# Isaac Sheets – BAS475 – Assignment4

Isaac Sheets

April 24, 2020

1. (similar) Problem 3.10 in text book Let xt represent the cardiovascular mortality series “cmort”.

```
library(astsa)
data("cmort")
```

(a) Fit an AR(2) to xt and report the fitted model and the estimate of the variance of the white noise (σb2w).

```
M <- arima(cmort, order=c(2,0,0))
M

##
## Call:
## arima(x = cmort, order = c(2, 0, 0))
##
## Coefficients:
##          ar1     ar2  intercept
##       0.4301  0.4424    88.8538
## s.e.  0.0397  0.0398     1.9407
##
## sigma^2 estimated as 32.37:  log likelihood = -1604.71,  aic = 3217.43
```

**Response: X(t)-88.8538 = 0.4301*(X(t-1)-88.8538) + 0.4424*(X(t-1)-88.8538) + Wt; sigma^2 estimated as 32.37**

(b) Assuming the fitted model in (a) is the true model, find the forecasts over a four week horizon, xnn+m, for m = 1, 2, 3, 4, and the corresponding 95% prediction intervals.

```
length(cmort)

## [1] 508

cmort[508] #85.49

## [1] 85.49

cmort[507] #89.43

## [1] 89.43

0.4301*(85.49-88.8538) + 0.4424*(89.43-88.8538) + 88.8538          #1-step ahe
ad prediction

## [1] 87.66194
```

```r
0.4301*(87.66194-88.8538) + 0.4424*(85.49-88.8538) + 88.8538      #2-steps ah
ead prediction
```

```
## [1] 86.85304
```

```r
0.4301*(86.85304-88.8538) + 0.4424*(87.66194-88.8538) + 88.8538   #3-steps ah
ead prediction
```

```
## [1] 87.46599
```

```r
0.4301*(87.46599-88.8538) + 0.4424*(86.85304-88.8538) + 88.8538   #4-steps ah
ead prediction
```

```
## [1] 87.37177
```

```r
fore <- predict(M, n.ahead=4) #predict
fore
```

```
## $pred
## Time Series:
## Start = c(1979, 41)
## End = c(1979, 44)
## Frequency = 52
## [1] 87.66207 86.85311 87.46615 87.37190
##
## $se
## Time Series:
## Start = c(1979, 41)
## End = c(1979, 44)
## Frequency = 52
## [1] 5.689543 6.193387 7.148343 7.612531
```

```r
#1-step 95%
c(fore$pred[1]-2*fore$se[1], fore$pred[1]+2*fore$se[1])
```

```
## [1] 76.28299 99.04116
```

```r
#2-steps 95%
c(fore$pred[2]-2*fore$se[2], fore$pred[2]+2*fore$se[2])
```

```
## [1] 74.46633 99.23988
```

```r
#3-steps 95%
c(fore$pred[3]-2*fore$se[3], fore$pred[3]+2*fore$se[3])
```

```
## [1]  73.16946 101.76283
```

```r
#4-steps 95%
c(fore$pred[4]-2*fore$se[4], fore$pred[4]+2*fore$se[4])
```

```
## [1]  72.14684 102.59696
```

**Response: 1-step ahead prediction: 87.66194; Corresponding confidence interval: 76.28299-99.04116. 2-steps ahead prediction: 86.85304; Corresponding confidence interval: 74.46633-99.23988. 3-steps ahead prediction: 87.46599; Corresponding confidence interval: 73.16946-101.76283. 4-steps ahead prediction: 87.37177; Corresponding confidence interval: 72.14684-102.59696.**

2. [(similar) Problem 3.33 in text book] (R) Let's consider fitting an ARIMA(p, d, q) model and generating predictions for the annual global temperature data "gtemp".

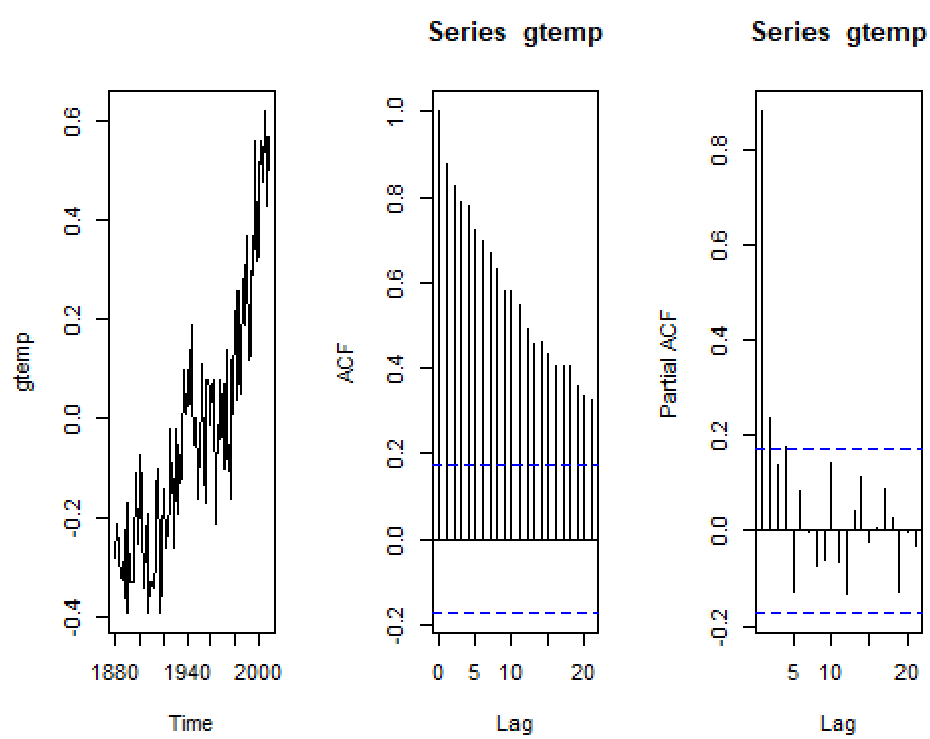
```
data("gtemp")
```

(a) Divide the data set into training and testing data, where the latter data set contains the last 13 data points, i.e., roughly the last 10% of data points, to check the performance of the predictions.

```
length(gtemp)

## [1] 130

TESTING <- gtemp[118:130]
TRAINING <- gtemp[1:117]
```

(b) Check the stationarity of (whole) "gtemp" by looking at the time series plot, ACF and PACF plots.

