STAT 505 Project

Daniel Girvitz, Additional authors redacted 2022-06-19

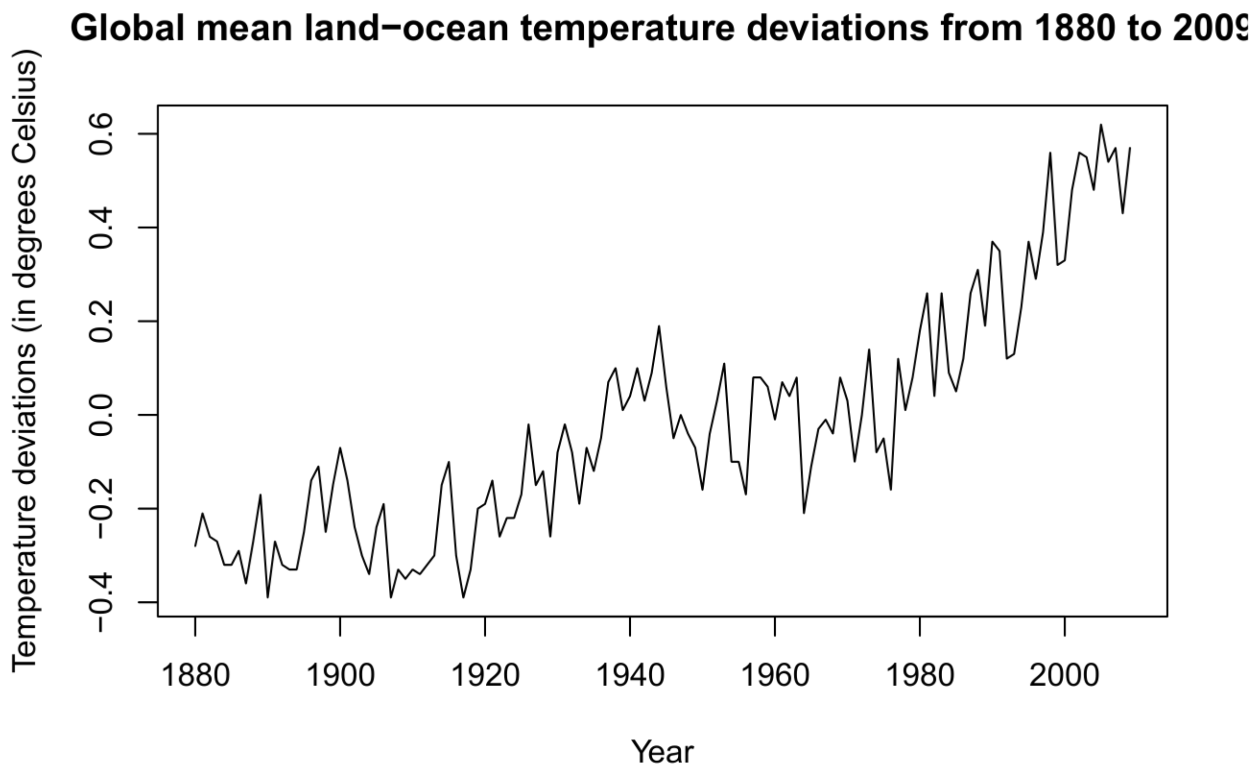
31/03/2022

Problem 1

In this problem, use the Box and Jenkins steps to model yearly data of Global Mean land-ocean temperature deviations (measured in degrees centigrade) from 1880 to 2009 (gtemp). The database is named gtemp.txt.

```r
# Load data
rm(list=ls())
eData <- scan(file="gtemp.txt", what=double())
```

```r
# Plot time series
year = 1880:2009
plot(year, eData,
     main = "Global mean land-ocean temperature deviations from 1880 to 2009",
     xlab = "Year",
     ylab = "Temperature deviations (in degrees Celsius)",
     type = "l")
```



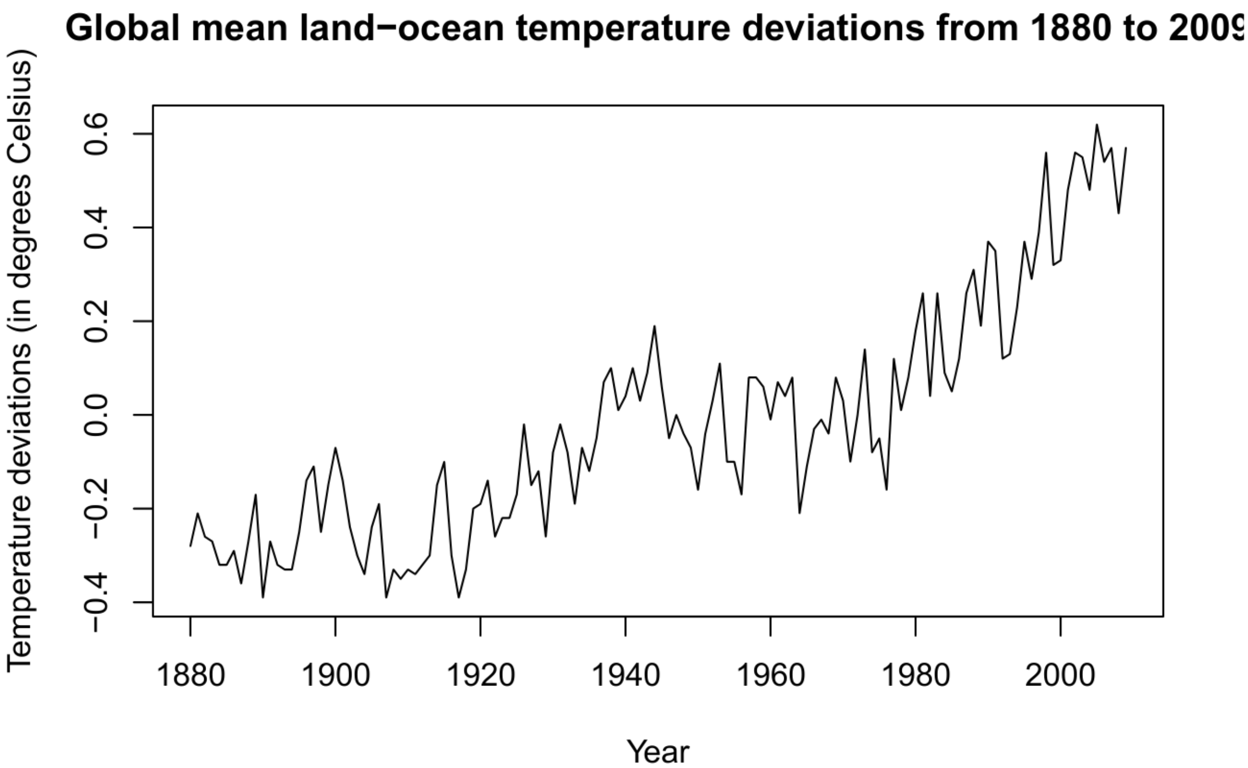

```
temp.ts = ts(data=eData,  start = 1880, end = 2009)
```

Problem 1: Identification

Should the time series be transformed? What is the choice of the degree of differencing? Use ACF and PACF plots, and unit root tests such as the ADF test.

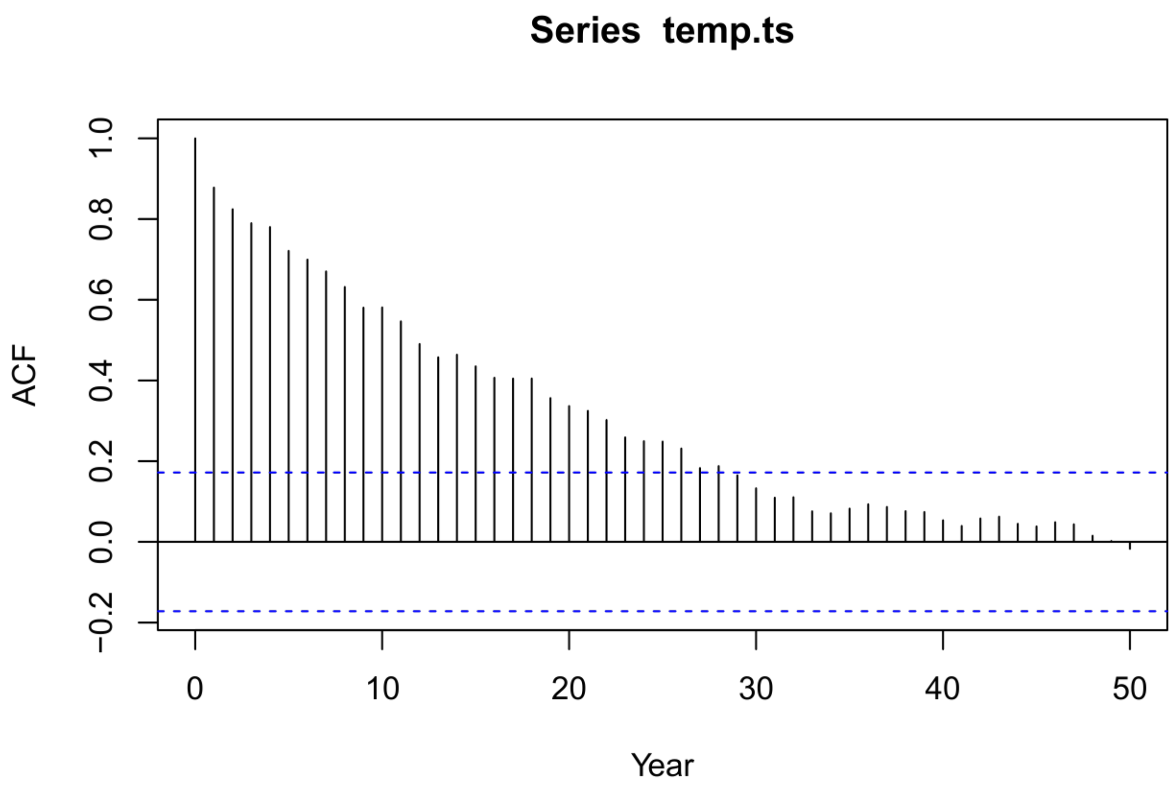
Should the time series be transformed? Yes, To remove the trend. From the ADF test we see lag 1 is significant, so we difference by lag 1.

```
# Plot time series
year = 1880:2009
plot(year, eData,
     main = "Global mean land-ocean temperature deviations from 1880 to 2009",
     xlab = "Year",
     ylab = "Temperature deviations (in degrees Celsius)",
     type = "l")
```

