econ144hw2

Sijia Hua

4/17/2019

```
library(seasonal)
library(dynlm)
```

```
## Warning: package 'dynlm' was built under R version 3.5.2
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

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

```
library(gdata)
```

```
## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.
```

```
##
```

```
## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.
```

```
##
## Attaching package: 'gdata'
```

```
## The following object is masked from 'package:stats':
##
##     nobs
```

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

```
## The following object is masked from 'package:base':
##
##     startsWith
```

```
require(graphics)
library("readxl")
```

```
## Warning: package 'readxl' was built under R version 3.5.2
```

```
library('xts')
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:gdata':
##
##     first, last
```

```
library('forecast');
```

```
## Warning: package 'forecast' was built under R version 3.5.2
```

```
library('fma')
library('expsmooth')
library('lmtest')
library('tseries')
library('Quandl')
library('fpp');
library('urca')
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 3.5.2
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

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

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
setwd("/Users/Renaissance/Desktop/econ144/econ144hw2")
```

A.Chapter3.3

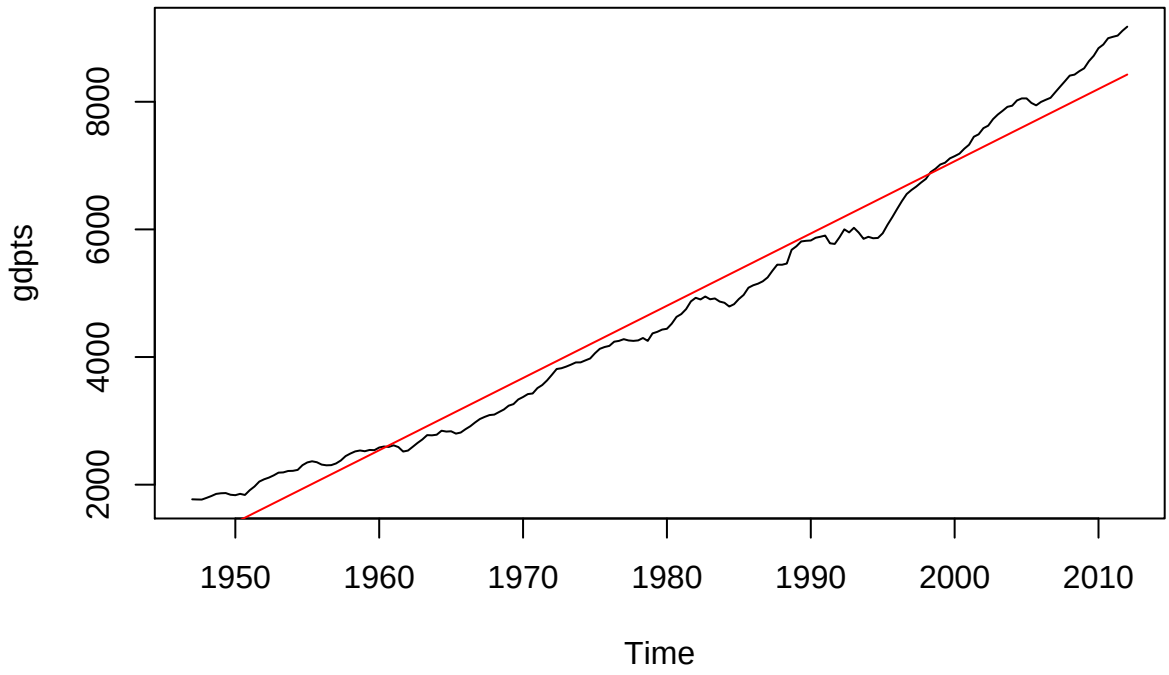
(a)

Exact Definition: Real Gross Domestic Product, 1 Decimal

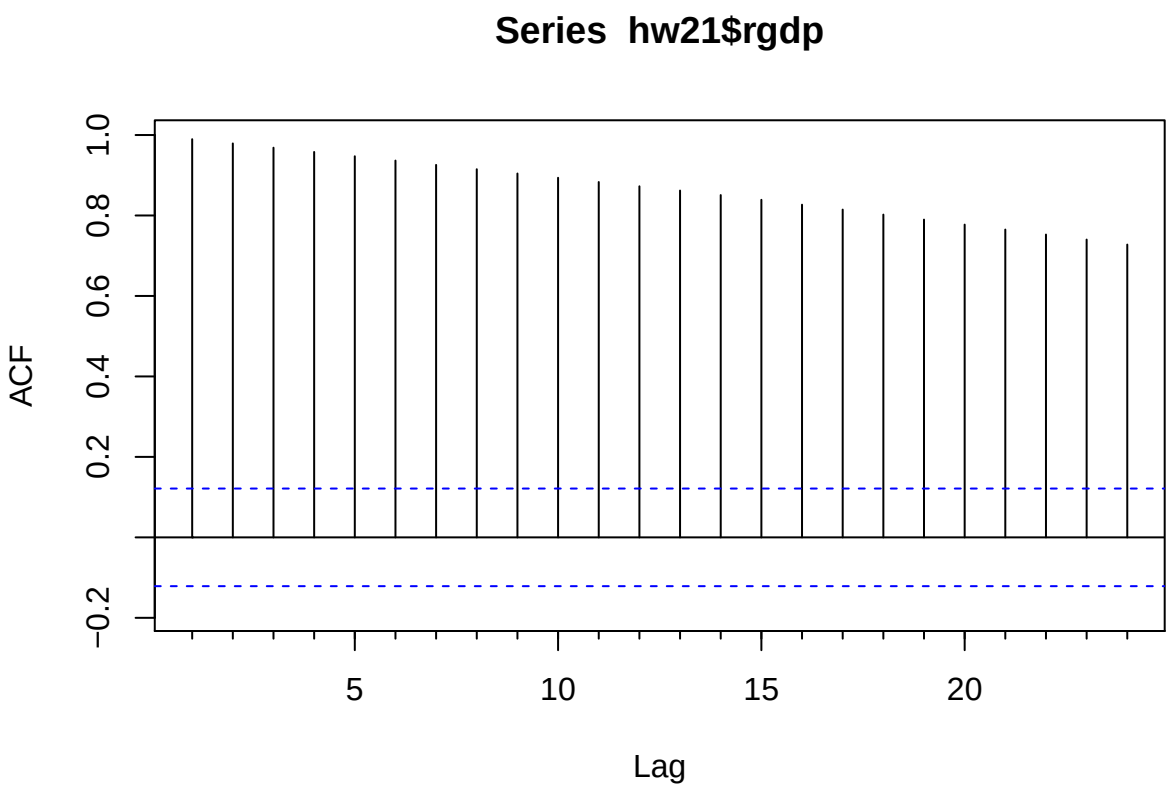
Periodicity: Quarterly

Unites: Billions of Chained 2005 Dollars

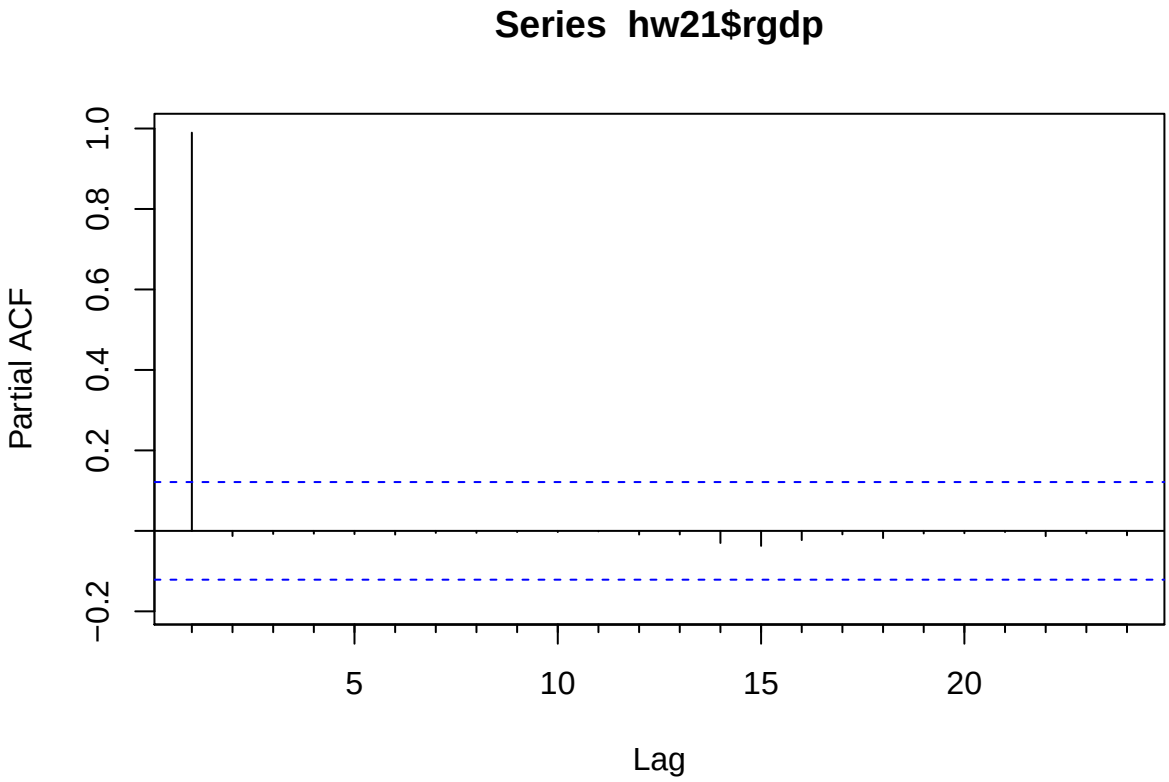
```
hw21<-read.xls('hw2adata.xlsx',sheet = 3) #import data
gdpts<-ts(hw21$rgdp,start=1947,end=2012,frequency  = 3)

fita=tslm(gdpts ~ trend)
plot(gdpts)
lines(fita$fitted.values,col="red")
```




```
# check acf and pacf
Acf(hw21$rgdp)
```




```
Pacf(hw21$rgdp)
```




```
# check ADF
aatest <- adf.test(hw21$rgdp,alternative = 'stationary')
print(aatest)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  hw21$rgdp
## Dickey-Fuller = -1.6762, Lag order = 6, p-value = 0.7123
## alternative hypothesis: stationary
```

```
# check KPSS
aktest <- kpss.test(hw21$rgdp)
```

```
## Warning in kpss.test(hw21$rgdp): p-value smaller than printed p-value
```

```
aktest
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  hw21$rgdp
## KPSS Level = 4.3083, Truncation lag parameter = 5, p-value = 0.01
```

From the plot graph of this data set, since its trend is not constant, this time series is probably not a stationary set. From the graph of Acf, this time series is non-stationary. From the test of ADF, H0 that this series is non-stationary can not be reject. From the test of KPSS, pvalue is less than 0.01,this time series is non-stationary.

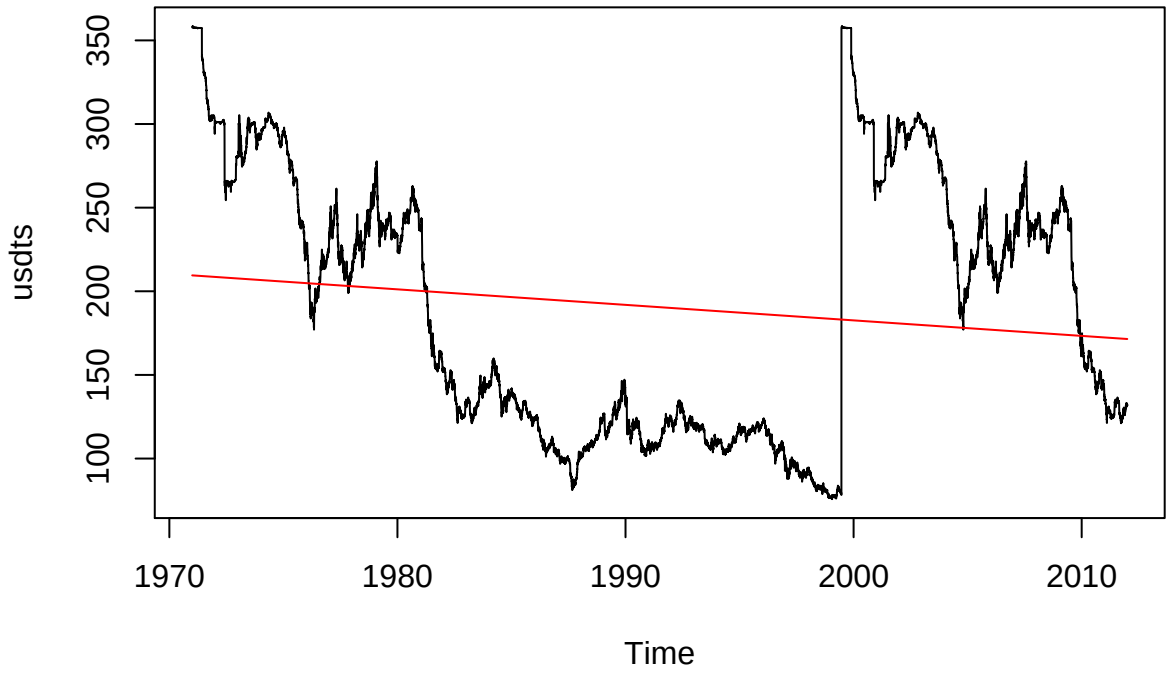
(b)

Exact Definition: Japan / U.S. Foreign Exchange Rate

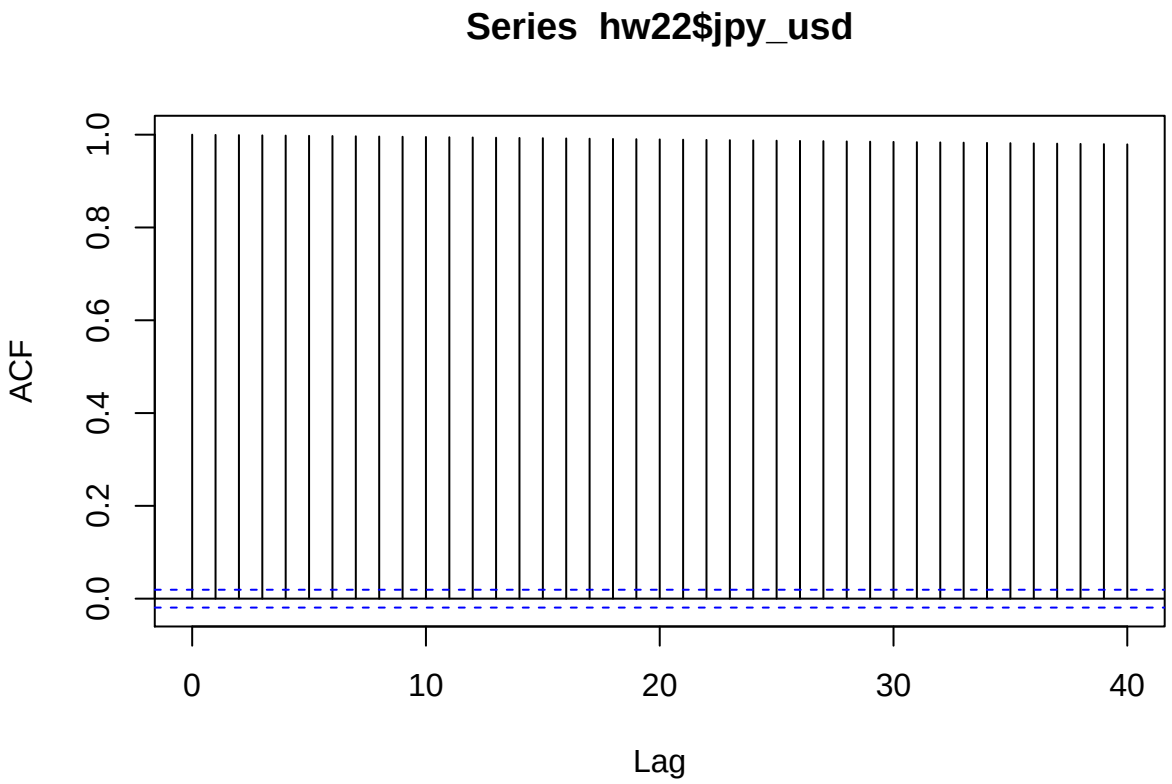
Periodicity: Daily

Unites: Japanese Yen to One U.S. Dollar

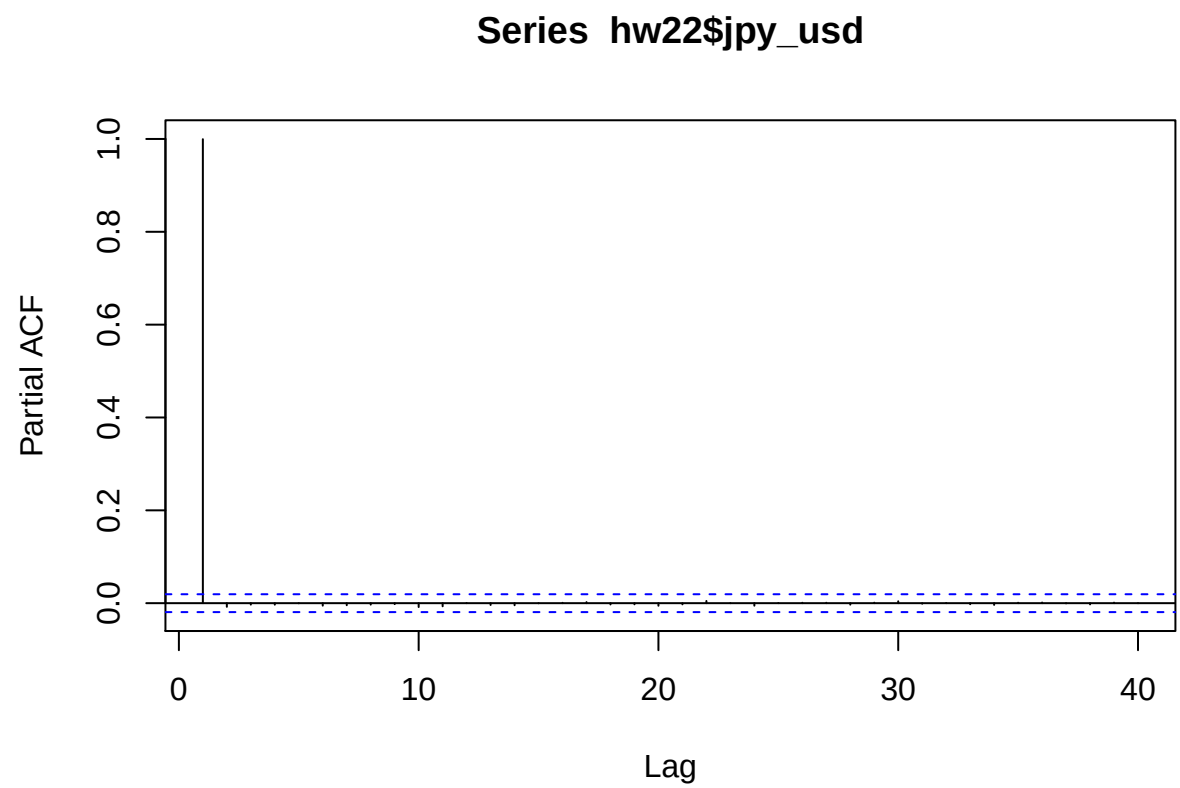
```
hw22<-read.xls('hw2adata.xlsx',sheet = 4) #import data
usdts <- ts(hw22$jpy_usd,start=1971,end=2012,freq=365)

fitb=tslm(usdts ~ trend)
plot(usdts)
lines(fitb$fitted.values,col="red")
```




```
# check acf and pacf
acf(hw22$jpy_usd)
```




```
pacf(hw22$jpy_usd)
```




```
# check ADF
batest <- adf.test(hw22$jpy_usd,alternative = 'stationary')
print(batest)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  hw22$jpy_usd
## Dickey-Fuller = -2.609, Lag order = 21, p-value = 0.3203
## alternative hypothesis: stationary
```

```
# check KPSS
bktest <- kpss.test(hw22$jpy_usd)
```

```
## Warning in kpss.test(hw22$jpy_usd): p-value smaller than printed p-value
```

