

**Practical Notebook**

**On**

**Statistical Data Analysis XIII (Lab XIII)**

Course Code: STAT 4109

Course: Time Series Analysis I

|  |  |
| --- | --- |
| **Submitted To:** | **Submitted By:** |
| Dr. Md. Shohel Rana | Md. Lotifur Rahman |
| Associate Professor | Roll: 1822017 |
| Department of Statistics | Session: 2018-2019 |
| Islamic University, Kushtia,  Bangladesh | Islamic University, Kushtia,  Bangladesh |

**Department of Statistics,**

**Islamic University, Kushtia**

**Submission Date: 03/02/2024**

Table of Contents

[Problem: -1 4](#_Toc157609310)

[Random Number Generating. 4](#_Toc157609311)

[Fig-01: Quarterly Plot Random Number 5](#_Toc157609312)

[Fig-02: Monthly Time Series 6](#_Toc157609313)

[Australia beer production Time series analysis. 6](#_Toc157609314)

[Fig-03: Plotting AusBeer Data 8](#_Toc157609315)

[Fig-04: Beer Sells with Moving Average 9](#_Toc157609316)

[Fig-05: Australia Beer Production After Moving Average 9](#_Toc157609317)

[Problem: -02 11](#_Toc157609318)

[Air Passengers data analysis from 1949-160. 11](#_Toc157609319)

[Fig-06: Boxplot of AirPassengers 12](#_Toc157609320)

[Fig-07: Plotting Airpassengrs with Abline. 13](#_Toc157609321)

[Fig-08: After differention and log transformation of Airpassengers. 14](#_Toc157609322)

[Fig-09: Auto Covariance Function (ACF) 15](#_Toc157609323)

[Fig-10: After Transformation of AirPassengers Data. 15](#_Toc157609324)

[Fig-11: Partial Auto Covariance Function. 16](#_Toc157609325)

[Fig-12: AirPassengers Prediction. 18](#_Toc157609326)

[Problem: -3 20](#_Toc157609327)

[Decomposition and modeling AirPassengers data. 20](#_Toc157609328)

[Fig-13: Plotting Airpassengers Data. 21](#_Toc157609329)

[Fig-14: Log Transformation of Air Passengers. 22](#_Toc157609330)

[Fig-15: Seasonality Plot 22](#_Toc157609331)

[Fig-16: Decomposition of Airpassengers. 23](#_Toc157609332)

[Fig-17: Residual Plot. 24](#_Toc157609333)

[Fig-18: Prediction of Next four year. 25](#_Toc157609334)

[Problem: -04 25](#_Toc157609335)

[Visualization of data by ggplot2. 25](#_Toc157609336)

[Anti diabetic drug subsidy (1991-2009) 26](#_Toc157609337)

[Fig-19: Antidiabetic Drug sales plot 27](#_Toc157609338)

[Fig-20: Antidiabetic Drug sales Seasonal Plot. 28](#_Toc157609339)

[Fig-21: Polar Plot of Antidiabetic Drug sales. 29](#_Toc157609340)

[Fig-22: Sub-plot Of Antidiabetic drug sales. 29](#_Toc157609341)

[*beer production in Australia (in megalitres) from 1956:Q1 to 2010:Q2.* data("ausbeer") ausbeer 30](#_Toc157609342)

[Fig-23: Potting Australia Beer Production. 31](#_Toc157609343)

[Fig-24: Forecasted Quarterly Beer Production. 32](#_Toc157609344)

[Google Stock (2013-2017) 33](#_Toc157609345)

[Fig-25: Auto Plot Google data. 35](#_Toc157609346)

[Fig-26: Forecast Google Stock Price. 36](#_Toc157609347)

[Fig-27: Forecast From Random Walk. 36](#_Toc157609348)

[Problem: -05 37](#_Toc157609349)

[Analyzing and forecasting monthly rainfall of Bangladesh. 37](#_Toc157609350)

[Fig-28: Plotting Rainfall Time Series Data. 41](#_Toc157609351)

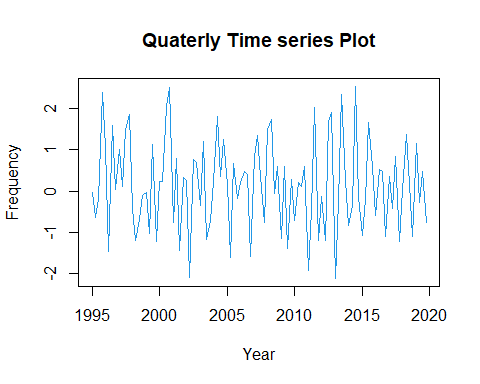
[Fig-29: Decomposition of Rainfall Data. 42](#_Toc157609352)

[Fig-30: Forecast Next Four Year. 43](#_Toc157609353)

# Problem: -1

## Random Number Generating.

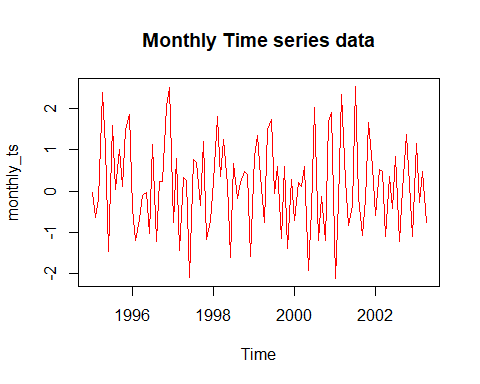
#-Generating random number from normal dist.  
x=rnorm(100,0,1)  
  
#help(ts)  
  
#Creating a time series   
quater\_ts=ts(data = x,start = 1995,frequency = 4)  
  
#---Creating time series plot for quaterly data-  
plot(quater\_ts,col=4,ylab="Frequency",  
 xlab="Year",main="Quaterly Time series Plot")



### Fig-01: Quarterly Plot Random Number

**Comment:** Those the quarterly and monthly data are the stationary   
cause data form normal distribution, Have equal mean and variance.

#Creating time series plot for monthly data  
monthly\_ts=ts(data = x,start = 1995,frequency = 12)  
  
#--------------Plot--------------------  
plot(monthly\_ts,col="red",main="Monthly Time series data")  
  
#-Detection of tend and get a stationary time series   
# install.packages("fpp")  
library(fpp)



### Fig-02: Monthly Time Series

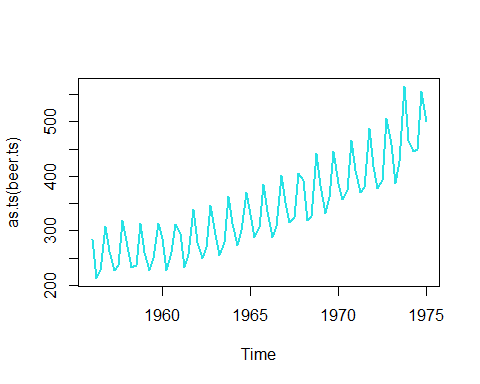
**Comment:** The plot show data are stationary because sample draw from normal distribution.

