######################################

#Practical No. 1: Estimation and elimination of trend component.

#Variate difference Method

######################

##QUESTION:1

library(datasets)

data(package="datasets")

data("AirPassengers")

AirPassengers

plot(AirPassengers, main="Original Air Passengers Data", xlab="Year",ylab="Passengers", col="blue",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(moving average)",xlab="Year",ylab="trend",col="red",type="o")

plot(trend\_estimate\_5yr,main="Estimated trend(moving average)",xlab="Year",ylab="trend",col="red",type="o")

plot(trend\_eliminate\_3yr,main="Estimated trend(moving average)",xlab="Year",ylab="detrended passenger",col="green",type="o")

plot(trend\_eliminate\_5yr,main="Estimated trend(moving average)",xlab="Year",ylab="detrended passenger",col="green",type="o")

first\_diff\_series=diff(AirPassengers)

first\_diff\_series

plot(first\_diff\_series,main="Difference AirPassengers data",xlab="year",ylab="passengers",col="blue",type="o")

##QUESTION 2

data("sunspots")

sunspots

plot(sunspots,main="Monthly sunspots data", xlab="year",ylab="sunspots number",col="blue",type="o")

alpha=0.2

alpha

smoothed\_values=numeric(length(sunspots))

smoothed\_values

smoothed\_values[1]=sunspots[1]

smoothed\_values

for(t in 2:length(sunspots))

{

smoothed\_values[t] <- alpha\* sunspots[t-1]+(1-alpha)\* smoothed\_values[t-1]

}

estimated\_trend=smoothed\_values

estimated\_trend

detrended\_data=sunspots-estimated\_trend

detrended\_data

plot(estimated\_trend,main="Estimated Trend(exponential smoothing)",xlab="year",ylab="Trend", col="red", type="o")

plot(detrended\_data,main="Detrended Data(Trend Eliminated)",xlab="year", ylab="detrended sunspots", col="red", type="o")

first\_diff\_series=diff(sunspots)

first\_diff\_series

plot(first\_diff\_series,main="Monthly Sunspots data",xlab="year",ylab="sunspots number",col="blue",type="o")

######################################

#Practical No 2: Estimation and Elimination of Seasonal Component

###########################

###QUESTION:1

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

plot(data,main="Original and smoothed Time series",ylab="value",xlab="time",col="blue")

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

filter\_coefficients

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

smoothed\_data

plot(smoothed\_data,main="smoothed Time Series",ylab="Value",xlab="Time",col="blue")

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

mad\_value

cat("Mean Absolute Deviation (MAD);",mad\_value,"\n")

#Mean Absolute Deviation (MAD); 117.3077

cat("Mean Squareed Deviation (MSD);",msd\_value,"\n")

#Mean Squareed Deviation (MSD); 117.3077

###QUESTION:2

library(datasets)

data(package="datasets")

data("AirPassengers")

AirPassengers

ts\_data=AirPassengers

ts\_data

plot(ts\_data, main="Monthly Airline Pssengers (1949-1960)",ylab="Number of passengers",xlab="year")

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

ma\_12

plot(ma\_12)

seasonal\_component=ts\_data/ ma\_12

seasonal\_component

plot(seasonal\_component)

deseasonalized\_data=ts\_data-seasonal\_component

deseasonalized\_data

plot(deseasonalized\_data, main="Deseasonalized Data",xlab="year",ylab="Desonalized passengers")

######################################

#Practical No 3: Examining Stationarity. Sample ACF and PACF

###########################

###QUESTION:1

install.packages("tseries")

install.packages("forecast")

library(tseries)

library(forecast)

data("LakeHuron")

LakeHuron

plot(LakeHuron)

L\_mean=mean(LakeHuron)

L\_mean

adf\_test=adf.test(LakeHuron)

adf\_test

LakeHuron\_diff=diff(LakeHuron)

LakeHuron\_diff

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

adf\_test\_diff

acf(LakeHuron\_diff, main="sample ACF of LakeHuron Time Series")

pacf(LakeHuron\_diff, main="sample PACF of LakeHuron Time Series")

adf\_result= adf.test(LakeHuron)

p = adf\_result$p.value

if(p > 0.05) {

print("H0: The time series is not stationary")