```
bktest
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  hw22$jpy_usd
## KPSS Level = 66.813, Truncation lag parameter = 12, p-value = 0.01
```

From the plot graph of this data set, since its trend is not constant, this time series is probably not a stationary set. By observing ACF graph, this time series probably is not stationary. From the test of ADF, H0 that this series is non-stationary can not be reject. By taking the test of KPSS, pvalue is less than 0.01,this time series is non-stationary.

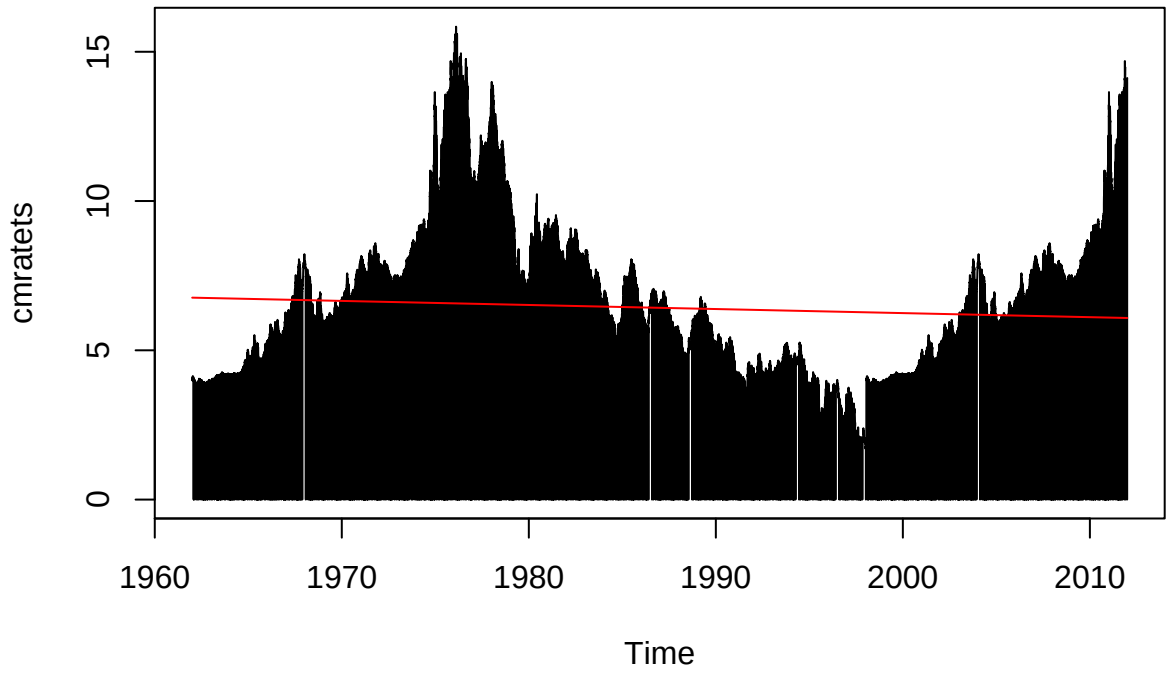
(c)

Exact Definition: 10-Year Treasury Constant Maturity Rate

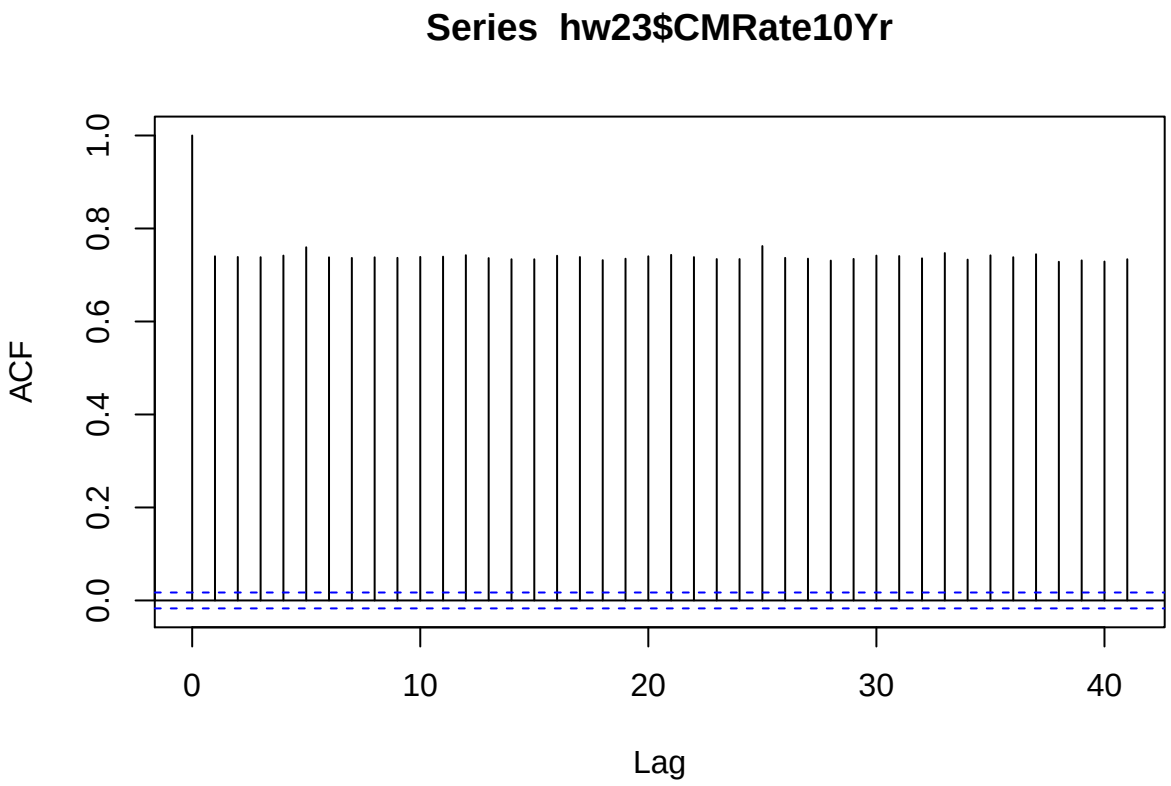
Periodicity: Daily

Unites: Percent

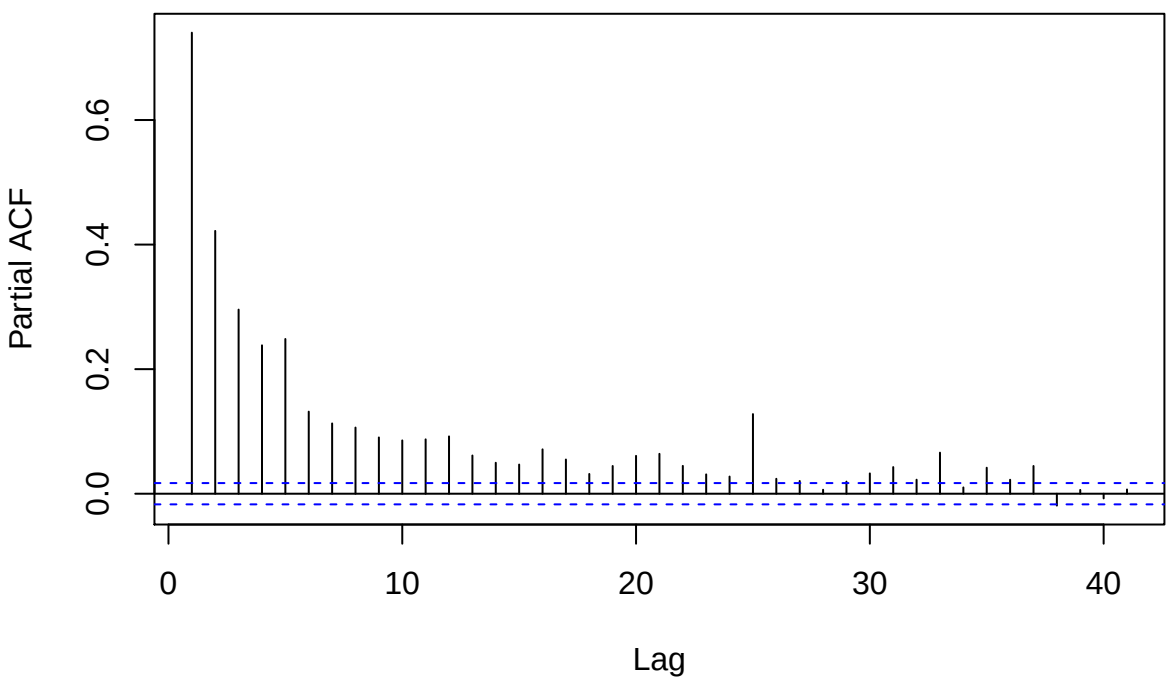
```
hw23<-read.xls('hw2adata.xlsx',sheet = 5) #import data
cmratets <- ts(hw23$CMRate10Yr,start=1962,end=2012,freq=365)

fitc=tslm(cmratets ~ trend)
plot(cmratets)
lines(fitc$fitted.values,col="red")
```




```
# check acf and pacf
acf(hw23$CMRate10Yr)
```




```
pacf(hw23$CMRate10Yr)
```




```
# check ADF
catest <- adf.test(hw23$CMRate10Yr,alternative = 'stationary')
print(catest)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  hw23$CMRate10Yr
## Dickey-Fuller = -2.9825, Lag order = 23, p-value = 0.162
## alternative hypothesis: stationary
```

```
# check KPSS
cktest <- kpss.test(hw23$CMRate10Yr)
```

```
## Warning in kpss.test(hw23$CMRate10Yr): p-value smaller than printed p-value
```

```
cktest
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  hw23$CMRate10Yr
## KPSS Level = 24.678, Truncation lag parameter = 13, p-value = 0.01
```

From the plot graph of this data set, since its trend is nearly constant, this time series is probably a stationary set. From the graph of Acf, this time series is non-stationary. From the test of ADF, H0 that this series is non-stationary can not be reject. From the test of KPSS, pvalue is less than 0.01,this time series is non-stationary.

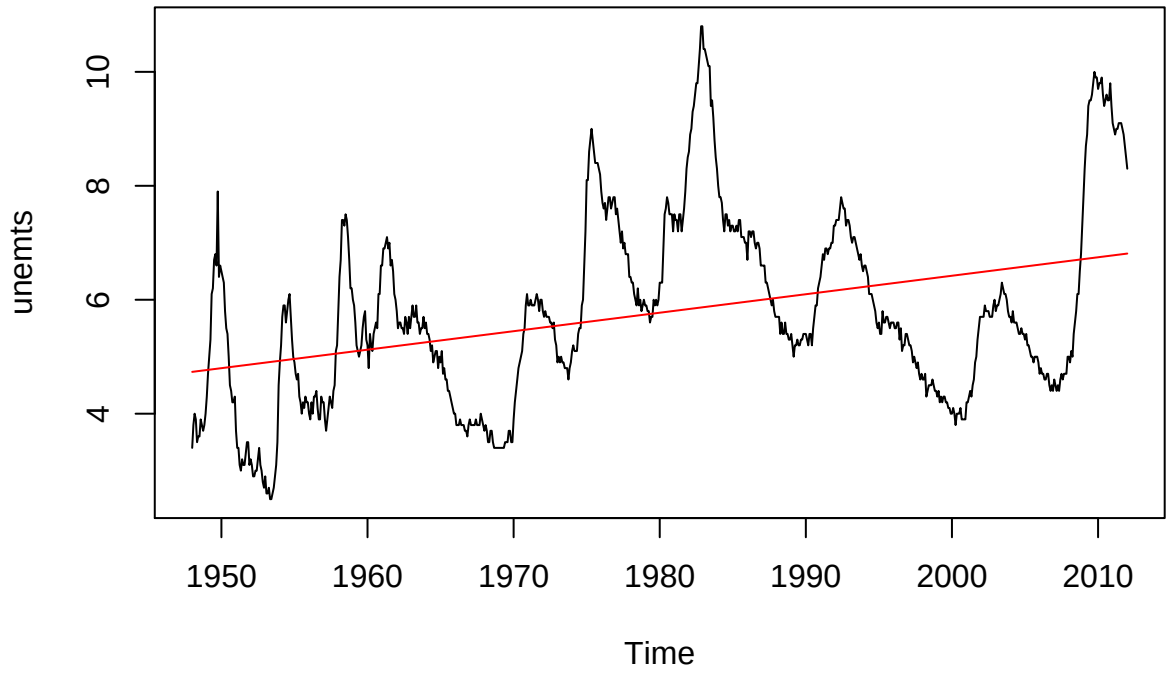
(d)

Exact Definition: Civilian Unemployment Rate

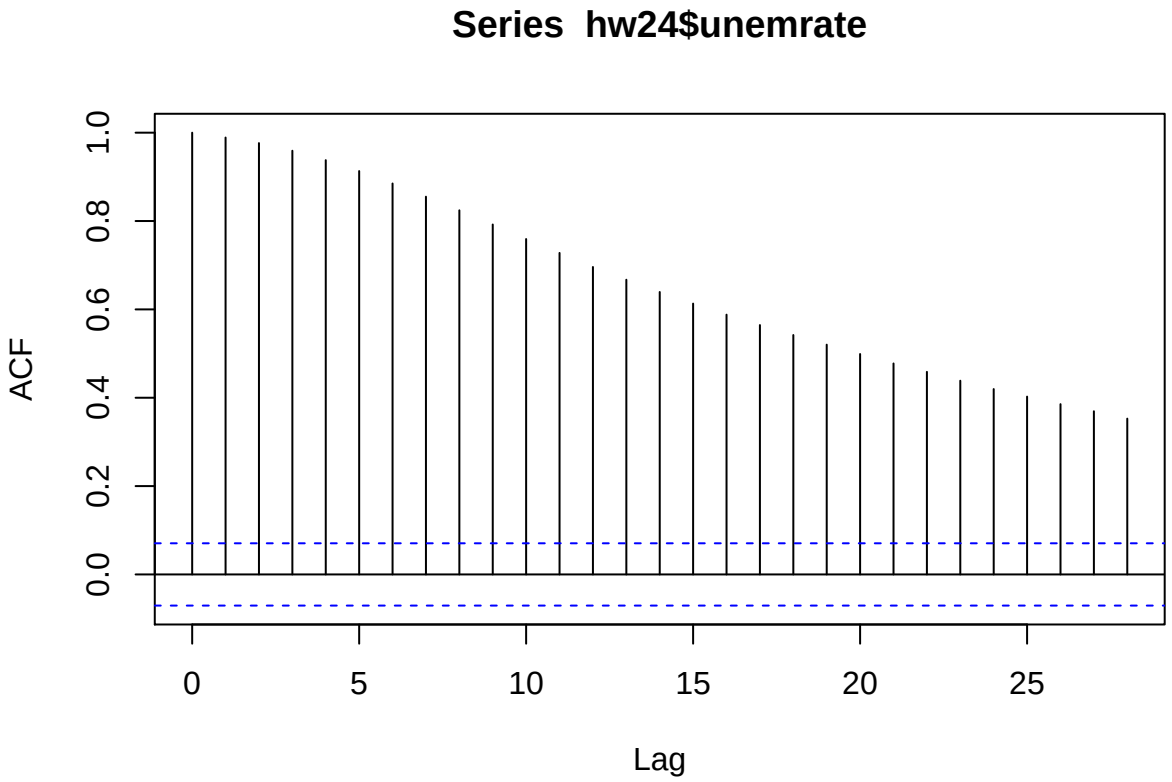
Periodicity: Monthly

Unites: Percent

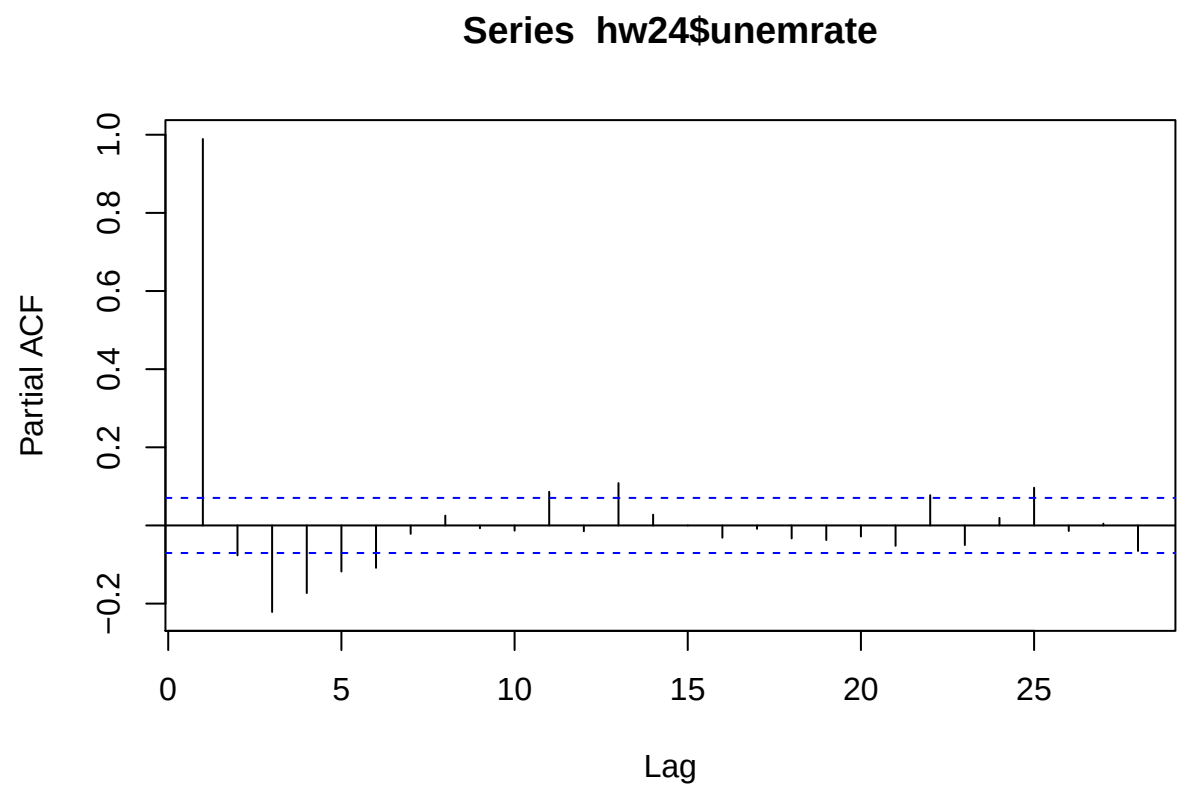
```
hw24<-read.xls('hw2adata.xlsx',sheet = 6) #import data
unemts <- ts(hw24$unemrate,start=1948,end=2012,freq=12)

fitd=tslm(unemts ~ trend)
plot(unemts)
lines(fitd$fitted.values,col="red")
```




```
# check acf and pacf
acf(hw24$unemrate)
```




```
pacf(hw24$unemrate)
```




```
# check ADF
datest <- adf.test(hw24$unemrate,alternative = 'stationary')
print(datest)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  hw24$unemrate
## Dickey-Fuller = -3.9275, Lag order = 9, p-value = 0.01262
## alternative hypothesis: stationary
```

```
# check KPSS
dktest <- kpss.test(hw24$unemrate)
```

```
## Warning in kpss.test(hw24$unemrate): p-value smaller than printed p-value
```

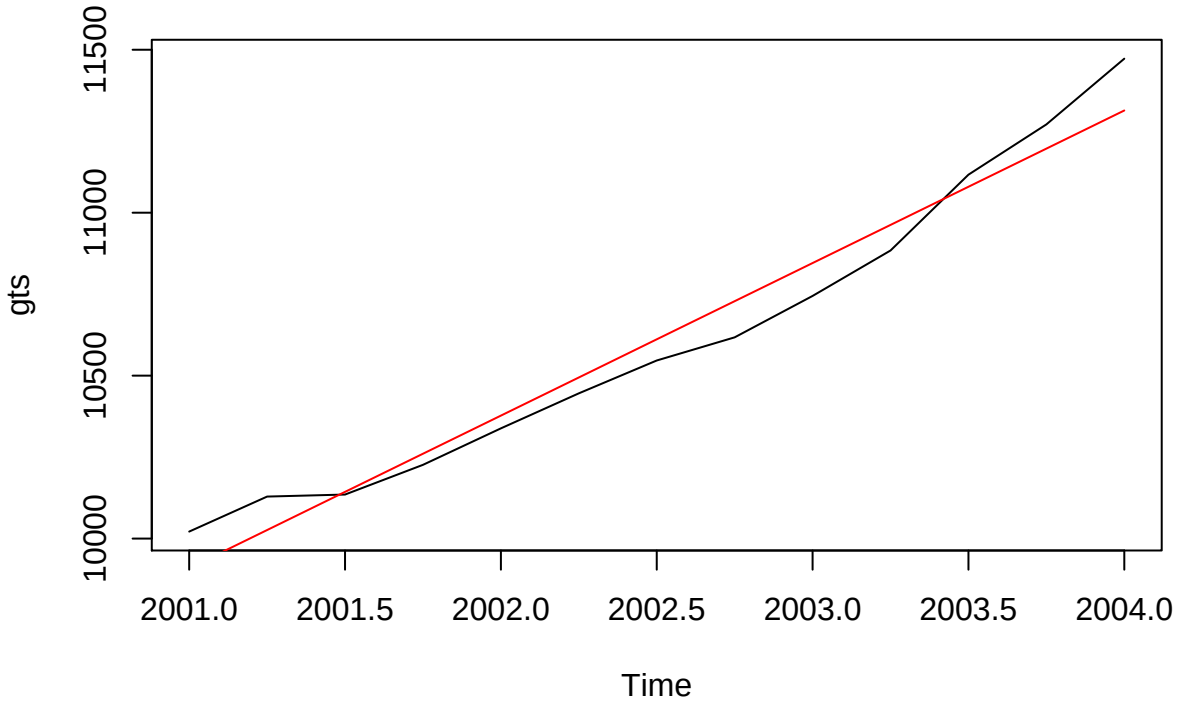
```
dktest
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  hw24$unemrate
## KPSS Level = 1.8714, Truncation lag parameter = 6, p-value = 0.01
```

From the plot graph of this data set, since its trend is not constant, this time series is probably not a stationary set. By observing ACF graph, this time series probably is not stationary. From the test of ADF, pvalue is less than 0.05, so H0 that this series is non-stationary can be reject. By taking the test of KPSS, pvalue is less than 0.01,this time series is non-stationary.

A.Chapter3.5

```
hw25<-read.xls('hw2adata.xlsx',sheet = 7) #import data
gts <- ts(hw25$GDP,start=2001,end=2004,freq=4)

fitg=tslm(gts ~ trend)
plot(gts)
lines(fitg$fitted.values,col="red")
```

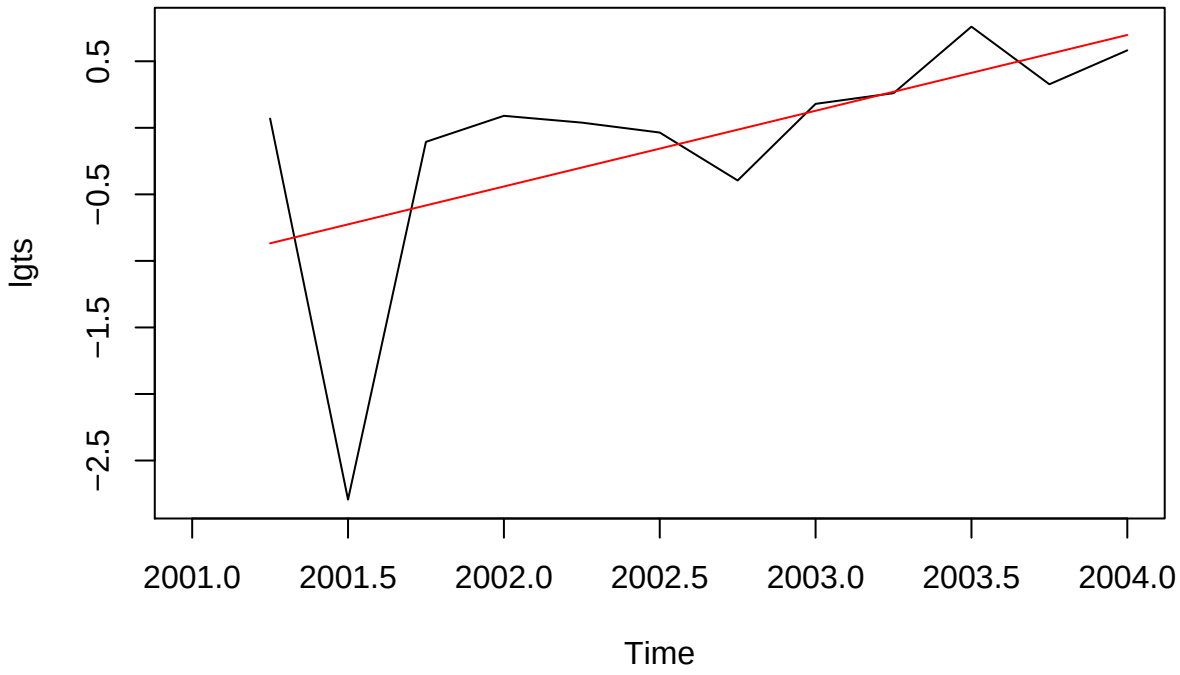


This can not be first order or second order weakly stationary, since it does not have a constant trend that indicates mean reversion. Since all orders of weakly stationary require each variable has identical mean, this time series does not have any order of weakly stationary.

```
glt <- vector(mode="numeric",length=0)
glt <- (hw25$GDP/Lag(hw25$GDP,1)-1)*100
glt
```

```
##  [1]         NA 1.07169585 0.06121099 0.89984312 1.09423741 1.03983285
##  [7] 0.96499038 0.67320912 1.19708029 1.29739590 2.13800074 1.38710229
## [13] 1.78956428 1.61166606 1.35020373 1.52265360
```

```
logg<-log(glt)

lgts <- ts(logg,start=2001,end=2004,freq=4)

fitllg=tslm(lgts ~ trend)
plot(lgts)
lines(fitllg$fitted.values,col="red")
```

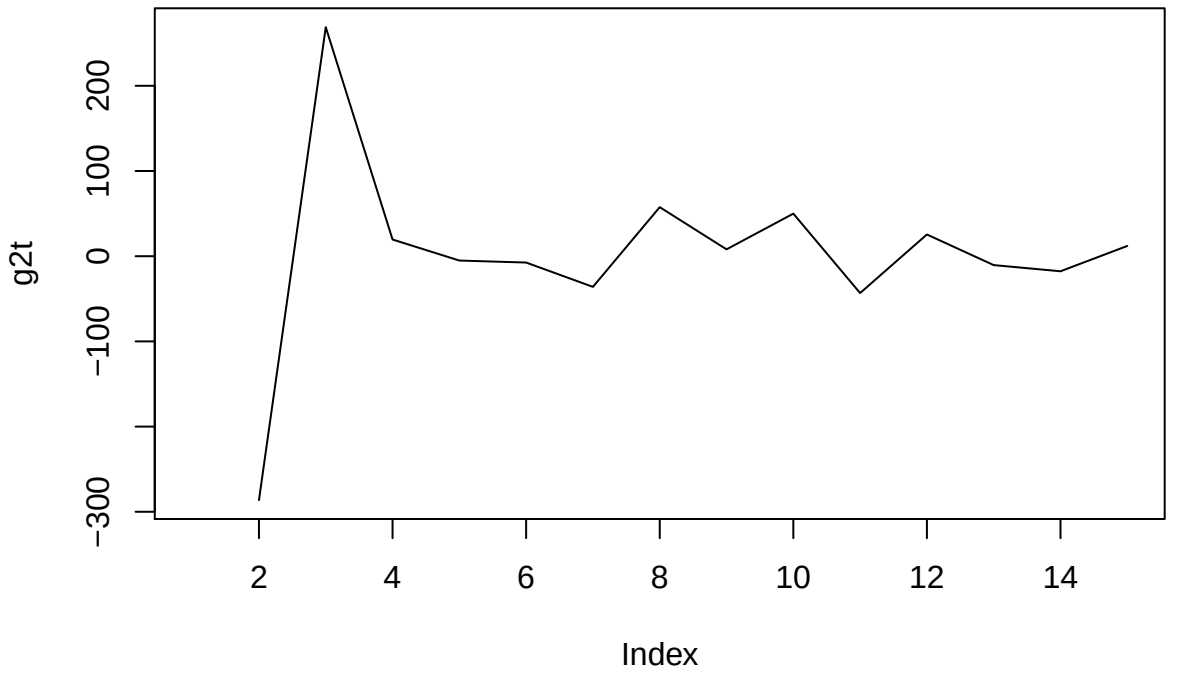


This can not be first order or second order weakly stationary, since it does not have a constant trend that indicates mean reversion. Since all orders of weakly stationary require each variable has identical mean, this time series does not have any order of weakly stationary.