## Australia beer production Time series analysis.

library(forecast)  
data(ausbeer)  
View(ausbeer)  
ausbeer

## Qtr1 Qtr2 Qtr3 Qtr4  
## 1956 284 213 227 308  
## 1957 262 228 236 320  
## 1958 272 233 237 313  
## 1959 261 227 250 314  
## 1960 286 227 260 311  
## 1961 295 233 257 339  
## 1962 279 250 270 346  
## 1963 294 255 278 363  
## 1964 313 273 300 370  
## 1965 331 288 306 386  
## 1966 335 288 308 402  
## 1967 353 316 325 405  
## 1968 393 319 327 442  
## 1969 383 332 361 446  
## 1970 387 357 374 466  
## 1971 410 370 379 487  
## 1972 419 378 393 506  
## 1973 458 387 427 565  
## 1974 465 445 450 556  
## 1975 500 452 435 554  
## 1976 510 433 453 548  
## 1977 486 453 457 566  
## 1978 515 464 431 588  
## 1979 503 443 448 555  
## 1980 513 427 473 526  
## 1981 548 440 469 575  
## 1982 493 433 480 576  
## 1983 475 405 435 535  
## 1984 453 430 417 552  
## 1985 464 417 423 554  
## 1986 459 428 429 534  
## 1987 481 416 440 538  
## 1988 474 440 447 598  
## 1989 467 439 446 567  
## 1990 485 441 429 599  
## 1991 464 424 436 574  
## 1992 443 410 420 532  
## 1993 433 421 410 512  
## 1994 449 381 423 531  
## 1995 426 408 416 520  
## 1996 409 398 398 507  
## 1997 432 398 406 526  
## 1998 428 397 403 517  
## 1999 435 383 424 521  
## 2000 421 402 414 500  
## 2001 451 380 416 492  
## 2002 428 408 406 506  
## 2003 435 380 421 490  
## 2004 435 390 412 454  
## 2005 416 403 408 482  
## 2006 438 386 405 491  
## 2007 427 383 394 473  
## 2008 420 390 410

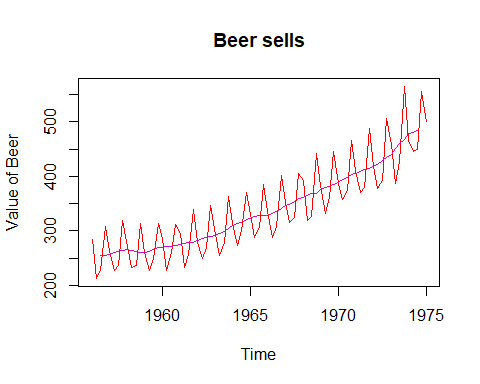
beer.ts=window(ausbeer,start=1956,end=1975)  
plot(as.ts(beer.ts),col=5,lwd=2)



### Fig-03: Plotting AusBeer Data

**Comment:** we can see that clearly an upward trend is presence in the data. So, this is a non-stationary data.

# Upper tend data  
#Creating a moving average that will be close to trend  
  
beer.trend=ma(beer.ts,order = 4,centre = T)  
  
#Plot trend a MA together  
plot(as.ts(beer.ts),col="red",ylab="Value of Beer",main="Beer sells")  
lines(beer.trend,col=6,lwd=1.5)



### Fig-04: Beer Sells with Moving Average

**Comment:** A moving average line is also plotted along with the trend which is a proper representative of the trend. And, due to 4th order moving average we will not get first 2 time points and last 2 time points and last 2 time points will not get moving average.

#Removing the trend from the time series  
beer.detrend=beer.ts-beer.trend  
  
#Plot the detrend data  
  
plot(as.ts(beer.detrend),col=7,  
 main="Australia Beer Production",ylab="Values",lwd=2)

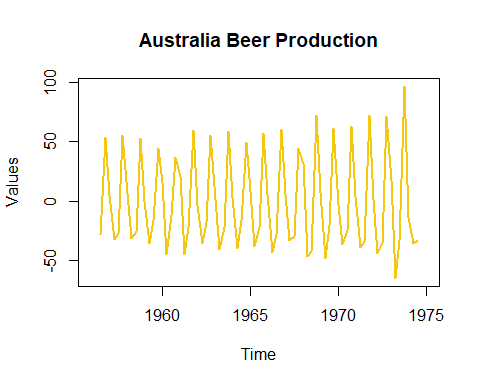


Fig-05: Australia Beer Production After Moving Average

**Comment:** Now our plot is stationary plot because here over the time mean is same and variance is also same.

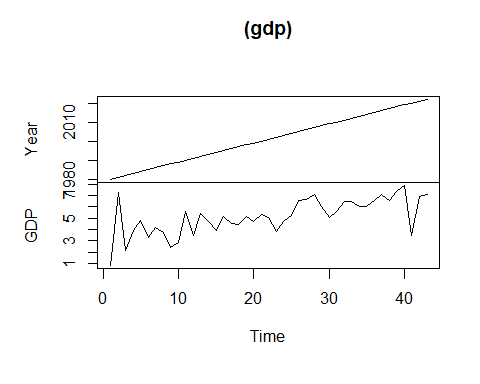
#But in real life problem all data sets are not stationary  
#In this situation we have to convert data into a stationary  
#data  
  
  
#-Detection of tend and get a stationary time series   
# install.packages("fpp")  
library(fpp)  
data(ausbeer)  
View(ausbeer)  
ausbeer

## Qtr1 Qtr2 Qtr3 Qtr4  
## 1956 284 213 227 308  
## 1957 262 228 236 320  
## 1958 272 233 237 313  
## 1959 261 227 250 314  
## 1960 286 227 260 311  
## 1961 295 233 257 339  
## 1962 279 250 270 346  
## 1963 294 255 278 363  
## 1964 313 273 300 370  
## 1965 331 288 306 386  
## 1966 335 288 308 402  
## 1967 353 316 325 405  
## 1968 393 319 327 442  
## 1969 383 332 361 446  
## 1970 387 357 374 466  
## 1971 410 370 379 487  
## 1972 419 378 393 506  
## 1973 458 387 427 565  
## 1974 465 445 450 556  
## 1975 500 452 435 554  
## 1976 510 433 453 548  
## 1977 486 453 457 566  
## 1978 515 464 431 588  
## 1979 503 443 448 555  
## 1980 513 427 473 526  
## 1981 548 440 469 575  
## 1982 493 433 480 576  
## 1983 475 405 435 535  
## 1984 453 430 417 552  
## 1985 464 417 423 554  
## 1986 459 428 429 534  
## 1987 481 416 440 538  
## 1988 474 440 447 598  
## 1989 467 439 446 567  
## 1990 485 441 429 599  
## 1991 464 424 436 574  
## 1992 443 410 420 532  
## 1993 433 421 410 512  
## 1994 449 381 423 531  
## 1995 426 408 416 520  
## 1996 409 398 398 507  
## 1997 432 398 406 526  
## 1998 428 397 403 517  
## 1999 435 383 424 521  
## 2000 421 402 414 500  
## 2001 451 380 416 492  
## 2002 428 408 406 506  
## 2003 435 380 421 490  
## 2004 435 390 412 454  
## 2005 416 403 408 482  
## 2006 438 386 405 491  
## 2007 427 383 394 473  
## 2008 420 390 410

library(readxl)

## Warning: package 'readxl' was built under R version 4.3.2

df<- read\_excel("GDP of bangladesh.xlsx")  
View(df)  
# gdp.ts=window(df,start = 1980,end = 2020)  
gdp=ts(df)  
plot((gdp))



# Problem: -02

## Air Passengers data analysis from 1949-160.

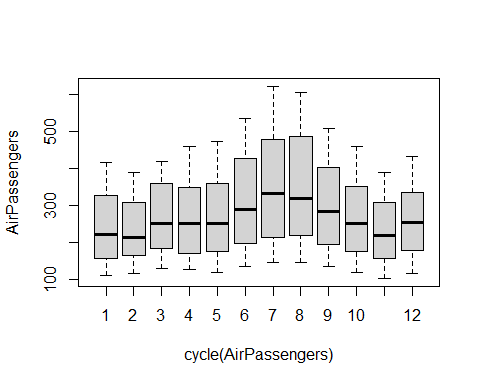
data("Airpassenger")

## Warning in data("Airpassenger"): data set 'Airpassenger' not found

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

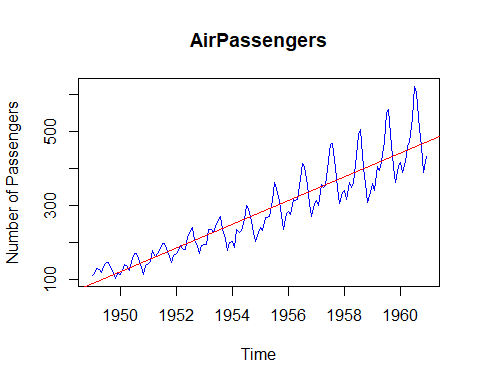
#Understanding and Preparing data  
boxplot(AirPassengers~cycle(AirPassengers))



### Fig-06: Boxplot of AirPassengers

**Comment:** Fig-06 shows us how many passengers travels by air in several months. Highest number of passengers travel in August and September and the lowest numbers of passengers travel in November.

