Timeseries

2023-08-07

```
knitr::opts_chunk$set(echo = TRUE)
```

Time Series

```
#libraries
library(readxl)
library(lubridate)
library(zoo)
library(xts)
library(forecast)
library(dplyr)
```

```
dir_name <- "Data Sources"
file_name <- "NY-JFKairport-temperatures.csv"
file_path <- paste(getwd(),dir_name,file_name,sep="/")
```

```
df_monthly <- read.csv(file_path)
```

Including Plots You can also embed plots, for example:

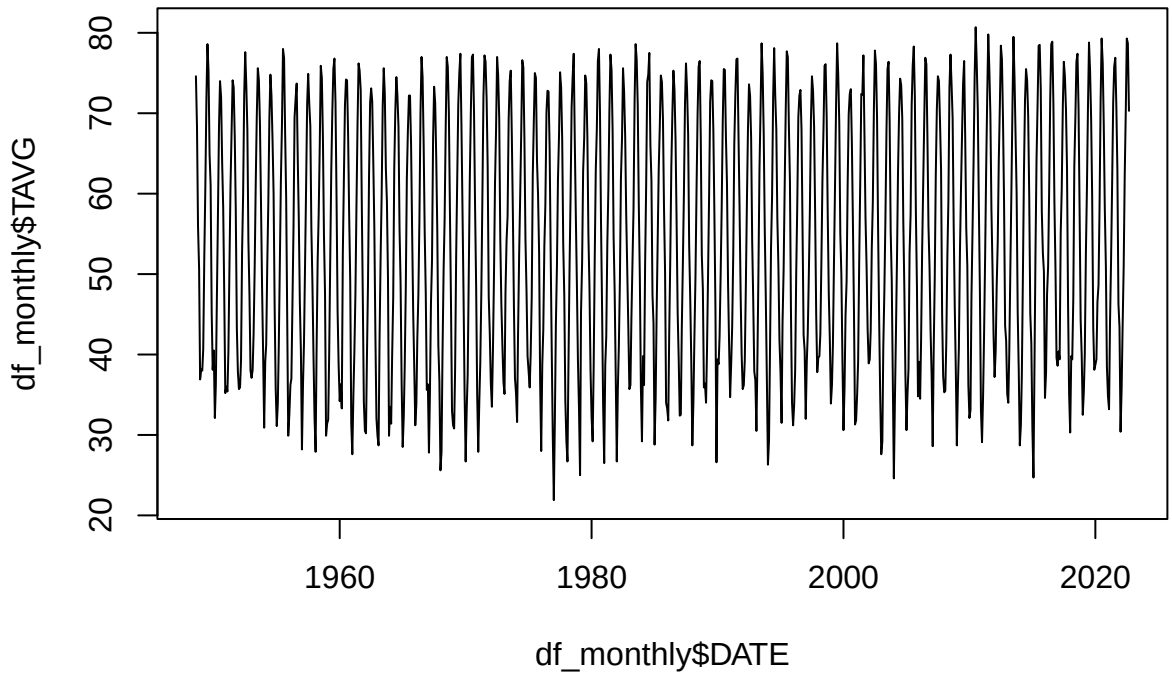
```
head(df_monthly)
```

```
##       STATION                              NAME LATITUDE LONGITUDE ELEVATION
## 1 USW00094789 JFK INTERNATIONAL AIRPORT, NY US 40.63915  -73.7639       2.7
## 2 USW00094789 JFK INTERNATIONAL AIRPORT, NY US 40.63915  -73.7639       2.7
## 3 USW00094789 JFK INTERNATIONAL AIRPORT, NY US 40.63915  -73.7639       2.7
## 4 USW00094789 JFK INTERNATIONAL AIRPORT, NY US 40.63915  -73.7639       2.7
## 5 USW00094789 JFK INTERNATIONAL AIRPORT, NY US 40.63915  -73.7639       2.7
## 6 USW00094789 JFK INTERNATIONAL AIRPORT, NY US 40.63915  -73.7639       2.7
##      DATE CDSD CDSD_ATTRIBUTES EMNT EMNT_ATTRIBUTES EMXT EMXT_ATTRIBUTES HDSD
## 1 1948-07   NA                NA                  NA                   NA
## 2 1948-08   NA                59          ,X,07,  101          ,X,27,   NA
## 3 1948-09   NA                49          ,X,27,   86          ,X,18,   NA
## 4 1948-10   NA                33          ,X,19,   79          ,X,02,   NA
## 5 1948-11   NA                31          ,X,30,   69          ,X,20,   NA
## 6 1948-12   NA                 8          ,X,27,   58          ,X,13,   NA
##   HDSD_ATTRIBUTES TAVG TAVG_ATTRIBUTES TMAX TMAX_ATTRIBUTES TMIN
## 1              NA                   NA                   NA
## 2            74.6             ,X 82.1            ,,,X 67.1
## 3            68.3             ,X 76.9            ,,,X 59.6
```

```
## 4                 55.9              ,X 63.8            ,,,X 47.9
## 5                 50.7              ,X 57.5            ,,,X 43.9
## 6                 36.9              ,X 43.8            ,,,X 29.9
##   TMIN_ATTRIBUTES
## 1
## 2             ,,,X
## 3             ,,,X
## 4             ,,,X
## 5             ,,,X
## 6             ,,,X
```

```
df_monthly$DATE <- as.Date(as.yearmon(df_monthly$DATE))
df_monthly$Year <- floor_date(df_monthly$Date)
```

```
plot(x = df_monthly$DATE, y = df_monthly$TAVG, type = "l")
```



Issue with Time and How to Solve it

1. Take consideration of Time Zone
2. Daylight Savings Time

How?

Use UTC or GMT Time

```
getwd()
```

```
## [1] "/Users/hyunjoonrhee/Downloads/Ex_Files_Time_Series_Modeling/Exercise Files"
```

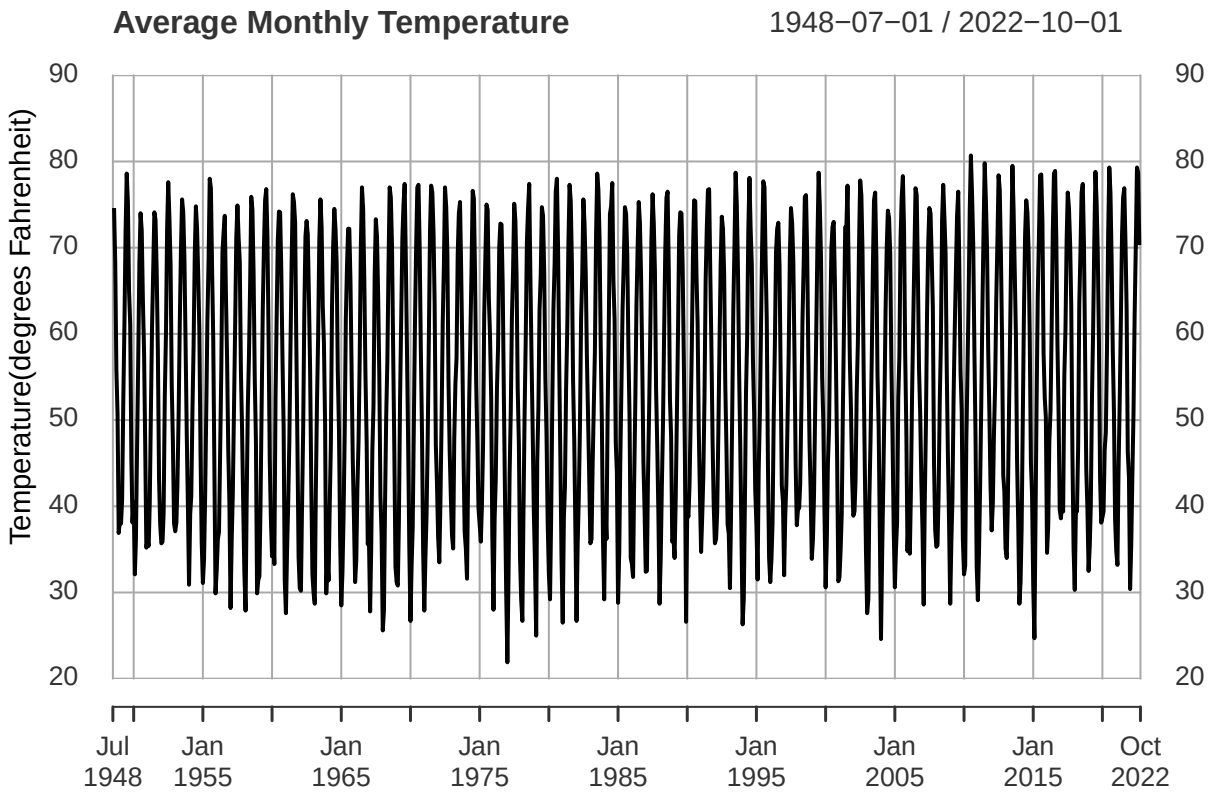
```
dir_name <- "Data Sources"
file_name_e <- "Raleigh-Durham_North_Carolina_area_electricity_demand.csv"
file_path_e <- paste(getwd(),dir_name,file_name_e,sep="/")
elec <- read.csv(file_path_e)
```

```
elec$LocalDatetime <- as.POSIXct(elec$Datetime, tz = "America/New_York")
elec$UTC <- as_datetime(elec$Datetime)
```

```
zoo_monthly <- zoo(df_monthly$TAVG, df_monthly$DATE)
xts_monthly <- xts(df_monthly$TAVG, df_monthly$DATE)
```

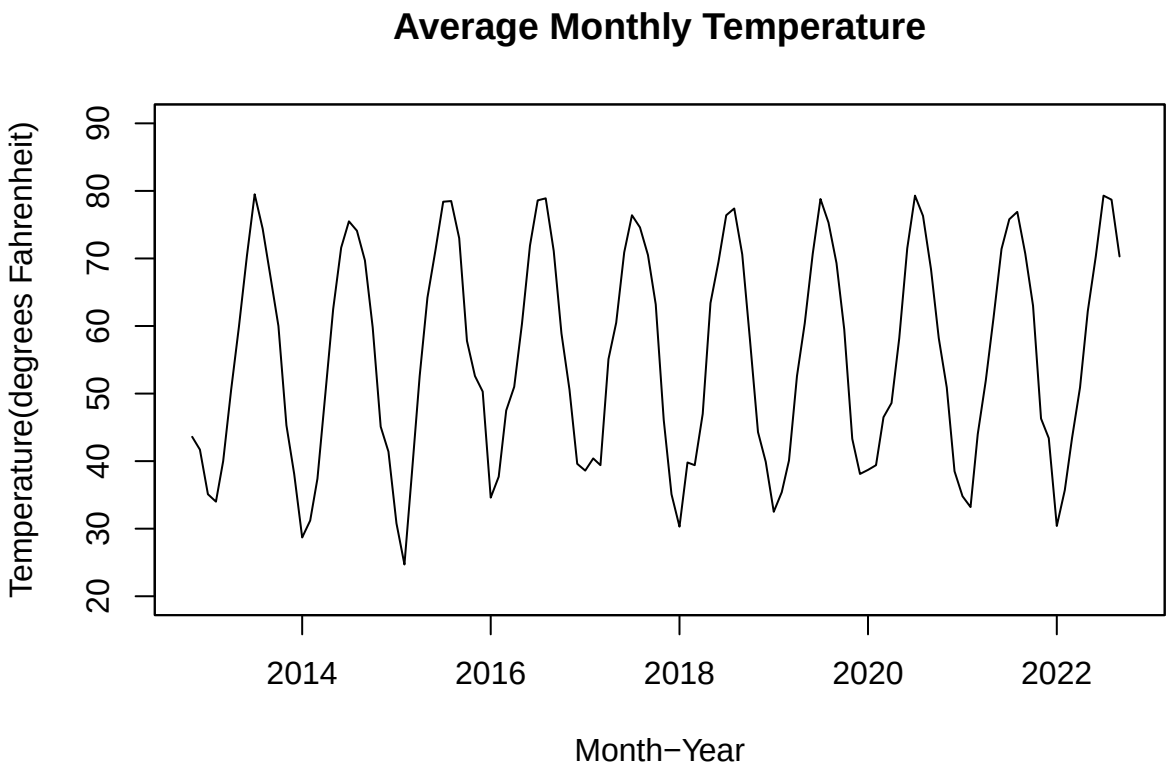
```
colnames(xts_monthly) <- c('Temperature')
#convert xts to zoo
#as.zoo(xts_monthly)
```

```
plot(xts_monthly, main = "Average Monthly Temperature", xlab=  "Month-Year",ylab = "Temperature(degrees
```



