Q1: How to decomposed time series? And using ARIMA for modeling and forecasting.

#Data

data("AirPassengers")

View(AirPassengers)

AirPassengers

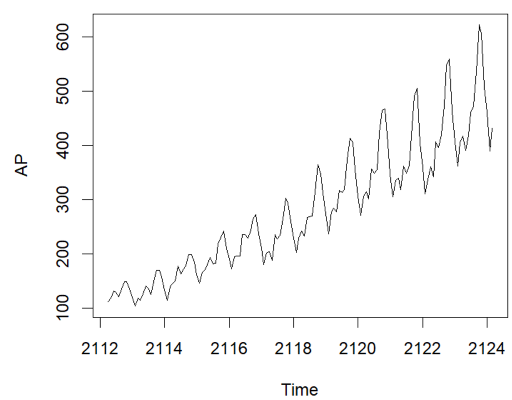
```
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

#preparing data

AP<-ts(AirPassengers, frequency =12, start =c(1949,1960))

 # Understanding data

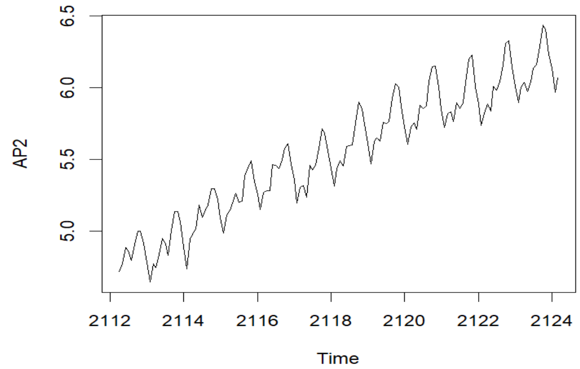
plot(AP)



#log-Transform to fix-up variation

AP2<-log(AP)

plot(AP2)



#Decompose

DAP<- decompose(AP2)

DAP$figure

```
[1] -0.085815019 -0.114412848  0.018113355 -0.013045611 -0.008966106  0.115392997
[7]  0.210816435  0.204512399  0.064836351 -0.075271265 -0.215845612 -0.100315075
```

plot(DAP$figure,

type = 'b',

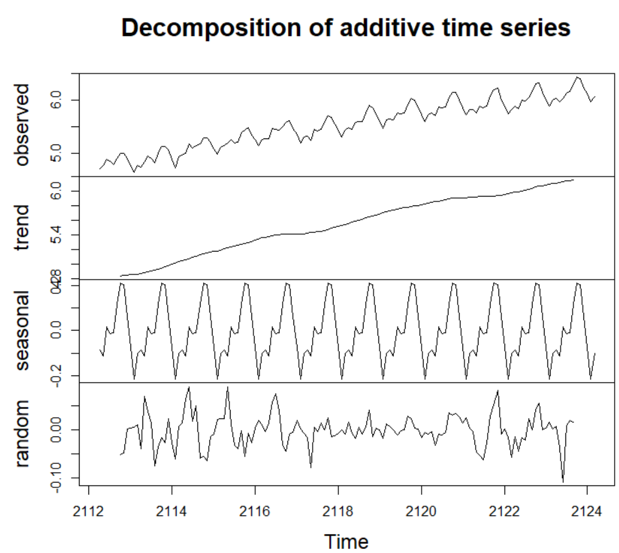
xlab = "month",

ylab = "seasonality Index",

colors("red"),

las= 2)

plot(DAP)



#Auto rigrative moving average model(ARIMA)

#ARIMA(p,d,q) model

library(forecast)

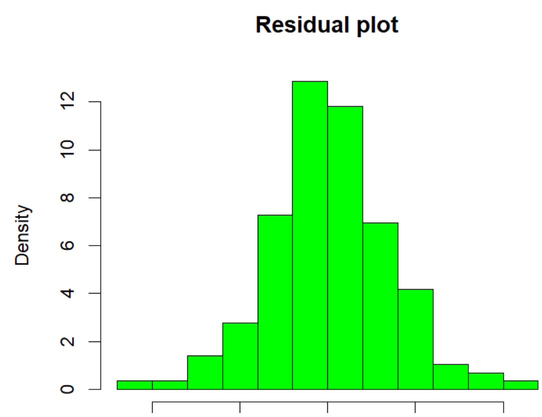
fitmodel<-auto.arima(AP2)

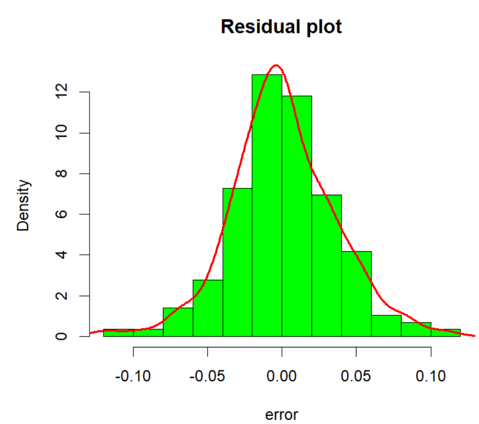
fitmodel

```
Series: AP2
ARIMA(0,1,1)(0,1,1)[12]

