1. Program to handle vectors and perform simple statistics on the vector using R

a. Declaration of vector
b. Generating useful vectors using :,seq,rep
c. Applying any ()and all() functions on vectors
d. Vector Indexing
e. Vector Arithmetic
f. Statistical functions on the vectors
g. Plotting of Vector
h. Find cumulative sum of a vector
i. Row and column summary commands
j. ap
ply() command enables applying a function to the rows or columns of amatrix or data frame

Answer:

```
#a. Declaration of vector
x <- c(2,4,6,8)

#b. Generating useful vectors using :,seq,rep
5:15
seq(4,20)
seq(12,30,3)
seq(1.1,2,length=10)
rep(8,4)
rep(1:3,2)

#c. Applying any ()and all() functions on vectors
x <-1:10
if (any(x>8)) print ("yes")
if (any(x>88)) print("yes")
if (all(x>0)) print("yes")

#d. Vector Indexing
k <-c(1.2,3.4,0.4,0.12)
k[c(1,3)]

#e. Vector Arithmetic
x <- c(2,4,6,8)
y <-c(3,5,7,9)
x+y

#f. Statistical functions on the vectors
mean(x)
sd(x)

#g. Plotting of Vector
k
hist(k)
plot(k)
```

```
#h. Find cumulative sum of a vector
vec = c(3,5,7,5,3,2,6) #Creating vector
cumsum(vec)

#Row and column summary commands
quiz <- data.frame("q1" = c(0, 0, 0, 0, 1),
                   "q2" = c(0, 1, 1, 0, 1),
                   "q3" = c(0, 0, 0, 1, 1),
                   "q4" = c(1, 1, 1, 1, 1),
                   "q5" = c(1, 0, 1, 0, 1))
quiz
rowMeans(quiz)
rowSums(quiz)
colMeans(quiz)
colSums(quiz)

#apply() command enables applying a function to the
#rows or columns of a matrix or data frame
apply(quiz, 1, mean, na.rm = TRUE)
```

2: To create a data frame in R and perform operations on it.

a. Create a variable df data frame containing three variables vec, char_vecand bool_vec
b. Perform various operations on employee data frame
c. Get the Structure of the Data Frame
d. Extract data from Data Frame
e. Extract first two rows
f. Extract 1st and 2nd row with the 3rd and 4th column
g. Add the "dept" column
h. Create the second R data frame and Bind the two data frames
i. Deleting Component
j. Manipulate data frame: set value, order, reverse order, grouping
k. Boxplots

Answer:

```
#a. Create a variable df data frame containing
#three variables vec, char_vec and bool_vec
int_vec <- c(1,2,3)
char_vec <- c("a", "b", "c")
bool_vec <- c(TRUE, TRUE, FALSE)
data_frame <- data.frame(int_vec, char_vec, bool_vec)
data_frame

#b. Perform various operations on employee data frame
employee_data <- data.frame(employee_id = c (1:5),
                  employee_name = c("James","Harry","Shinji","Jim","Oliver"),
                  sal = c(642.3,535.2,681.0,739.0,925.26),
                  join_date = as.Date(c("2013-02-04", "2017-06-21", "2012-11-14", "2018-05-
19","2016-03-25")),
                  stringsAsFactors = FALSE)
employee_data
```

```
#Get the Structure of the Data Frame
str(employee_data)

#Extract data from Data Frame
output <- data.frame(employee_data$employee_name,employee_data$employee_id)
print(output)

#Extract first two rows
output <- employee_data[1:2,]
print(output)

#Extract 1st and 2nd row with the 3rd and 4th column
result <- employee_data[c(1,2),c(3,4)]
result

#Add the "dept" column
employee_data$dept <- c("IT","Finance","Operations","HR","Administration")
out <- employee_data
print(out)

#Create the second R data frame and Bind the two data frames
employee_new_data <- data.frame(employee_id = c (6:8),
                    employee_name = c("Aman", "Piyush", "Aakash"),
                    sal = c(523.0,721.3,622.8),
                    join_date = as.Date(c("2015-06-22","2016-04-30","2011-03-17")),
                    stringsAsFactors = FALSE)
employee_data
employee_new_data
employee_out_data <- rbind(employee_data,employee_new_data)
employee_out_data

#Deleting Component
x <- data.frame("SN" = 1:2, "Age" = c(21,15), "Name" = c("John","Dora"))
x
str(x)
x$SN <- NULL
x

#Manipulate data frame
id <- c(1,2,3)
name <- c("John", "Kirk", "AJ")
age <- c(21,27,18)
employees <- data.frame(ID=id, Name=name, Age=age)
employees

#set value
employees[3,"Age"] <- 29
employees

#order
```

```
employees[order(employees$Age),]

#reverse order
employees[order(employees$Age, decreasing=T),]

#grouping -- aggregate is similar to group by in SQL. Here are the # employees grouped by age
aggregate(employees[,2], list(Age=employees$Age), FUN=length)
#e. Boxplots
x = c(4,3,4,5,2,3,4,5)
y = c(4,4,5,5,4,5,4,4)
z = c(3,4,2,4,5,5,4,4)
scores = data.frame(x,y,z)
boxplot(scores)
```