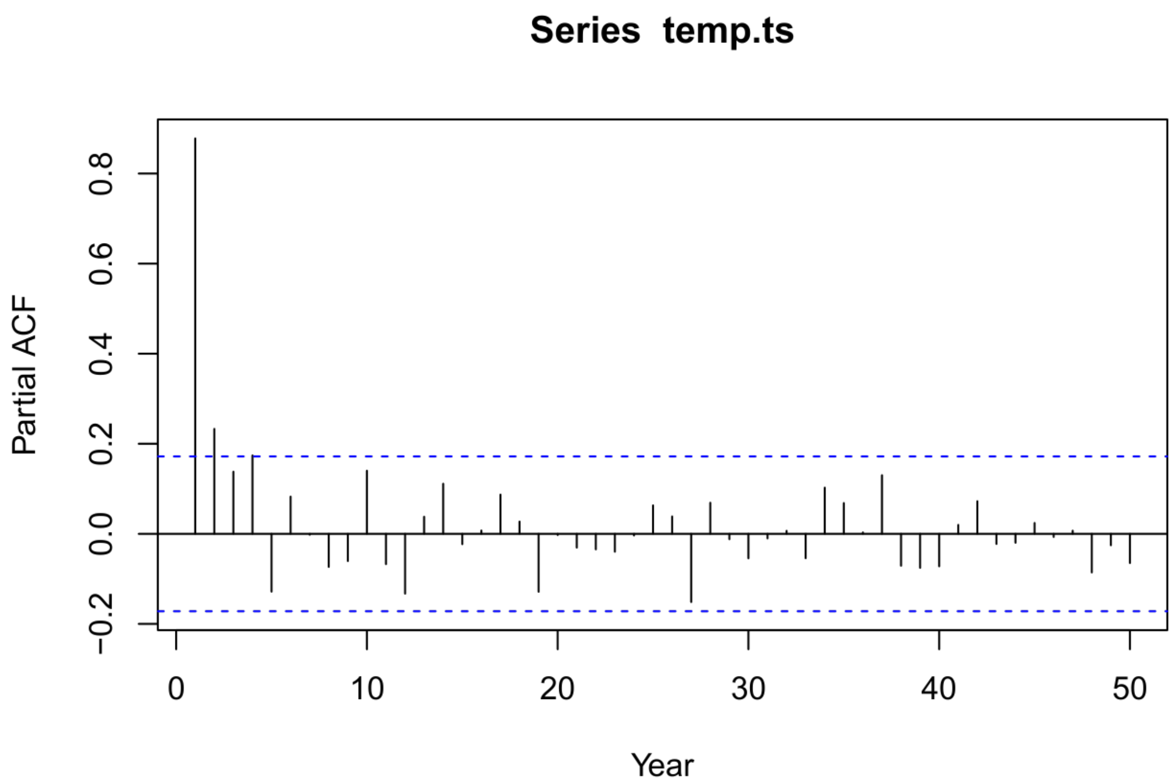



```r
temp.ts = ts(data=eData,  start = 1880, end = 2009)
```

```r
#ACF and PACF plots
acf(temp.ts,xlab="Year", lag.max =50)
```




```
pacf(temp.ts,xlab="Year",lag.max =50)
```

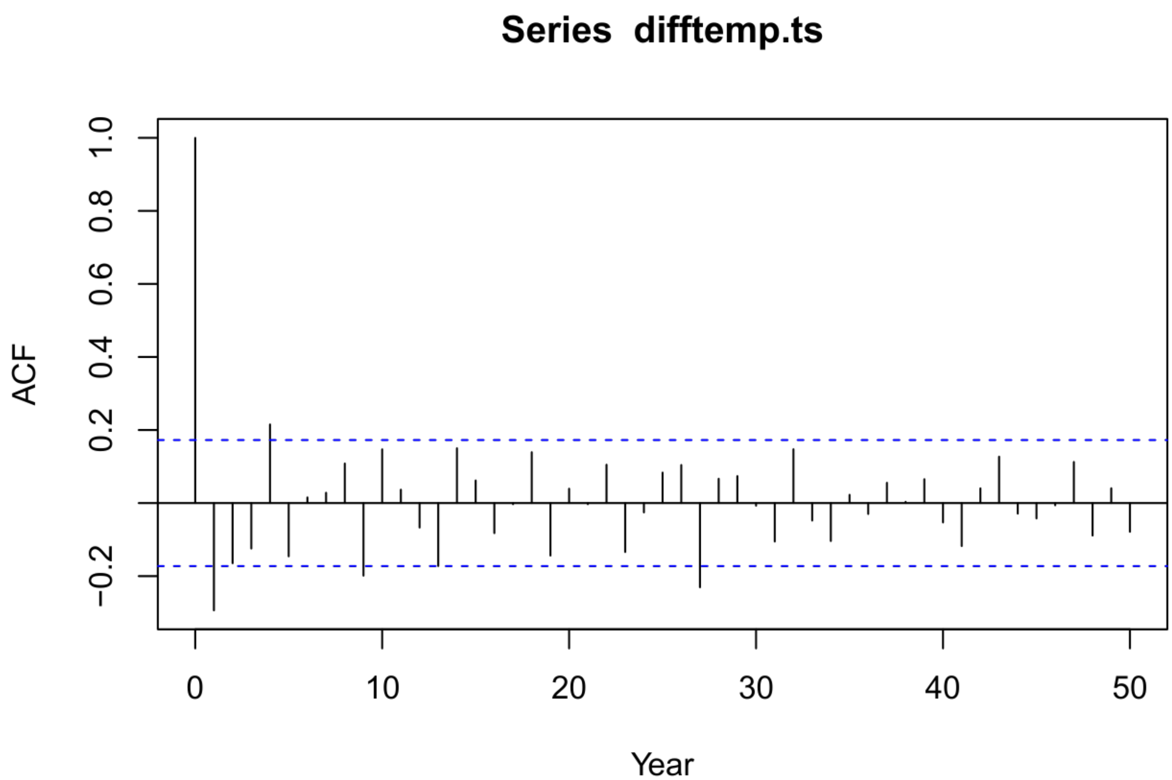



```r
#Augmented Dickey Fuller
aTSA::adf.test(temp.ts)
```

