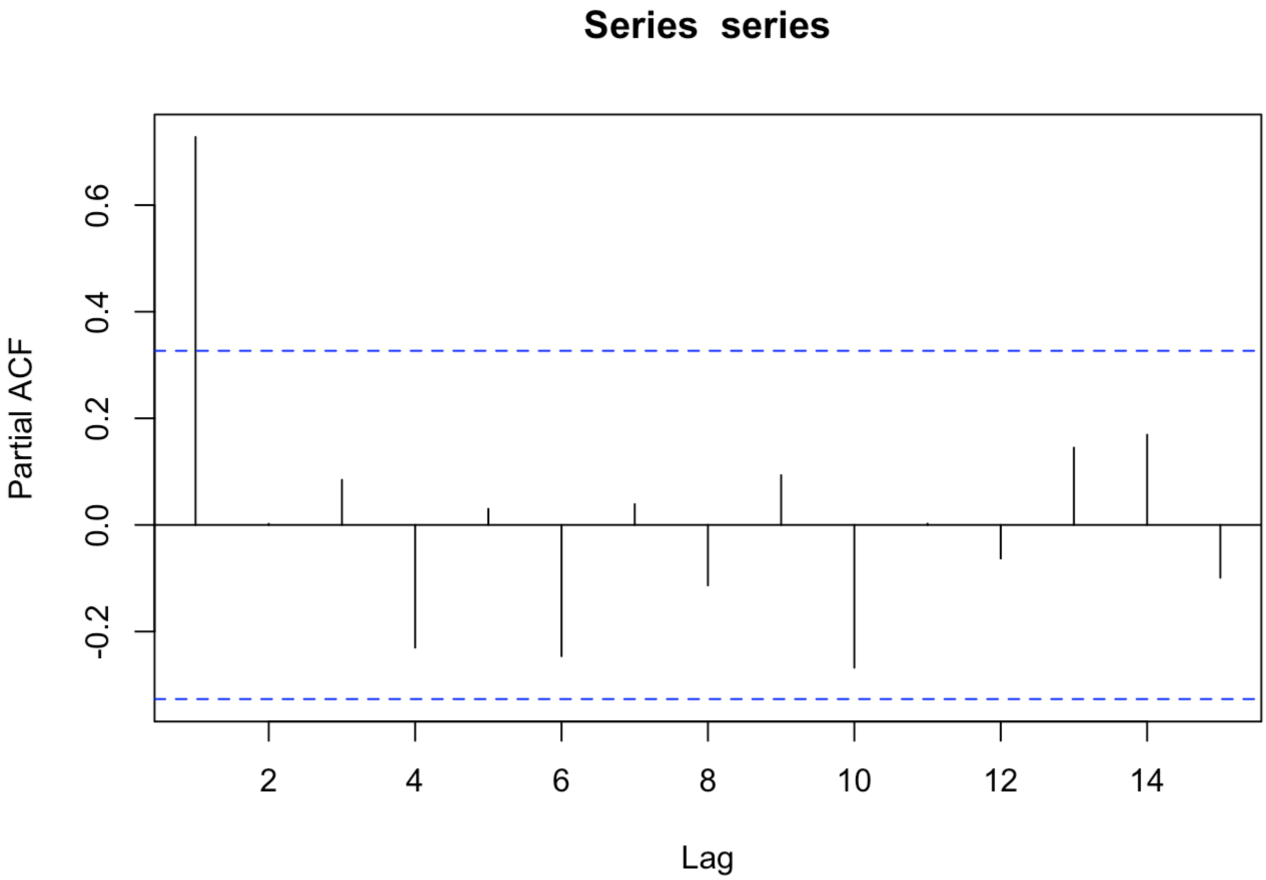
d. Standard Deviation = $ = = 0.12 $. The sample auto correlation r1 is within two standard deviations.

```
plot(y=ACF[-1],x=1:10,xlab='Lag',ylab='ACF',type='h'); abline(h=0)
```



e. The pattern is a decent match.

