# Homework3

seongbin AN

2020 9 23

# SOL0

Datacheck

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.
3.0 --
```

```
## √ ggplot2 3.3.2     √ purrr   0.3.4
## √ tibble  3.0.3     √ dplyr   1.0.2
## √ tidyr   1.1.2     √ stringr 1.4.0
## √ readr   1.3.1     √ forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts
() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(forecast)
```

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

```
library(seasonal)
```

```
##
## Attaching package: 'seasonal'
```

```
## The following object is masked from 'package:tibble':
##
##     view
```

```
library(aTSA)
```

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

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

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

```
library(timsac)
X<-read.csv("C://Users//stat//Desktop//학교//2-2//시계열//loadregr.csv")
dim(X)
```

```
## [1] 120   3
```

```
head(X)
```

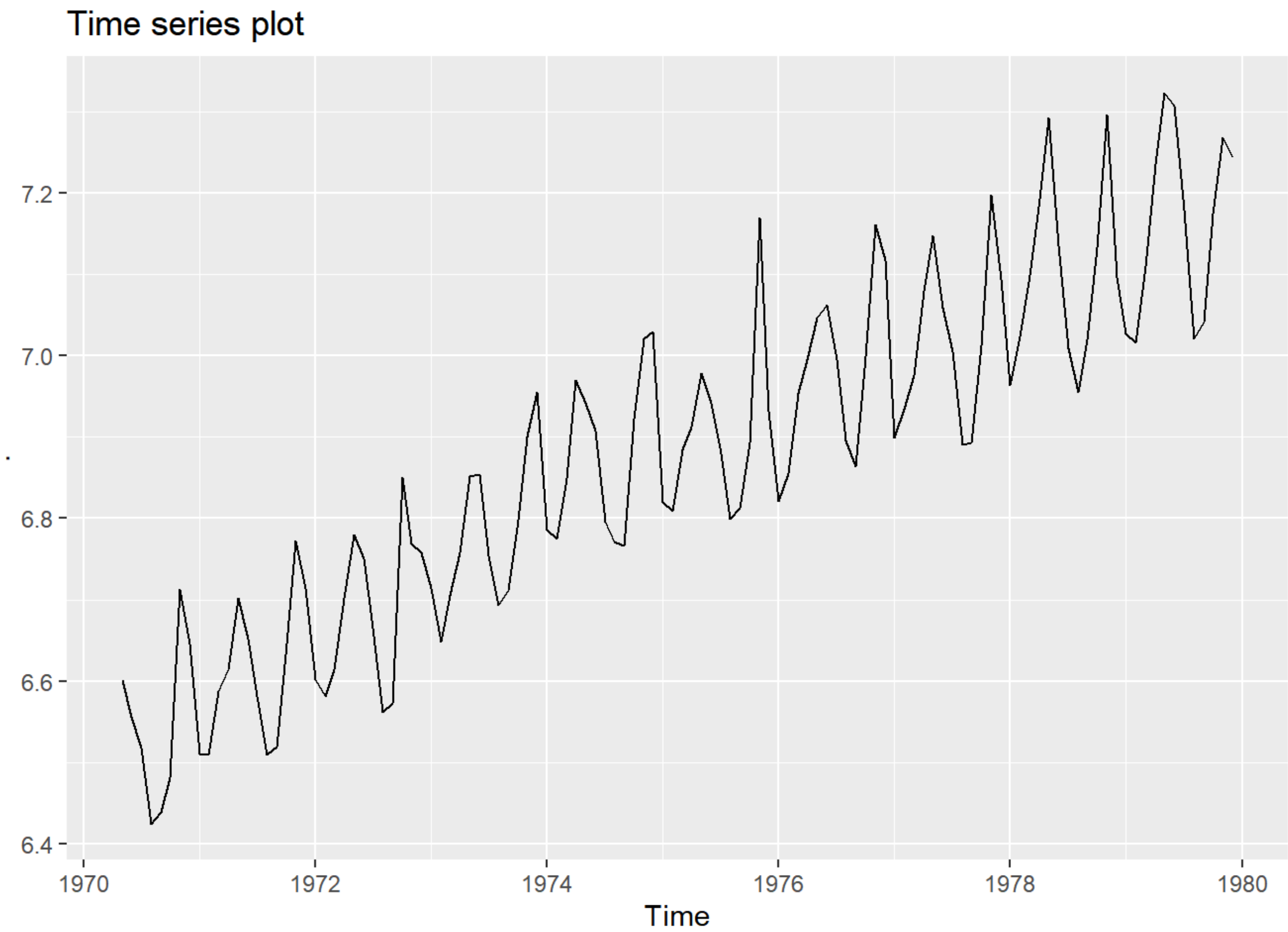
```
##       MKw Month     Time
## 1 6.60110   Jan 1970.000
## 2 6.55576   Feb 1970.083
## 3 6.51810   Mar 1970.167
## 4 6.42498   Apr 1970.250
## 5 6.43868   May 1970.333
## 6 6.48173   Jun 1970.417
```

