tharunte_Homework6

Chapter 9 R Commands

```
library(TSA)
```

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

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

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

Exhibit 9.2 Forecasts and Limits for the Temperature Cosine Trend

```
data(tempdub)
tempdub1=ts(c(tempdub,rep(NA,24)),start=start(tempdub),freq=frequency(tempdub))
```

```
har.=harmonic(tempdub,1)
m5.tempdub=arima(tempdub,order=c(0,0,0),xreg=har.)
```

```
har.=harmonic(tempdub,1); model4=lm(tempdub~har.)
summary(model4)
```

```
##
## Call:
## lm(formula = tempdub ~ har.)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1580  -2.2756  -0.1457   2.3754  11.2671
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      46.2660     0.3088 149.816  < 2e-16 ***
## har.cos(2*pi*t) -26.7079     0.4367 -61.154  < 2e-16 ***
## har.sin(2*pi*t)  -2.1697     0.4367  -4.968 1.93e-06 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.706 on 141 degrees of freedom
## Multiple R-squared:  0.9639, Adjusted R-squared:  0.9634
## F-statistic:  1882 on 2 and 141 DF,  p-value: < 2.2e-16
```

```
arima(tempdub,order=c(0,0,0),xreg=data.frame(har.,trend=time(tempdub)))
```

```
##
## Call:
## arima(x = tempdub, order = c(0, 0, 0), xreg = data.frame(har., trend = time(tempdub)))
##
## Coefficients:
##       intercept  cos.2.pi.t.  sin.2.pi.t.   trend
##         23.8569     -26.7070      -2.1662  0.0114
## s.e.   174.1505       0.4322       0.4330  0.0884
##
## sigma^2 estimated as 13.45:  log likelihood = -391.43,  aic = 790.86
```

```
m5.tempdub
```

```
##
## Call:
## arima(x = tempdub, order = c(0, 0, 0), xreg = har.)
##
## Coefficients:
##       intercept  cos(2*pi*t)  sin(2*pi*t)
##         46.2660     -26.7079      -2.1697
## s.e.     0.3056       0.4322       0.4322
##
## sigma^2 estimated as 13.45:  log likelihood = -391.44,  aic = 788.88
```

```
newhar.=harmonic(ts(rep(1,24), start=c(1976,1),freq=12),1)
plot(m5.tempdub,n.ahead=24,n1=c(1972,1),newxreg=newhar., col = 'red', type='b',ylab='Temperature',
xlab='Year')
```

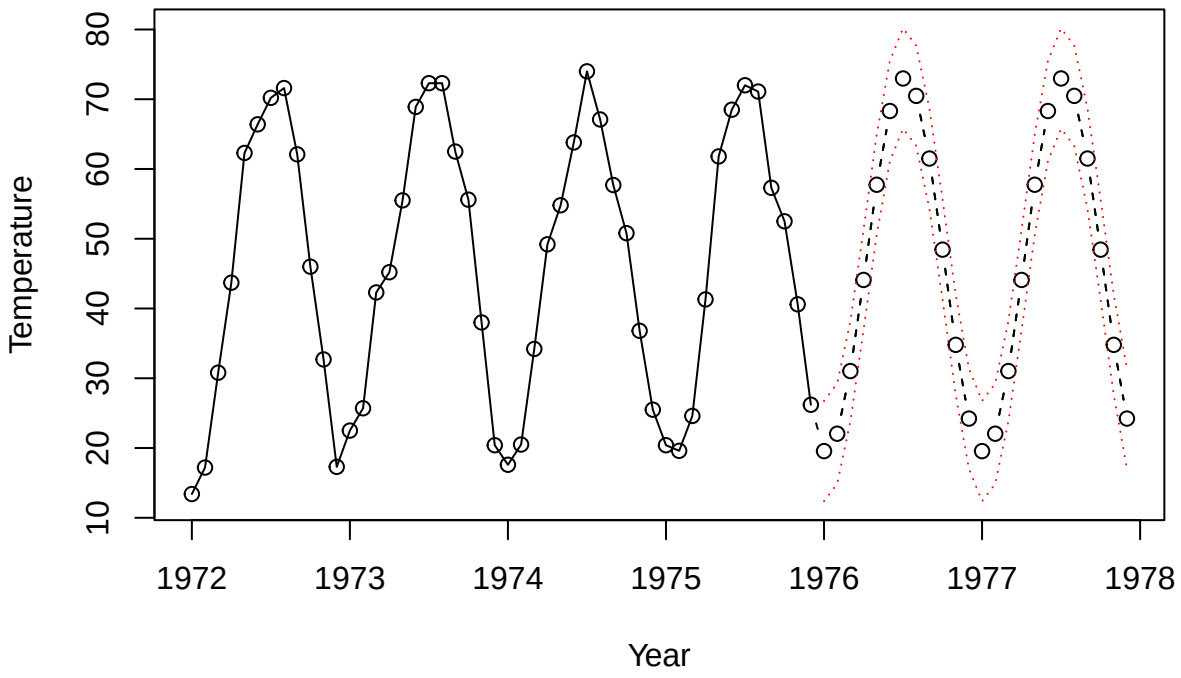
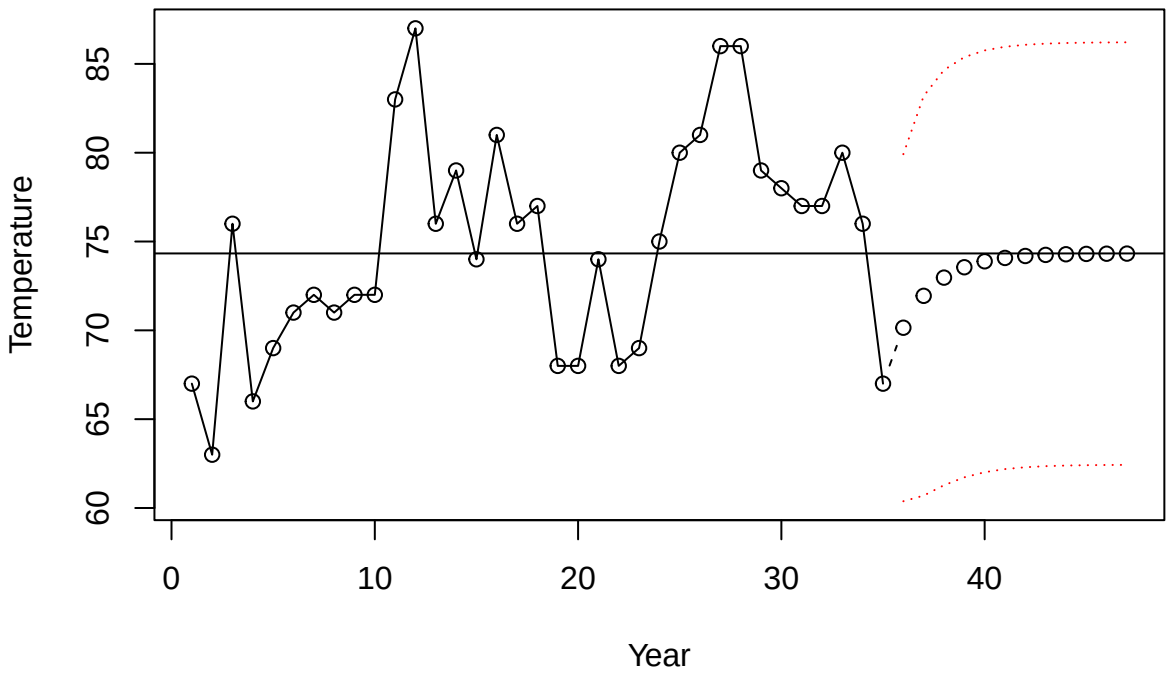


Exhibit 9.3

```
data(color)
m1.color=arima(color,order=c(1,0,0))
plot(m1.color,n.ahead=12,col='red',type='b',xlab='Year',ylab='Temperature')
abline(h=coef(m1.color)[names(coef(m1.color))=='intercept'])
```




```
v=1:5
names(v)
```

```
## NULL
```

```
names(v)=c('A','B','C','D','E')
names(v)=='C'
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE
```

```
v[names(v)=='C']
```

```
## C
## 3
```

```
v[3]
```

```
## C
## 3
```

```
v[-3]
```

```
## A B D E
## 1 2 4 5
```

Chapter 10 R Commands

```
plot(y=ARMAacf(ma=c(0.5,rep(0,10),0.8,0.4),lag.max=13)[-1],x=1:13,type='h',xlab='Lag k',
ylab=expression(rho[k]),axes=F,ylim=c(0,0.6))
points(y=ARMAacf(ma=c(0.5,rep(0,10),0.8,0.4),lag.max=13)[-1],x=1:13,pch=20)
abline(h=0)
axis(1,at=1:13,
labels=c(1,NA,3,NA,5,NA,7,NA,9,NA,11,NA,13))
axis(2)
```

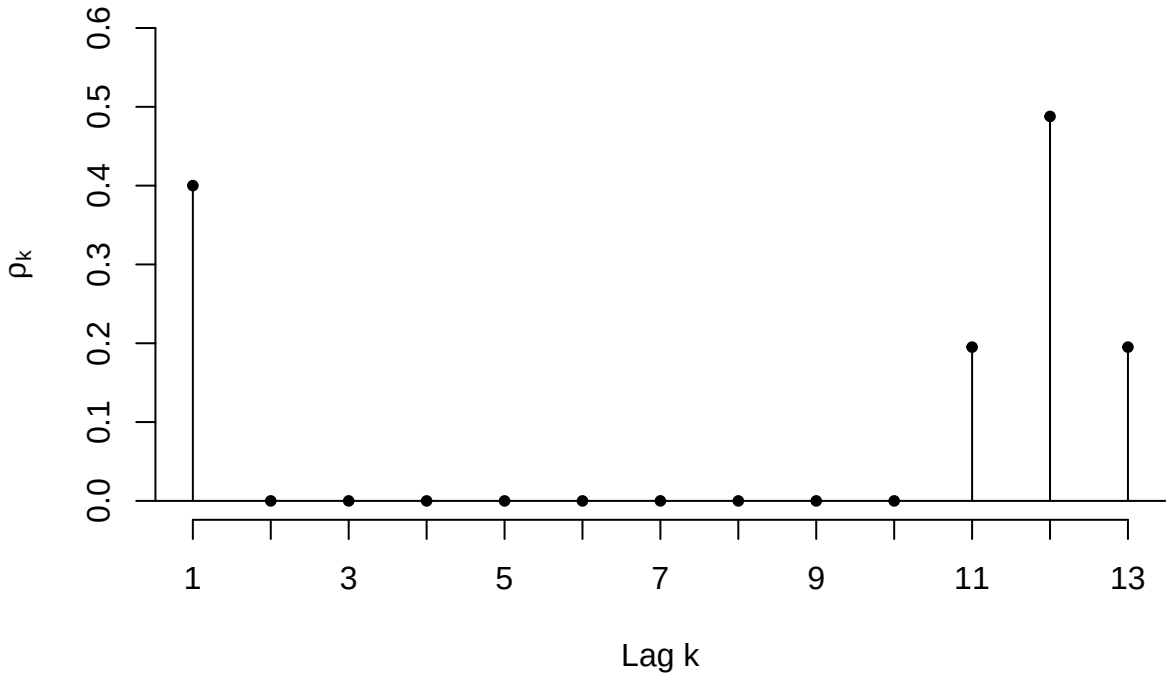


Exhibit 10.10

```
m1.co2=arima(co2,order=c(0,1,1),seasonal=list(order=c(0,1,1),period=12))
```

```
m1.co2
```

```
##
## Call:
## arima(x = co2, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12))
##
## Coefficients:
##          ma1     sma1
```

```
##       -0.3501  -0.8506
## s.e.   0.0496   0.0257
##
## sigma^2 estimated as 0.0826:  log likelihood = -86.08,  aic = 176.16
```

```
# This prints a summary of the fitted seasonal ARIMA model.
```

Question 2

Exercise 9.9 Simulate an AR(1) process with = 0.8 and = 100. Simulate 48 values but set aside the last 8 values to compare forecasts to actual values

```
set.seed(132456)
series=arima.sim(n=48,list(ar=0.8))+100
series
```

```
## Time Series:
## Start = 1
## End = 48
## Frequency = 1
##  [1]  99.32174 100.54041  99.18426 100.62163 100.70306 102.16264 101.18007
##  [8] 101.61750  99.17599 100.14493  99.77017  99.93022  99.42998 100.36642
## [15]  99.20188 101.42770 101.34206 101.54077 102.22879 102.08809 103.06755
## [22] 101.55893 101.39909 101.06198  99.14545  97.33168  97.70120  95.73272
## [29]  94.55397  96.66799  98.13306  99.88324 101.57048 102.75689 101.81881
## [36] 102.22101 100.98817 100.24840  98.71716  97.60278  98.08208  97.23044
## [43]  99.10154  99.92605  99.91232 100.52927 100.77788  99.19141
```

```
future=window(series,start=41)
series=window(series,end=40) # Set aside future
future
```

```
## Time Series:
## Start = 41
## End = 48
## Frequency = 1
## [1]  98.08208  97.23044  99.10154  99.92605  99.91232 100.52927 100.77788
## [8]  99.19141
```

(a) Using the first 40 values of the series, find the values for the maximum likelihood estimates of and

```
model=arima(series,order=c(1,0,0))
model
```