```
g2t<-round(diff(logg)*100,5)
g2t
```

```
##  [1]         NA -286.26708  268.78937   19.55925   -5.09977   -7.46971
##  [7]  -36.00621   57.55848    8.04736   49.95121  -43.26543   25.47553
## [13]  -10.47037  -17.70130   12.01991
```

```
plot(g2t,type="l")
```

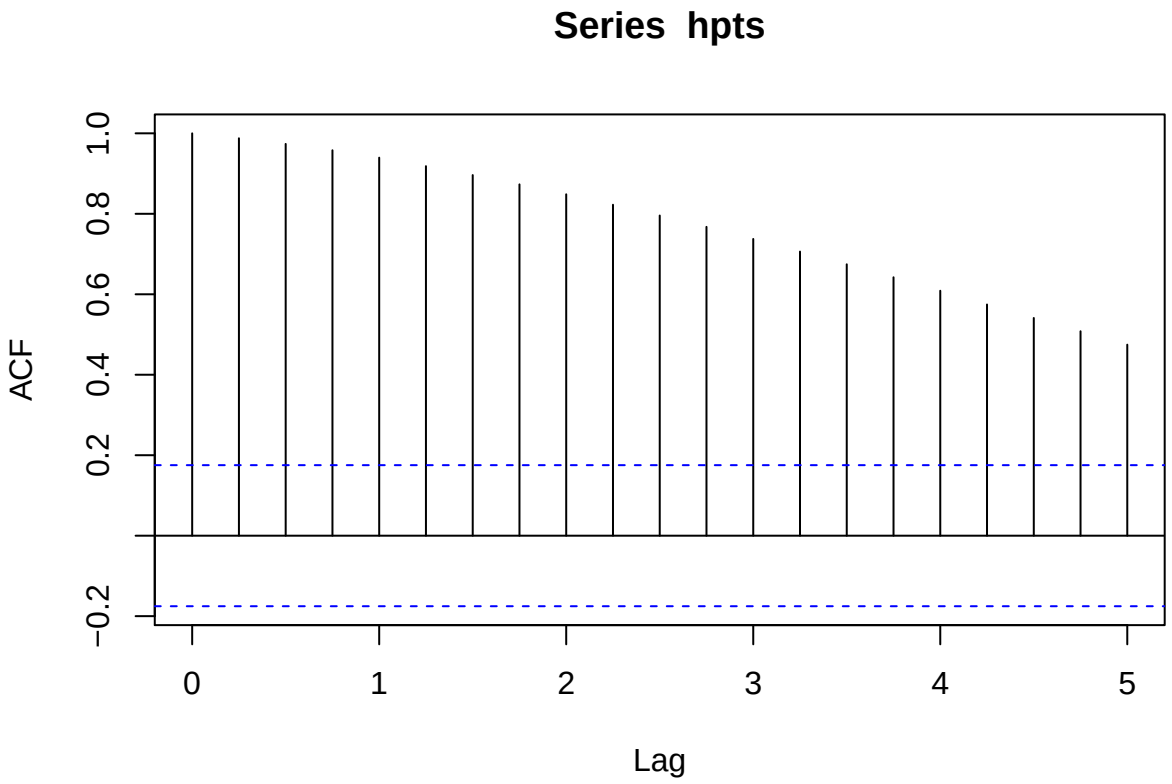


```
# The results of g1t and g2t are quite similar, they all indicate return of GDP. They have siginificant
```

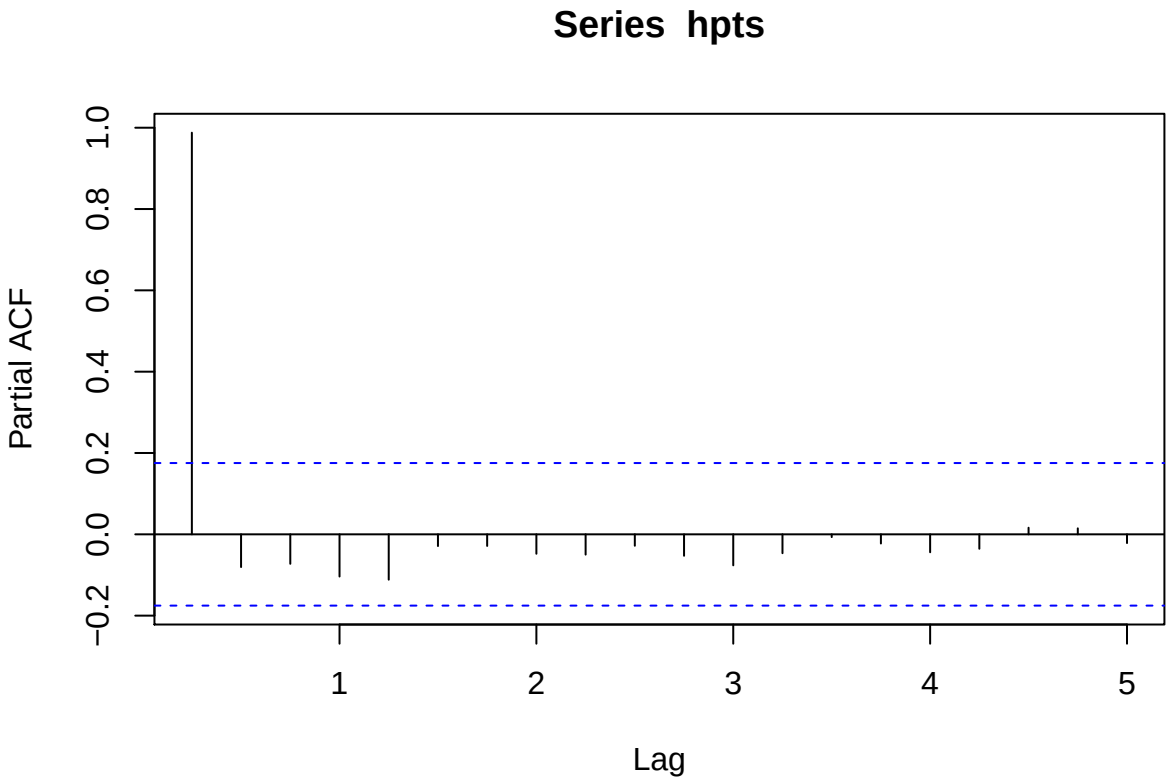
Book a Chapter 4

4.3

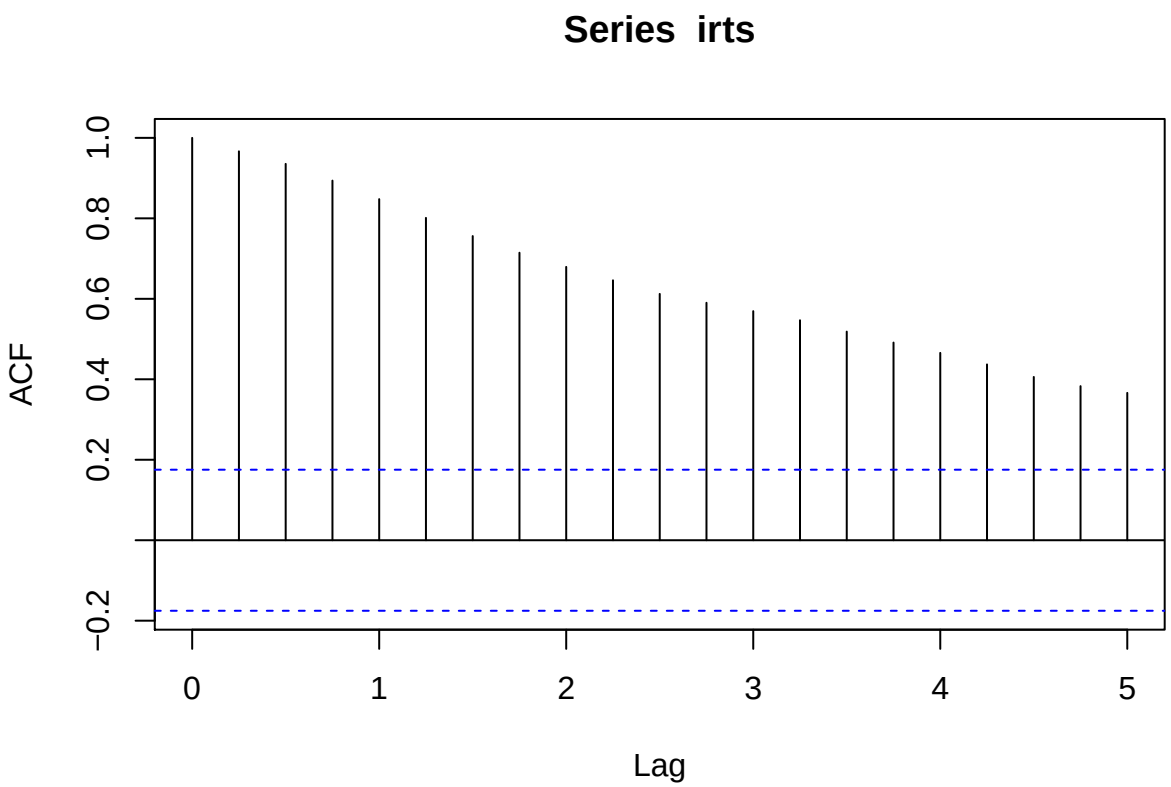
```
hw26<-read.xls('Econhw2data2.xls',sheet = 2) #import data

hpts<-ts(hw26$P,1980,2011,freq=4)  # time series of house price
irts<-ts(hw26$R..in...,1980,2011,freq=4) # times series of interest rate
diffhpts<-ts(diff(hw26$P),1980,2011,freq=4) # time series of difference in house price
diffirts<-ts(diff(hw26$R..in...),1980,2011,freq=4) # time series of change in interest rate

#acf & pacf of quarterly house price
acf(hpts)
```



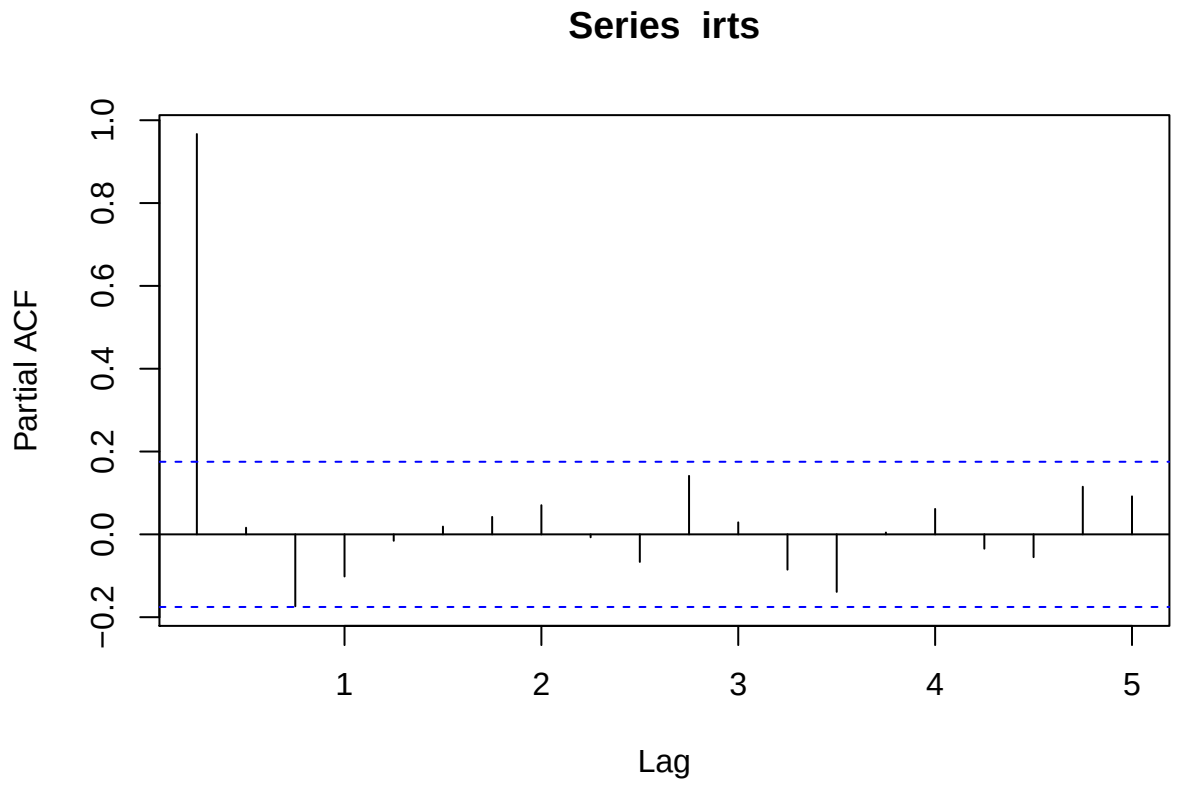

```
pacf(hpts)
```



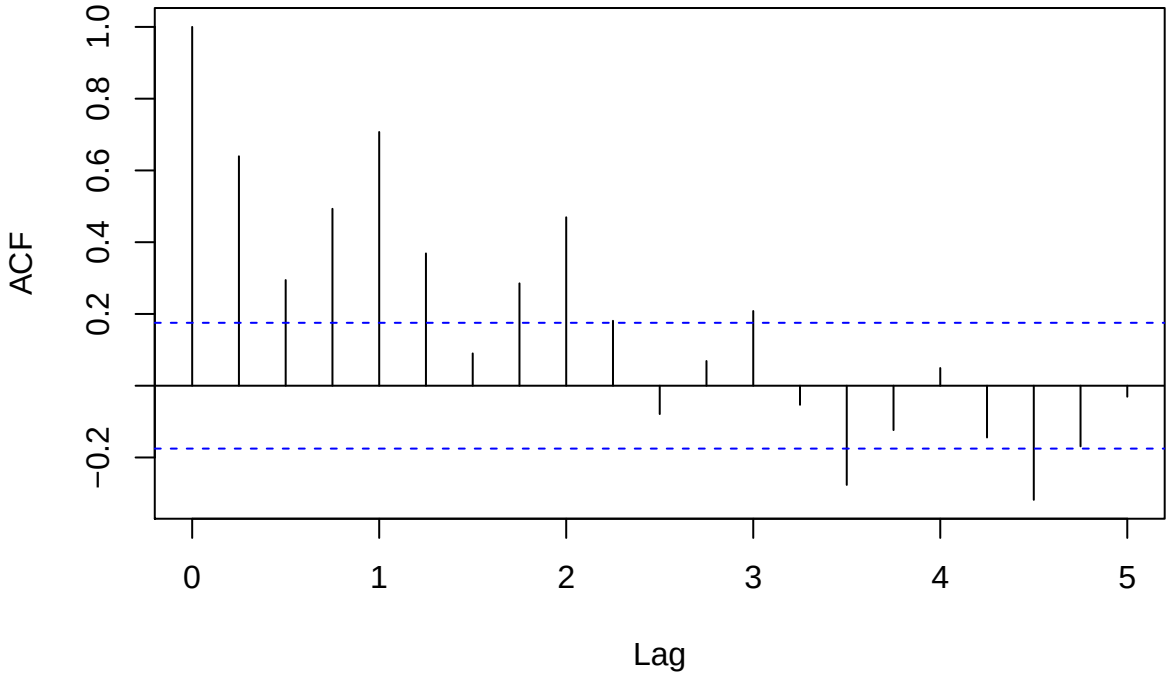

```
#acf & pacf of interest rate
acf(irts)
```



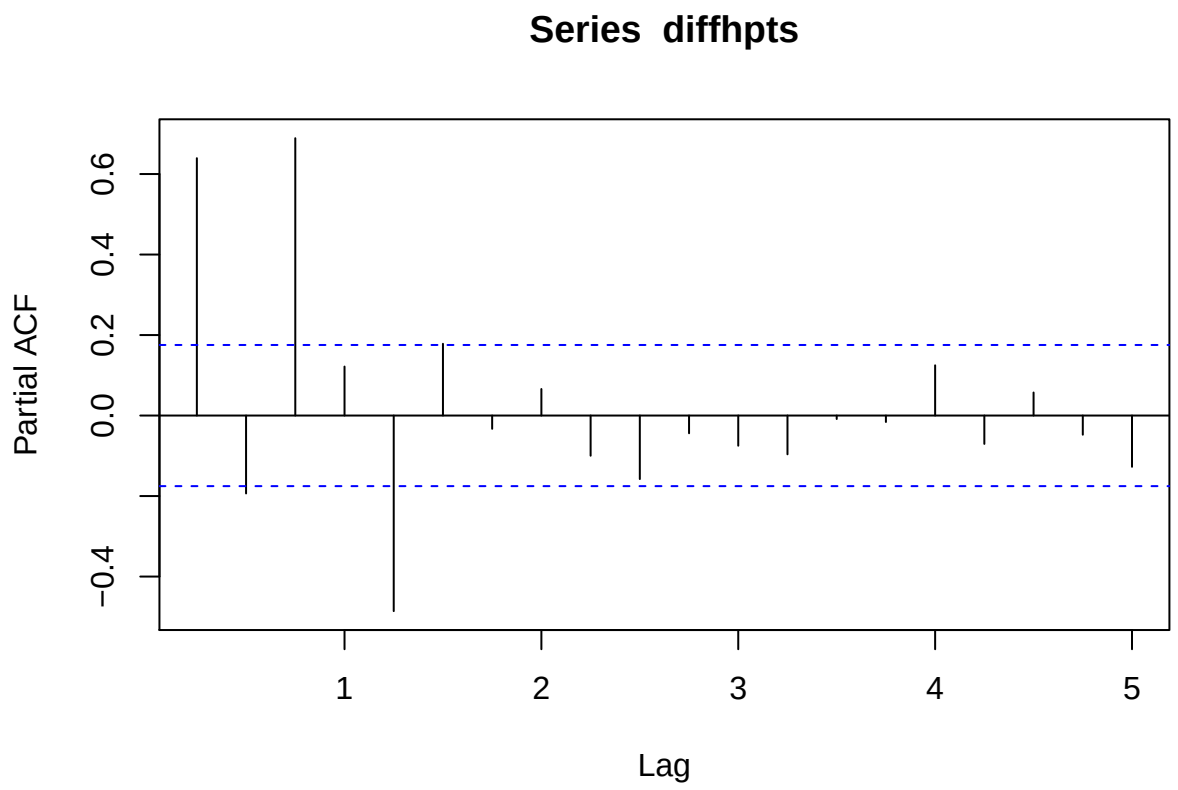
```
pacf(irts)
```



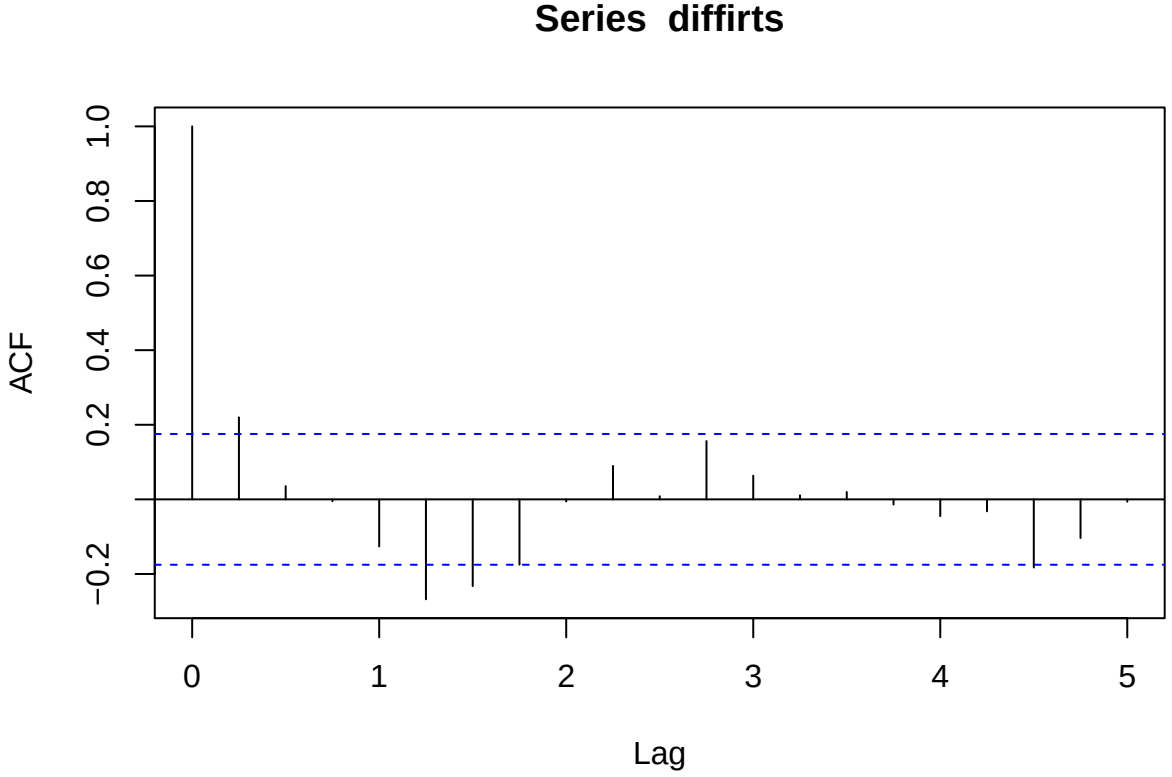

```
#acf & pacf of difference in house price
acf(diffhpts)
```



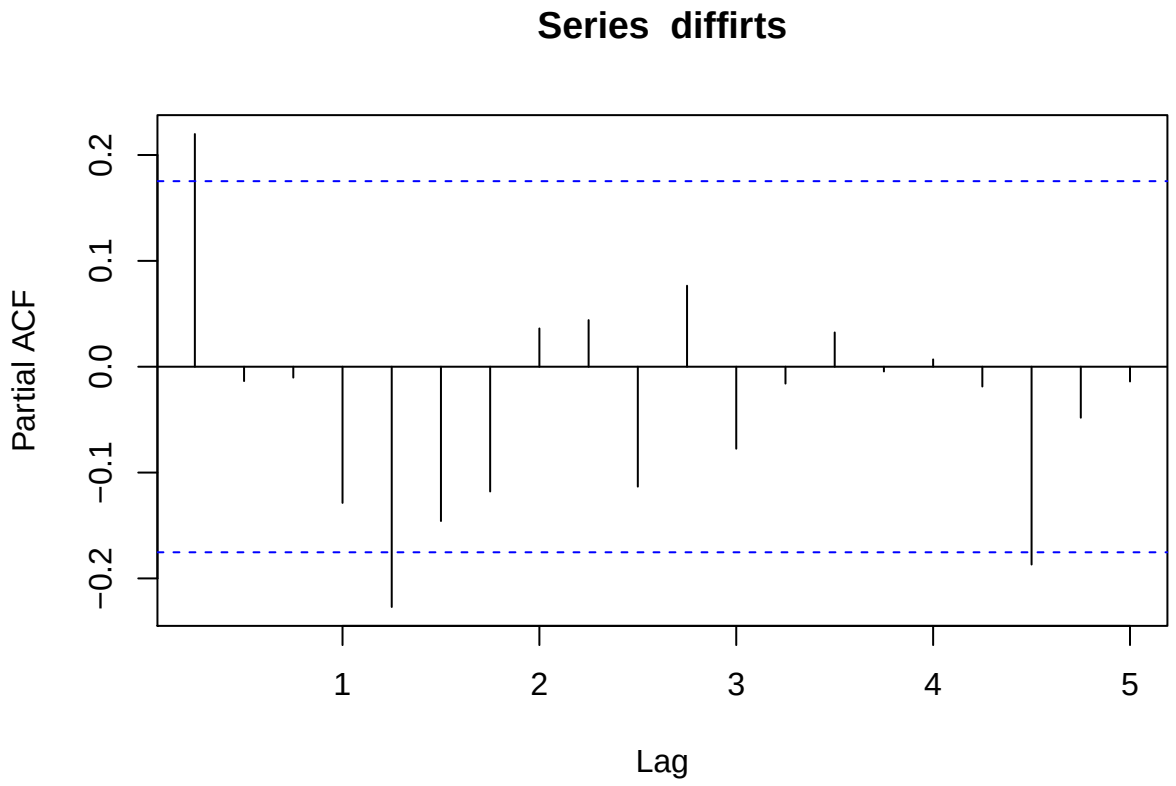
```
pacf(diffhpts)
```



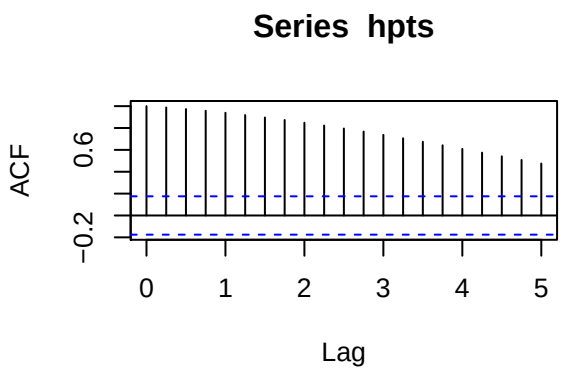