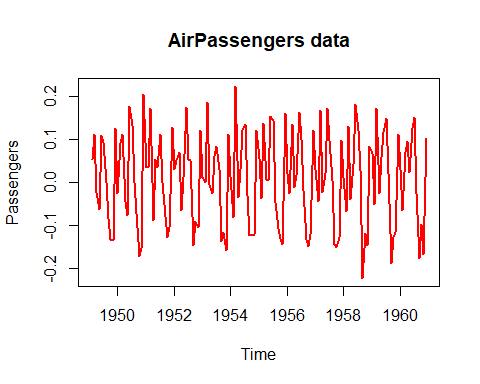
plot(AirPassengers,main="AirPassengers",ylab="Number of Passengers",  
 col="Blue",lwd=1.5)  
  
# upward and seasonality Trend data  
abline(lm(AirPassengers~time(AirPassengers)),lwd=1.3,col="red")



### Fig-07: Plotting Airpassengrs with Abline.

**Comment:** Show the trend of fig-07 using average method. Here, mean and variance change over time which implies non-stationary.

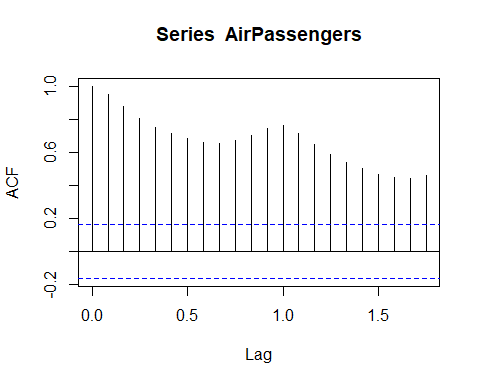
#Make it stationary  
# do variance constant   
plot(diff(log(AirPassengers)),col="red",lwd=2,ylab="Passengers",main="AirPassengers data")



### Fig-08: After differention and log transformation of Airpassengers.

**Comment:** Mean zero and variance constant. Hence, our data now Stationary.

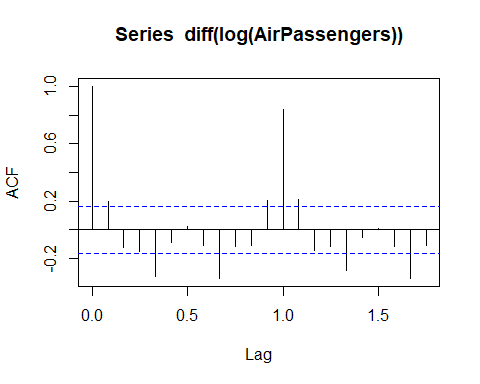
#After stationary we can fit ARIMA model  
# we need tseries  
library(tseries)  
# ARIMA(P=Order of AR,D=differencing order,  
# order of MA)  
#Selecting the value of p(AR)  
acf(AirPassengers)



### Fig-09: Auto Covariance Function (ACF)

**Comment:** Here the number of line cross or touch the blue line is no. of AR. we avoid the for zero value and the data is sessional that's why q=1.

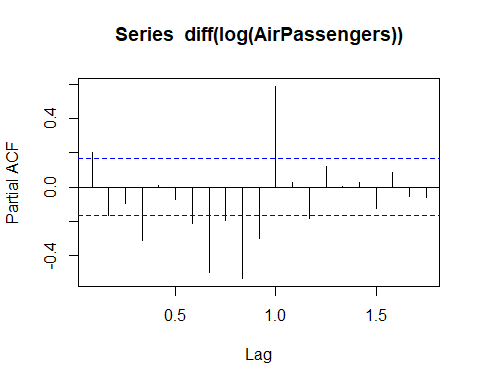
acf(diff(log(AirPassengers)))



### Fig-10: After Transformation of AirPassengers Data.

**Comment:** Here the number of line cross or touch the blue line is the value of q we ignore the first line for zero value and the next after 1 line because data is seasonal i.e. q=1.

#Here the number of line cross or touch the blue line is no.  
# of AR .we avoid the for zero value and the data  
# is sessional that's why q=1   
  
# Selecting the value of p  
pacf(diff(log(AirPassengers)))



### Fig-11: Partial Auto Covariance Function.

**Comment:** The line only at zero (0) cross the blue line. So, p=0.

# Fitting an ARIMA(0,1,1)  
# help(arima)  
  
m\_fit=arima(log(AirPassengers),order = c(0,1,1),seasonal =list(order=c(0,1,1),period=12))  
  
m\_fit

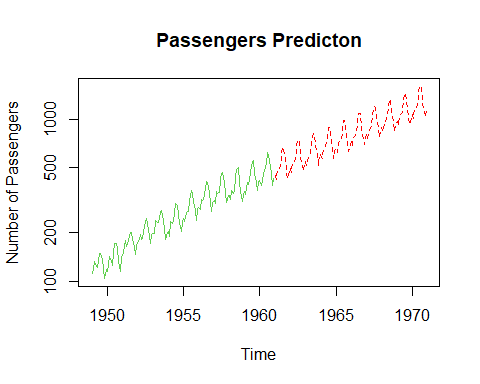
##   
## Call:  
## arima(x = log(AirPassengers), order = c(0, 1, 1), seasonal = list(order = c(0,   
## 1, 1), period = 12))  
##   
## Coefficients:  
## ma1 sma1  
## -0.4018 -0.5569  
## s.e. 0.0896 0.0731  
##   
## sigma^2 estimated as 0.001348: log likelihood = 244.7, aic = -483.4