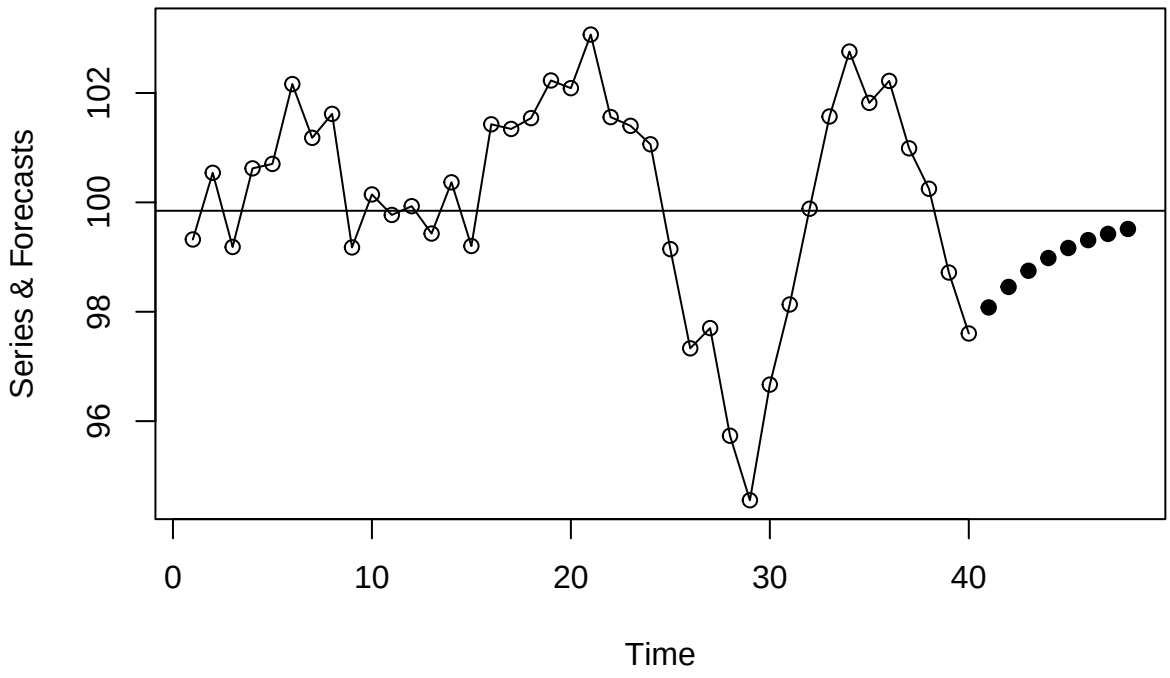
```
##
## Call:
## arima(x = series, order = c(1, 0, 0))
##
## Coefficients:
```

```
##          ar1  intercept
##       0.7878    99.8465
## s.e.  0.0943     0.8110
##
## sigma^2 estimated as 1.372:  log likelihood = -63.57,  aic = 131.14
```

The maximum likelihood estimates are quite accurate in this particular simulation.

(b) Using the estimated model, forecast the next eight values of the series. Plot the series together with the eight forecasts. Place a horizontal line at the estimate of the process mean

```
plot(model,n.ahead=8,ylab='Series & Forecasts',col=NULL,pch=19)
# col=NULL suppresses plotting the prediction intervals
abline(h=coef(model)[names(coef(model))=='intercept'])
```



(c) Compare the eight forecasts with the actual values that you set aside.

```
forecast_values <- predict(model, n.ahead = 8)$pred

comparison <- data.frame(Actual = future, Forecast = forecast_values)
print(comparison)
```

```
##      Actual Forecast
## 1  98.08208 98.07883
```

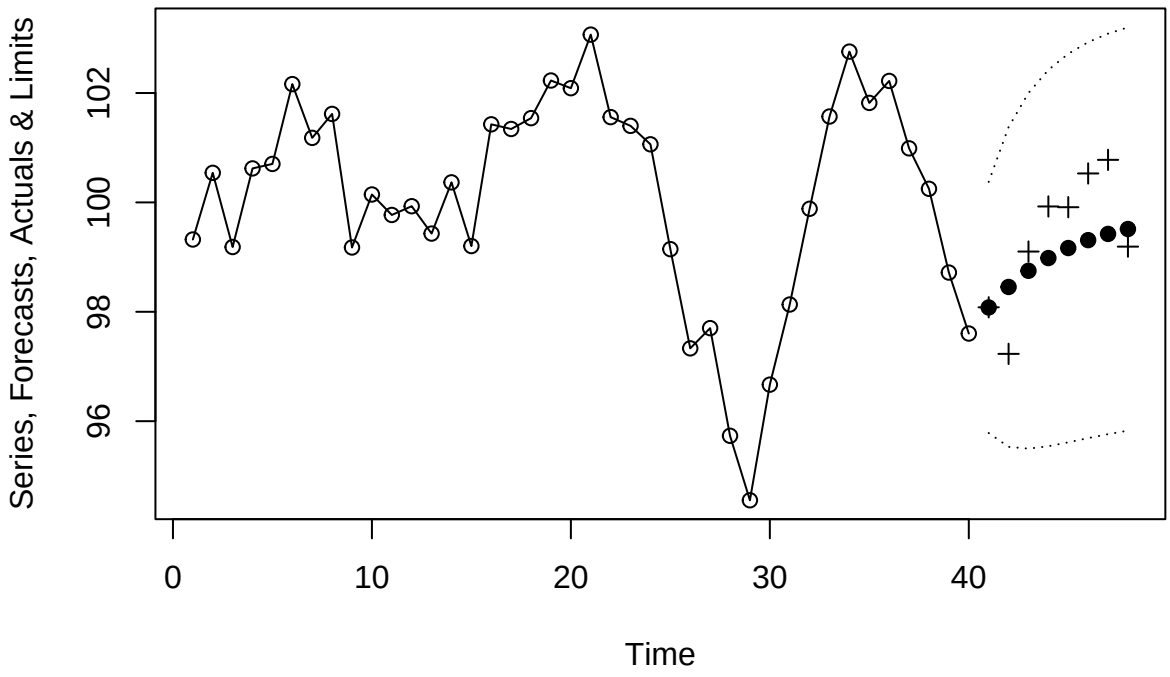
```
## 2  97.23044 98.45387
## 3  99.10154 98.74934
## 4  99.92605 98.98213
## 5  99.91232 99.16553
## 6 100.52927 99.31001
## 7 100.77788 99.42384
## 8  99.19141 99.51353
```

```r
error <- future - forecast_values
cat("Forecast errors: ", error, "\n")
```

```
## Forecast errors:  0.003249388 -1.223434 0.3521927 0.9439243 0.7467971 1.21926 1.354032 -0.3221114
```

```r
# can use cbind
```

```r
plot(model,n.ahead=8,ylab='Series, Forecasts, Actuals & Limits',pch=19)
points(x=(41:48),y=future,pch=3) # Add the actual future values to the plot
```

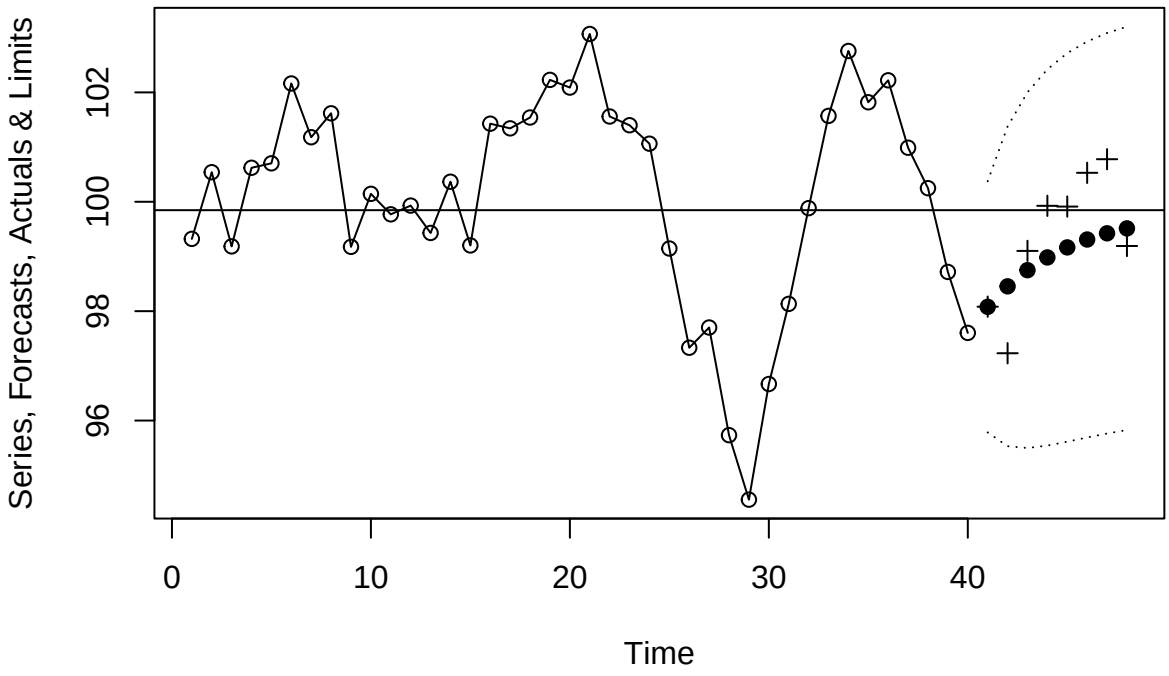


Actual future series values are plotted as plus signs (+)

The forecast errors are small, indicating that the AR(1) model has made accurate predictions, with values close to zero. The model's forecasts generally fall within the 95% confidence limits, suggesting good predictive performance.

(d) Plot the forecasts together with 95% forecast limits. Do the actual values fall within the forecast limits?

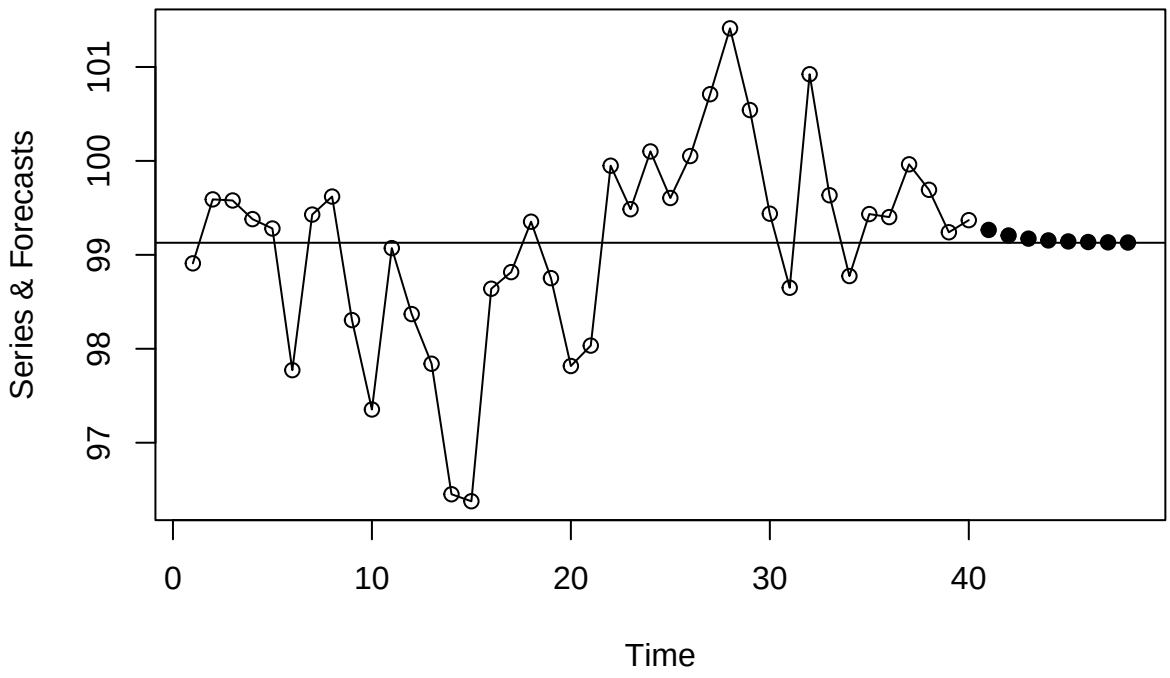
```r
# Including Prediction Intervals (95% by default)
plot(model,n.ahead=8,ylab='Series, Forecasts, Actuals & Limits',pch=19)
points(x=(41:48),y=future,pch=3) # Add the actual future values to the plot
abline(h=coef(model)[names(coef(model))=='intercept'])
```



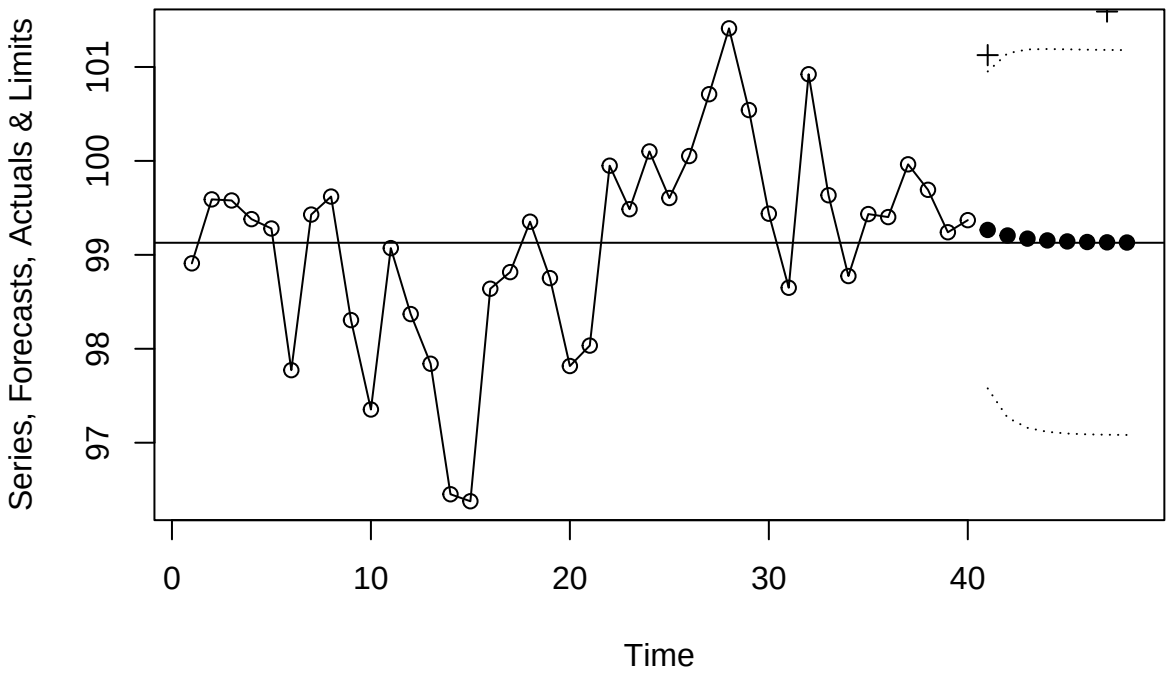
The forecast prediction limits shown as dotted lines. Yes it falls with in the limits.