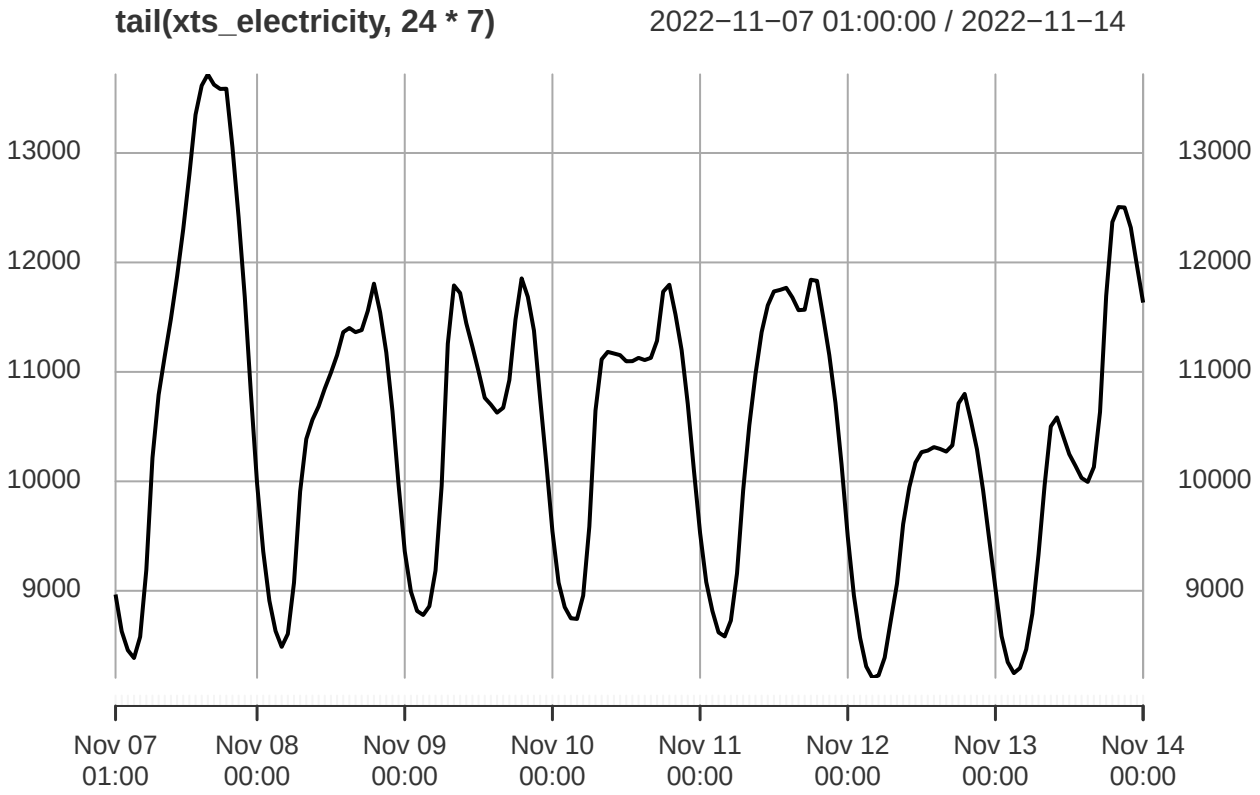
Notice hotter temp in hotter months, lower temp in colder months

```
#last 10 years
plot(tail(zoo_monthly,120), main = "Average Monthly Temperature", xlab=  "Month-Year",ylab = "Temperatu
```

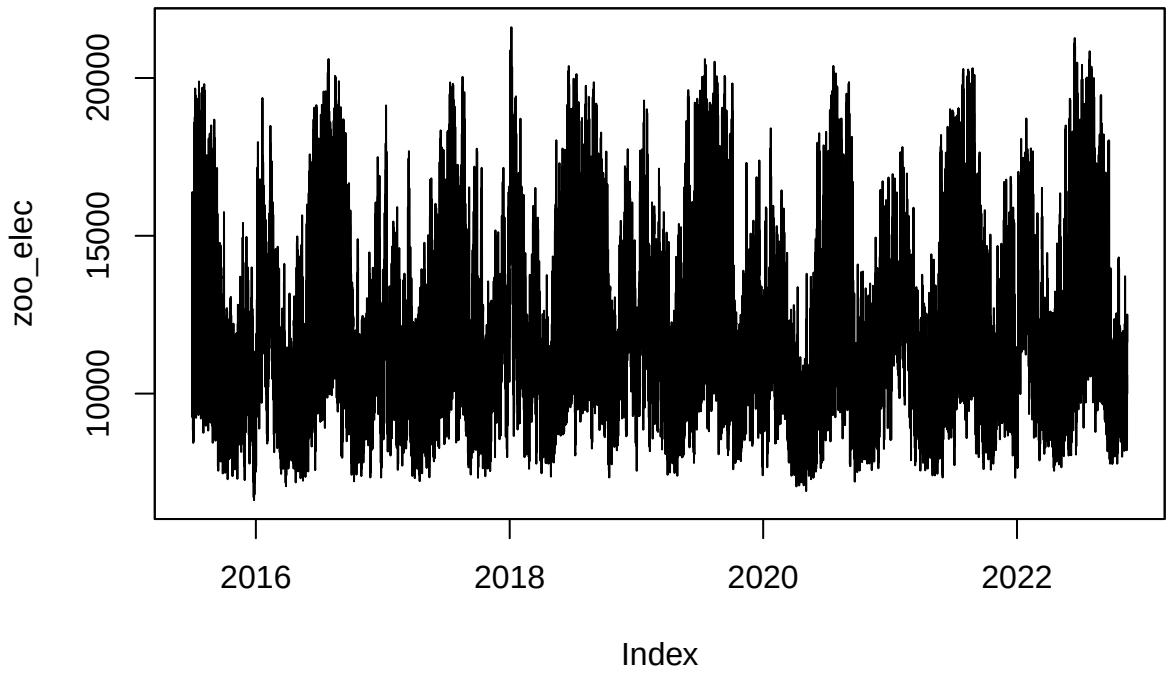



```
xts_electricity <- xts(elec$Demand, elec$LocalDatetime)
#in xts function first argument goes to the column and second goes to row
colnames(xts_electricity) <- c('Demand')
```

```
plot(tail(xts_electricity,24*7)) #how it oscillates for each week
```




```
zoo_elec <- zoo(elec$Demand, elec$UTC)
plot(zoo_elec)
```



xts and zoo elec won't line up because xts used local time and zoo used UTC time!!

```
df_monthly_again <- as.data.frame(xts_monthly)
df_monthly_again$Date <- rownames(df_monthly_again)
```

Look at Subset of the data Filtering data that we want to look at

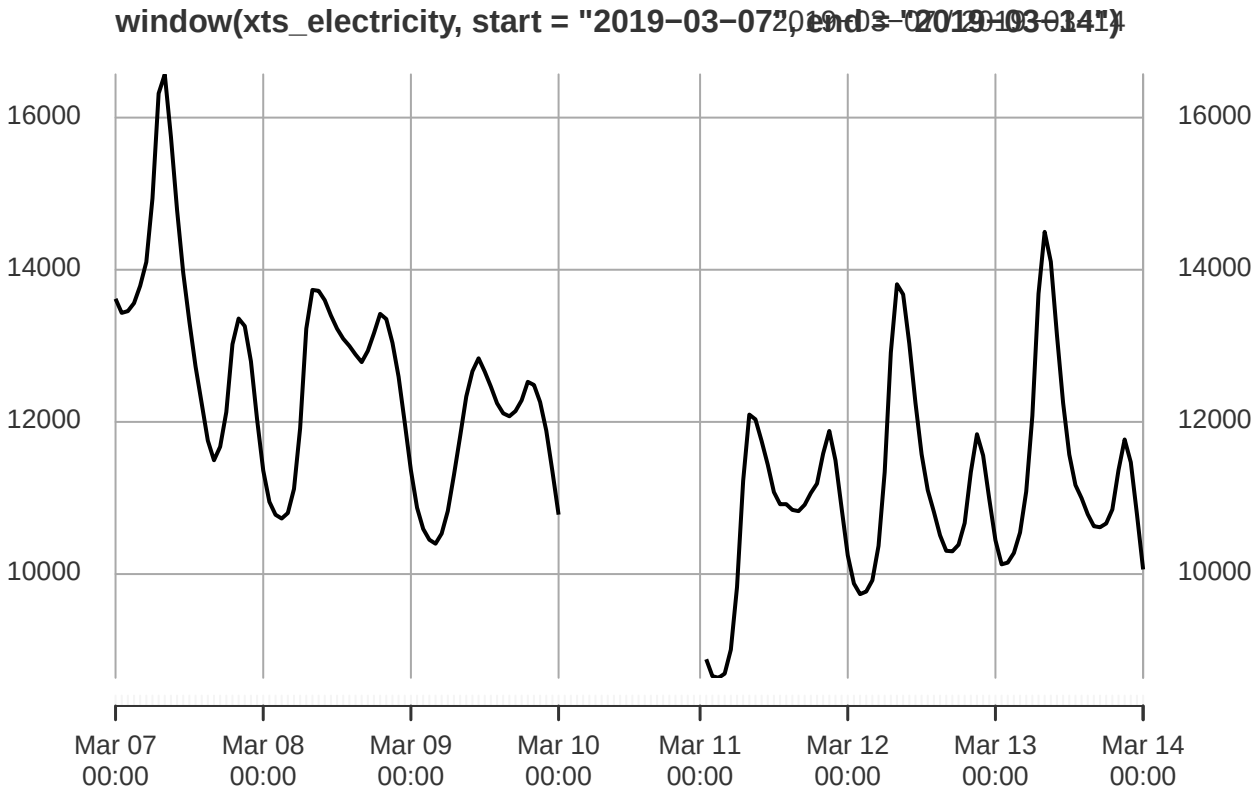
```
df_monthly[df_monthly$TAVG == 76.9,]
```

```
##          STATION                              NAME LATITUDE LONGITUDE ELEVATION
## NA          <NA>                              <NA>       NA        NA        NA
## 86   USW00094789 JFK INTERNATIONAL AIRPORT, NY US 40.63915  -73.7639       2.7
## 697  USW00094789 JFK INTERNATIONAL AIRPORT, NY US 40.63915  -73.7639       2.7
## 878  USW00094789 JFK INTERNATIONAL AIRPORT, NY US 40.63915  -73.7639       2.7
## NA.1        <NA>                              <NA>       NA        NA        NA
##            DATE CDSD CDSD_ATTRIBUTES EMNT EMNT_ATTRIBUTES EMXT EMXT_ATTRIBUTES
## NA         <NA>   NA            <NA>   NA            <NA>   NA            <NA>
## 86   1955-08-01  944               Z   59          ,Z,29,   99          ,Z,20,
## 697  2006-07-01  545               0   62         ,0,07,+   95          ,0,18,
## 878  2021-08-01  950               W   62          ,W,02,   93          ,W,27,
## NA.1       <NA>   NA            <NA>   NA            <NA>   NA            <NA>
##      HDSD HDSD_ATTRIBUTES TAVG TAVG_ATTRIBUTES TMAX TMAX_ATTRIBUTES TMIN
## NA     NA            <NA>   NA            <NA>   NA            <NA>   NA
## 86      0               Z 76.9              ,Z 83.2            ,,,Z 70.5
## 697     0               0 76.9              ,0 84.1            ,,,0 69.6
## 878     1               W 76.9              ,W 83.4            ,,,W 70.5
## NA.1   NA            <NA>   NA            <NA>   NA            <NA>   NA
##      TMIN_ATTRIBUTES
## NA              <NA>
## 86              ,,,Z
## 697             ,,,0
## 878             ,,,W
## NA.1            <NA>
```

Window function works on xts and zoo

```
xts_monthly_2010s <- window(xts_monthly,start = "2010-01-01", end = "2019-12-31")
```

```
plot(window(xts_electricity, start = "2019-03-07", end="2019-03-14"))
```