3. (a) Create two datasets as follows:

Dataset 1

Make	Num models
Honda	63
BMW	10

Dataset 2

Make	Num models
Ford	26
Tesla	4

Answer:

```
#Create two datasets as follows:
dataset1 = data.frame(Make = c("Honda","BMW"),Num_models = c(63,10))
dataset2 = data.frame(Make = c("Ford","Tesla"),Num_models = c(26,4))
reordered_dataset1 <- rbind(dataset1, dataset2)
reordered_dataset2<- cbind(dataset1, dataset2)
reordered_dataset1
reordered_dataset2
```

(b) Log Transformation: In this experiment, the effect of vitamin supplements on weight gain is being investigated in three animal species (mice, chickens, and sheep). The experiment is designed as an RCBD with one replication (i.e. animal) per block*treatment combination. The six treatment levels are MC (mouse control), MV (mouse + vitamin), CC (chicken control), CV (chicken + vitamin), SC (sheep control), and SV (sheep + vitamin). The response variable is the weight of the animal at the end of the experiment. Create ANOVA model of the log-transformed data

trtmt	block	weight
MC	I	0.18
MC	II	0.3
MC	III	0.28
...	...	...
SV	II	153
SV	III	148
SV	IV	176

Answer:

```
vit_mod = data.frame(trtmt =
c("MC","MV","CC","CV","SC","SV","MC","MV","CC","CV","SC","SV","MC","MV","CC","CV","SC","SV",
"MC","MV","CC","CV","SC","SV"),
            block =
c("I","I","I","I","I","I","II","II","II","II","II","II","III","III","III","III","III","III","IV","IV","IV","IV","IV","I
V"),
            weight =
c(0.18,0.32,2.00,2.50,108.00,127.00,0.30,0.40,3.00,3.30,140.00,153.00,0.28,0.42,1.80,2.50,135.
00,148.00,0.44,0.46,2.80,3.30,165.00,176.00))
head(vit_mod,3)
str(vit_mod)
vit_mod$block<-as.factor(vit_mod$block)
vit_mod$trtmt<-as.factor(vit_mod$trtmt)
str(vit_mod)
vit_lm<-lm(weight ~ trtmt + block, vit_mod)
anova(vit_lm)
```

4 (a) Find the factorial of a number using recursion in R
(b) Print numbers from 1 to 100 using while loop and for loop in R
(c) Convert Decimal into Binary using Recursion in R
d. Program to display the Fibonacci sequence up to n-th term usingrecursive functions
e. Finding the sum of natural numbers using the recursive function
f. Finding sum of series $1^2+2^2+3^2+.....+n^2$ using the recursive function.

Answer:

```
#a. Recursive function to find factorial

recursive.factorial <- function(x) {
  if (x == 0) return (1)
  else return (x * recursive.factorial(x-1))
}
recursive.factorial(0)
recursive.factorial(5)
recursive.factorial(7)

#b. Print numbers from 1 to 100 using while and for loop

#Using for loop
for (i in 1:10)
{
  print(i)
}

#Using while loop
i<-0
while (i <=10)
{
  print(i)
  i=i+1
}
```

```
#c. Decimal to Binary
binary <- function(deci) {
 if(deci > 1) {
  binary(as.integer(deci/2))
 }
 cat(deci %% 2)
}
binary(13)

#d. Fibonacci upto nth term
fib_n <- function(n) {
 if (n<=1) {
  return (n)
 } else {
  temp = fib_n(n-1) + fib_n(n-2)
  return (temp)
 }
}
i <- 0
n <- 10
while(i < n) {
 print(fib_n(i))
 i <- i+1
}

#e. Sum of n natural numbers
sum_n <- function(n) {
 if (n<=0) {
  return (0)
 } else {
  return (n + sum_n(n-1))
 }
}
sum_n(10)

#f. Sum of n^2 numbers
sum_sq <- function(n) {
 if (n<=0) {
  return (0)
 } else {
  return (n*n + sum_sq(n-1))
 }
}
sum_sq(5)
```