(e) Repeat parts (a) through (d) with a new simulated series using the same values of the parameters and the same sample size

```r
set.seed(98765)
series=arima.sim(n=48,list(ar=0.8))+100
future=window(series,start=41)
series=window(series,end=40) # Set aside future
model=arima(series,order=c(1,0,0))
plot(model,n.ahead=8,ylab='Series & Forecasts',col=NULL,pch=19)
abline(h=coef(model)[names(coef(model))=='intercept'])
```




```
plot(model,n.ahead=8,ylab='Series, Forecasts, Actuals & Limits',pch=19)
points(x=(41:48),y=future,pch=3) # Add the actual future values to the plot
abline(h=coef(model)[names(coef(model))=='intercept'])
```



Exercise 9.12 Simulate an MA(2) process with $\ 1 = 1,\ 2 = -0.6$, and $\ = 100$. Simulate 36 values but set aside the last 4 values with compare forecasts to actual values.

```
set.seed(123)
series=arima.sim(n=36,list(ma=c(-1,0.6)))+100
actual=window(series,start=33)
series=window(series,end=32)
```

(a) Using the first 32 values of the series, find the values for the maximum likelihood estimates of the 's and

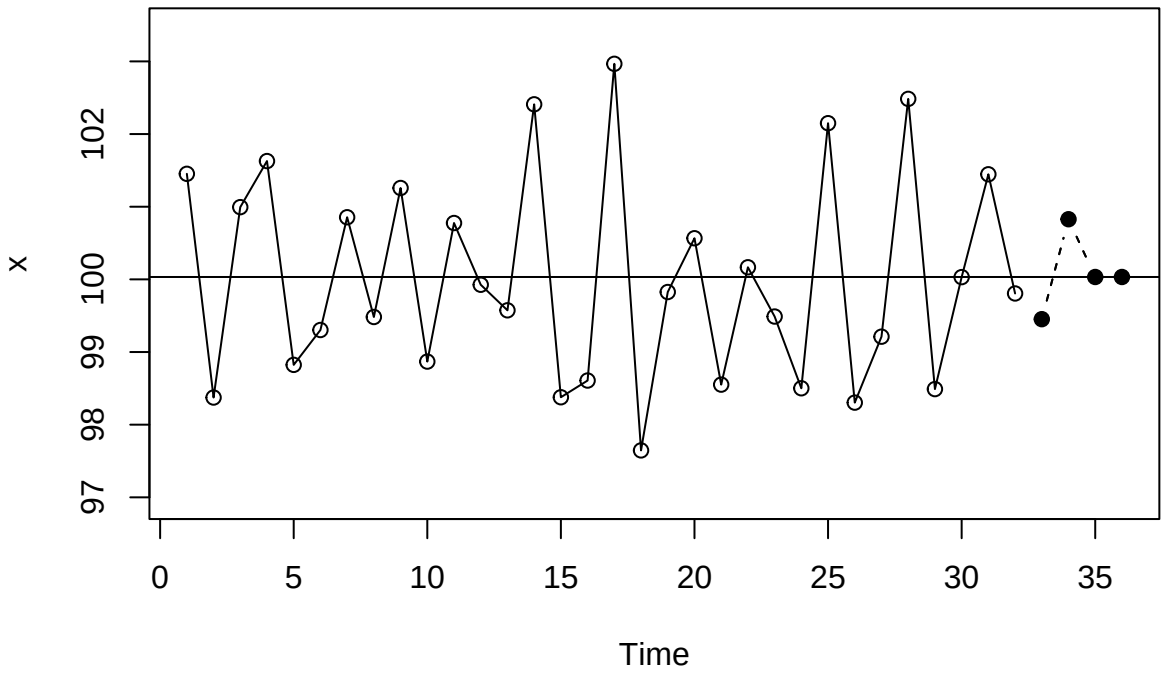
```
model=arima(series,order=c(0,0,2))
model
```

```
##
## Call:
## arima(x = series, order = c(0, 0, 2))
##
## Coefficients:
##           ma1     ma2  intercept
##       -1.2776  1.0000   100.0326
```

```
## s.e.   0.1403  0.1641     0.1023
##
## sigma^2 estimated as 0.6759:  log likelihood = -42.23,  aic = 90.47
```

(b) Using the estimated model, forecast the next four values of the series. Plot the series together with the four forecasts. Place a horizontal line at the estimate of the process mean.

```
result_graph = plot(model, n.ahead = 4, col = NULL, pch = 19)
abline(h=coef(model)[names(coef(model)) == "intercept"])
```



(c) What is special about the forecasts at lead times 3 and 4?

For the MA(2) model they are simply the estimated process mean. (data points are on th mean line)

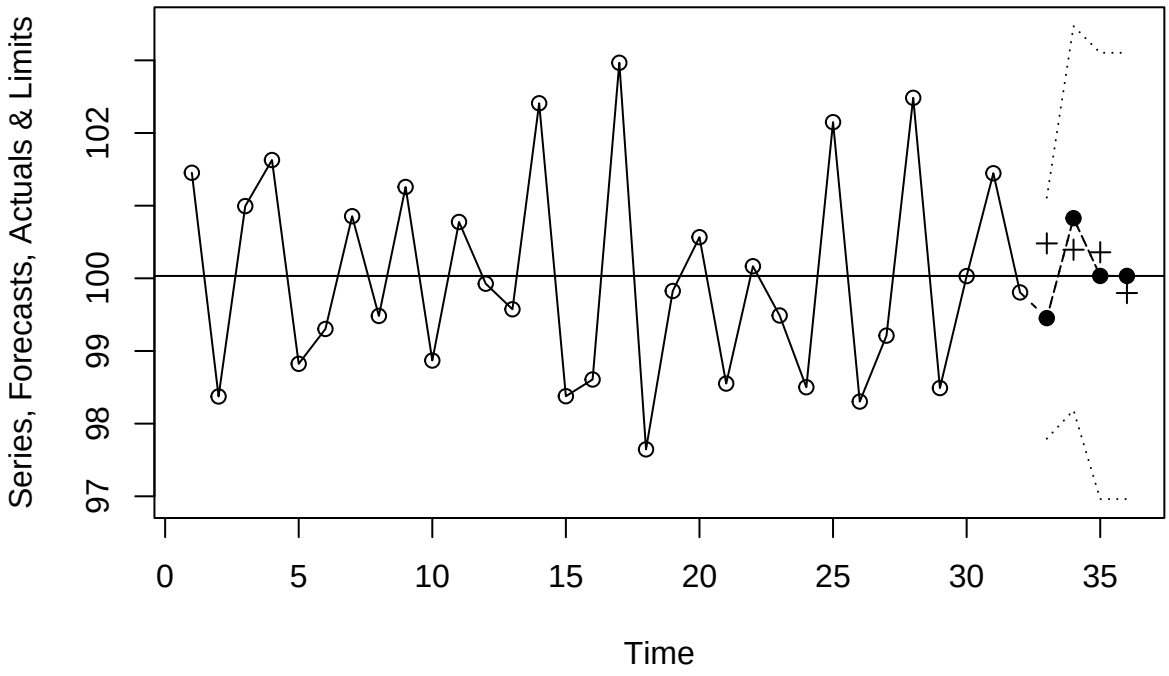
(d) Compare the four forecasts with the actual values that you set aside

```
cbind(actual,result_graph$pred)
```

```
## Time Series:
## Start = 33
## End = 36
## Frequency = 1
##       actual result_graph$pred
## 33 100.48052           99.4522
## 34 100.39394          100.8275
## 35 100.35823          100.0326
## 36  99.79735          100.0326
```

(e) Plot the forecasts together with 95% forecast limits. Do the actual values fall within the forecast limits?

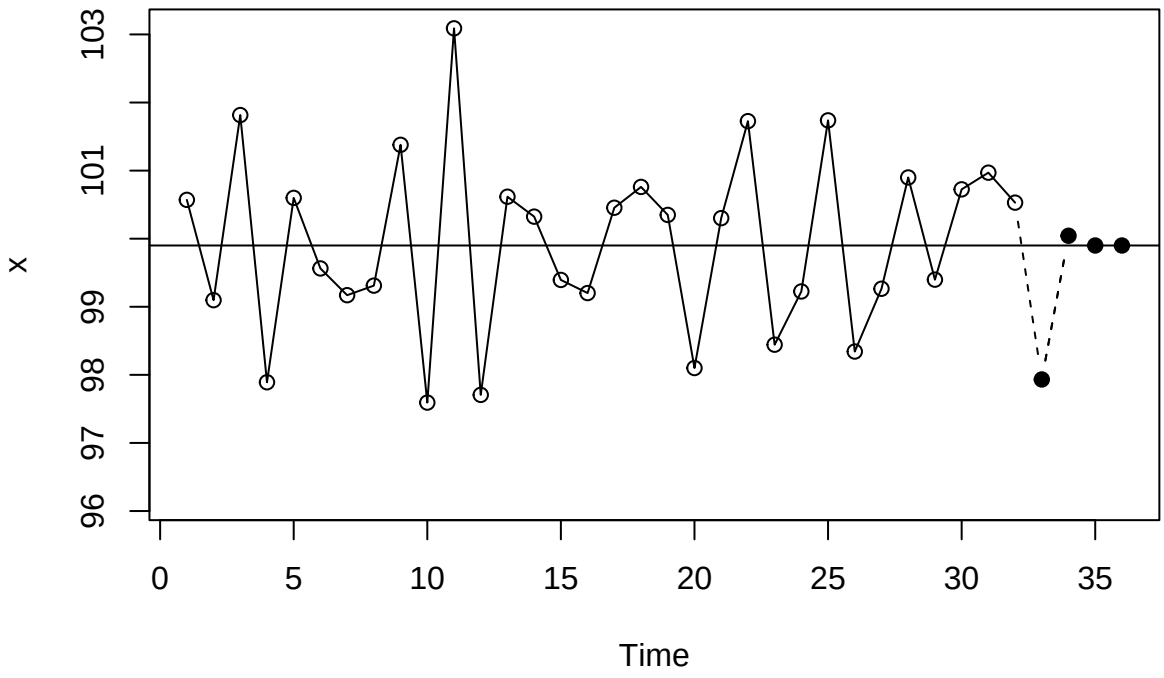
```
plot(model, n.ahead = 4, ylab = "Series, Forecasts, Actuals & Limits",type = 'o',pch = 19)
points(x=(33:36), y = actual, pch = 3)
abline(h=coef(model)[names(coef(model))=="intercept"])
```



The actuals are all within the forecast limits.

(f) Repeat parts (a) through (e) with a new simulated series using the same values of the parameters and same sample size.

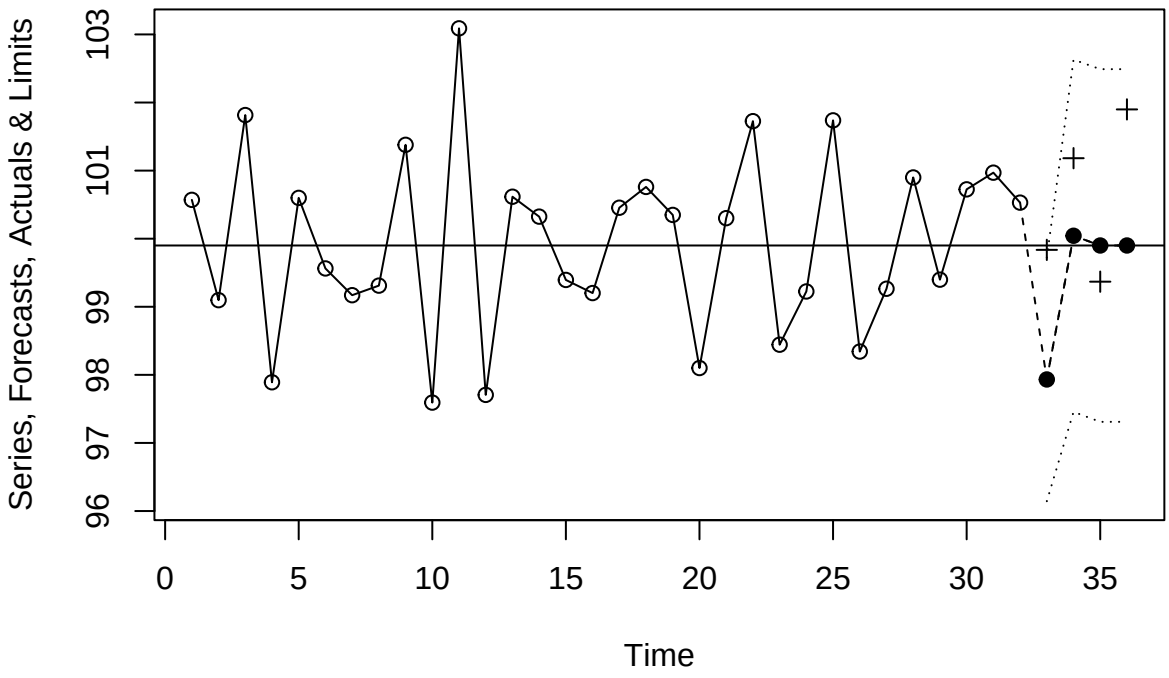
```
set.seed(987)
series=arima.sim(n=36,list(ma=c(-1,0.6)))+100
actual=window(series,start=33)
series=window(series,end=32)
model=arima(series,order=c(0,0,2))
result_graph = plot(model, n.ahead = 4, col = NULL, pch = 19)
abline(h=coef(model)[names(coef(model)) == "intercept"])
```




```
cbind(actual,result_graph$pred)
```

```
## Time Series:
## Start = 33
## End = 36
## Frequency = 1
##       actual result_graph$pred
## 33  99.83595          97.93229
## 34 101.18218         100.04310
## 35  99.36855          99.89990
## 36 101.89890          99.89990
```

```
plot(model, n.ahead = 4, ylab = "Series, Forecasts, Actuals & Limits",type = 'o',pch = 19)
points(x=(33:36), y = actual, pch = 3)
abline(h=coef(model)[names(coef(model))=="intercept"])
```



For this simulation and this model, the forecasts are rather far from the actual values.