Coefficients:
          ma1     sma1
      -0.4018  -0.5569
s.e.   0.0896   0.0731

sigma^2 = 0.001371:  log likelihood = 244.7
AIC=-483.4   AICc=-483.21   BIC=-474.77
```

#check Residual pot

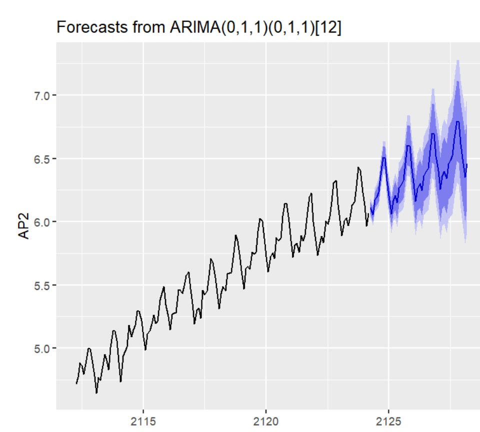
```
hist(fitmodel$residuals,
     main = "Residual plot",
     xlab = "error",
     col = "green",
     freq = F)
lines(density(fitmodel$residuals),
      col="red",
      lw=2)
```





#Forecast for the next 4 years

```
pred<-forecast(fitmodel,4*12)
library(ggplot2)
autoplot(pred)
```



accuracy(pred)

```
                        ME       RMSE         MAE         MPE      MAPE       MAS
E
Training set 0.0005730622 0.03504883 0.02626034 0.01098898 0.4752815 0.216952

2
                    ACF1
Training set 0.01443892
```

Q2: How to select model and using selected model how to forecast by the fitted model?

#Data

data("AirPassengers")

AirPassengers

```
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

#understanding and preparing data

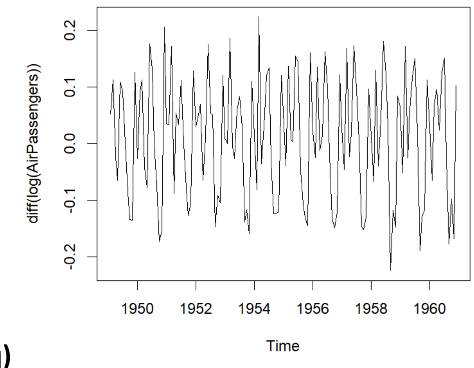
boxplot(AirPassengers-cycle(AirPassengers))

plot(AirPassengers)

abline(lm(AirPassengers-time(AirPassengers)))

#make it stationary

plot(diff(log(AirPassengers)))



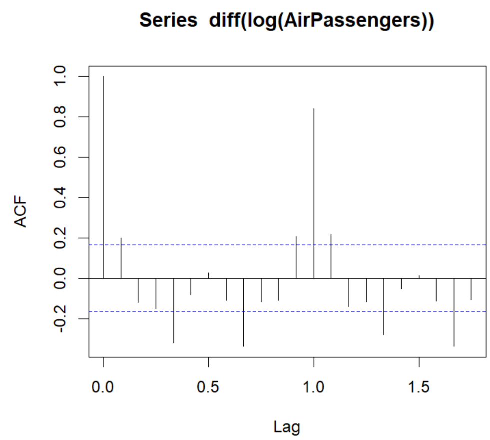
#Modelling: ARIMA(p,d,q)

library(tseries)

selecting the value of q

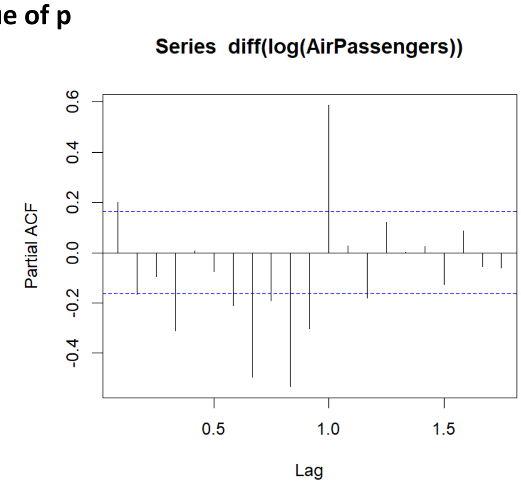
acf(AirPassengers)

acf(diff(log(AirPassengers)))



 # selecting the value of p

pacf(diff(log(AirPassengers)))



fit ARIMA(0,1,1)model

myfit<-arima(log(AirPassengers),

order =c(0,1,1),

seasonal=list(order=c(0,1,1),

period=12))

myfit

```
Call:
arima(x = log(AirPassengers), order = c(0, 1, 1), seasonal = list(order = c(0
,
```

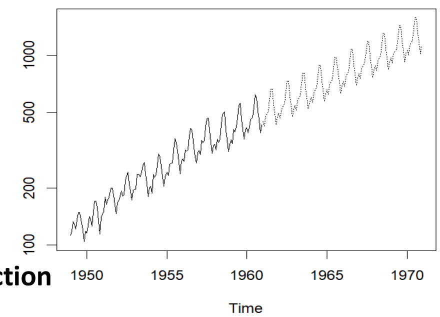
```
    1, 1), period = 12))

Coefficients:
          ma1      sma1
      -0.4018   -0.5569
s.e.   0.0896    0.0731

sigma^2 estimated as 0.001348:  log likelihood = 244.7,  aic = -483.4
```

