**Practical No.1 Estimation and Elimination of trend component various difference method.**

**Q1)**

data()

data("AirPassengers")

AirPassengers

plot(AirPassengers,main="Original AirPassengers data",xlab="Year",ylab="Passengers",type="o")

trend\_estimate\_3yr=filter(AirPassengers,rep(1/36,36),sides=2)

trend\_estimate\_3yr

trend\_estimate\_5yr=filter(AirPassengers,rep(1/60,60),sides=2)

trend\_estimate\_5yr

trend\_eliminate\_3yr=AirPassengers-trend\_estimate\_3yr

trend\_eliminate\_3yr

trend\_eliminate\_5yr=AirPassengers-trend\_estimate\_5yr

trend\_eliminate\_5yr

plot(trend\_estimate\_3yr,main="estimated trend of AirPassenger data",xlab="year",ylab="estimated AirPassengers",type="o")

plot(trend\_estimate\_5yr,main="estimated trend of AirPassenger data",xlab="year",ylab="estimated AirPassengers",type="o")

plot(trend\_eliminate\_3yr,main="eliminate trend of AirPassenger data",xlab="year",ylab="eliminate AirPassengers",type="o")

plot(trend\_eliminate\_5yr,main="eliminate trend of AirPassenger data",xlab="year",ylab="eliminate AirPassengers",type="o")

first\_diff\_series=diff(AirPassengers)

first\_diff\_series

plot(first\_diff\_series,main="differencing series of AirPassenger data",xlab="year",ylab="estimated AirPassengers",type="o")

**Q2)**

data()

data("sunspots")

sunspots

plot(sunspots,main="Original sunspots data",xlab="year",ylab="Observation",type="o")

alpha=0.2

alpha

smoothed\_value=numeric(length(sunspots))

smoothed\_value

for(t in 2:length(sunspots))

{

smoothed\_value[t]=alpha\*sunspots[t-1]+(1-alpha)\*smoothed\_value[t-1]

}

estimated\_trend=smoothed\_value

estimated\_trend

plot(estimated\_trend,main="Estimated trend(Expoential Smoothing)",xlab="Year",ylab="observation")

detrend\_data=sunspots-estimated\_trend

detrend\_data

plot(detrend\_data,main="Detrend data(Trend Estimate)",xlab="Year",ylab="Observation",type="o")

first\_order\_series=diff(sunspots)

first\_order\_series

plot(first\_order\_series,main="monthly sunspots",xlab="year",ylab="Observation",col="red",type="o")

**Practical No.2:Estimation and Elimination of Seasonal component**

**Q1)** data=c(486,474,434,441,435,401,414,414,386,405,411,389,414,426,410,441,459,449,486,510,506,549,579,581,630,666,674,729,771,785)

data

length(data)

plot(data)

plot(data,main="original and smoothed Time Series",ylab="value",xlab="Time",col="blue",type="o")

#it shows their is increasing trend present and no seasonality visible

filter\_coefficient=c(-1,4,3,4,-1)/9

filter\_coefficient

smoothed\_data=filter(data,filter\_coefficient,sides=2)

smoothed\_data

plot(smoothed\_data,main="smoothed Time Series",ylab="value",xlab="Time",type="o",col="red")

mad\_value=mean(abs(data-smoothed\_data),na.rm=TRUE)

mad\_value

msd\_value=mean((data-smoothed\_data)^2,na.rm=TRUE)

msd\_value

cat("mean Square Deviation (MAD):",msd\_value,"/n")

cat("mean Square Deviation (MSD):",mad\_value,"/n")

**Q2)**

data()

original\_data=AirPassengers

original\_data

plot(original\_data,main="Original AirPassengers data",xlab="year",ylab="Passenger",type="o",col="red")

ma\_12=filter(original\_data,rep(1/12,12),sides=2)

ma\_12

plot(ma\_12)

seasonal\_component=original\_data/ma\_12

seasonal\_component

plot(seasonal\_component)

deseasonalized\_data=original\_data/seasonal\_component

deseasonalized\_data

plot(deseasonalized\_data,main="deseasonalized data",xlab="year",ylab="Deseasonalized passengers",col="red",type="o")

diff\_deseasonalized\_data=diff(deseasonalized\_data)

diff\_deseasonalized\_data

plot(diff\_deseasonalized\_data,main="deseasonalized data",xlab="year",ylab="Deseasonalized passengers",col="red",type="o")

**Practical No.4:identification of MA and AR process and its order selection**

**Q1)**

library(tseries)

library(forecast)

data=read.csv("C:\\Users\\DELL\\Desktop\\pollution.csv")

data

head(data)