#Forecast for next 10 years  
  
pred=predict(m\_fit,n.ahead = 10\*12)  
pred

## $pred  
## Jan Feb Mar Apr May Jun Jul Aug  
## 1961 6.110186 6.053775 6.171715 6.199300 6.232556 6.368779 6.507294 6.502906  
## 1962 6.206435 6.150025 6.267964 6.295550 6.328805 6.465028 6.603543 6.599156  
## 1963 6.302684 6.246274 6.364213 6.391799 6.425054 6.561277 6.699792 6.695405  
## 1964 6.398933 6.342523 6.460463 6.488048 6.521304 6.657526 6.796042 6.791654  
## 1965 6.495183 6.438772 6.556712 6.584297 6.617553 6.753776 6.892291 6.887903  
## 1966 6.591432 6.535022 6.652961 6.680547 6.713802 6.850025 6.988540 6.984153  
## 1967 6.687681 6.631271 6.749210 6.776796 6.810051 6.946274 7.084789 7.080402  
## 1968 6.783930 6.727520 6.845460 6.873045 6.906301 7.042523 7.181039 7.176651  
## 1969 6.880180 6.823769 6.941709 6.969294 7.002550 7.138773 7.277288 7.272900  
## 1970 6.976429 6.920019 7.037958 7.065544 7.098799 7.235022 7.373537 7.369150  
## Sep Oct Nov Dec  
## 1961 6.324698 6.209008 6.063487 6.168025  
## 1962 6.420947 6.305257 6.159737 6.264274  
## 1963 6.517197 6.401507 6.255986 6.360523  
## 1964 6.613446 6.497756 6.352235 6.456773  
## 1965 6.709695 6.594005 6.448484 6.553022  
## 1966 6.805944 6.690254 6.544734 6.649271  
## 1967 6.902194 6.786504 6.640983 6.745520  
## 1968 6.998443 6.882753 6.737232 6.841770  
## 1969 7.094692 6.979002 6.833481 6.938019  
## 1970 7.190941 7.075251 6.929731 7.034268  
##   
## $se  
## Jan Feb Mar Apr May Jun  
## 1961 0.03671562 0.04278291 0.04809072 0.05286830 0.05724856 0.06131670  
## 1962 0.09008475 0.09549708 0.10061869 0.10549195 0.11014981 0.11461854  
## 1963 0.14650643 0.15224985 0.15778435 0.16313118 0.16830825 0.17333075  
## 1964 0.20896657 0.21513653 0.22113442 0.22697386 0.23266679 0.23822371  
## 1965 0.27748210 0.28408309 0.29053414 0.29684503 0.30302451 0.30908048  
## 1966 0.35174476 0.35876289 0.36564634 0.37240257 0.37903840 0.38556004  
## 1967 0.43142043 0.43883816 0.44613258 0.45330963 0.46037481 0.46733319  
## 1968 0.51620376 0.52400376 0.53168935 0.53926541 0.54673651 0.55410688  
## 1969 0.60582584 0.61399203 0.62205103 0.63000694 0.63786363 0.64562471  
## 1970 0.70005133 0.70856907 0.71698563 0.72530453 0.73352910 0.74166246  
## Jul Aug Sep Oct Nov Dec  
## 1961 0.06513124 0.06873441 0.07215787 0.07542612 0.07855851 0.08157070  
## 1962 0.11891946 0.12307018 0.12708540 0.13097758 0.13475740 0.13843405  
## 1963 0.17821177 0.18296261 0.18759318 0.19211216 0.19652727 0.20084534  
## 1964 0.24365393 0.24896574 0.25416656 0.25926308 0.26426132 0.26916676  
## 1965 0.31502004 0.32084967 0.32657525 0.33220217 0.33773535 0.34317933  
## 1966 0.39197318 0.39828307 0.40449455 0.41061207 0.41663978 0.42258152  
## 1967 0.47418947 0.48094803 0.48761291 0.49418791 0.50067658 0.50708223  
## 1968 0.56138049 0.56856106 0.57565206 0.58265678 0.58957827 0.59641945  
## 1969 0.65329361 0.66087351 0.66836746 0.67577831 0.68310877 0.69036139  
## 1970 0.74970759 0.75766731 0.76554426 0.77334099 0.78105989 0.78870326

#final predicton

finalpred=exp(pred$pred)  
ts.plot(AirPassengers,finalpred,log="y",lty=c(1:3),col=c(3,'red'),  
 main="Passengers Predicton",ylab="Number of Passengers")



### Fig-12: AirPassengers Prediction.

**Comment:** Shows predict for next 10 years Air passengers. We indicate the future in red color.

#Checking accuracy of the prediction  
train\_data=ts(AirPassengers,frequency = 12,start=c(1949,1),end = c(1959,12))  
  
m\_fit2=arima(log(train\_data),order = c(0,1,1),seasonal =list(order=c(0,1,1),period=12))  
  
m\_fit2

##   
## Call:  
## arima(x = log(train\_data), order = c(0, 1, 1), seasonal = list(order = c(0,   
## 1, 1), period = 12))  
##   
## Coefficients:  
## ma1 sma1  
## -0.3484 -0.5623  
## s.e. 0.0943 0.0774  
##   
## sigma^2 estimated as 0.001313: log likelihood = 223.63, aic = -441.26

#Predict for one year  
pred2=predict(m\_fit2,n.ahead = 1\*12)  
pred2

## $pred  
## Jan Feb Mar Apr May Jun Jul Aug  
## 1960 6.038647 5.988763 6.145428 6.118993 6.159652 6.304666 6.433288 6.445958  
## Sep Oct Nov Dec  
## 1960 6.266719 6.136192 6.007899 6.114338  
##   
## $se  
## Jan Feb Mar Apr May Jun  
## 1960 0.03622957 0.04324114 0.04926471 0.05462807 0.05951001 0.06402074  
## Jul Aug Sep Oct Nov Dec  
## 1960 0.06823394 0.07220170 0.07596249 0.07954568 0.08297427 0.08626671

pred3=exp(pred2$pred)  
pred3

## Jan Feb Mar Apr May Jun Jul Aug  
## 1960 419.3252 398.9209 466.5792 454.4070 473.2633 547.1189 622.2166 630.1501  
## Sep Oct Nov Dec  
## 1960 526.7465 462.2898 406.6279 452.2965

pred\_1960=round(pred3,0)  
true\_1960=tail(AirPassengers,12)  
df=data.frame(pred\_1960,true\_1960)  
df

## pred\_1960 true\_1960  
## 1 419 417  
## 2 399 391  
## 3 467 419  
## 4 454 461  
## 5 473 472  
## 6 547 535  
## 7 622 622  
## 8 630 606  
## 9 527 508  
## 10 462 461  
## 11 407 390  
## 12 452 432

Comment: The difference between the true value and the predicted value is the smallestso the accuracy of the model is large.

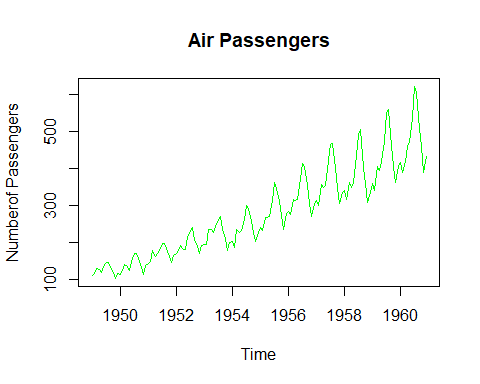
# Problem: -3

## Decomposition and modeling AirPassengers data.

#---------------lecture-5.3--------------------------  
#Decomposition and modeling   
#Data Air Passengers  
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

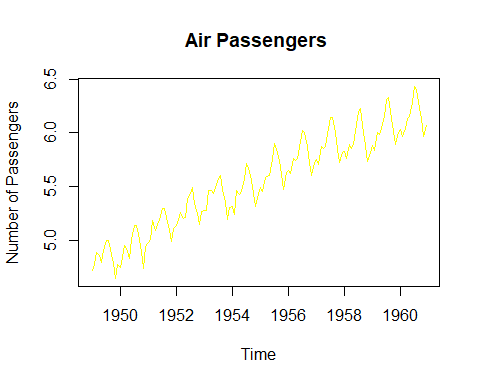
#Preparing the time series data  
AP=ts(AirPassengers,frequency = 12,start = c(1949,1))  
  
#Understanding data  
# Trend(T)  
#Seasonal(S)  
#Cyclical(C)  
#Irregular(I)  
plot(AP,col="green",lwd=1.4,main="Air Passengers",ylab="Numberof Passengers")



### Fig-13: Plotting Airpassengers Data.

**Comment:** Here, mean and variance both are changed over time also there is a upward trend and follow seasonality. Hence, this plot indicates non-stationary.

