1. **PCA**

str(USArrests)

pca<-prcomp(USArrests,scale=TRUE,center=TRUE,retx=TRUE)

pca$rotation<--1\*pca$rotation

pca$x<--1\*pca$x

summary(pca)

pca

pca$x

biplot(pca, main='Biplot', scale=0)

pca\_var<-pca$sdev^2

pro\_var<-pca\_var/sum(pca\_var)

plot(pro\_var,xlab="Principal Components",ylab="Proportion of Variance Explained",ylim=c(0,1),type='b',main="Scree Plot")

which(cumsum(pro\_var)>0.9)

plot(cumsum(pro\_var),xlab="Principal Component",ylab="Cumulative Proportion of Variance Explained",ylim=c(0,1),type="b")

1. **Sample variance-covariance matrix**

y1 <- c(35, 35, 40, 10, 6, 20, 35, 35, 35, 30)

y2 <- c(3.5, 4.9, 30.0, 2.8, 2.7, 2.8, 4.6, 10.9, 8.0, 1.6)

y3 <- c(2.80, 2.70, 4.38, 3.21, 2.73, 2.81, 2.88, 2.90, 3.28, 3.20)

data <- data.frame(y1, y2, y3)

cov\_matrix <- cov(data)

print(cov\_matrix)

cor\_matrix <- cor(data)

print(cor\_matrix)

1. **Linear Regression Model**

data(mtcars)

#visualise the data to identify outliers

boxplot(mtcars, main = "Boxplot of mtcars Data")

#Removing outliers

Q1 <- quantile(mtcars$mpg, 0.25)

Q3 <- quantile(mtcars$mpg, 0.75)

IQR <- IQR(mtcars$mpg)

mtcars\_filtered <- mtcars[mtcars$mpg >= (Q1 - 1.5\*IQR)&mtcars$mpg <= (Q3 + 1.5\*IQR), ]

#Create a linear regression model

model <- lm(mpg~.,data = mtcars\_filtered)

summary(model)

1. **KS Test for gamma & normal distribution**

data = rivers

data

Q1 = quantile(data, 0.25)

Q3 = quantile(data, 0.75)

IQR = Q3 - Q1

lower\_bound = Q1-1.5\*IQR

upper\_bound = Q3+1.5\*IQR

indices = which(data < lower\_bound | data>upper\_bound)

cleaned\_data <- data[-indices]

cleaned\_data

library(MASS)

gamma\_fit <- fitdistr(cleaned\_data, 'gamma')

gamma\_fit

gamma\_cdf = pgamma(cleaned\_data, shape = 4.906370024 , rate = 0.010276277)

gamma\_cdf

ks\_test\_result = ks.test(cleaned\_data, "pgamma", shape = 4.906370024, rate = 0.010276277)

ks\_test\_result

hist(cleaned\_data, probability = TRUE, main = 'Histogram of Cleaned Rivers', xlab = 'River Length')

curve(dgamma(x, shape = 4.906370024, rate = 0.010276277), add = TRUE, col='red')

qqplot(cleaned\_data, qgamma(ppoints(length(cleaned\_data))), shape = 4.906370024, rate = 0.010276277)

1. **KS test for checking if specific variable gives good fit to normal & Weibull distribution**

data = rock

data

library(MASS)

ks\_area\_normal <- ks.test(data$area, 'pnorm', mean=mean(data$area), sd= sd(data$area))

ks\_area\_normal

weibull\_fit <- fitdistr(data$shape, 'weibull')

weibull\_fit

ks\_shape\_weibull <- ks.test(data$shape, 'pweibull', shape =2.74966322 , scale = 0.24537815)

ks\_shape\_weibull

hist(rock$shape,, probability = TRUE, main = 'Weibull Fit for Shape')

1. **Differentiating the expression and value of differentials at given point**

library("numDeriv")

f1 = function(x){2\*x^3+x^2+x}

grad(f1, 2)

f2 = function(x){(2\*x-1)/(1+x-x^2)}

grad(f2,3)

grad(f2,1)

f3 = function(x){-3\*x^5+8\*x^(0.5)-9.3}

grad(f3,0.5)

1. **Cayley Hamilton**

# Define matrix

x <- matrix(c(1, 2, 3, 5, 7, 11, 13, 17, 19), nrow = 3, ncol = 3, byrow = TRUE)

# Install and load 'pracma' package

install.packages('pracma')

library(pracma)

# Compute the characteristic polynomial

E <- charpoly(x)

# Compute the expression

result <- 1 \* (x %\*% x %\*% x) - 27 \* (x %\*% x) - 77 \* x - 24 \* diag(3)