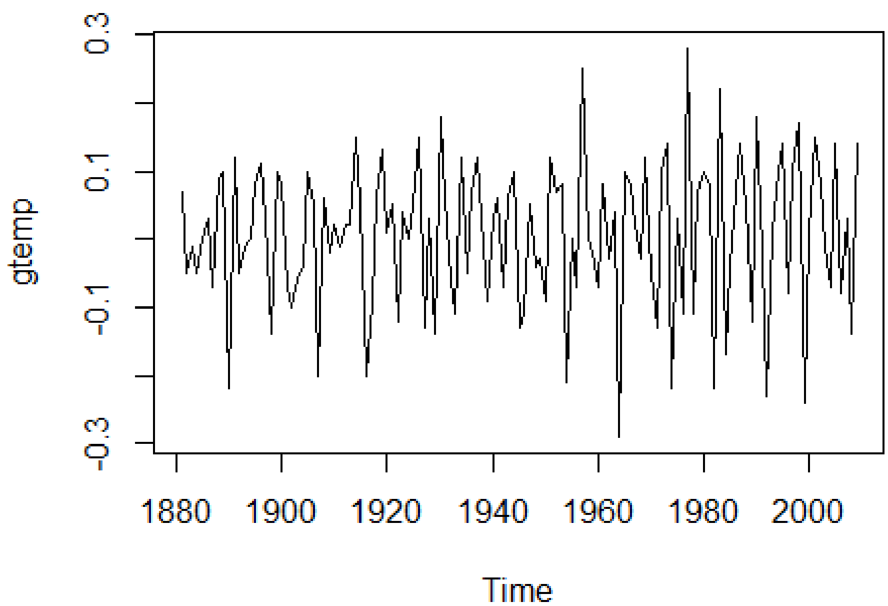
```
par(mfrow = c(1,3))
ts.plot(gtemp)
acf(gtemp)
pacf(gtemp)
```



**Response: The time series is not stationary as the mean changes over time.**

(c) Apply the appropriate order of the difference to “gtemp” and make it stationary.

```
gtemp <- diff(gtemp)
plot.ts(gtemp)
```




**Response: Since the time series plot was linear, only a single difference needs to be applied here.**

(d) Search the optimal ARMA(p, q) model for the differenced training series chosen by AIC/BIC. Try all ARMA(p, q), p, q = 0, 1, · · · , 5 and report the selected optimal model(s).

```
n<-length(diff(TRAINING))
P=5
Q=5
crit<-matrix(0,P+1,Q+1)
for (j in 0:P)
{
for (k in 0:Q)
{
dataML<-arima(diff(TRAINING),order=c(j,0,k),method="ML")

#BIC
crit[j+1,k+1]<-n*log(dataML$sigma2)+(j+k+1)*log(n)
```

```
}
}

## Warning in arima(diff(TRAINING), order = c(j, 0, k), method = "ML"): possi
ble
## convergence problem: optim gave code = 1

crit

##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] -512.1347 -533.5280 -535.3289 -531.1026 -529.7051 -525.7690
## [2,] -517.2771 -534.9841 -530.7836 -532.2049 -527.7289 -523.6427
## [3,] -520.5419 -531.3013 -526.8192 -525.2847 -523.1166 -518.9282
## [4,] -530.3512 -527.7152 -528.1880 -523.6153 -519.1286 -519.3701
## [5,] -525.7062 -530.1493 -527.6085 -524.0235 -519.2679 -514.4944
## [6,] -526.1900 -522.0180 -523.6455 -519.2660 -514.2593 -514.2610

min(crit)

## [1] -535.3289

n<-length(diff(TRAINING))
P=5
Q=5
crit<-matrix(0,P+1,Q+1)
for (j in 0:P)
{
for (k in 0:Q)
{
dataML<-arima(diff(TRAINING),order=c(j,0,k),method="ML")

#AIC
crit[j+1,k+1]<-n*log(dataML$sigma2)+2*(j+k+1)

}
}

## Warning in arima(diff(TRAINING), order = c(j, 0, k), method = "ML"): possi
ble
## convergence problem: optim gave code = 1

crit

##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] -514.8883 -539.0351 -543.5896 -542.1170 -543.4731 -542.2905
## [2,] -522.7843 -543.2449 -541.7979 -545.9729 -544.2505 -542.9178
## [3,] -528.8027 -542.3156 -540.5871 -541.8062 -542.3917 -540.9569
## [4,] -541.3655 -541.4831 -544.7095 -542.8904 -541.1573 -544.1524
## [5,] -539.4742 -546.6708 -546.8836 -546.0522 -544.0502 -542.0303
## [6,] -542.7115 -541.2932 -545.6742 -544.0483 -541.7952 -544.5505

min(crit)
```