#Log-transform to fix-up variation  
AP2=log(AP)  
plot(AP2,lwd=1.2,col="Yellow",  
 main="Air Passengers ",ylab="Number of Passengers") #here we see the variance are same



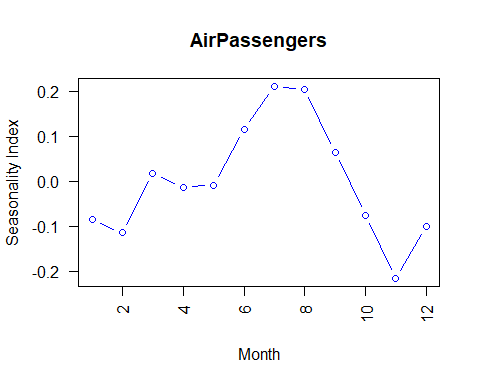
### Fig-14: Log Transformation of Air Passengers.

**Comment:** By log transform we fix up the variance. Here we see the variance is constant. However, still now trend, seasonality and random component are existed in the data.

#Decompose  
DAP=decompose((AP2))  
DAP$figure

## [1] -0.085815019 -0.114412848 0.018113355 -0.013045611 -0.008966106  
## [6] 0.115392997 0.210816435 0.204512399 0.064836351 -0.075271265  
## [11] -0.215845612 -0.100315075

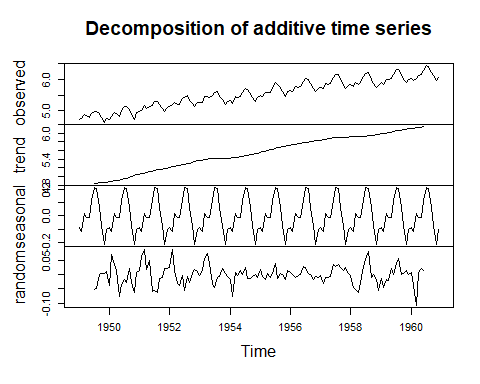
plot(DAP$figure,  
 type='b',  
 xlab = 'Month',  
 ylab = 'Seasonality Index',  
 col="blue",  
 las=2,  
 main = "AirPassengers",  
 lwd=1.3)



### Fig-15: Seasonality Plot

**Comment:** Decomposed value of the month March, April and May almost close to zero (0). Value of July and August are two times more and value of November two times less. Alternatively, we can say the seasonality index of July and August are four times more than November.

#Decompose plot   
plot(DAP)



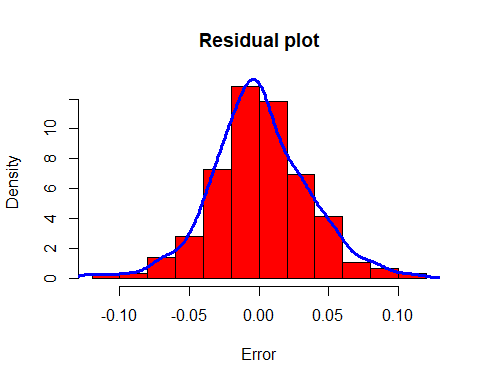
### Fig-16: Decomposition of Airpassengers.

**Comment:** Here we see the nature of the several components exist in the data.

#ARIMA(p,d,q) model  
library(forecast)  
model\_fit=auto.arima(AP2)  
model\_fit

## 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

#Residual plot  
hist(model\_fit$residuals,  
 col='red',  
 main = 'Residual plot',  
 xlab = 'Error',  
 freq=F)  
lines(density(model\_fit$residuals),  
 col='blue',lw=3)



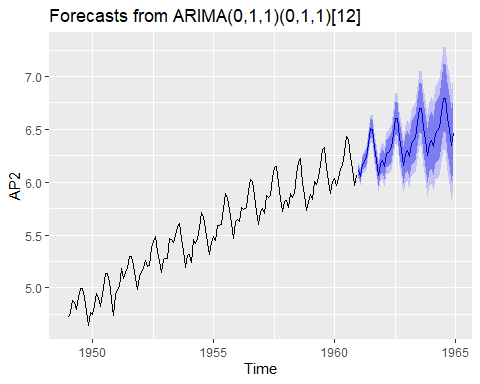
### Fig-17: Residual Plot.

**Comment:** we can see that most of the observations are around 0 and residual are symmetrically distributed. So, we can say that residuals are normally distributed with mean 0.

#Forecast for the next 4 years  
pred=forecast(model\_fit,4\*12)  
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.3.2

autoplot(pred) #deep blue 80% CI and light blue 95% CI



### Fig-18: Prediction of Next four year.

**Comment:** Our data was for 1960, data are forecasted for next 4 years which shown by blue line. 80% confidence interval is shown by deep blue region and 95% CI is shown by shaded blue line.

#Checking accuracy  
accuracy(pred)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.0005730622 0.03504883 0.02626034 0.01098898 0.4752815 0.2169522  
## ACF1  
## Training set 0.01443892

# Problem: -04

## Visualization of data by ggplot2.

## Anti diabetic drug subsidy (1991-2009)

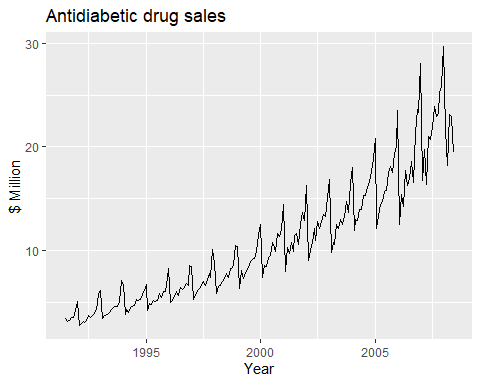
The a10 dataset, which is supposed to represent monthlyanti-diabetic drug subsidy in Australia from 1991 to 2008Visualization of data by ggplot2.

library(fpp)  
library(fpp2)

library(forecast)  
library(ggplot2)  
# help("autoplot")  
data(a10)  
a10

## Jan Feb Mar Apr May Jun Jul  
## 1991 3.526591  
## 1992 5.088335 2.814520 2.985811 3.204780 3.127578 3.270523 3.737851  
## 1993 6.192068 3.450857 3.772307 3.734303 3.905399 4.049687 4.315566  
## 1994 6.731473 3.841278 4.394076 4.075341 4.540645 4.645615 4.752607  
## 1995 6.749484 4.216067 4.949349 4.823045 5.194754 5.170787 5.256742  
## 1996 8.329452 5.069796 5.262557 5.597126 6.110296 5.689161 6.486849  
## 1997 8.524471 5.277918 5.714303 6.214529 6.411929 6.667716 7.050831  
## 1998 8.798513 5.918261 6.534493 6.675736 7.064201 7.383381 7.813496  
## 1999 10.391416 6.421535 8.062619 7.297739 7.936916 8.165323 8.717420  
## 2000 12.511462 7.457199 8.591191 8.474000 9.386803 9.560399 10.834295  
## 2001 14.497581 8.049275 10.312891 9.753358 10.850382 9.961719 11.443601  
## 2002 16.300269 9.053485 10.002449 10.788750 12.106705 10.954101 12.844566  
## 2003 16.828350 9.800215 10.816994 10.654223 12.512323 12.161210 12.998046  
## 2004 18.003768 11.938030 12.997900 12.882645 13.943447 13.989472 15.339097  
## 2005 20.778723 12.154552 13.402392 14.459239 14.795102 15.705248 15.829550  
## 2006 23.486694 12.536987 15.467018 14.233539 17.783058 16.291602 16.980282  
## 2007 28.038383 16.763869 19.792754 16.427305 21.000742 20.681002 21.834890  
## 2008 29.665356 21.654285 18.264945 23.107677 22.912510 19.431740   
## Aug Sep Oct Nov Dec  
## 1991 3.180891 3.252221 3.611003 3.565869 4.306371  
## 1992 3.558776 3.777202 3.924490 4.386531 5.810549  
## 1993 4.562185 4.608662 4.667851 5.093841 7.179962  
## 1994 5.350605 5.204455 5.301651 5.773742 6.204593  
## 1995 5.855277 5.490729 6.115293 6.088473 7.416598  
## 1996 6.300569 6.467476 6.828629 6.649078 8.606937  
## 1997 6.704919 7.250988 7.819733 7.398101 10.096233  
## 1998 7.431892 8.275117 8.260441 8.596156 10.558939  
## 1999 9.070964 9.177113 9.251887 9.933136 11.532974  
## 2000 10.643751 9.908162 11.710041 11.340151 12.079132  
## 2001 11.659239 10.647060 12.652134 13.674466 12.965735  
## 2002 12.196500 12.854748 13.542004 13.287640 15.134918  
## 2003 12.517276 13.268658 14.733622 13.669382 16.503966  
## 2004 15.370764 16.142005 16.685754 17.636728 18.869325  
## 2005 17.554701 18.100864 17.496668 19.347265 20.031291  
## 2006 18.612189 16.623343 21.430241 23.575517 23.334206  
## 2007 23.930204 22.930357 23.263340 25.250030 25.806090  
## 2008