```
X_ts<-ts(X$MKw, start=c(1970,5),end=c(1979,12), frequency=12)
```

# SOL1

# Draw time series plot

```
X_ts %>% autoplot() +
  ggtitle("Time series plot")
```




## SOL2 # Fit decomposition model and draw a plot with the original data, seasonally adjusted, trend-cycle component all together.

I make stationary test to choose best decomposition way

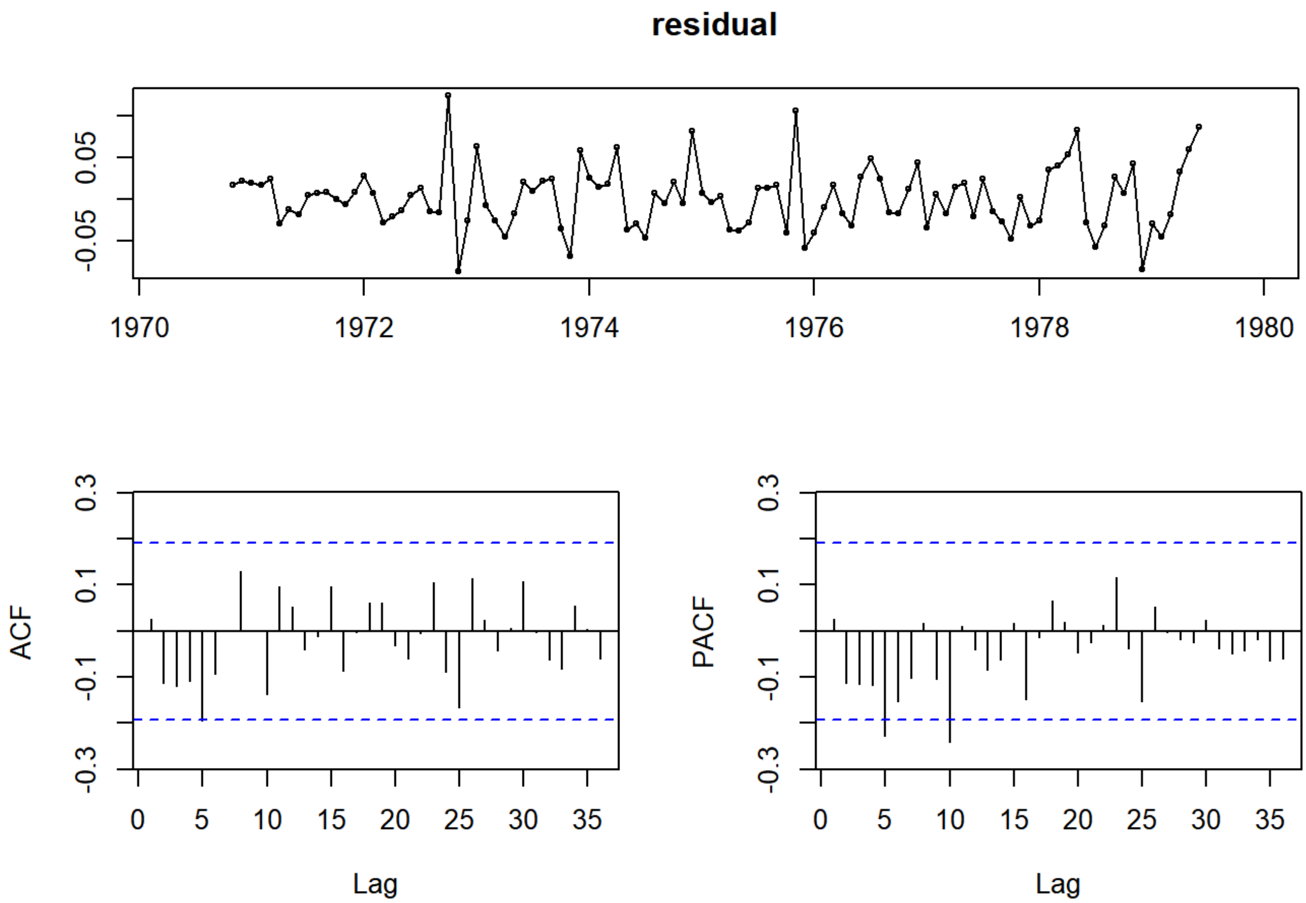
Classical decomposition(additive)

```
dd1=decompose(X_ts,type="additive")
tseries::kpss.test(dd1$random,null="Level")
```

```
## Warning in tseries::kpss.test(dd1$random, null = "Level"): p-value greater
than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  dd1$random
## KPSS Level = 0.025425, Truncation lag parameter = 4, p-value = 0.1
```