#forecast for next 10 year

pred<-predict(myfit,n.ahead = 10*12)

finalpred<-exp(pred$pred)

ts.plot(AirPassengers,finalpred, log='y',lty=c(1,3))



#cheaking accuracy of the prediction

datat<-ts(AirPassengers,frequency = 12,start = c(1949,1),end = c(1960,12))

myfit2<-arima(log(datat),

order =c(0,1,1),

seasonal=list(order=c(0,1,1),

period=12))

pred2<-predict(myfit2,n.ahead = 1*12)

finalpred2<-exp(pred2$pred)

finalpred2<-exp(pred2$pred)

pred_1960<-round(finalpred2,0)

true_1960<-tail(AirPassengers,12)

data.frame(pred_1960,true_1960)

	pred_1960	true_1960
1	450	417
2	426	391
3	479	419
4	492	461
5	509	472
6	583	535
7	670	622

```
8          667        606
9          558        508
10         497        461
11         430        390
12         477        432
```

Q3: How to time series analysis nicely and fancy style visualizing by using ggplot?

 # Declaration Library Function

library(forecast)

library(fpp)

library(fpp2)

library(ggplot2)

#time series plot

data(a10)

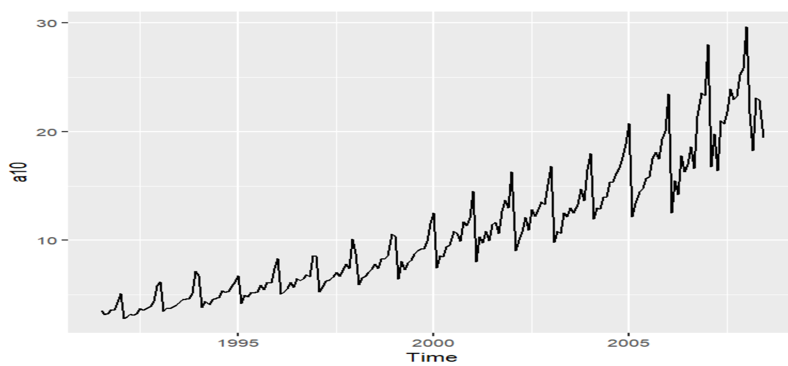
View(a10)

a10

```
   Jan        Feb        Mar        Apr        May        Jun        Jul        Aug
1991                                                                    3.526591  3
                                         .180891
1992  5.088335  2.814520  2.985811  3.204780  3.127578  3.270523  3.737851  3
                                         .558776
1993  6.192068  3.450857  3.772307  3.734303  3.905399  4.049687  4.315566  4
                                         .562185
1994  6.731473  3.841278  4.394076  4.075341  4.540645  4.645615  4.752607  5
                                         .350605
1995  6.749484  4.216067  4.949349  4.823045  5.194754  5.170787  5.256742  5
                                         .855277
1996  8.329452  5.069796  5.262557  5.597126  6.110296  5.689161  6.486849  6
                                         .300569
1997  8.524471  5.277918  5.714303  6.214529  6.411929  6.667716  7.050831  6
                                         .704919
1998  8.798513  5.918261  6.534493  6.675736  7.064201  7.383381  7.813496  7
                                         .431892
1999 10.391416  6.421535  8.062619  7.297739  7.936916  8.165323  8.717420  9
                                         .070964
2000 12.511462  7.457199  8.591191  8.474000  9.386803  9.560399 10.834295 10
                                         .643751
2001 14.497581  8.049275 10.312891  9.753358 10.850382  9.961719 11.443601 11
                                         .659239
2002 16.300269  9.053485 10.002449 10.788750 12.106705 10.954101 12.844566 12
                                         .196500
2003 16.828350  9.800215 10.816994 10.654223 12.512323 12.161210 12.998046 12
                                         .517276
2004 18.003768 11.938030 12.997900 12.882645 13.943447 13.989472 15.339097 15
                                         .370764
2005 20.778723 12.154552 13.402392 14.459239 14.795102 15.705248 15.829550 17
                                         .554701
```

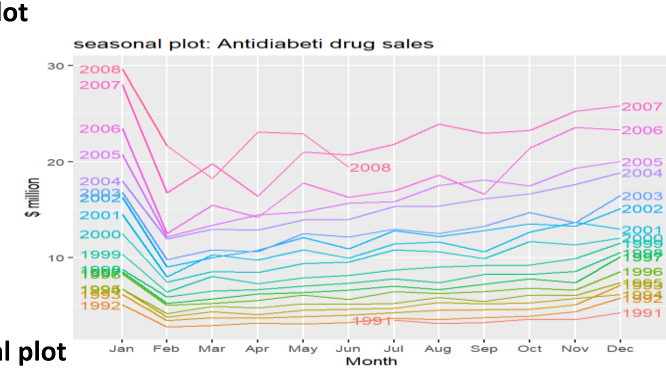
```
2006 23.486694 12.536987 15.467018 14.233539 17.783058 16.291602 16.980282 18
.612189
2007 28.038383 16.763869 19.792754 16.427305 21.000742 20.681002 21.834890 23
.930204
2008 29.665356 21.654285 18.264945 23.107677 22.912510 19.431740


           Sep       Oct       Nov       Dec
1991  3.252221  3.611003  3.565869  4.306371
1992  3.777202  3.924490  4.386531  5.810549
1993  4.608662  4.667851  5.093841  7.179962
1994  5.204455  5.301651  5.773742  6.204593
1995  5.490729  6.115293  6.088473  7.416598
1996  6.467476  6.828629  6.649078  8.606937
1997  7.250988  7.819733  7.398101 10.096233
1998  8.275117  8.260441  8.596156 10.558939
1999  9.177113  9.251887  9.933136 11.532974
2000  9.908162 11.710041 11.340151 12.079132
2001 10.647060 12.652134 13.674466 12.965735
2002 12.854748 13.542004 13.287640 15.134918
2003 13.268658 14.733622 13.669382 16.503966
2004 16.142005 16.685754 17.636728 18.869325
2005 18.100864 17.496668 19.347265 20.031291
2006 16.623343 21.430241 23.575517 23.334206
2007 22.930357 23.263340 25.250030 25.806090
2008
```

```
autoplot(a10)+
ggtitle("Antidiabeteic drug sales")+
ylab("$ million")+
xlab("Year")
```