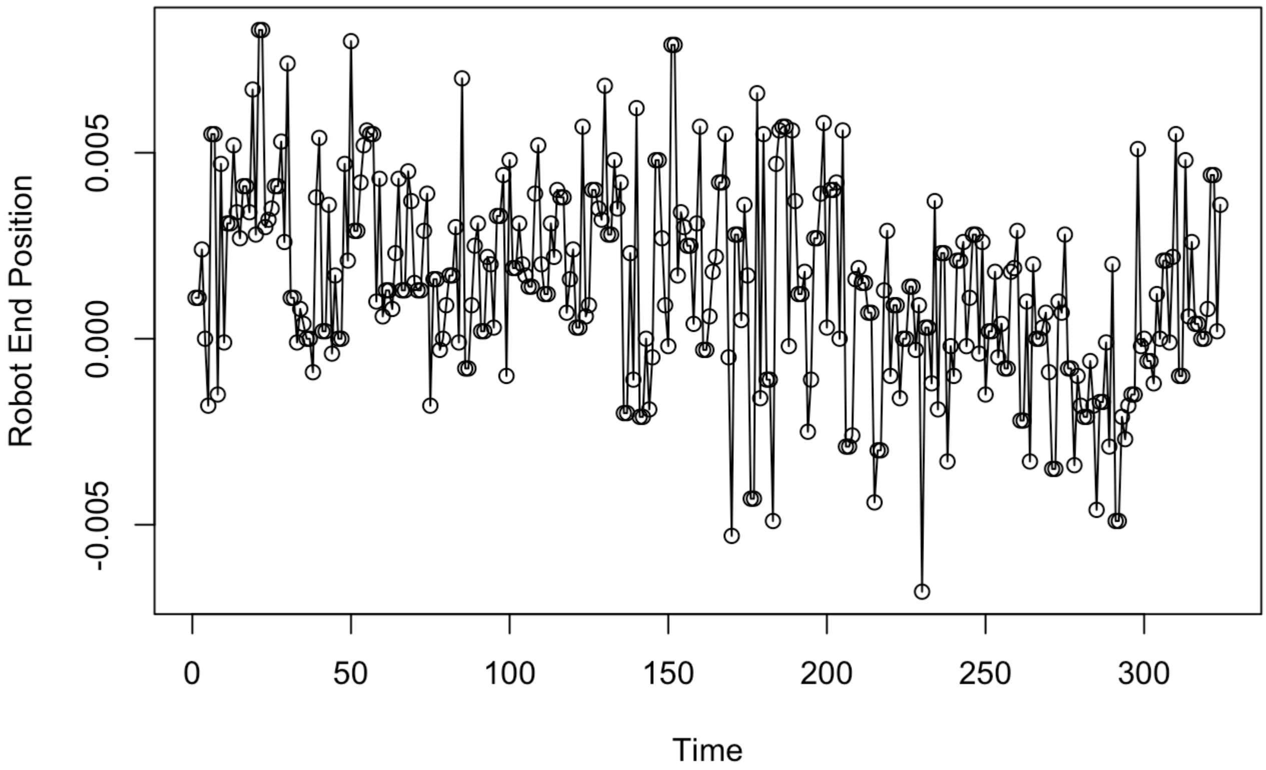
```
pacf(series)
```



Exercise 6.36

a.

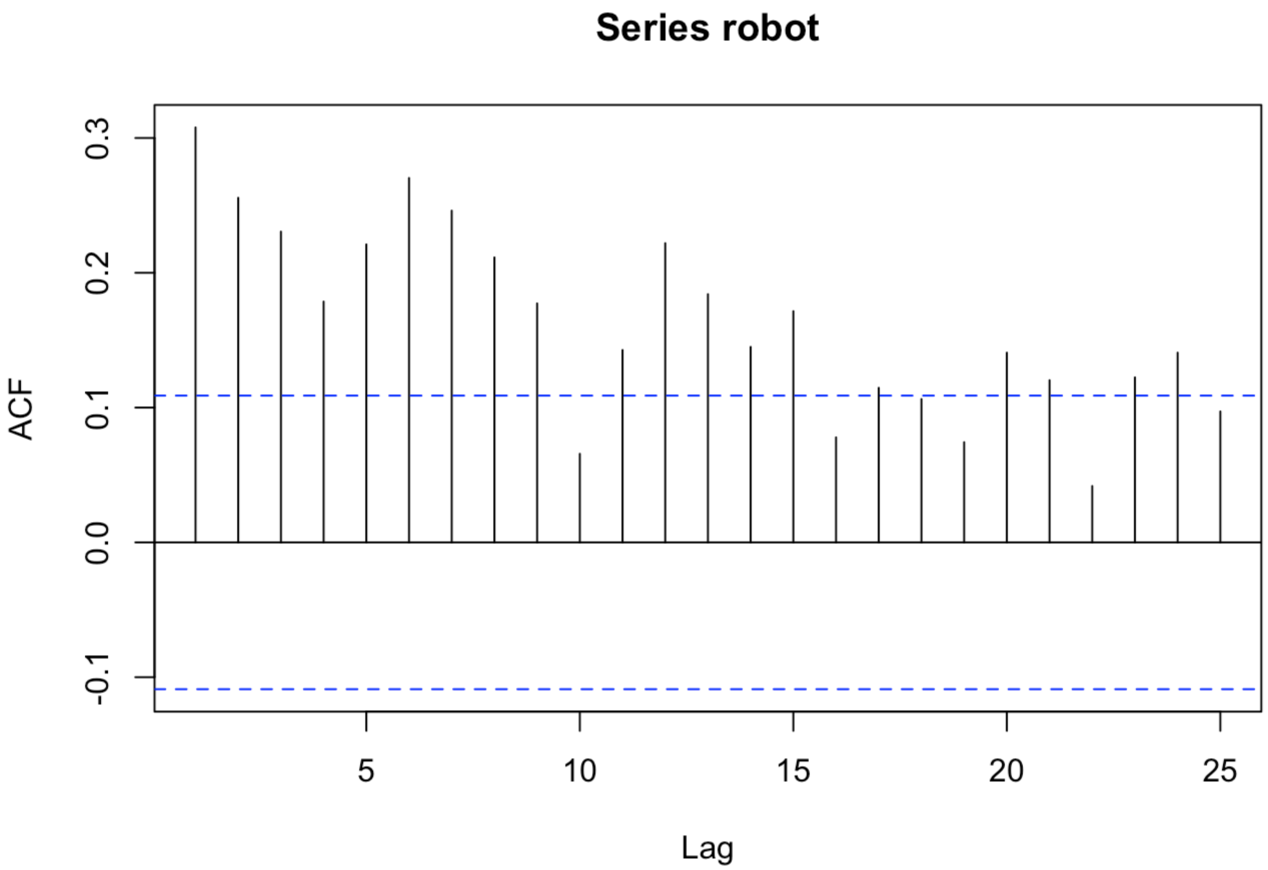
```
data(robot)
plot(robot, type='o',ylab='Robot End Position')
```