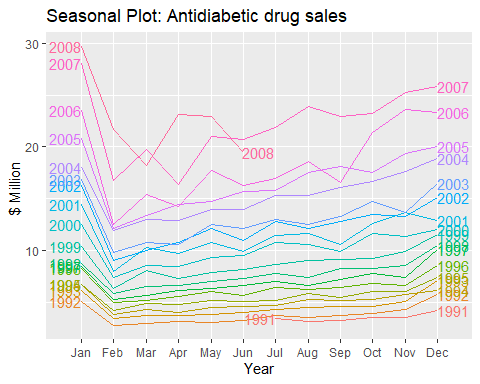
#-Plotting the dataset--  
autoplot(a10)+  
 ggtitle("Antidiabetic drug sales")+  
 ylab("$ Million")+  
 xlab("Year")



### Fig-19: Antidiabetic Drug sales plot

**Comment:** The plot show that Antidiabetic drug sales in Australia is increase. And This is seasonal upper trend data non stationary data.

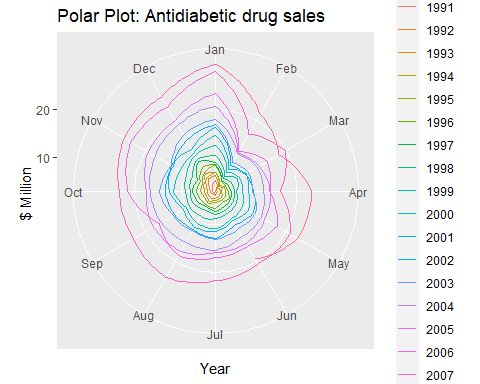
#Seasonal plot  
ggseasonplot(a10,year.labels = T,year.labels.left = T)+  
 ggtitle("Seasonal Plot: Antidiabetic drug sales")+  
 ylab("$ Million ")+  
 xlab("Year")



### Fig-20: Antidiabetic Drug sales Seasonal Plot.

**Comment:** Visualization of year wise monthly anti diabetic drug sells and comparative picture of several year. This is seasonal plot.

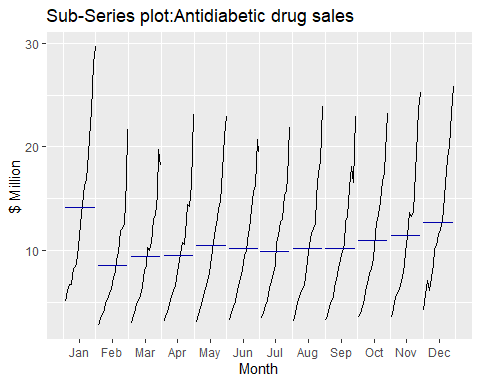
#Polar seasonal plot  
ggseasonplot(a10,polar =T)+  
 ggtitle("Polar Plot: Antidiabetic drug sales")+  
 ylab("$ Million ")+  
 xlab("Year")



### Fig-21: Polar Plot of Antidiabetic Drug sales.

**Comment:** This is year wise monthly anti diabetic drug sells and comparative picture of several year. This is polar plot.

#seasonal sub-series plot   
  
ggsubseriesplot(a10)+  
 ggtitle("Sub-Series plot:Antidiabetic drug sales")+  
 ylab('$ Million')



### Fig-22: Sub-plot Of Antidiabetic drug sales.

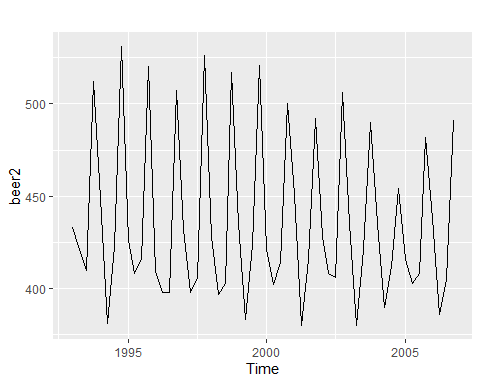
**Comment: The** plot represents the average sells of drug of several month.

#Visualizing ausbeer data

### Total quarterly beer production in Australia (in megalitres) from 1956:Q1 to 2010:Q2. ausbeer

## Qtr1 Qtr2 Qtr3 Qtr4  
## 1956 284 213 227 308  
## 1957 262 228 236 320  
## 1958 272 233 237 313  
## 1959 261 227 250 314  
## 1960 286 227 260 311  
## 1961 295 233 257 339  
## 1962 279 250 270 346  
## 1963 294 255 278 363  
## 1964 313 273 300 370  
## 1965 331 288 306 386  
## 1966 335 288 308 402  
## 1967 353 316 325 405  
## 1968 393 319 327 442  
## 1969 383 332 361 446  
## 1970 387 357 374 466  
## 1971 410 370 379 487  
## 1972 419 378 393 506  
## 1973 458 387 427 565  
## 1974 465 445 450 556  
## 1975 500 452 435 554  
## 1976 510 433 453 548  
## 1977 486 453 457 566  
## 1978 515 464 431 588  
## 1979 503 443 448 555  
## 1980 513 427 473 526  
## 1981 548 440 469 575  
## 1982 493 433 480 576  
## 1983 475 405 435 535  
## 1984 453 430 417 552  
## 1985 464 417 423 554  
## 1986 459 428 429 534  
## 1987 481 416 440 538  
## 1988 474 440 447 598  
## 1989 467 439 446 567  
## 1990 485 441 429 599  
## 1991 464 424 436 574  
## 1992 443 410 420 532  
## 1993 433 421 410 512  
## 1994 449 381 423 531  
## 1995 426 408 416 520  
## 1996 409 398 398 507  
## 1997 432 398 406 526  
## 1998 428 397 403 517  
## 1999 435 383 424 521  
## 2000 421 402 414 500  
## 2001 451 380 416 492  
## 2002 428 408 406 506  
## 2003 435 380 421 490  
## 2004 435 390 412 454  
## 2005 416 403 408 482  
## 2006 438 386 405 491  
## 2007 427 383 394 473  
## 2008 420 390 410 488  
## 2009 415 398 419 488  
## 2010 414 374

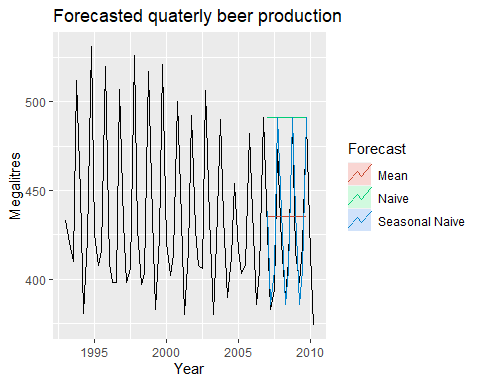
beer2=window(ausbeer,start=1993,end=c(2006,4))  
autoplot(beer2)