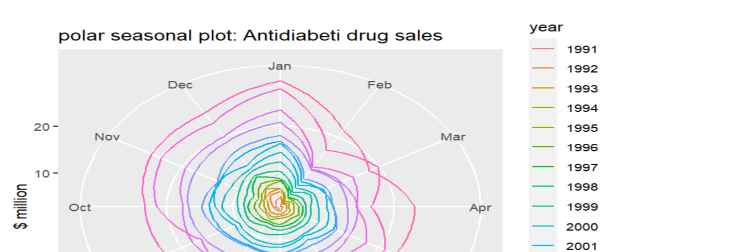


#seasonal plot

```
ggseasonplot(a10,year.labels = T,year.labels.left = T)+
  ggtitle("seasonal plot: Antidiabeti drug sales")+
  ylab("$ million")
```



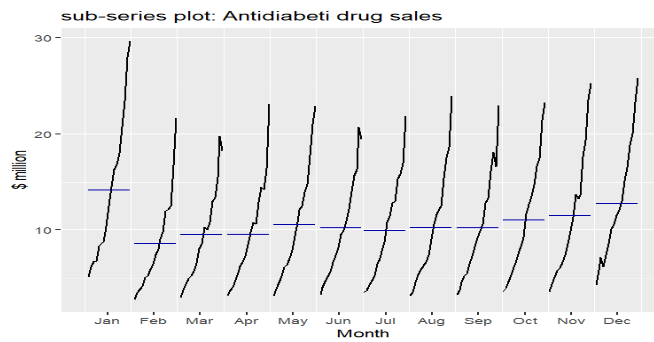
 # polar seasonal plot



```
ggseasonplot(a10,polar = T)+
  ggtitle("polar seasonal plot: Antidiabeti drug sales")+
  ylab("$ million")
```

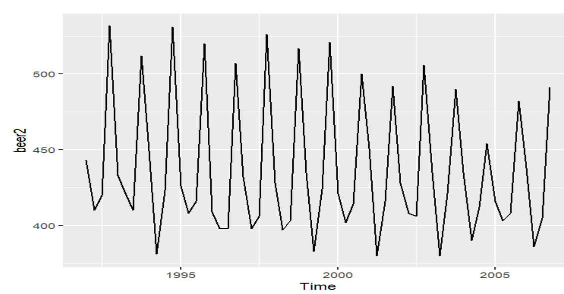
#seasonal sub-series plot

```
ggsubseriesplot(a10)+
  ggtitle("sub-series plot: Antidiabeti drug sales")+
  ylab("$ million")
```

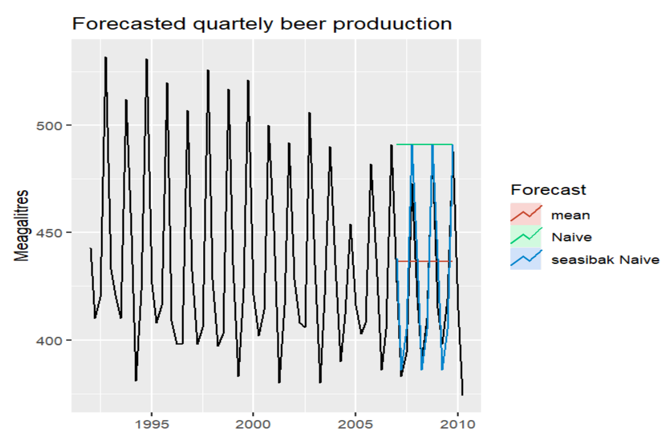


 # visualizing ausbeer data

```
data(ausbeer)
beer2<-window(ausbeer,start=1992, end=c(2006,4))
autoplot(beer2)
beerfit1<-meanf(beer2,h=12)
beerfit2<-rwf(beer2,h=12)
beerfit3<-snaive(beer2,h=12)
```




```
autoplot(window(ausbeer,start=1992))+
  autolayer(beerfit1,series="mean", PI=F)+
  autolayer(beerfit2,series="Naive", PI=F)+
  autolayer(beerfit3,series="seasibak Naive", PI=F)+
  xlab("year")+ ylab("Meagalitres")+
  ggtitle("Forecasted quartely beer produuction")+
```



guides(colour=guide_legend(title = "Forecast"))