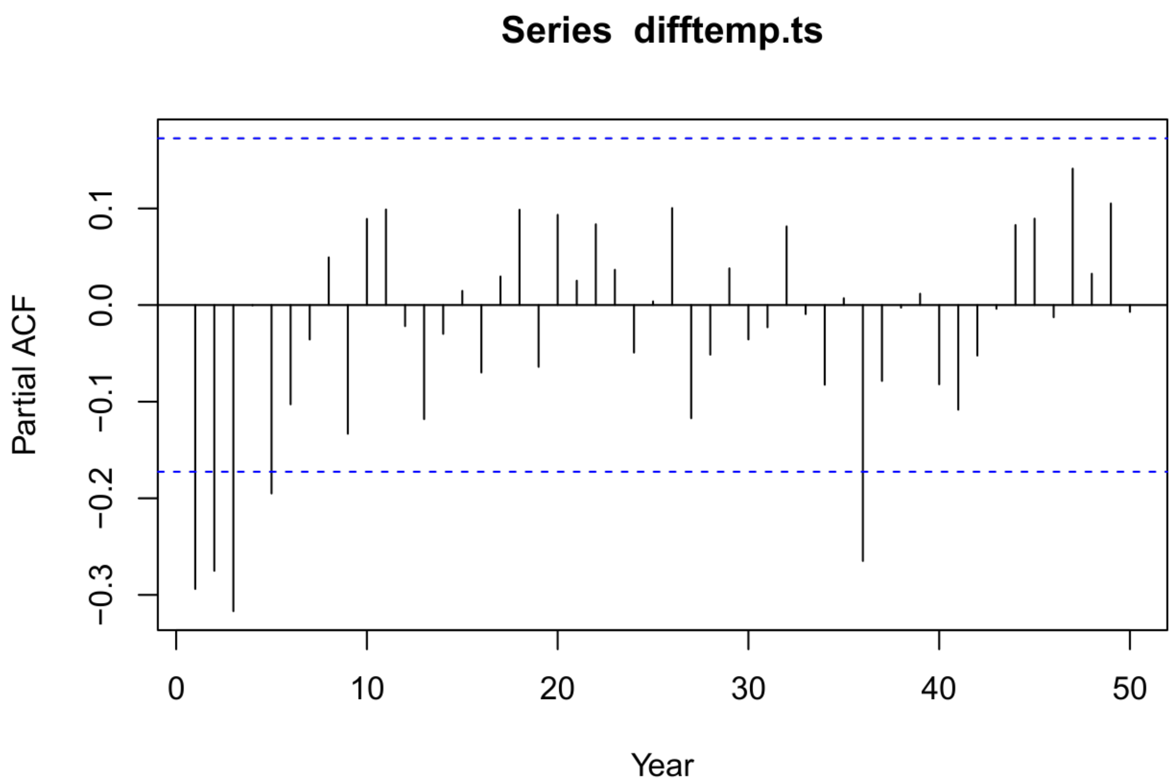
```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -2.156  0.0322
## [2,]   1 -1.378  0.1840
## [3,]   2 -0.795  0.3940
## [4,]   3 -0.166  0.5956
## [5,]   4 -0.280  0.5629
## Type 2: with drift no trend
##      lag     ADF p.value
## [1,]   0 -2.0881   0.295
## [2,]   1 -1.2888   0.595
## [3,]   2 -0.6567   0.817
## [4,]   3  0.0555   0.959
## [5,]   4 -0.0159   0.954
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -5.39  0.0100
## [2,]   1 -4.29  0.0100
## [3,]   2 -3.29  0.0761
## [4,]   3 -2.29  0.4520
```

```
## [5,]    4 -2.29  0.4498
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

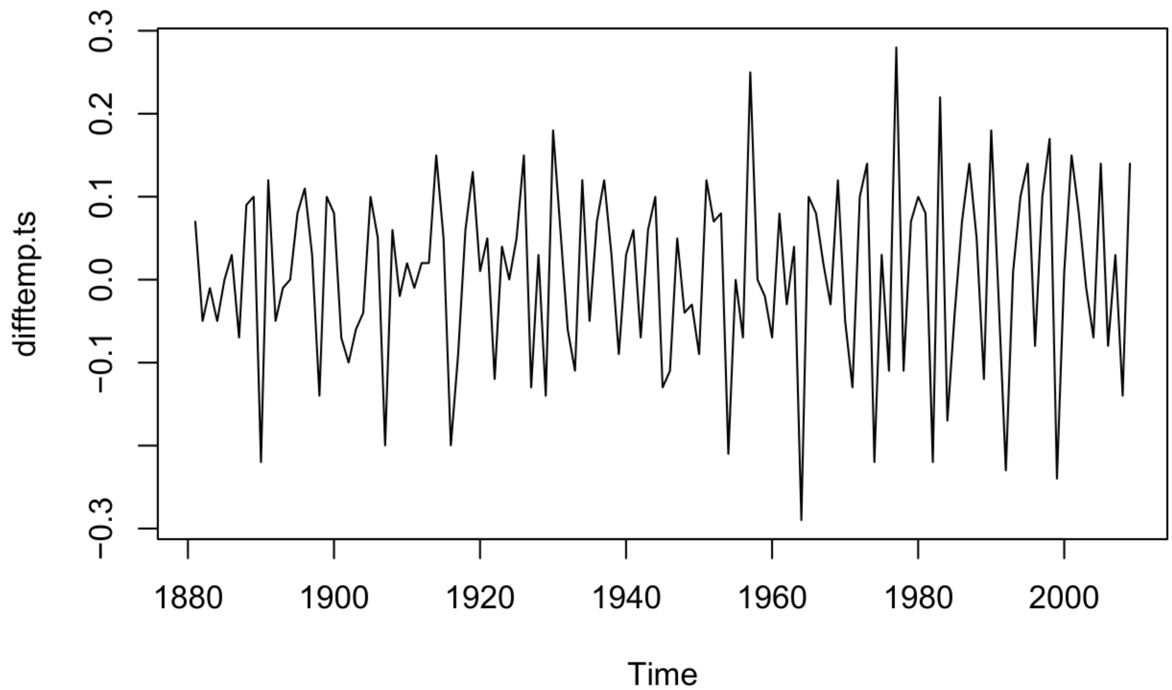
```r
#Difference by 1
difftemp.ts = diff(temp.ts, lag = 1)
#ACF and PACF plots
acf(difftemp.ts,xlab="Year", lag.max =50)
```




```r
pacf(difftemp.ts,xlab="Year", lag.max =50)
```




```
plot.ts(difftemp.ts)
```




```r
aTSA::adf.test(difftemp.ts)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -15.18    0.01
## [2,]   1 -11.80    0.01
## [3,]   2 -11.02    0.01
## [4,]   3  -7.56    0.01
## [5,]   4  -7.39    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -15.18    0.01
## [2,]   1 -11.87    0.01
## [3,]   2 -11.20    0.01
## [4,]   3  -7.77    0.01
## [5,]   4  -7.72    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -15.16    0.01
## [2,]   1 -11.88    0.01
## [3,]   2 -11.27    0.01
## [4,]   3  -7.84    0.01
```

```
## [5,]    4  -7.83    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
AIC1=BIC1=matrix(0,4,4)
for (i in 1:4){ for (j in 1:4){
        fit=arima(difftemp.ts, order = c(i-1, 0,j-1))
        AIC1[i,j]=AIC(fit);
        BIC1[i,j]=BIC(fit);
}}
AIC1
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] -201.0879 -228.0342 -231.4121 -229.4476
## [2,] -210.8328 -230.6761 -229.4253 -232.3445
## [3,] -218.8417 -230.2359 -228.6083 -232.1489
## [4,] -230.2707 -229.2290 -231.8402 -229.9940
```

```
BIC1
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] -195.3683 -219.4548 -219.9729 -215.1485
## [2,] -202.2533 -219.2368 -215.1262 -215.1856
## [3,] -207.4024 -215.9368 -211.4495 -212.1303
## [4,] -215.9716 -212.0702 -211.8216 -207.1155
```

```
idxminA=which(AIC1 == min(AIC1), arr.ind = TRUE)
idxminA
```

```
##      row col
## [1,]   2   4
```

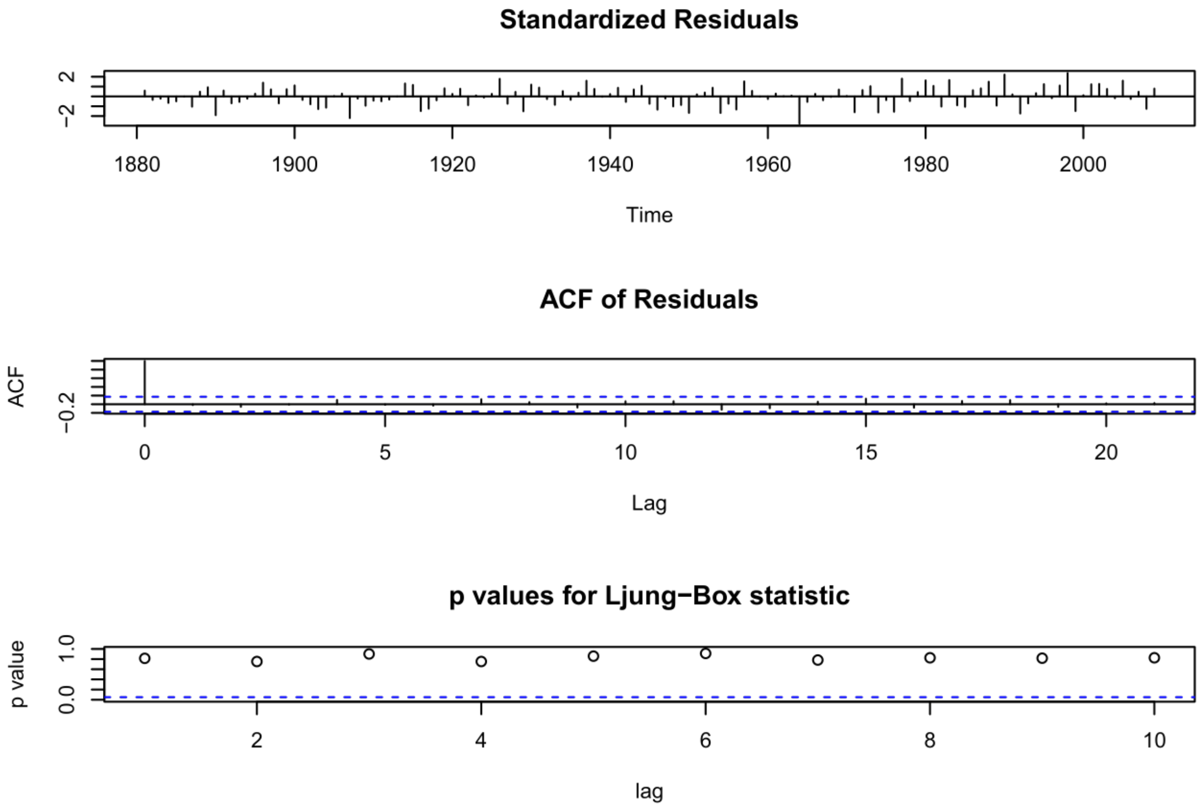
```
idxminB=which(BIC1 == min(BIC1), arr.ind = TRUE)
idxminB
```

```
##      row col
## [1,]   1   3
```

Problem 1: Parameter Estimation

[Use Maximum Likelihood approach] The AIC and BIC minimizing models are tested. We find that only the ARMA(1,3) model achieves significance on all the coefficients.