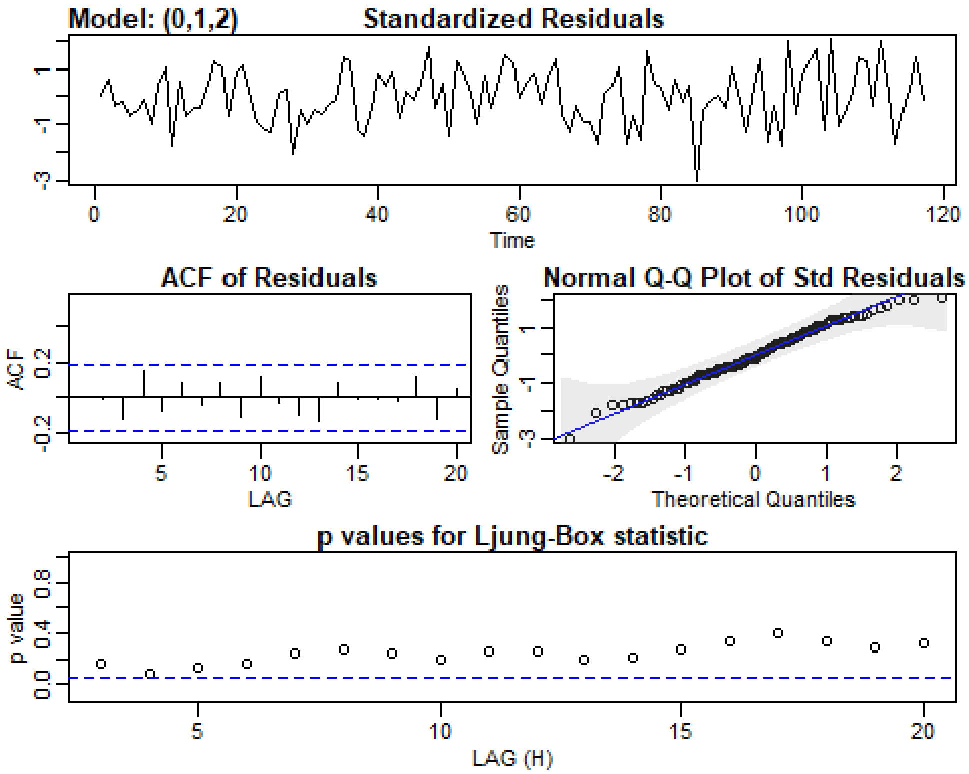
```
## [1] -546.8836
```

**Response: BIC suggest an MA(2) model; AIC also suggests an ARMA(4,2) model.**

(e) Fit the optimal ARIMA(p, d, q) model to (no differenced) training data, where p, q are selected from (c) and d is selected from (b).

(f) Check the diagnostics plots and comment on each plot.

```
M <- sarima(TRAINING, 0,1,2)

## initial  value -2.227967
## iter   2 value -2.325083
## iter   3 value -2.365718
## iter   4 value -2.367238
## iter   5 value -2.367902
## iter   6 value -2.368766
## iter   7 value -2.368774
## iter   8 value -2.368775
## iter   8 value -2.368775
## final  value -2.368775
## converged
## initial  value -2.365734
## iter   2 value -2.365746
## iter   3 value -2.365767
## iter   4 value -2.365775
## iter   4 value -2.365775
## iter   4 value -2.365775
## final  value -2.365775
## converged
```




```
M

## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D
,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixe
d,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ma1      ma2  constant
##       -0.5323  -0.2212    0.0048
## s.e.   0.0848   0.0822    0.0022
##
## sigma^2 estimated as 0.008758:  log likelihood = 109.83,  aic = -211.67
##
## $degrees_of_freedom
## [1] 113
##
## $ttable
##          Estimate     SE t.value p.value
## ma1       -0.5323 0.0848 -6.2795  0.0000
## ma2       -0.2212 0.0822 -2.6913  0.0082
## constant   0.0048 0.0022  2.1718  0.0320
##
```

```
## $AIC
## [1] -1.824708
##
## $AICc
## [1] -1.82286
##
## $BIC
## [1] -1.729756
```