5. Recursion execution for Merge Sort.

Answer:

```
#5. Merge Sort using recursion
merge <- function(a,b) {
 r <- length(a)+length(b)
```

```
 ai <- 1;
 bi <- 1;
 j <- 1;
 for(j in 1:r) {
  if((ai <= length(a) && a[ai] < b[bi]) || bi > length(b)) {
   r[j] <- a[ai]
   ai <- ai+1
  } else {
   r[j] <- b[bi]
   bi <- bi+1
  }
 }
 return (r)
}
mergesort<-function(A) {
 if(length(A)>1) {
  q <- ceiling(length(A)/2)
  a <- mergesort(A[1:q])
  b <- mergesort(A[(q+1):length(A)])
  merge(a,b)
 } else {
  return (A)
 }
}
mergesort(c(1,2,7,5,4))
```

6. (a) Plot graphs such as scatter plot, box plot and bar plot.
b. Use the built-in dataset airquality which has "Daily air quality measurements in New York, May to September 1973."
c. consider the Ozone and Temp field of airquality dataset. Let us also generate normal distribution with the same mean and standard deviation and plot them side by side for comparison.
d. The function boxplot() can also take in formulas of the form y~x where, y is a numeric vector which is grouped according to the value of x. For example, in our dataset airquality, the Temp can be our numeric vector. Month can be our grouping variable, so that we get the boxplot for each month separately. In our dataset, month is in the form of number (1=January, 2-Febuary and so on).
e. Plot the count of each item as bar plots from categorical data. For example, here is a vector of age of 10 college freshmen
age <- c(17,18,18,17,18,19,18,16,18,18)
Plot 10 bars with height equal to the student's age, i.e number of student in each age category

Answer:

```
#a,b
str(airquality)
boxplot(airquality$Ozone)
boxplot(airquality$Ozone,
      main = "Mean ozone in parts per billion at Roosevelt Island",
      xlab = "Parts Per Billion",
      ylab = "Ozone",
      col = "orange",
```

```
    border = "brown",
    horizontal = TRUE,
    notch = TRUE
)
plot(airquality$Ozone)
#c.
ozone <- airquality$Ozone
temp <- airquality$Temp
# gererate normal distribution with same mean and sd
ozone_norm <- rnorm(200,mean=mean(ozone, na.rm=TRUE), sd=sd(ozone,
                                        na.rm=TRUE))
temp_norm <- rnorm(200,mean=mean(temp, na.rm=TRUE), sd=sd(temp,
                                       na.rm=TRUE))
boxplot(ozone, ozone_norm, temp, temp_norm,
    main = "Multiple boxplots for comparision",
    at = c(1,2,4,5),
    names = c("ozone", "normal", "temp", "normal"),
    las = 2,
    col = c("orange","red"),
    border = "brown",
    horizontal = TRUE,
    notch = TRUE
)
#d.
boxplot(Temp~Month,
    data=airquality,
    main="Different boxplots for each month",
    xlab="Month Number",
    ylab="Degree Fahrenheit",
    col="orange",
    border="brown"
)
#e.
age <- c(17,18,18,17,18,19,18,16,18,18)
barplot(table(age),
    main="Age Count of 10 Students",
    xlab="Age",
    ylab="Count",
    border="red",
    col="blue",
    density=10
)
```

7.a. Create Restaurant dataset as follows:

```
##      sex   time total_bill
## 1 Female  Lunch      13.53
## 2 Female Dinner      16.81
## 3   Male  Lunch      16.24
## 4   Male Dinner      17.42
```