# Print results

print(E)

print(result)

OR

x<-matrix(c(1,2,3,5,7,11,13,17,19),nrow=3,ncol=3,byrow=TRUE)

install.packages('pracma')

library(pracma)

E<-charpoly(x)

1\*sq(sq(x))-27\*sq(x)-77\*x-24\*diag(nrow=c(3,3))

1. **T Test**

income = c(6.5, 10.5, 12.7, 13.8, 13.2, 11.4, 5.5, 8.0, 9.6, 9.1, 9.0, 8.5, 4.8, 7.3, 8.4, 8.7, 7.3, 7.4, 5.6, 6.8, 6.9, 6.8, 6.1, 6.5, 4.0, 6.4, 6.4, 8.0, 6.6, 6.2, 4.7, 7.4, 8.0, 8.3, 7.6, 6.7)

mu\_0 = 10

alpha = 0.05

# two-sided t-test

t\_test = t.test(income, mu = mu\_0)

t\_test$statistic

t\_test$p.value

if (t\_test$p.value < alpha/2) {

print("Reject the null hypothesis. The mean income is significantly different from Rs. 10,000")

} else {

print("Fail to reject the null hypothesis. There is not enough evidence to conclude a difference in the mean income")

}

# one-sided (left-tailed) t-test

t\_test = t.test(income, mu = mu\_0, alternative = "less")

t\_test$statistic

t\_test$p.value

if (t\_test$p.value < alpha) {

print("Reject the null hypothesis. The mean income is significantly less than Rs. 10,000")

} else {

print("Fail to reject the null hypothesis. There is not enough evidence to conclude the mean income is less than Rs. 10,000")

}

1. **Integrating**

integrand1=function(x)+ sqrt(x)/(sqrt(3-x)+sqrt(x))

result1=integrate(integrand1,lower=1,upper=2)

print(result1)

1. **Gram Schmidt and orthonormal basis for R^3**v1 <- c(1, 1, 0)

v2 <- c(1, 0, 1)

v3 <- c(0, 1, 1)

#To calculate orthogonal basis

b1 = v1

b2 = v2 -((sum(v2\*b1))\*b1)/(sum(b1^2))

b3 = v3 - (((sum(v3\*b1))\*b1)/sum(b1^2) - (((sum(v3\*b2))\*b2)/sum(b2^2)))

orth\_basis <- list(b1, b2, b3)

orth\_basis

nb1 = b1/(sqrt(sum(b1^2)))

nb2 = b2/(sqrt(sum(b2^2)))

nb3 = b3/(sqrt(sum(b3^2)))

orth\_norm\_basis <- list(nb1, nb2, nb3)

orth\_norm\_basis

1. **Generalized Inverse and conditions and property verified**

data <- c(3, 1, 5, 2, 1, 11, 7, 6, 3, 7, 4, 1, 6, 3, 7, 1)

A <- matrix(data, nrow = 4, ncol = 4, byrow = TRUE)

A

#To obtain Moore-Penrose Generalised Inverse

library(MASS)

G\_inv <- ginv(A)

G\_inv

#To Verify the condition for G-Inverse

result <- A %\*% G\_inv %\*% A

print("Result of A \* G\_inv \* A:")

print(result)

if (all.equal(result, A)) {

print("AGA is equal to A")

} else {

print("AGA is not equal to A")

}

if (all.equal(G\_inv %\*% A %\*% G\_inv, G\_inv))

{

print("GAG is equal to G")

} else {

print("GAG is not equal to G")

}

#Rank properties verified  
verify <- function(A, G)

{

prop\_1 = qr(A)$rank == qr(G)$rank

prop\_2 = all(round(A%\*%G, 0) == round(G%\*%A, 0) & round(A%\*%G, 0) == diag(4))

prop\_3 = all.equal(ginv(A%\*%G), A%\*%G)

return(list(Property\_1 = prop\_1, Property\_2 = prop\_2, Property\_3 = prop\_3))

}

verify(A, G\_inv)

1. **Transition probability(Calculate P^100)**X=c(.5,.5,0,0,0,.5,.5,0,0,0,0,0,.6667,.03333,0,0,0,.3333,.6667,0,.2,.2,.2000,.2000,.2)

P=matrix(X, nrow = 5, ncol = 5,byrow = TRUE)

P

P\_100 = P

for (i in 1:99)