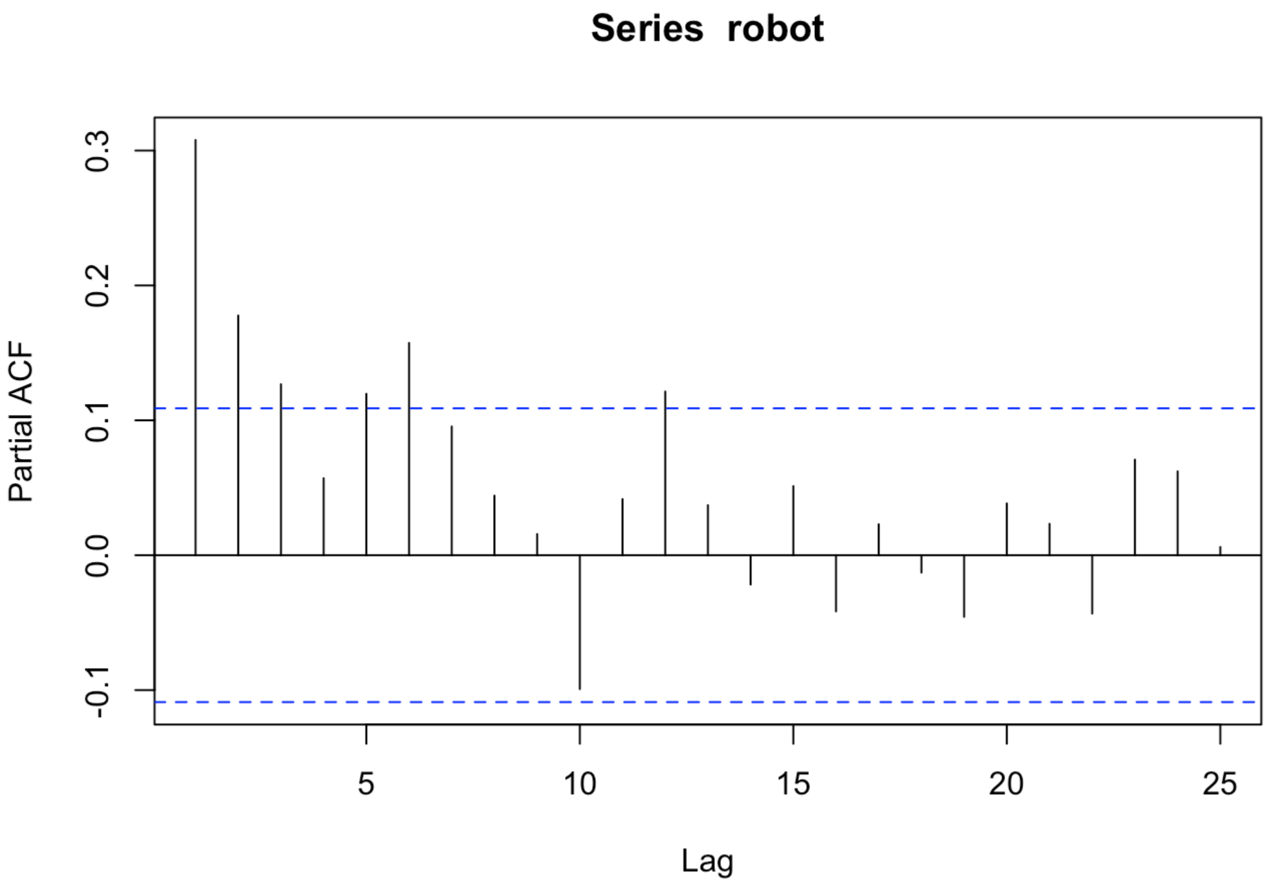
From this plot we might try a stationary model but there is also enough “drift” that we might also suspect non-stationarity

b.

```
acf(robot)
```


Series robot
ACF
0.3
0.2
0.1
0.0
-0.1
5
10
15
20
25
Lag

```
pacf(robot)
```


Series robot
Partial ACF
0.3
0.2
0.1
0.0
-0.1
5
10
15
20
25
Lag

From this plot we might try a stationary model but there is also enough “drift” that we might also suspect non-stationarity

c.

```
eacf(robot)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x o x  x  x  x
## 1 x o o o o o o o o o o  o  o  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x x o o o o o o o o o  o  o  o
## 4 x x x x o o o o o o o  o  x  o
## 5 x x x o o o o o o o o  o  x  o
## 6 x o o o o x o o o o o  o  o  o
## 7 x o o x o x x o o o o  o  o  o
```

The EACF suggests an ARMA(1,1) model

Exercise 6.37

```
data(larain)
eacf(log(larain))
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o o o o o o o o o o  o  o  o
## 1 o o o o o o o o o o o  o  o  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x o o o o o o o o o o  o  o  o
## 4 x o o o o o o o o o o  o  o  o
## 5 x x x x x o o o o o o  o  o  o
## 6 x x o o x o o o o o o  o  o  o
## 7 x o x o o o o o o o o  o  o  o
```