```
tsdisplay(dd1$random, main="residual")
```

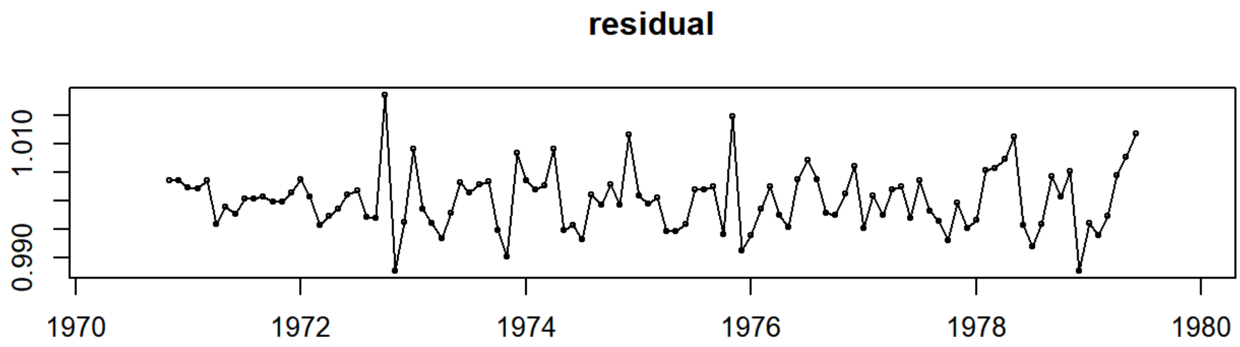


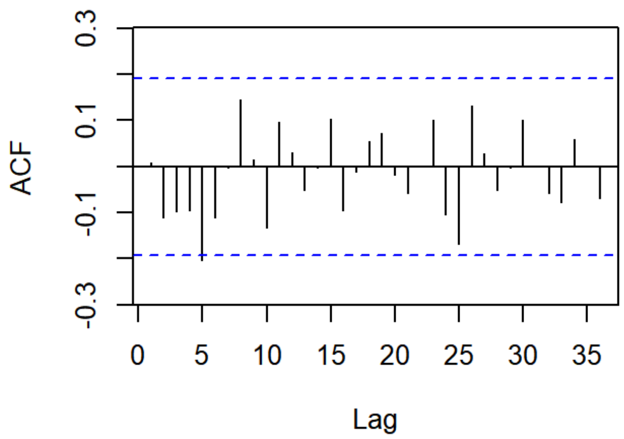

## Classical decomposition(multiplicative)

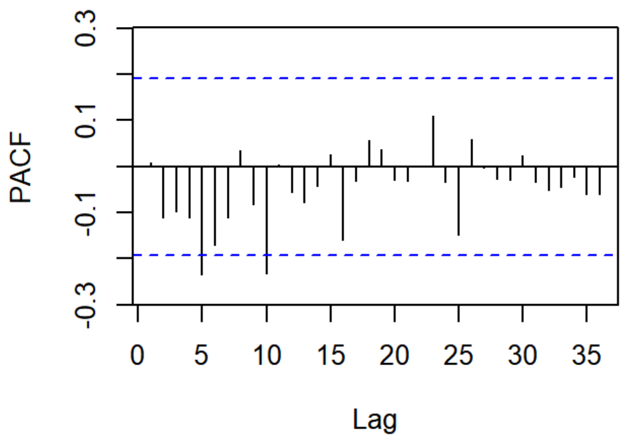
```
dd2=decompose(X_ts,type="multiplicative")
tseries::kpss.test(dd2$random,null="Level")
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  dd2$random
## KPSS Level = NaN, Truncation lag parameter = 4, p-value = NA
```