} else {

print("H1: The time series is stationary")

}

###QUESTION:2

library(tseries)

library(forecast)

data("BJsales")

BJsales

plot(BJsales)

L\_mean=mean(BJsales)

L\_mean

adf\_test=adf.test(BJsales)

adf\_test

BJsales\_diff=diff(BJsales)

BJsales\_diff

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

adf\_test\_diff

acf(BJsales\_diff, main="sample ACF of BJsales Time Series")

pacf(BJsales\_diff, main="sample PACF of BJsales Time Series")

if(p>0.05)

{

print("h0:the time series not a stationary")

}else

{

print("h1:the time series is stationary")

}

###QUESTION:3

library(tseries)

library(forecast)

data("JohnsonJohnson")

JohnsonJohnson

plot(JohnsonJohnson)

L\_mean=mean(JohnsonJohnson)

L\_mean

adf\_test=adf.test(JohnsonJohnson)

adf\_test

JohnsonJohnson\_diff=diff(JohnsonJohnson)

JohnsonJohnson\_diff

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

adf\_test\_diff

acf(JohnsonJohnson\_diff, main="sample ACF of JohnsonJohnson Time Series")

pacf(JohnsonJohnson\_diff, main="sample PACF of JohnsonJohnson Time Series")

if(p>0.05)

{

print("h0:the time series not a stationary")

}else

{

print("h1:the time series is stationary")

}

###QUESTION:4

library(tseries)

library(forecast)

data("AirPassengers")

AirPassengers

plot(AirPassengers)

L\_mean=mean(AirPassengers)

L\_mean

adf\_test=adf.test(AirPassengers)

adf\_test

AirPassengers\_diff=diff(AirPassengers)

AirPassengers\_diff

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

adf\_test\_diff

acf(AirPassengers\_diff, main="sample ACF of AirPassengers Time Series")

pacf(AirPassengers\_diff, main="sample PACF of AirPassengers Time Series")

if(p>0.05)

{

print("h0:the time series not a stationary")

}else

{

print("h1:the time series is stationary")

}

#########################

#Practical No. 4: Identification of moving average (MA) and Auto regressive (AR) process

#and its order selection

#####################

QUESTION:1

library(tseries)

library(forecast)

data=read.csv("C:\\Users\\Dell07\\Downloads\\pollution.csv")

data

View(data)

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

T\_data

View(T\_data)

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

T\_data

View(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 (PACF)")

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

T\_data\_diff

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

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

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

ar\_model

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

ma\_model

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

ar\_forecast

plot(ar\_forecast)

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

ma\_forecast

plot(ma\_forecast)

#################

#Practical No 5 : Yule-Walker estimation for AR(p) model.

#########################

###QUESTION:1

data("AirPassengers")

AirPassengers

plot(AirPassengers)

acf(AirPassengers)

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

acf\_values

acf\_vals=acf\_values$acf

acf\_vals

gamma\_0=acf\_vals[1,1,1]

gamma\_0

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

gamma\_1

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

gamma\_2

yule\_walker\_matrix= matrix(c(gamma\_0,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 squares regression.

##########################

###QUESTION:1

library(tseries)

data("sunspot.month")

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

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

print(adf\_test)

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

resid\_est = ar\_model$resid

resid\_est =na.omit(resid\_est)

n = length(resid\_est)

y = sunspot.month[3:(n+2)]

e1 = resid\_est[1:n]

e2 = resid\_est[1:(n-1)] # or shift properly for true lag-2

e2 = c(NA, e2) # shift e2 to line up

# Trim all to same length (removing first NA row)

df = data.frame(y = y, e1 = e1, e2 = e2)

df = na.omit(df)

# Fit MA(2 model using regression

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

summary(ma2\_model)

####################

#Practicle 7: Residual Analysis and Diagnostic checking

#########################

###QUESTION:1

library(forecast)

library(tseries)

library(datasets)

data(package="datasets")

data("AirPassengers")

AirPassengers

plot(AirPassengers, main="AirPassengers Datasets", xlab="Year",ylab="Number of Passengers")

fit=auto.arima(AirPassengers)

fit

residuals=residuals(fit)

residuals

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

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

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

print(ljung\_box\_test)

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

print("resuals are independent, suggesting that the model is appropriate.")