```
#acf & pacf of change in interest rate
acf(diffirts)
```

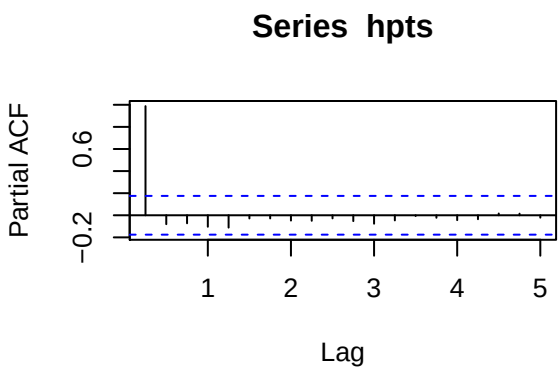


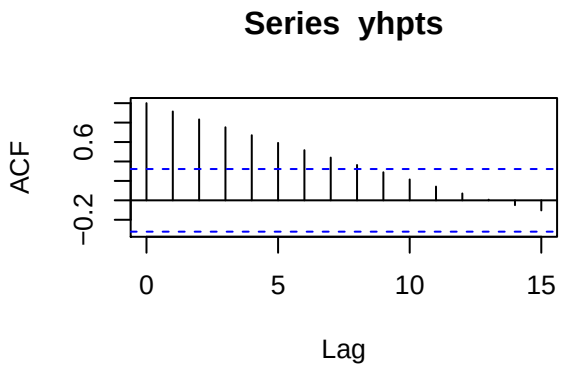
```
pacf(diffirts)
```

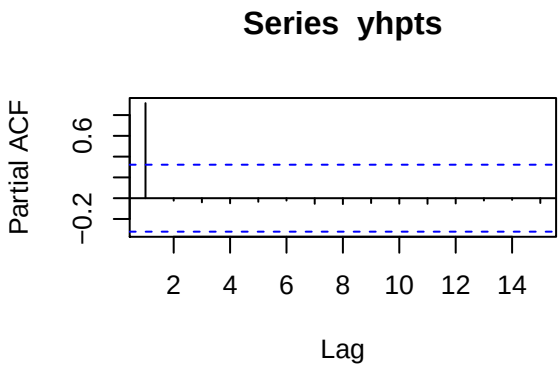



```
# comparig with annual
hw27<-read_xls('Econhw2data2.xls',sheet = 1) #import data

yhpts<-ts(hw26$P,1975,2011,freq=1)  # time series of house price
yirts<-ts(hw26$R..in...,1975,2011,freq=1) # times series of interest rate
ydiffhpts<-ts(diff(hw26$P),1975,2011,freq=1) # time series of difference in house price
ydiffirts<-ts(diff(hw26$R..in...),1975,2011,freq=1) # time series of change in interest rate

quartz()
par(mfrow=c(2,2))
acf(hpts)
pacf(hpts)
acf(yhpts)
pacf(yhpts)
```

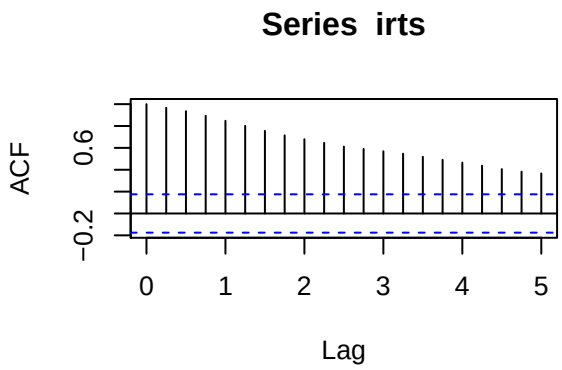


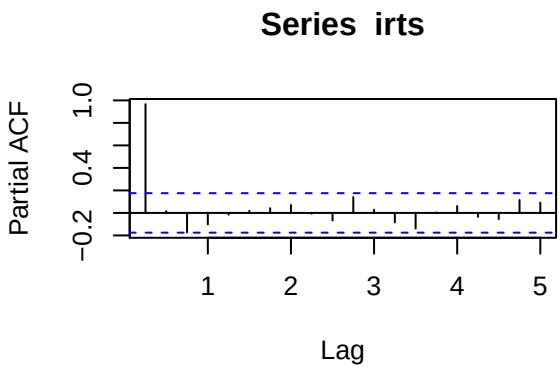


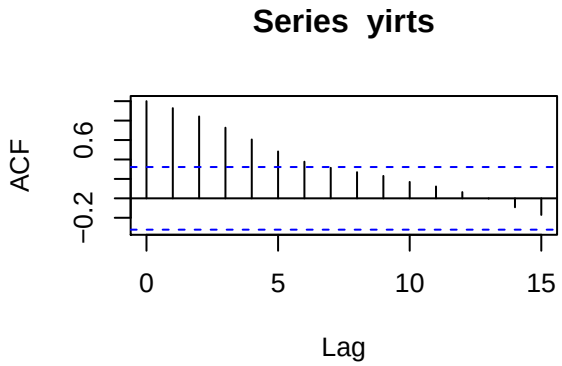


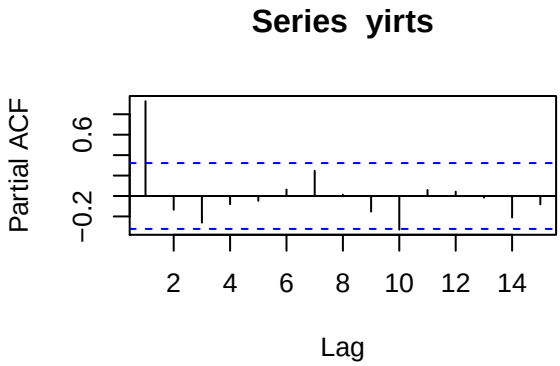



```
quartz()
par(mfrow=c(2,2))
acf(irts)
pacf(irts)
acf(yirts)
pacf(yirts)
```

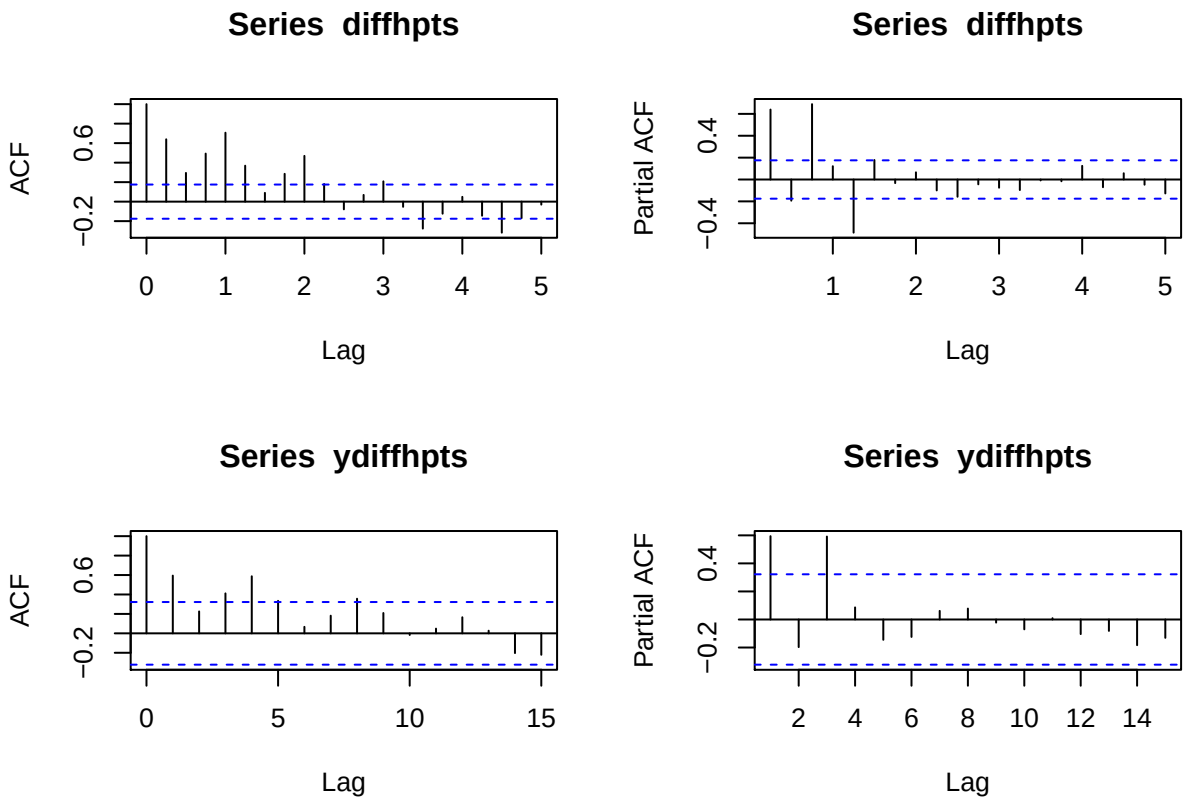




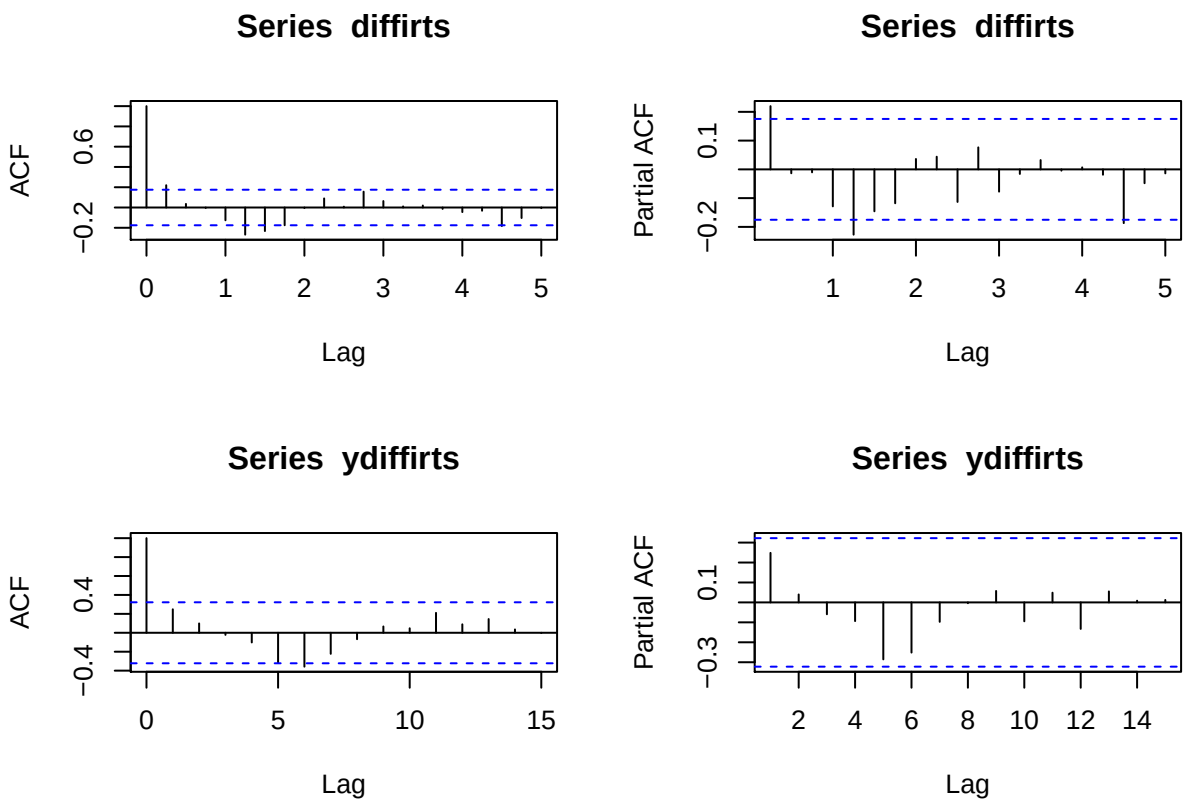





```
quartz()
par(mfrow=c(2,2))
acf(diffhpts)
pacf(diffhpts)
acf(ydiffhpts)
pacf(ydiffhpts)
```




```
quartz()
par(mfrow=c(2,2))
acf(diffirts)
pacf(diffirts)
acf(ydiffirts)
pacf(ydiffirts)
```



Summary: Acf of quarterly house price shows strong time dependence, since each lag reaches the significant level, which indicates a serial correlation. Acf of interest rate also shows strong time dependence, but not as strong as the acf of house price. It demonstrates a quicker decreasing trend compared to house price. The acf of change in house price does not show strong time dependency, it decays relatively significantly. The acf of change in interest rate does not show strong time depency, as after first two lags, no other lags reach the significant level. The yearly time series looks similar to the quarterly time series. In the annually house price and interest rate, the correlations between lags decay quicker as the number of lag increasing compared to the quarterly data.

```
# quaterly
x<-diffhpts
#model with one lag
modela=dynlm(x~L(x,1))
#plot(modela)
summary(modela)
```

```
##
## Time series regression with "ts" data:
## Start = 1980(2), End = 2011(1)
##
## Call:
## dynlm(formula = x ~ L(x, 1))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5254 -0.4331  0.0073  0.4639  5.0902
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.21766    0.14427   1.509    0.134
```

```
## L(x, 1)      0.64203    0.06965   9.218  1.1e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.527 on 122 degrees of freedom
## Multiple R-squared:  0.4106, Adjusted R-squared:  0.4057
## F-statistic: 84.97 on 1 and 122 DF,  p-value: 1.105e-15
```

```
#model with two lag
modelb=dynlm(x~L(x,1)+L(x,2))
#plot(modelb)
summary(modelb)
```

```
##
## Time series regression with "ts" data:
## Start = 1980(3), End = 2011(1)
##
## Call:
## dynlm(formula = x ~ L(x, 1) + L(x, 2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5115 -0.4247  0.0203  0.4915  4.3083
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.26448    0.14444   1.831   0.0696 .
## L(x, 1)      0.77169    0.08994   8.580 3.98e-14 ***
## L(x, 2)     -0.20130    0.08997  -2.237   0.0271 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.509 on 120 degrees of freedom
## Multiple R-squared:  0.4342, Adjusted R-squared:  0.4248
## F-statistic: 46.05 on 2 and 120 DF,  p-value: 1.441e-15
```

```
#model with three lag
modelc=dynlm(x~L(x,1)+L(x,2)+L(x,3))
#plot(modelc)
summary(modelc)
```

```
##
## Time series regression with "ts" data:
## Start = 1980(4), End = 2011(1)
##
## Call:
## dynlm(formula = x ~ L(x, 1) + L(x, 2) + L(x, 3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4846 -0.2904  0.0509  0.3729  2.8346
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02131    0.09877   0.216     0.83
## L(x, 1)      0.95101    0.06180  15.389   <2e-16 ***
```

```
## L(x, 2)     -0.82698    0.07872 -10.505   <2e-16 ***
## L(x, 3)      0.78569    0.06389  12.297   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.007 on 118 degrees of freedom
## Multiple R-squared:  0.752,  Adjusted R-squared:  0.7457
## F-statistic: 119.3 on 3 and 118 DF,  p-value: < 2.2e-16
```

```
#model with four lag
modeld=dynlm(x~L(x,1)+L(x,2)+L(x,3)+L(x,4))
#plot(modela)
summary(modeld)
```

```
##
## Time series regression with "ts" data:
## Start = 1981(1), End = 2011(1)
##
## Call:
## dynlm(formula = x ~ L(x, 1) + L(x, 2) + L(x, 3) + L(x, 4))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5127 -0.2659  0.0852  0.3825  2.7311
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01389    0.10043   0.138    0.890
## L(x, 1)      0.90457    0.09336   9.689  < 2e-16 ***
## L(x, 2)     -0.77624    0.10985  -7.066 1.27e-10 ***
## L(x, 3)      0.72458    0.11195   6.472 2.41e-09 ***
## L(x, 4)      0.06483    0.09716   0.667    0.506
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.014 on 116 degrees of freedom
## Multiple R-squared:  0.753,  Adjusted R-squared:  0.7444
## F-statistic: 88.39 on 4 and 116 DF,  p-value: < 2.2e-16
```

```
AIC(modela,modelb,modelc,modeld)
```

```
## Warning in AIC.default(modela, modelb, modelc, modeld): models are not all
## fitted to the same number of observations
```

```
##        df      AIC
## modela  3 460.9100
## modelb  4 455.1926
## modelc  5 353.9216
## modeld  6 353.6341
```

```
BIC(modela,modelb,modelc,modeld)
```

```
## Warning in BIC.default(modela, modelb, modelc, modeld): models are not all
## fitted to the same number of observations
```

```
##        df      BIC
## modela  3 469.3708
```

```
## modelb  4 466.4413
## modelc  5 367.9417
## modeld  6 370.4089
```

```
#rolling scheme
coeff<-list()
num=1
for(i in 40:length(x)-5)
{
ils <- subset(x,start=4+(i-39),end = 4+i ) # learning set
its <- subset(x,start = 5+i,end=length(x)) # testing set
mc<-dynlm(ils~L(ils,1)+L(ils,2)+L(ils,3))
coeff[[num]]<-coef(mc)
num=num+1
}
print(coeff)
```

```
## [[1]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1461034   0.8631132  -0.6944997   0.6471894
##
## [[2]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1503750   0.8704827  -0.6662766   0.5934725
##
## [[3]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1358691   0.8689151  -0.6467769   0.5832714
##
## [[4]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.07079536  0.94754884 -0.70359264  0.62554106
##
## [[5]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1101904   0.8918171  -0.6656395   0.6037032
##
## [[6]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.0715135   0.9075638  -0.6217810   0.5839917
##
## [[7]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.09780137  0.93475892 -0.68914556  0.60497289
##
## [[8]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1799490   0.8587212  -0.5946450   0.4981092
##
## [[9]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1667834   0.8771733  -0.6007884   0.5000549
##
## [[10]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
```

```
##   0.1410315   0.8797059  -0.5867000   0.5093192
##
## [[11]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1135450   0.8378510  -0.5528821   0.5489867
##
## [[12]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.07284724  0.82839645 -0.53716585  0.57906169
##
## [[13]]
##  (Intercept)    L(ils, 1)    L(ils, 2)    L(ils, 3)
##  0.002568826  0.887059821 -0.581074756  0.639853150
##
## [[14]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1343724   0.6431173  -0.4684610   0.6190020
##
## [[15]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.09757112  0.56214427 -0.30642095  0.56820801
##
## [[16]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.09254532  0.55331967 -0.29867145  0.57060202
##
## [[17]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.08666605  0.58352241 -0.29559453  0.54858365
##
## [[18]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1475751   0.5829240  -0.4122558   0.6014154
##
## [[19]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1485315   0.5790714  -0.4186071   0.6134894
##
## [[20]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1216849   0.5802967  -0.4629500   0.6682589
##
## [[21]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1106413   0.6525128  -0.4775386   0.6126135
##
## [[22]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.09961644  0.66179621 -0.47998791  0.60792482
##
## [[23]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1139807   0.6570528  -0.4713098   0.5708307
##
```

```
## [[24]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.08965039  0.63442194 -0.47649579  0.60418981
##
## [[25]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.09601921  0.62232863 -0.47963993  0.60599510
##
## [[26]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1596547   0.6189291  -0.5923781   0.6403610
##
## [[27]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1620430   0.5841191  -0.5697014   0.6297001
##
## [[28]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1682704   0.5676847  -0.5722888   0.6246503
##
## [[29]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1626727   0.6018714  -0.5746802   0.5857568
##
## [[30]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1946065   0.6028433  -0.6421760   0.6168731
##
## [[31]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1985592   0.5567352  -0.6153609   0.5915065
##
## [[32]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2171586   0.5149569  -0.5551050   0.5435930
##
## [[33]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2341511   0.4967942  -0.5737746   0.5053715
##
## [[34]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2772826   0.5086032  -0.6341269   0.5238007
##
## [[35]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2864787   0.5153197  -0.6441805   0.4932527
##
## [[36]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2870760   0.4897138  -0.5782490   0.4502923
##
## [[37]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
```

```
##   0.2672908   0.4986618  -0.5722498   0.4930081
##
## [[38]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2106097   0.6037564  -0.6125422   0.6180833
##
## [[39]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1546251   0.7167767  -0.6104036   0.6839244
##
## [[40]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1628477   0.7177205  -0.6176843   0.6919738
##
## [[41]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1647374   0.7147465  -0.5876596   0.7353422
##
## [[42]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1474172   0.7450635  -0.6103907   0.7878576
##
## [[43]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1723031   0.7324526  -0.6288078   0.7898409
##
## [[44]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1819253   0.7296180  -0.6530558   0.7886808
##
## [[45]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1933076   0.7409413  -0.6659992   0.7334948
##
## [[46]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1871058   0.7659438  -0.7081128   0.7986868
##
## [[47]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2027835   0.7580695  -0.7075737   0.7925169
##
## [[48]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1928913   0.7621806  -0.7065963   0.7955300
##
## [[49]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2077103   0.7514433  -0.7036940   0.8127125
##
## [[50]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2425674   0.7698687  -0.7458800   0.8464735
##
```

```
## [[51]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1285926   0.8710234  -0.7602167   0.8668732
##
## [[52]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1850347   0.8925183  -0.8151376   0.8707837
##
## [[53]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2000442   0.9032383  -0.8428681   0.8469523
##
## [[54]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1946265   0.9091632  -0.8588898   0.8790538
##
## [[55]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1349927   0.9403811  -0.8556138   0.8886518
##
## [[56]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1625208   0.9517431  -0.8666080   0.8767187
##
## [[57]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1447623   0.9544371  -0.8651283   0.8879840
##
## [[58]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  0.07333053  0.99125506 -0.93391132  1.02674395
##
## [[59]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1854752   0.9076906  -0.9061604   1.0042451
##
## [[60]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1695857   0.9097277  -0.8917304   0.9999464
##
## [[61]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2343769   0.9055868  -0.8946846   0.9690329
##
## [[62]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2110294   0.9178540  -0.9116385   0.9910799
##
## [[63]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1633794   0.9150959  -0.8945930   0.9923949
##
## [[64]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
```

```
##   0.2242524   0.9189346  -0.9136331   0.9812995
##
## [[65]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.3495509   0.9591539  -0.9606129   0.8907766
##
## [[66]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.4338589   0.9336562  -0.9337190   0.8479364
##
## [[67]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.4267915   0.8228579  -0.8089413   0.7939893
##
## [[68]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.3788774   0.8930976  -0.9046009   0.8331279
##
## [[69]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.2078087   0.9530662  -0.9127633   0.8431487
##
## [[70]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##   0.1994272   0.9428477  -0.8908077   0.8371448
##
## [[71]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.1811495   0.9703785  -0.8854135   0.9465509
##
## [[72]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.4285734   1.0721686  -0.9748520   1.0340314
##
## [[73]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.8140373   1.2083795  -0.9827547   1.0566833
##
## [[74]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.4536169   1.1657974  -1.0744887   1.0309342
##
## [[75]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.5638574   1.1545700  -1.0483338   1.0546205
##
## [[76]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.5209296   1.1333016  -1.0173373   1.0201238
##
## [[77]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.4241339   1.0638638  -0.9808195   1.0051003
##
```

```
## [[78]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.2983659   1.0708779  -1.0899674   1.0435722
##
## [[79]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.2201977   1.1046313  -1.1214143   1.0103208
##
## [[80]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
## -0.03786208  1.07058945 -0.98057896  0.84342777
##
## [[81]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
## -0.02716034  1.06325682 -0.97462825  0.84189743
##
## [[82]]
##  (Intercept)    L(ils, 1)    L(ils, 2)    L(ils, 3)
##  0.001206577  1.061534131 -0.983833052  0.851432223
##
## [[83]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.1626422   1.0009220  -0.9093011   0.8505327
##
## [[84]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
## -0.09840737  0.96640421 -0.85990992  0.81648624
##
## [[85]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
##  -0.1817815   0.9878031  -0.8349280   0.7871369
##
## [[86]]
## (Intercept)   L(ils, 1)   L(ils, 2)   L(ils, 3)
## -0.09905326  0.98479850 -0.86588820  0.77878696
```