Notice that there isn't any data

how to fill in the empty value in excel it doesn't consider empty values

```
window(zoo_monthly,start = "1948-07-01", end = "1948-12-01")
```

```
## 1948-07-01 1948-08-01 1948-09-01 1948-10-01 1948-11-01 1948-12-01
##         NA       74.6       68.3       55.9       50.7       36.9
```

```
mean(window(zoo_monthly,start = "1948-07-01", end = "1948-12-01"))
```

```
## [1] NA
```

```
mean(window(zoo_monthly,start = "1948-08-01", end = "1948-12-01"))
```

```
## [1] 57.28
```

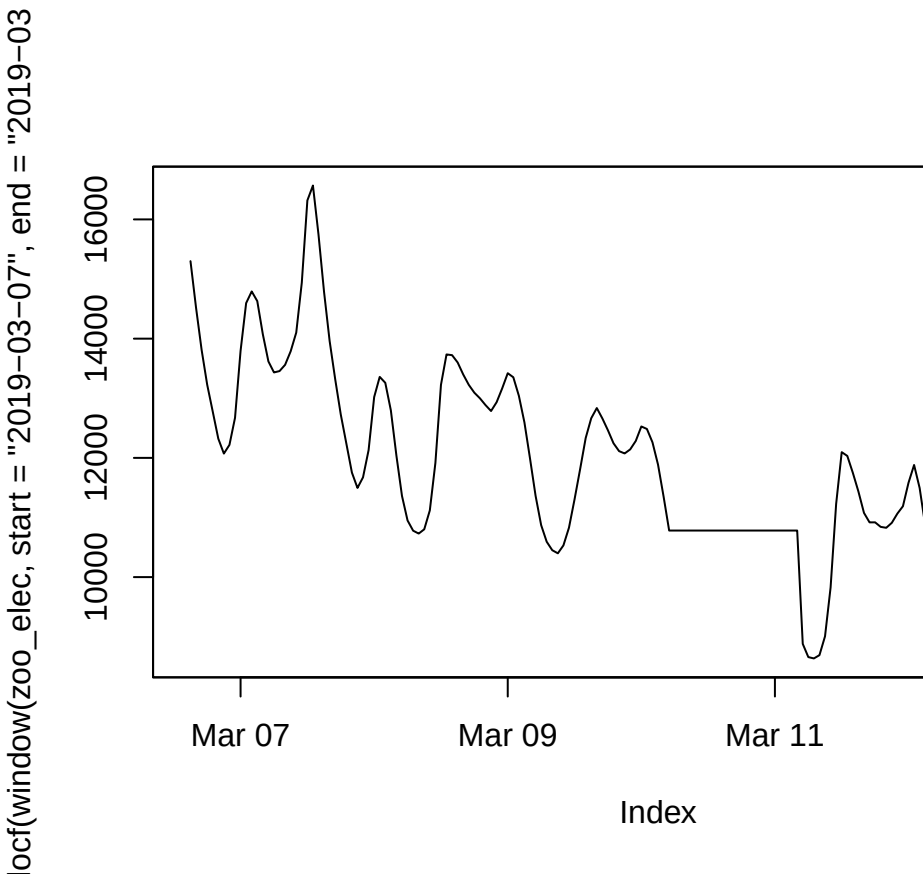
```
summary(xts_monthly)
```

```
##       Index             Temperature
##  Min.   :1948-07-01   Min.   :21.90
##  1st Qu.:1967-01-24   1st Qu.:39.83
##  Median :1985-08-16   Median :54.10
##  Mean   :1985-08-16   Mean   :54.24
##  3rd Qu.:2004-03-08   3rd Qu.:69.28
##  Max.   :2022-10-01   Max.   :80.70
##                       NA's   :2
```

Notice there are 2 NA's only

```
xts_monthly <- na.omit(xts_monthly)
```

```
plot(na.locf(window(zoo_elec, start = "2019-03-07", end="2019-03-14")))
```



But how do you fill in the empty ones?
This carries last observation forward until the new data value and fill in the NA with it

```
zoo_update <-plot(window(xts_electricity, start = "2019-03-07", end="2019-03-14"))
```

```
apply.daily(xts_electricity,sum)
```

Pivot table

```
## Warning: object timezone (America/New_York) is different from system timezone ()
##   NOTE: set 'options(xts_check_TZ = FALSE)'to disable this warning
##     This note is displayed once per session

##                     Demand
```

```
## 2015-07-01 23:00:00 296549
## 2015-07-02 23:00:00 297387
## 2015-07-03 23:00:00 264682
## 2015-07-04 23:00:00 266008
## 2015-07-05 23:00:00 274607
## 2015-07-06 23:00:00 319825
## 2015-07-07 23:00:00 342596
## 2015-07-08 23:00:00 351333
## 2015-07-09 23:00:00 365122
## 2015-07-10 23:00:00 364505
##                  ...
## 2022-11-05 23:00:00 238726
## 2022-11-06 23:00:00 254265
## 2022-11-07 23:00:00 271707
## 2022-11-08 23:00:00 249790
## 2022-11-09 23:00:00 253415
## 2022-11-10 23:00:00 252669
## 2022-11-11 23:00:00 254153
## 2022-11-12 23:00:00 231178
## 2022-11-13 23:00:00 245093
## 2022-11-14 00:00:00  11633
```

```
apply.weekly(xts_electricity,sum)
```

```
## Warning: object timezone (America/New_York) is different from system timezone ()
```

```
##                      Demand
## 2015-07-05 19:00:00 1347673
## 2015-07-12 19:00:00 2379568
## 2015-07-19 19:00:00 2327030
## 2015-07-26 19:00:00 2394999
## 2015-08-02 19:00:00 2395788
## 2015-08-09 19:00:00 2321042
## 2015-08-16 19:00:00 2205565
## 2015-08-23 19:00:00 2261134
## 2015-08-30 19:00:00 2127589
## 2015-09-06 19:00:00 2204871
##                  ...
## 2022-09-18 19:00:00 2012965
## 2022-09-25 19:00:00 1990418
## 2022-10-02 19:00:00 1693056
## 2022-10-09 19:00:00 1694820
## 2022-10-16 19:00:00 1714644
## 2022-10-23 19:00:00 1829588
## 2022-10-30 19:00:00 1720000
## 2022-11-06 18:00:00 1732436
## 2022-11-13 18:00:00 1752829
## 2022-11-14 00:00:00   73298
```

```
apply.monthly(xts_electricity,sum)
```

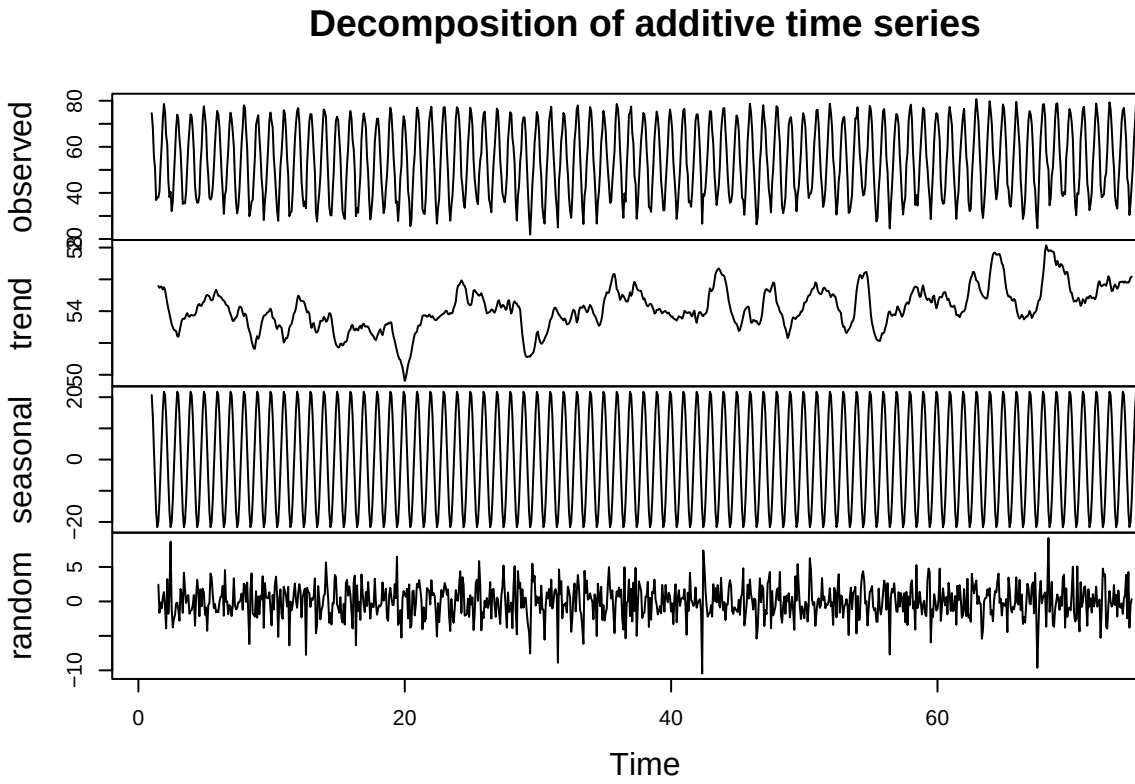
```
## Warning: object timezone (America/New_York) is different from system timezone ()
```

```
apply.quarterly(xts_monthly,mean)
```

```
##            Temperature
## 1948-09-01    71.45000
## 1948-12-01    47.83333
## 1949-03-01    38.96667
## 1949-06-01    61.56667
## 1949-09-01    72.93333
## 1949-12-01    48.16667
## 1950-03-01    36.23333
## 1950-06-01    57.46667
## 1950-09-01    69.93333
## 1950-12-01    46.86667
##        ...
## 2020-06-01    59.43333
## 2020-09-01    74.66667
## 2020-12-01    49.20000
## 2021-03-01    37.33333
## 2021-06-01    61.50000
## 2021-09-01    74.46667
## 2021-12-01    50.90000
## 2022-03-01    36.50000
## 2022-06-01    61.13333
## 2022-09-01    76.10000
```

```
xts_yearly <- apply.yearly(xts_monthly,start = "1949-01-01", end = "2021-12-31",mean)
```

```
plot(decompose(ts(xts_monthly,frequency = 12)),ylim=c(30,100))
```

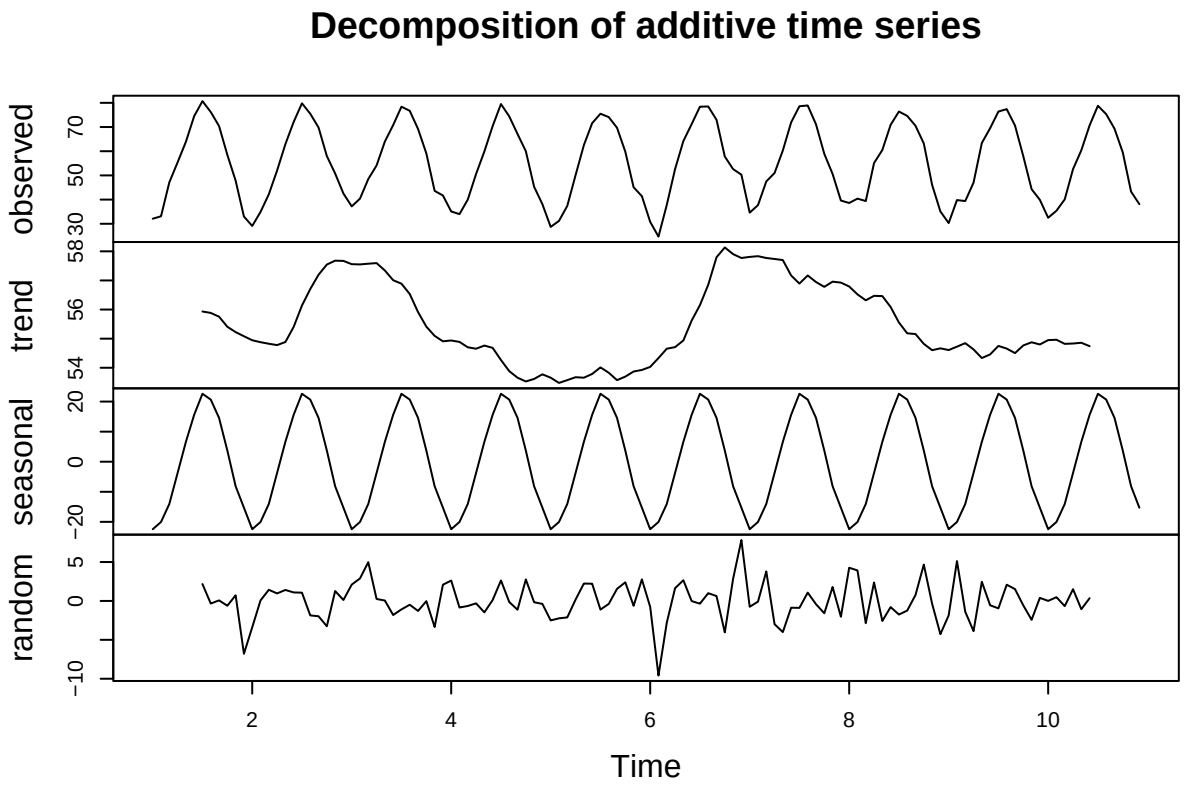


Line chart breakdown

R doesn't know frequency –> convert time series with ts function

1. Original Time series
2. Trend fluctuate 50 ~58 –> smaller range narrower b/c it removes seasonality from the range
3. Seasonal Look at the diff between hot and cold weather
4. Random represent abnormalities in the data(look at the spike)

```
plot(decompose(ts(window(xts_monthly,start = "2010-01-01", end = "2019-12-31"),frequency = 12)))
```