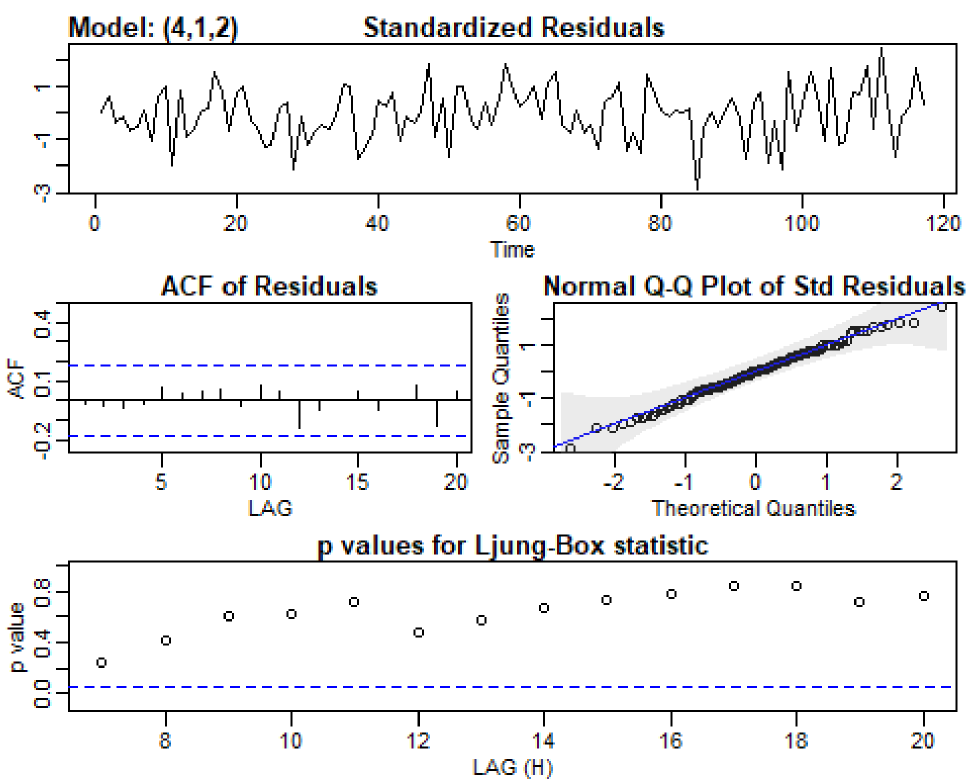
```
N <- sarima(TRAINING, 4,1,2)
```

```
## initial  value -2.214331
## iter   2 value -2.350495
## iter   3 value -2.369009
## iter   4 value -2.372395
## iter   5 value -2.376101
## iter   6 value -2.378948
## iter   7 value -2.381024
## iter   8 value -2.383901
## iter   9 value -2.387263
## iter  10 value -2.392602
## iter  11 value -2.399314
## iter  12 value -2.401882
## iter  13 value -2.404254
## iter  14 value -2.407097
## iter  15 value -2.410847
## iter  16 value -2.415446
## iter  17 value -2.415682
## iter  18 value -2.427094
## iter  19 value -2.430705
## iter  20 value -2.433987
## iter  21 value -2.434603
## iter  22 value -2.435243
## iter  22 value -2.435243
## iter  23 value -2.439460
## iter  24 value -2.439819
## iter  24 value -2.439819
## iter  25 value -2.439933
## iter  26 value -2.440027
## iter  27 value -2.440224
## iter  28 value -2.440307
## iter  29 value -2.440492
## iter  30 value -2.440573
## iter  31 value -2.440736
## iter  32 value -2.440817
## iter  33 value -2.440956
## iter  34 value -2.441039
## iter  35 value -2.441155
## iter  36 value -2.441240
## iter  37 value -2.441334
```

```
## iter  38 value -2.441420
## iter  39 value -2.441494
## iter  40 value -2.441581
## iter  40 value -2.441581
## iter  41 value -2.441638
## iter  42 value -2.441723
## iter  42 value -2.441723
## iter  43 value -2.441765
## iter  44 value -2.441849
## iter  44 value -2.441849
## iter  45 value -2.441881
## iter  46 value -2.441960
## iter  47 value -2.441960
## iter  48 value -2.441984
## iter  49 value -2.442061
## iter  50 value -2.442064
## iter  51 value -2.442105
## iter  52 value -2.442264
## iter  52 value -2.442264
## iter  53 value -2.442467
## iter  54 value -2.442472
## iter  54 value -2.442472
## iter  55 value -2.442502
## iter  56 value -2.442605
## iter  56 value -2.442605
## iter  57 value -2.442647
## iter  58 value -2.442666
## iter  59 value -2.442668
## iter  60 value -2.442677
## iter  61 value -2.442690
## iter  61 value -2.442690
## iter  62 value -2.442696
## iter  63 value -2.442704
## iter  63 value -2.442704
## iter  64 value -2.442712
## iter  65 value -2.442722
## iter  65 value -2.442722
## iter  66 value -2.442727
## iter  67 value -2.442736
## iter  67 value -2.442736
## iter  68 value -2.442742
## iter  69 value -2.442750
## iter  69 value -2.442750
## iter  70 value -2.442756
## iter  71 value -2.442764
## iter  71 value -2.442764
## iter  72 value -2.442770
## iter  73 value -2.442777
## iter  73 value -2.442777
## iter  74 value -2.442784
```

```
## iter  75 value -2.442791
## iter  75 value -2.442791
## iter  76 value -2.442797
## iter  77 value -2.442804
## iter  77 value -2.442804
## iter  78 value -2.442810
## iter  79 value -2.442816
## iter  79 value -2.442816
## iter  80 value -2.442823
## iter  81 value -2.442829
## iter  81 value -2.442829
## iter  82 value -2.442835
## iter  83 value -2.442841
## iter  83 value -2.442841
## iter  84 value -2.442848
## iter  85 value -2.442853
## iter  85 value -2.442853
## iter  86 value -2.442860
## iter  87 value -2.442865
## iter  87 value -2.442865
## iter  88 value -2.442872
## iter  89 value -2.442877
## iter  89 value -2.442877
## iter  90 value -2.442884
## iter  91 value -2.442889
## iter  91 value -2.442889
## iter  92 value -2.442896
## iter  93 value -2.442901
## iter  93 value -2.442901
## iter  94 value -2.442908
## iter  95 value -2.442913
## iter  95 value -2.442913
## iter  96 value -2.442919
## iter  97 value -2.442924
## iter  97 value -2.442924
## iter  98 value -2.442931
## iter  99 value -2.442936
## iter  99 value -2.442936
## iter 100 value -2.442942
## final  value -2.442942
## stopped after 100 iterations
## initial  value -2.227967
## iter   2 value -2.347959
## iter   3 value -2.377795
## iter   4 value -2.379996
## iter   5 value -2.383626
## iter   6 value -2.386832
## iter   7 value -2.388224
## iter   8 value -2.391012
## iter   9 value -2.392302
```

```
## iter  10 value -2.393382
## iter  11 value -2.395564
## iter  12 value -2.395738
## iter  13 value -2.397841
## iter  14 value -2.400238
## iter  15 value -2.401668
## iter  16 value -2.402203
## iter  17 value -2.402228
## iter  18 value -2.402267
## iter  19 value -2.402275
## iter  20 value -2.402278
## iter  21 value -2.402295
## iter  22 value -2.402325
## iter  23 value -2.402351
## iter  24 value -2.402358
## iter  25 value -2.402359
## iter  26 value -2.402359
## iter  26 value -2.402359
## final  value -2.402359
## converged
```




```
N

## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D
```

```
,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixe
d,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ar2      ar3     ar4      ma1      ma2  constant
##       -0.1902  0.3022  -0.0115  0.2499  -0.3026  -0.6974    0.0046
## s.e.   0.2470  0.1435   0.0972  0.1000   0.2537   0.2530    0.0006
##
## sigma^2 estimated as 0.007945:  log likelihood = 114.08,  aic = -212.15
##
## $degrees_of_freedom
## [1] 109
##
## $ttable
##          Estimate     SE t.value p.value
## ar1       -0.1902 0.2470 -0.7700  0.4430
## ar2        0.3022 0.1435  2.1066  0.0374
## ar3       -0.0115 0.0972 -0.1178  0.9065
## ar4        0.2499 0.1000  2.4998  0.0139
## ma1       -0.3026 0.2537 -1.1925  0.2357
## ma2       -0.6974 0.2530 -2.7562  0.0069
## constant   0.0046 0.0006  7.6481  0.0000
##
## $AIC
## [1] -1.82891
##
## $AICc
## [1] -1.81997
##
## $BIC
## [1] -1.639007
```

