NAME : Avinash Gautam
COLLEGE ROLL NO. : 20201407
EXAM ROLL NO. : 20020570009
COURSE : B.Sc. (H) Computer Science
SEMISTER : 4th

R Programming

Practical File

Question 1 : Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset.

Solution :

Code :

```
data("iris")          # getting iris dataset
str(iris)             # iris dataset info

# subset with Sepal.Length > 5 and Sepal.Width > 4
subset(iris,Sepal.Length > 5.0 & Sepal.Width > 4)
# subset with Petal.Length>5 and Petal.Width>3.5
subset(iris,Petal.Length > 1.0& Petal.Width==0.4)
# subset with Species of “setosa” and Sepal’s Length > 5.4
subset(iris,Species=="setosa" & Sepal.Length > 5.4)

iris_num <- iris[-5]          # getting only numeric type column
str(iris_num)                 # updated dataset info

# Sum by species type
aggregate(iris_num,by=list(iris$Species),FUN = sum)
# Mean by species type
aggregate(iris_num,by=list(iris$Species),FUN = mean)
# Standard Deviation by species type
aggregate(iris_num,by=list(iris$Species),FUN = sd)
# Maximum by species type
aggregate(iris_num,by=list(iris$Species),FUN = max)
# Minimum by species type
aggregate(iris_num,by=list(iris$Species),FUN = min)
```

Output :

```
> data("iris")              # getting iris dataset
> str(iris)                   # iris dataset info
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1
 1 1 ...
> subset(iris,Sepal.Length > 5.0 & Sepal.Width > 4)   # subset with Sepal.Leng
th > 5 and Sepal.Width > 4
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
16          5.7         4.4          1.5         0.4  setosa
33          5.2         4.1          1.5         0.1  setosa
34          5.5         4.2          1.4         0.2  setosa
> subset(iris,Petal.Length > 1.0& Petal.Width==0.4)    # subset with Petal.Len
gth>5 and Petal.Width>3.5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
6           5.4         3.9          1.7         0.4  setosa
16          5.7         4.4          1.5         0.4  setosa
17          5.4         3.9          1.3         0.4  setosa
22          5.1         3.7          1.5         0.4  setosa
27          5.0         3.4          1.6         0.4  setosa
32          5.4         3.4          1.5         0.4  setosa
45          5.1         3.8          1.9         0.4  setosa
> subset(iris,Species=="setosa" & Sepal.Length > 5.4)  # subset with Species o
f “setosa” and Sepal’s Length > 5.4
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
15          5.8         4.0          1.2         0.2  setosa
16          5.7         4.4          1.5         0.4  setosa
19          5.7         3.8          1.7         0.3  setosa
34          5.5         4.2          1.4         0.2  setosa
37          5.5         3.5          1.3         0.2  setosa
```

```
> iris_num <- iris[-5]                      # getting only numeric type column
> str(iris_num)                               # updated dataset info
'data.frame':   150 obs. of  4 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
> aggregate(iris_num,by=list(iris$Species),FUN = sum)    # Sum by species type
     Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa        250.3       171.4         73.1        12.3
2 versicolor        296.8       138.5        213.0        66.3
3  virginica        329.4       148.7        277.6       101.3
> aggregate(iris_num,by=list(iris$Species),FUN = mean)   # Mean by species typ
e
     Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa        5.006       3.428        1.462       0.246
2 versicolor        5.936       2.770        4.260       1.326
3  virginica        6.588       2.974        5.552       2.026
> aggregate(iris_num,by=list(iris$Species),FUN = sd)     # Standard Deviation
 by species type
     Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa    0.3524897   0.3790644    0.1736640   0.1053856
2 versicolor    0.5161711   0.3137983    0.4699110   0.1977527
3  virginica    0.6358796   0.3224966    0.5518947   0.2746501
> aggregate(iris_num,by=list(iris$Species),FUN = max)    # Maximum by species
 type
     Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa          5.8         4.4          1.9         0.6
2 versicolor          7.0         3.4          5.1         1.8
3  virginica          7.9         3.8          6.9         2.5
> aggregate(iris_num,by=list(iris$Species),FUN = min)    # Minimum by species
 type
     Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa          4.3         2.3          1.0         0.1
2 versicolor          4.9         2.0          3.0         1.0
3  virginica          4.9         2.2          4.5         1.4
```

Question 2 : Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets.