Exercise 9.13 Simulate an ARMA(1,1) process with $= 0.7$, $= -0.5$, and $= 100$. Simulate 50 values but set aside the last 10 values to compare forecasts with actual values

```
set.seed(5323)
series = arima.sim(n=50, list(ar=0.7,ma=-0.5))+100
actual = window(series, start = 41)
series_to_40 = window(series, end=40)
```

(a) Using the first 40 values of the series, find the values for the maximum likelihood estimates of , , and

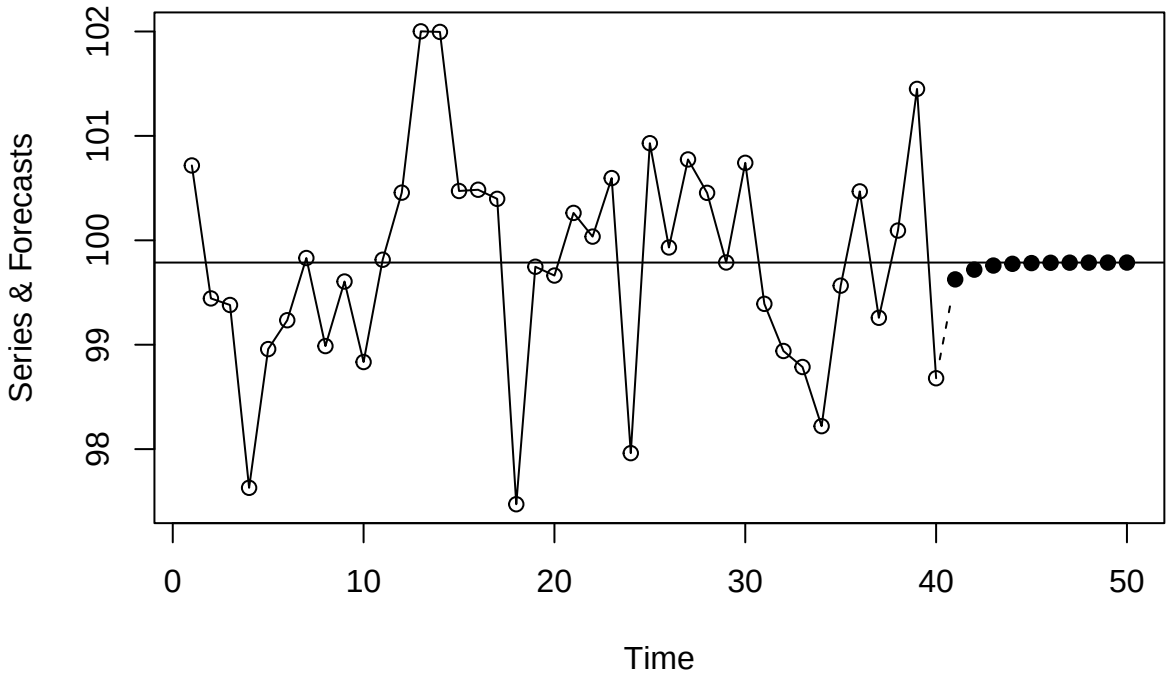
```
model = arima(series_to_40,order=c(1,0,1))
model
```

```
##
## Call:
## arima(x = series_to_40, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
```

```
##       0.4212  -0.2074    99.7869
## s.e.  0.3562   0.3679     0.2145
##
## sigma^2 estimated as 1.004:  log likelihood = -56.87,  aic = 119.73
```

(b) Using the estimated model, forecast the next ten values of the series. Plot the series together with the ten forecasts. Place a horizontal line at the estimate of the process mean.

```
result_graph = plot(model, n.ahead=10, ylab = "Series & Forecasts", col = NULL, pch=19)
abline(h = coef(model)[names(coef(model))=="intercept"])
```



The forecfasts approach the series mean fairly quickly.

(c) Compare the ten forecasts with the actual values that you set aside.

```
cbind(actual, result_graph$pred)
```

```
## Time Series:
## Start = 41
## End = 50
## Frequency = 1
##       actual result_graph$pred
## 41  99.14840          99.62577
## 42  98.92234          99.71901
```

```
## 43 101.17251          99.75828
## 44 101.80098          99.77482
## 45 100.13930          99.78179
## 46 100.89658          99.78472
## 47 100.79189          99.78596
## 48  99.27489          99.78648
## 49  99.86969          99.78670
## 50 101.25117          99.78679
```

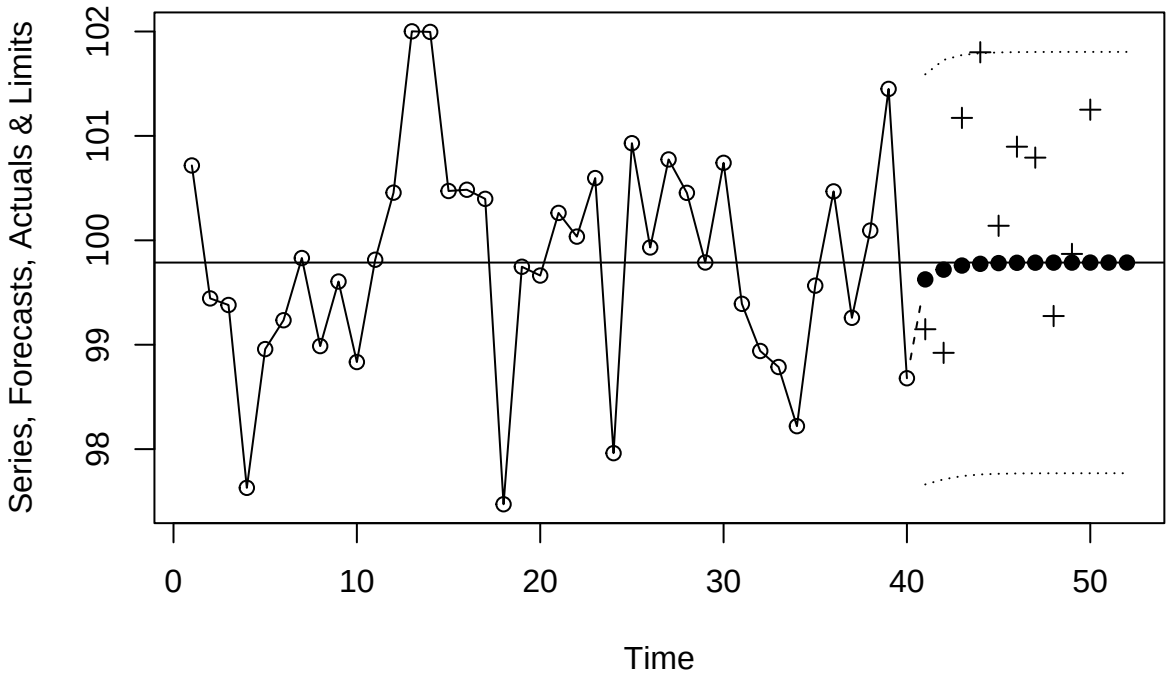
The values are quite close to each other

(d) Plot the forecasts together with 95% forecast limits. Do the actual values fall within the forecast limits?

```
plot(model, n.ahed=10,ylab='Series, Forecasts, Actuals & Limits', pch = 19)
```

```
## Warning in plot.window(xlim, ylim, log, ...): "n.ahed" is not a graphical
## parameter

## Warning in title(main = main, xlab = xlab, ylab = ylab, ...): "n.ahed" is not a
## graphical parameter

## Warning in axis(1, ...): "n.ahed" is not a graphical parameter

## Warning in axis(2, ...): "n.ahed" is not a graphical parameter

## Warning in box(...): "n.ahed" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "n.ahed" is not a
## graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "n.ahed" is not a
## graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "n.ahed" is not a
## graphical parameter
```

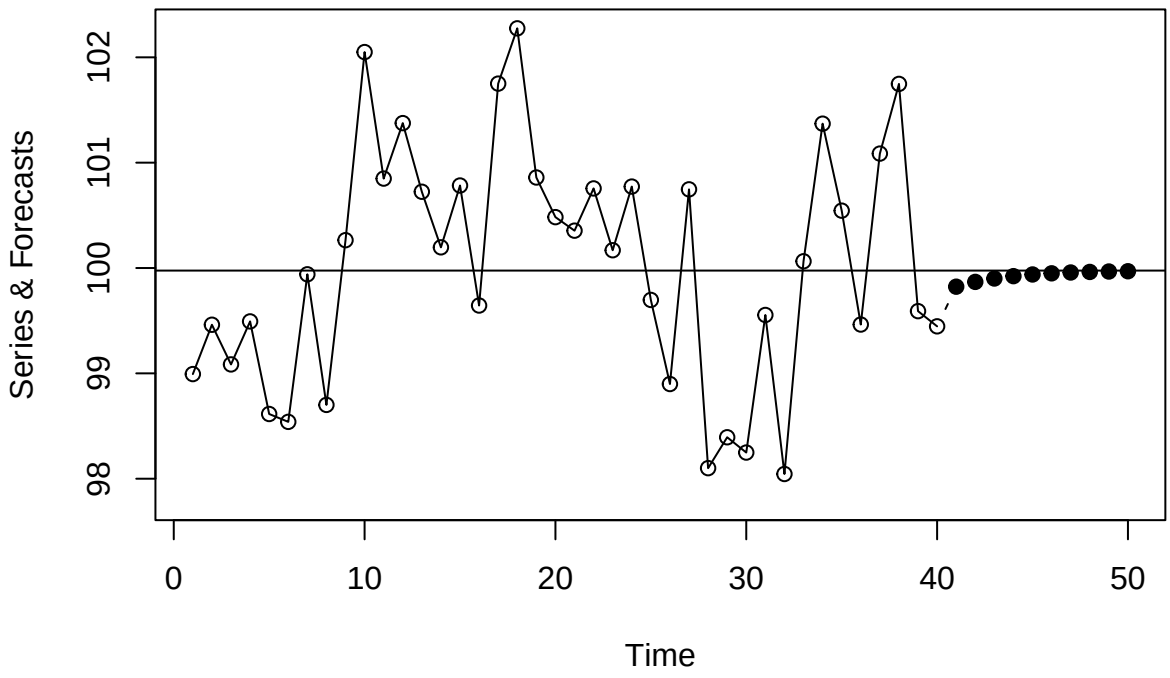
```
points(x=(41:50), y = actual, pch = 3)
abline(h = coef(model)[names(coef(model))=="intercept"])
```



The actual values are within the forecast limits but forecasts decay to the estimated process mean rather quickly and the prediction limits are quite wide.