} else {

print("residuals are autocorrelated, suggesting the model might need improvement.")

}

##################

#Practical No 8 : Fitting ARMA Model

############################

###QUESTION:1

library(tseries)

data=read.csv("C:\\Users\\nikam\\Music\\msc 2 dataset\\Amazon.csv");data

data=data[ ,"rt"];data

plot(data)

adf\_test=adf.test(data)

adf\_test

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

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

model=arima(data,order=c(1,0,1));model

###QUESTION:2

library(tseries)

Data=read.csv("C:\\Users\\nikam\\Music\\msc 2 dataset\\Gold.csv")

Data

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(Data\_diff)

pacf(Data\_diff)

model=arima(data,order=c(1,0,1));model

bic=BIC(model);bic

aicc=AIC(model);aicc

##################################

#Practical No. 9: Dickey Fuller Unit Root Test.

##############################

#QUESTION:1

library(tseries)

library(forecast)

data=read.csv("C:\\Users\\lenovo\\Downloads\\monthly-housing.csv")

head(data)

data=na.omit(data)

cat("H0:The data is not stationary","\n","H1:The data is stationary","\n")

data\_hpi=data[,"hpi"]

data\_numsold=data[,"numsold"]

plot(data\_hpi,main="Monthly household data(hpi)",xlab="month",ylab="hpi observation")

#The time series data shows an increasing trend, with no clear seasonal pattern.That

#means data is not stationay due to increasing trend and seasonality.

adf\_test\_hpi=adf.test(data\_hpi);adf\_test\_hpi

p\_val\_hpi=adf\_test\_hpi$p.value;p\_val\_hpi

if (p\_val\_hpi>0.05)

{

print("Accept H0,Therefor time series is not stationary")

} else

{

print("Reject H0,Therefor time series is stationary")

}

# The given time series data of monthly

#household\_hpi is not stationary by

#graphical its show upward trend and by

#using ADF test is has a unit root(i.e.it is

#non-stationary) because p value of adf test is 0.883 which is greater than 0.05.

##ii)

plot(data\_numsold,main="Monthly household data\_numsold",xlab="month",ylab="numsold observation")

adf\_test\_numsold=adf.test(data\_numsold);adf\_test\_numsold

Augmented Dickey-Fuller Test

p\_val\_numsold=adf\_test\_numsold$p.value;p\_val\_numsold

if (p\_val\_numsold>0.05)

if (p\_val\_numsold>0.05)

{

print("Accept H0,Therefor time series is not stationary")

} else

{

print("Reject H0,Therefor time series is stationary")

}

# The given time series data of monthly household(numsold) is not stationary by

#graphical its show upward trend and by using ADF test is has a unit root(i.e.it is

#non-stationary) because p value of adf test is 0.758 which is greater than 0.05.

###QUESTION:2

library(tseries)

library(forecast

data=read.csv("C:\\Users\\lenovo\\Downloads\\Gold.csv")

head(data)

data=na.omit(data)

cat("H0:The data is not stationary","\n","H1:The data is stationary","\n")

data=data[,"VALUE"]

plot(data,main="Monthly Gold Data",xlab="month",ylab="VALUE")

#The plot shows a long-term upward trend in gold prices with periods of rapid growth,

#peaks, and volatility. That means data is non-stationary

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

p\_val=adf\_test$p.value;p\_val

if (p\_val>0.05)

{

print("Accept H0,Therefor time series is not stationary")

} else {

print("Reject H0,Therefor time series is stationary")

}

#The ADF test p-value is 0.6653, which is greater than 0.05, indicating that the time

#series is non-stationary.