```
# #recursive scheme
n=1
coeff2<-list()
ls<-subset(x,start=4,end=40)
for(i in 4:89)
{
  ts<-subset(x,start=37+i,end=length(x))
  mc2<-dynlm(ls~Lag(ls,1)+Lag(ls,2)+Lag(ls,3))
  coeff2[[n]]<-coef(mc2)
  p<-coef(mc2)[1]+coef(mc2)[2]*x[36+i]+coef(mc2)[3]*x[37+i]+coef(mc2)[4]*x[38+i]
  ls<-c(ls,p)
  n=n+1
}
print(coeff2) # print coefficient in each regression step
```

```
## [[1]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1906430   0.8466609  -0.6480186   0.5675656
##
```

```
## [[2]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1615585   0.8449595  -0.6117034   0.5488941
##
## [[3]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.2170567   0.7715740  -0.5586927   0.5129589
##
## [[4]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1771227   0.6826714  -0.3818317   0.4479190
##
## [[5]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1792472   0.6798653  -0.3786420   0.4451952
##
## [[6]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1761944   0.6818279  -0.3746716   0.4423586
##
## [[7]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1635556   0.6783083  -0.3649248   0.4497680
##
## [[8]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1639817   0.6780934  -0.3646240   0.4492235
##
## [[9]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1738755   0.6709934  -0.3649076   0.4469327
##
## [[10]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1545429   0.6622930  -0.3430189   0.4507595
##
## [[11]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1404014   0.6764644  -0.3573471   0.4655565
##
## [[12]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1767352   0.6396542  -0.3685345   0.4757841
##
## [[13]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1337175   0.5200077  -0.2184543   0.4768956
##
## [[14]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1743685   0.4409958  -0.1239410   0.4133067
##
## [[15]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
```

```
##   0.1638489   0.4359159  -0.0796899   0.3832608
##
## [[16]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.19297010  0.43873693 -0.06991078  0.33726311
##
## [[17]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.18706434  0.43625069 -0.06843828  0.34258106
##
## [[18]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.19536872  0.42825515 -0.06159351  0.33749758
##
## [[19]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.19222675  0.42395290 -0.05427589  0.33512723
##
## [[20]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.18575455  0.42743476 -0.06253918  0.34390873
##
## [[21]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1800145   0.4358170  -0.0628679   0.3396403
##
## [[22]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.17135321  0.43974740 -0.05599549  0.33686874
##
## [[23]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.17477456  0.43804622 -0.05631768  0.33521569
##
## [[24]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1806417   0.4384381  -0.0601842   0.3328817
##
## [[25]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.17550057  0.43615682 -0.05934976  0.33862178
##
## [[26]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.18507145  0.42616275 -0.05096078  0.33241387
##
## [[27]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1830111   0.4215643  -0.0444555   0.3306478
##
## [[28]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.18463288  0.42069673 -0.04140605  0.32758539
##
```

```
## [[29]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.18106733  0.42169211 -0.03716289  0.32087571
##
## [[30]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.19101890  0.40969390 -0.03184416  0.32043985
##
## [[31]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.18951638  0.40308967 -0.02485384  0.31947367
##
## [[32]]
##  (Intercept)   Lag(ls, 1)   Lag(ls, 2)   Lag(ls, 3)
##  0.196414746  0.397484593 -0.001720187  0.299950769
##
## [[33]]
##   (Intercept)    Lag(ls, 1)    Lag(ls, 2)    Lag(ls, 3)
##  0.1965519454  0.3966300806 -0.0009073328  0.2992328339
##
## [[34]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.19675653  0.40010674  0.01865701  0.28764284
##
## [[35]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.19391511  0.40787631  0.01578446  0.29027363
##
## [[36]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.18884032  0.41288907  0.02763544  0.28658720
##
## [[37]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.17883579  0.42026053  0.03038824  0.29905187
##
## [[38]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.16322405  0.44216109  0.03364779  0.30786084
##
## [[39]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.14381195  0.46807825  0.04382598  0.31080938
##
## [[40]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.13562171  0.47602908  0.04878982  0.31342355
##
## [[41]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1230565   0.4811552   0.0587928   0.3218196
##
## [[42]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
```

```
##  0.10567681  0.49316787  0.06152383  0.33906843
##
## [[43]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.1168095   0.4818367   0.0577595   0.3341170
##
## [[44]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.11617164  0.48159243  0.05872419  0.33458245
##
## [[45]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.11813236  0.48098406  0.05956203  0.33069431
##
## [[46]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.10517868  0.48989745  0.06743685  0.33940997
##
## [[47]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.09725968  0.50271919  0.06564279  0.34279121
##
## [[48]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.09176985  0.50713248  0.07197705  0.34175522
##
## [[49]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.07128000  0.51448221  0.08197854  0.35978085
##
## [[50]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.04157207  0.56045425  0.06969253  0.37455955
##
## [[51]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.04112554  0.56117761  0.06990027  0.37431245
##
## [[52]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.04889957  0.56242652  0.05827288  0.37271588
##
## [[53]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.05048264  0.56280373  0.05820931  0.36993947
##
## [[54]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.03292789  0.57420938  0.05425885  0.38966794
##
## [[55]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.02116101  0.59286958  0.05266562  0.38984939
##
```

```
## [[56]]
##  (Intercept)   Lag(ls, 1)   Lag(ls, 2)   Lag(ls, 3)
## -0.002907555  0.611068297  0.076502711  0.382151386
##
## [[57]]
##  (Intercept)   Lag(ls, 1)   Lag(ls, 2)   Lag(ls, 3)
## -0.007049155  0.616164583  0.075426009  0.383753691
##
## [[58]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
## -0.04234985  0.62783453  0.11555423  0.37935277
##
## [[59]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
## -0.03246492  0.61069606  0.12522958  0.37445436
##
## [[60]]
##  (Intercept)   Lag(ls, 1)   Lag(ls, 2)   Lag(ls, 3)
## -0.004505767  0.620779170  0.062528985  0.391589909
##
## [[61]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
## 0.007161272 0.639290502 0.054890516 0.364777838
##
## [[62]]
##  (Intercept)   Lag(ls, 1)   Lag(ls, 2)   Lag(ls, 3)
## -0.007458503  0.642447971  0.033171411  0.402781641
##
## [[63]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.01934545  0.57721974  0.05464711  0.41531515
##
## [[64]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.05470235  0.69451079 -0.11721416  0.42159895
##
## [[65]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.10066415  0.92612897 -0.09436008  0.09060182
##
## [[66]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.10026878  0.72692081 -0.04793779  0.26462811
##
## [[67]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.11001154  0.52602900  0.33864970  0.05387763
##
## [[68]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.08566255  0.69069665 -0.00266546  0.23223658
##
## [[69]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
```

```
##  0.08003078  0.72665568  0.01227823  0.18244437
##
## [[70]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.09632063  0.73047700 -0.03086744  0.21317439
##
## [[71]]
##  (Intercept)   Lag(ls, 1)   Lag(ls, 2)   Lag(ls, 3)
##  0.033920504  0.664008427 -0.007214665  0.290088821
##
## [[72]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
## -0.03291313  0.82676331 -0.18466675  0.34579828
##
## [[73]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
## -0.09559553  0.96384880 -0.12524646  0.19242690
##
## [[74]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.01149345  0.86588221 -0.17977934  0.26269464
##
## [[75]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.01247739  0.86716635 -0.18173410  0.26266161
##
## [[76]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
## -0.07082348  0.86681768 -0.29138641  0.43224532
##
## [[77]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
## -0.07588234  0.87737533 -0.30037312  0.43447826
##
## [[78]]
##  (Intercept)   Lag(ls, 1)   Lag(ls, 2)   Lag(ls, 3)
##  0.003788583  0.854712583 -0.462841308  0.558692158
##
## [[79]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.04360374  0.93795806 -0.54415577  0.53001221
##
## [[80]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.0771740   0.9520678  -0.4700112   0.4214606
##
## [[81]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.03873472  0.98701807 -0.54765672  0.47425276
##
## [[82]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##   0.0954083   0.8159236  -0.4243445   0.4921644
##
```

```
## [[83]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.05823167  0.66792148 -0.13097103  0.35148861
##
## [[84]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.06945201  0.65554862 -0.09944626  0.32522395
##
## [[85]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.04914454  0.66796990 -0.06826362  0.28993800
##
## [[86]]
## (Intercept)  Lag(ls, 1)  Lag(ls, 2)  Lag(ls, 3)
##  0.04914454  0.66796990 -0.06826362  0.28993800
```

By comparing Adjusted-Rsquared, AIC and BIC, I am interested in using model c, that is with 3 lags.

```
hw28<-read_xls('Econhw2data2.xls',sheet = 3) #import data

difpred <-hw28$`Actual RGDP Quarterly Growth (in %)`-hw28$`Greenbook RGDP Quarterly Growth Forecast (in
mean(difpred)
```

```
## [1] 0.07746318
```

```
# run on 5 lags
t=6:152
modelf=dynlm(difpred[t]~L(difpred[t],1)+L(difpred[t],2)+L(difpred[t],3)+L(difpred[t],4)+L(difpred[t],5)
summary(modelf)
```

```
## Warning in summary.lm(modelf): essentially perfect fit: summary may be
## unreliable
```

```
##
## Time series regression with "numeric" data:
## Start = 1, End = 147
##
## Call:
## dynlm(formula = difpred[t] ~ L(difpred[t], 1) + L(difpred[t],
##     2) + L(difpred[t], 3) + L(difpred[t], 4) + L(difpred[t],
##     5))
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -3.041e-15 -4.100e-19  2.337e-17  4.034e-17  3.226e-16
##
## Coefficients: (4 not defined because of singularities)
##                     Estimate Std. Error    t value Pr(>|t|)
## (Intercept)      -4.578e-17  2.161e-17 -2.118e+00   0.0358 *
## L(difpred[t], 1)  1.000e+00  2.862e-17  3.494e+16   <2e-16 ***
## L(difpred[t], 2)         NA         NA         NA       NA
## L(difpred[t], 3)         NA         NA         NA       NA
## L(difpred[t], 4)         NA         NA         NA       NA
## L(difpred[t], 5)         NA         NA         NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 2.608e-16 on 145 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.221e+33 on 1 and 145 DF,  p-value: < 2.2e-16
```

Summary: The expected value of the forecast errors does not equal to zero, but near 0. The expectation of forecast error should be 0, otherwise this model is not appropriate. In this case, the 1-quarter-ahead forecasts of real gdp growth is not an optimal model to do the forecasting. For 5 lags regression, it shows a perfect fit on intercept and lag1, lag2 to lag5 are not useful in this regression. The p value of F-statistic is less than 2.2*10^-16. Hence the null hypothsis that beta_0 to beta_5 is zero can be rejected.

C Chapter

##6.2

```
library(Ecdat)
```

```
## Loading required package: Ecfun
```

```
## Warning: package 'Ecfun' was built under R version 3.5.2
```

```
##
## Attaching package: 'Ecfun'
```

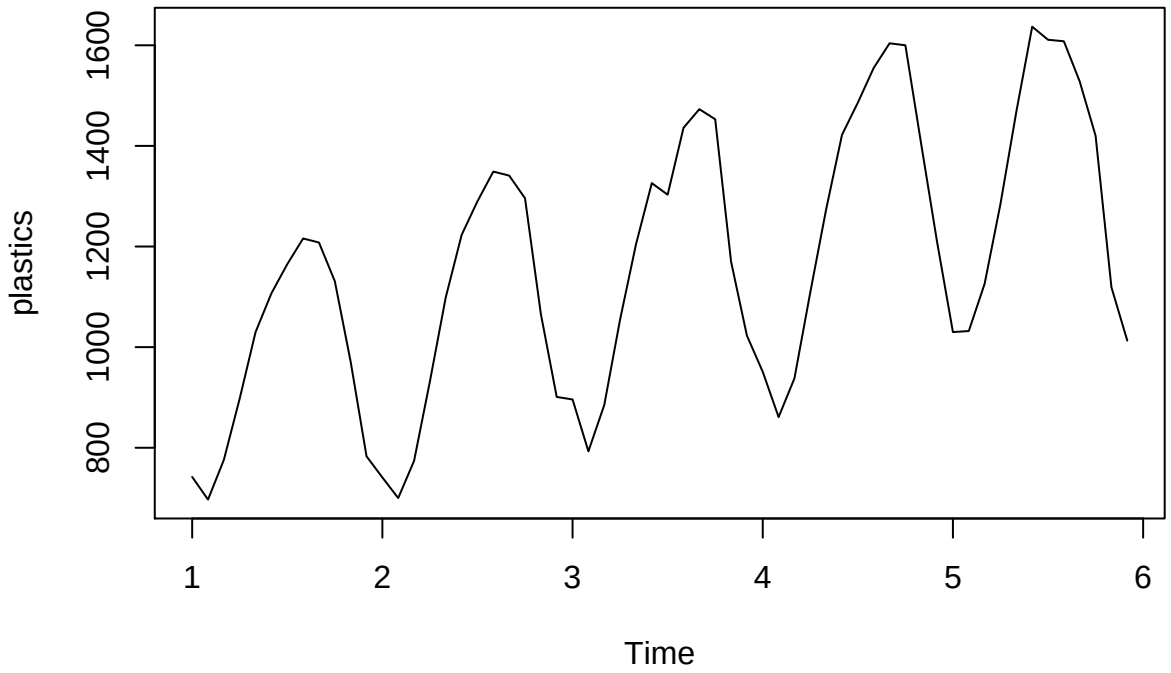
```
## The following object is masked from 'package:forecast':
##
##     BoxCox
```

```
## The following object is masked from 'package:base':
##
##     sign
```

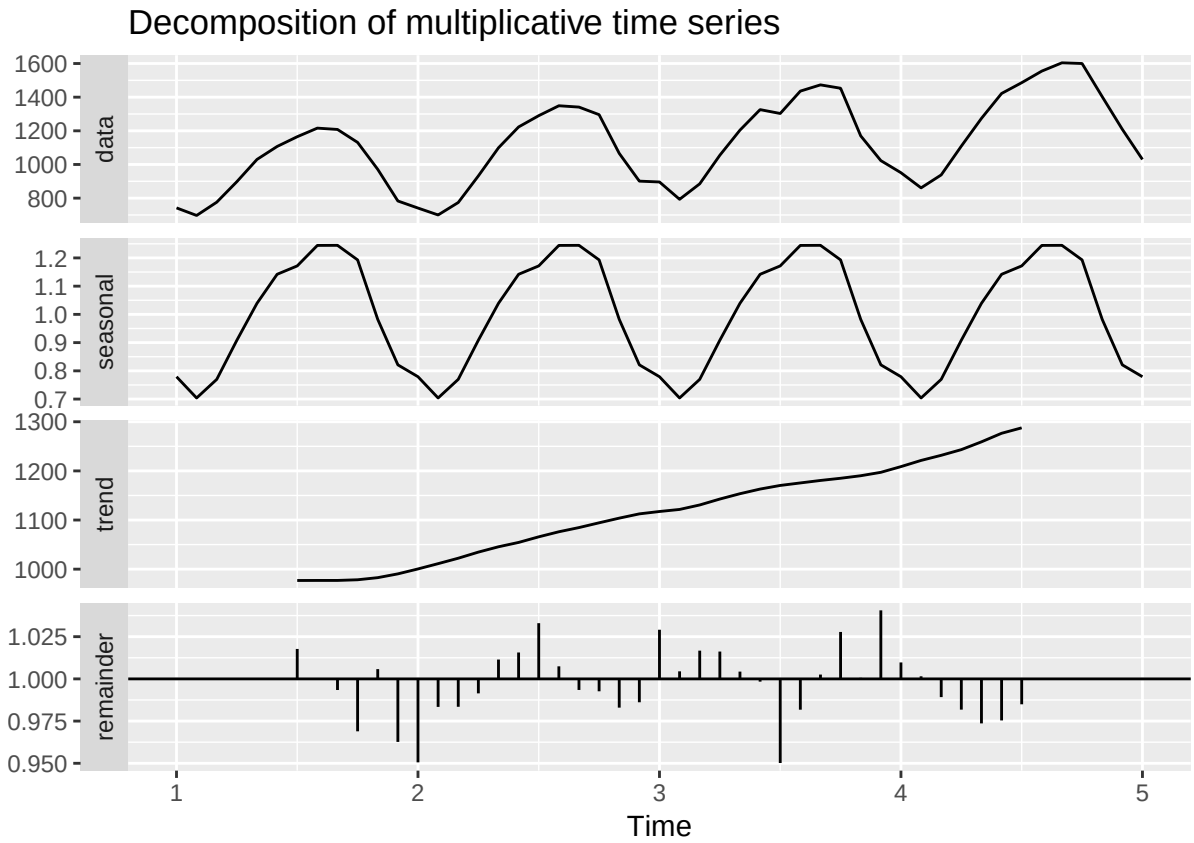
```
##
## Attaching package: 'Ecdat'
```

```
## The following object is masked from 'package:datasets':
##
##     Orange
```

```
# a
plot(plastics)
```




```
# b
plasticts<-ts(plastics,1,5,frequency = 12)
plasticts %>% decompose(type="multiplicative") %>%autoplot()
```



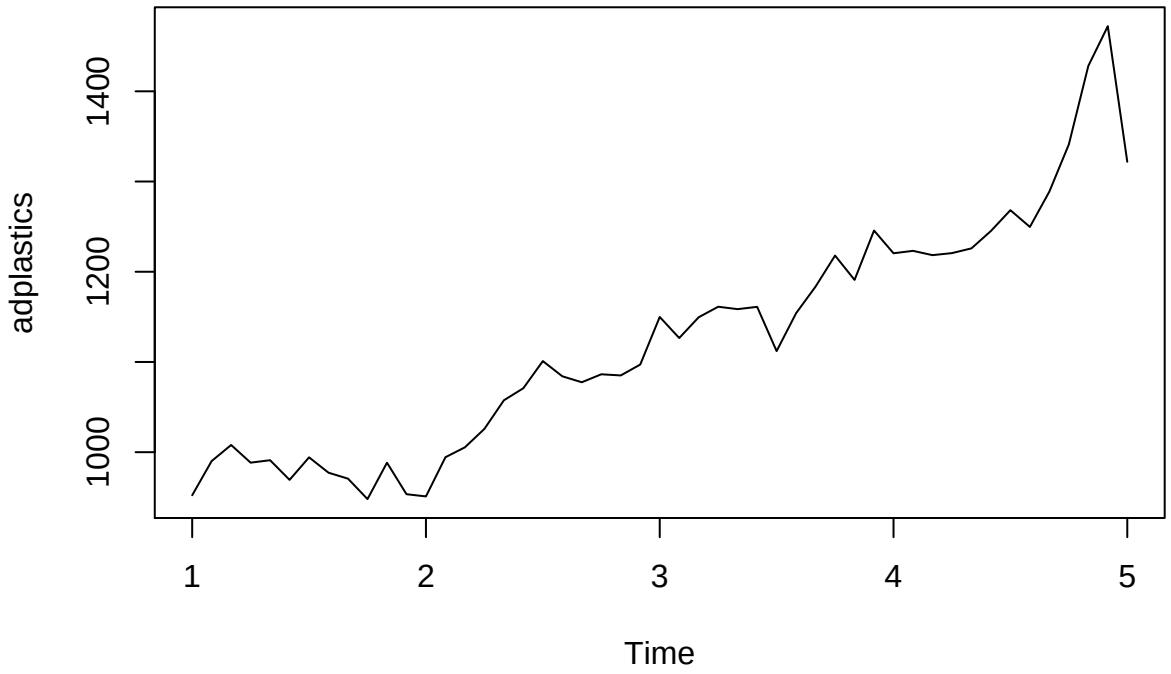

```
splastics<-decompose(plasticts,"multiplicative")
stlplastic <- stl(plasticts, s.window = "periodic")
```

```
stlplastic # seasonal indices
```

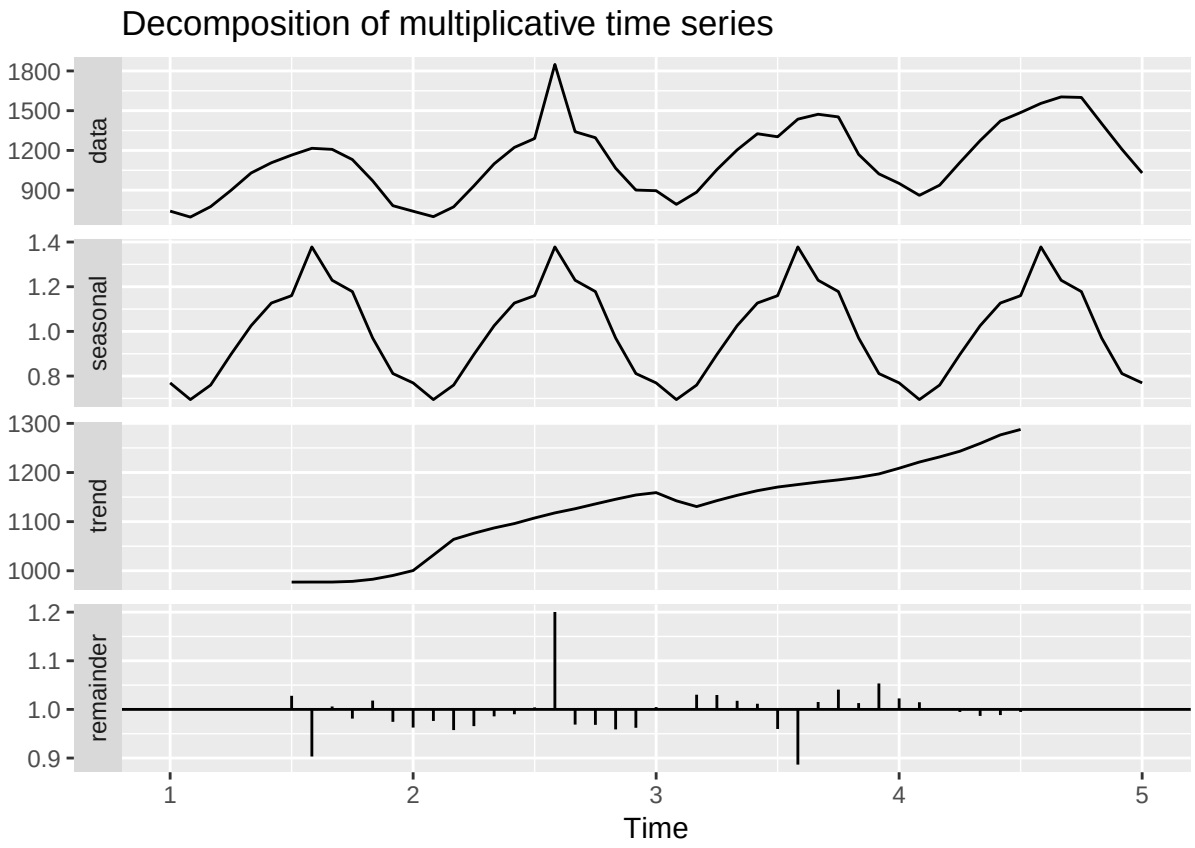
```
##  Call:
##  stl(x = plasticts, s.window = "periodic")
##
## Components
##          seasonal     trend   remainder
## Jan 1 -267.401536 1008.6525   0.7489974
## Feb 1 -328.779407 1003.4665  22.3129389
## Mar 1 -254.910287  998.2804  32.6298894
## Apr 1 -107.081843  994.0930  10.9888880
## May 1   38.746597  989.9055   1.3478902
## Jun 1  150.562280  986.7182 -30.2805216
## Jul 1  186.127941  983.5310  -4.6589110
## Aug 1  259.038125  980.8859 -23.9240554
## Sep 1  271.448290  978.2409 -41.6891809
## Oct 1  228.114417  979.6957 -76.8101479
## Nov 1    3.780506  981.1506 -13.9310760
## Dec 1 -179.644999  990.3793 -27.7343497
## Jan 2 -267.401536  999.6081   8.7934080
## Feb 2 -328.779407 1011.8558  16.9235699
## Mar 2 -254.910287 1024.1035   4.8067409
## Apr 2 -107.081843 1035.7099   3.3719315
## May 2   38.746597 1047.3163  12.9371256
## Jun 2  150.562280 1056.9442  15.4935119
## Jul 2  186.127941 1066.5721  37.2999206
## Aug 2  259.038125 1075.3960  14.5659196
## Sep 2  271.448290 1084.2198 -14.6680623
## Oct 2  228.114417 1092.8688 -24.9831943
## Nov 2    3.780506 1101.5178 -39.2982873
## Dec 2 -179.644999 1109.1337 -28.4887097
## Jan 3 -267.401536 1116.7496  46.6518993
## Feb 3 -328.779407 1125.2394  -3.4600208
## Mar 3 -254.910287 1133.7292   6.1810680
## Apr 3 -107.081843 1143.4251  18.6567381
## May 3   38.746597 1153.1210  12.1324118
## Jun 3  150.562280 1161.2976  14.1401009
## Jul 3  186.127941 1169.4742 -52.6021877
## Aug 3  259.038125 1175.0520   1.9098819
## Sep 3  271.448290 1180.6297  20.9219705
## Oct 3  228.114417 1186.2888  38.5967548
## Nov 3    3.780506 1191.9479 -25.7284219
## Dec 3 -179.644999 1199.9288   2.7161509
## Jan 4 -267.401536 1207.9098  10.4917551
## Feb 4 -328.779407 1218.7223 -28.9428455
## Mar 4 -254.910287 1229.5347 -36.6244370
## Apr 4 -107.081843 1244.4991 -28.4172294
## May 4   38.746597 1259.4634 -24.2100181
## Jun 4  150.562280 1274.3761  -2.9384245
## Jul 4  186.127941 1289.2889  10.5831916
## Aug 4  259.038125 1304.0947  -8.1327889
## Sep 4  271.448290 1318.9005  13.6512496
## Oct 4  228.114417 1334.1942  37.6913688
## Nov 4    3.780506 1349.4880  49.7315270
```