**Response:**

**ARIMA(0,1,2):**

**X(t)-0.0048 = W(t) - 0.5323*W(t-1) - 0.2212*W(t-2); sigma^2 estimated as 0.008758**

**ARIMA(4,1,2):**

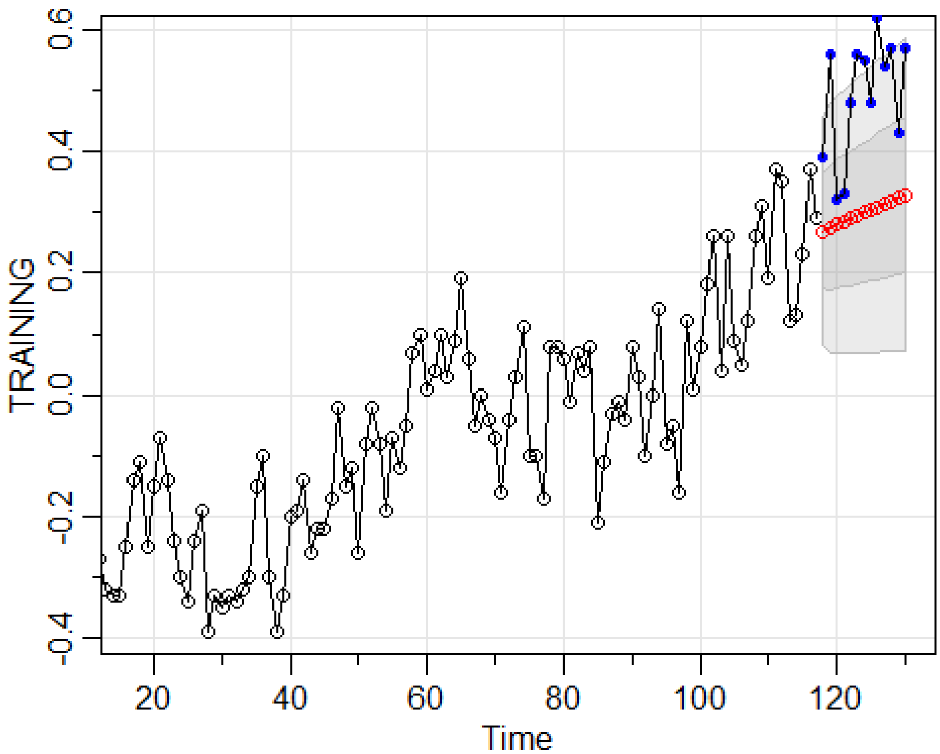
**X(t)-0.0046 = -0.1902*(X(t-1)-0.0046) + 0.3022*(X(t-2)-0.0046) - 0.0115*(X(t-3)-0.0046) + 0.2499*(X(t-4)-0.0046) + W(t) - 0.3026W(t-1) - 0.6974W(t-2); sigma^2 estimated as 0.007945**

**The diagnostics for the ARIMA(0,1,2) look great. All points on the Standardized Residuals plot appear to be within the +/- 3 interval. No ACF's of Residuals appear to be signifigant. The QQ plot looks great, and all p-values for the Ljung-Box test are above 0.05.**

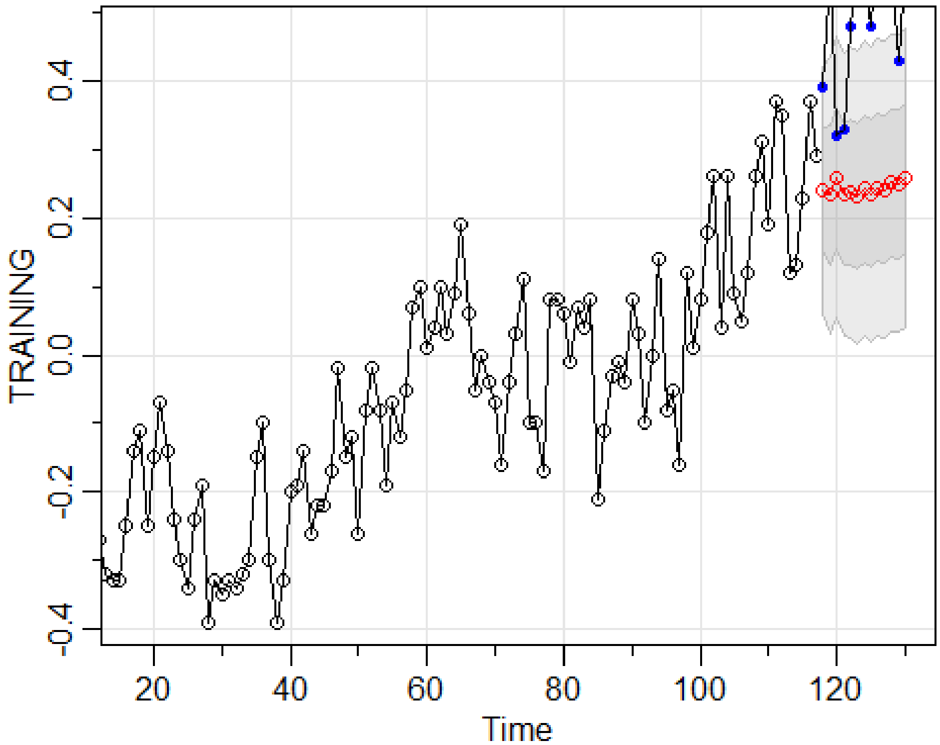
**The diagnostics for the ARIMA(4,1,2) also look great. All points on the Standardized Residuals plot appear to be within the +/- 3 interval. No ACF's of Residuals appear to be significant. The QQ plot looks great, and all p-values for the Ljung-Box test are above 0.05.**