```
#Test for ARMA(1,3)
fit1 = arima(difftemp.ts, order = c(1,0,3))
tsdiag(fit1)
```




```
library(lmtest)
```

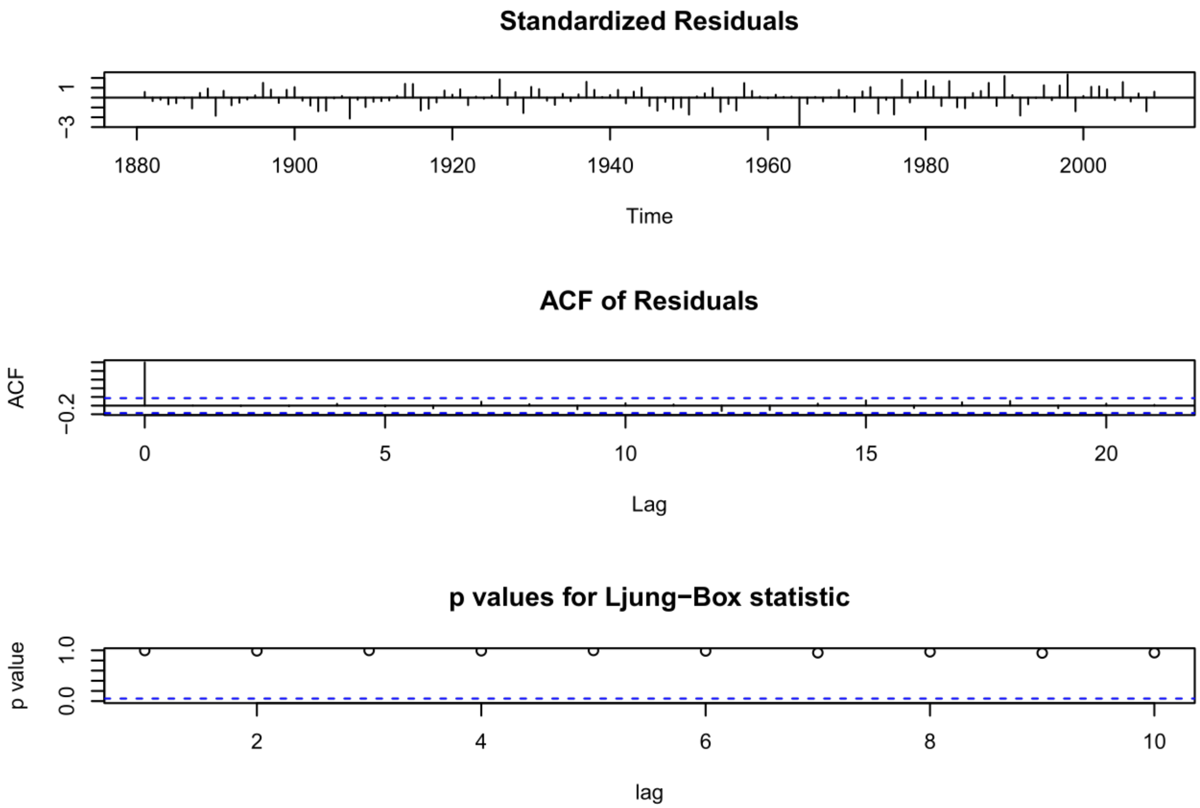
```
## Warning: package 'lmtest' was built under R version 4.1.3

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.1.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
coeftest(fit1)
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value  Pr(>|z|)
## ar1        -0.9376353  0.0946242 -9.9090 < 2.2e-16 ***
## ma1         0.4846151  0.1164615  4.1612 3.166e-05 ***
```

```
## ma2       -0.6341030  0.0955370 -6.6373 3.196e-11 ***
## ma3       -0.2856936  0.0866066 -3.2988 0.0009712 ***
## intercept  0.0064613  0.0024689  2.6171 0.0088681 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Test for ARMA(2,4)
fit2 = arima(difftemp.ts, order = c(2,0,4))
tsdiag(fit2)
```

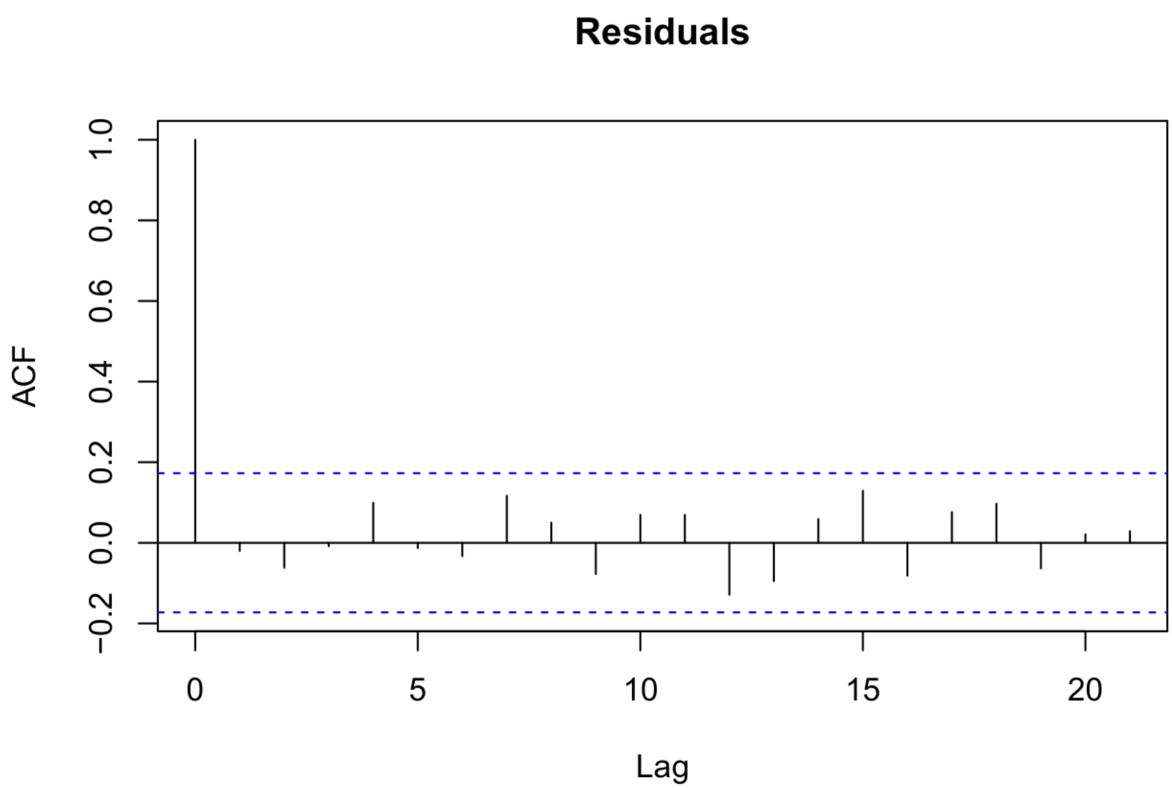



```
coeftest(fit2)
```

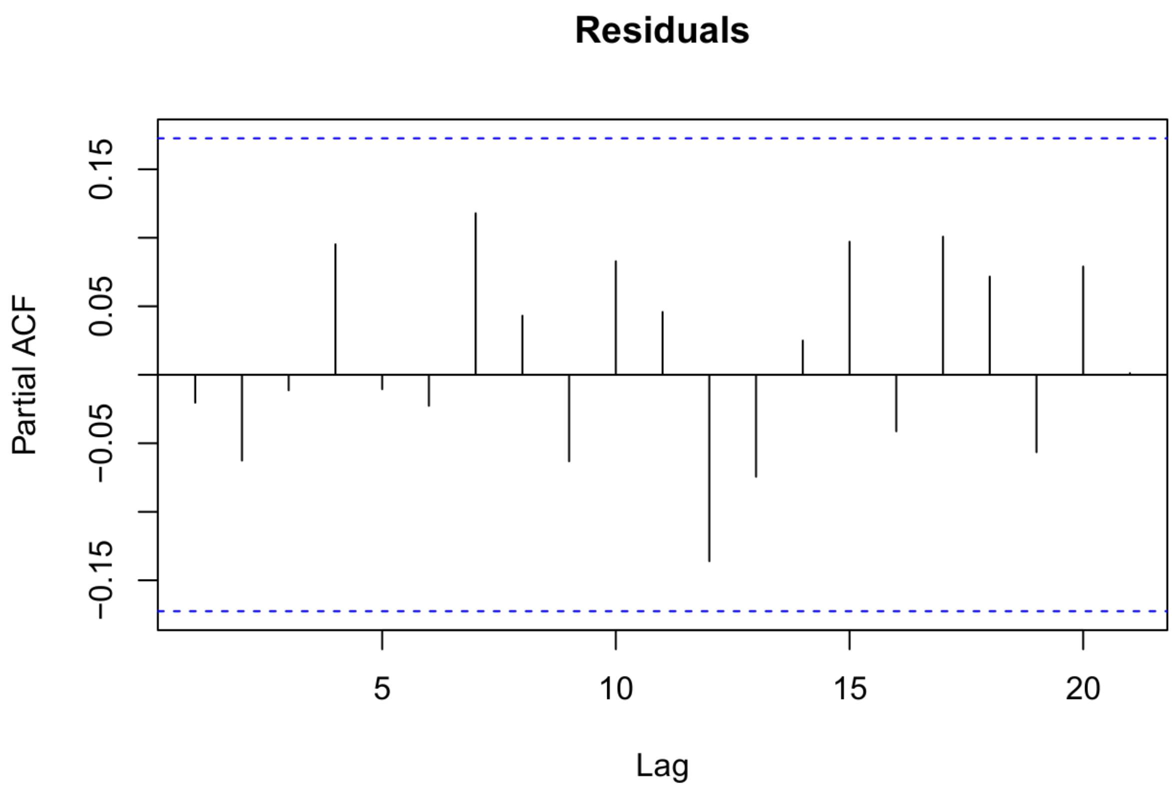
```
##
## z test of coefficients:
##
##             Estimate Std. Error z value Pr(>|z|)
## ar1       -0.4316585  0.5157592 -0.8369 0.402627
## ar2        0.4022230  0.5130463  0.7840 0.433046
## ma1       -0.0488326  0.5051245 -0.0967 0.922985
## ma2       -0.8532493  0.3063676 -2.7851 0.005352 **
## ma3        0.0272467  0.3405038  0.0800 0.936222
## ma4        0.2224738  0.1432431  1.5531 0.120394
## intercept  0.0064689  0.0028137  2.2991 0.021498 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Problem 1: Validation of the model