```
## Dec 4 -179.644999 1365.0308  23.6141821
## Jan 5 -267.401536 1380.5737 -83.1721314
```

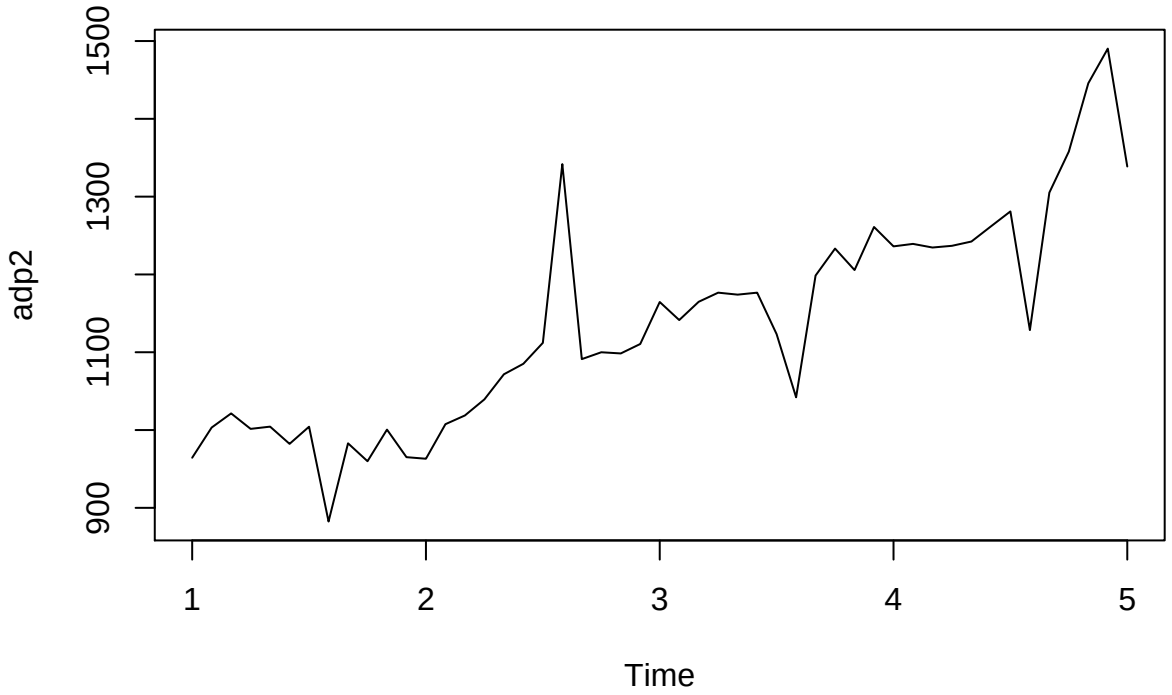
```r
# d
adplastics <- plasticts/splastics$seasonal
plot(adplastics) # adjust plastics data
```



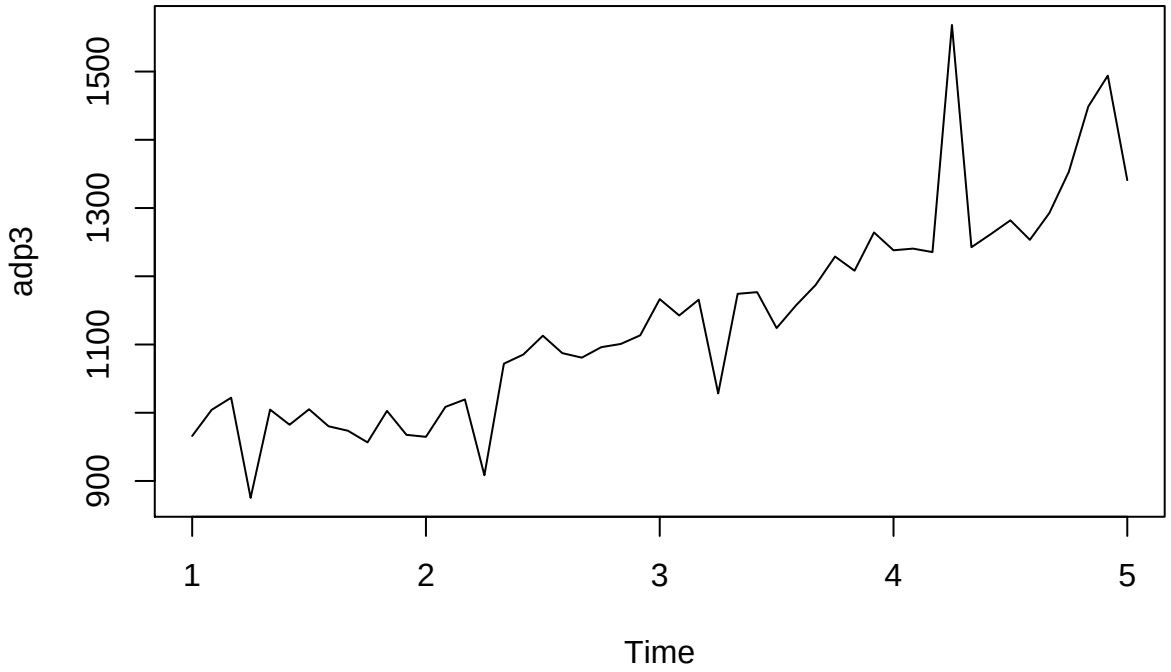

```r
#e
p2<-plastics
p2[20]<-plastics[20]+500
p2ts<-ts(p2,1,5,frequency = 12)
p2ts %>% decompose(type="multiplicative") %>%autoplot()
```




```
psd<-decompose(p2ts,type="multiplicative")
adp2<-seasadj(psd)
plot(adp2)
```

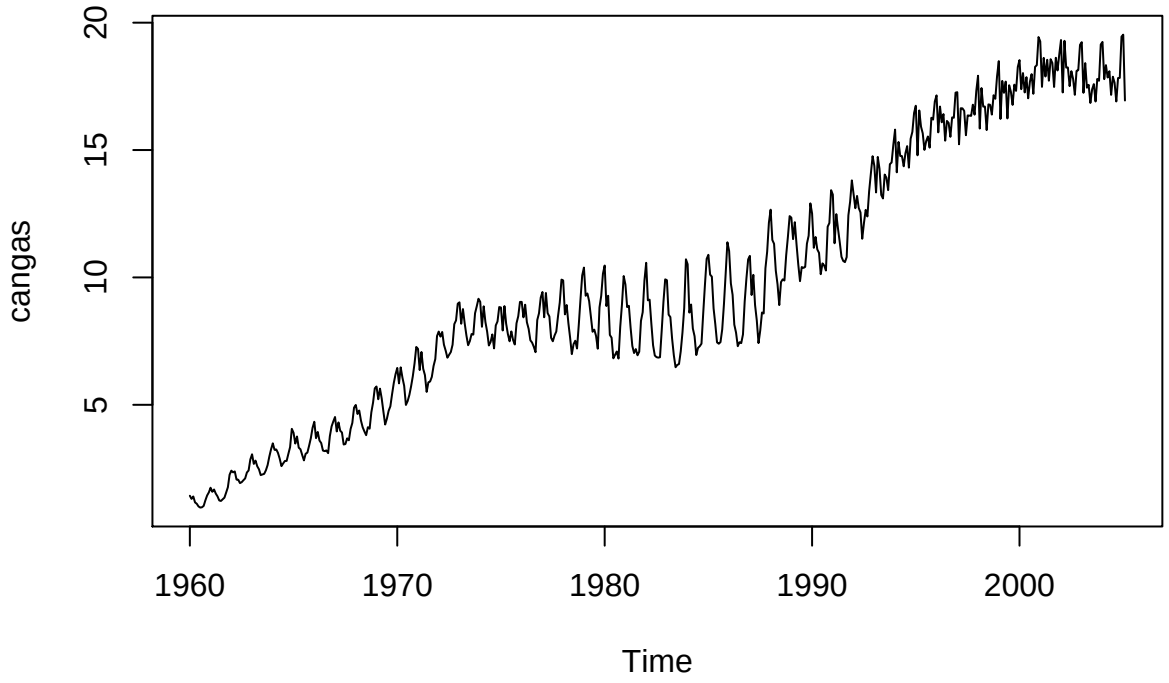


```r
#f
p3<-plastics
p3[40]<-plastics[40]+500
p3ts<-ts(p3,1,5,frequency = 12)
psd3 <- decompose(p3ts, type='multiplicative')
adp3 <- seasadj(psd3)
plot(adp3)
```

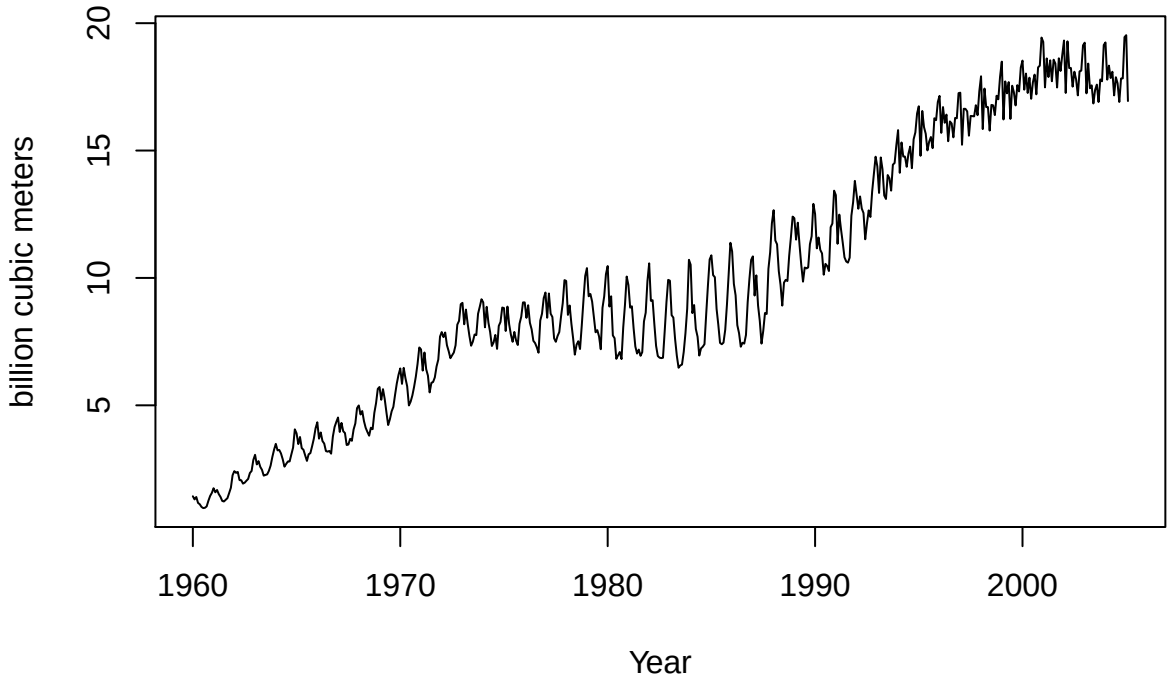


Sumamry: By observation, this time series has a seasonal cycle for each year in the plot. The result from part b supprt the result from part a. In part b, it has a upper trend with a yearly In e, the new outlier drives the seasonal adjusted time series to have a spike at the point we add 500. In f, no matter where the outlier exists, it will always generate similar spike but at different location, while other parts give the same result.

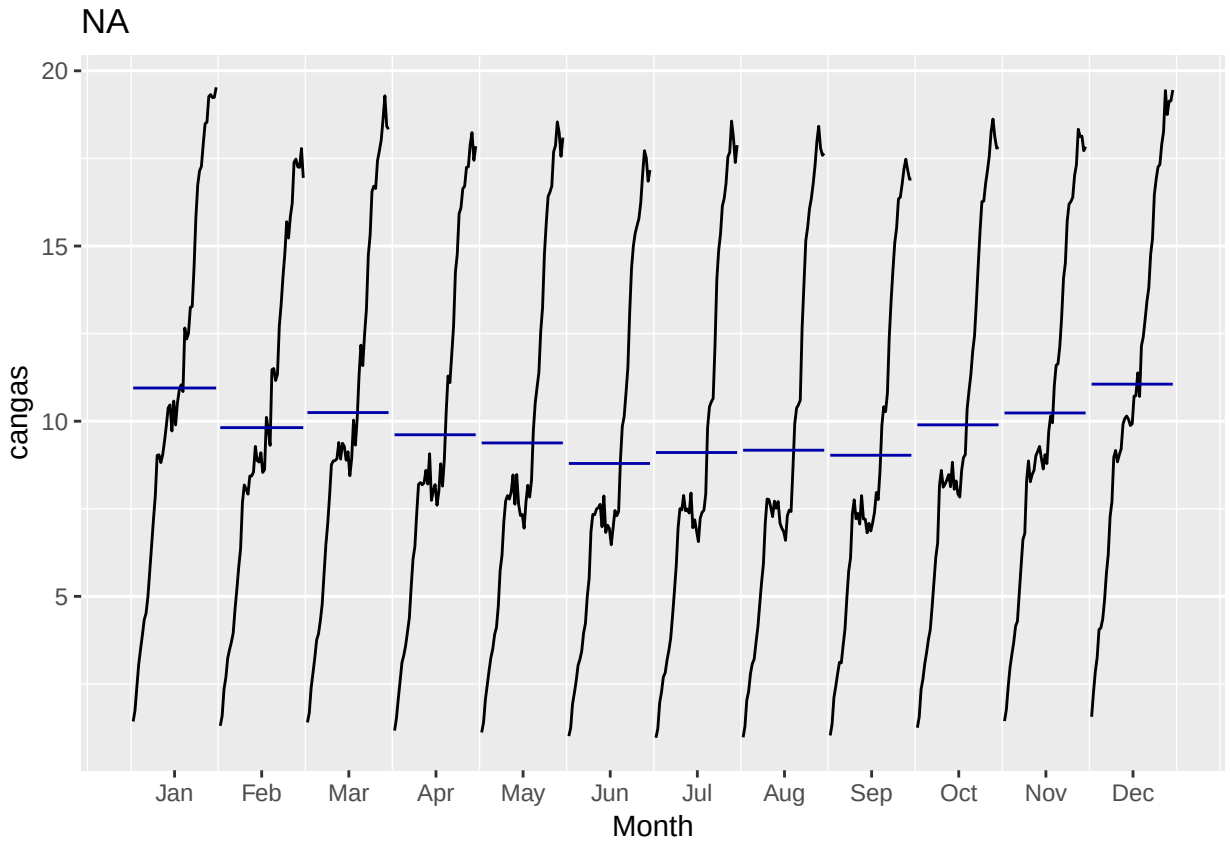
```r
plot(cangas)
```



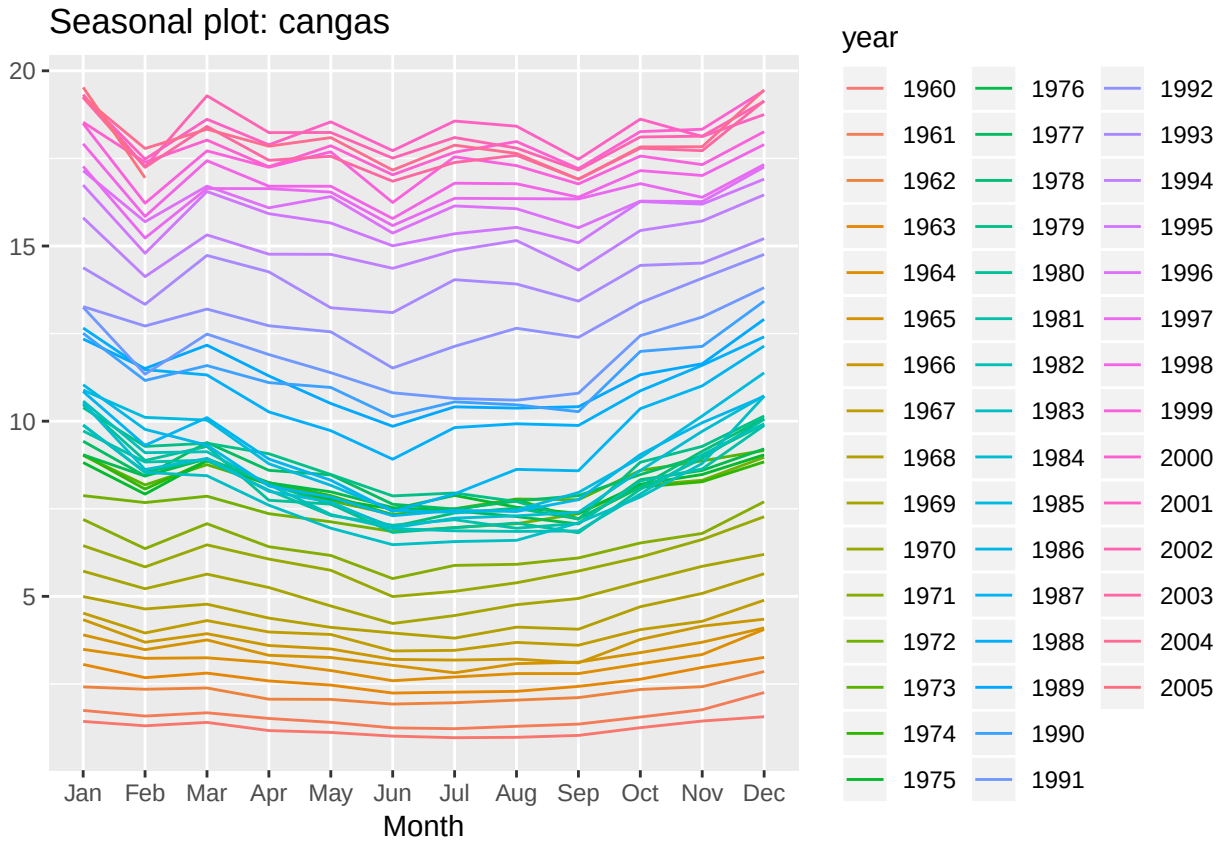

```
#a
plot(cangas,ylab = "billion cubic meters",xlab = "Year")
```



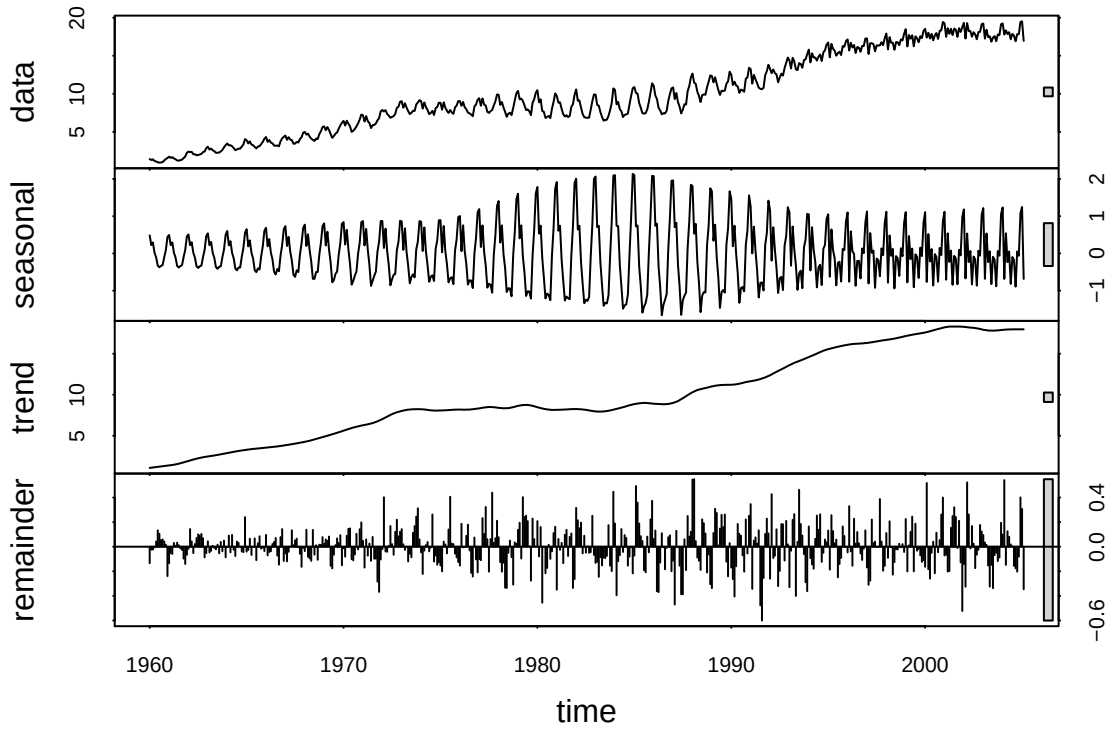