Make Barplot with multiple groups

1. Filter and count the number of records by groups and create the grouped bar plots

Answer:

```
s = c('Female','Female','Male','Male')
t = c('Lunch','Dinner','Lunch','Dinner')
b = c(13.53, 16.81, 16.24, 17.42)
restaurant <- data.frame(sex = s, time = t, total_bill = b)
df1 <- restaurant
head(df1)

#Make Barplot with multiple groups

#install.packages('ggplot2')
library('ggplot2')
ggplot(data=df1, aes(x=time, y=total_bill,fill=sex),) +
  geom_bar(stat="identity",width=0.5)

# Use position=position_dodge()
ggplot(data=df1, aes(x=time, y=total_bill,fill=sex),) +
  geom_bar(stat="identity",width=0.5, position=position_dodge())
```

2. Create a vector of maximum temperatures (in degree Celsius) for seven days and make a bar plot out of this data. Give the title, xlab and ylab to provide labels for the axes, names.arg for naming each bar, col to define color etc

Answer:

```
temps = c(24,18,36,44,27,26,32)
barplot(temps,
      main = "Maximum Temperatures",
      xlab = "Days",
      ylab = "'Temperatures",
      names.arg = c('Sunday','Monday','Tuesday','Wednesday','Thursday',
                'Friday','Saturday'))
```

3. Use the iris dataset and make box plot, Scatterplot with Trend Line, Classed-up Scatterplot using ggplot2, Add the labels, Interaction plot

Answer:

```
head(iris,n=5)
boxplot(iris$Petal.Width,
      main = 'Petal Width of Iris')
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length,
            group=Species, fill=Species)) +
  geom_boxplot()

ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point()

l = iris$Petal.Length
w = iris$Petal.Width
plot(l,w,
    xlab="Petal Length",
    ylab="Petal Width",
    main = "Iris Petals")
```

```
abline(lm(w~l))

#install.packages("car")
library(car)

scatterplotMatrix(~ Petal.Length+Petal.Width, data = iris)
```

4. Make Box-and-Whisker Plots for a sample vector of data

Answer:

```
temps = c(24,18,36,44,27,26,32)
days = c('Sunday','Monday','Tuesday','Wednesday','Thursday',
      'Friday','Saturday')
tempdf <- data.frame(temps=temps,days=days)
ggplot(tempdf, aes(x=days, y=temps)) +
 geom_boxplot()
```

Create a list in R and perform operations on it like list Slicing, sum and mean functions, head and tail functions and finally delete the list using rm() function

8.a. Non- linear regression: Illustrate the difference between linear and nonlinear models, fit them both:

Answer:

```
set.seed(23)
#Generate x as 100 integers using seq function
x<-seq(0,100,1)
#Generate y as a*e^(bx)+c
y<-runif(1,0,20)*exp(runif(1,0.005,0.075)*x)+runif(101,0,5)
#How does our data look like? Lets plot it
plot(x,y)
#Linear model
lin_mod=lm(y~x)
#Plotting the model
plot(x,y)
abline(lin_mod)
nonlin_mod=nls(y~a*exp(b*x),start=list(a=13,b=0.1))
#a is the starting value and b is the exponential start
#This new plot can be made by using the lines() function
plot(x,y)
lines(x,predict(nonlin_mod),col="red")
```

b. Understanding Self-Starting Functions

Answer:

```
#b. Understanding Self-Starting Functions
#(i)
attach(Puromycin)
```

```
plot(Puromycin$conc,Puromycin$rate)
#(ii)
library(deSolve)
#simulating some population growth from the logistic equation and estimating
#the parameters using nls
log_growth <- function(Time, State, Pars) {
  with(as.list(c(State, Pars)), {
    dN <- R*N*(1-N/K)
    return(list(c(dN)))
  })
}
#the parameters for the logisitc growth
pars <- c(R=0.2,K=1000)
#the initial numbers
N_ini <- c(N=1)
#the time step to evaluate the ODE
times <- seq(0, 50, by = 1)
#the ODE
out <- ode(N_ini, times, log_growth, pars)
#add some random variation to it
N_obs<-out[,2]+rnorm(51,0,50)
#numbers cannot go lower than 1
N_obs<-ifelse(N_obs<1,1,N_obs)
#plot
plot(times,N_obs)
```

c. Simulate some data, this without a priori knowledge of the parameter value

Answer:

```
y<-runif(1,5,15)*exp(-runif(1,0.01,0.05)*x)+rnorm(51,0,0.5)
#visually estimate some starting parameter values
plot(x,y)
#from this graph set approximate starting values
a_start<-8 #param a is the y value when x=0
b_start<-2*log(2)/a_start #b is the decay rate
#model
m<-nls(y~a*exp(-b*x),start=list(a=a_start,b=b_start))
#get some estimation of goodness of fit
cor(y,predict(m))
#plot the fit
lines(x,predict(m),col="red",lty=2,lwd=3)
```

d. Nonlinear regression: This example is based on the relationship between jaw bone length and age in deers.