ARIMA Model

Autoregressive integrated moving average =arima(, order = c(p,d,q)) p,d,q vector

ARIMA models: fitted

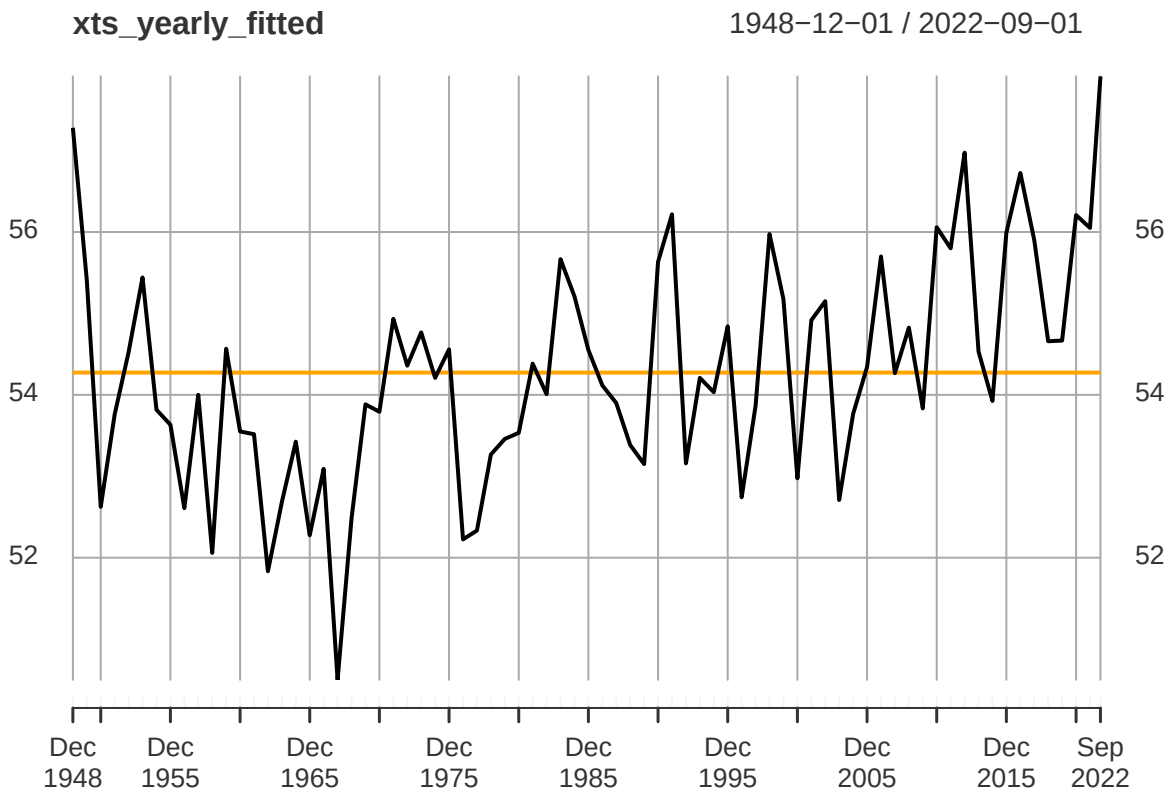
```
results_yearly <-arima(xts_yearly,order=c(0,0,0))
```

```
mean(xts_yearly)
```

```
## [1] 54.27333
```

```
xts_yearly_fitted <- merge(xts_yearly, xts(fitted(results_yearly),index(xts_yearly)))
colnames(xts_yearly_fitted) <- c('Temperature','Fitted')
```

```
plot(xts_yearly_fitted,col=c('black','orange'))
```



calculate std dev

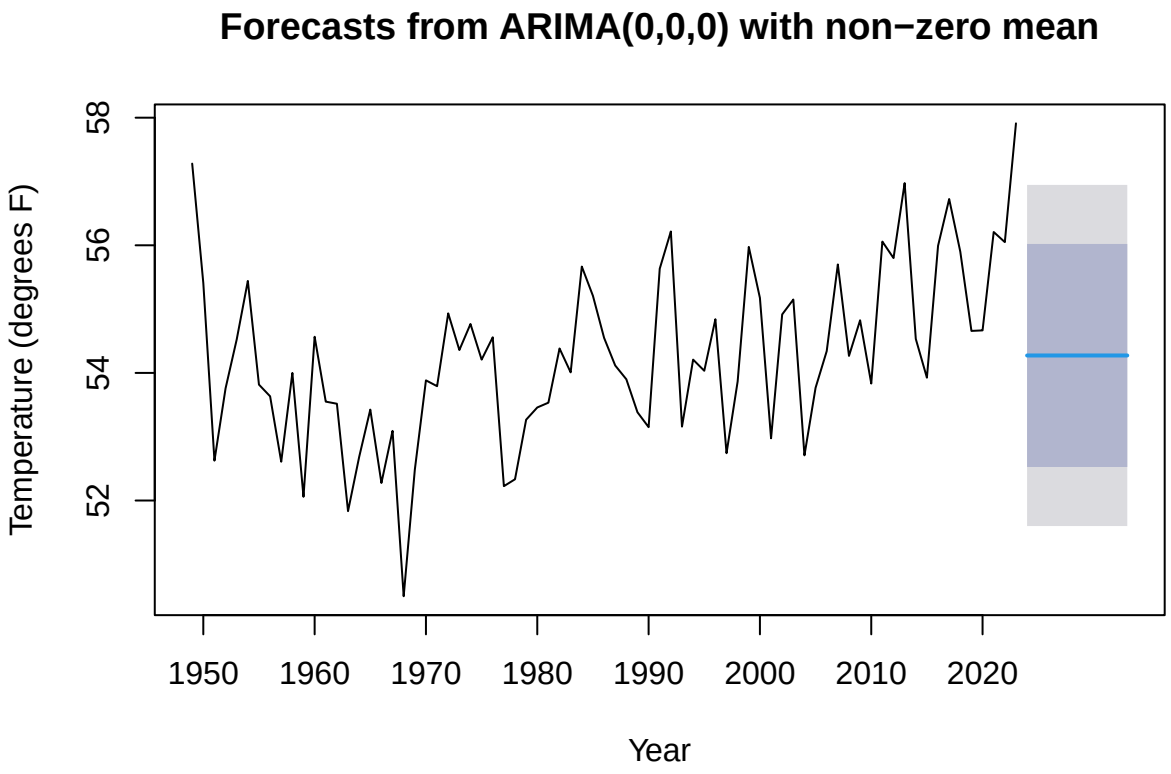
```
sd(xts_yearly)
```

```
## [1] 1.37298
```

```
predict(results_yearly,10)
```

```
## $pred
## Time Series:
## Start = 76
## End = 85
## Frequency = 1
##  [1] 54.27333 54.27333 54.27333 54.27333 54.27333 54.27333 54.27333 54.27333
##  [9] 54.27333 54.27333
##
## $se
## Time Series:
## Start = 76
## End = 85
## Frequency = 1
##  [1] 1.363796 1.363796 1.363796 1.363796 1.363796 1.363796 1.363796 1.363796
##  [9] 1.363796 1.363796
```

```
plot(forecast(results_yearly,10), xaxt = 'n', xlab="Year",ylab="Temperature (degrees F)")
axis(1, at = c(2,12,22,32,42,52,62,72), labels=c(1950,1960,1970,1980,1990,2000,2010,2020))
```




```
#xaxt='n' removes tickmarks
#reference x axis = axis(1,)
```

Forecasted future temp as Blue line 80% CI blue box, 90% CI grey box

Aggregating data -> github.com/helenrmwall/Time-Series-Models Use hourly electricity demand for San Diego stored on GitHub and in exercise files Determine which week over an entire time period has highest total electricity demand set tz = "America/Los_Angeles"

```
file_name_sd <- "San Diego electricity demand.csv"
file_path_sd <- paste(getwd(),dir_name,file_name_sd,sep="/")
df_sandiego <- read.csv(file_path_sd)
```

```
df_sandiego$LocalDateTime <- as.POSIXct(df_sandiego$Datetime,tz="America/Los_Angeles")
```

```
xts_sandiego <- xts(df_sandiego$Demand,df_sandiego$LocalDateTime)
colnames(xts_sandiego) <- c('demand')
```

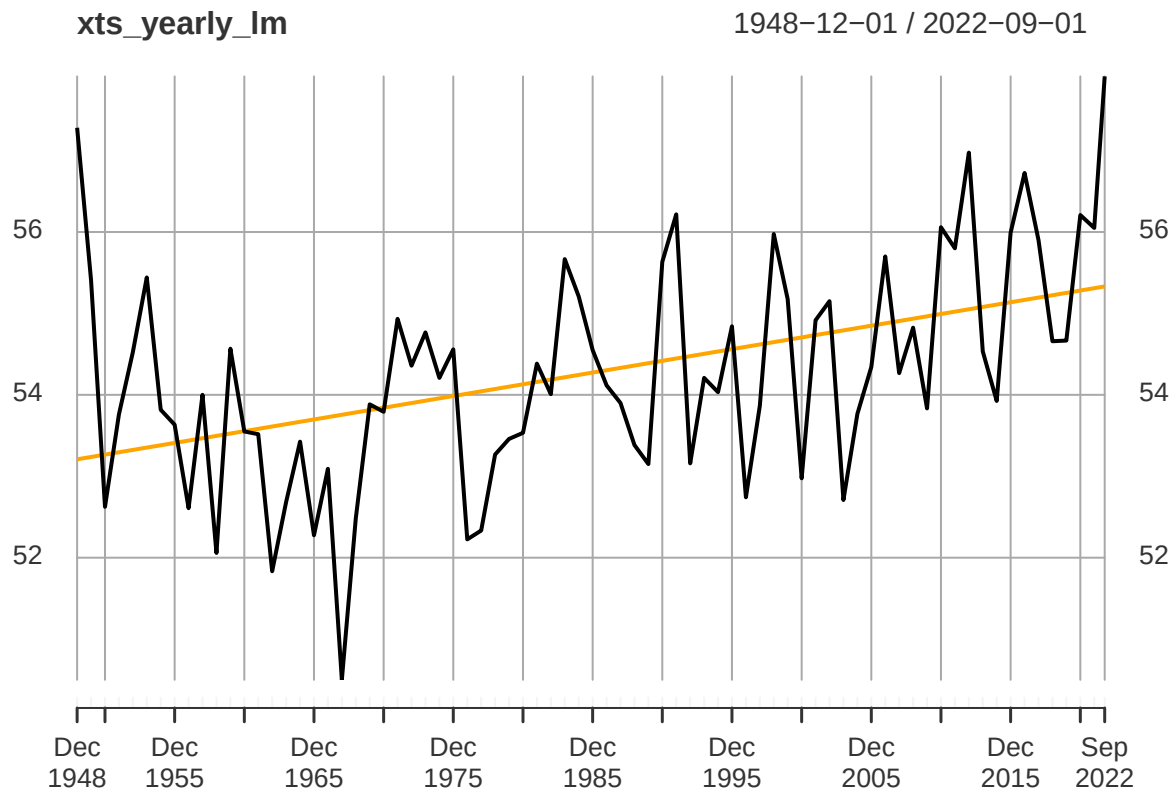
```
plot(apply.weekly(xts_sandiego,sum))
```


```
df_sandiego_weekly <- as.data.frame(apply.weekly(xts_sandiego,sum))
df_sandiego_weekly$Date <- as.Date(rownames(df_sandiego_weekly))
df_sandiego_weekly[df_sandiego_weekly$demand == max(df_sandiego_weekly$demand),]
```

```
##                      demand       Date
## 2022-09-11 16:00:00 544787 2022-09-11
```