### Fig-23: Potting Australia Beer Production.

**Comment:** Time series plot of Australia beer production.

#--Simple forecasting methods  
# Average   
# Naive(work with most recent values)  
# Seasonal Naive  
# Drift  
  
beerfit1=meanf(beer2,h=12)  
beerfit2=rwf(beer2,h=12)  
beerfit3=snaive(beer2,h=12)  
  
#We plot full data set  
autoplot(window(ausbeer,start=1993))+  
 autolayer(beerfit1,series="Mean",PI=F)+  
 autolayer(beerfit2,series="Naive",PI=F)+  
 autolayer(beerfit3,series="Seasonal Naive",PI=F)+  
 xlab("Year")+ylab("Megalitres")+  
 ggtitle("Forecasted quaterly beer production ")+  
 guides(colour=guide\_legend(title = "Forecast"))



### Fig-24: Forecasted Quarterly Beer Production.

**Comment:** The figure shows the forecasted value using three descriptive methods.

#Accuracy   
beer3=window(ausbeer,start=2008)  
accuracy(beerfit1,beer3)

## ME RMSE MAE MPE MAPE MASE  
## Training set -2.436072e-14 43.42591 35.19005 -0.928920 7.875648 2.395134  
## Test set -6.892857e+00 36.33196 33.19643 -2.269843 7.659919 2.259443  
## ACF1 Theil's U  
## Training set -0.11704504 NA  
## Test set -0.07153733 0.7810277

accuracy(beerfit2,beer3)

## ME RMSE MAE MPE MAPE MASE  
## Training set 1.054545 65.06416 54.47273 -0.7865823 12.10955 3.707568  
## Test set -62.500000 71.96353 62.50000 -15.3314577 15.33146 4.253927  
## ACF1 Theil's U  
## Training set -0.22916007 NA  
## Test set -0.07153733 1.45868

accuracy(beerfit3,beer3)

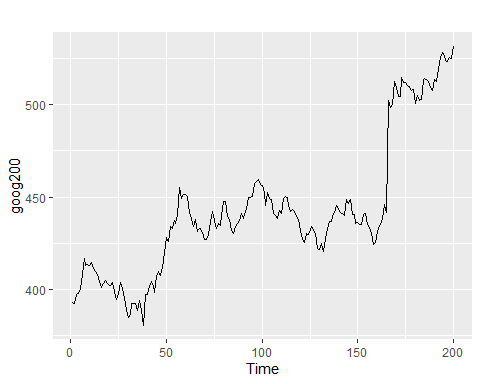
## ME RMSE MAE MPE MAPE MASE  
## Training set -1.076923 17.32162 14.69231 -0.3312997 3.415323 1.000000  
## Test set -1.500000 12.51000 10.25000 -0.3069842 2.457364 0.697644  
## ACF1 Theil's U  
## Training set -0.2880890 NA  
## Test set -0.1108185 0.2180679

#Another data  
# Closing stock prices of GOOG from the NASDAQ exchange, for 1000 consecutive trading days  
# between 25 February 2013 and 13 February 2017. Adjusted for splits. goog200 contains the first  
# 200 observations from goog

## Google Stock (2013-2017)

## Time Series:  
## Start = 1   
## End = 200   
## Frequency = 1   
## [1] 392.8300 392.5121 397.3059 398.0113 400.4902 408.0957 416.5905 413.0038  
## [9] 413.6099 413.0734 414.7127 411.1310 409.9884 408.1156 404.5190 401.2850  
## [17] 403.0386 404.7227 403.0088 402.5369 402.2040 403.5851 398.7366 394.5290  
## [25] 398.0063 403.8931 400.4952 394.9661 388.9950 384.9214 386.3124 392.5369  
## [33] 392.6412 392.4724 388.4386 394.1216 388.7516 380.4803 397.3506 397.4698  
## [41] 401.3397 404.0967 401.9358 398.1206 406.8836 409.6208 407.5642 412.1245  
## [49] 420.1275 427.9913 425.8453 433.9923 432.9243 437.2710 435.9297 440.6838  
## [57] 454.9857 449.0146 451.6524 451.3295 450.5546 441.8363 438.5427 433.8383  
## [65] 437.7876 431.3495 432.5666 432.7951 431.0117 426.7742 427.0723 429.5263  
## [73] 437.0226 442.2337 437.0623 433.1726 435.6664 434.6927 440.2615 447.4001  
## [81] 447.4299 439.5114 437.6187 432.0847 430.3013 434.0022 435.7012 437.3405  
## [89] 441.0713 438.3043 440.3510 443.8581 449.6206 449.6952 450.0677 457.1467  
## [97] 458.5178 459.3573 456.8337 456.3072 452.3976 445.4031 452.4075 448.9798  
## [105] 448.5327 440.9818 439.8144 438.2844 442.5814 441.0067 449.1885 450.3559  
## [113] 449.5759 445.3882 442.4473 443.4458 442.3281 439.8939 437.7777 432.0946  
## [121] 427.0524 425.6863 430.0281 429.9138 431.8562 434.0320 432.2933 430.3957  
## [129] 422.3282 421.5333 424.9511 420.7137 427.4101 432.9987 436.9381 436.9481  
## [137] 441.1557 441.4637 445.1994 443.6445 441.6624 441.0117 440.1920 448.7414  
## [145] 446.2923 448.6371 440.3857 440.5546 435.7807 436.2476 435.3634 435.1249  
## [153] 440.6341 441.1259 435.2144 433.3564 430.0728 424.0768 425.1647 431.3147  
## [161] 433.1776 435.2243 438.1552 446.1135 441.5233 502.4371 498.4083 500.2464  
## [169] 512.3725 509.4615 504.3199 504.2205 514.7719 511.8807 511.9602 510.2016  
## [177] 509.7396 507.4595 508.0705 500.7183 504.7322 502.0298 502.6209 512.8991  
## [185] 514.2701 513.4406 512.4421 509.2876 507.8519 513.6939 512.6110 519.5856  
## [193] 525.7853 528.1201 526.3715 523.8329 523.2269 525.6710 525.2537 531.4783

autoplot(goog200)

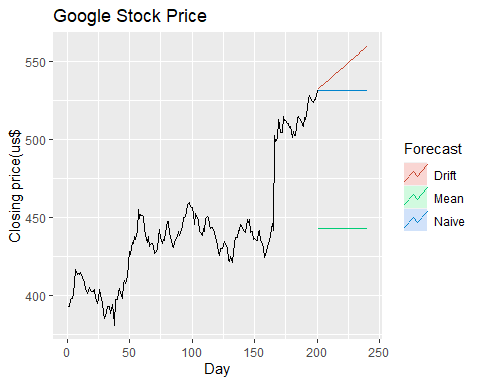


### Fig-25: Auto Plot Google data.

**Comment:** This figure shows time series plot of stock prices of GOOG from the NASDAQ exchange, for 1000 consecutive trading day.