Answer:

```
#Download the jaws.txt from the following link:
#           https://drive.google.com/file/d/1RmDs1hWcgYGD0GxbXliLAOoDT50p-
cL0/view
deer<-read.table("C:\\Users\\Admin\\Documents\\R programs\\R
Lab\\jaws.txt",header=T)
head(deer)
#C:\Users\Admin\Documents\R programs\R Lab
```

```
#Fitting the model - Nonlinear equation is an argument in nls() command with
#starting values of a, b and c parameters. The result goes in the model object.
model<-nls(bone~a-b*exp(-c*age), data=deer,
        start=list(a=120,b=110,c=0.064))

#Displaying information about a model object using the summary() command. The
#model object is an argument to the summary() command as shown below:
summary(model)

#Applying nls() command to the new model for the modified regression model. The
#result goes in the model2 object.
model2<-nls(bone~a*(1-exp(-c*age)),data=deer,start=list(a=120,c=0.064))

#Comparing the models as below - Use anova() command to compare result
#objectsmodel1 and model2. These objects then act as arguments to anova()
#command.
anova(model,model2)

#Viewing the components of the New Model2 as below:
summary(model2)
```

e. Multivariate Adaptive Regression Splines (MARS) is a non-parametric regression method that models multiple nonlinearities in data using hinge functions

Answer:

```
library(earth)
# load data
data(longley)
# fit model
fit <- earth(Employed~., longley)
# summarize the fit
summary(fit)
# summarize the importance of input variables
evimp(fit)
# make predictions
predictions <- predict(fit, longley)
# summarize accuracy
mse <- mean((longley$Employed - predictions)^2)
print(mse)
```

9. Implement linear regression on cars dataset and use Scatter Plot To Visualise The Relationship

Answer:

```
library(e1071) # for skewness function
par(mfrow=c(1, 2)) # divide graph area in 2 columns
plot(density(cars$speed), main="Density Plot: Speed", ylab="Frequency",
   sub=paste("Skewness:", round(e1071::skewness(cars$speed), 2))) # density plot
#for 'speed'
polygon(density(cars$speed), col="red")
plot(density(cars$dist), main="Density Plot: Distance", ylab="Frequency",
   sub=paste("Skewness:", round(e1071::skewness(cars$dist), 2))) # density plot for
```

```
#'dist'
polygon(density(cars$dist), col="red")
cor(cars$speed, cars$dist)
linearMod <- lm(dist ~ speed, data=cars) # build linear regression model on full data
print(linearMod)
summary(linearMod)
modelSummary <- summary(linearMod)
modelCoeffs <- modelSummary$coefficients
beta.estimate <- modelCoeffs["speed", "Estimate"]
std.error <- modelCoeffs["speed", "Std. Error"]
t_value <- beta.estimate/std.error
p_value <- 2*pt(-abs(t_value), df=nrow(cars)-ncol(cars))
f_statistic <- linearMod$fstatistic[1]

# parameters for model p-value calc
f <- summary(linearMod)$fstatistic
model_p <- pf(f[1], f[2], f[3], lower=FALSE)
t_value
p_value
f_statistic
model_p
```

10. Create a dataset house.csv as follows and perform encoding categorical data.

	R.D.Spend	Administration	Marketing.Spend	State	Profit
1	165349.2	136897.80	471784.1	New York	192261.8
2	162597.7	151377.59	443898.5	California	191792.1
3	153441.5	101145.55	407934.5	Florida	191050.4
4	144372.4	118671.85	383199.6	New York	182902.0
5	142107.3	91391.77	366168.4	Florida	166187.9
6	131876.9	99814.71	362861.4	New York	156991.1
7	134615.5	147198.87	127716.8	California	156122.5
8	130298.1	145530.06	323876.7	Florida	155752.6
9	120542.5	148718.95	311613.3	New York	152211.8
10	123334.9	108679.17	304981.6	California	149760.0