######################

#Practical:10: Identification of ARIMA(p d q)

# process and order selection .

###########################

QUESTION:1

library(forecast)

data=read.csv("C:\\Users\\Dell07\\Downloads\\Gold.csv")

data

T\_data=data[,"VALUE"]

T\_data

plot(T\_data)

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

print(adf\_test\_diff)

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

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

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

print(arima)

#################################

#Practical No 11 : Select a series and obtain Mean, Variance and

#autocovariance Autocorrelation upto lag 5.

#################################

###QUESTION:1

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=F);auto

data\_ts=ts(data);data\_ts

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

###QUESTION:2

data=read.csv("C:\\Users\\nikam\\Music\\msc 2 dataset\\Gold.csv");data

Gold\_data=data[,"VALUE"];Gold\_data

View(Gold\_data)

plot(Gold\_data)

mean=mean(Gold\_data);mean

var=var(Gold\_data);var

auto=acf(data,lag=5,plot=F)$acf[,,1]

auto

acvf=function(x,lag){

n=length(x)

mean\_x=mean(x)

acvf\_vals=numeric(lag+1)

for(h in 0:lag){

acvf\_vals[h+1]=sum((x[1:(n-h)]- mean\_x)\*(x[(h+1):n]-mean\_x))/n

}

return(acvf\_vals)

}

acvf=acvf(Gold\_data,50)

acvf

############################

#Practical 12:Compute and plot the Empirical Autocovariance function

#and the Empirical Autocorrelation.

###########################

###QUESTION:1

data("AirPassengers")

AirPassengers

plot(AirPassengers,xlab="Year",ylab="Monthly AirPassengers Data",col="Red",typr="o")

library(tseries)

adf\_test=adf.test(AirPassengers)

adf\_test

p\_value=0.01

if(p\_value>0.05)

{

print("Data is stationary")

}else

{

print("Data is not stationary")

}

acvf=acf(AirPassengers,type="covariance")

acf=acf(AirPassengers)

diff\_data=diff(AirPassengers)

diff\_data

acvf=acf(diff\_data,type="covariance")

acf1=acf(diff\_data)

pacf=pacf(diff\_data)

pacf

p=2

q=1

#################

#Practical No 13 stratified random sampling

############################

###QUESTION:1

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

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

x2

N1=length(x1);N1

N2=length(x2);N2

N=N1+N2;N

n=24

s1=sample(c(x1,x2),n,TRUE);s1

s2=sample(c(x1,x2),n,FALSE);s2

s1bar=mean(s1);s1bar

s2bar=mean(s2);s2bar

a=sum(s1^2);a

ssq=(a-(n\*(s1bar^2)))/n-1;ssq

est=((N-1)\*ssq)/(N\*n);est

se=sqrt(est);se

b=sum(s2^2);b

ssq=(b-(n\*s2bar^2))/n-1;ssq

est1=((N-n)\*ssq)/(N\*n);est1

se1=sqrt(est1);se1

#################

#Practical No 14:stratified random sampling

#ratio and regression method estimation)

#######################

###question:1

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

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

n=5

N=12;N

xbr=mean(x);xbr

ybr=mean(y);ybr

Rn=ybr/xbr;Rn

#estimate ratio

YR=Rn\*xbr;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

Xbr=988.75

#estimate reg

Yd=ybr+beta\*(Xbr-xbr);Yd

eff=Yd/YR;eff

###QUESTION:2

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

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

N=200

n=10

xbr=mean(x);xbr

ybr=mean(y);ybr

Rn=ybr/xbr;Rn

#estimate ratio

YR=Rn\*xbr;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

Xbr=11600/200;Xbr

#estimate reg

Yd=ybr+beta\*(Xbr-xbr);Yd

55.45998

eff=Yd/YR;eff

#######################

#Practical:15 Circular Systematic Sampling

###############################

###QUESTION:1

x=c(26,28,11,16,07,22,44,26,31,26,16,9,22,26,17,39,21,14,40,30,27,20,25,39,24,25,18,44,55,39,37,14,14,24,18,17,14,38,36,29,04,05,11,09,25,16,13,22,18,06,36,20,43,27,20,21,18,19,24,30,20,21,15,14,13,09,25,17,07,30,21,26,16,18,11,19,27,29)