(e) Repeat parts (a) through (d) with a new simulated series using the same values of the parameters and same sample size.

```
set.seed(2353)
series = arima.sim(n=50, list(ar=0.7,ma=-0.5))+100
actual = window(series, start = 41)
series_to_40 = window(series, end=40)
model = arima(series_to_40,order=c(1,0,1))
result_graph = plot(model, n.ahead=10, ylab = "Series & Forecasts", col = NULL, pch=19)
abline(h = coef(model)[names(coef(model))=="intercept"])
```




```
cbind(actual, result_graph$pred)
```

```
## Time Series:
## Start = 41
## End = 50
## Frequency = 1
##       actual result_graph$pred
## 41 100.08924          99.82260
## 42 100.68292          99.86825
## 43 100.31205          99.90028
## 44  99.60758          99.92276
## 45 100.82513          99.93853
## 46  99.74718          99.94959
## 47 100.89360          99.95735
## 48 100.11310          99.96280
## 49  99.59170          99.96662
## 50  99.60645          99.96930
```

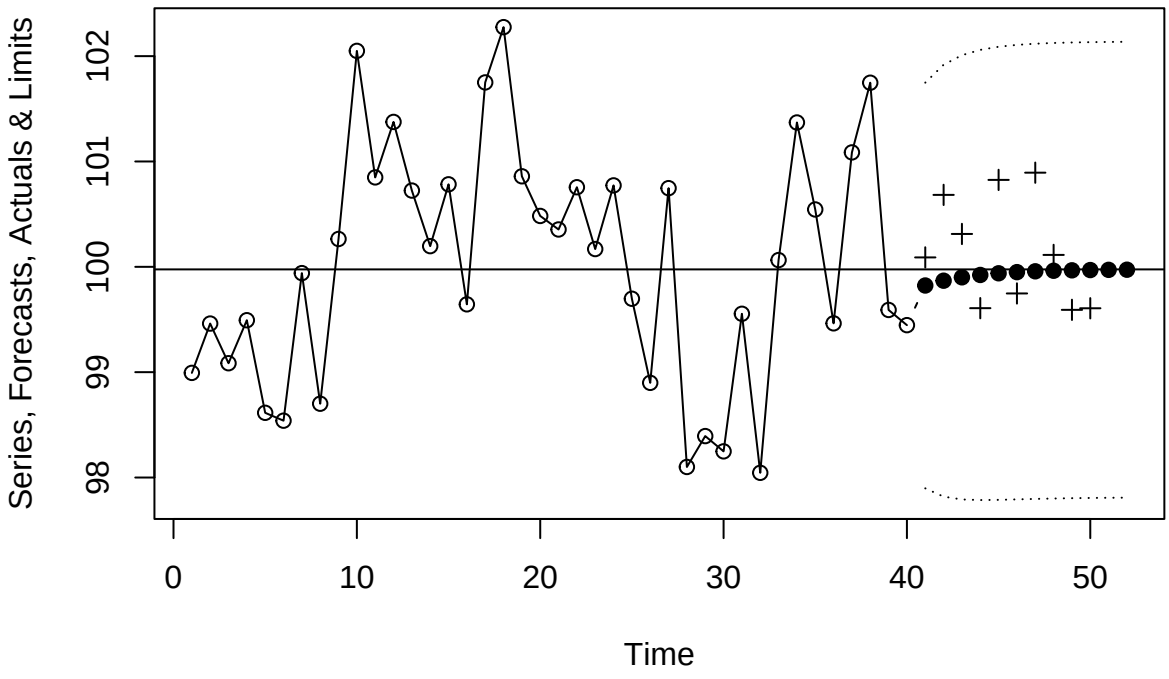
```
plot(model, n.ahed=10,ylab='Series, Forecasts, Actuals & Limits', pch = 19)
```

```
## Warning in plot.window(xlim, ylim, log, ...): "n.ahed" is not a graphical
## parameter
```

```
## Warning in title(main = main, xlab = xlab, ylab = ylab, ...): "n.ahed" is not a
## graphical parameter
```

```
## Warning in axis(1, ...): "n.ahed" is not a graphical parameter

## Warning in axis(2, ...): "n.ahed" is not a graphical parameter

## Warning in box(...): "n.ahed" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "n.ahed" is not a
## graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "n.ahed" is not a
## graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "n.ahed" is not a
## graphical parameter
```

```
points(x=(41:50), y = actual, pch = 3)
abline(h = coef(model)[names(coef(model))=="intercept"])
```



Exercise 9.16 Simulate an IMA(2,2) process with $\theta_1 = 1$, $\theta_2 = -0.75$, and $\theta_0 = 0$. Simulate 45 values, but set aside the last five values to compare forecasts with actual values.

```
set.seed(8899)
series = arima.sim(n = 45, list(order = c(0, 2, 2), ma = c(-1, 0.75)))
```

```
# series = arima.sim(n = 45, list(ma = c(-1, 0.75)))
series = (series[-1])[-1]
actual = window(series,start = 41)
series_upto_40 = window(series, end=40)
series
```

```
##  [1]   -0.4451137   -1.4091980   -3.4278513   -6.6341654   -6.3568065
##  [6]   -8.3667047  -11.4893176  -12.5944253  -13.5215139  -14.6795142
## [11]  -16.9326527  -17.7348829  -18.9447112  -20.5897099  -21.6184994
## [16]  -23.0420777  -25.0185752  -26.1402509  -29.5212823  -31.2011344
## [21]  -33.1955634  -35.1118104  -36.6386615  -38.8893063  -39.3983149
## [26]  -39.8087391  -41.5074975  -41.7382600  -44.6557347  -47.2895735
## [31]  -50.4134054  -54.9727334  -58.0817164  -64.2203362  -68.8251605
## [36]  -73.0989777  -77.4711434  -81.1571110  -84.4664929  -89.3466802
## [41]  -93.0098986  -96.5500250 -100.5211211 -104.5557101 -107.1139236
```

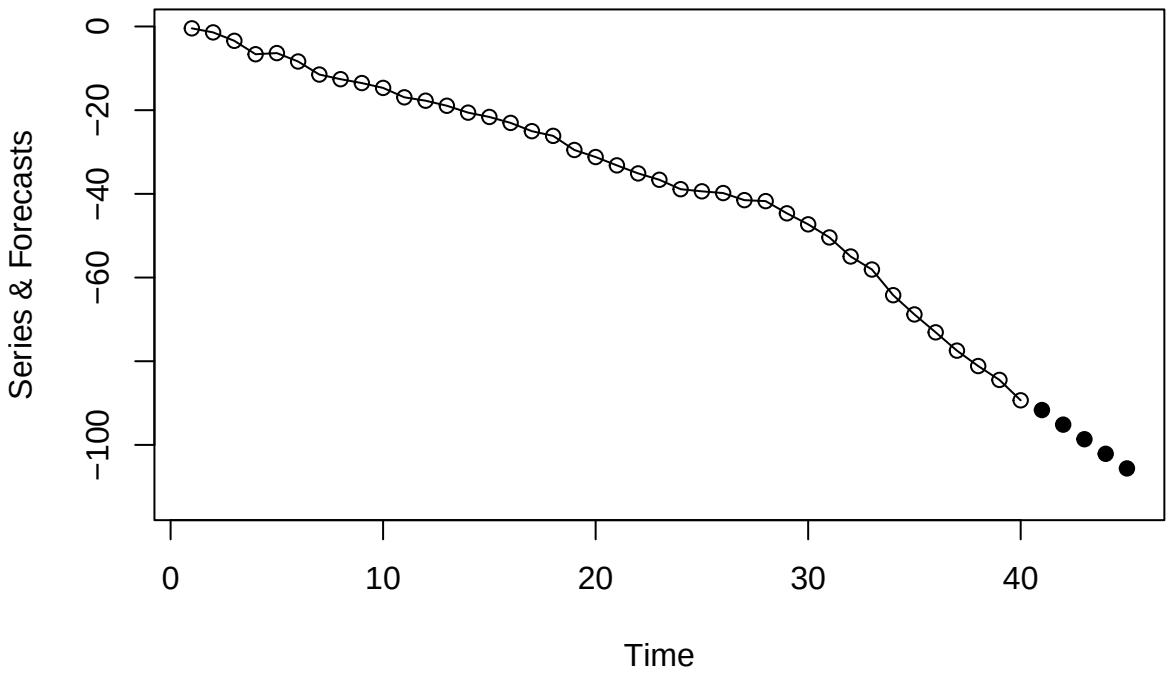
(a) Using the first 40 values of the series, find the value for the maximum likelihood estimate of 1 and 2.

```
model=arima(series_upto_40,order=c(0,2,2))
model
```

```
##
## Call:
## arima(x = series_upto_40, order = c(0, 2, 2))
##
## Coefficients:
##           ma1     ma2
##       -1.2654  1.0000
## s.e.   0.1037  0.1357
##
## sigma^2 estimated as 0.8906:  log likelihood = -54.97,  aic = 113.94
```

(b) Using the estimated model, forecast the next five values of the series. Plot the series together with the five forecasts. What is special about the forecasts?

```
result=plot(model,n.ahead=5,ylab='Series & Forecasts',col=NULL,pch=19)
```



The forecasts seem to follow a straight line.

(c) Compare the five forecasts with the actual values that you set aside.

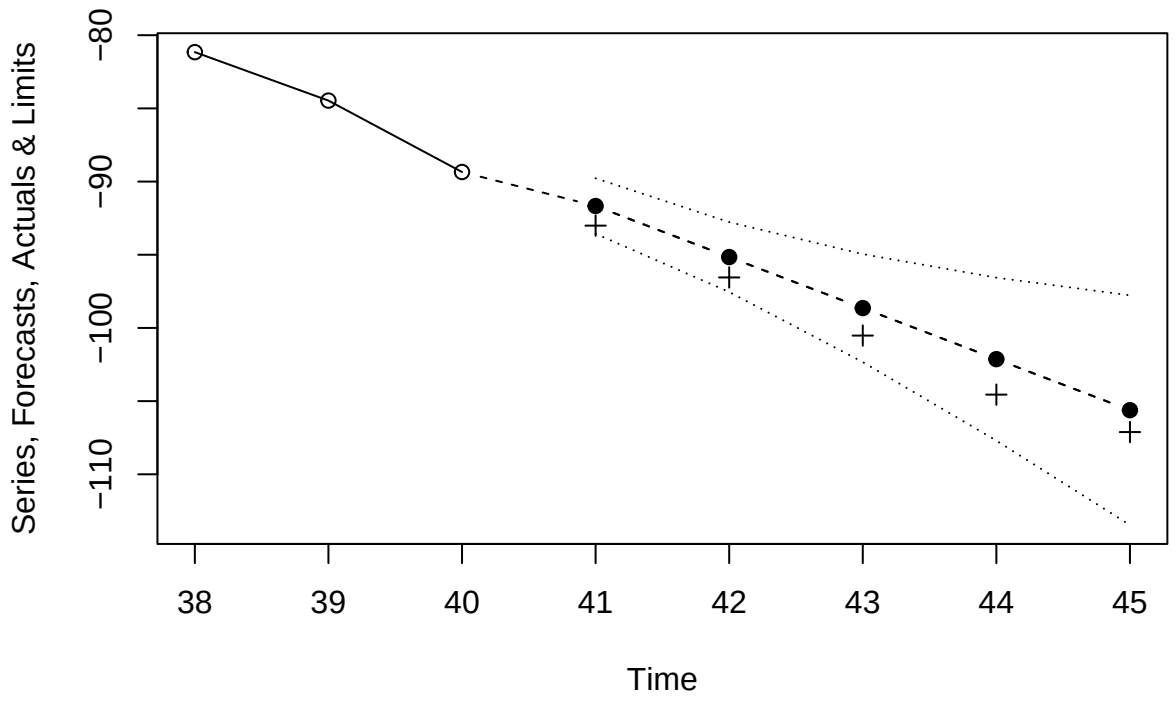
```
forecast=result$pred
cbind(actual,forecast)
```

```
## Time Series:
## Start = 41
## End = 45
## Frequency = 1
##        actual   forecast
## 41  -93.00990  -91.66983
## 42  -96.55003  -95.15725
## 43 -100.52112  -98.64467
## 44 -104.55571 -102.13209
## 45 -107.11392 -105.61951
```

All of the forecasts are a bit higher than the actual values

(d) Plot the forecasts together with 95% forecast limits. Do the actual values fall within the forecast limits?

```
plot(model,n1=38,n.ahead=5,ylab='Series, Forecasts, Actuals & Limits', pch=19)
points(x=seq(41,45),y=actual,pch=3)
```



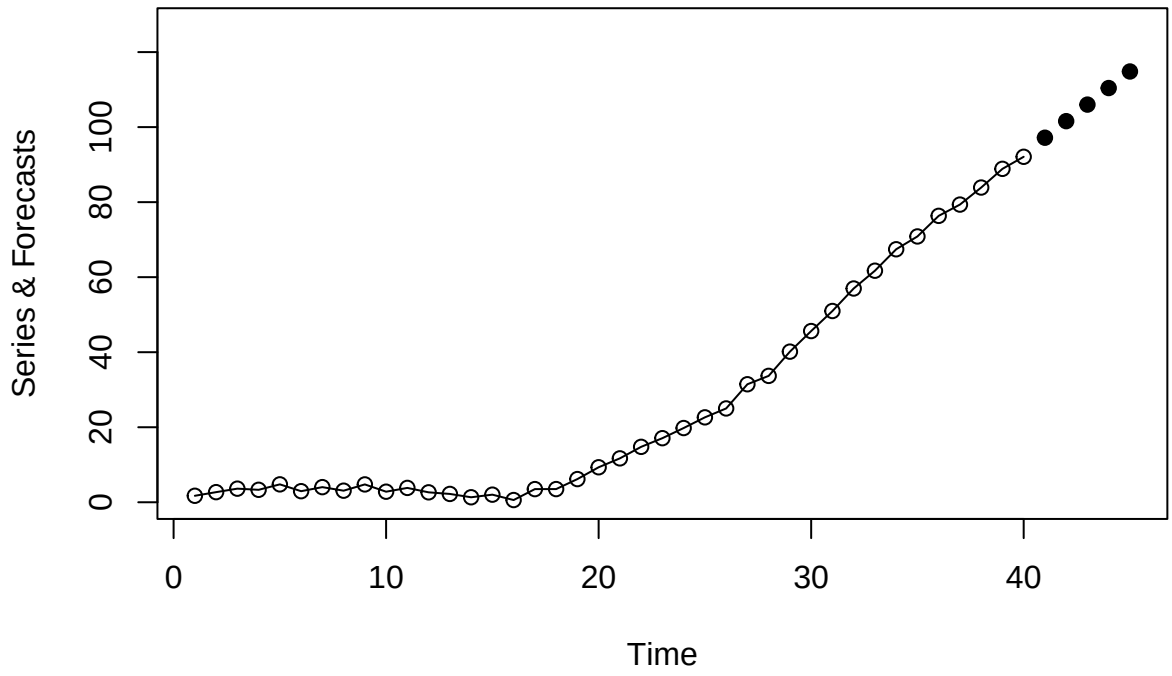
Yes, actual values fall within the forecast limits. As the lead time increases, the forecast limits widen, which is typical for nonstationary models. The forecast for lead time 1 is very close to the lower forecast limit.