```
tsdisplay(dd2$random, main="residual")
```




## SEATS decomposition

```
dd3<-seas(X_ts)
names(dd3)
```

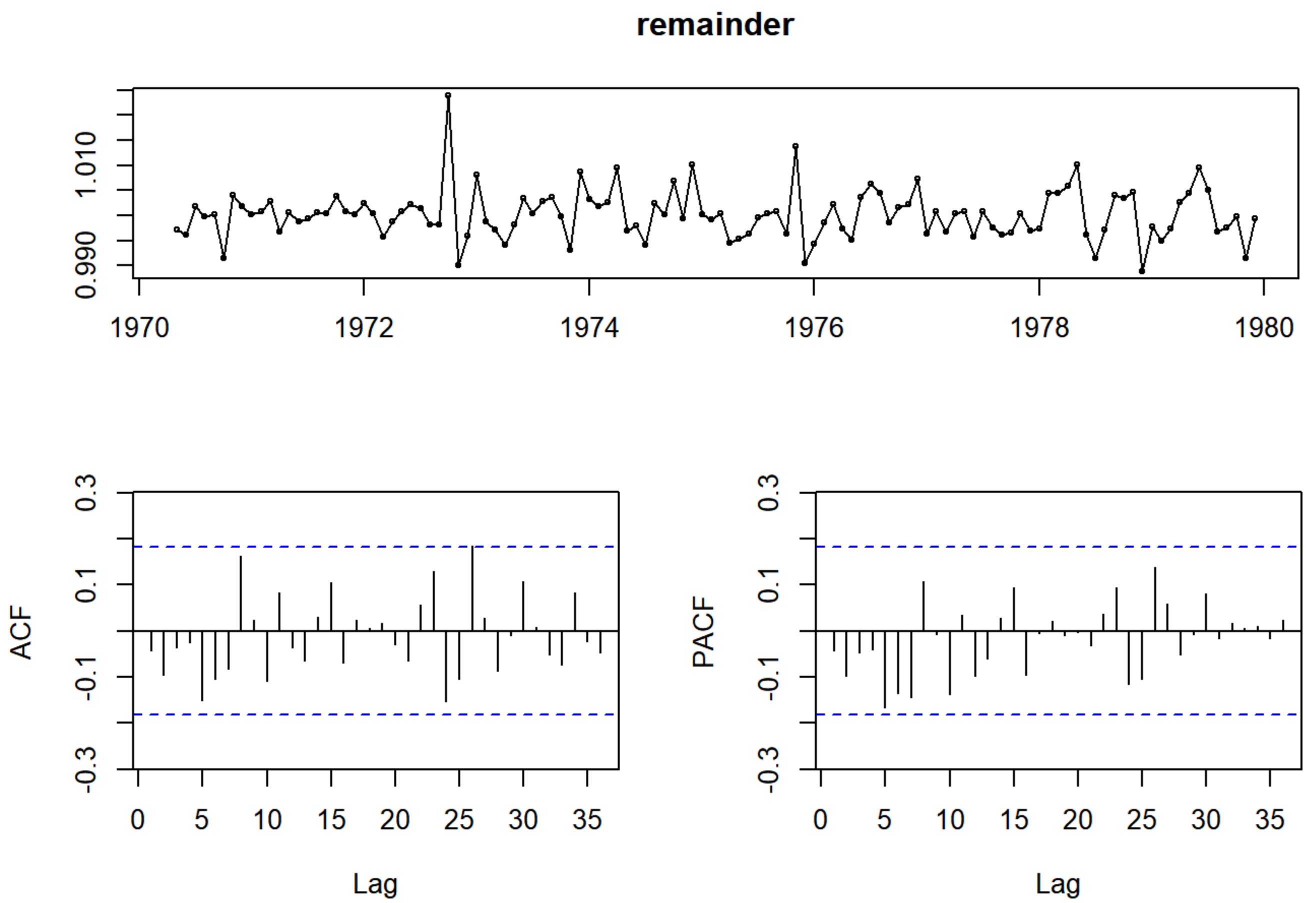
```
##  [1] "call"          "list"          "series"        "data"          "err"
##  [6] "udg"           "est"           "model"         "fivebestmdl"   "x"
## [11] "spc"           "wdir"
```

```
dd3_r<-remainder(dd3)
tseries::kpss.test(dd3_r,null="Level")
```

```
## Warning in tseries::kpss.test(dd3_r, null = "Level"): p-value greater than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  dd3_r
## KPSS Level = 0.066507, Truncation lag parameter = 4, p-value = 0.1
```