N=length(x); N

n=8

k=floor(N / n); k

y=seq(1, N); y

Z=matrix(0, nrow=n, ncol=N); Z

for (j in 1:N) {

r[j] = sample(y, 1)

for (i in 1:n) {

idx = r[j] + i \* k

if (idx <= N) {

Z[i, j] = idx

} else {

Z[i, j] = idx - N

}

}

}

sample=matrix(x[Z], nrow=n, ncol=N);sample

mean=colMeans(sample);mean

var=var(col\_means); var

s=sample(x, 8, replace=FALSE); s

m=mean(s); m

v=var(s); v

###########################

#Practical No 16: Cluster sampling with equal and unequal sampling

######################################

###QUESTION:1

N=77

M=4

n=15

T1=c(5.53,4.48,0.69,15.79)

T2=c(26.11,10.93,10.08,11.18)

T3=c(11.08,0.65,4.21,7.56)

T4=c(12.66,32.52,16.92,37.02)

T5=c(0.87,3.56,4.81,27.54)

T6=c(6.40,11.68,40.05,5.12)

T7=c(54.21,34.63,52.55,37.20)

T8=c(1.24,35.97,29.54,25.28)

T9=c(37.94,47.07,19.64,28.11)

T10=c(54.92,17.69,26.24,6.77)

T11=c(25.52,38.10,24.74,1.90)

T12=c(45.98,5.17,1.17,6.53)

T13=c(7.13,34.35,12.18,9.86)

T14=c(14.23,16.89,28.93,21.70)

T15=c(3.53,40.76,5.15,1.25)

m=matrix(c(T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15),nrow=15,ncol=4,byrow=TRUE)

m

mean=rowMeans(m)#yibar

mean

yn.b=sum(mean)/n#yn.doublebar

yn.b

d=matrix(c(mean),nrow=15,ncol=1,byrow=TRUE)

d

newd=d[,rep(1,4)]

newd

sub=m-newd;sub

sub2=sub^2;sub2

sisq=rowSums(sub2)/(M-1)

sisq

swsq=sum(sisq)/n

swsq

sbsq=sum((mean-yn.b)^2)/(n-1)

sbsq

Ssq=sum((m-yn.b)^2)/(n\*M-1)

Ssq

roh=(((n-1)\*M\*sbsq)-(n\*swsq))/(((n-1)\*M\*sbsq)+(n\*(M-1)\*swsq))

roh

var=((1/n)-(1/N))\*sbsq

var

eff=Ssq/(M\*sbsq)

eff

###QUESTION:2

data=read.csv("C:\\Users\\nikam\\downlods\\dd.csv")

data

M=600

N=35

n=6

m1=9

m2=2

m3=8

m4=70

m5=1

m6=35

Mbar=M/N

Mbar

d=data.frame(data)

y1=d[1:9,1]

y2=d[1:2,2]

y3=d[,4]

y5=d[1,5]

y6=[1:35,6]

y1.b=sum(y1)/m1

y1.b

y2.b=sum(y2)/m3

y2.b

y3.b=sum(y3)/m3

y3.b

y4.b=sum(y4)/m4

y4.b

y5.b=sum(y5)/m5

y5.b

y6.b=sum(y6)/m6

y6.b

ynstb=((m1\*y1.b)+(m2\*y2.b)+(m3\*y3.b)+(m4\*y4.b)+(m5\*y5.b)+(m6\*y6.b))/(n\*Mbar)

ynstb

F=n/N

F

sb2=(((m1\*y1.b/Mbar)-ynstb)^2+((m2\*y2.b/Mbar)-ynstb)^2+((m3.b/Mbar)-ynstb)^2+((m4\*y4.b/Mbar)-ynstb)^2+((m5\*y5.b/Mbar)-ynstb)^2+((m6\*y6.b/Mbar)-ynstb)^2/(n-1)

sb2

var=((1-F)/n)\*sb2

var

data1=c(y1,y2,y3,y4,y5,y6)

s2=sum((data1-ynstb)^2)/(n\*M-1)

s2

E=s2/(Mbar\*sb2)