(e) Repeat parts (a) through (d) with a new simulated series using the same values of the parameters and same sample size.

```
set.seed(9876)
series = arima.sim(n = 45, list(order = c(0, 2, 2), ma = c(-1, 0.75)))
# series = arima.sim(n = 45, list(ma = c(-1, 0.75)))
series = (series[-1])[-1]
actual = window(series,start = 41)
series_upto_40 = window(series, end=40)
series
```

```
##  [1]   1.7316181   2.6944340   3.6259939   3.3200515   4.7728511   2.9528472
##  [7]   4.0340646   3.0900245   4.7555550   2.8175124   3.8304772   2.6471112
## [13]   2.2231767   1.3356895   2.0304483   0.5953426   3.5052157   3.5142463
## [19]   6.1782882   9.3279490  11.6999732  14.7715973  17.0841475  19.7829442
## [25]  22.6331389  25.0117129  31.4537359  33.6902474  40.1479809  45.6399023
## [31]  50.9812665  56.9795691  61.7257705  67.4420304  70.8551539  76.3483011
## [37]  79.3536521  83.8963268  88.8794644  92.0925672  97.9872728 103.0996905
## [43] 106.9582046 113.1669728 116.8845109
```

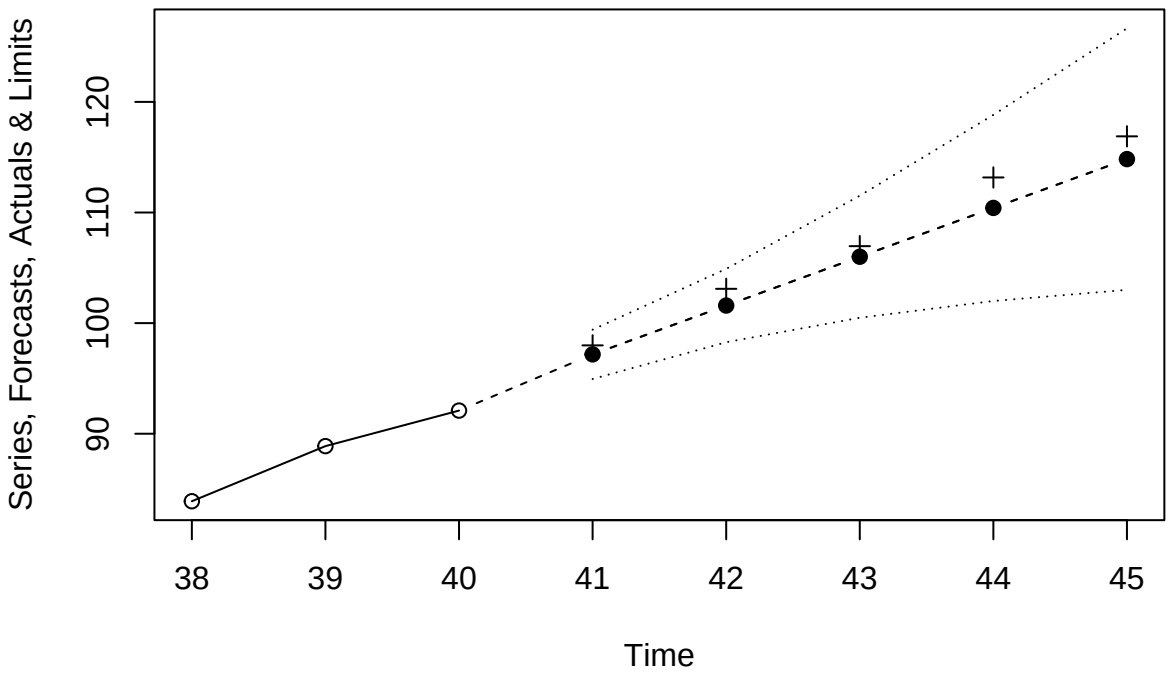
```
model=arima(series_upto_40,order=c(0,2,2))
result=plot(model,n.ahead=5,ylab='Series & Forecasts',col=NULL,pch=19)
```




```
forecast=result$pred
cbind(actual,forecast)
```

```
## Time Series:
## Start = 41
## End = 45
## Frequency = 1
##       actual  forecast
## 41  97.98727  97.17485
## 42 103.09969 101.58773
## 43 106.95820 106.00062
## 44 113.16697 110.41350
## 45 116.88451 114.82638
```

```
plot(model,n1=38,n.ahead=5,ylab='Series, Forecasts, Actuals & Limits', pch=19)
points(x=seq(41,45),y=actual,pch=3)
```



Exercise 10.8 Consider the Alert, Canada, monthly carbon dioxide time series shown in Exhibit (10.1), page 227. The data are in the file named co2.

(a) Fit a deterministic seasonal means plus linear time trend model to these data. Are any of the regression coefficients "statistically significant"?

```
data(co2)
month.=season(co2)
trend=time(co2)
model=lm(co2~month.+trend)
summary(model)
```

```
##
## Call:
## lm(formula = co2 ~ month. + trend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.73874 -0.59689 -0.06947  0.54086  2.15539
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -3290.5412    44.1790 -74.482  < 2e-16 ***
## month.February      0.6682     0.3424   1.952 0.053320 .
```

```
## month.March         0.9637     0.3424   2.815 0.005715 **
## month.April         1.2311     0.3424   3.595 0.000473 ***
## month.May           1.5275     0.3424   4.460 1.87e-05 ***
## month.June         -0.6761     0.3425  -1.974 0.050696 .
## month.July         -7.2851     0.3426 -21.267  < 2e-16 ***
## month.August      -13.4414     0.3426 -39.232  < 2e-16 ***
## month.September   -12.8205     0.3427 -37.411  < 2e-16 ***
## month.October      -8.2604     0.3428 -24.099  < 2e-16 ***
## month.November     -3.9277     0.3429 -11.455  < 2e-16 ***
## month.December     -1.3367     0.3430  -3.897 0.000161 ***
## trend               1.8321     0.0221  82.899  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8029 on 119 degrees of freedom
## Multiple R-squared:  0.9902, Adjusted R-squared:  0.9892
## F-statistic: 997.7 on 12 and 119 DF,  p-value: < 2.2e-16
```

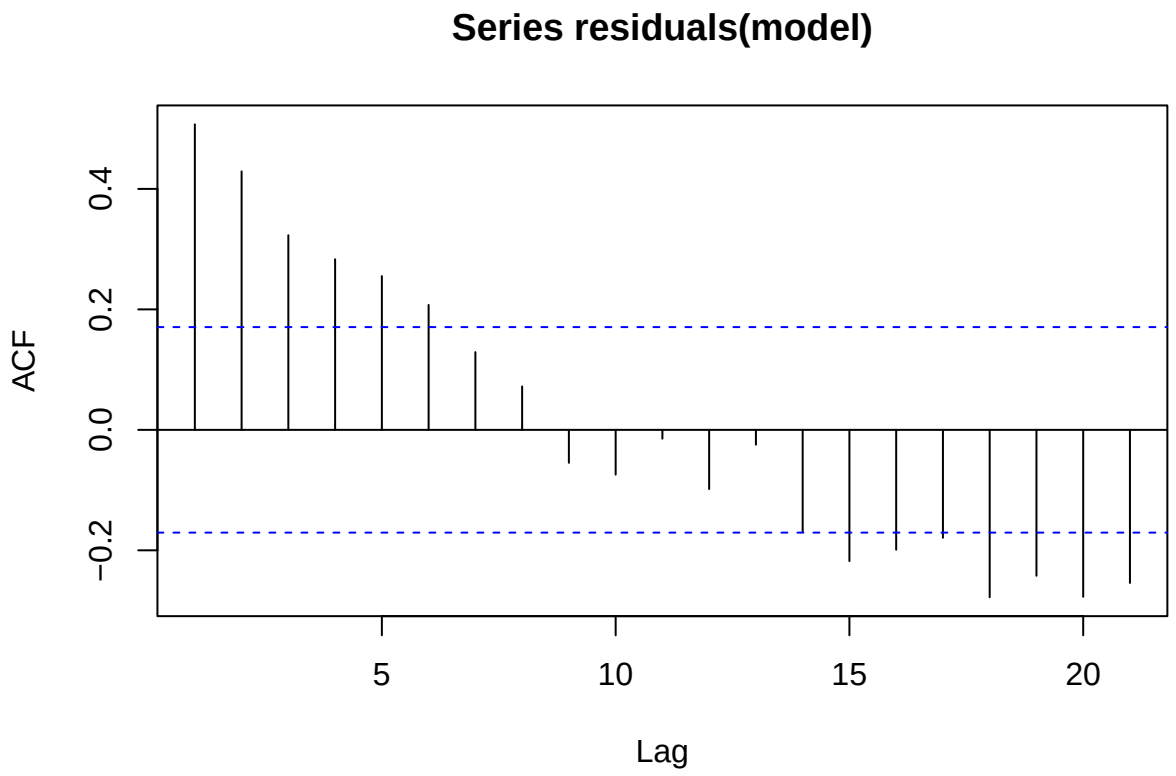
All of the regression coefficients are statistically significant except for the seasonal effects for February and June. Those two have p-values just above 0.05.

(b) What is the multiple R-squared for this model?

Multiple R-squared: 0.9902

(c) Now calculate the sample autocorrelation of the residuals from this model. Interpret the results.

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

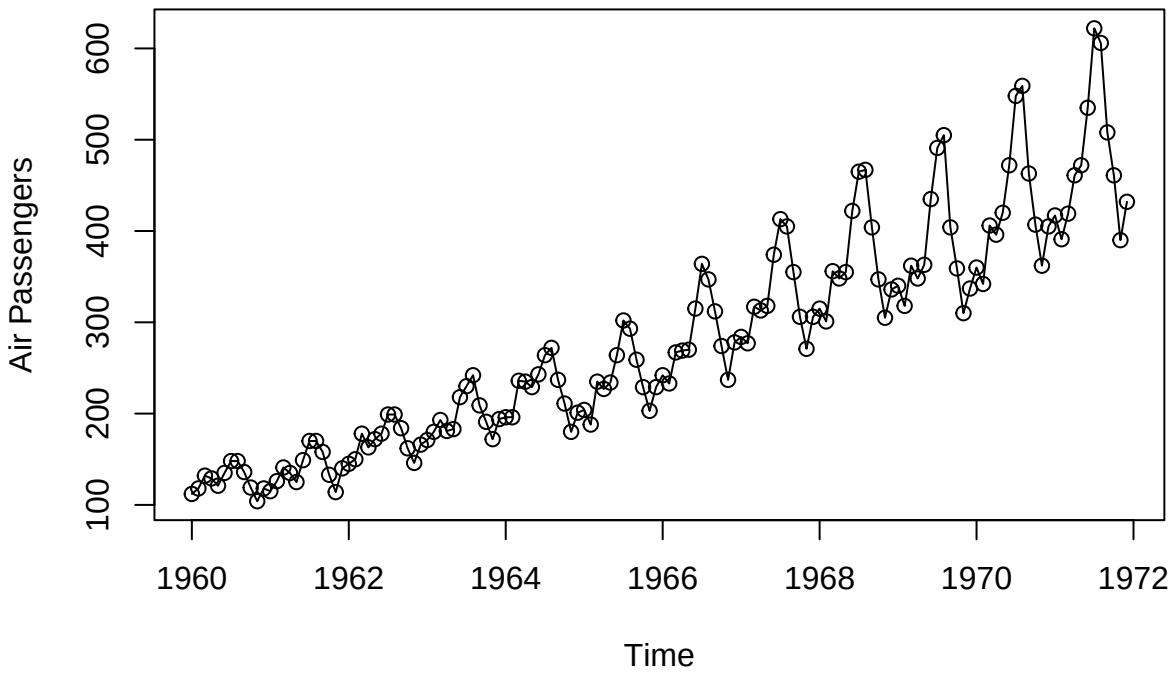


Clearly, this deterministic trend model has not captured the autocorrelation in this time series. The seasonal ARIMA model illustrated in Chapter 10 is a much better model for these data.

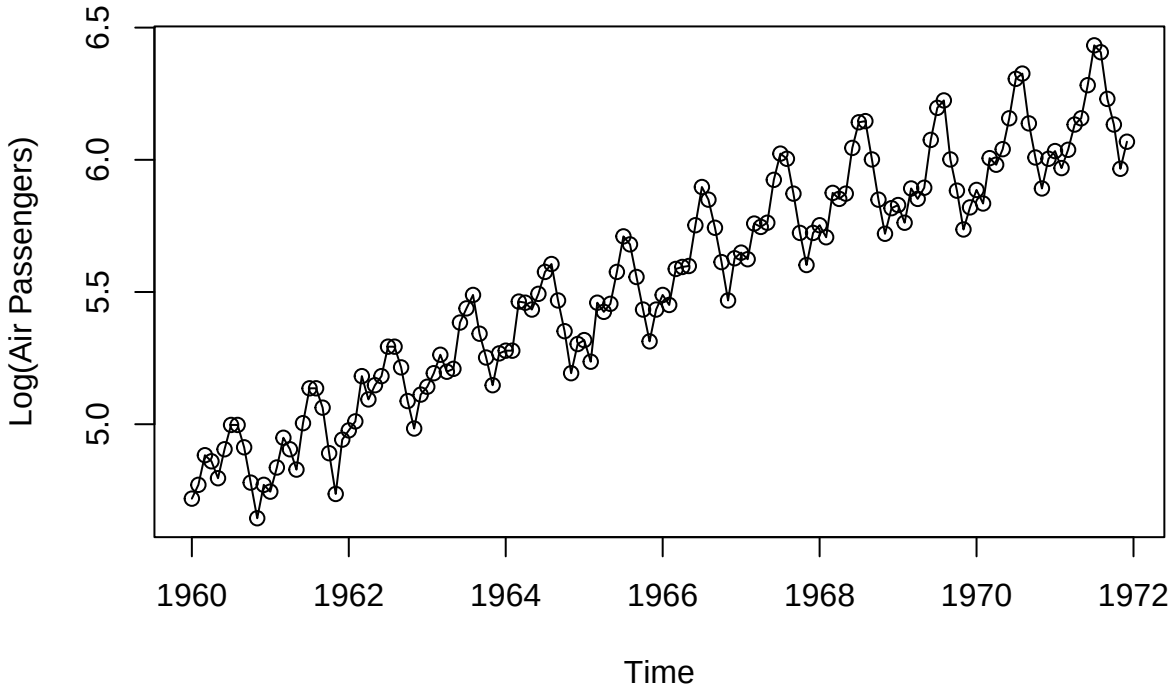
Exercise 10.9 The monthly airline passenger time series, first investigated in Box and Jenkins (1976), is considered a classic time series. The data are in the file named airline. [Typo: The filename is airpass.]