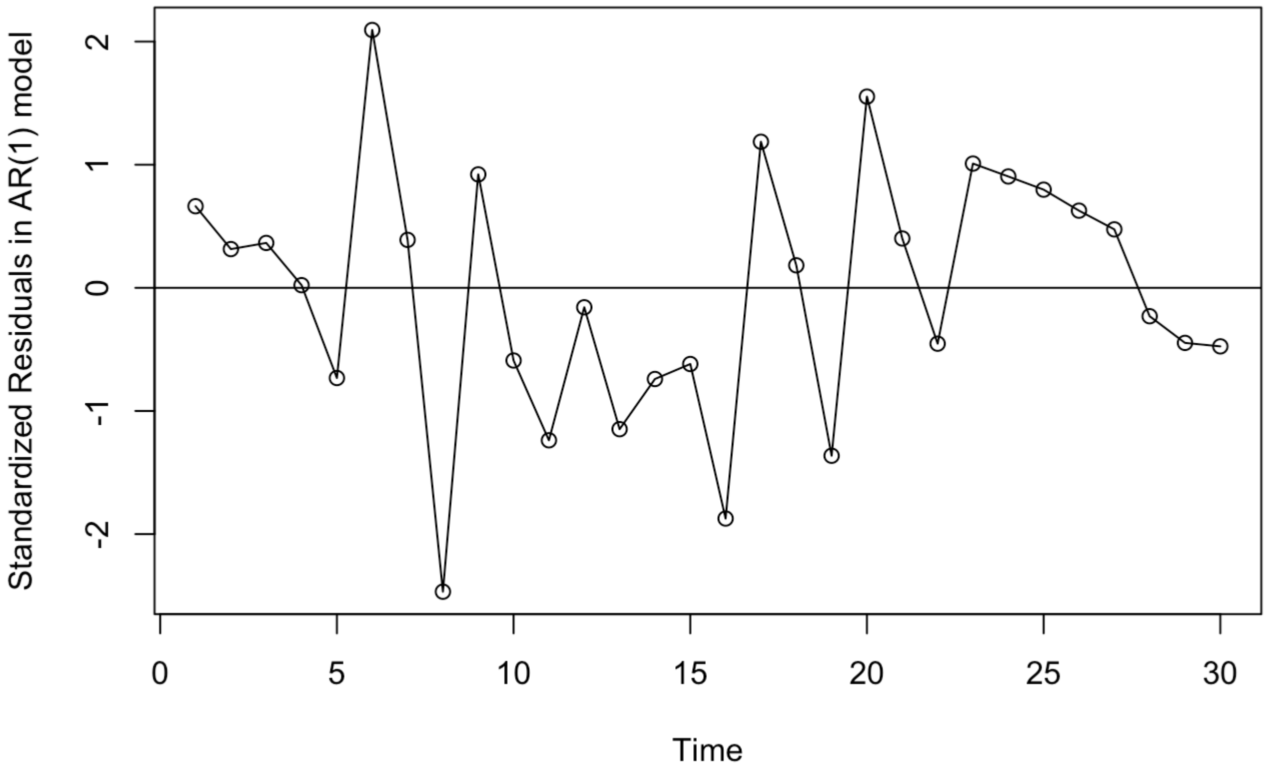
The EACF table for the Los Angeles rainfall series, which is dominated by “o”s at lower AR and MA orders, indicates non-significant autocorrelations, consistent with white noise characteristics.

Exercise 8.4

a.

```
set.seed(123)
Y = arima.sim(model=list(ar=0.5), n=30)
```

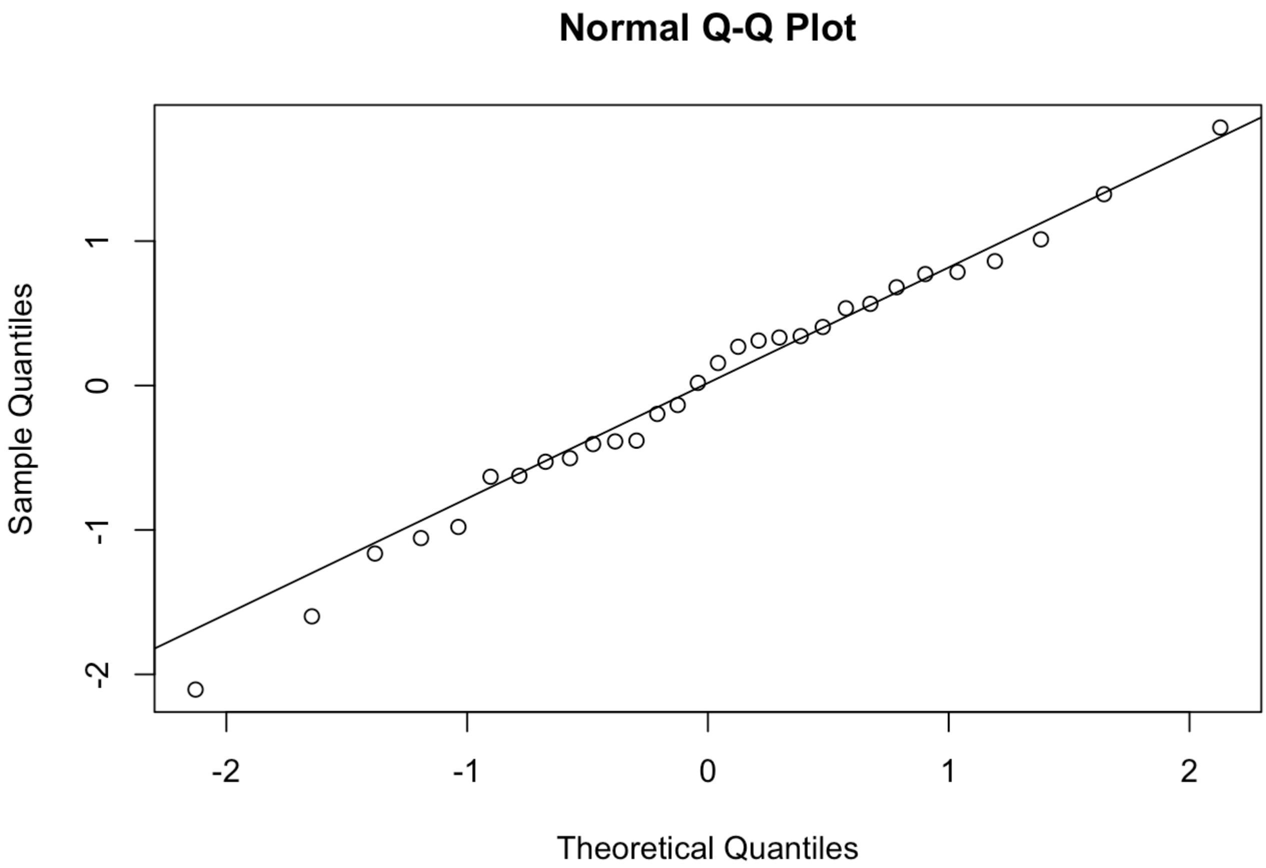
```
model = arima(Y, order=c(1,0,0), method='ML')
plot(rstandard(model),ylab ='Standardized Residuals in AR(1) model', type='o');
abline(h=0)
```