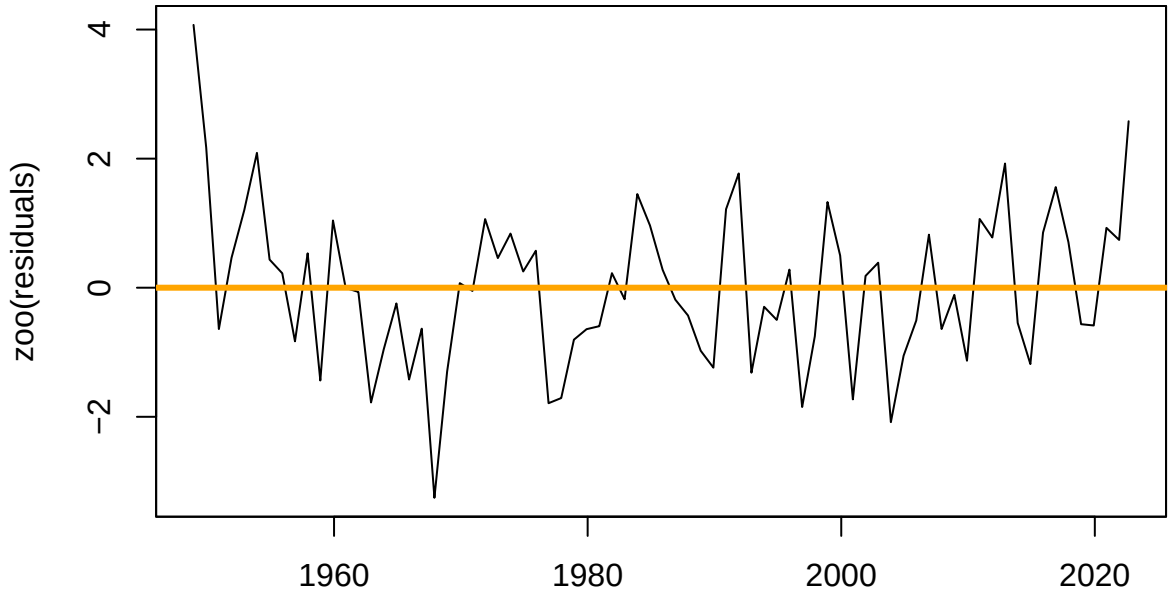
```
m <- lm(coredata(xts_yearly) ~ index(xts_yearly))
xts_yearly_lm <- merge(xts_yearly,xts(predict(m, newdata = xts_yearly, response="type"),index(xts_yearly
colnames(xts_yearly_lm) <- c('Temperature','Fitted')
```

```
plot(xts_yearly_lm, col=c('black','orange'))
```



Linear Regression

```
residuals <- xts(resid(m),index(xts_yearly))
plot(zoo(residuals))
abline(h=0,col='orange',lwd=3)
```



Residuals

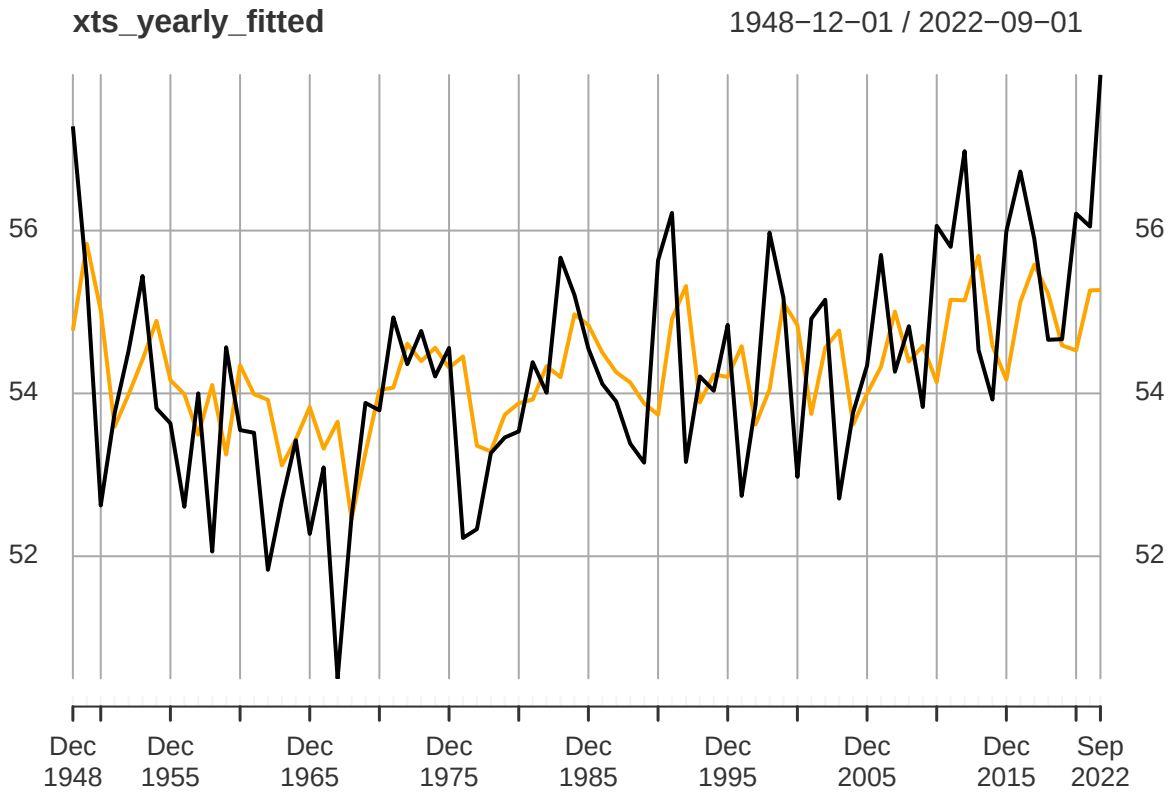
Lag Delay or move a time series by n periods of time Line up separate time data observatiosn Bootstrap dat within single time series

```
lead(xts_yearly,1)
#lag(xts_yearly,1)
```

```
xts_monthly_lagged <- xts_monthly
for (i in 1:12) {
  xts_monthly_lagged <- merge(xts_monthly_lagged,lag(xts_monthly,i))
}
```

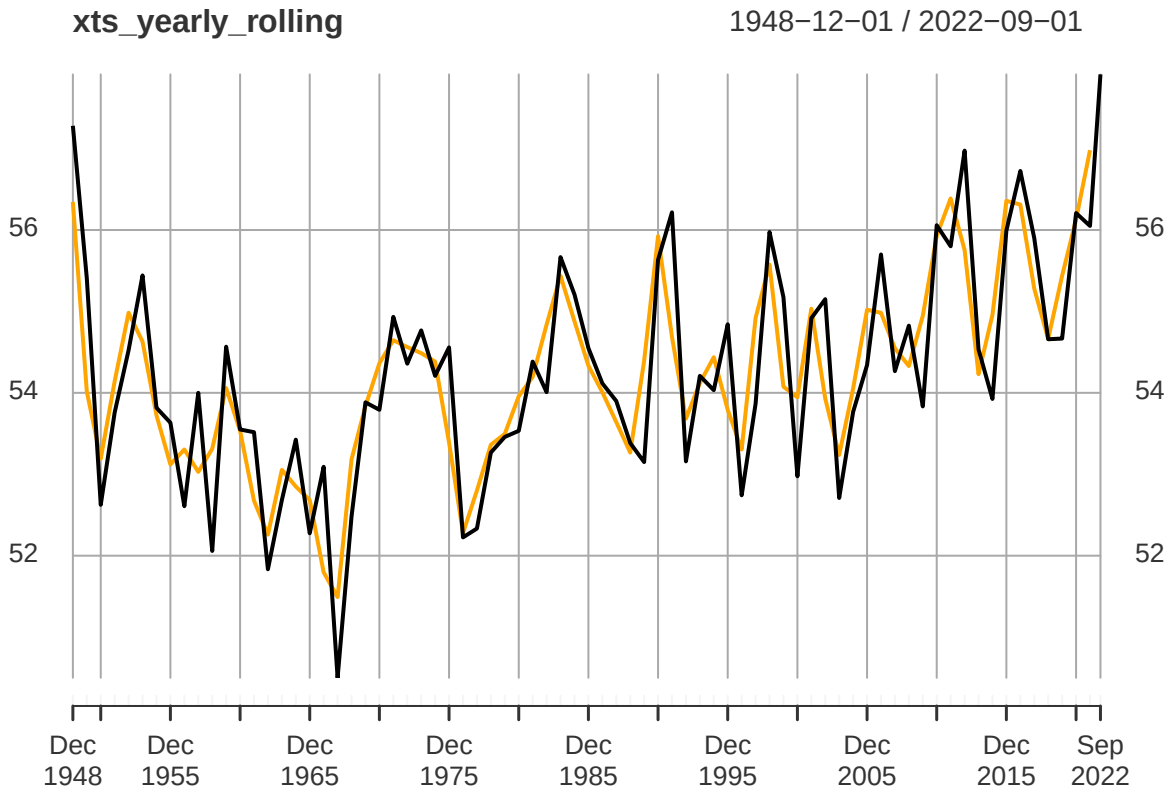
Autoregression Autoregression coefficient of 1 = linear fit of current point and the point right before it = 2 points & 1 line

```
results_yearly <-arima(xts_yearly,order=c(2,0,0))
xts_yearly_fitted <- merge(xts_yearly, xts(fitted(results_yearly),index(xts_yearly)))
colnames(xts_yearly_fitted) <- c('Temperature','Fitted')
plot(xts_yearly_fitted,col=c('black','orange'))
```

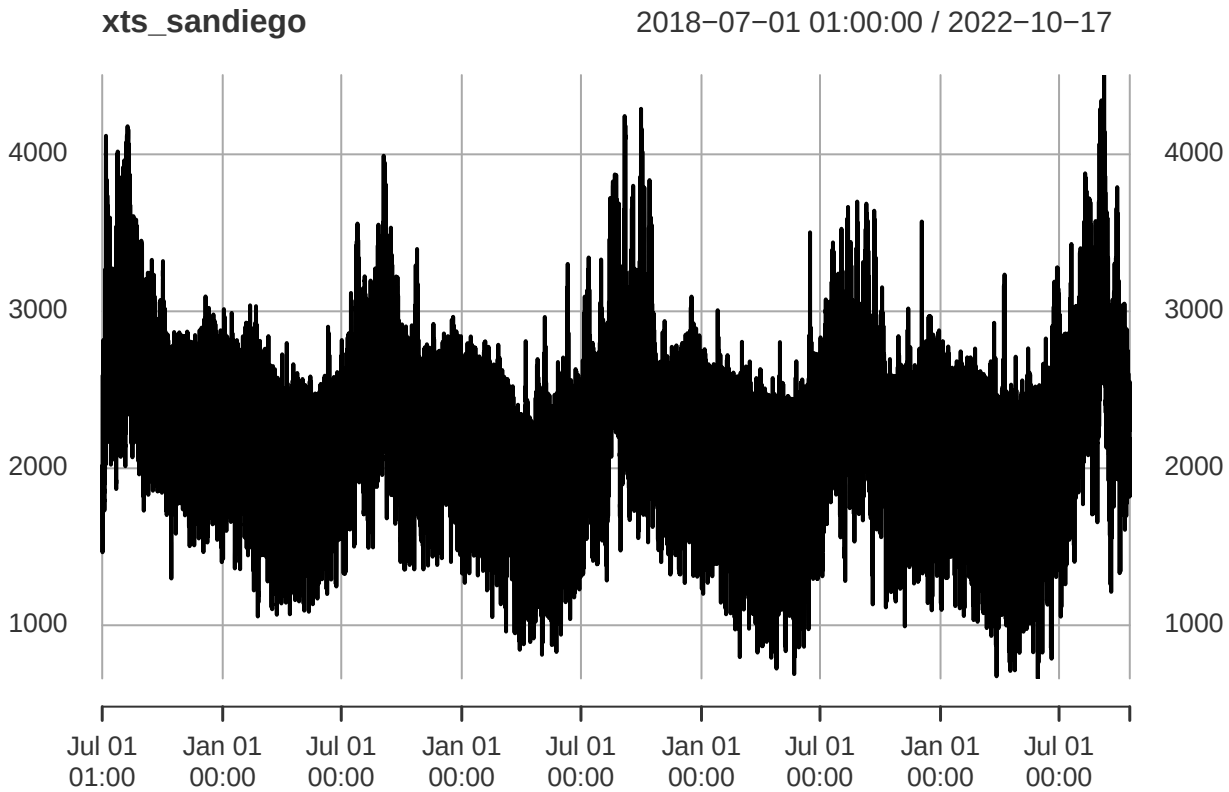


Moving Averages, Rolling Averages Even out fluctuations within time seires models moving average of period of 2 –> use from one previous to one future (total of 3)

```
xts_yearly_rolling <- merge(xts_yearly,rollmean(xts_yearly,2,align='left'))
plot(xts_yearly_rolling,col=c('black','orange'))
```




```
plot(xts_sandiego)
```