Answer:

```
dataset = read.csv('C:\\Users\\Admin\\Documents\\R programs\\R Lab\\house.csv')
dataset$State = factor(dataset$State,levels = c('New York', 'California', 'Florida'), labels = c(1,
2, 3))
dataset$State
```

11. Implement logistic regression using the BreastCancer dataset in mlbench package.

Answer:

```
install.packages("mlbench")
data(BreastCancer, package="mlbench")
bc <- BreastCancer[complete.cases(BreastCancer), ] # create copy
str(bc)
glm(Class ~ Cell.shape, family="binomial", data = bc)
# remove id column
bc <- bc[,-1]
# convert factors to numeric
for(i in 1:9) {
```

```
  bc[, i] <- as.numeric(as.character(bc[, i]))
}
bc$Class <- ifelse(bc$Class == "malignant", 1, 0)
bc$Class <- factor(bc$Class, levels = c(0, 1))
table(bc$Class)

library(caret)
'%ni%' <- Negate('%in%') # define 'not in' func
options(scipen=999) # prevents printing scientific notations.
# Prep Training and Test data.
set.seed(100)
trainDataIndex <- createDataPartition(bc$Class, p=0.7, list = F) # 70%
#training data
trainData <- bc[trainDataIndex, ]
testData <- bc[-trainDataIndex, ]
table(trainData$Class)

# Down Sample
set.seed(100)
down_train <- downSample(x = trainData[, colnames(trainData) %ni%
                                 "Class"], y = trainData$Class)
table(down_train$Class)
# Up Sample.
set.seed(100)
up_train <- upSample(x = trainData[, colnames(trainData) %ni% "Class"],
                     y = trainData$Class)
table(up_train$Class)
# Build Logistic Model
logitmod <- glm(Class ~ Cl.thickness + Cell.size + Cell.shape, family =
                  "binomial", data=down_train)
summary(logitmod)
pred <- predict(logitmod, newdata = testData, type = "response")
y_pred_num <- ifelse(pred > 0.5, 1, 0)
y_pred <- factor(y_pred_num, levels=c(0, 1))
y_act <- testData$Class
mean(y_pred == y_act)
```

12. Download Smarket data and perform visualization of regression analysis

Answer:

```
require(ISLR)
names(Smarket)
head(Smarket)
summary(Smarket)
#visualize the data
par(mfrow=c(1,8))
for(i in 1:8) {
  hist(Smarket[,i], main=names(Smarket)[i])
}

par(mfrow=c(1,8))
for(i in 1:8) {
  boxplot(Smarket[,i], main=names(Smarket)[i])
```

```
}

library(Amelia)
library(mlbench)
missmap(Smarket, col=c("blue", "red"), legend=FALSE)

library(corrplot)
correlations <- cor(Smarket[,1:8])
corrplot(correlations, method="circle")
pairs(Smarket, col=Smarket$Direction)

library(caret)
x <- Smarket[,1:8]
y <- Smarket[,9]
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=x, y=y, plot="density", scales=scales)
```

13. a. Compute the ANOVA test on a sample dataset
b. Create a dataset with 40 records and perform two way ANOVA testInput = (" id Sex Genotype Activity

```
1 male ff 1.884
2 male ff 2.283
3 male fs 2.396
4 female ff 2.838
5 male fs 2.956
6 female ff 4.216
7 female ss 3.620
8 female ff 2.889
9 female fs 3.550
10 male fs 3.105
11 female fs 4.556
.....
40 female fs 3.087)
```