Residuals look random.

b.

```
qqnorm(model$resid); qqline(model$resid)
```



Lower part contains points outside of the range. We can do Shapiro-Wilk to verify normality.

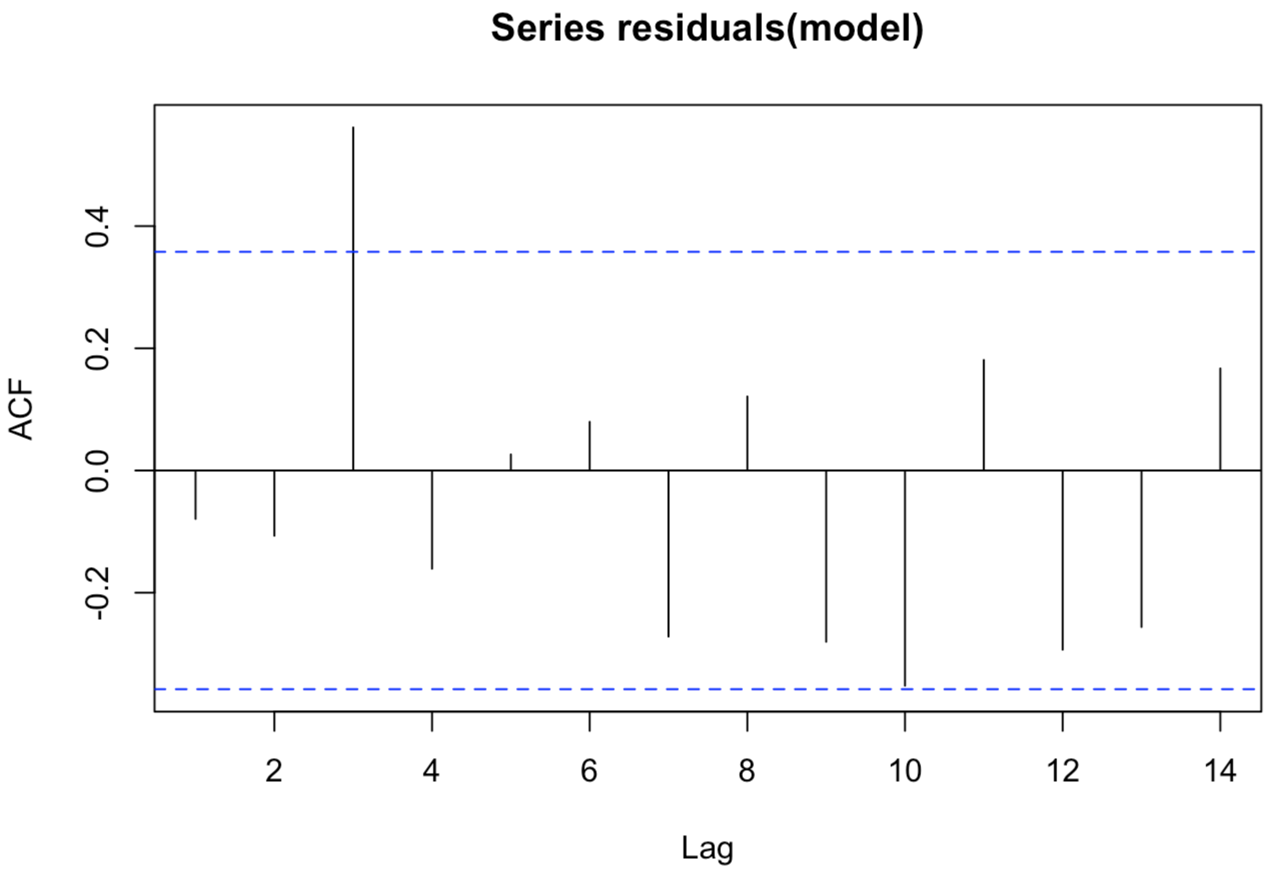
```
shapiro.test(model$resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$resid
## W = 0.98779, p-value = 0.9748
```