```
tsdisplay(dd3_r, main="remainder")
```



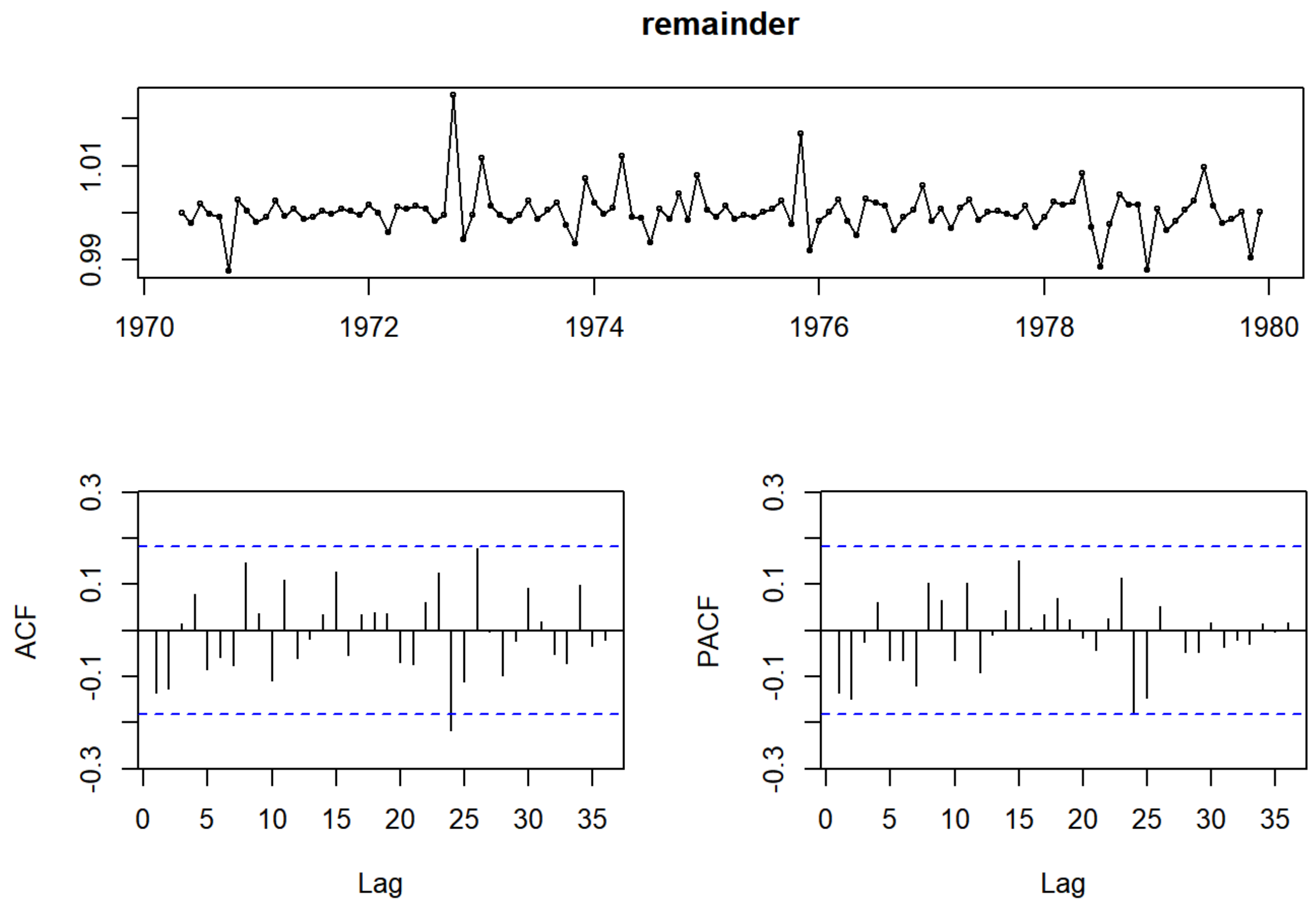

## X11 decomposition

```
X_ts %>% seas(x11="") ->dd4
dd4_r=remainder(dd4)
tseries::kpss.test(dd4_r,null="Level")
```

```
## Warning in tseries::kpss.test(dd4_r, null = "Level"): p-value greater than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  dd4_r
## KPSS Level = 0.16255, Truncation lag parameter = 4, p-value = 0.1
```