 # Accuracy

beer3<-window(ausbeer,start=2008)

accuracy(beerfit1,beer3)

```
         ME      RMSE       MAE          MPE      MAPE      MASE          ACF1
Training set  1.137019e-14 43.93382 35.64833 -0.9445257 7.954263 2.449456 -0.
                                  11987178
Test set     -7.950000e+00 36.54726 33.72500 -2.5181562 7.799919 2.317301 -0.
                                  07153733
                                 Theil's U
                         Training set        NA
                         Test set     0.7843278
```

accuracy(beerfit2,beer3)

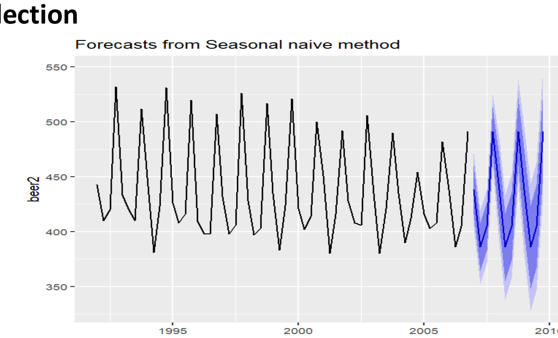
```
        ME      RMSE       MAE           MPE      MAPE      MASE          ACF1
Training set   0.8135593 65.91829 55.08475  -0.8600163 12.20968 3.784964 -0.2
                                  4368919
Test set     -62.5000000 71.96353 62.50000 -15.3314577 15.33146 4.294479 -0.0
                                  7153733
                                 Theil's U
                         Training set        NA
                         Test set       1.45868
```

accuracy(beerfit3,beer3)

```
                    ME      RMSE       MAE          MPE       MAPE       MASE
ACF1
Training set -1.517857 17.07285 14.55357 -0.4155268 3.372578 1.0000000 -0.300
0376
Test set     -1.500000 12.51000 10.25000 -0.3069842 2.457364 0.7042945 -0.110
8185
             Theil's U
Training set        NA
Test set     0.2180679
```

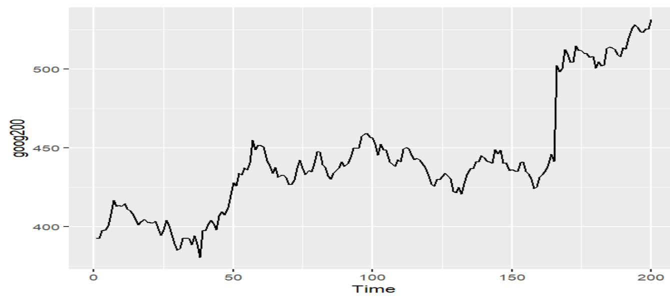
 # final selection

autoplot(beerfit3)

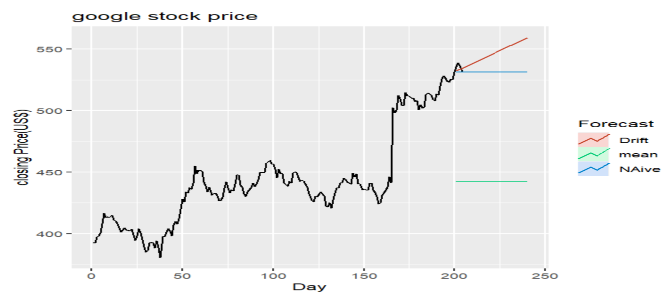


#another data

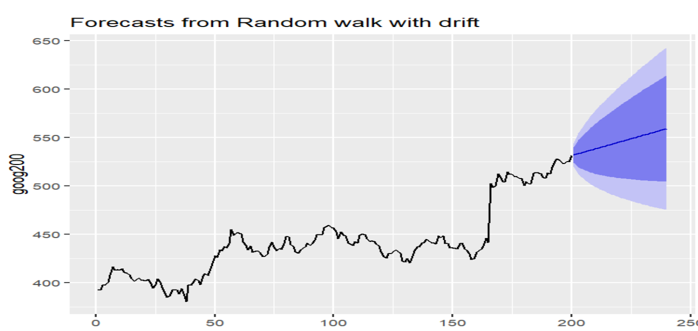
```
goog200

autoplot(goog200)

googf1<-meanf(goog200, h=40)

googf2<-rwf(goog200, h=40)

googf3<-rwf(goog200,drift=T, h=40)
```




```
autoplot(subset(goog,end=204))+
  autolayer(googf1,series="mean",PI=F)+
  autolayer(googf2,series="NAive",PI=F)+
  autolayer(googf3,series="Drift",PI=F)+
  xlab("Day")+ ylab("closing Price(US$)")+
  ggtitle("google stock price")+
  guides(colour=guide_legend(title = "Forecast"))
```




```
autoplot(googf3)
```

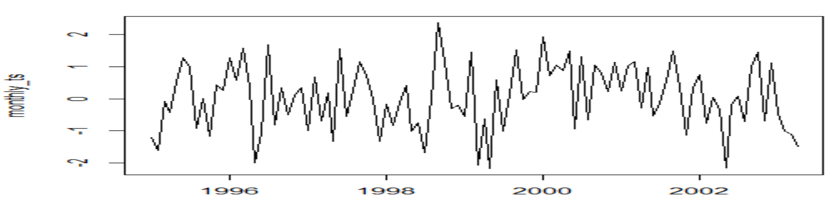


Q4: How to generating Data and detection of trend and Get a stationary Time Series?