The test fails to reject normality.

c. The sample acf at lag 4 is statistically signifi cant among all autocorrelations observed.

```
acf(residuals(model))
```

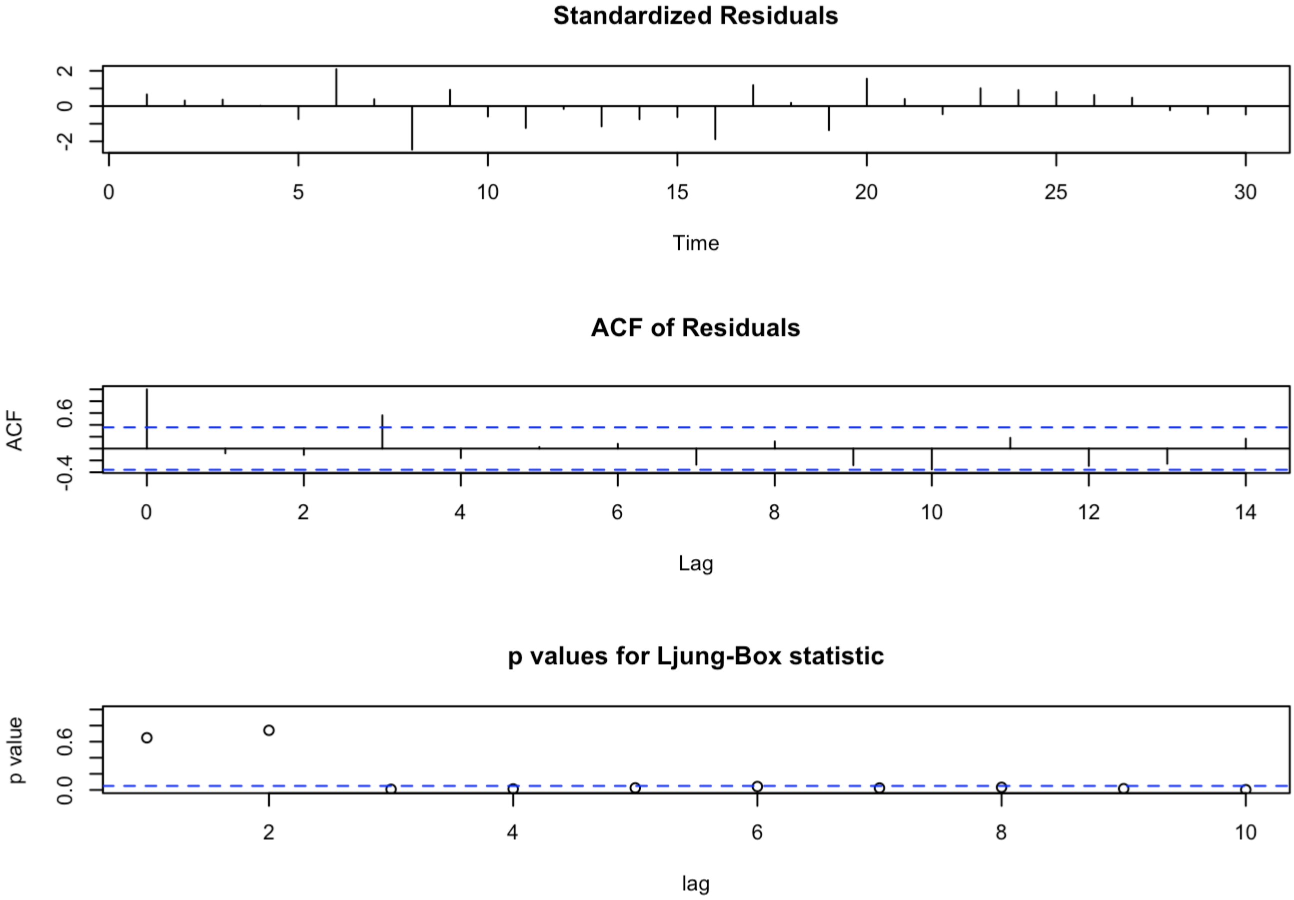


d.

```
LB.test(model,lag=8)
```

```
##
##  Box-Ljung test
##
## data:  residuals from  model
## X-squared = 16.765, df = 7, p-value = 0.01898
```

```
tsdiag(model)
```