T\_data=data[,'pm2.5']

T\_data

head(T\_data)

T\_data=na.omit(T\_data)

T\_data

view(T\_data)

head(T\_data)

plot(T\_data)

adf\_test=adf.test(T\_data)

adf\_test

acf(T\_data,main="Autocorrelation Function(ACF)")

pacf(T\_data,main="Partial Autocorrelation Function(ACF)")

T\_data\_diff=diff(T\_data)

T\_data\_diff

head(T\_data\_diff)

acf(T\_data\_diff,main="Autocorrelation Function(ACF)")

pacf(T\_data\_diff,main=" Partial Autocorrelation Function(ACF)")

ar\_model=ar(T\_data\_diff,order=4)

ar\_model

ma\_model=ma(T\_data\_diff,order=2)

ma\_model

head(ma\_model)

ar\_forecast=forecast(ar\_model,h=12)

ar\_forecast

ma\_forecast=forecast(ma\_model,h=12)

ma\_forecast

plot(ma\_forecast)

plot(ar\_forecast)

**Practical No.5:Yule Walker Equation for AR(p) Model**

**Q1)**

data()

data(AirPassengers)

AirPassengers

plot(AirPassengers)

acf\_values=acf(AirPassengers,plot=FALSE)

acf\_values

acf\_vals=acf\_values$acf

acf\_vals

gamma\_o=acf\_vals[1,1,1]

gamma\_o

gamma\_1=acf\_vals[2,1,1]

gamma\_1

gamma\_2=acf\_vals[3,1,1]

gamma\_2

Yule\_walker\_matrix=matrix(c(gamma\_o,gamma\_1,gamma\_2),nrow=2,byrow=TRUE)

Yule\_walker\_matrix

R1=c(gamma\_1,gamma\_2)

R1

Ar=solve(Yule\_walker\_matrix,R1)

Ar

**Practical No.6:Fitting MA Model using Least Square Regression**

**Q2)**

library(tseries)

data()

data(sunspot.month)

sunspot.month

plot(sunspot.month,main="Monthly sunspots Data",ylab="Sunspots Number",xlab="Time")

adf\_test=adf.test(sunspot.month)

adf\_test

ar\_model=ar(sunspot.month,order.max=2,method="yw")

ar\_model

resid\_est=ar\_model$resid

resid\_est

n=length(resid\_est)

n

y=sunspot.month[3:n]

y

e1=resid\_est[1:(n-1)]

e1

e2=resid\_est[2:(n-2)]

e2

head(e2)

ma2\_model=lm(y~e1+e2)

ma2\_model

summary(ma2\_model)

**Practical No.7: Residual Analysis and Diagnostic Checking**

**Q1)**

library(tseries)

library(forecast)

data()

data(AirPassengers)

AirPassengers

plot(AirPassengers,main="AirPassengers Dataset",ylab="No of Passengers",xlab="Year")

fit=auto.arima(AirPassengers)

fit

residuals=residuals(fit)

residuals

head(residuals)

plot(residuals,main="residuals form fitted",xlab="year",ylab="residuals")

acf(residuals,main="acf of residuals")

ljung\_box\_test=Box.test(residuals,lag=20,type="Ljung-Box")

ljung\_box\_test

print(ljung\_box\_test)

if(ljung\_box\_test$p.value>0.05)

{print("Residual are independent, model is appropriate")}

else

{print("Residual are autocorrelated, model might need improvement")}

**Practical No.8: Fitting ARMA Model**

**Q1)**

data=read.csv("C:\\Users\\DELL\\Desktop\\Amazon.csv")

data

head(data)

library(tseries)

library(forecast)

data=data[,"rt"]

data

plot(data)

adf\_test=adf.test(data)

adf\_test

acf=acf(data,main="ACF of amazon data")

pacf=pacf(data,main="PACF of amazon data")

p=1

q=1

model=arima(data,order=c(p,0,q))

model

bic\_value=BIC(model)

bic\_value

aic\_value=AIC(model)

aic\_value

aicc\_value=AIC(model,k=log(length(data)))

aicc\_value

**Q2)**

data=read.csv("C:\\Users\\DELL\\Desktop\\Gold.csv")

data

head(data)

library(tseries)

library(forecast)

data=data[,"VALUE"]

data

plot(data)

adf\_test=adf.test(data)

adf\_test

data\_diff=diff(data)

data\_diff

adf\_test=adf.test(data\_diff)

adf\_test

acf=acf(data,main="ACF of Gold data")

pacf=pacf(data,main="PACF of Gold data")

p=1

q=1

model=arima(data,order=c(p,0,q))

model

bic\_value=BIC(model)

bic\_value

aic\_value=AIC(model)

aic\_value

aicc\_value=AIC(model,k=log(length(data)))

aicc\_value

**Practical No.10: Identification of ARIMA(p,d,q) Process and order selection.**

**Q1)**

library(forecast)

data=read.csv("C:\\Users\\DELL\\Desktop\\Gold.csv")

data

T\_data=data[,"VALUE"]