{

P\_100 = P\_100%\*%P

}

P\_100

1. **Root polynomial using newton rhaphson**

newton\_raphson <- function(initial\_guess, tolerance=0.0001){

v1 = initial\_guess

iterations=0

repeat{

v2 = v1 - ((exp(-v1)-5\*v1-2)/(-exp(-v1)-5))

d=v2-v1

if(abs(d)<tolerance){

break

}

v1=v2

}

results = list(root=v2, iterations=iterations)

return(results)

}

newton\_raphson(initial\_guess=0)

1. **Lapply and Sapply loops and descriptive statistics**

x= runif(50)

y= rnorm(100,50,1)

z=rexp(50,0.1)

x

y

z

fun = c("sum", "mean", "range", "median", "min", "max", "log","log10")

x1=x2=c()

#Descriptive Stats for X:

for (f in fun)

{ x1 = sapply(x,FUN = f)

cat(f,x1,"\n") }

for (f in fun)

{ x2 = lapply(x,FUN = f)

print(f)

print(x2) }

#Descriptive stats for Y

y = unlist(y)

y\_new =c()

for (i in 1:length(y))

{ if(y[i]>0)

{

y\_new = c(y\_new,y[i])} }

y\_new =list(y\_new)

for (f in fun)

{ x1 = sapply(y\_new,FUN = f)

cat(f,x1,"\n")}

for (f in fun)

{ x2 = lapply(y\_new,FUN = f)

print(f)

print(x2)}

#Descriptive Stats for Z:

for (f in fun)

{ x1 = sapply(z,FUN = f)

cat(f,x1,"\n") }

for (f in fun)

{ x2 = lapply(z,FUN = f)

print(f)

print(x2) }

1. **MLE of gamma distribution and plot of log likelihood**

set.seed(69)

x = rgamma(100,shape = 3, scale = 2)

mle\_scale = mean(x)/3

print(mle\_scale)

beta = seq(from = 0.05, to=10, by = 0.1)

log\_likelihood = 2\*sum(log(x))-sum(x)/beta - 200\*log(beta)-100\*lgamma(3)

plot(beta, log\_likelihood,type = "l",main = "Log Likelihood Gamma ")

abline(v =mle\_scale, lty = 2)

1. **MLE of exponential distribution and plot of log likelihood**

set.seed(69)

y = rexp(100,rate = 2.5)

mle\_rate = 1/mean(y)

print(mle\_rate)

rate = seq(from = 0.05, to=10, by = 0.1)

log\_likelihood\_e = length(y)\*log(rate) - rate\*sum(y)

plot(rate, log\_likelihood\_e,type = "l", ,main = "Log Likelihood Exponential ")

abline(v =mle\_rate, lty = 2)

1. **Drawing random Numbers and plotting there pdf and cdf (exponential dist)**

# Set seed for reproducibility

set.seed(123)

# Parameters

n <- 60 # Size of the Binomial distribution

sample\_size <- 25 # Number of random samples

x <- 0:n # Possible values of the Binomial distribution

# Binomial with prob = 0.3

p1 <- 0.3

samples1 <- rbinom(sample\_size, n, p1)

cat("Random samples for B(60, 0.3):", samples1, "\n")

pmf1 <- dbinom(x, n, p1)

cdf1 <- pbinom(x, n, p1)

plot(x, pmf1, type = "h", main = "PMF of B(60, 0.3)", xlab = "x", ylab = "Probability", col = "blue")

plot(x, cdf1, type = "s", main = "CDF of B(60, 0.3)", xlab = "x", ylab = "Cumulative Probability", col = "red")

1. **Drawing random Numbers and plotting there pdf and cdf (poisson dist)**

#Drawing random numbers of size 25 from Pois(2), Pois(5) and Pois(10).

rpois(n=25,lambda=2)

rpois(n=25,lambda=5)

rpois(n=25,lambda=10)

#Plotting the pdf and cdf of Pois(2), Pois(5) and Pois(10).

x=seq(from=0,to=40)

par(mfrow=c(3,2))

#When lambda=2;

y1=dpois(x,2)

z1=ppois(x,2)

plot(x,y1,type="h",main="pmf of Pois(2)",xlab="x-values",ylab="P(X=x)-Probability Mass Function")

plot(x,z1,type="s",main="cdf of Pois(2)",xlab="x-values",ylab="F(X=x)-Distribution Function")

1. **Drawing random Numbers and plotting there pdf and cdf (gamma dist)**

#Drawing random numbers of size 25 from Gamma Distribution.

rgamma(25,shape=1,scale=2)

#Plotting the pdf and cdf of Gamma Distribution.