E

###############################

#Practical No 17 : Jackknife and Bootstrap methods of estimation For Ratio

#and Regression coefficient,Coefficient of variation,Correlation coefficient)

############################

###QUESTION:1

x=c(8,26,6.33,10.4,5.27,5.35,5.61,6.12,6.19,5.2,7.01,8.74,7.78,7.01,6,6.5,5.12,7.41,6.52,6.21,12.28,5.6,5.38,6.6,8.74)

x

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

n\_bootstrap=1000; n\_bootstrap

bootstrap\_means=numeric(n\_bootstrap); bootstrap\_means

bootstrap\_vars=numeric(n\_bootstrap);bootstrap\_vars

bootstrap\_cvs=numeric(n\_bootstrap);bootstrap\_cvs

for(i in 1:n\_bootstrap){

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

bootstrap\_means[i]=mean(bootstrap\_sample)

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

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

}

bootstrap\_means

bootstrap\_vars

bootstrap\_cvs

hist(bootstrap\_cvs)

lowerquantile=quantile(bootstrap\_cvs,0.025);lowerquantile

upperquantile=quantile(bootstrap\_cvs,0.095);upperquantile

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

###QUESTION:2

x=c(24,26,32,36,43,52,62,56,52,21);x

y=c(22,28,5,18,14,14,8,8,10,24);y

original\_corr=cor(x,y);original\_cv

n=length(x);n

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

for(i in 1:n){

x\_jackknife=x[-i]

y\_jackknife=y[-i]

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

}

jackknife\_corr

jackknife\_estimator=mean(jackknife\_corr);jackknife\_estimator

bias=jackknife\_estimator-jackknife\_corr;bias

se=sqrt((n-1)\*var(jackknife\_corr));se

###QUESTION:3

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

n=length(x);n

m=mean(x);m

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

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

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

gamma=sqrt(beta);gamma

n\_bootstrap=8;n\_bootstrap

m1=numeric(n\_bootstrap);m1

m2=numeric(n\_bootstrap);m2

mu2=numeric(n\_bootstrap);mu2

mu3=numeric(n\_bootstrap);mu3

b1=numeric(n\_bootstrap);b1

g1=numeric(n\_bootstrap);g1

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

mu2

mu3

b1

g1

mg1=mean(g1);mg1

bias=mg1-gamma;bias

bias=mg1-gamma;bias

###QUESTION:4

x=c(32,4,16,7,12,27);x

y=c(2300,30,1500,150,700,1800);y

mx=mean(x);mx

my=mean(y);my

n\_bootstrap=6;n\_bootstrap

m1=numeric(n\_bootstrap);m1

m2=numeric(n\_bootstrap);m2

m3=numeric(n\_bootstrap);m3

m4=numeric(n\_bootstrap);m4

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

m2

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

m4

mean(m1)-mx

######################

#PRACTICLE:18 Two stage sampling

##############################

###QUESTION:1

N=17

n=10

mi=2

Mi=c(15,19,19,16,16,18,20,18,16,16)

y1i=c(47,38,43,55,59,39,71,35,63,63)

y2i=c(30,51,35,41,45,38,64,46,47,47)

mb=sum(mi)/n

mb

ui=Mi/mb

ui

yib=(y1i+y2i)/2

yib

yb=sum(ui\*yib)/n

yb

v=(ui\*yib)

v

sb2=sum((v-yb)^2)/(n-1)

sb2

yij=c(47,38,43,55,59,39,71,35,63,63,30,51,35,41,45,38,64,64,47,47)

yij

si2=sum((yij-yib)^2)/(mi-1)

si2

a=((1/n)-(1-N))\*sb2

a

s=((1/mi)-(1-Mi))

s

b=sum(ui^2\*((1/mi)-(1/Mi))\*si2)/(N\*n)

bv=a+b

v

sqrt(v)