#Time series analysis

#Generating data

```
x<-rnorm(100)

view(x)

x

[1] -0.747902421 -0.473104847 -1.043941493  0.301811740 -0.768790292 -1.720081168  0.941182594

  [8]  0.098858238 -0.172633287  0.005243204 -2.274052694 -0.241189658 -0.963456340 -1.269223866

 [15] -0.204649214 -0.578163420  0.792070404  0.994943032  0.570492941  1.985719223 -0.401389408

 [22] -1.047643412  1.222313278 -0.262379784 -0.100142898  0.206916475 -0.768081751  1.501641738

 [29]  0.376477770 -1.342183435  0.180629301  1.402199601  0.137801130 -0.389898041  0.557685488

 [36] -0.178110744  0.615545590 -0.787168830  2.713623354 -0.836511777  0.970026107  0.624321476

 [43] -1.470236404 -0.225251506  0.470230281  0.252471030 -0.157419568 -0.292884635 -
0.922608559

 [50]  0.182791739  0.664990652  0.036177997  0.708713968  0.567625313  0.160513568 -0.253858410

 [57]  0.767612825  0.122492916  1.210583084  0.297926273  0.610016130 -0.039471050 -0.227182402

 [64]  1.363345872 -1.310373224 -1.098287587  1.546954349 -1.277452050 -1.173382852 -
0.424480775

 [71] -0.832962303  0.765737671 -1.045133807  0.915634701  1.138473152 -0.777817327  0.732703687

 [78] -0.908835893  0.949568183 -0.476987385 -0.562065630 -0.338426928  0.886970819  0.259039398

 [85]  1.431263319 -0.883622185  1.422749875  0.779002661 -0.013507713 -0.036438356  0.239295459

monthly_ts<-ts(data = x,start = 1995,frequency = 12)

plot(monthly_ts)
```



Here we see that the time series data is stationary but real data are not stationary all time. In practically time series may often have trends. Trends alter mean and variance. If we predict a model where trends associated then we get result which will be misleading forecasting. So now we will see detection of trends and get a stationary time series.

#Detection of Trend and Get A Stationary Time Series

install.packages("fpp")

library(fpp)

library(forecast)

data("ausbeer")

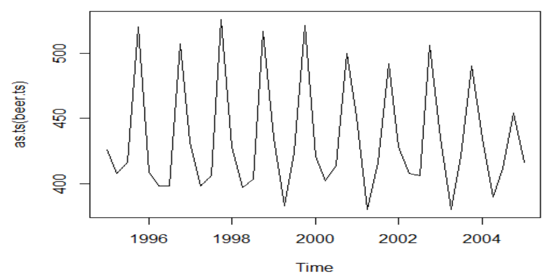
View(ausbeer)

Ausbeer

284 213 227 308 262 228 ………..

beer.ts<-window(ausbeer,start=1995,end=2005)

plot(as.ts(beer.ts))



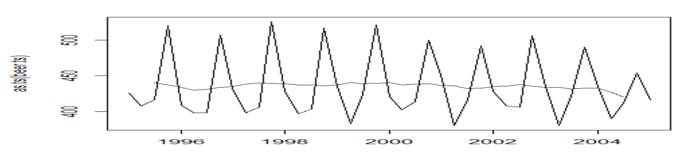
#Creating Moving Average That Will Be Close To Trend

beer.trend <- ma(beer.ts,order=4,centre = T)

#Plot Trend and MA Together

plot(as.ts(beer.ts))

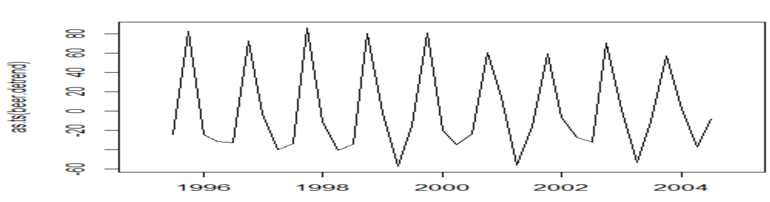
lines(beer.trend)



#Remove the trend from the time series

beer.detrend=beer.ts-beer.trend

plot(as.ts(beer.detrend))



How to analysis time series using prophet package???

Prophet is a procedure for the forecasting time series data based on an additive model where non –linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effect.

Prophet is used in many application across facebook for producing reliable forecast for planning and goal setting.

This model includes trends, seasonality and holiday effect. Mathematical form is given by,

$$Y(t)=g(t)+s(t)+h(t)+€(t)$$

Where

g(t) refers to trends

s(t) refers to seasonality,

h(t) refers to effect of holidays to the forecast

e(t) refers to error term

Q5: How to Analyzing and forecasting from Wikipedia data?