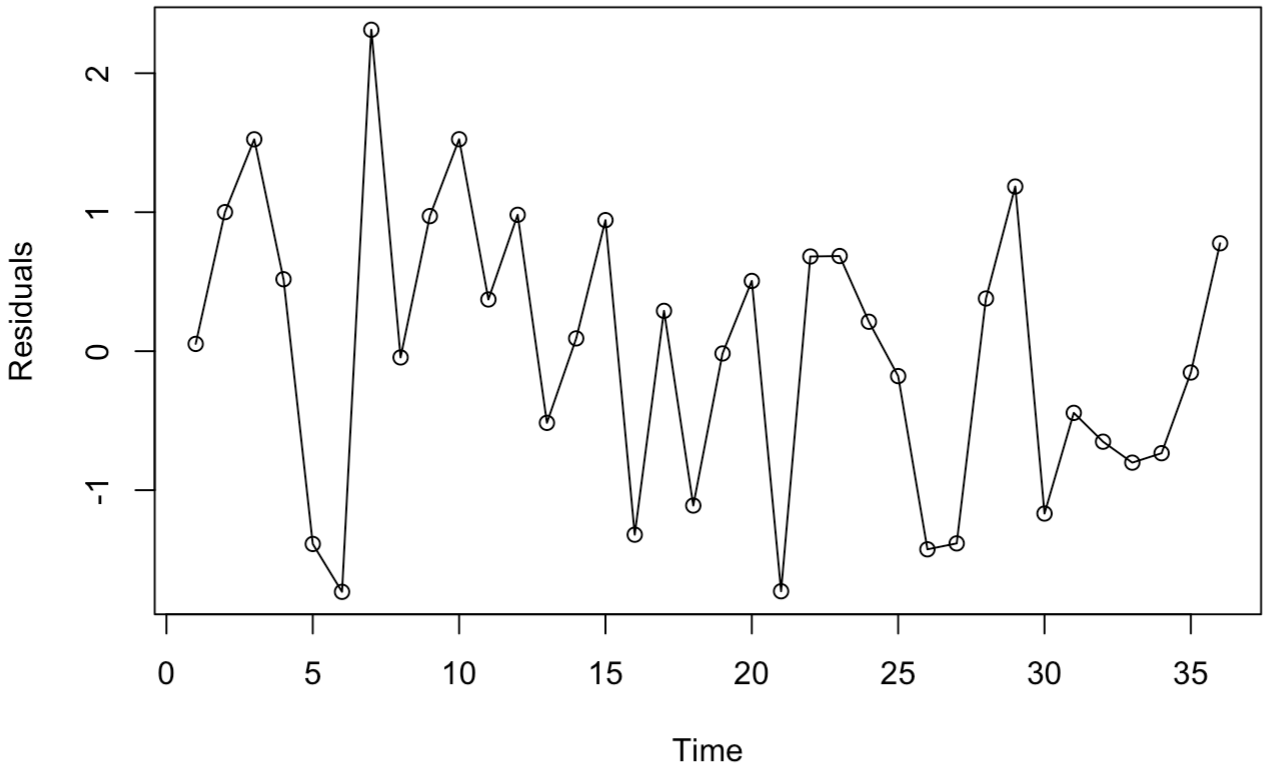
The test does not reject randomness of error based on the first 8 autocorrelations.

Exercise 8.5

```
set.seed(2000)
Y = arima.sim(model=list(ma=-0.5), n=36)
```

a.

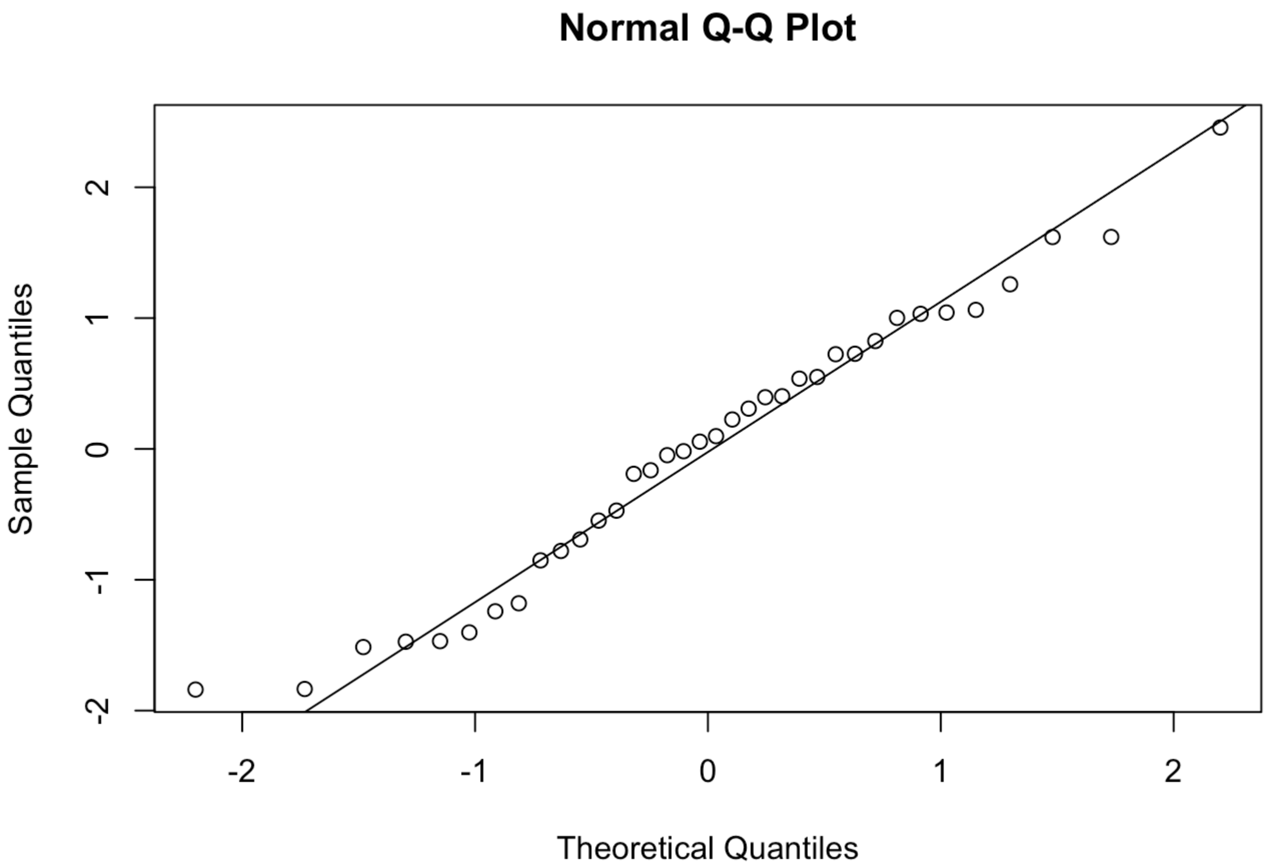
```
model = arima(Y, order=c(0,0,1), method='ML')
plot(rstandard(model),ylab ='Residuals', type='o')
```