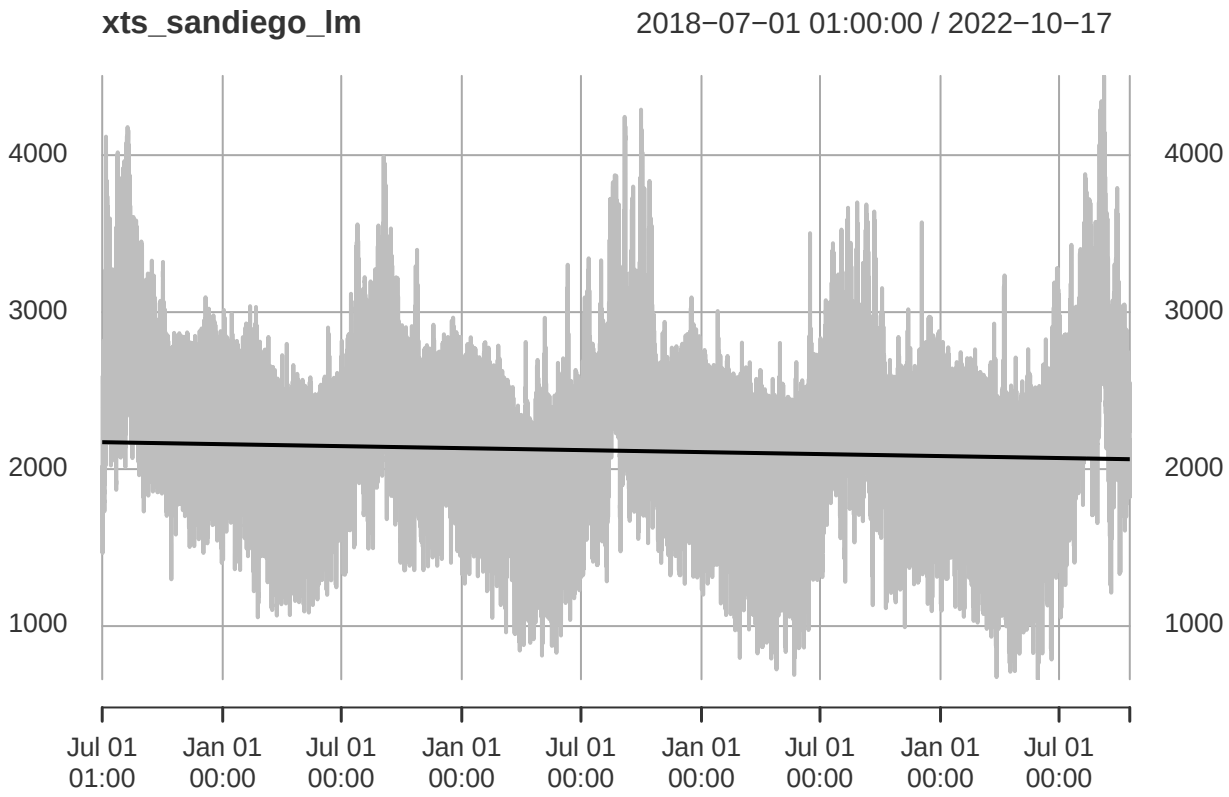


```
m_sandiego <- lm(coredata(xts_sandiego)~index(xts_sandiego))
m_sandiego
```

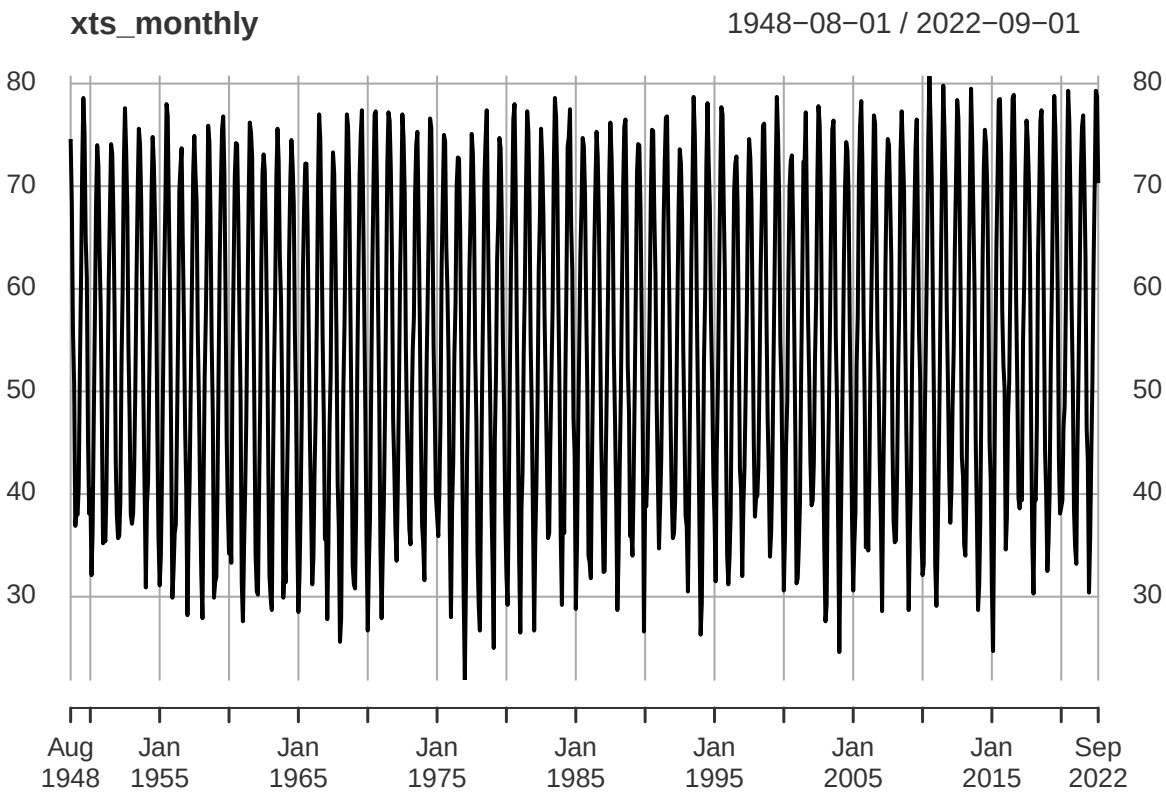
```
##
## Call:
## lm(formula = coredata(xts_sandiego) ~ index(xts_sandiego))
##
## Coefficients:
##          (Intercept)  index(xts_sandiego)
##            3.396e+03           -8.000e-07
```

```
xts_sandiego_lm <- merge(xts(predict(m_sandiego,newdata=xts_sandiego,response='type'),index(xts_sandiego
```

```
colnames(xts_sandiego_lm) <- c('Fitted','Demand')
plot(xts_sandiego_lm,col=c('black','grey'))
```

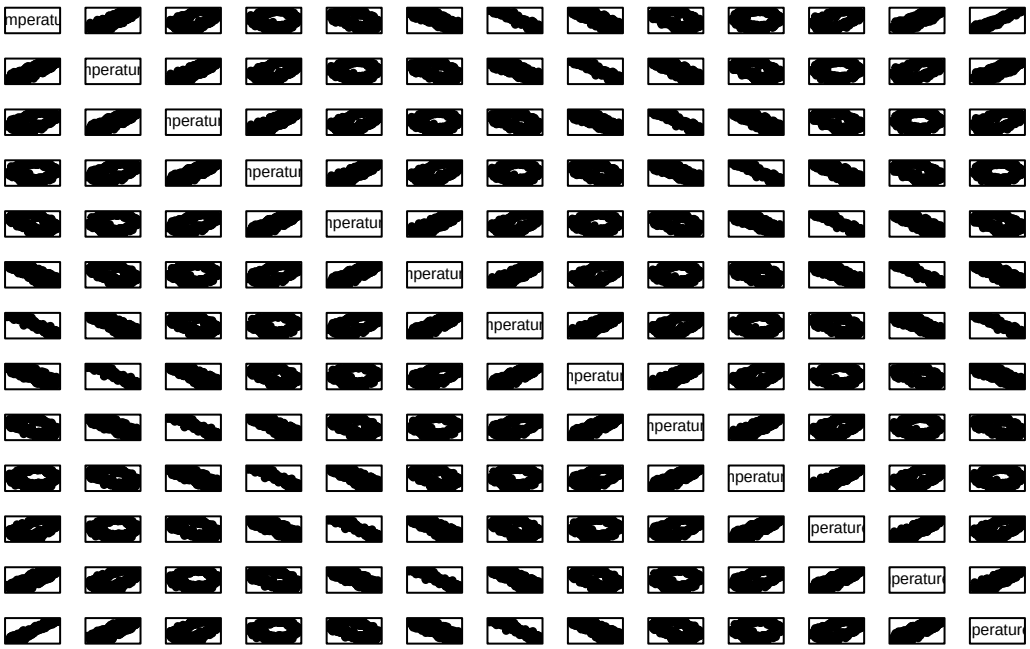



```
plot(xts_monthly)
```



Autocorrelation

```
pairs(as.data.frame(na.omit(xts_monthly_lagged)),xaxt='n',yaxt='n',pch=20)
```



```
#cor(na.omit(xts_monthly_lagged))
acf(xts_monthly,plot=FALSE,lag.max = 12)
```

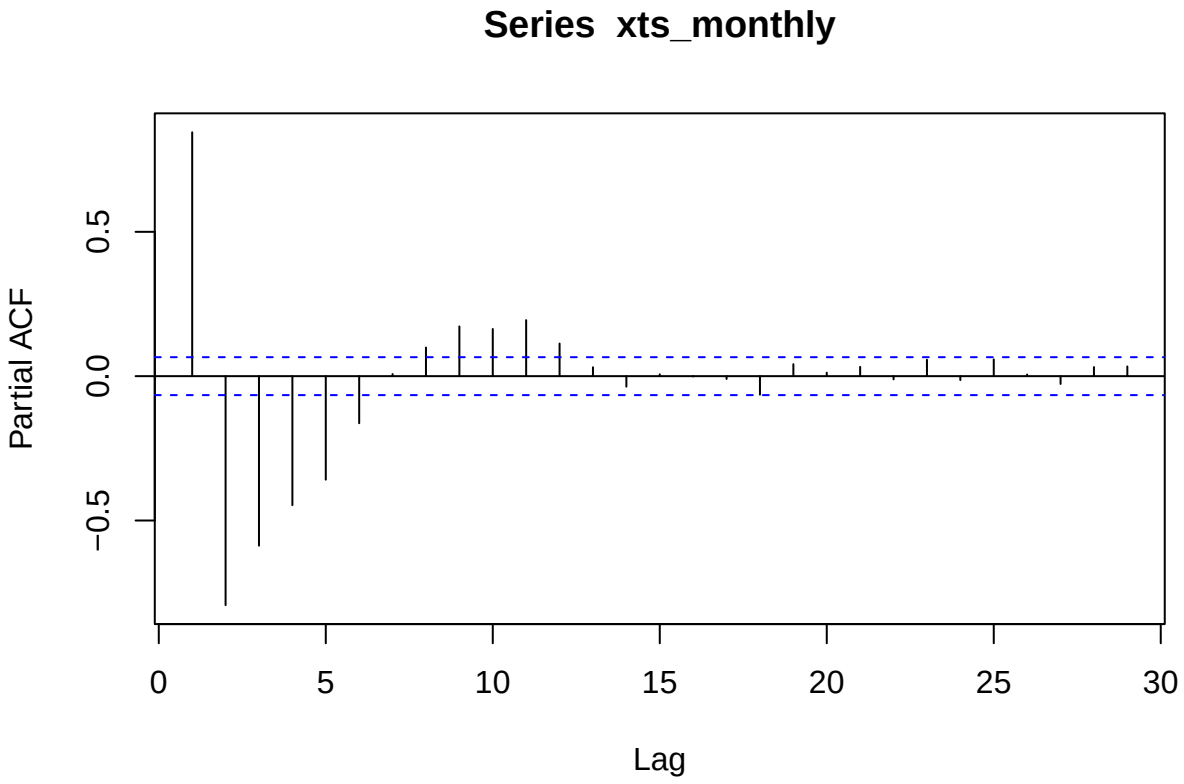
```
##
```

```
## Autocorrelations of series 'xts_monthly', by lag
##
##     0     1     2     3     4     5     6     7     8     9    10
## 1.000 0.845 0.487 0.005 -0.475 -0.828 -0.956 -0.827 -0.476 0.003 0.480
##    11    12
## 0.831 0.958
```

This compares value of current to the subsequent month. For example, it compares 1st month to 2nd month, 2nd to 3rd, ...