```
tsdisplay(dd4_r, main="remainder")
```



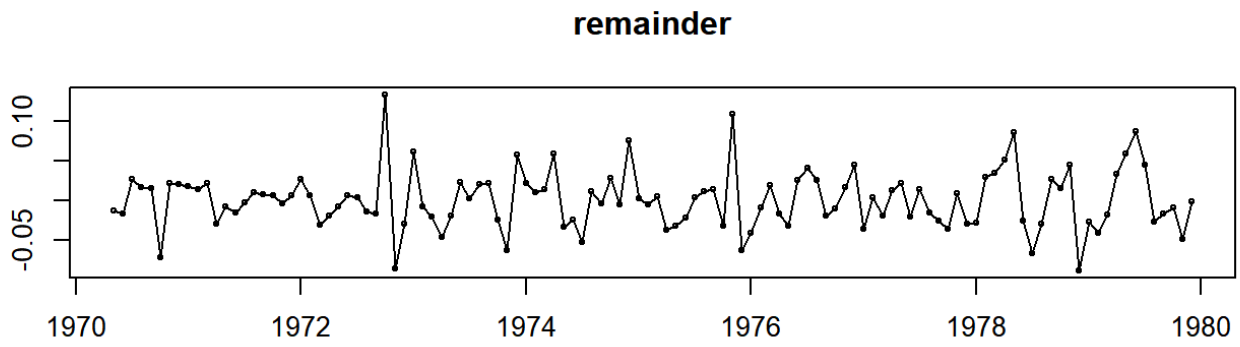

## STL decomposition

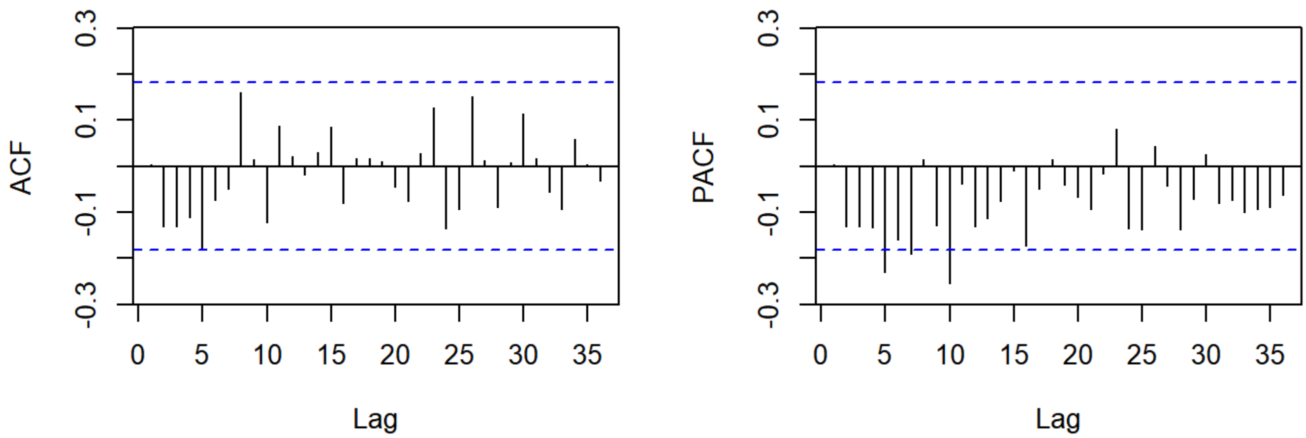
```
dd5<-stl(X_ts,'periodic')
dd5_r=remainder(dd5)
tseries::kpss.test(dd5_r,null="Level")
```

```
## Warning in tseries::kpss.test(dd5_r, null = "Level"): p-value greater than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  dd5_r
## KPSS Level = 0.016203, Truncation lag parameter = 4, p-value = 0.1
```