Answer:

```
Input = ("
id Sex Genotype Activity
 1 male ff 1.884
 2 male ff 2.283
 3 male fs 2.396
 4 female ff 2.838
 5 male fs 2.956
```

```
 6 female ff 4.216
 7 female ss 3.620
 8 female ff 2.889
 9 female fs 3.550
 10 male fs 3.105
 11 female fs 4.556
 12 female fs 3.087
 13 male ff 4.939
 14 male ff 3.486
 15 female ss 3.079
 16 male fs 2.649
 17 female fs 1.943
 19 female ff 4.198
 20 female ff 2.473
 22 female ff 2.033
 24 female fs 2.200
 25 female fs 2.157
 26 male ss 2.801
 28 male ss 3.421
 29 female ff 1.811
 30 female fs 4.281
 32 female fs 4.772
 34 female ss 3.586
 36 female ff 3.944
 38 female ss 2.669
 39 female ss 3.050
 41 male ss 4.275
 43 female ss 2.963
 46 female ss 3.236
 48 female ss 3.673
 49 male ss 3.110
 ")
Data = read.table(textConnection(Input),header=TRUE)
#install.packages("Rmisc")
library(Rmisc)
sum = summarySE(Data,
            measurevar="Activity",
            groupvars=c("Sex","Genotype"))
sum
#Simple box plot of main effect and interaction
boxplot(Activity ~ Genotype,
      data = Data,
      xlab = "Genotype",
      ylab = "MPI Activity")
boxplot(Activity ~ Genotype:Sex,
      data = Data,
      xlab = "Genotype x Sex",
      ylab = "MPI Activity")
#Fit the linear model and conduct ANOVA
model = lm(Activity ~ Sex + Genotype + Sex:Genotype,
        data=Data)
library(car)
Anova(model, type="II") # Can use type="III"
### If you use type="III", you need the following line before the analysis
### options(contrasts = c("contr.sum", "contr.poly"))
anova(model) # Produces type I sum of squares
```

```
summary(model) # Produces r-square, overall p-value, parameter estimates
#Checking assumptions of the model
hist(residuals(model),
   col="darkgray")
#A histogram of residuals from a linear model. The distribution of these
#residuals should be approximately normal.
plot(fitted(model),
   residuals(model))
### additional model checking plots with:
plot(model)
```

14. Download wine dataset from https://archive.ics.uci.edu/ml/index.php and perform PCA on it

```
wine <- read.table("C:/Users/Admin/Documents/R programs/R Lab/wine.data", sep=",")

# Name the variables
colnames(wine) <- c("Cvs","Alcohol","Malic acid","Ash","Alcalinity of ash",
            "Magnesium", "Total phenols", "Flavanoids", "Nonflavanoid phenols",
            "Proanthocyanins", "Color intensity", "Hue", "OD280/OD315 of diluted wines",
"Proline")

# The first column corresponds to the classes
wineClasses <- factor(wine$Cvs)

# Use pairs
pairs(wine[,-1], col = wineClasses, upper.panel = NULL, pch = 16, cex = 0.5)
legend("topright", bty = "n", legend = c("Cv1","Cv2","Cv3"), pch = 16,
     col = c("black","red","green"),xpd = T, cex = 1.2, y.intersp = 0.9)
# pairwise interactions in a set of 13 variables,
dev.off()
# clear the format from the previous plot
winePCA <- prcomp(scale(wine[,-1]))
plot(winePCA$x[,1:2], col = wineClasses)

# repeat the procedure after introducing an outlier in place of the 10th observation.
wineOutlier <- wine
wineOutlier[10,] <- wineOutlier[10,]*10
# change the 10th obs. into an extreme one by multiplying its profile by 10
outlierPCA <- prcomp(scale(wineOutlier[,-1]))
plot(outlierPCA$x[,1:2], col = wineClasses)

#if (!require("BiocManager", quietly = TRUE))
 # install.packages("BiocManager")
#BiocManager::install()
#BiocManager::install("pcaMethods")
library(pcaMethods)
winePCAmethods <- pca(wine[,-1], scale = "uv", center = T, nPcs = 2, method = "svd")
slplot(winePCAmethods, scoresLoadings = c(T,T), scol = wineClasses)
```

1. Perform the following tasks on tumor dataset

(i) Standardize the data (Center and scale).

(ii) Calculate the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix.

(iii) Sort the Eigenvalues in descending order and choose the K largest Eigenvectors (Where K is the desired number of dimensions of the new feature subspace k ≤ d).