Residuals look random.

b.

```
qqnorm(model$resid)
qqline(model$resid)
```



The residuals look normal. Doing Shapiro-Wilk test.

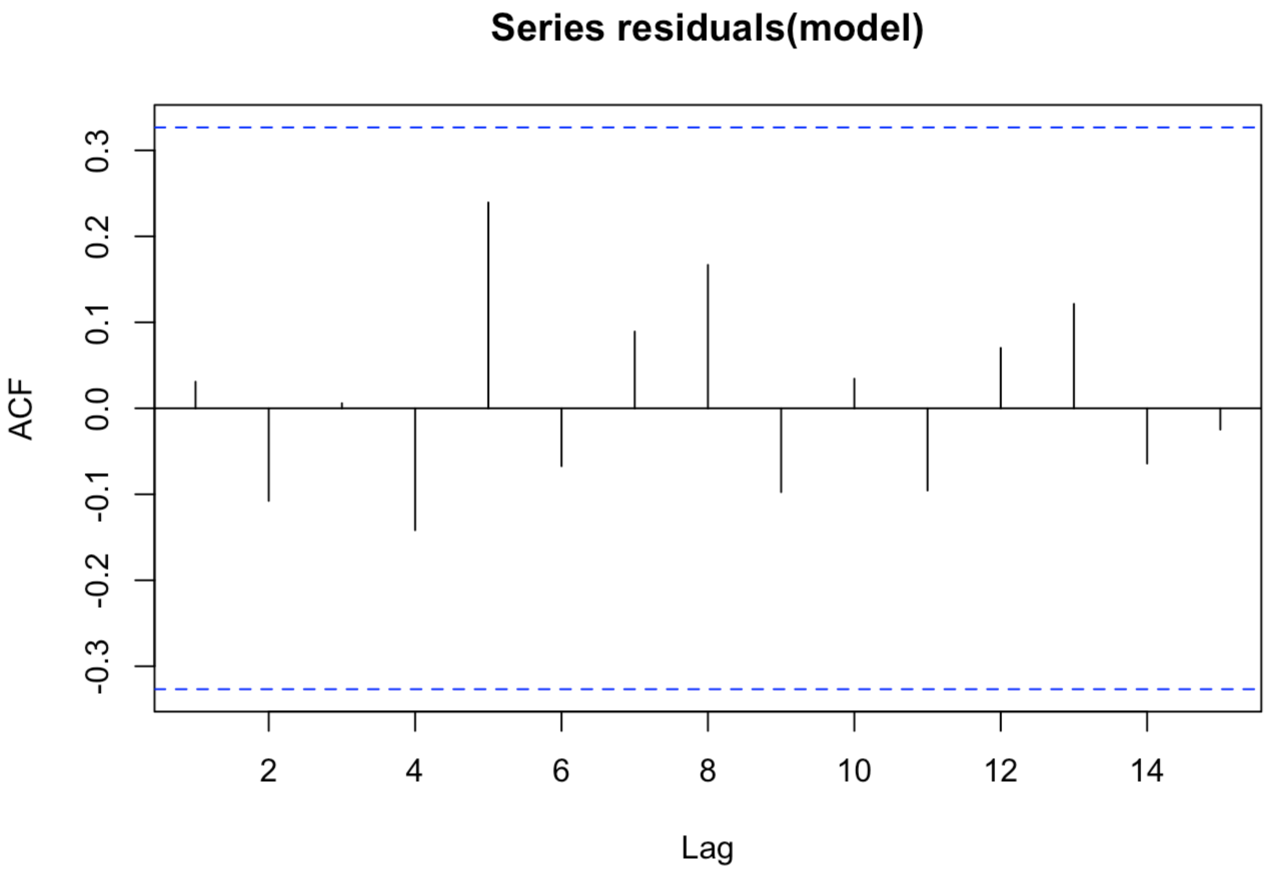
```
shapiro.test(model$resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$resid
## W = 0.97292, p-value = 0.5105
```

The Shapiro-Wilk test fails to reject normality.

c.

```
acf(residuals(model))
```



ACF suggests the residuals are white noise.

d.

```
LB.test(model,lag=6)
```

```
##
##  Box-Ljung test
##
## data:  residuals from  model
## X-squared = 4.0998, df = 5, p-value = 0.5351
```

The test does not reject randomness of error based on the first 6 autocorrelations.