(g) After the optimal model(s) is fitted, forecast (with respect to "gtemp" not the differenced series) the next 13 years which belongs to the testing set. Plot the predictions, 95% prediction intervals with the observed data in one graph. Assess the performance of the predictions, i.e., observe if the actual testing data points are included in the prediction intervals.

```
par(mfrow = c(1,1))
sarima.for(TRAINING, n.ahead = 13, 0,1,2)

## $pred
## Time Series:
## Start = 118
## End = 130
## Frequency = 1
##  [1] 0.2693195 0.2758779 0.2807009 0.2855238 0.2903468 0.2951697 0.2999927
##  [8] 0.3048156 0.3096386 0.3144615 0.3192845 0.3241074 0.3289303
##
## $se
## Time Series:
## Start = 118
## End = 130
## Frequency = 1
##  [1] 0.09358163 0.10331195 0.10585581 0.10833995 0.11076840 0.11314474
##  [7] 0.11547218 0.11775363 0.11999171 0.12218881 0.12434709 0.12646854
## [13] 0.12855498

points(118:130,TESTING, col="blue", pch=20)
lines(118:130, TESTING)
```




```
par(mfrow = c(1,1))
sarima.for(TRAINING, n.ahead = 13, 4,1,2)

## $pred
## Time Series:
## Start = 118
## End = 130
## Frequency = 1
##  [1] 0.2404997 0.2332837 0.2586037 0.2351919 0.2380202 0.2313221 0.2430551
##  [8] 0.2359254 0.2446202 0.2420125 0.2511590 0.2497590 0.2580014
##
## $se
## Time Series:
## Start = 118
## End = 130
## Frequency = 1
##  [1] 0.08949941 0.10068152 0.10254911 0.10307143 0.10653665 0.10704765
##  [7] 0.10772170 0.10779898 0.10831371 0.10834159 0.10854430 0.10854417
## [13] 0.10867296

points(118:130,TESTING, col="blue", pch=20)
lines(118:130, TESTING)
```



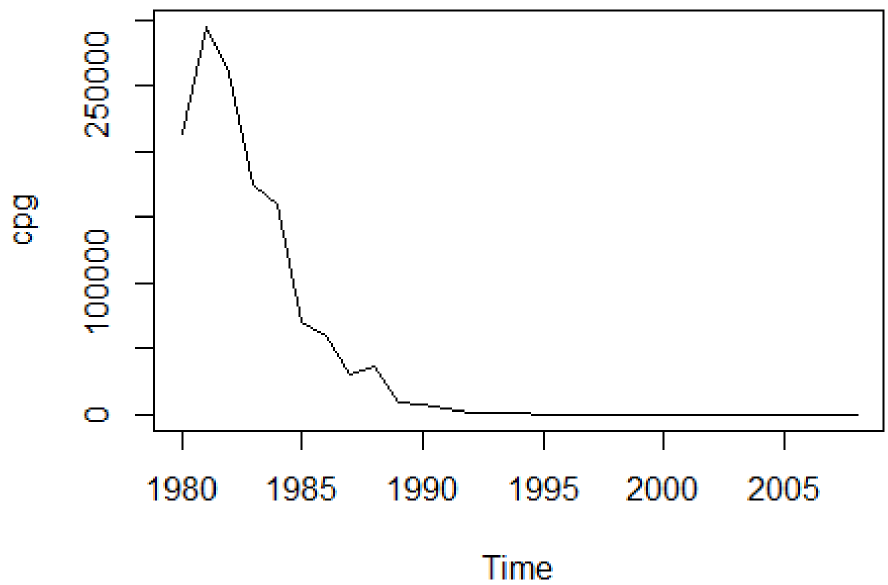

**Response: There are 4 points outside of the 95% confidence interval in the ARIMA(0,1,2) model, and there are 9 points outside of the 95% confidence interval for the ARIMA(4,1,2). It is clear that ARIMA(0,1,2) is the better performing model.**

3. [Problem 3.36 in text book] (R) One of the remarkable technological developments in the computer industry has been the ability to store information densely on a hard drive. In addition, the cost of storage has steadily declined causing problems of too much data as opposed to big data. The data set for this assignment is “cpg”, which consists of the median annual retail price per GB of hard drives, say ct, taken from a sample of manufacturers from 1980 to 2008.

```
data(cpg)
```

(a) Plot ct and describe what you see.

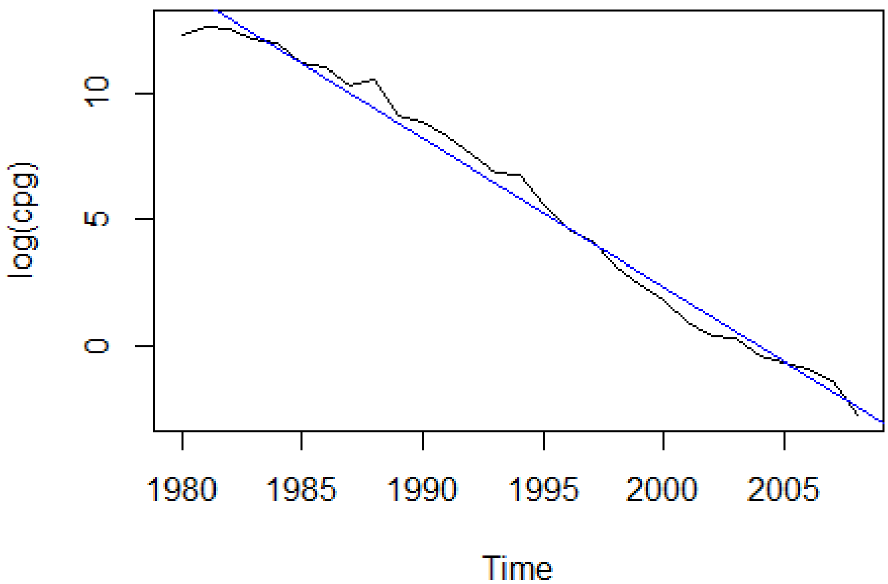
```
plot.ts(cpg)
```




**Response: There appears to be an exponential (quadratic) decrease.**

(b) It seems that the curve ct versus t behaves like ct ≈ αeβ·t. Then let's fit a linear regression of logct on t and plot the fitted line to compare it to the logged data. Comment on what you observe. [hint: When you fit a linear regression, use lm(log(cpg)~time(cpg)). After fitting a regression model, first plot thelog(cpg) and use abline() function to add the fitted line]