T\_data

plot(T\_data)

gold\_diff=(T\_data)

gold\_diff

adf\_test=adf.test(gold\_diff)

adf\_test

print(adf\_test\_diff)

acf(gold\_diff,main="ACF of Diff Series")

pacf(gold\_diff,main="PACF of Diff Series")

arima=arima(gold\_diff,order=c(1,1,1))

arima

**Practical No.11: select a series and obtain mean ,variance and auto covariance autocorrelation upto lag 5.**

**Q1)**

data=c(47,64,23,71,38,64,55,41,59,48,71,35,57,40,58,44,80,55,37,74,51,57,50,60,45,57,50,45,25,59,50,71,56,74,58,58,45,54,36,54,48,55,45,57,50,62,44,64,43,52,38,59,55,41,53,49,34,35,54,45,68,38,50,60,39,59,40,57,54,23)

data

length(data)

m=mean(data)

m

v=var(data)

v

auto=acf(data,lag=5,plot=T)

auto

data\_ts=ts(data)

data\_ts

acvf\_result=acf(data\_ts,lag.max=5,type="covariance",plot=T)

acvf\_result

acvf\_result$acf

**Q2)**

data1=read.csv("C:\\Users\\DELL\\Desktop\\Gold.csv")

data1

head(data1)

m=mean(data1)

m

v=var(data1)

v

auto=acf(data1,lag=50,plot=T)

auto

data\_ts=ts(data)

data\_ts

acvf\_result=acf(data\_ts,lag.max=50,type="covariance",plot=T)

acvf\_result

acvf\_result$acf

**Practical No.13:Stratified random sample(Various type of allocation method**)

**Q1)**

x1=c(797,773,748,734,588,577,507,507,457,438,415,401,387,381,324,315)

x1

N1=length(x1)

N1

x2=c(314,298,296,258,256,243,238,237,172,172,163,162,161,159,153,144,121,120,119,118,118,116,116,113,235,235,216,208,201,192,180,179,138,138,138,138,136,132,130,126,113,110,110,108,106,104,101,100)

x2

N2=length(x2)

N2

X=c(x1,x2)

X

length(X)

N=length(X)

N

n=24

s1=sample(X,n,replace=TRUE)

s1

m1=mean(s1)

m1

S1=sum((s1-m1)^2)/(n-1)

S1

SD1=sqrt(((N-1)\*S1)/(N\*n))

SD1

#Sample for SRSWOR

s2=sample(X,n,replace=FALSE)

s2

m2=mean(s2)

m2

S2=sum((s2-m2)^2)/(n-1)

S2

SD2=sqrt(((N-1)\*S2)/(N\*n))

SD2

n1=(n/N)\*N1

n1

n2=(n/N)\*N2

n2

W1=N1/N

W1

W2=N2/N

W2

s2=sample(x2,n2,TRUE)

s2

s=c(s1,s2)

s

m3=mean(m1)

m3

m4=mean(m2)

m4

Sq1=sum((s1-m3)^2)(n1-1)

Sq1

Sq2=sum((s2-m4)^2)(n2-1)

Sq2

SD=sqrt(((1/n)-)(1/N))\*(W1\*Sq1+W2\*Sq2))

SD

#neymann allocation

t1=var(x1)

t1

t2=var(x2)

t2

n1=(W1\*sqrt(t1)/(W1\*sqrt(t1)+W2\*sqrt(t2)))\*n

n1

n2=(W2\*sqrt(t2)/(W1\*sqrt(t1)+W2\*sqrt(t2)))\*n

n2

S1=sample(x1,n1,FALSE)

S1

S1=sample(x2,n2,FALSE)

S1

Si1=sum((S1-mean(S1))^2)/(n1-1)

Si1

Si2=sum((S2-mean(S2))^2)/(n2-1)