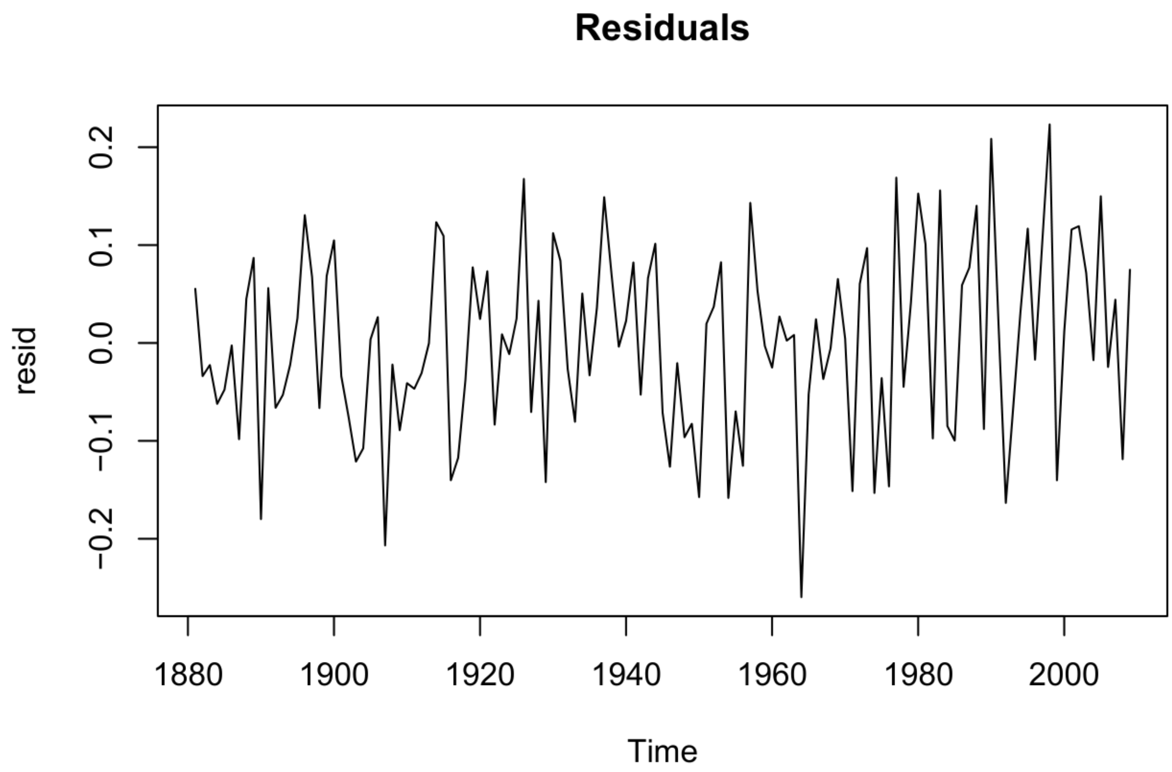
```r
#Analyze Residuals
resid=fit1$residuals
acf(resid,main="Residuals")
```




```r
pacf(resid,main="Residuals")
```




```
plot(resid,main="Residuals") # The residuals look like white noise with 0 mean and constant variance.
```




```
print(AIC(fit1))
```

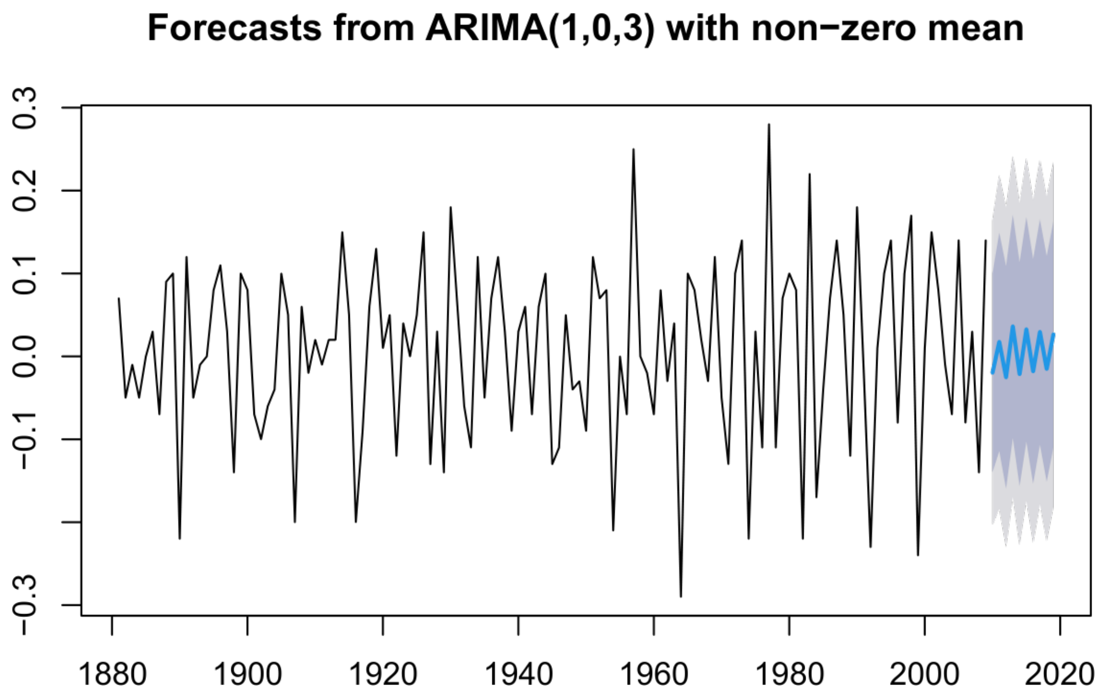
```
## [1] -232.3445
```

```
print(BIC(fit1))
```

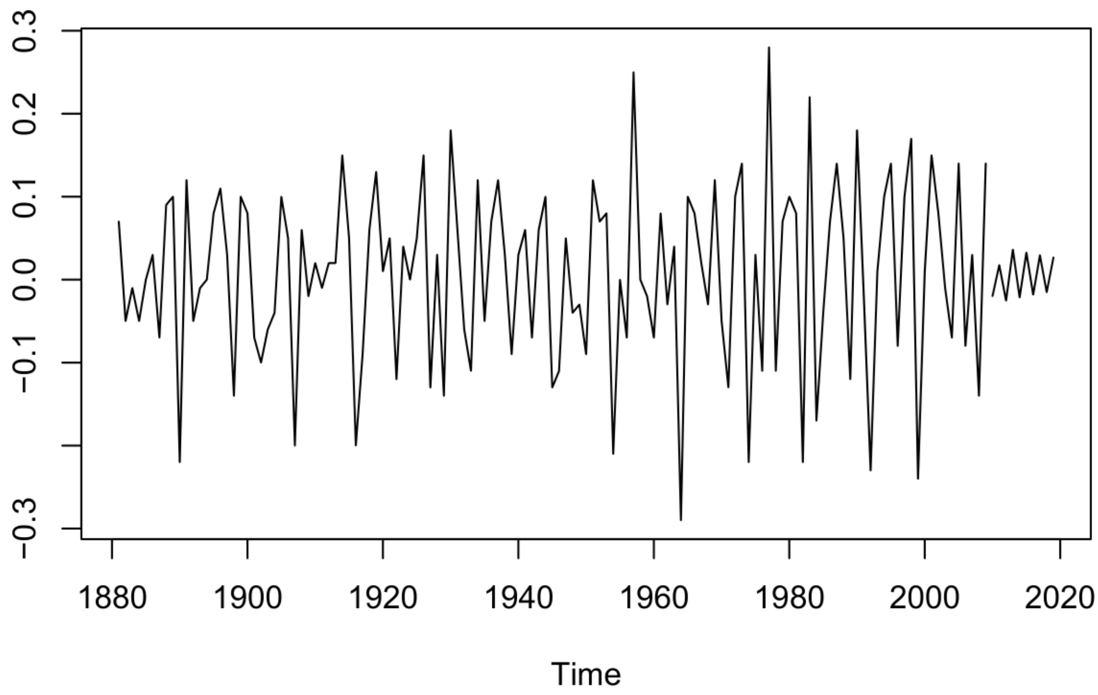
```
## [1] -215.1856
```

Problem 1: Forecast

```
library(forecast)
pm <- forecast(fit1, h = 10)
plot(pm)
```




```
pred = predict(fit1, n.ahead = 10)
ts.plot(difftemp.ts, pred$pred)
```