(a) Display the time series plots of both the original series and the logarithms of the series. Argue that taking logs is an appropriate transformation.

```
data(airpass)
plot(airpass, type='o',ylab='Air Passengers')
```



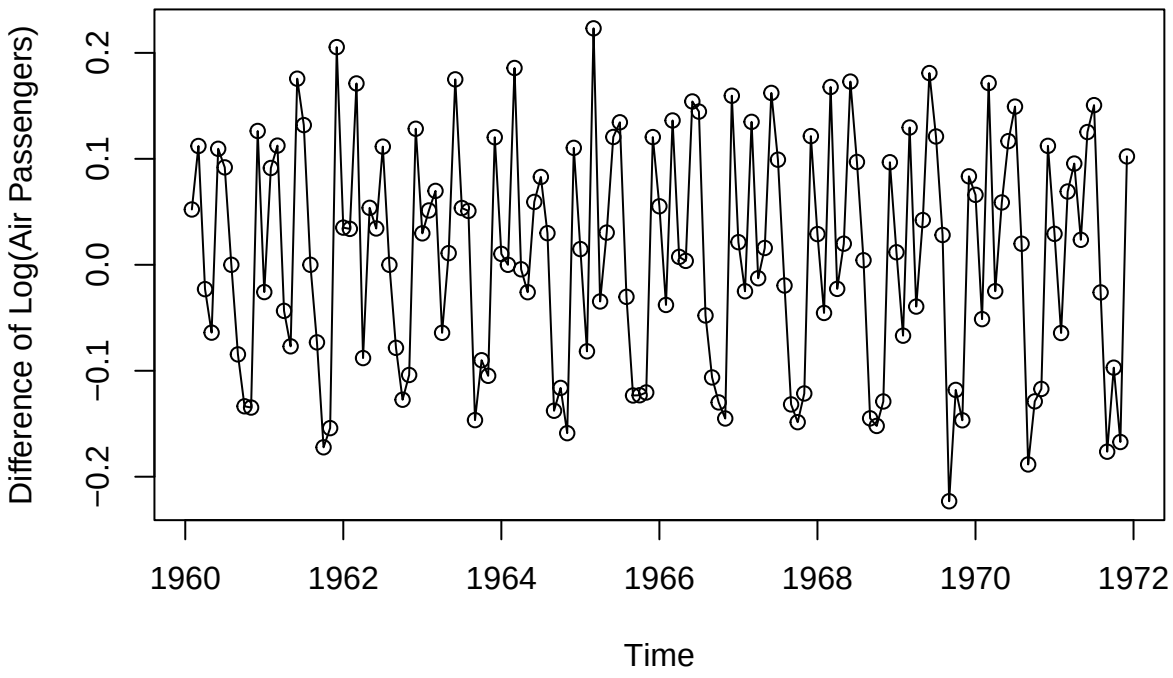

```
plot(log(airpass), type='o',ylab='Log(Air Passengers)')
```



In the original series, the amplitude of seasonal fluctuations increases over time, indicating a non-constant variance. In the log-transformed series, the seasonal fluctuations appear more stable, indicating that the logarithmic transformation has stabilized the variance.

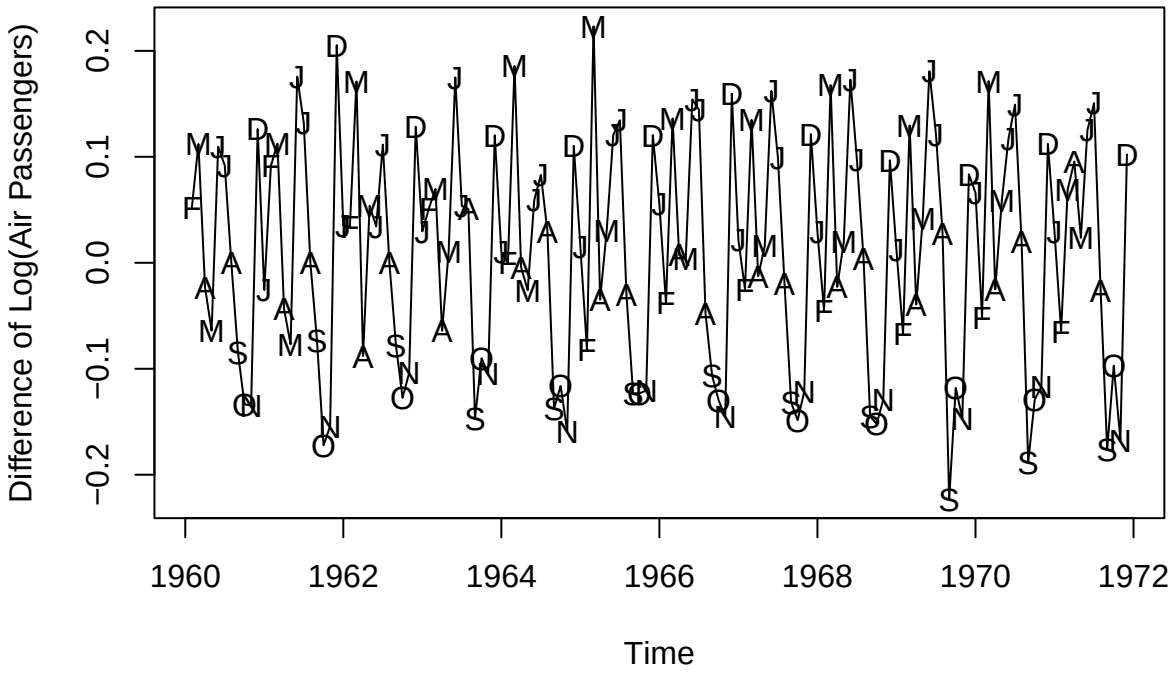
(b) Display and interpret the time series plots of the first difference of the logged series.

```
plot(diff(log(airpass)),type='o',ylab='Difference of Log(Air Passengers)')
```



The first difference of the logged airline passenger series removes the trend, making the series approximately stationary. The seasonal pattern is still visible, indicating periodic fluctuations in the rate of change of passengers. This transformation stabilizes the variance and prepares the series for time series modeling like ARIMA.

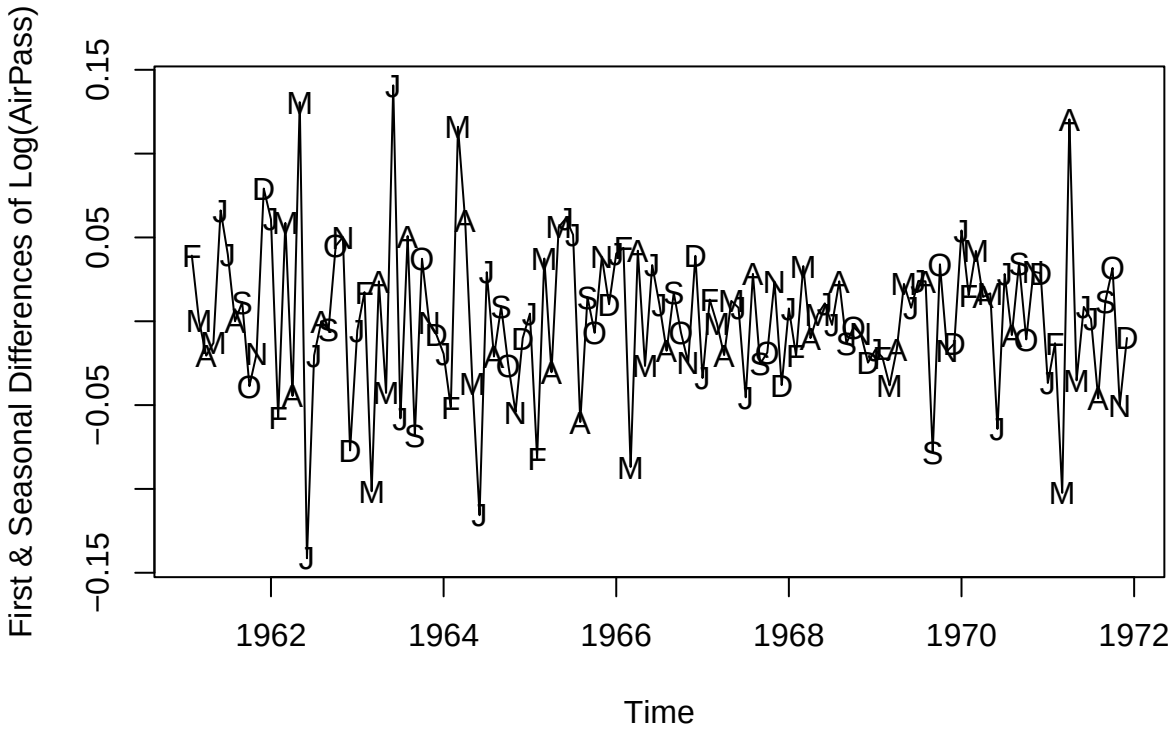
```
plot(diff(log(airpass)),type='l',ylab='Difference of Log(Air Passengers)')
points(diff(log(airpass)),x=time(diff(log(airpass))),pch=as.vector(season(diff(log(airpass)))))
```



The plot shows the first differenced logarithm of airline passenger data, with points labeled by their corresponding seasons (months). The seasonal patterns are evident, as similar fluctuations repeat annually, confirming the importance of seasonality. This transformation stabilizes variance and removes the trend, making the series suitable for seasonal time series modeling like SARIMA.

(c) Display and interpret the time series plot of the seasonal difference of the first difference of the logged series

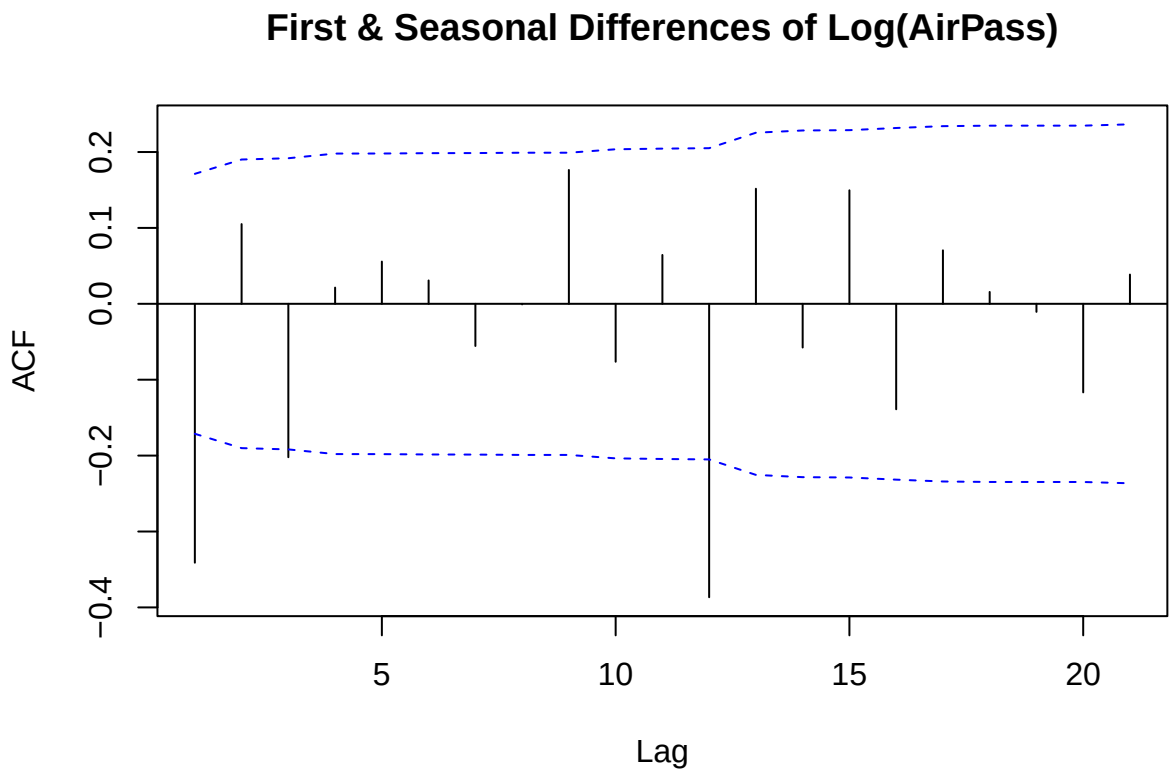
```
plot(diff(diff(log(airpass)),lag=12),type='l', ylab='First & Seasonal Differences of Log(AirPass)')
points(diff(diff(log(airpass)),lag=12),x=time(diff(diff(log(airpass)),lag=12)),
pch=as.vector(season(diff(diff(log(airpass)),lag=12))))
```



The plot represents the seasonal difference of the first difference of the logged airline passenger series, highlighting both seasonal and non-seasonal stationarity. By applying the first difference and seasonal differencing (lag = 12), the trend and seasonal patterns are removed, resulting in an approximately stationary series. The remaining fluctuations primarily reflect random noise and residual variability, suitable for modeling with SARIMA.