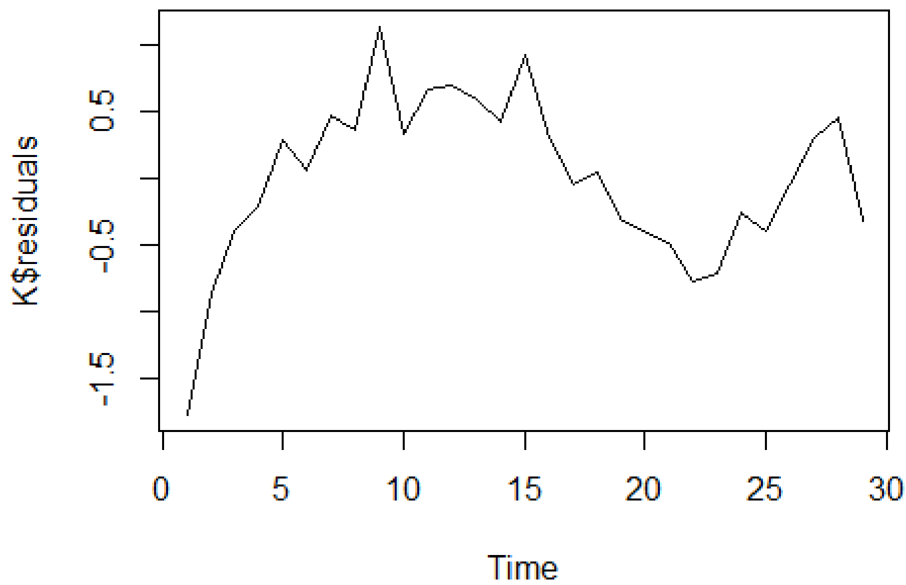
```
K <- lm(log(cpg)~time(cpg))
plot.ts(log(cpg))
abline(K, col="blue")
```



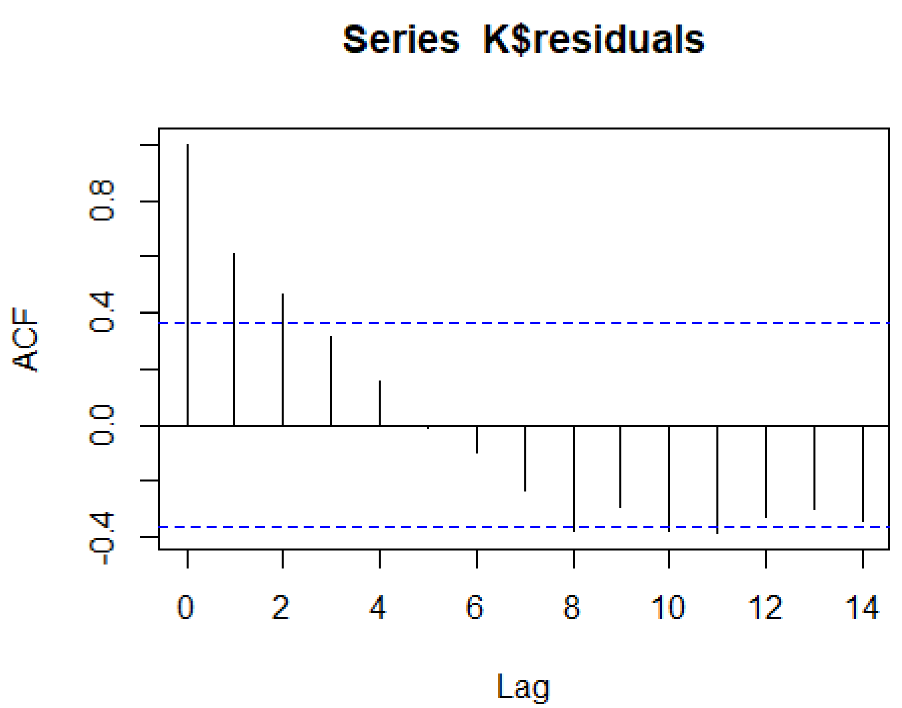

**Response: It appears that it changed the exponential decrease into a linear decrease.**

(c) Inspect the residuals of the linear regression fit by generating the time series plot,ACF and PACF plot. Visually decide which ARMA model is appropriate for the residual.

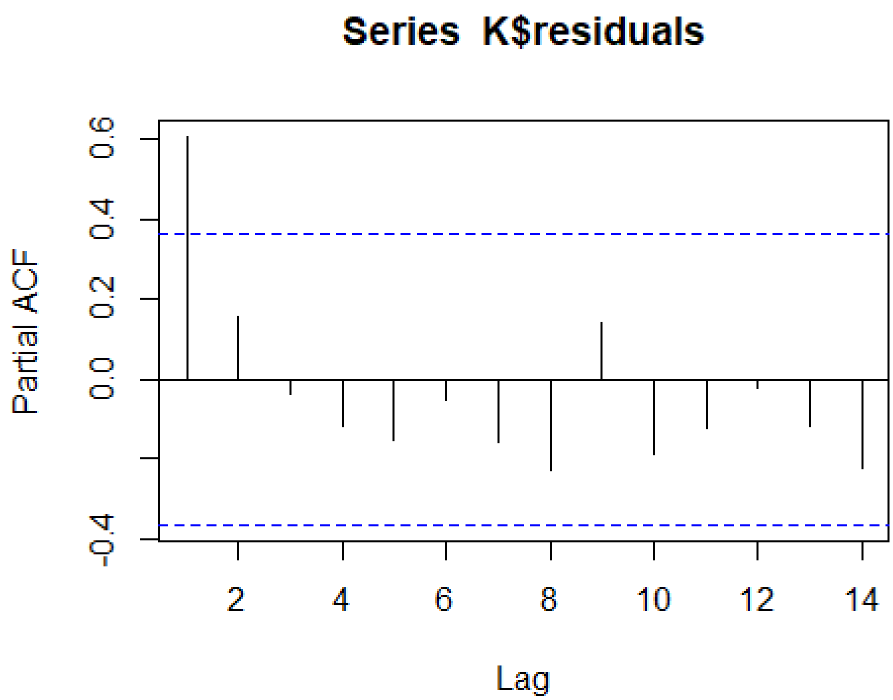
```
K$residuals

##           1           2           3           4           5           6 
## -1.77156094 -0.86080012 -0.40201622 -0.21283425  0.28263122  0.05521491 
##           7           8           9          10          11          12 
##  0.47195722  0.36388767  1.13128685  0.33007012  0.66383332  0.68929516 
##          13          14          15          16          17          18 
##  0.58122560  0.42186275  0.91320790  0.29238409 -0.04463736  0.04725744 
##          19          20          21          22          23          24 
## -0.31053798 -0.39840482 -0.48119657 -0.76816142 -0.71782497 -0.26173946 
##          25          26          27          28          29 
## -0.39922290 -0.04854598  0.30390936  0.44715423 -0.31769486 

plot.ts(K$residuals)
```




```
acf(K$residuals)
```




```
pacf(K$residuals)
```