#Time series analysis

#Getting wikipedia trend data

```
install.packages("wikipediatrend")

library(wikipediatrend)

data<-wp_trend(page = "Sakib_al_hasan",

        from = "2015-01-01",

        to="2021-08-20")

View(data)
```

	Language	article	date	views
1	en	sakib_al_hasan	2015-01-01	4
2	en	sakib_al_hasan	2015-01-02	0
3	en	sakib_al_hasan	2015-01-03	5
4	en	sakib_al_hasan	2015-01-04	8
5	en	sakib_al_hasan	2015-01-05	9

361 en   sakib_al_hasan 2015-12-27   0

362 en   sakib_al_hasan 2015-12-28   0

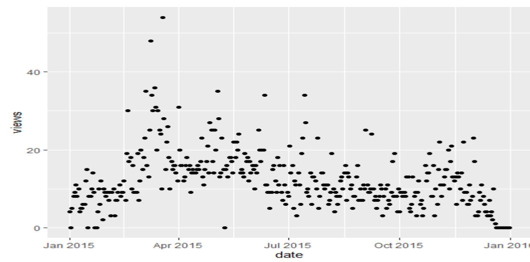
363 en   sakib_al_hasan 2015-12-29   0

364 en   sakib_al_hasan 2015-12-30   0   ……………

#plot

library(ggplot2)

qplot(date,views,data = data)



Here we see from the graph or plot some year visitor number is high cause of his performance and attitude.

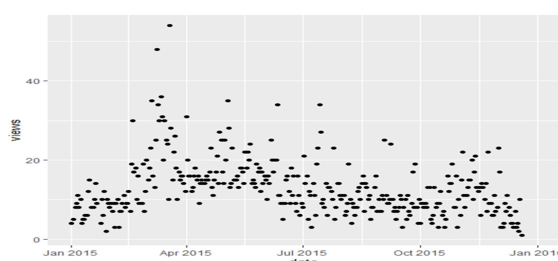
summary(data)

```
  language          article                date                              views
 Length:365         Length:365           Min.   :2015-01-01       Min.   : 0.00
 Class :character   Class :character    1st Qu.:2015-04-02      1st Qu.: 8.00
 Mode  :character  Mode  :character    Median :2015-07-02     Median :11.00
                                          Mean   :2015-07-02     Mean   :12.29
                                         3rd Qu.:2015-10-01    3rd Qu.:16.00
                                         Max.   :2015-12-31     Max.   :54.00
```

#manupulation data

data$views[data$views==0]<-NA

qplot(date,views,data = data



After manipulate data all off zero value used as a missing value with NA in data table and from the plot we seeing that here has no zero value for any single day.

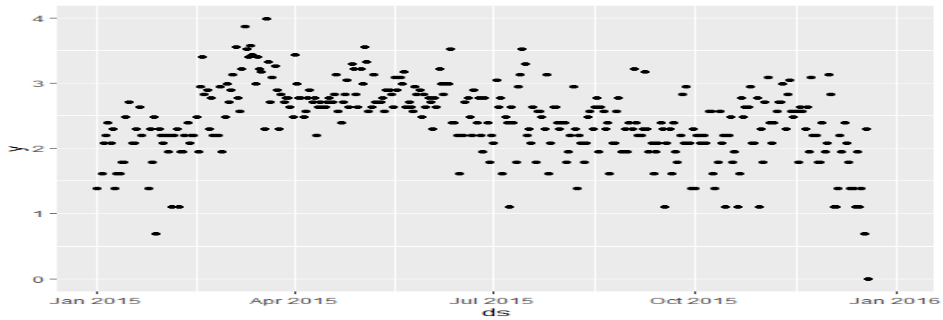
ds<-data$date

y<-log(data$views)

df<-data.frame(ds,y)

View(df)

qplot(ds,y,data = df)



ds y

1 2015-01-01 5.476464

2 2015-01-02 NA

3 2015-01-03 4.976734

4 2015-01-04 5.247024

5 2015-01-05 5.451038

6 2015-01-06 6.538140

#forcusting with facebook prophet

install.packages("prophet")

library(prophet)

mf<-prophet(df)

Disabling yearly seasonality. Run prophet with yearly.seasonality=TRUE to override this.

Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.

#prediction

prdict<-make_future_dataframe(mf,365)

tail(predict)

y ds

725 2016-12-25

726 2016-12-26

727	2016-12-27
728	2016-12-28
729	2016-12-29
730	2016-12-30

forecast<-predict(mf,predict)

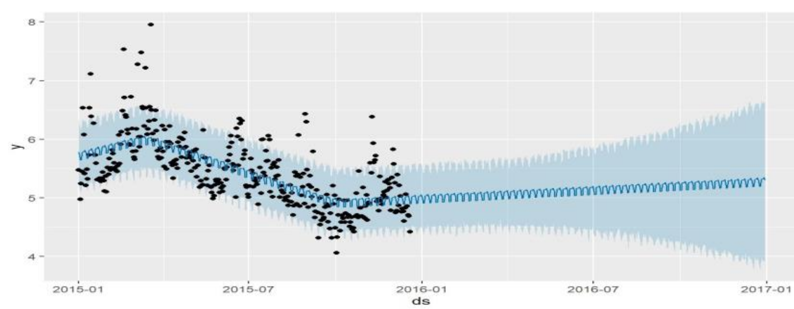
tail(forecast(c['ds','yhat','yhat_lower','yhat_upper')