(d) Calculate and interpret the sample ACF of the seasonal difference of the first difference of the logged series.

```
acf(as.vector(diff(diff(log(airpass)),lag=12)),ci.type='ma',
main='First & Seasonal Differences of Log(AirPass)')
```



The ACF plot of the seasonal difference of the first difference of the logged series shows that significant autocorrelations exist at seasonal lags (e.g., 12, 24), indicating residual seasonality. The decay of autocorrelations at other lags suggests the presence of non-seasonal patterns. This behavior supports the need for a SARIMA model to account for both seasonal and non-seasonal components in the time series.

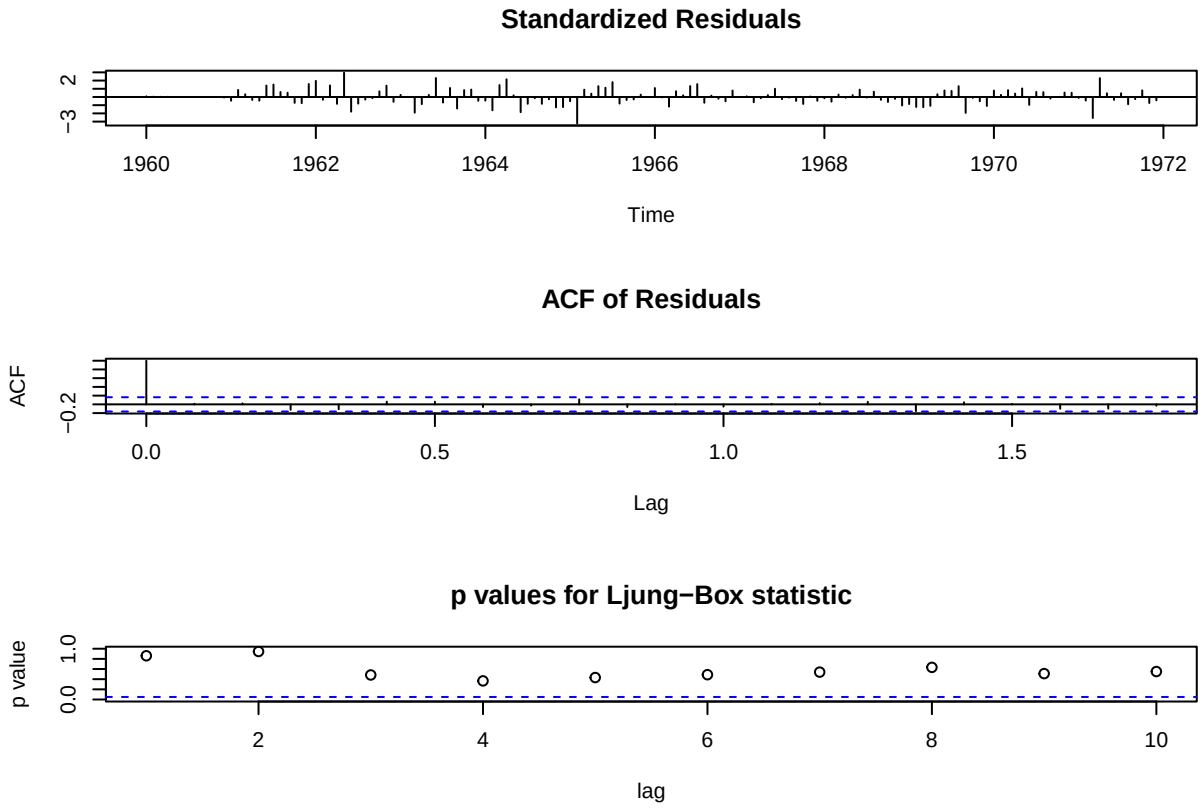
(e) Fit the "airline model" (ARIMA(0,1,1)×(0,1,1)12) to the logged series

```
airline_model=arima(log(airpass),order=c(0,1,1),seasonal=list(order=c(0,1,1),period=12))
airline_model
```

```
##
## Call:
## arima(x = log(airpass), order = c(0, 1, 1), seasonal = list(order = c(0, 1,
##     1), period = 12))
##
## Coefficients:
##           ma1     sma1
##       -0.4018  -0.5569
## s.e.   0.0896   0.0731
##
## sigma^2 estimated as 0.001348:  log likelihood = 244.7,  aic = -485.4
```

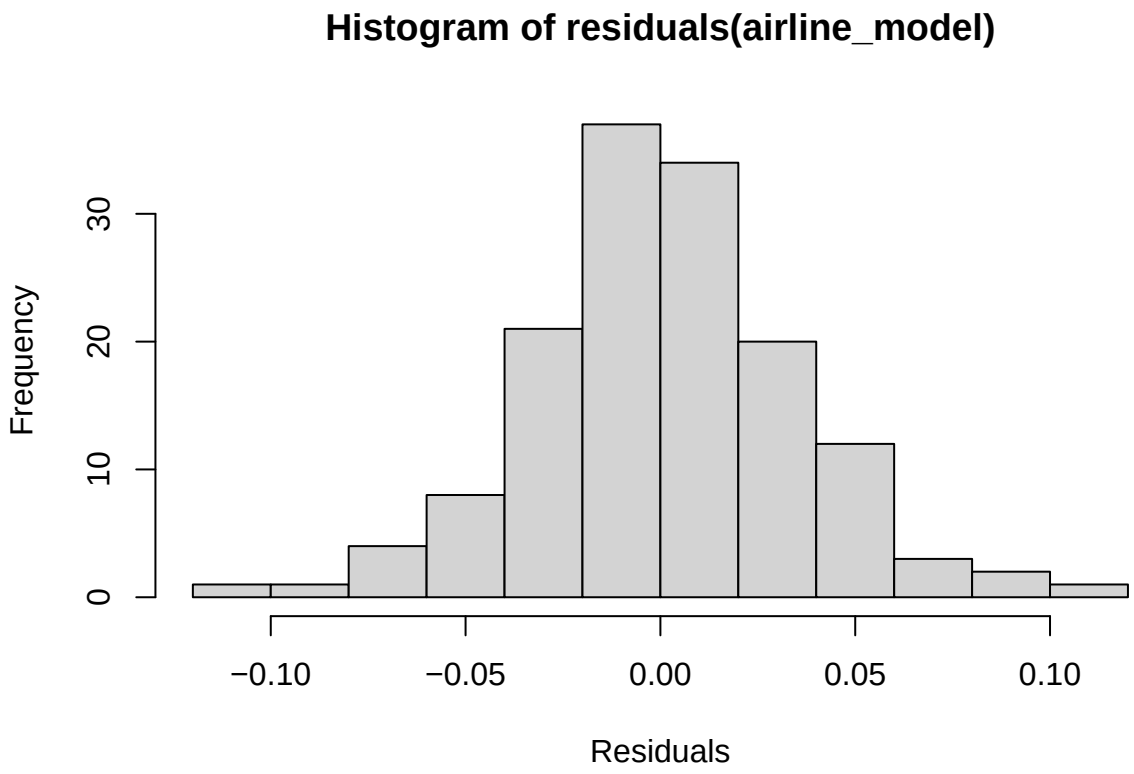
(f) Investigate diagnostics for this model, including autocorrelation and normality of the residuals.

```
tsdiag(airline_model)
```

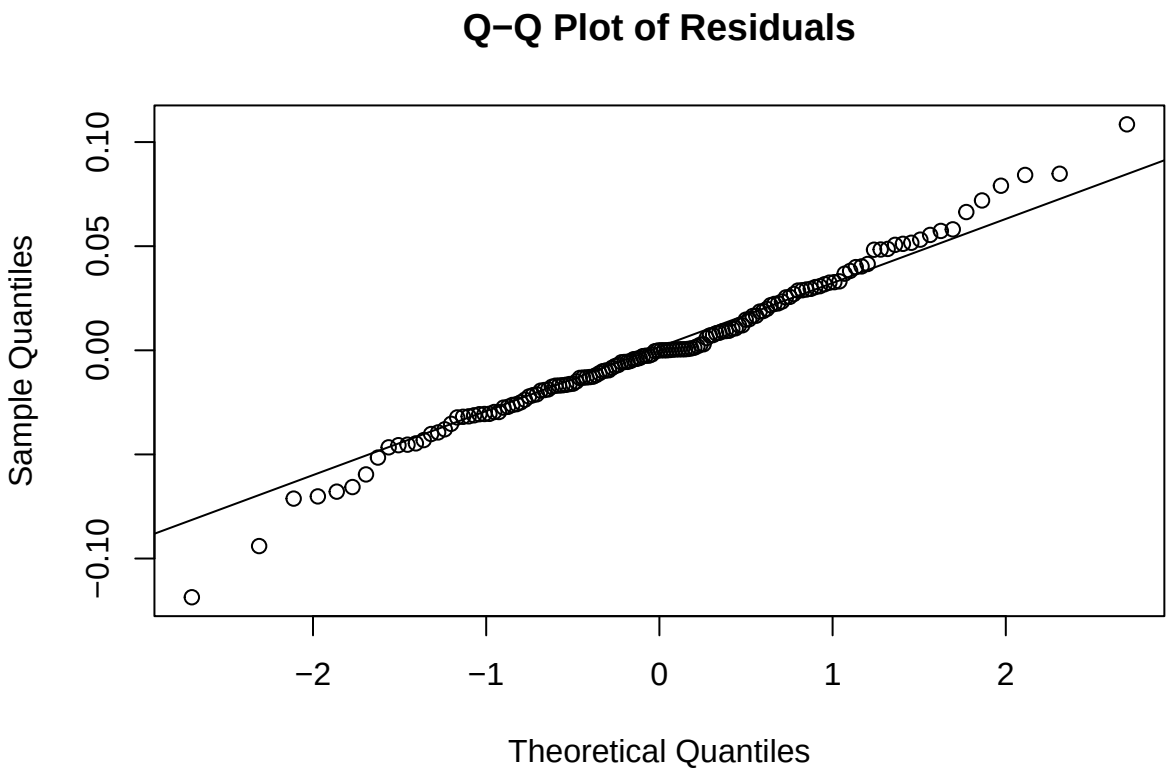


Normality of Residuals:

```
hist(residuals(airline_model),xlab='Residuals')
```




```
qqnorm(residuals(airline_model), main = "Q-Q Plot of Residuals")
qqline(residuals(airline_model))
```



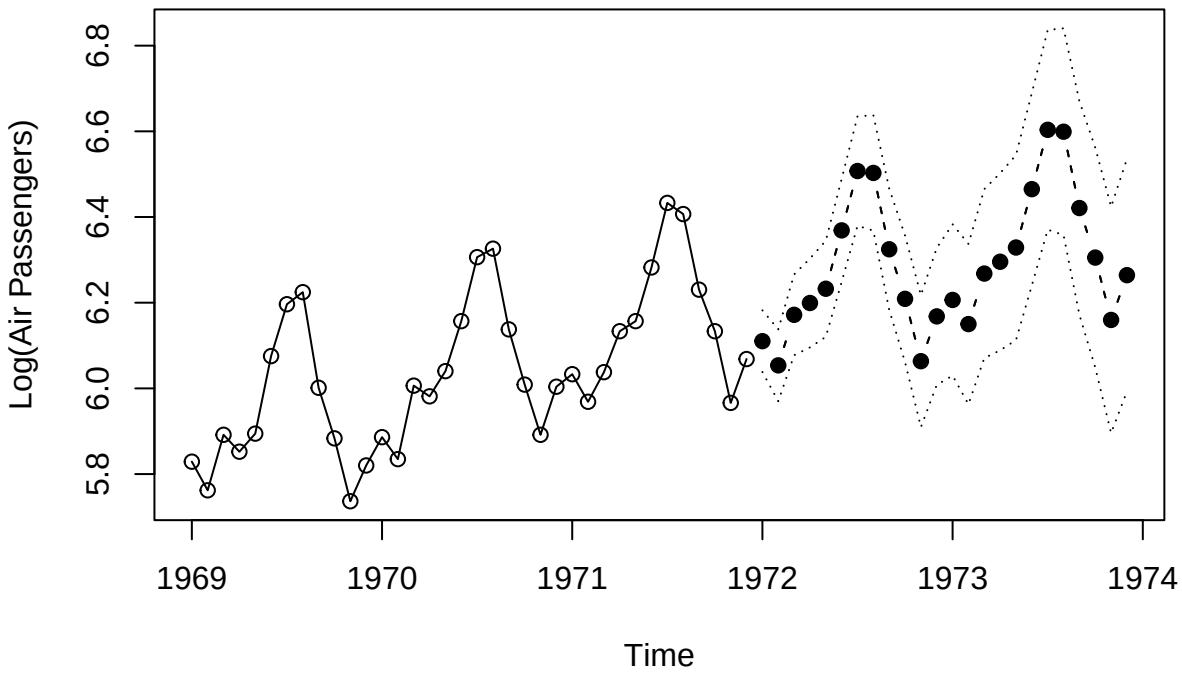

```
shapiro.test(residuals(airline_model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(airline_model)
## W = 0.98637, p-value = 0.1674
```

The Shapiro-Wilk test does not reject normality of the error terms at any of the usual significance levels and we procede to use the model for forecasting.

(g) Produce forecasts for this series with a lead time of two years. Be sure to include forecast limits

```
plot(airline_model,n1=c(1969,1),n.ahead=24,pch=19,ylab='Log(Air Passengers)')
```



The plot shows the model's forecast of log-transformed airline passengers with an upward trend and seasonal fluctuations, matching the historical pattern. The widening confidence intervals indicate increasing uncertainty as the forecast horizon extends.