Problem 2

```r
library(aTSA)
```

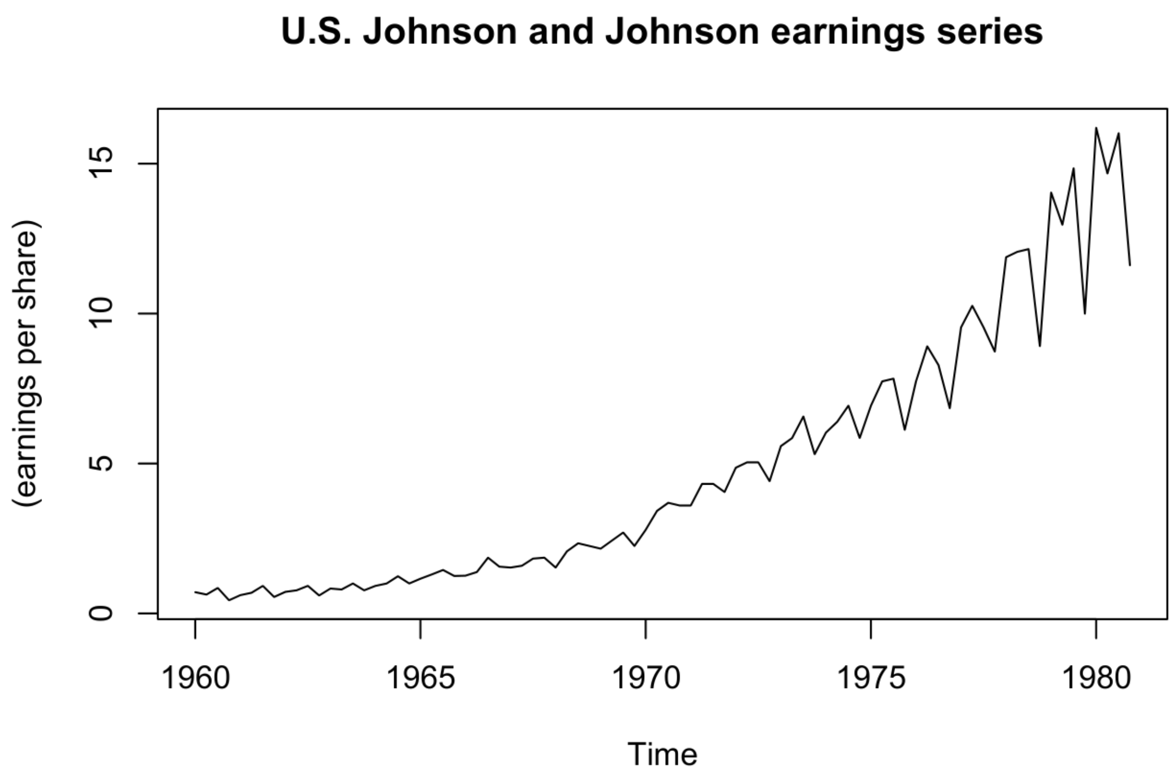
```
##
## Attaching package: 'aTSA'

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

## The following object is masked from 'package:graphics':
##
##     identify
```

```r
library(lmtest)
library(forecast)
library(uroot)
```

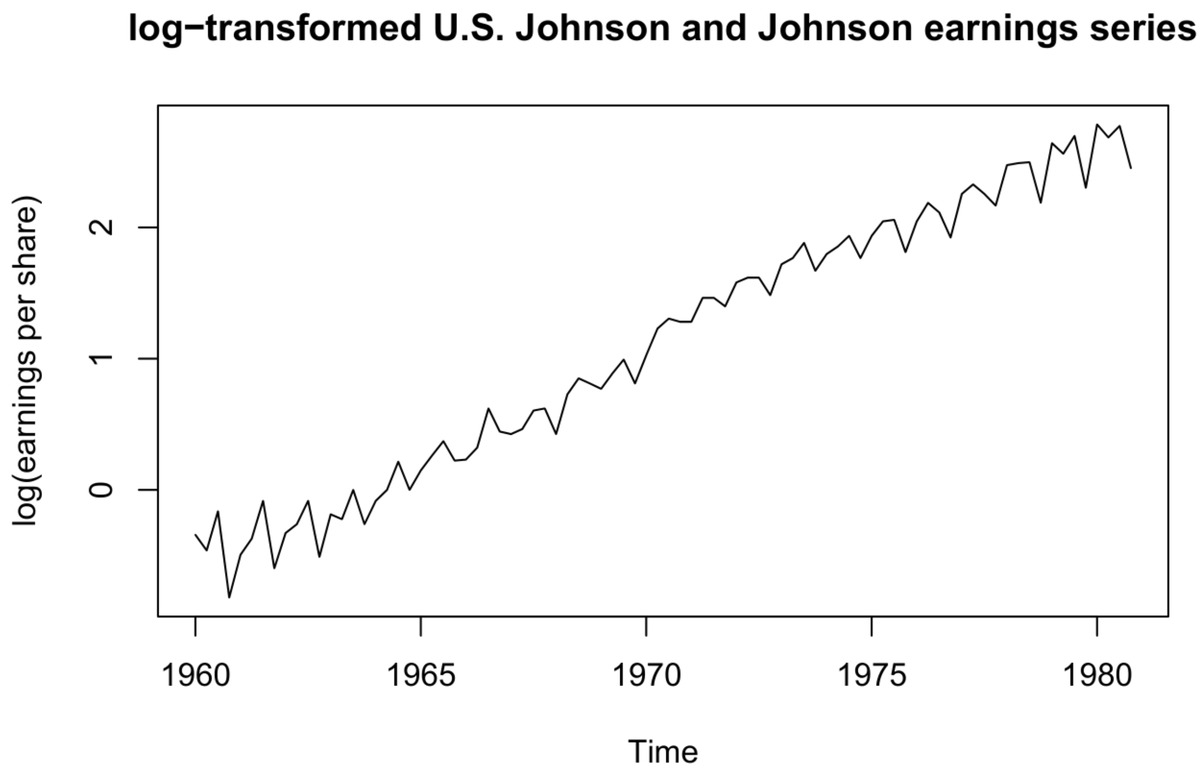
```r
# Load data
rm(list=ls())
nonlogeData <- scan(file="jj.txt", what=double())
```

```r
eData<-log(nonlogeData)
time = (seq(1960,1980.75,by=0.25))
# Note:
# .00 - first quarter
# .25 - second "
# .50 - third "
# .75 - fourth quarter
```

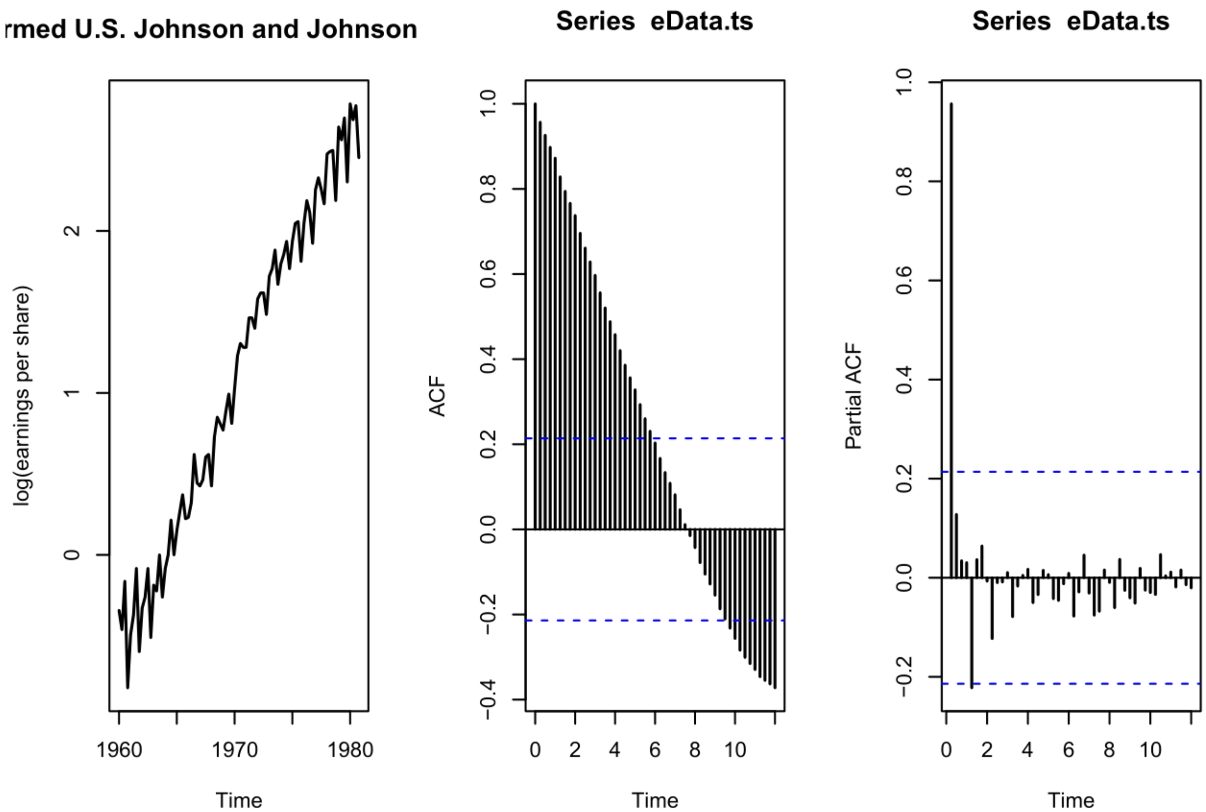
```r
# Plot time series
plot(time, nonlogeData,
     main = "U.S. Johnson and Johnson earnings series",
     xlab = "Time",
     ylab = "(earnings per share)",
     type = "l")
```