Si2

sd=sqrt((((W!\*sqrt(Si1))^+(W2\*sqrt(Si2))^2)/n)-(W1\*Si1+W2\*Si2)/N))

sd

**Practical no.14: sratified random sampling and regression method of estimation**

**Q1)**

x=c(1054,973,1089,1054,894)

x

y=c(10316,7025,10512,8963,8783)

y

n=5

n

xbar=mean(x)

xbar

ybar=mean(y)

ybar

Rn=ybar/xbar

Rn

YR=Rn\*xbar

YR

s2x=var(x)

s2x

s2y=var(y)

s2y

sxy=cov(x,y)

sxy

S=s2y+Rn^2\*s2x-2\*Rn\*sxy

S

SE=sqrt(((1/n)-(1/N))\*S)

SE

beta=sxy/s2x

beta

P=s2y+beta^2\*s2x-2\*beta\*sxy

p

se=sqrt(((1/n)-(1/N)\*p))

se

Xbar=988.75

Yd=ybar+beta\*(Xbar-xbar)

Yd

eff=Yd/YR

eff

**Q2)**

x=c(61,42,50,58,67,45,39,57,71,53)

x

y=c(59,47,52,60,67,48,44,58,76,58)

y

n=10

n

N=200

N

xbar=mean(x)

xbar

ybar=mean(y)

ybar

Rn=ybar/xbar

Rn

YR=Rn\*xbar

YR

s2x=var(x)

s2x

s2y=var(y)

s2y

sxy=cov(x,y)

sxy

S=s2y+Rn^2\*s2x-2\*Rn\*sxy

S

SE=sqrt(((1/n)-(1/N))\*S)

SE

beta=sxy/s2x

beta

P=s2y+beta^2\*s2x-2\*beta\*sxy

p

se=sqrt(((1/n)-(1/N))\*p)

se

Xbar=11600/200

Xbar

Yd=ybar+beta\*(Xbar-xbar)

Yd

eff=Yd/YR

eff

# Practical No. 17-Jacknife and bootstrap method

**Question 1**

>x=c(8,26,6.33,10.4,5.27,5.35,5.61,6.12,6.

19,5.2,7.01,8.74,7.78,7.01,6,6.5,8,5.12,7.4

1,6.52,6.21,12.28,5.6,5.38,6.6,8.74)

* x

[1] 8.00 26.00 6.33 10.40 5.27 5.35

5.61 6.12 6.19 5.20 7.01 8.74 7.78

7.01

[15] 6.00 6.50 8.00 5.12 7.41 6.52

6.21 12.28 5.60 5.38 6.60 8.74

* original\_cv=(sd(x)/mean(x))\*100
* original\_cv

[1] 53.50442

* n\_bootstrap=1000
* n\_bootstrap

[1] 1000

* bootstrap\_means=numeric(n\_bootstrap)
* bootstrap\_vars=numeric(n\_bootstrap)
* bootstrap\_cvs=numeric(n\_bootstrap)
* for(i in 1:n\_bootstrap)

+ {

+

bootstrap\_sample=sample(x,size=length(x)

,replace=TRUE)

+

bootstrap\_means[i]=mean(bootstrap\_samp le)

+ bootstrap\_vars[i]=var(bootstrap\_sample)

+

bootstrap\_cvs[i]=(sd(bootstrap\_sample)/m ean(bootstrap\_sample))\*100

+

* #bootstrap\_means
* #bootstrap\_vars
* #bootstrap\_cvs
* hist(bootstrap\_cvs)

>lowerquartile=quantile(bootstrap\_cvs,0.0 25)

* lowerquartile 2.5%

17.0763

>upperquartile=quantile(bootstrap\_cvs,0.0 95)

* upperquartile 9.5%

20.95215

* bias\_estimate=mean(bootstrap\_cvs)- original\_cv

bias\_estimate

[1] -8.174178

* corrected\_cv=original\_cv-bias\_estimate
* corrected\_cv

[1] 61.6786

# Question 2

* x=c(24,26,32,36,43,52,62,56,52,21)
* y=c(22,28,5,18,14,14,8,8,10,24)
* original\_corr=cor(x,y)
* original\_corr

[1] -0.7284006

* n=length(x)
* n

[1] 10

* jackknife\_corr=numeric(n)
* jackknife\_corr

[1] 0 0 0 0 0 0 0 0 0 0

* for(i in 1:n)

+ {

+ x\_jackknife=x[-i]

+ y\_jackknife=y[-i]

+

jackknife\_corr[i]=cor(x\_jackknife,y\_jackk nife)

+ }

* jackknife\_corr

[1] -0.6927381 -0.6920750 -0.9451279 -

0.7248160 -0.7276008 -0.7447716 -

0.6926761

[8] -0.6920055 -0.7104119 -0.6667885

* ​

jackknife\_estimator=mean(jackknife\_corr)