Answer:

```
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data"
data <- read.csv(url)
head(data)

#Standardize data
data <- read.csv(file = url, header = FALSE,
          col.names = c("id","CT", "UCSize", "UCShape", "MA", "SECS", "BN", "BC", "NN","M", "diagnosis") )

data$outcome[data$diagnosis==4] = 1
data$outcome[data$diagnosis==2] = 0
data$outcome = as.integer(data$outcome)
head(data)

library(dplyr)
library(tidyverse)

data2 <- data %>% select(-id, -BN)
data2$outcome[data2$diagnosis==4] = 1
data2$outcome[data2$diagnosis==2] = 0
data2$outcome = as.integer(data2$outcome)
head(data2)
glimpse(data2)
#install.packages('PerformanceAnalytics')
library(PerformanceAnalytics)
chart.Correlation(data2[, c(1:7)], histogram=TRUE, col="grey10", pch=1, main="Cancer Means")

corr <- cor(data2,method='pearson')

ecor <- eigen(corr)

ord1 <- order(ecor$vectors,decreasing = TRUE)
ord2 <- order(ecor$values,decreasing = TRUE)

head(ord1,5)
head(ord2,5)
```

15. Download Titanic dataset. The purpose of this dataset is to predict which people are more likely to survive after the collision with the iceberg. The dataset contains 13 variables and 1309 observations. Construct decision tree for this dataset

Answer:

```
set.seed(678)
path <- 'https://raw.githubusercontent.com/guru99-edu/R-Programming/master/titanic_data.csv'
```

```
titanic <-read.csv(path)
head(titanic)
shuffle_index <- sample(1:nrow(titanic))
head(shuffle_index)
titanic <- titanic[shuffle_index, ]
head(titanic)

library(dplyr)
# Drop variables
clean_titanic <- titanic %>%
select(-c(home.dest, cabin, name, X, ticket)) %>% mutate(pclass = factor(titanic$pclass, levels =
c(1, 2, 3), labels = c('Upper', 'Middle', 'Lower')),
       survived = factor(titanic$survived, levels = c(0, 1), labels = c('No', 'Yes'))) %>% na.omit()
glimpse(clean_titanic)

create_train_test <- function(data, size = 0.8, train = TRUE) {
 n_row = nrow(data)
 total_row = size * n_row
 train_sample = c(1: total_row)
 if (train == TRUE) {
  return (data[train_sample, ])
 } else {
  return (data[-train_sample, ])
 }
}

data_train <- create_train_test(clean_titanic, 0.8, train = TRUE)
data_test <- create_train_test(clean_titanic, 0.8, train = FALSE)
create_train_test(df, size = 0.8, train = TRUE)

dim(data_train)

dim(data_test)

prop.table(table(data_train$survived))

prop.table(table(data_test$survived))

library(rpart)
library(rpart.plot)
fit <- rpart(survived~pclass, data=data_train,
         method = "class")
rpart.plot(fit,nn=TRUE)
```

16. Implement multiple Random Forest models with different hyper parameters, and compare one of the Random Forest model with Decision Tree model. Download the dataset from UCI website https://archive.ics.uci.edu/ml/machine-learning-databases/car/ The data contains 7 variables – six explanatory (Buying Price, Maintenance, NumDoors, NumPersons, BootSpace, Safety) and one response variable (Condition). The variables are self-explanatory and refer to the attributes of cars and the response

variable is 'Car Acceptability'. All the variables are categorical in nature and have 3-4 factor levels in each.

Answer:

```
library(randomForest)
data1 <- read.csv(file.choose(), header = TRUE)
names(data1) <- c("BuyingPrice","Maintenance","NumDoors","NumPersons","BootSpace","Safety","Condition")
head(data1)
str(data1)
summary(data1)
set.seed(100)
train <- sample(nrow(data1), 0.7*nrow(data1), replace = FALSE)
TrainSet <- data1[train,]
ValidSet <- data1[-train,]
summary(TrainSet)
summary(ValidSet)
head(TrainSet)
Condition <- c("acc","vgood","unacc","good")
model1 <- randomForest(as.factor(Condition) ~ ., data = TrainSet, importance = TRUE)
model1

# Predicting on train set

predTrain <- predict(model1, TrainSet, type = 'class')

# Checking classification accuracy
table(predTrain, TrainSet$Condition)
table(predTrain, TrainSet$Condition)
# Predicting on Validation set
predValid <- predict(model1, ValidSet, type = 'class')

# Checking classification accuracy
mean(predValid == ValidSet$Condition)
table(predValid,ValidSet$Condition)
mean(predValid == ValidSet$Condition)
table(predValid,ValidSet$Condition)
importance(model1)
varImpPlot(model1)
```