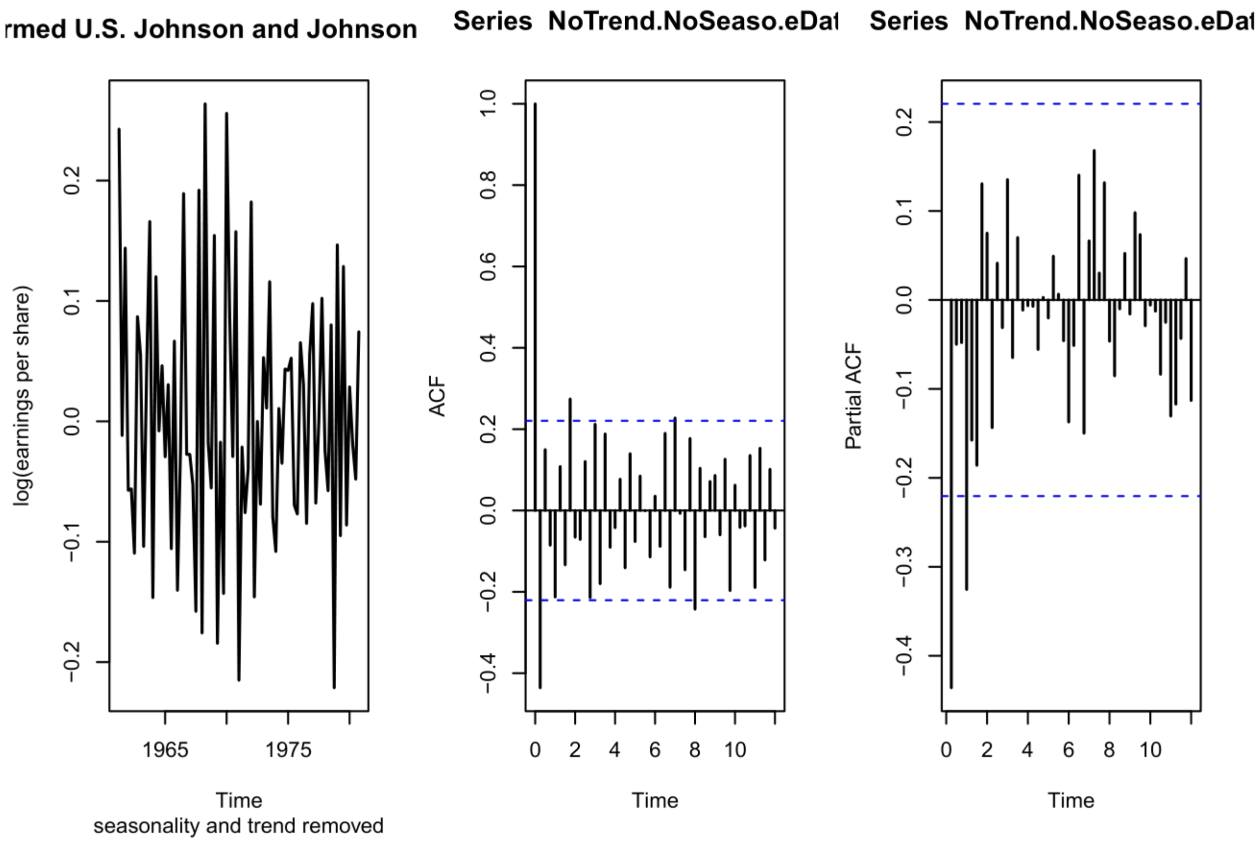

```r
plot(time, eData,
     main = "log-transformed U.S. Johnson and Johnson earnings series",
     xlab = "Time",
     ylab = "log(earnings per share)",
     type = "l")
```




```
#log transfer
par(mfrow=c(1,3))
# Plot time series with ts() function
eData.ts = ts(data=eData, frequency = 4, start = c(1960,1), end = c(1980,4))
plot.ts(eData.ts,
        main = "log-transformed U.S. Johnson and Johnson earnings series",
        xlab = "Time",
        ylab = "log(earnings per share)",
        lwd = 1.5)
# Plot corresponding acf
acf(eData.ts, lag.max=48, xlab="Time", lwd=1.5)
# Plot corresponding pacf
pacf(eData.ts, lag.max=48, xlab="Time", lwd=1.5)
```




```
#observe
par(mfrow=c(1,3))


# Plot time series, seasonality removed, then trend removed, with ts() function
NoSeaso.eData = diff(eData.ts, lag=4) # 1-B^4
NoTrend.NoSeaso.eData = diff(NoSeaso.eData, lag=1) #1-B
plot.ts(NoTrend.NoSeaso.eData,
        main = "log-transformed U.S. Johnson and Johnson earnings series",
        sub = "seasonality and trend removed",
        xlab = "Time",
        ylab = "log(earnings per share)",
        lwd = 1.5)
# Plot corresponding acf
acf(NoTrend.NoSeaso.eData, lag.max=48, xlab="Time", lwd=1.5)
# Plot corresponding pacf
pacf(NoTrend.NoSeaso.eData, lag.max=48, xlab="Time", lwd=1.5)
```




```
#ADF test
adf.test(NoTrend.NoSeaso.eData)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
## [1,]    0 -14.57    0.01
## [2,]    1  -8.11    0.01
## [3,]    2  -6.70    0.01
## [4,]    3  -7.59    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
## [1,]    0 -14.47    0.01
## [2,]    1  -8.05    0.01
## [3,]    2  -6.65    0.01
## [4,]    3  -7.53    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
## [1,]    0 -14.40    0.01
## [2,]    1  -8.01    0.01
## [3,]    2  -6.59    0.01
## [4,]    3  -7.50    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```r
#trial and error
#All Sarima model
#model 1 p = q= P = Q = 1
sarfit1 = arima(NoTrend.NoSeaso.eData, order = c(1,0,1), seasonal = list(order = c(1,0,1)))
coeftest(sarfit1)
```

```
##
## z test of coefficients:
##
##              Estimate  Std. Error  z value  Pr(>|z|)
## ar1        0.17429464  0.16473226   1.0580    0.2900
## ma1       -0.84075361  0.10982979  -7.6551 1.932e-14 ***
## sar1       0.81721593  0.11644742   7.0179 2.252e-12 ***
## sma1      -0.99999271  0.09987851 -10.0121 < 2.2e-16 ***
## intercept  0.00057063  0.00120763   0.4725    0.6366
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#model 2 p = 2, P = q = Q = 1
sarfit2 = arima(NoTrend.NoSeaso.eData, order = c(2,0,1), seasonal = list(order = c(1,0,1)))
coeftest(sarfit2)
```

```
##
## z test of coefficients:
##
##              Estimate  Std. Error z value  Pr(>|z|)
## ar1        0.18811464  0.20539448  0.9159    0.3597
## ar2        0.22669538  0.17035830  1.3307    0.1833
## ma1       -0.87291971  0.16668990 -5.2368 1.634e-07 ***
## sar1      -0.18445186  0.34061696 -0.5415    0.5881
## sma1      -0.10854701  0.34460064 -0.3150    0.7528
## intercept  0.00079263  0.00187944  0.4217    0.6732
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#model 3 p = 0, q = P = Q = 1
sarfit3 = arima(NoTrend.NoSeaso.eData, order = c(0,0,1), seasonal = list(order = c(1,0,1)))
coeftest(sarfit3)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## ma1       -0.6761976  0.0968571 -6.9814 2.923e-12 ***
## sar1      -0.2000908  0.3001443 -0.6666    0.5050
## sma1      -0.1398545  0.2982684 -0.4689    0.6391
## intercept  0.0010614  0.0024370  0.4355    0.6632
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#model 4 p = P = q = 1, Q = 0
sarfit4 = arima(NoTrend.NoSeaso.eData, order = c(1,0,1), seasonal = list(order = c(1,0,0)))
coeftest(sarfit4)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value Pr(>|z|)
## ar1       -0.0117778  0.2224559 -0.0529 0.957776
## ma1       -0.6729858  0.1813236 -3.7115 0.000206 ***
## sar1      -0.3261073  0.1322660 -2.4655 0.013681 *
## intercept  0.0010963  0.0025525  0.4295 0.667540
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#model 5 p = Q = 0, P = q = 1
sarfit5 = arima(NoTrend.NoSeaso.eData, order = c(0,0,1), seasonal = list(order = c(1,0,0)))
coeftest(sarfit5)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## ma1       -0.6810164  0.0968857 -7.0291 2.079e-12 ***
## sar1      -0.3223730  0.1123146 -2.8703  0.004101 **
## intercept  0.0010894  0.0025251  0.4314  0.666161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# both Significant

#model 6 p = q = 0, P = Q =1
sarfit6 = arima(NoTrend.NoSeaso.eData, order = c(0,0,0), seasonal = list(order = c(1,0,1)))
coeftest(sarfit6)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value Pr(>|z|)
## sar1      -0.0407396  0.3170231 -0.1285   0.8977
## sma1      -0.2022336  0.3007672 -0.6724   0.5013
## intercept  0.0028567  0.0092753  0.3080   0.7581
```

```r
#model 7 p = q = 1, P = Q = 0
sarfit7 = arima(NoTrend.NoSeaso.eData, order = c(1,0,1), seasonal = list(order = c(0,0,0)))
coeftest(sarfit7)
```

```
##
## z test of coefficients:
##
##               Estimate  Std. Error z value Pr(>|z|)
```