Partial Autocorrelation Steps between the lags of the model

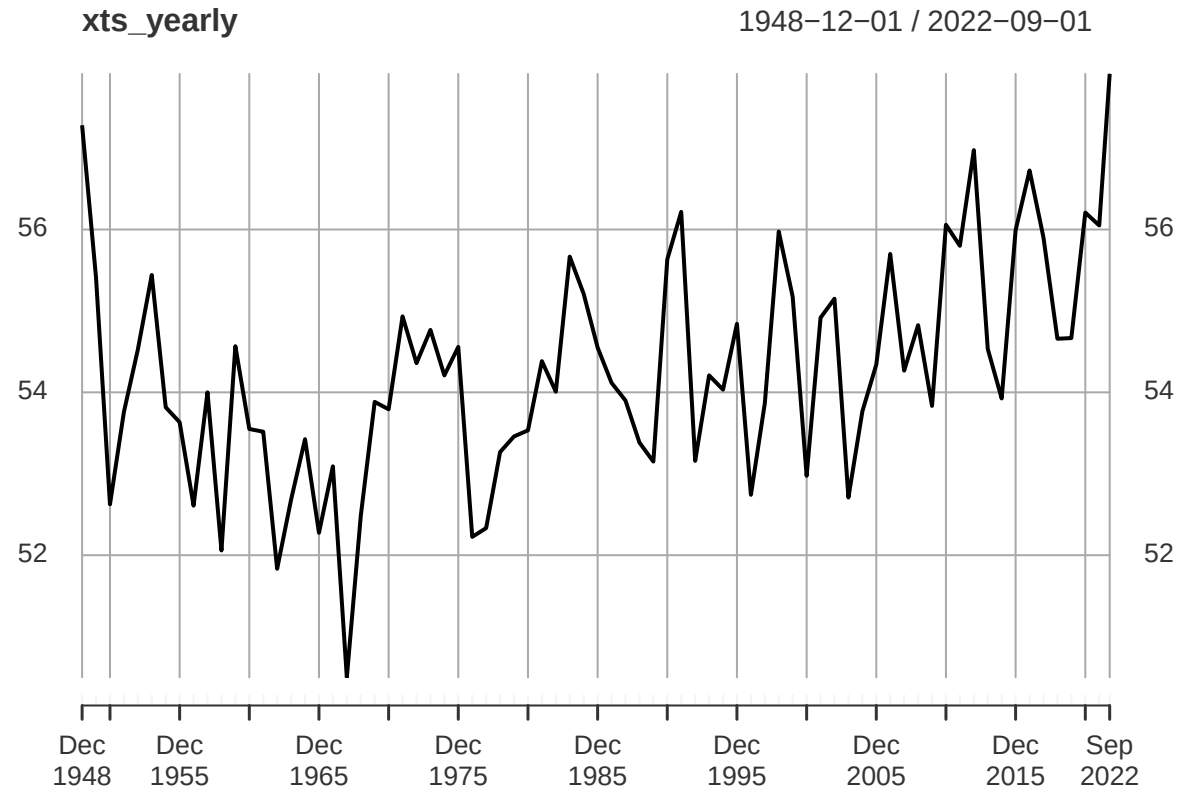
```
pacf(xts_monthly)
```




```
pacf(xts_monthly,plot=FALSE,lag.max=12)
```

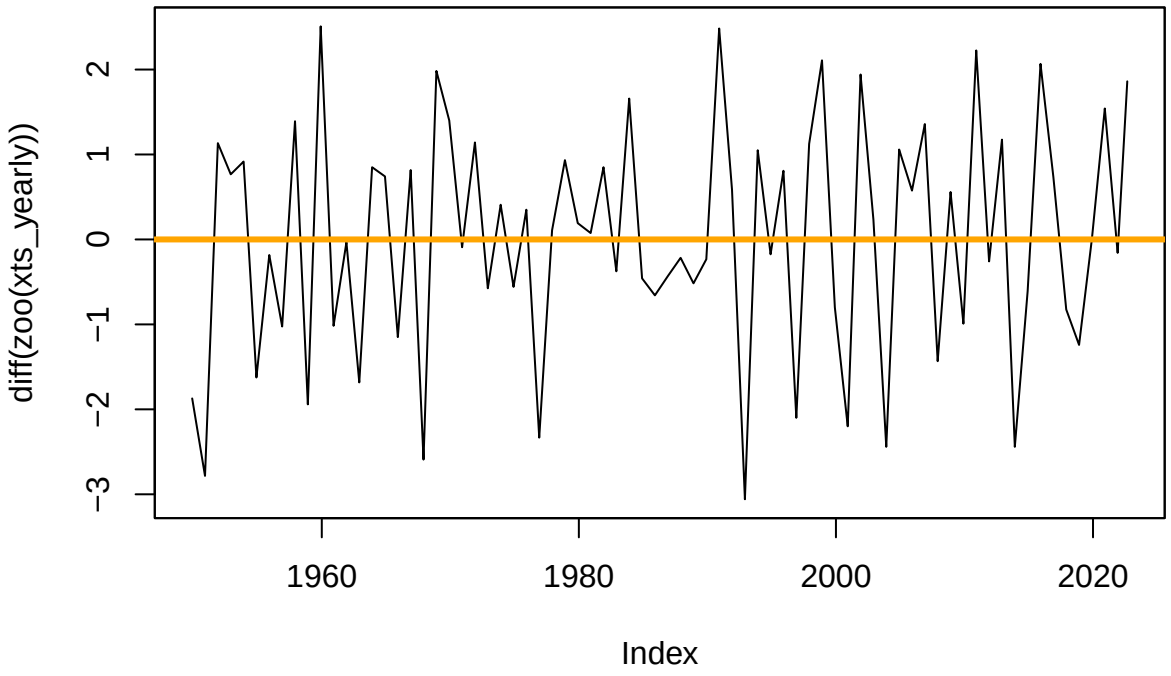
```
##
## Partial autocorrelations of series 'xts_monthly', by lag
##
##     1     2     3     4     5     6     7     8     9    10    11
## 0.845 -0.793 -0.587 -0.447 -0.359 -0.163 0.007 0.099 0.172 0.163 0.194
##    12
## 0.113
```

```
plot(xts_yearly)
```

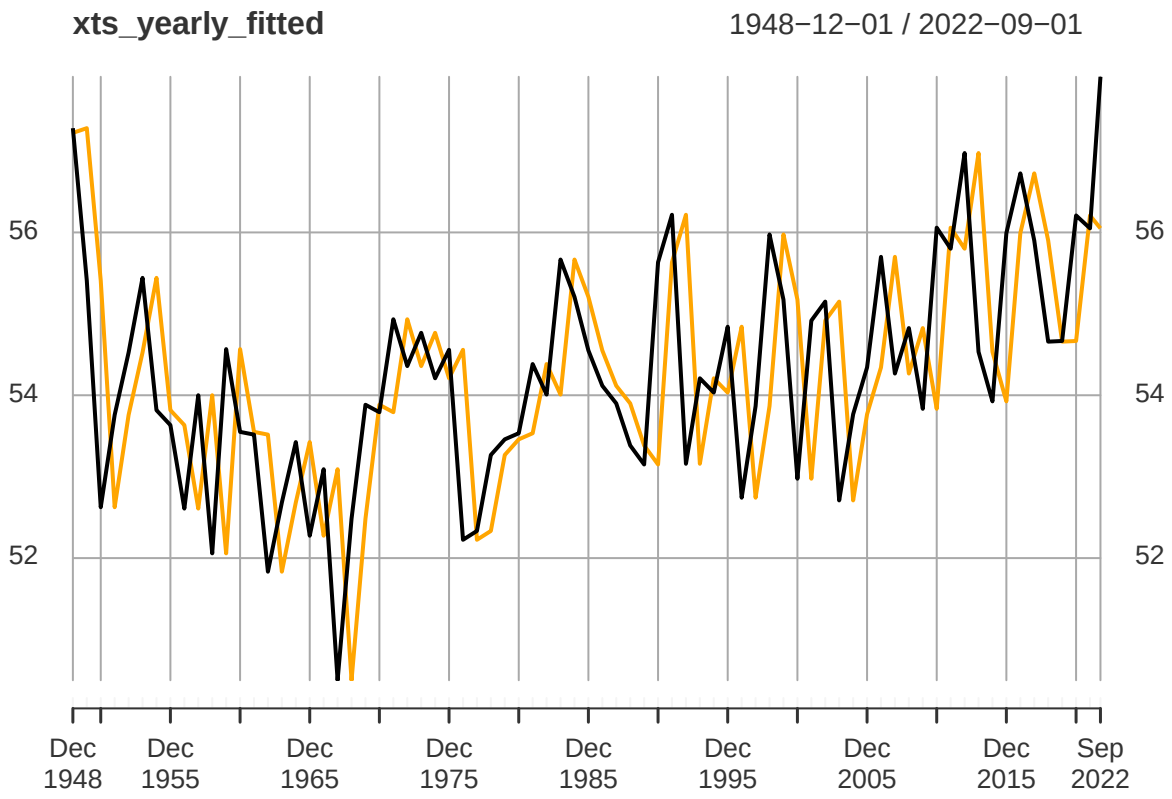


Stationarities

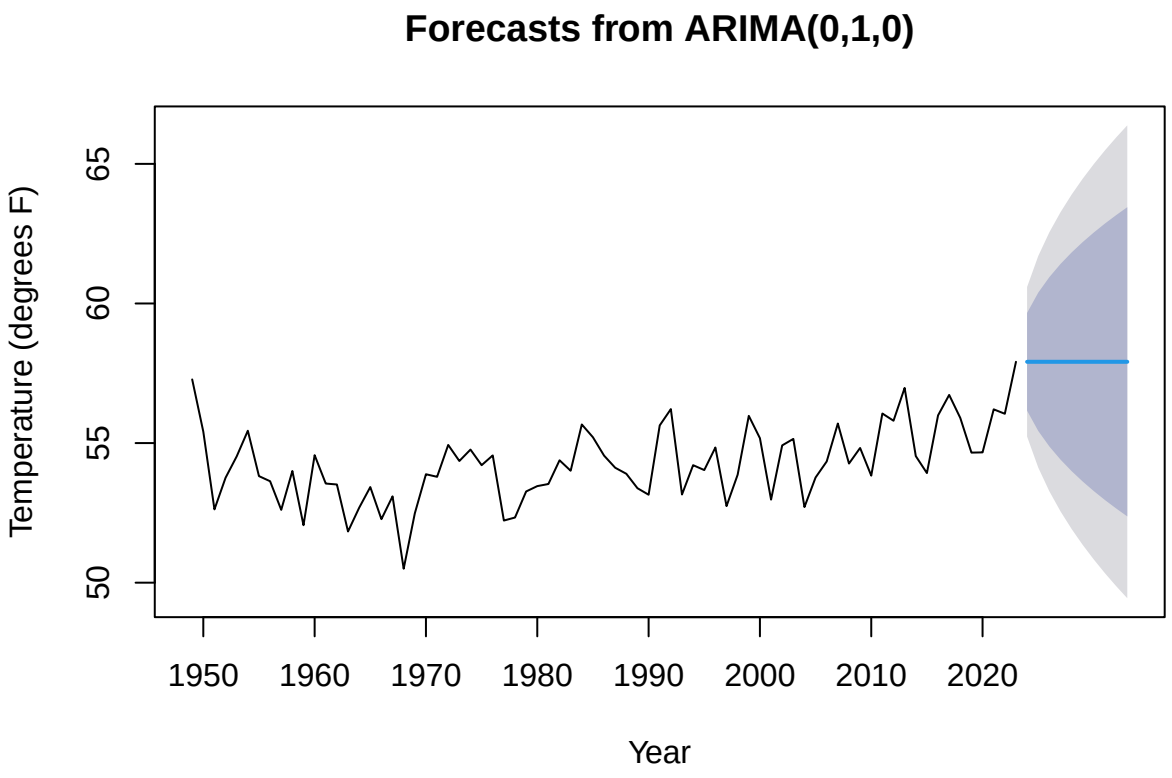
```
plot(diff(zoo(xts_yearly)))
abline(h=0,col='orange',lwd=3)
```