* jackknife\_estimator

[1] -0.7289011

* bias=jackknife\_estimator-jackknife\_corr
* bias

[1] -0.036163078 -0.036826127

0.216226749 -0.004085145 -0.001300329

0.015870459

[7] -0.036225077 -0.036895587 -

0.018489236 -0.062112630

* sd=sqrt(((n-1)/n)\*sum((jackknife\_corr- jackknife\_estimator)^2))
* sd

[1] 0.2256223

# Question 3

* x=c(22,26,58,54,30,35,12,28)
* x

[1] 22 26 58 54 30 35 12 28

* n=length(x)
* n

[1] 8

* m=mean(x)
* m

[1] 33.125

* u2=sum((x-m)^2)/n
* u2

[1] 214.3594

* u3=sum((x-m)^3)/n
* u3

[1] 1645.488

* beta=(u3^2)/(u2^3)
* beta

[1] 0.274892

* gamma=sqrt(beta)
* gamma

[1] 0.5243015

* n\_bootstrap=8
* n\_bootstrap

[1] 8

* m1=numeric(n\_bootstrap)
* m1

[1] 0 0 0 0 0 0 0 0

* mu2=numeric(n\_bootstrap)
* mu2

[1] 0 0 0 0 0 0 0 0

* mu3=numeric(n\_bootstrap)
* mu3

[1] 0 0 0 0 0 0 0 0

* b1=numeric(n\_bootstrap)
* b1

[1] 0 0 0 0 0 0 0 0

* g1=numeric(n\_bootstrap)
* g1

[1] 0 0 0 0 0 0 0 0

* for(i in 1:n\_bootstrap)

+ {

+

bootstrap\_sample=sample(x,size=length(x)

,replace=TRUE)

+ m1[i]=mean(bootstrap\_sample)

+ mu2[i]=sum((bootstrap\_sample- m1[i])^2)/n

+ mu3[i]=sum((bootstrap\_sample- m1[i])^3)/n

+ b1[i]=(mu3[i]^2)/(mu2[i]^3)

+ g1[i]=sqrt(b1[i])

+ }

* m1

[1] 33.375 32.875 33.625 37.125 35.625

34.000 29.125 29.875

* mu2

[1] 180.2344 140.1094 181.9844 269.3594

222.2344 210.7500 127.3594 124.6094

* mu3

[1] 863.3086 828.6680 2308.9102

318.3633 813.6914 1109.2500 1268.6133

995.3555

* b1

[1] 0.127297318 0.249666065

0.884528364 0.005186205 0.060323391

0.131448648 0.779050750

[8] 0.512040438

* g1

[1] 0.35678750 0.49966595 0.94049368

0.07201531 0.24560821 0.36255848

0.88263852

[8] 0.71557001

* mg1=mean(g1)
* mg1

[1] 0.5094172

* bias=mg1-gamma
* bias

[1] 0.05180846

* se\_gamma=sd(g1)
* se\_gamma

[1] 0.362564

# Question 4

* x=c(32,4,16,7,12,27)
* y=c(2300,30,1500,15,700,1800)
* mx=mean(x)
* mx

[1] 16.33333

* my=mean(y)
* my

[1] 1057.5

* n\_bootstrap=6
* n\_bootstrap

[1] 6

* m1=numeric(n\_bootstrap)
* m1

[1] 0 0 0 0 0 0

* m2=numeric(n\_bootstrap)
* m2

[1] 0 0 0 0 0 0

* m3=numeric(n\_bootstrap)
* m3

[1] 0 0 0 0 0 0

* m4=numeric(n\_bootstrap)
* m4

[1] 0 0 0 0 0 0

* for(i in 1:n\_bootstrap){

+

bootstrap\_sample1=sample(x,size=length( x),replace=TRUE)

+

bootstrap\_sample2=sample(y,size=length( y),replace=TRUE)

+ m1[i]=mean(bootstrap\_sample1)

+ m2[i]=mean(bootstrap\_sample2)

+ }

* m1

[1] 20.16667 23.50000 24.83333 14.50000

21.16667 19.00000

* m2

[1] 1326.6667 985.8333 1035.8333

729.1667 1519.1667 607.5000

* for(i in 1:n\_bootstrap)

+ {

+ x\_jackknife=x[-i]

+ y\_jackknife=y[-i]

+ m3[i]=mean(x\_jackknife)

+ m4[i]=mean(y\_jackknife)

+ }

* m3

[1] 13.2 18.8 16.4 18.2 17.2 14.2

* m4

[1] 809 1263 969 1266 1129 909

* mean(m1)-mx

[1] 4.194444