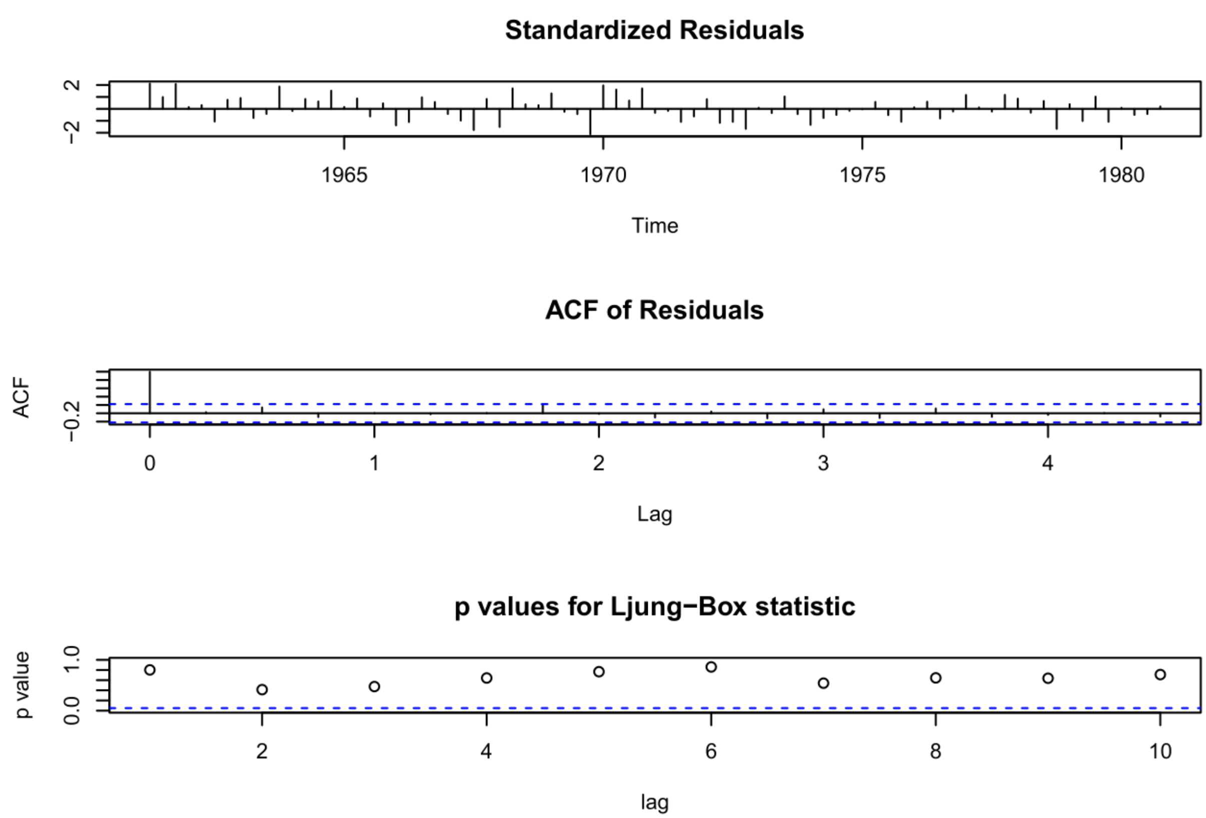
```
## ar1         0.23693974  0.15031508  1.5763   0.1150
## ma1        -0.88789693  0.09325722 -9.5209   <2e-16 ***
## intercept   0.00071103  0.00175999  0.4040   0.6862
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#model 8 p = Q = 1, q = P = 0
sarfit8 = arima(NoTrend.NoSeaso.eData, order = c(1,0,0), seasonal = list(order = c(0,0,1)))
coeftest(sarfit8)
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value  Pr(>|z|)
## ar1        -0.5079714  0.1008151 -5.0386 4.688e-07 ***
## sma1       -0.3244254  0.1044776 -3.1052  0.001901 **
## intercept   0.0017536  0.0047829  0.3666  0.713890
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#both significant
#model 5 is the best fit by comparing autos
```

```r
#Residuals of the process
tsdiag(sarfit5)
```



Our apologies for the use of .png image files: R Markdown failed to run the last two chunks, but the code runs in the console... Must be a bug?

```
# #forecast
# plot(forecast(arima(eData.ts, order=c(0,0,1),
#                         seasonal = list(order=c(1,0,0),period=12)),h=12),
#       xlab="Year", ylab="log(earning per share)",
#       main="Forecast ofU.S. Johnson and Johnson earnings series.")
```

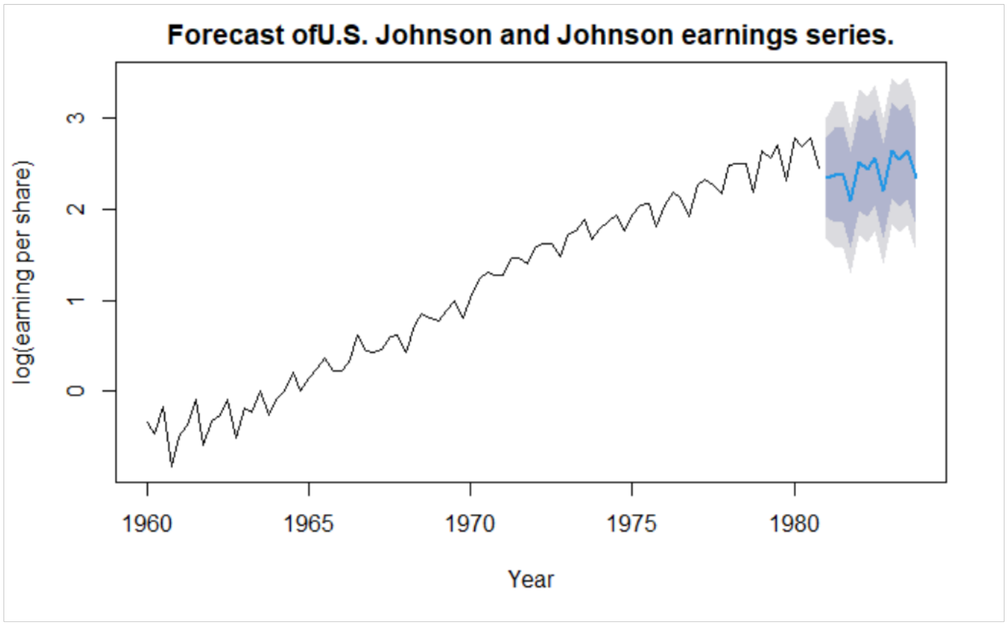


Figure 1: Forecast of U.S. Johnson and Johnson earnings series

```
# #closer look for the forecast
# plot(forecast(arima(eData.ts, order=c(0,0,1),
#                         seasonal = list(order=c(1,0,0),period=12)),h=12),
#       xlab="Year", ylab="log(earning per share)",
#       main="Forecast of U.S. Johnson and Johnson earnings series.",
#       xlim=c(1980,1985))
```

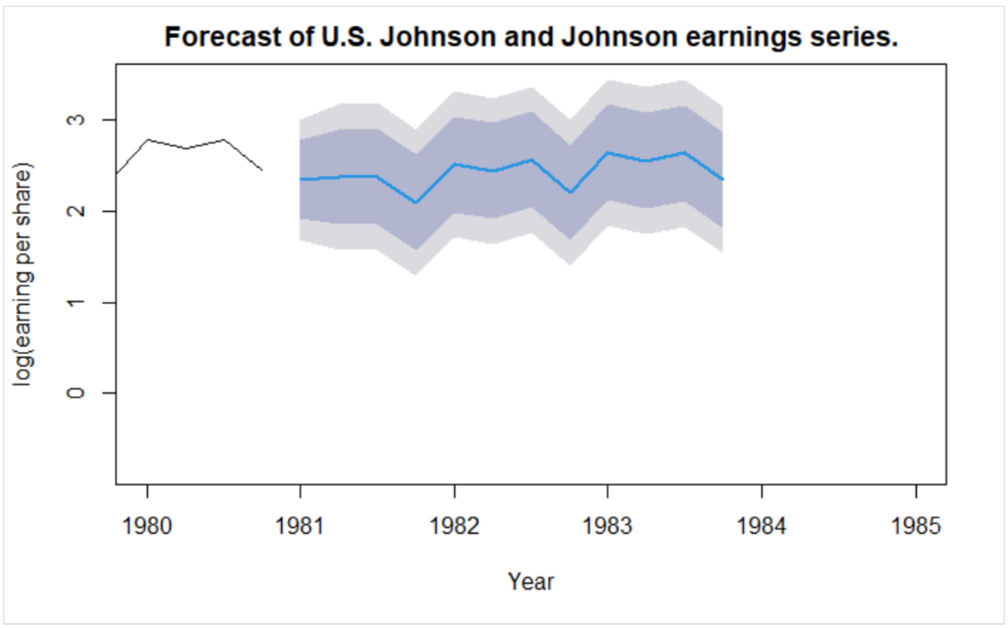


Figure 2: Forecast of U.S. Johnson and Johnson earnings series