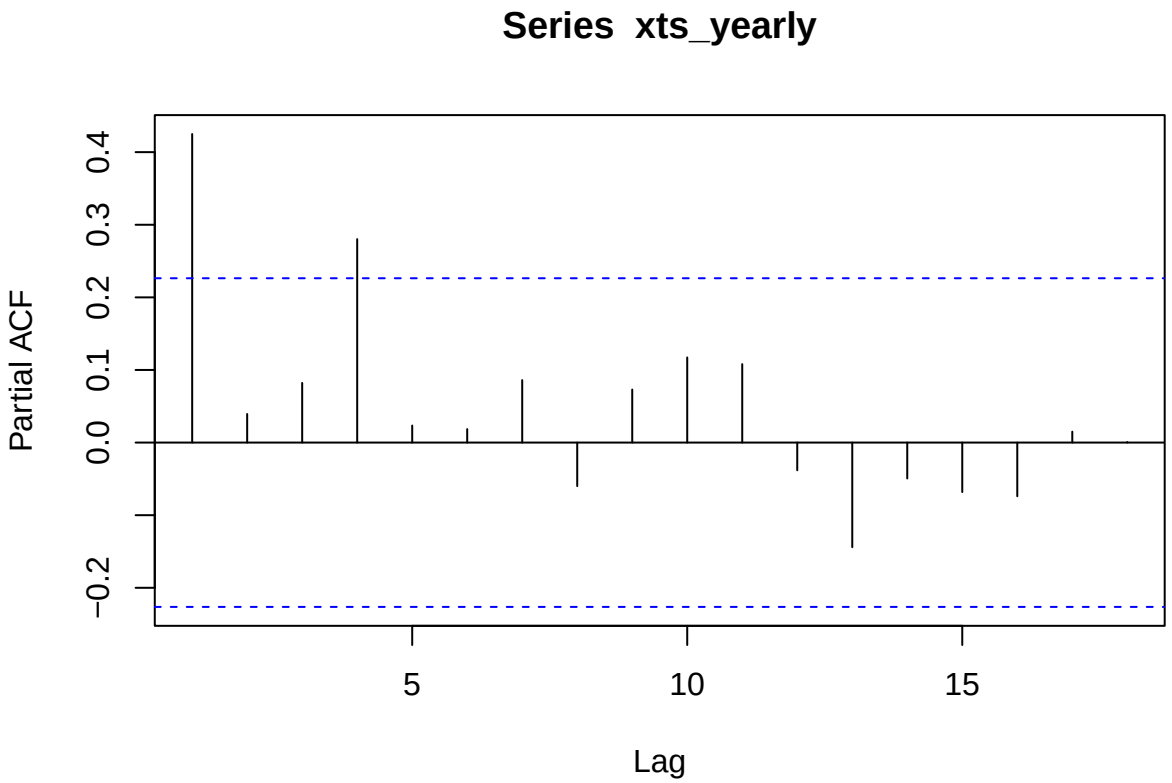
```
results_yearly <-arima(xts_yearly,order=c(0,1,0))
xts_yearly_fitted <- merge(xts_yearly, xts(fitted(results_yearly),index(xts_yearly)))
colnames(xts_yearly_fitted) <- c('Temperature','Fitted')
plot(xts_yearly_fitted,col=c('black','orange'))
```




```
plot(forecast(results_yearly,10), xaxt = 'n', xlab="Year",ylab="Temperature (degrees F)")
axis(1, at = c(2,12,22,32,42,52,62,72), labels=c(1950,1960,1970,1980,1990,2000,2010,2020))
```

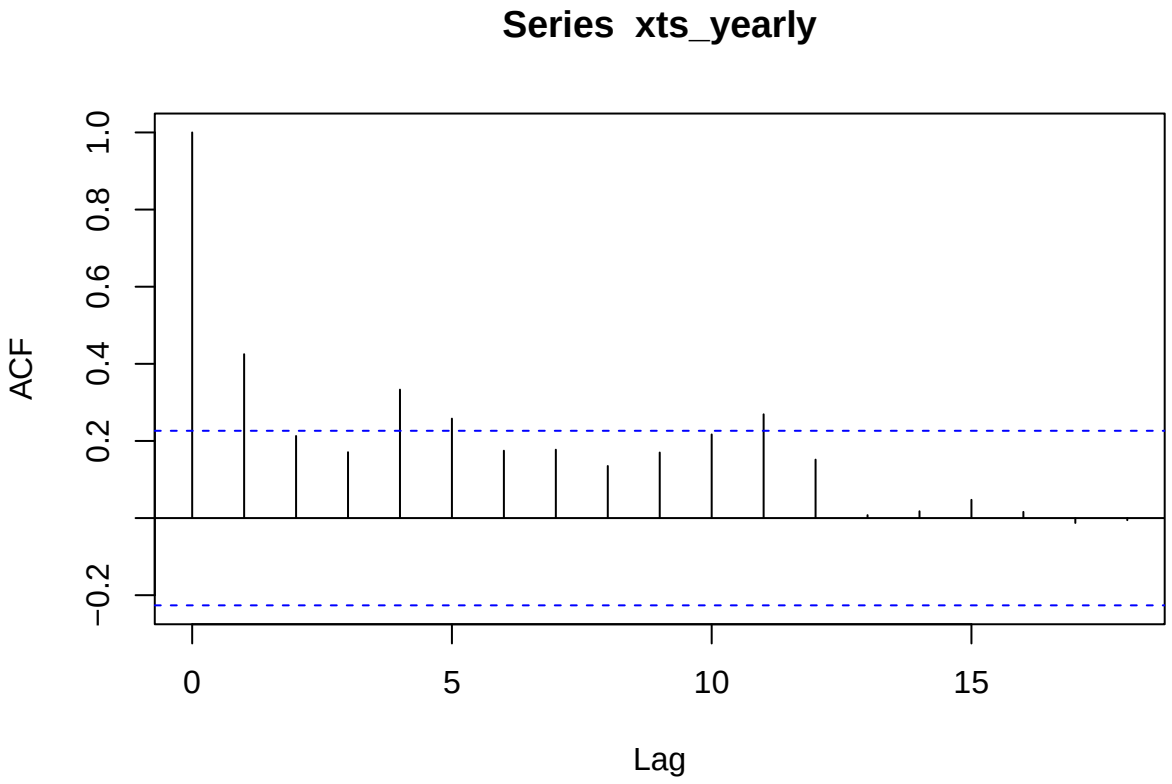



```
pacf(xts_yearly)
```



1st 0.4, 4th 0.3

```
acf(xts_yearly)
```



first 2

pair strong coefficient

```
results_yearly <-auto.arima(xts_yearly)
results_yearly
```

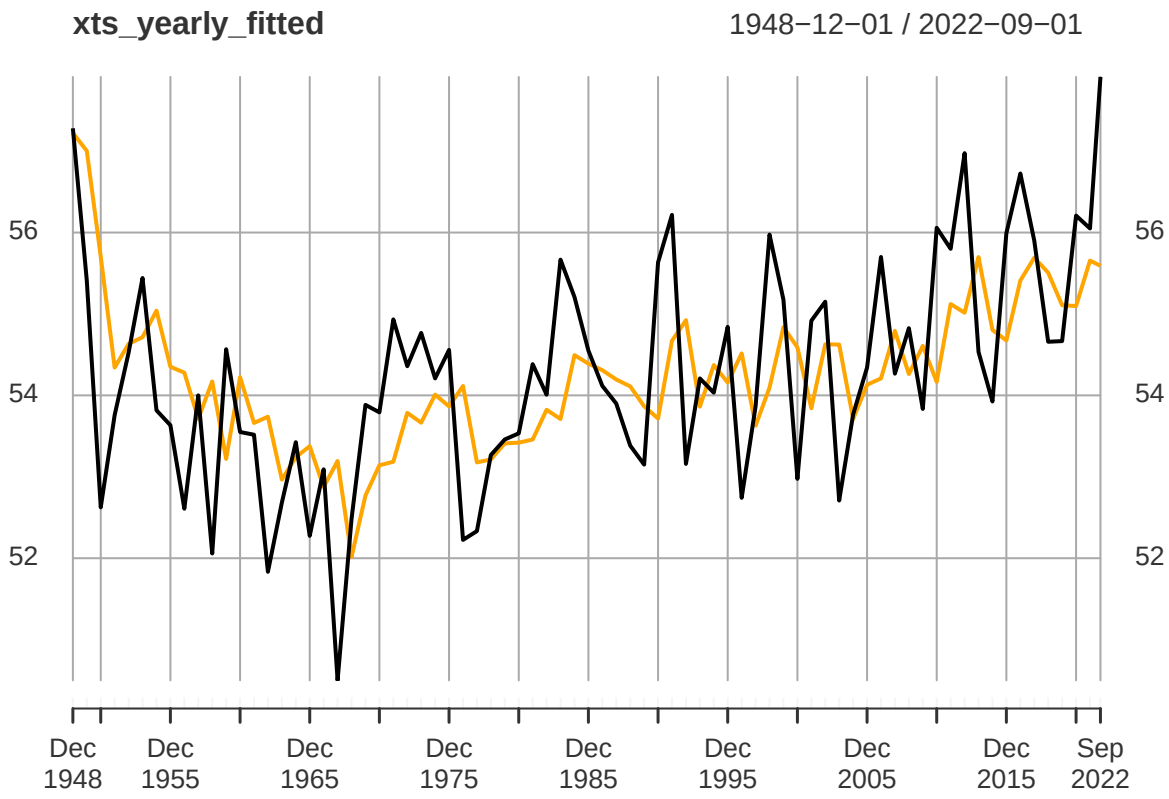
```
## Series: xts_yearly
## ARIMA(0,1,2)
##
## Coefficients:
##           ma1      ma2
##       -0.5795  -0.2070
## s.e.   0.1152   0.1073
##
## sigma^2 = 1.44:  log likelihood = -117.89
## AIC=241.79   AICc=242.13   BIC=248.7
```

p = 0 d = 1 q = 2

```
confint(results_yearly)
```

```
##          2.5 %       97.5 %
## ma1 -0.8052474 -0.353738801
## ma2 -0.4173303  0.003398144
```

```
xts_yearly_fitted <- merge(xts_yearly, xts(fitted(results_yearly),index(xts_yearly)))
colnames(xts_yearly_fitted) <- c('Temperature','Fitted')
plot(xts_yearly_fitted,col=c('black','orange'))
```




```
auto.arima(xts_monthly,D=12, max.p=24, max.q=24)
```

```
## Series: xts_monthly
## ARIMA(4,0,0) with non-zero mean
##
## Coefficients:
##          ar1     ar2      ar3      ar4     mean
##       0.7638  0.1700  -0.1211  -0.4619  54.1948
## s.e.  0.0297  0.0391   0.0391   0.0298   0.1811
##
## sigma^2 = 12.32:  log likelihood = -2380.72
## AIC=4773.44   AICc=4773.54   BIC=4802.19
```