```
tsdisplay(dd5_r, main="remainder")
```

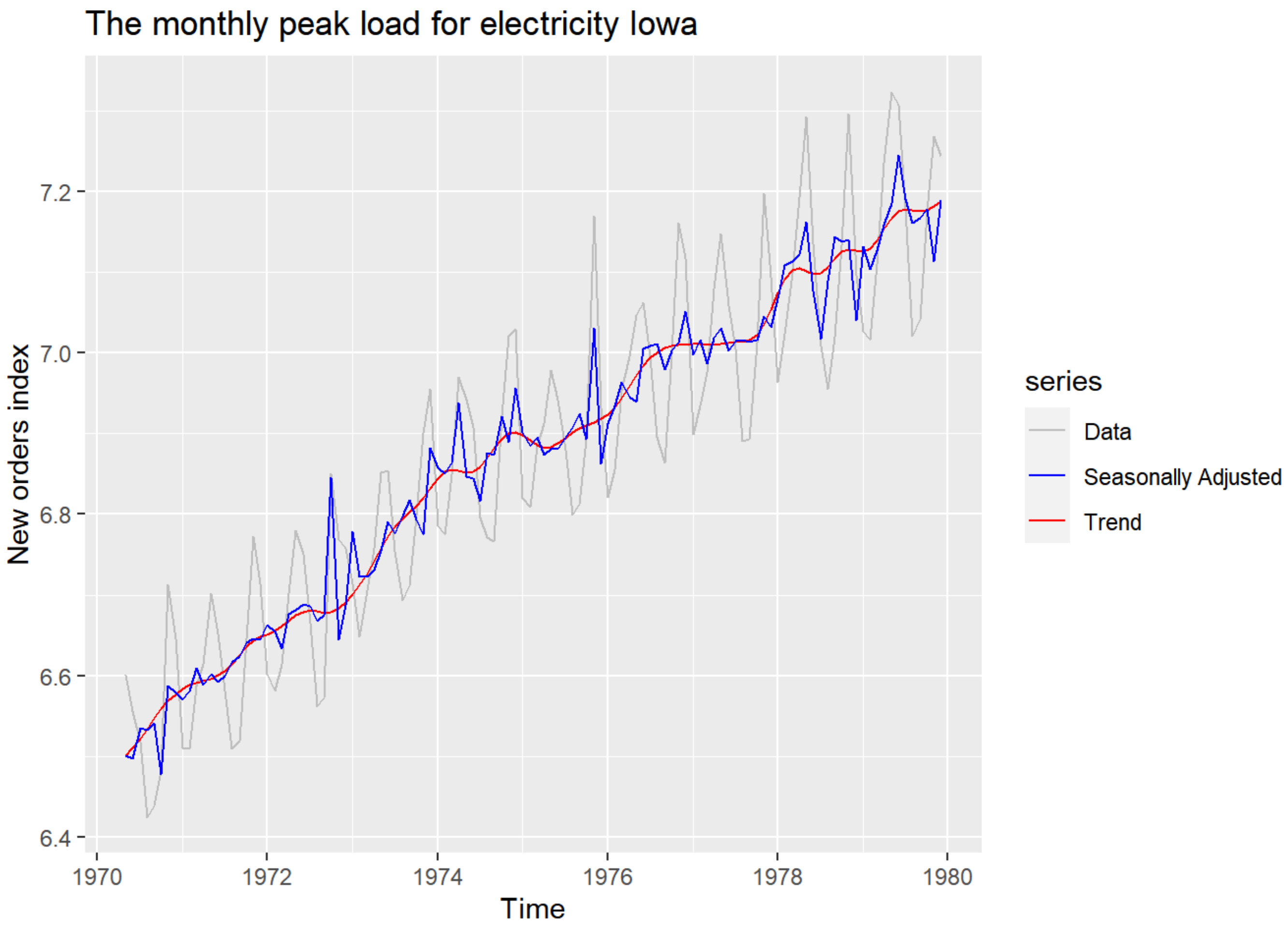



Based on residual ACF&PACF plot and stationary test, X11 decomposition is being used

A plot with the original data, seasonally adjusted, trend-cycle component all together

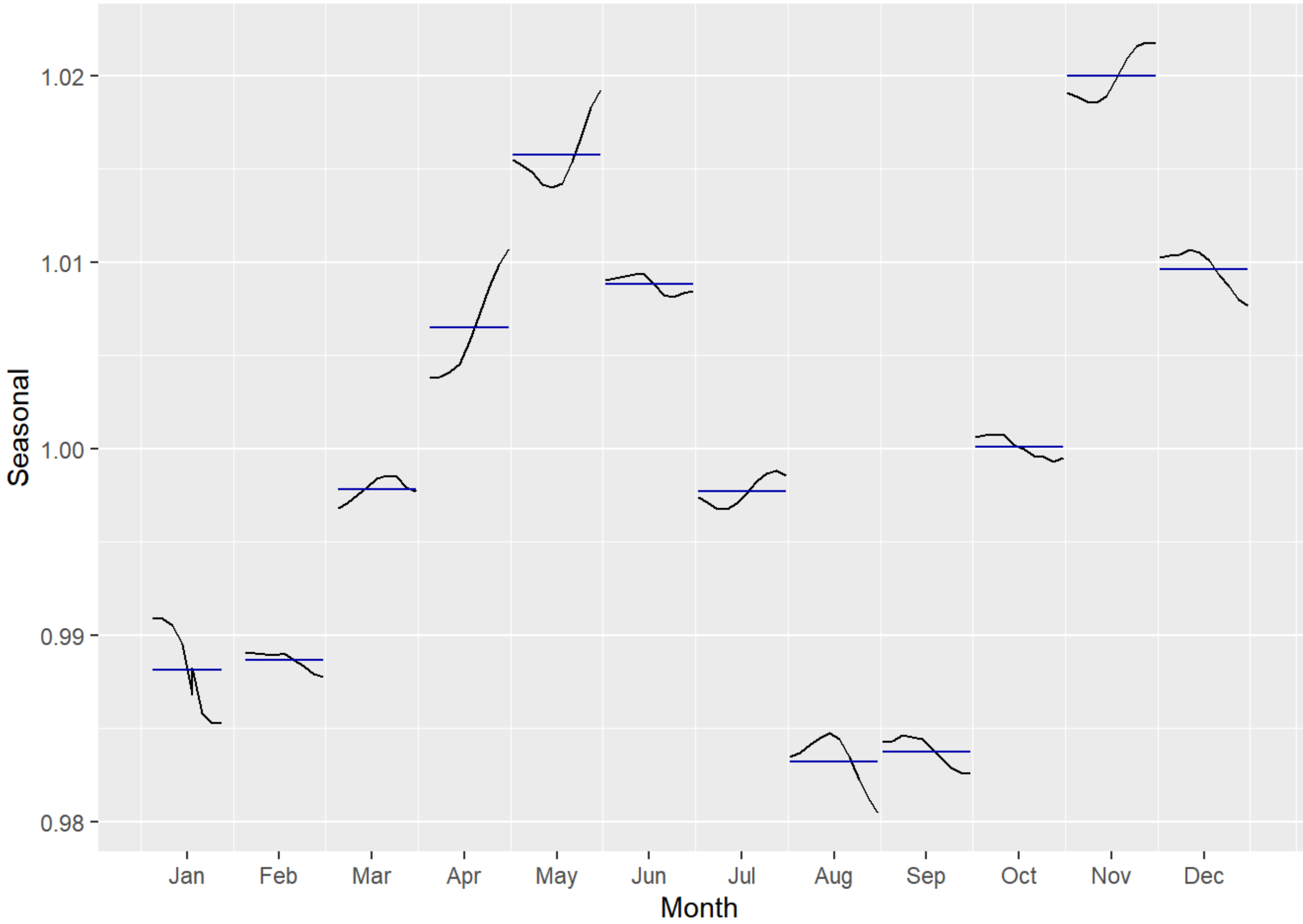
```
X_ts %>% seas(x11="") ->fit
autoplot(X_ts, series="Data")+
  autolayer(trendcycle(fit),series="Trend") +
  autolayer(seasadj(fit),series="Seasonally Adjusted") +
  xlab("Time") + ylab("New orders index")+
  ggtitle("The monthly peak load for electricity Iowa") +
  scale_color_manual(values=c("gray","blue","red"),breaks=c("Data","Seasonally
Adjusted","Trend"))
```




# SOL3

# Draw sub-series seasonal effect graph in each month.