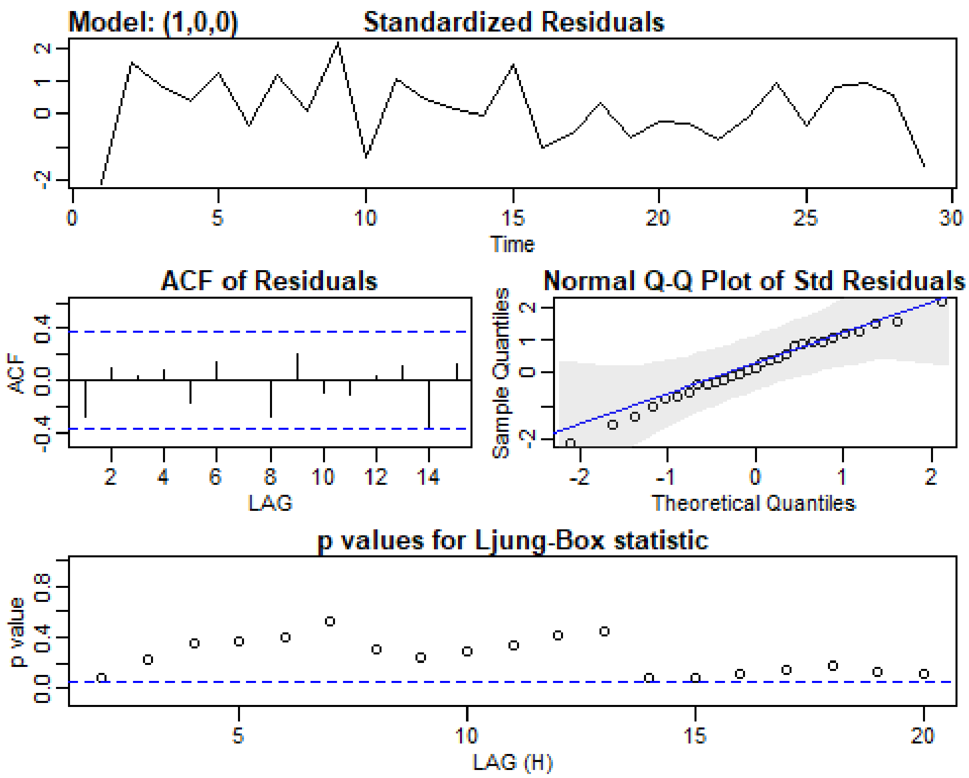



**Response: ACF tails off and the PACF cuts off at lag=1. Therefore, we should use the AR(1) model.**

(d) Fit the model chosen from (c) to the residuals and report the fitted model.

```
G <- sarima(K$residuals, 1,0,0)

## initial  value -0.669056
## iter   2 value -0.968880
## iter   3 value -0.991232
## iter   4 value -1.051693
## iter   5 value -1.053940
## iter   6 value -1.061132
## iter   7 value -1.061236
## iter   8 value -1.061270
## iter   9 value -1.061293
## iter  10 value -1.061293
## iter  10 value -1.061293
## iter  10 value -1.061293
## final  value -1.061293
## converged
## initial  value -0.809581
## iter   2 value -0.868173
## iter   3 value -0.873691
## iter   4 value -0.876061
## iter   5 value -0.876326
## iter   6 value -0.876361
## iter   7 value -0.876367
```

```
## iter   8 value -0.876370
## iter   9 value -0.876370
## iter  10 value -0.876370
## iter  10 value -0.876370
## iter  10 value -0.876370
## final  value -0.876370
## converged
```




G

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D
,
##     Q), period = S), xreg = xmean, include.mean = FALSE, transform.pars =
trans,
##     fixed = fixed, optim.control = list(trace = trc, REPORT = 1, reltol =
tol))
##
## Coefficients:
##          ar1    xmean
##       0.8239  -0.2548
## s.e.  0.1253   0.4096
##
## sigma^2 estimated as 0.1666:  log likelihood = -15.73,  aic = 37.47
##
```

```
## $degrees_of_freedom
## [1] 27
##
## $ttable
##        Estimate     SE t.value p.value
## ar1      0.8239 0.1253  6.5763  0.0000
## xmean   -0.2548 0.4096 -0.6221  0.5391
##
## $AIC
## [1] 1.292033
##
## $AICc
## [1] 1.307948
##
## $BIC
## [1] 1.433478
```

```
summary(K)
```

```
##
## Call:
## lm(formula = log(cpg) ~ time(cpg))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.77156 -0.39840  0.04726  0.42186  1.13129
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1172.49431   27.57793   42.52   <2e-16 ***
## time(cpg)     -0.58508    0.01383  -42.30   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6231 on 27 degrees of freedom
## Multiple R-squared:  0.9851, Adjusted R-squared:  0.9846
## F-statistic:  1790 on 1 and 27 DF,  p-value: < 2.2e-16
```

**Response: X(t) = 0.8239X(t-1) + Wt; sigma^2 estimated as 0.1666**

(e) Report the final model with respect to log(ct). Combine the results of a linear regression part and time series part.

**Response:**

**Y(t) - 1172.49431 + 0.58508*t* = *0.82*(Y(t-1) - 1172.49431 + 0.58508*(t-1)) + W(t); sigma^2 estimated as 0.1666**