###QESTION:2

N=100

n=10

M=16

m=4

yim=matrix(c(4.31,4.78,3.86,4.02,4.61,4.12,3.16,4.12,3.72,4.11,4.17,5.70,3.75,4.58,3.62,3.78,3.12,4.68,3.92,4.32,4.08,4.24,4.04,5,4.28,4.66,4.04,3.84,4.20,4.72,4.96,3.08,4.40,4.66,3,4.04,4.16,4.24,4.32,4.02),nrow=10,ncol=4,byrow=TRUE)

yim

yimb=rowSums(yim)/m

yimb

ybb=(1/n)\*sum(yimb)

ybb

sb2=(1/(n-1))\*sum((yimb-ybb)^2)

sb2

d=matrix(yimb,nrow=10,ncol=1,byrow=TRUE)

d

newd=d[,rep(1,4)]

newd

sw2=sum((yim-newd)^2)/(n\*(m-1))

sw2

v=(((1/n)-(1/N))\*sb2)+((1/n)\*((1/m)-(1/M))\*sw2)

v

sd1=sqrt(v)

sd1

#######################

#Practical No 19:PPS

############################

###QUESTION:1

n=c(1,2,3,4,5,6,7,8,9,10);n

trees=c(150,50,80,100,200,160,40,220,60,140);

trees

c=c()

for(i in 2:length(trees))

{

trees[i]=trees[i-1]+trees[i]

c[i]=trees[i]

}

c

ns=4;ns

sn=numeric(ns);sn

s=numeric(ns);s

for(j in 1:ns){

sn[j]=sample(1:c[length(c)],1)

sample\_index=which(c>=sn[j])[1]

s[j]=sample\_index

}

sn

s

###QUESTION:2

n=10;n

N=800;N

x=c(5511,865,2535,3523,8368,7357,5131,4654,1146,1165);x

y=c(4824,924,1948,3013,7378,5506,4051,4060,809,1013);y

Total\_population=415149

Total\_population

p=x/sum(x);p

Ty=(N/n)\*sum(y/p);Ty

var=(N/n)\*sum((y/p-y)^2\*p);var

SE=sqrt(var);SE

###QUESTION:3

states=c("AI","AK","AZ","AR","CA","CO","CT","DE","FL","GA","HI","ID","IL","IN","IA");states

non\_real\_estate=c(348.334, 3.433, 431.439, 848.317, 3928.732, 906.281, 4.373, 43.229, 464.516, 540.696, 38.067, 1006.036, 2610.572, 1022.782, 3909.738)

non\_real\_estate

real\_estate=c(409, 2.605, 54.633, 907.7, 1343.461, 315.809, 7.13, 42.808, 825.748, 939.46, 40.775, 53.753, 2131.048, 1213.024, 2327.025)

real\_estate

data=data.frame(states,non\_real\_estate,real\_estate)

data

total\_non\_real\_estate=sum(non\_real\_estate);total\_non\_real\_estate

selection\_prob=non\_real\_estate/total\_non\_real\_estate;selection\_prob

#Draw a PPSWR sample of size 5

sample\_indices=sample(1:length(states),5,replace=TRUE,prob=selection\_prob);sample\_indices

sample\_data=data[sample\_indices,];sample\_data

#Estimate the population total for real estate farm loans

population\_total\_est=(total\_non\_real\_estate/5)\*sum(sample\_data$real\_estate/sample\_data$non\_real\_estate)

population\_total\_est

#Estimate the population mean for real estate farm loans

population\_mean\_est=population\_total\_est/length(states)

population\_mean\_est

#Estimate the variance of the population total estimate

variance\_total\_est=(total\_non\_real\_estate^2/5)\*(sum((sample\_data$real\_estate/sample\_data$non\_real\_estate-population\_mean\_est)^2)/4)

variance\_total\_est

#Estimate the variance of the population mean estimate

variance\_mean\_est=variance\_total\_est/length(states)^2

variance\_mean\_est

cat("PPSWR Sample:\n")

print(sample\_data)

cat("Estimated Population Total for Real Estate Farm Loans:",population\_total\_est,"\n")

cat("Estimated Population Mean for Real Estate Farm Loans:",population\_mean\_est,"\n")

cat("Estimated Variance of Population Total Estimate:",variance\_total\_est,"\n")