x=seq(from=0,to=50,length.out=1000)

par(mfrow=c(1,2))

y=dgamma(x,shape=1,scale=2)

z=pgamma(x,shape=1,scale=2)

plot(x,y,type="h",main="pdf of Gamma(1,2)",xlab="x-values",ylab="f(x)-probability density function")

plot(x,z,type="s",main="cdf of Gamma(1,2)",xlab="x-values",ylab="F(x)-Cumulative distribution function")

1. **Drawing random Numbers and plotting there pdf and cdf (Cauchy dist)**

#Drawing random numbers of size 25 from Standard cauchy distribution.

par(mfrow=c(1,2))

rcauchy(n=25,location=0,scale=1)

#Plotting the pdf and cdf of Standard cauchy distribution.

x=seq(from=0,to=80,length.out=500)

y=dcauchy(x,location=0,scale=1)

z=pcauchy(x,location=0,scale=1)

plot(x,y,type="h",main="pdf of standard cauchy distribution",ylab="f(x)-probability density function",xlab="x-values")

plot(x,z,type="s",main="cdf of standard cauchy distribution",ylab="F(x)-Cumulative distribution function",xlab="x-values")

1. **Drawing random Numbers and plotting there pdf and cdf (Weibull dist)**

#Drawing random numbers of size 25 from Weibull distribution.

rweibull(n=25,shape=2,scale=1)

#Plotting the pdf and cdf of Weibull distribution.

par(mfrow=c(1,2))

x=seq(from=0,to=20,length.out=500)

y=dweibull(x,shape=2,scale=1)

z=pweibull(x,shape=2,scale=1)

plot(x,y,type="h",main="pdf of Weibull distribution",ylab="f(x)-probability density function",xlab="x-values")

plot(x,z,type="s",main="cdf of Weibull distribution",ylab="F(x)-Cumulative distribution function",xlab="x-values")

1. **Density function and distribution curve of the standard normal distribution.**

x=seq(from=-6,to=6,length.out=1000)

y=dnorm(x,mean=0,sd=1)

par(mfrow=c(1,2))

#Plotting the density function of of the standard normal distribution.

plot(x,y,main="Standard Normal Distribution",ylab="f(x)-density function")

points(seq(from=-6,to=6),c(dnorm(-6),dnorm(-5),dnorm(-4),dnorm(-3),dnorm(-2),dnorm(-1),dnorm(0),dnorm(1),dnorm(2),dnorm(3),dnorm(4),dnorm(5),dnorm(6)),col='red',pch=16)

z=pnorm(x,mean=0,sd=1)

plot(x,z,pch=15,type="b",main="cdf N(0,1)",ylab="F(x)- Cumulative Distribution Function")

1. **Working with data frame.**

#Creating a data frame.

SI\_No=seq(from=1,to=20)

Sales\_Kg=c(200,100,100,400,500,800,150,200,120,400,500,185,200,150,240,500,250,170,350,180)

PricePerKg=c(40,80,40,60,70,90,40,90,30,30,40,60,100,65,45,60,65,35,85,60)

Color\_flag=c(2,1,1,2,2,1,2,2,2,1,2,1,2,2,1,1,1,2,2,1)

Veg\_df=data.frame(SI\_No,Sales\_Kg,PricePerKg,Color\_flag)

Veg\_df

#1.Adding rownames for the dataframe.

rownames(Veg\_df)=c('Eggplant','Pea','Cucumber','Potato','Pumpkin','Lettuce','Tomato','Sweet potato','Mushroom','Green bean','Corn','Cauliflower','Beetroot','Bell pepper','Broccoli','Celery','Cabbage','Carrot','Onion','Lady finger')

Veg\_df

#2.Calculating the revenue for each vegetable and its proportion in the total revenue.

Veg\_df$Revenue=Veg\_df$Sales\_Kg\*Veg\_df$PricePerKg

Veg\_df

#3. Adding the labels in the color flags.

Veg\_df$Color\_flag=ifelse(Veg\_df$Color\_flag==1,'green','other.color')

Veg\_df

#4.Calculating the difference in revenues.

green\_rev=sum(Veg\_df$Revenue[Veg\_df$Color\_flag=='green'])

green\_rev

OthCol\_rev=sum(Veg\_df$Revenue[Veg\_df$Color\_flag=='other.color'])

OthCol\_rev

rev\_diff=green\_rev-OthCol\_rev

rev\_diff

#5.Replacing the label.

Veg\_df$Color\_flag[Veg\_df$Color\_flag=='other.color']='not.green'

Veg\_df

#6.Extracting the 'not.green' rows.

Veg\_df[Veg\_df$Color\_flag=='not.green',]