```
ggsubseriesplot(cangas)
```



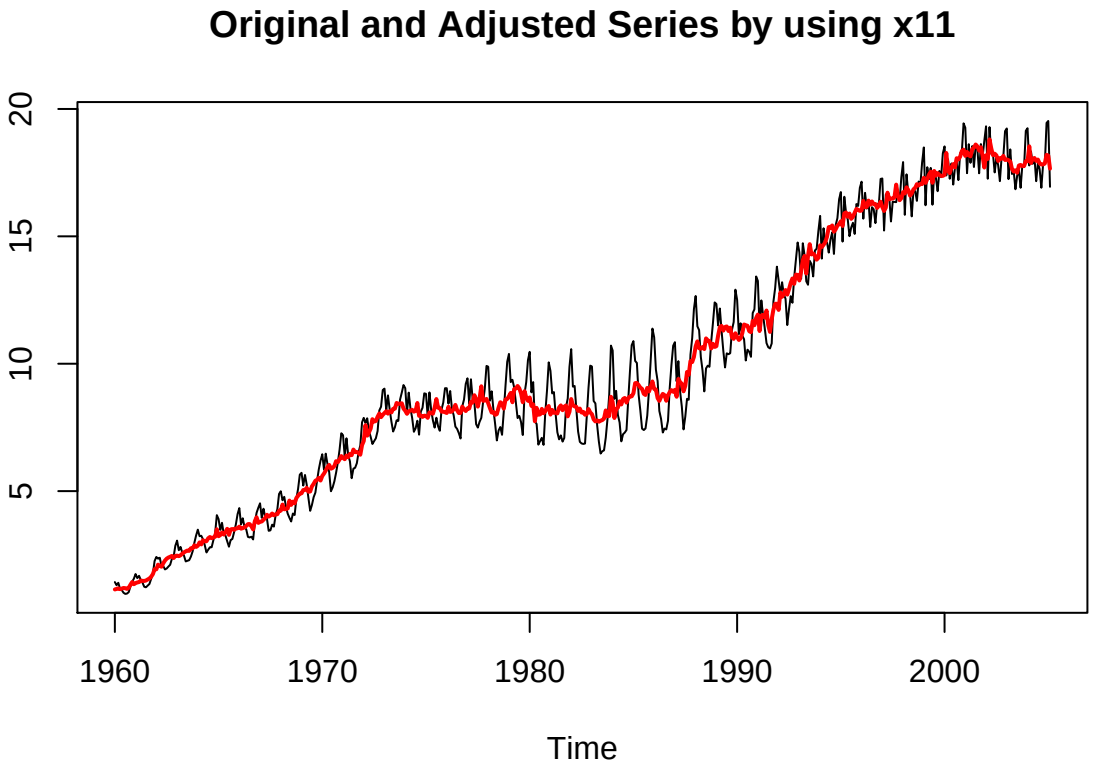

```
ggseasonplot(cangas)
```



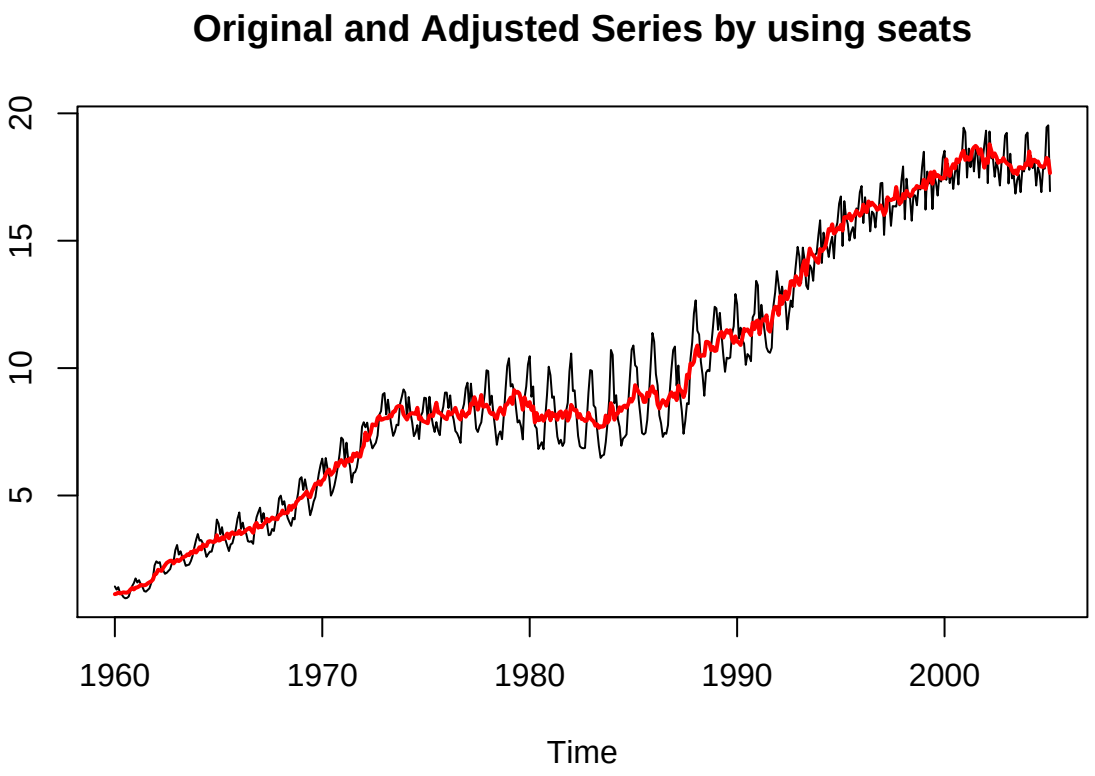

```r
#b
stlc<- stl(cangas,s.window = 8)
plot(stlc)
```



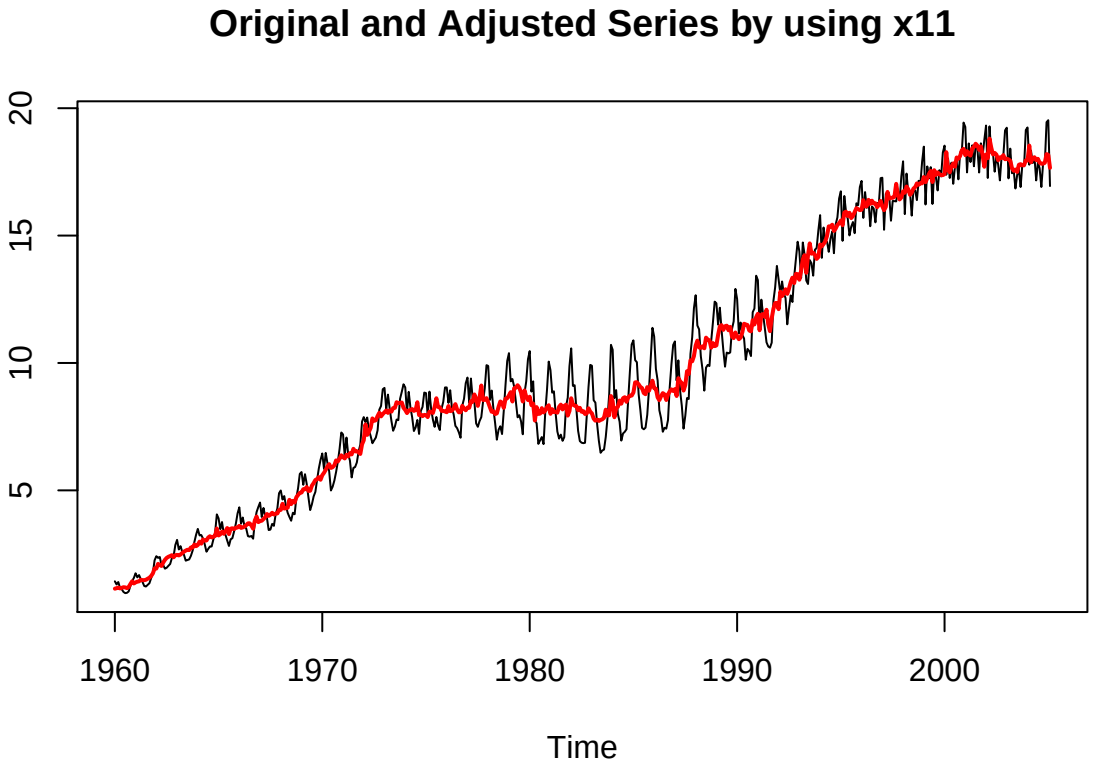
```r
#c
x11c<- seas(cangas,x11="")
seasc<- seas(cangas)
plot(x11c,main="Original and Adjusted Series by using x11")
```



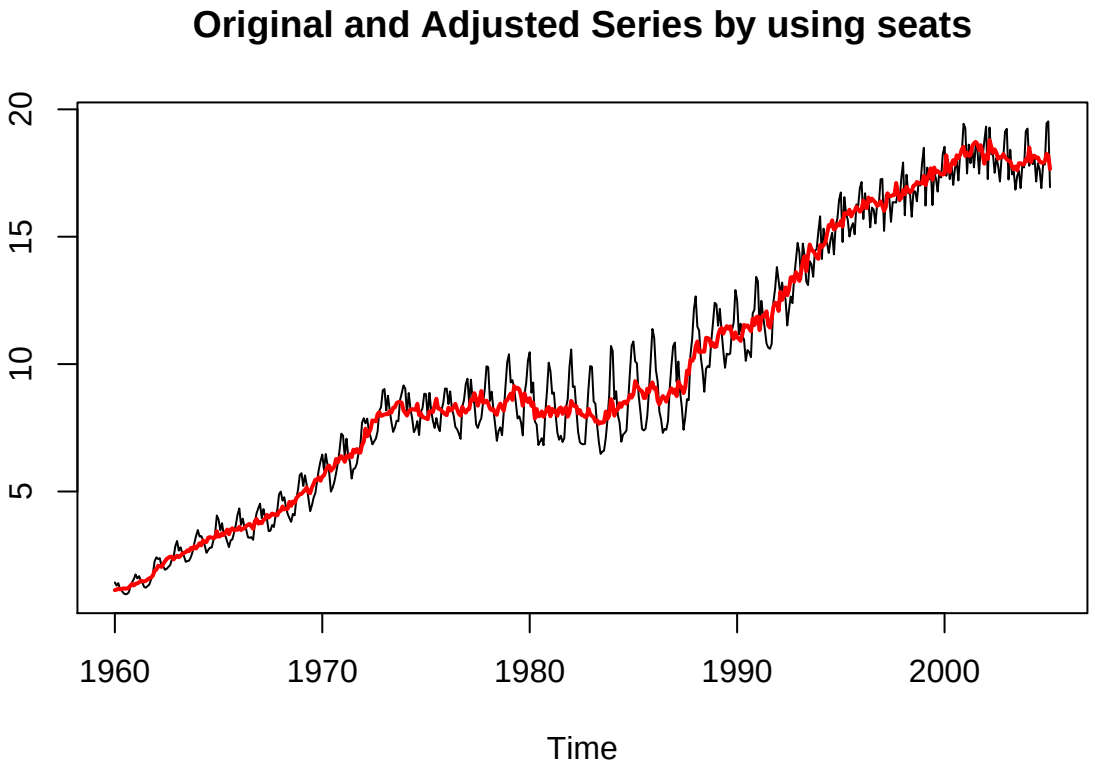

```r
plot(seasc,main="Original and Adjusted Series by using seats")
```




```
plot(x11c,main="Original and Adjusted Series by using x11")
```

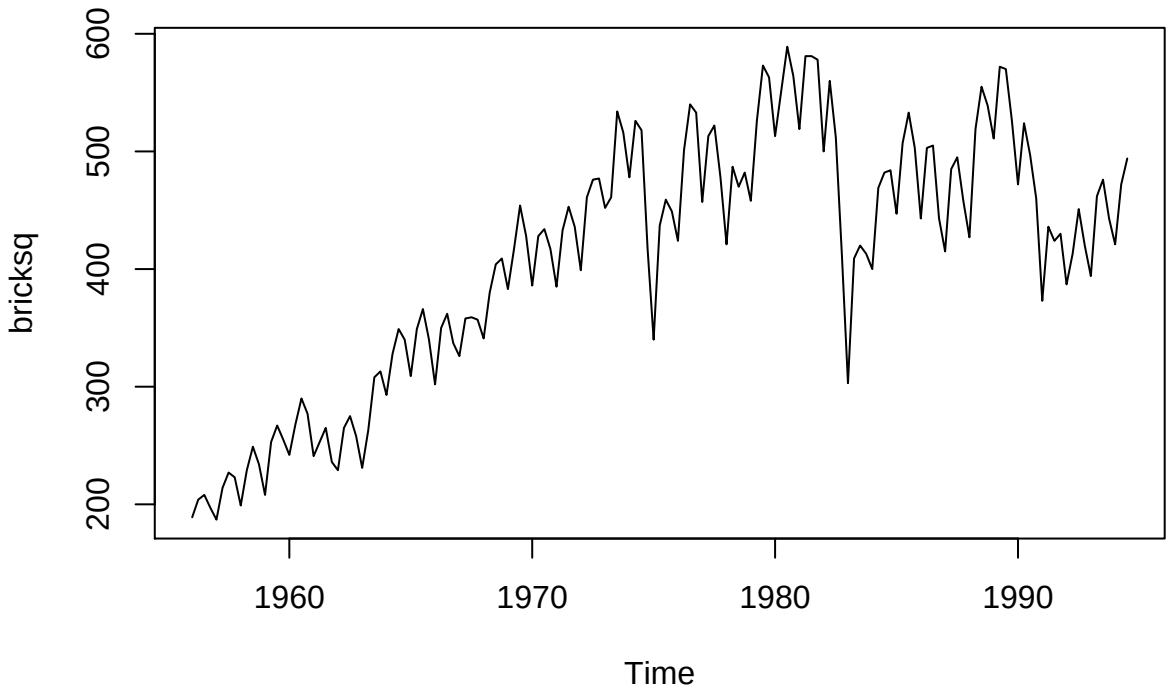



```
plot(seasc,main="Original and Adjusted Series by using seats")
```



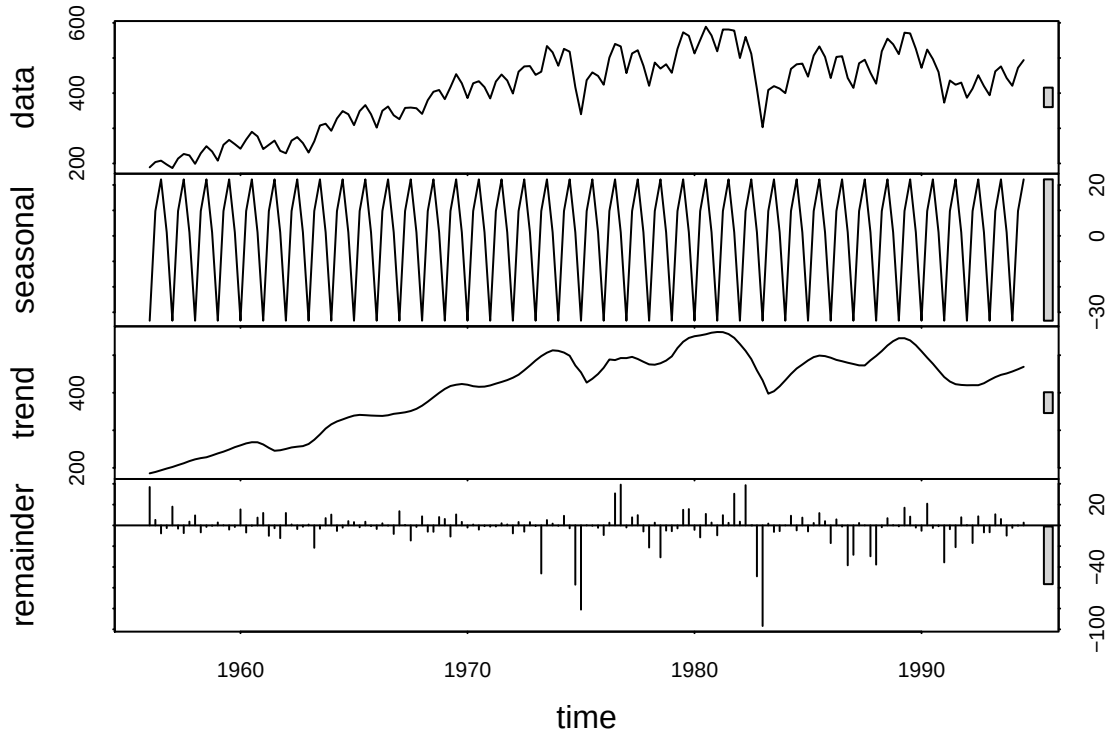
Summary: In a, the trend of monthly Canadian gas production is increasing from 1960 to 2000, as time increases. From seasonal plot, the increasing rate in winter is higher than that in summer.Probaly it is much easier to produce gas in winter. In c, comparing x11 and seats decomposition: they have very closed seasonal and trend component, but share different residuals.

```
plot(bricksq)
```

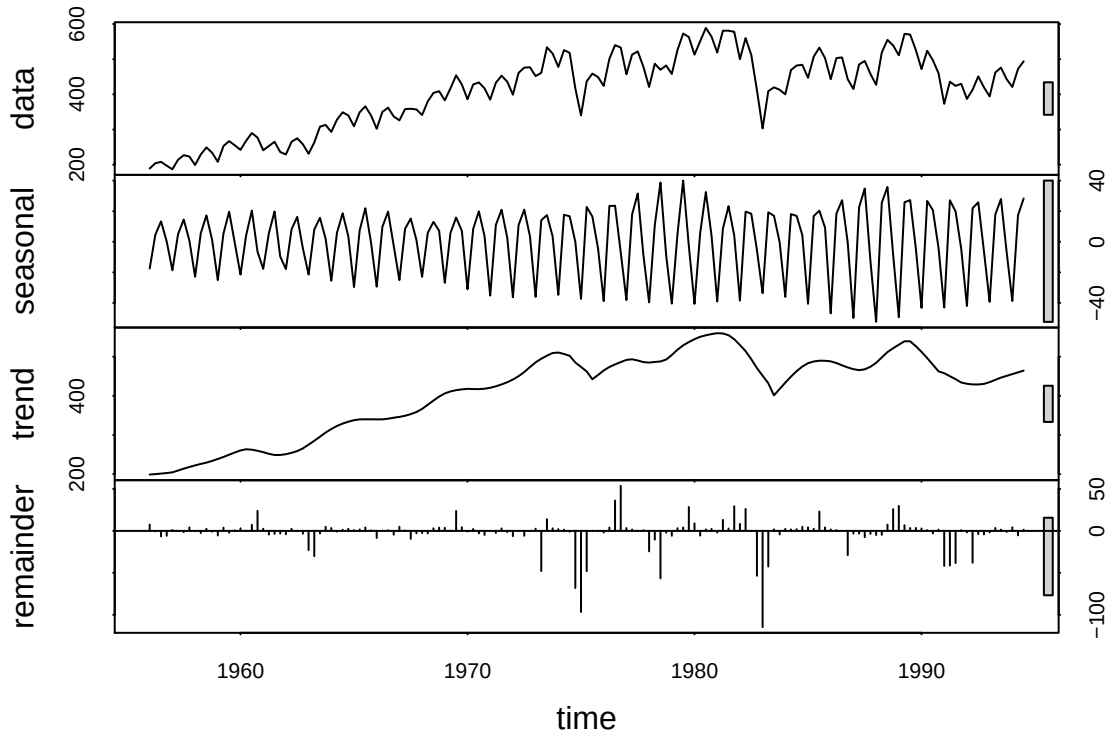



```
#a
# stl with fixed seasonality
```

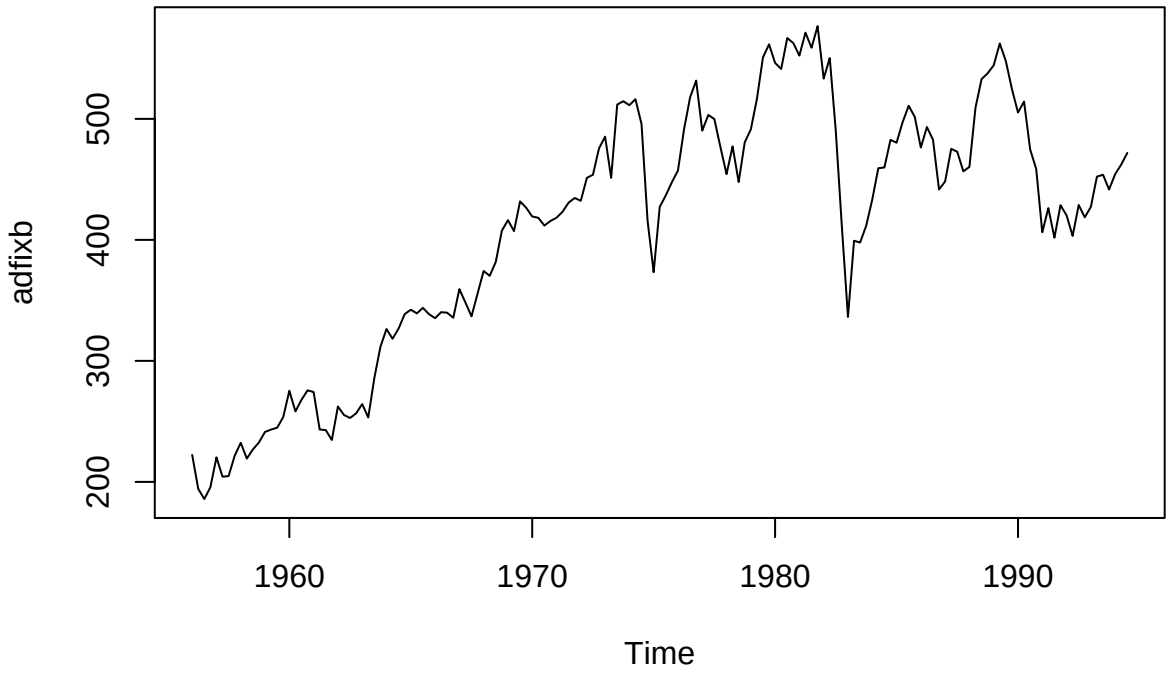
```r
fixstlb <- stl(bricksq,s.window = "periodic",robust = TRUE)
# stl with changing seasonality
changestlb <- stl(bricksq,s.window = 5,robust = TRUE)
plot(fixstlb)
```



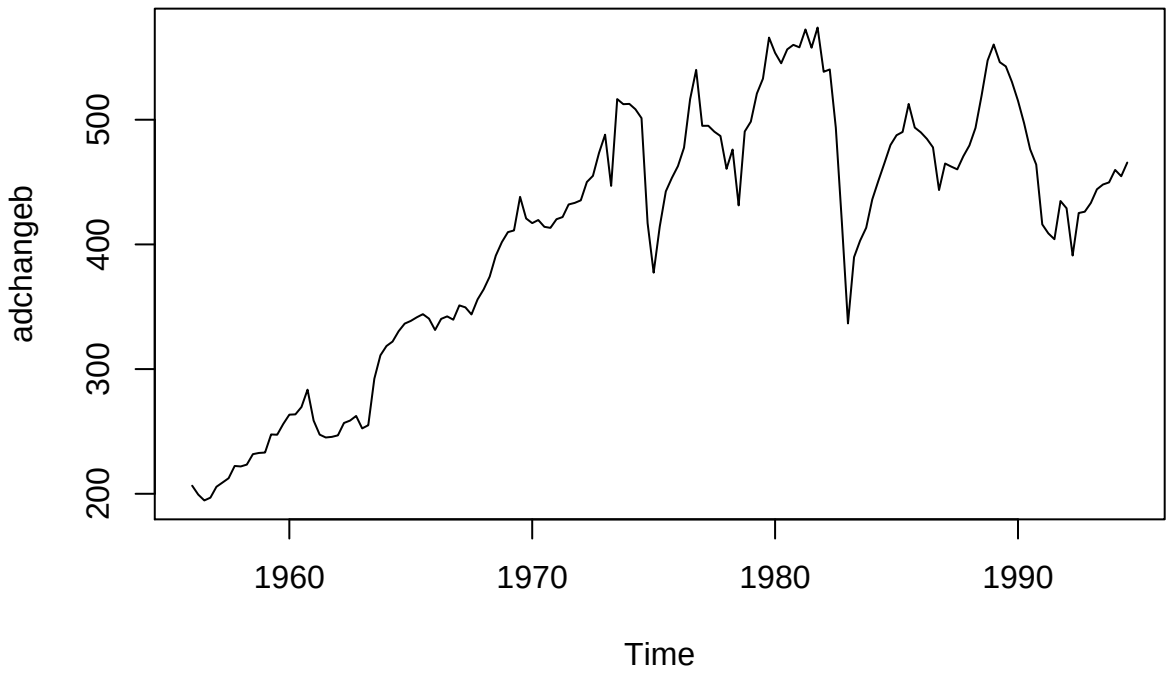