	Ds	Yhat	yhat_lower	yhat_upper
725	2016-12-25	5.196761	3.872847	6.535854
726	2016-12-26	5.318334	4.010592	6.626629
727	2016-12-27	5.329512	3.951958	6.645157
728	2016-12-28	5.323802	3.996477	6.627882
729	2016-12-29	5.346154	4.030595	6.651266
730	2016-12-30	5.304304	3.959696	6.630339

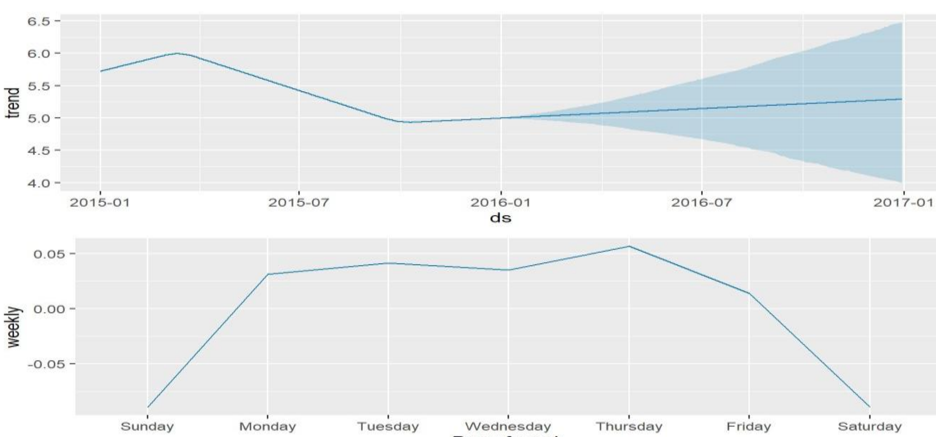
exp(5.304)

[1] 201.1398

plot(mf,forecast)



plot(mf,forecast)



Q6: How to analyzing and forecasting COVID-19 data of Bangladesh ?

#Time series analysis

#Getting lattest data of Bangladesh

data<-read.csv(choose.files(),header=T,sep=",")

View(data)

data

```
iso_code continent    location       date total_cases new_cases new_cases_smo
othed
1       AFG      Asia Afghanistan 2020-01-03          NA         0
NA
2       AFG      Asia Afghanistan 2020-01-04          NA         0
NA
3       AFG      Asia Afghanistan 2020-01-05          NA         0
NA
4       AFG      Asia Afghanistan 2020-01-06          NA         0
NA
5       AFG      Asia Afghanistan 2020-01-07          NA         0
NA
6       AFG      Asia Afghanistan 2020-01-08          NA         0
0
7       AFG      Asia Afghanistan 2020-01-09          NA         0
0
8       AFG      Asia Afghanistan 2020-01-10          NA         0
0
9       AFG      Asia Afghanistan 2020-01-11          NA         0
0
10      AFG      Asia Afghanistan 2020-01-12          NA         0
```

………..

#packages

install.packages("ggplot2")

install.packages("dplyr")

library(ggplot2)

library(dplyr)

#screaning Data for analysis

data1<-filter(data,location--"Bangladesh")

View(data1)

```
data1<-filter(data1,date>="2020-03-15")

data2<-select(data1,date,new_cases)

View(data2)

str(data2)

data2$date<-as.date(data2$date)
```

#plot

```
qplot(date,new_cases,data = data2,
   main="covid-19 new cases in Bangladesh")

df<-data2$date

y<-data.frame(ds,y)

View(df)
```

#Forecasting

```
install.packages("prophet")

library(prophet)

mcc<-prophet(df)


#prediction

predict<-make_future_dataframe(mcc,periods = 130)

forecast<-predict(mcc,predict)

plot(mcc,forecast,xlab="data",ylab="newcases")
```

Q7: **How to analyzing and forecasting latest COVID-19 data of Bangladesh?**

#Time series analysis

#Getting lattest data of Bangladesh

```
data<-read.csv(choose.files(),header=T,sep=",")

View(data)
```

#packages

```
install.packages("ggplot2")

install.packages("dplyr")

library(ggplot2)

library(dplyr)
```

#screaning Data for analysis

```
data1<-filter(data,location--"Bangladesh")

View(data1)

data1<-filter(data1,date>="2020-03-15")

cc<-select(data1,date,total_cases)

View(cc)

str(cc)

cc$total_cases<-as.numeric(cc$total_cases)

cc$date<-as.date(cc$data)
```

#plot

```
qplot(date,total_cases,data =cc,)
```

```
    main="covid-19 confirmed cases in Banglade"

df<-cc $date
y<-data.frame(ds,y)
View(df)
```

#Forecasting

```
install.packages("prophet")
library(prophet)
mcc<-prophet(df)
```

#prediction

```
predict<-make_future_dataframe(mcc,periods = 180)
tail(predict)
forecast<-predict(mcc,predict)
tail(forecast[c('ds','yhat','yhat_lower','yhat_upper')])
plot(mcc,forecast)
dyplot.prophet(mcc,forecast)
prophet_plot_components(mcc,forecast)
```

#model 1 performance

```
pred<-forecast$yhat[1:332]
actual<-mcc$history$y
plot(actul,pred)
abline(lm(pred~actual),col="red" )
```

Index