```
fit %>% seasonal() %>% ggsubseriesplot()+ylab("Seasonal")
```

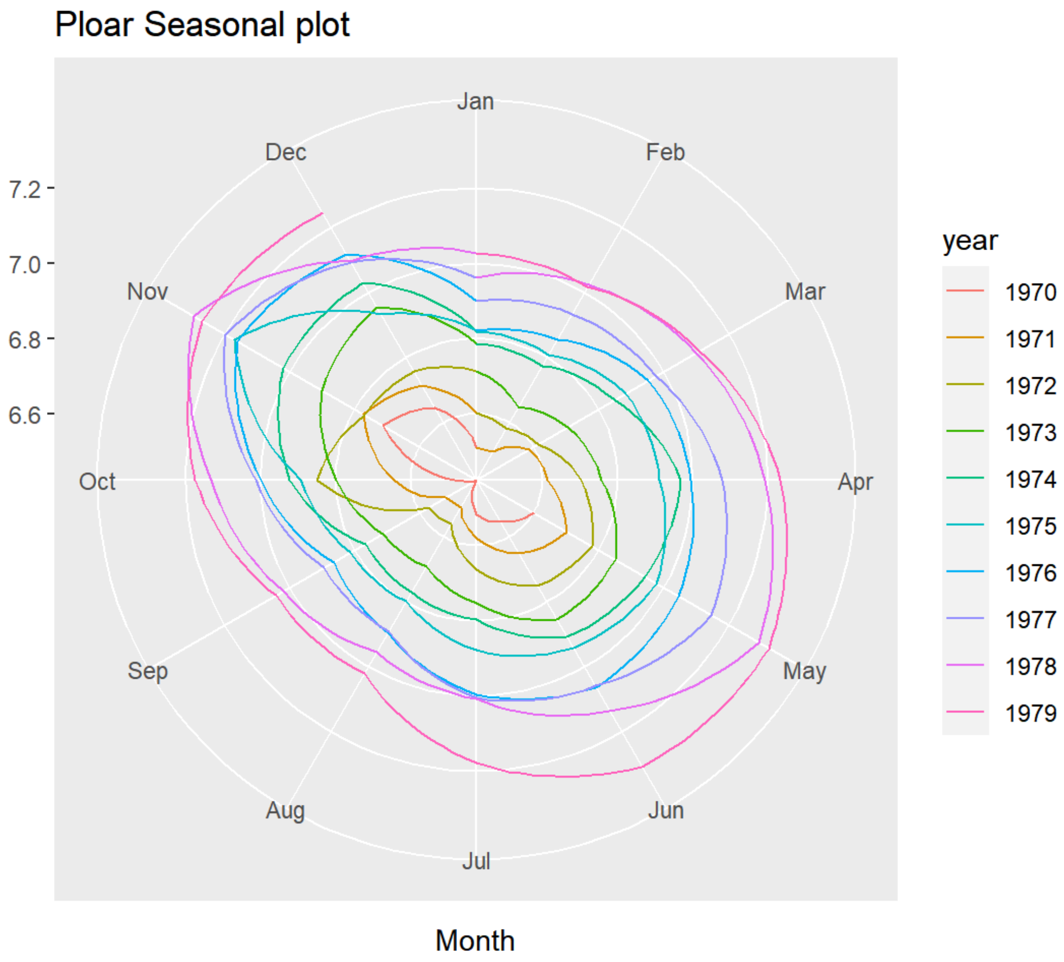



This graph shows shape of seasonal effect of each month (Blue line is constant seasonal effect of each month)

# SoL4

# Draw a polar seasonal plot.

```
ggseasonplot(X_ts,polar=TRUE)+
  ggtitle("Ploar Seasonal plot")
```


Ploar Seasonal plot
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
7.2
7.0
6.8
6.6
year
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
Month