```r
plot(changestlb)
```



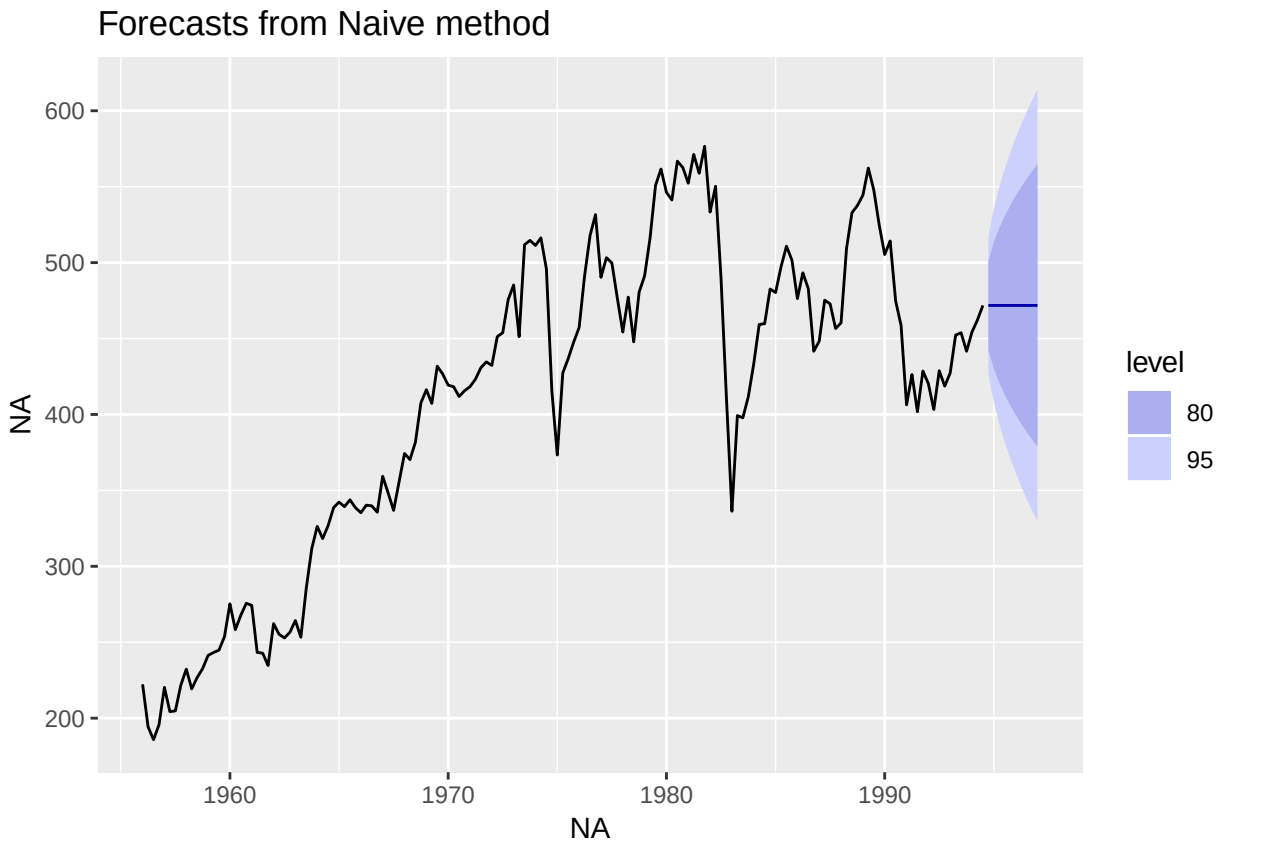
```
#b
#seasonal adjust for fixed seasonality
adfixb <- seasadj(fixstlb)
plot(adfixb)
```



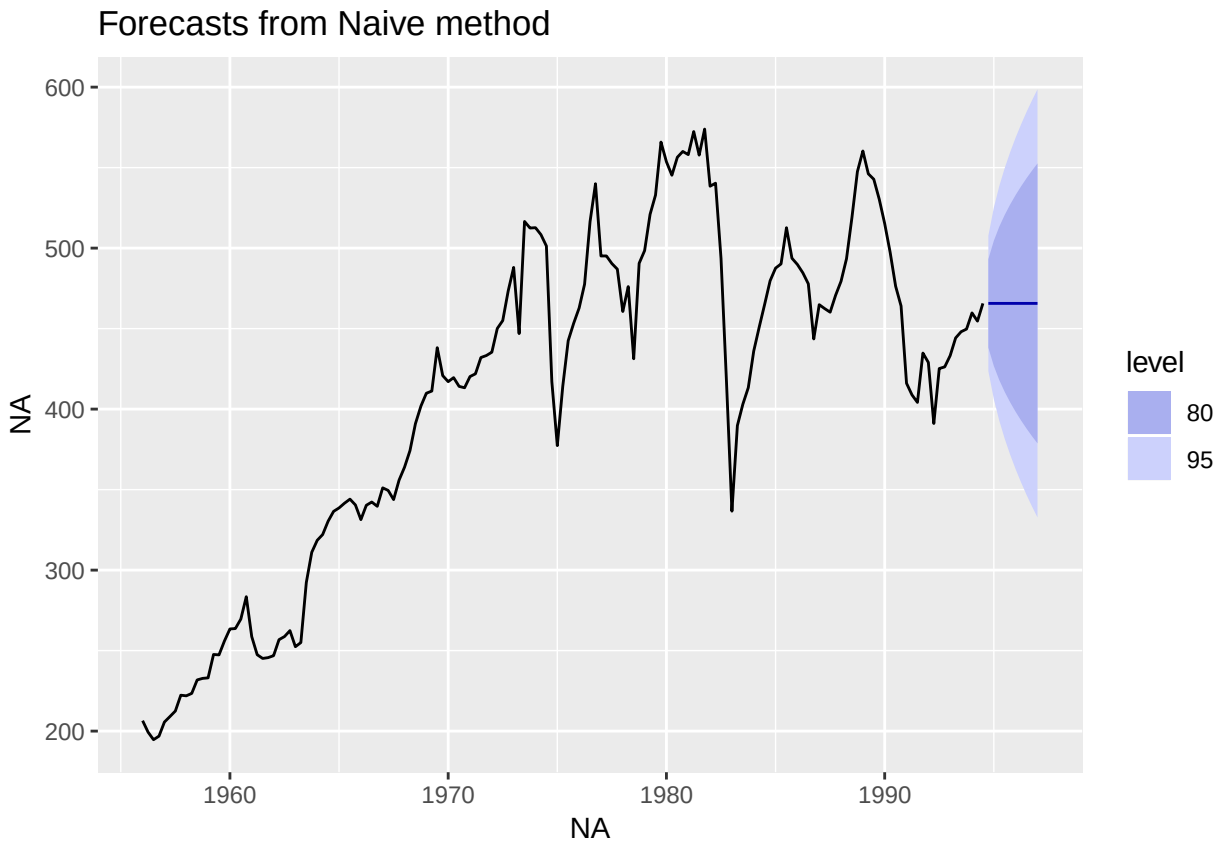

```
#seasonal adjust for chaning seasonality
adchangeb<-seasadj(changestlb)
plot(adchangeb)
```



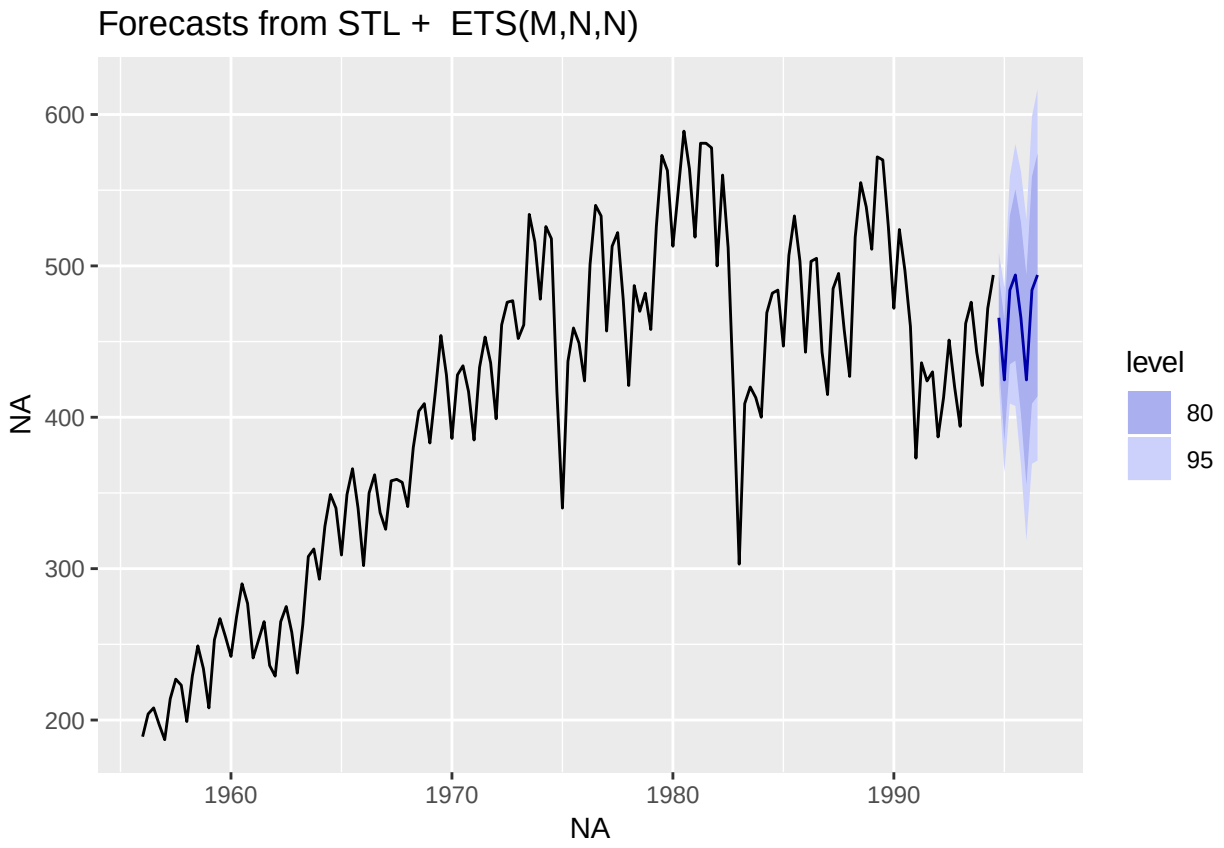

```
#c
fixstlb %>% seasadj() %>% naive() %>% autoplot()
```




```
changestlb %>% seasadj() %>% naive() %>% autoplot()
```

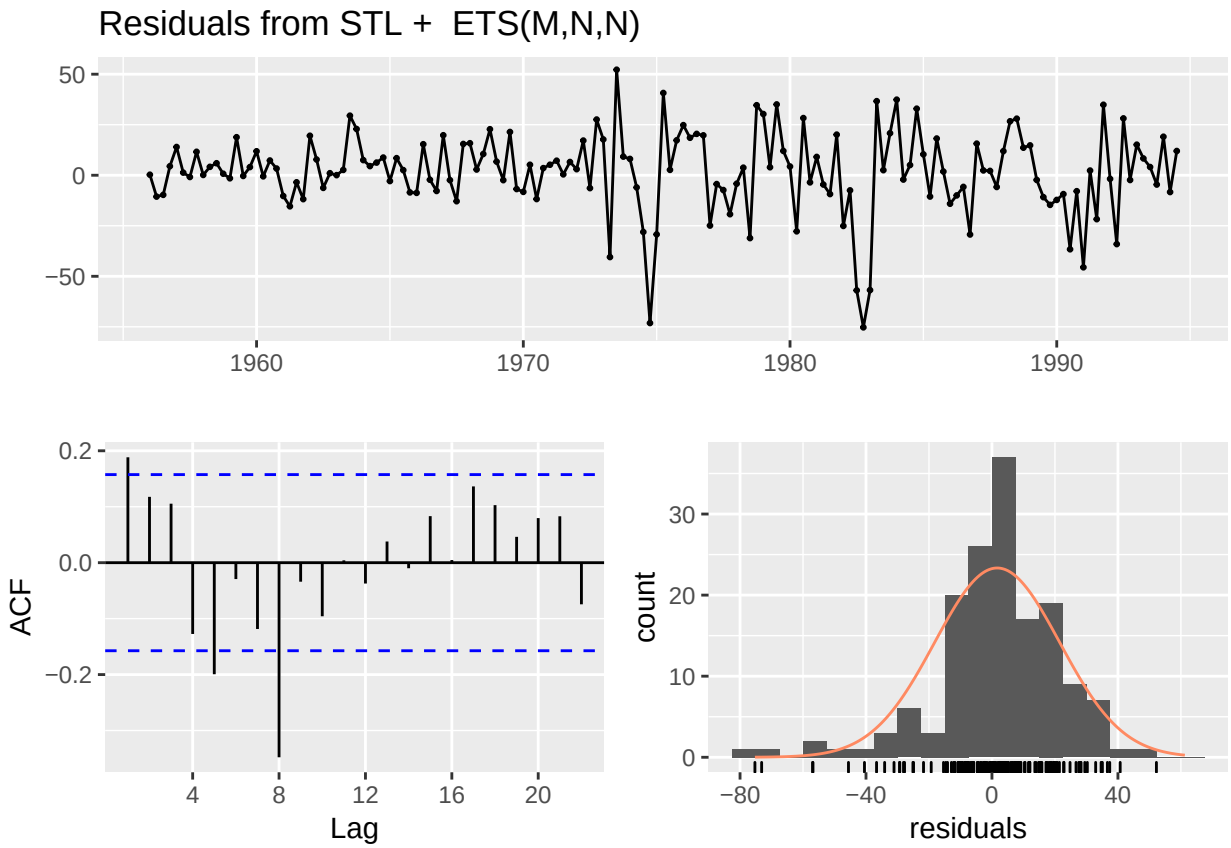



```
## They have similar prediction

#d
stlfb <- stlf(bricksq)
autoplot(stlfb)
```



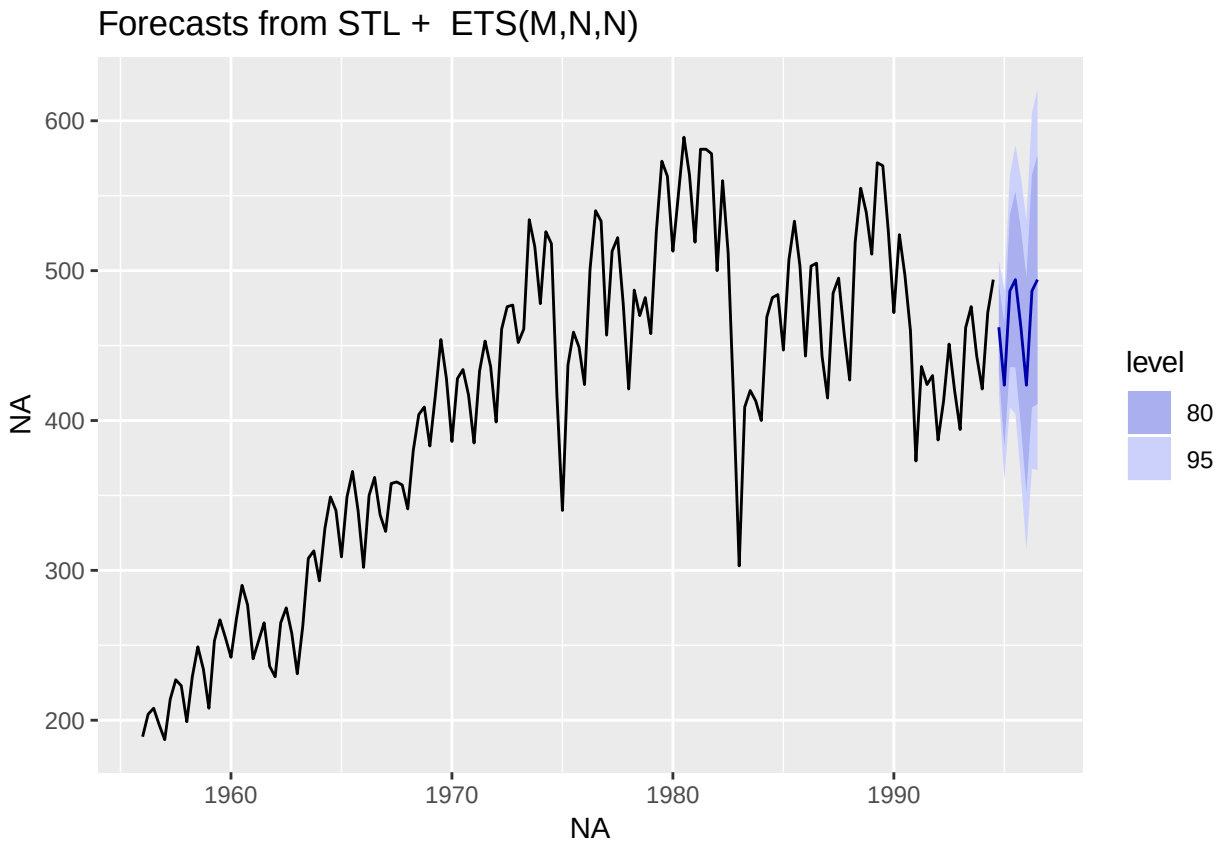

```
#e
checkresiduals(stlfb)
```

```
## Warning in checkresiduals(stlfb): The fitted degrees of freedom is based on
## the model used for the seasonally adjusted data.
```



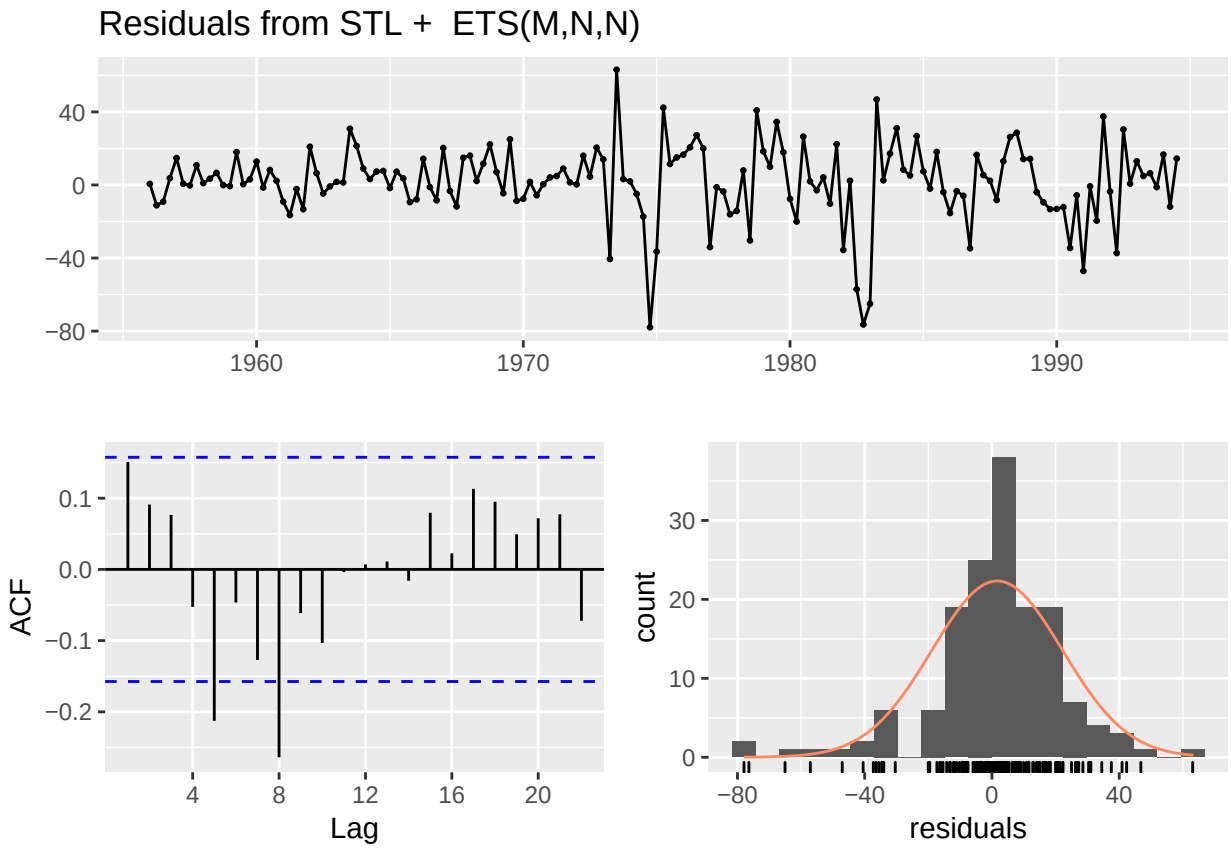

```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ETS(M,N,N)
## Q* = 41.128, df = 6, p-value = 2.733e-07
##
## Model df: 2.   Total lags used: 8
```

```
# the p value is less than 0.05, this time series is not white noise.

#f
stlfr <- stlf(bricksq, robust = TRUE)
autoplot(stlfr)
```




```
checkresiduals(stlfr)
```

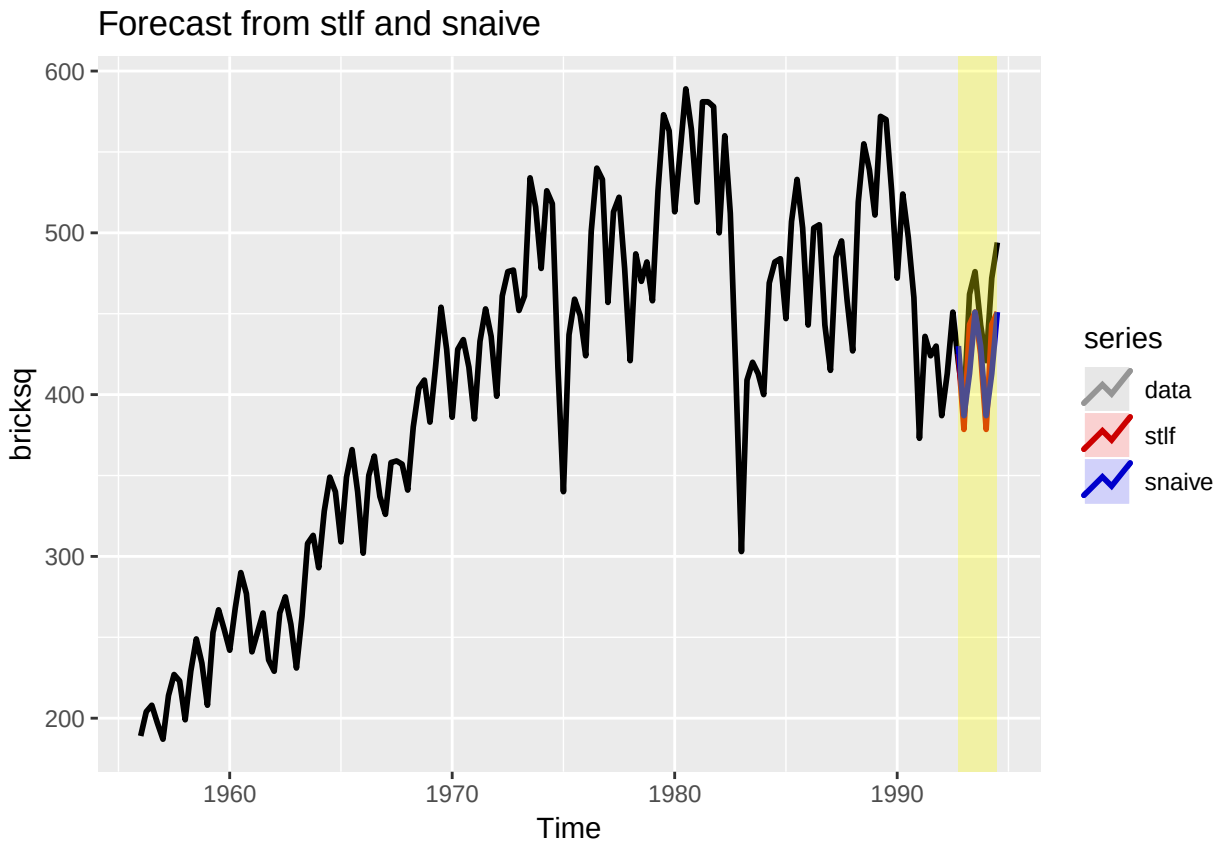
```
## Warning in checkresiduals(stlfr): The fitted degrees of freedom is based on
## the model used for the seasonally adjusted data.
```




```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ETS(M,N,N)
## Q* = 28.163, df = 6, p-value = 8.755e-05
##
## Model df: 2.   Total lags used: 8
```

```
# There are not big difference btw standard stlf and robust stlf. the p value is less than 0.05, this t

#g
library("ggplot2")
trainset_brick <- subset(bricksq,end = length(bricksq) - 8)
testset_brick <- subset(bricksq,start = length(bricksq) - 7)

snaive_brick <- snaive(trainset_brick)
stlf_brick_part <- stlf(trainset_brick, robust = TRUE)

# plot data and forecast results
autoplot(bricksq, series = "data") +
  geom_line(size = 1) +
  autolayer(stlf_brick_part, PI = FALSE, size = 1,series = "stlf") +
  autolayer(snaive_brick, PI = FALSE, size = 1,series = "snaive") +
  scale_color_manual(values = c("gray", "blue", "red"),
                     breaks = c("data", "stlf", "snaive")) +
  ggtitle("Forecast from stlf and snaive") +
  annotate("rect",xmin=1992.75,xmax=1994.5,ymin=-Inf,ymax=Inf,
```

```
    fill="yellow",alpha = 0.3)
```




```
#Sumamry: These two methods are quite similar, from observation, stlf gives a better approximation.
```