#Forecast for next 40

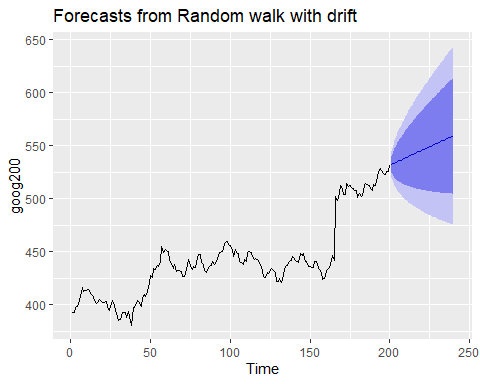
googf1=meanf(goog200,h=40)  
googf2=rwf(goog200,h=40)  
googf3=rwf(goog200,drift=T,h=40)  
  
autoplot(subset(goog200,end = 240))+  
 autolayer(googf1,series="Mean",PI=F)+  
 autolayer(googf2,series="Naive",PI=F)+  
 autolayer(googf3,series="Drift",PI=F)+  
 xlab("Day")+ylab("Closing price(us$")+  
 ggtitle("Google Stock Price")+  
 guides(colour=guide\_legend(title = "Forecast"))



### Fig-26: Forecast Google Stock Price.

**Comment:** Forecasted value of Mean, Naïve and Drift method. Here we see that Drift method provide best forecast.

autoplot(googf3)



### Fig-27: Forecast From Random Walk.

**Comment:** This is forecasted value for Drift method where deep blue region shows 80% CI and light blue shaded shows 95% CI.

# Problem: -05

## Analyzing and forecasting monthly rainfall of Bangladesh.

df=read.csv(file.choose(),header = T,sep = ",")  
df

head(df)

## Year Station Month Rainfall StationIndex  
## 1 1970 Barisal 1 0 2  
## 2 1970 Barisal 2 24 2  
## 3 1970 Barisal 3 5 2  
## 4 1970 Barisal 4 91 2  
## 5 1970 Barisal 5 124 2  
## 6 1970 Barisal 6 408 2

View(df)  
#Packages  
library(ggplot2)  
library(dplyr)

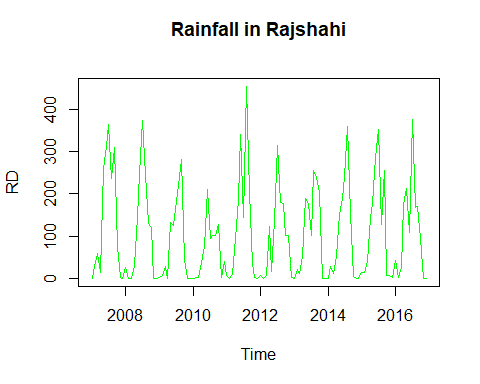
library(forecast)  
  
#Preparing data for analysis  
df1<-filter(df,df$Year>="2007")  
df1=filter(df1,Station=="Rajshahi")  
View(df1)  
  
df2=select(df1,Rainfall)  
df2

## Rainfall  
## 1 0  
## 2 27  
## 3 59  
## 4 13  
## 5 260  
## 6 313  
## 7 364  
## 8 236  
## 9 309  
## 10 76  
## 11 1  
## 12 0  
## 13 26  
## 14 0  
## 15 0  
## 16 30  
## 17 144  
## 18 247  
## 19 373  
## 20 245  
## 21 129  
## 22 121  
## 23 0  
## 24 0  
## 25 1  
## 26 7  
## 27 28  
## 28 0  
## 29 131  
## 30 126  
## 31 183  
## 32 240  
## 33 282  
## 34 45  
## 35 0  
## 36 0  
## 37 0  
## 38 2  
## 39 2  
## 40 37  
## 41 75  
## 42 211  
## 43 94  
## 44 101  
## 45 101  
## 46 127  
## 47 3  
## 48 39  
## 49 6  
## 50 0  
## 51 10  
## 52 94  
## 53 187  
## 54 341  
## 55 144  
## 56 454  
## 57 203  
## 58 35  
## 59 1  
## 60 0  
## 61 6  
## 62 0  
## 63 6  
## 64 123  
## 65 17  
## 66 137  
## 67 314  
## 68 179  
## 69 178  
## 70 102  
## 71 101  
## 72 1  
## 73 0  
## 74 22  
## 75 12  
## 76 51  
## 77 188  
## 78 178  
## 79 101  
## 80 254  
## 81 238  
## 82 204  
## 83 0  
## 84 0  
## 85 0  
## 86 27  
## 87 12  
## 88 51  
## 89 151  
## 90 188  
## 91 242  
## 92 359  
## 93 153  
## 94 5  
## 95 0  
## 96 0  
## 97 14  
## 98 14  
## 99 39  
## 100 144  
## 101 177  
## 102 285  
## 103 353  
## 104 127  
## 105 254  
## 106 7  
## 107 6  
## 108 1  
## 109 42  
## 110 3  
## 111 25  
## 112 175  
## 113 212  
## 114 109  
## 115 376  
## 116 168  
## 117 170  
## 118 95  
## 119 0  
## 120 0

#Convert the value in continuous formate  
  
df2$Rainfall=as.numeric(df2$Rainfall)  
#Convert into a time series data  
  
RD=ts(df2$Rainfall,start = 2007,frequency = 12)  
RD

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
## 2007 0 27 59 13 260 313 364 236 309 76 1 0  
## 2008 26 0 0 30 144 247 373 245 129 121 0 0  
## 2009 1 7 28 0 131 126 183 240 282 45 0 0  
## 2010 0 2 2 37 75 211 94 101 101 127 3 39  
## 2011 6 0 10 94 187 341 144 454 203 35 1 0  
## 2012 6 0 6 123 17 137 314 179 178 102 101 1  
## 2013 0 22 12 51 188 178 101 254 238 204 0 0  
## 2014 0 27 12 51 151 188 242 359 153 5 0 0  
## 2015 14 14 39 144 177 285 353 127 254 7 6 1  
## 2016 42 3 25 175 212 109 376 168 170 95 0 0

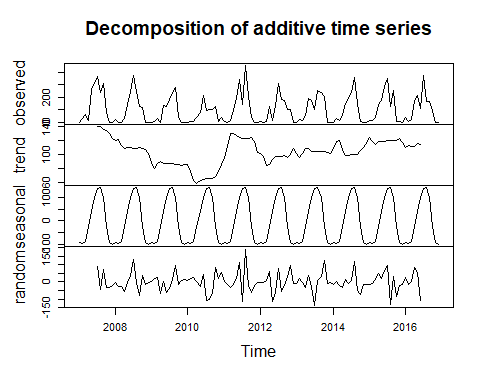
#Time series Plot  
plot(RD,lwd=1.2,col="green",main="Rainfall in Rajshahi  
 ")



### Fig-28: Plotting Rainfall Time Series Data.

**Comment:** Here mean and variance more or less constant, seasonal component exists.

#Decompose  
RDR=decompose(RD)  
plot(RDR)



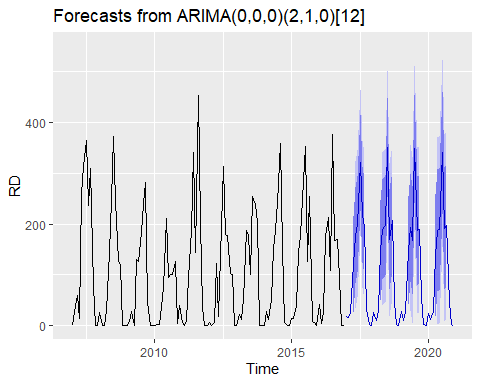
### Fig-29: Decomposition of Rainfall Data.

**Comment:** The figure shows visualization of several time series components.

#Fitting an arima model  
fit\_model=auto.arima(RD)  
fit\_model

## Series: RD   
## ARIMA(0,0,0)(2,1,0)[12]   
##   
## Coefficients:  
## sar1 sar2  
## -0.7061 -0.3468  
## s.e. 0.0922 0.0936  
##   
## sigma^2 = 5435: log likelihood = -620.14  
## AIC=1246.28 AICc=1246.51 BIC=1254.33

#Forecast  
pred=forecast(fit\_model,4\*12)  
autoplot(pred,PI=T)+  
 ylim(0,550)



### Fig-30: Forecast Next Four Year.

**Comment:** Here, deep blue region shows 80% CI and light blue shaded region shows 95% CI and now forecasted value with 80% and 95% CI show only positive value.