Solution :

Code :

```
data("mtcars")            # getting mtcars dataset
summary(mtcars)           # summary of mtcars dataset
str(mtcars)               # dataset info
quantile(mtcars$disp)     # getting quantiles for displacement
quantile(mtcars$wt)       # getting quantiles for weight

data("cars")              # getting cars dataset
summary(cars)             # summary of cars dataset
str(cars)                 # dataset info
quantile(cars$speed)      # getting quantiles for speed
quantile(cars$dist)       # getting quantiles for distance
```

Output :

```
> data("mtcars")                  # getting mtcars dataset
> summary(mtcars)               # summary of mtcars dataset
      mpg             cyl             disp             hp
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
 Median :19.20   Median :6.000   Median :196.3   Median :123.0
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
      drat             wt             qsec             vs
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
       am              gear            carb
 Min.   :0.0000   Min.   :3.000   Min.   :1.000
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
 Median :0.0000   Median :4.000   Median :2.000
 Mean   :0.4062   Mean   :3.688   Mean   :2.812
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :1.0000   Max.   :5.000   Max.   :8.000
> str(mtcars)                       # dataset info
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
> quantile(mtcars$disp)    # getting quantiles for displacement
     0%     25%     50%     75%    100%
 71.100 120.825 196.300 326.000 472.000
> quantile(mtcars$wt)      # getting quantiles for weight
     0%     25%     50%     75%    100%
1.51300 2.58125 3.32500 3.61000 5.42400
> data("cars")                  # getting cars dataset
> summary(cars)               # summary of cars dataset
     speed           dist
 Min.   : 4.0   Min.   :  2.00
 1st Qu.:12.0   1st Qu.: 26.00
 Median :15.0   Median : 36.00
 Mean   :15.4   Mean   : 42.98
 3rd Qu.:19.0   3rd Qu.: 56.00
 Max.   :25.0   Max.   :120.00
> str(cars)                       # dataset info
'data.frame':   50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
> quantile(cars$speed)     # getting quantiles for speed
  0%  25%  50%  75% 100%
   4   12   15   19   25
> quantile(cars$dist)       # getting quantiles for distance
  0%  25%  50%  75% 100%
   2   26   36   56  120
```

Question 3 : Reading different types of data sets (.txt, .csv) from web or from existing data set, do the following:

a. Reading Excel data sheet in R.

b. Reading CSV dataset in R.

Solution :

Code :

```
install.packages("readxl")        # installing required package “readxl”
library(readxl)                   # getting library of package “readxl”

# url for xlsx file
url <- "http://www.eia.gov/petroleum/drilling/xls/dpr-data.xlsx"
dm<-basename(url)

# Part (a)
# downloading xlsx file
download.file(url = url, destfile = dm,mode = "wb")
data_excel<-read_excel(dm)          # reading xlsx file in read_excel variable
dim(data_excel)                     # dimensions of xlsx file
head(data_excel)                    # showing first 5 entries

# part (b)
# url for csv file
url <-"http://gattonweb.uky.edu/sheather/book/docs/datasets
      /magazines.csv"
data_csv <- read.csv(url)           # reading csv file in data_csvl variable
dim(data_csv)                       # dimensions of xlsx file
head(data_csv)                      # showing first 5 entries
```

Output :

```
> library(readxl)                          # getting library of package “readxl”
Warning message:
package 'readxl' was built under R version 4.0.5
> url <- "http://www.eia.gov/petroleum/drilling/xls/dpr-data.xlsx"  # url for xlsx file
> dm<-basename(url)
> download.file(url = url, destfile = dm,mode = "wb")    # downloading xlsx file
trying URL 'http://www.eia.gov/petroleum/drilling/xls/dpr-data.xlsx'
Content type 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet' length 15837
0 bytes (154 KB)
downloaded 154 KB

> data_excel<-read_excel(dm)           # reading xlsx file in read_excel variable
New names:
* `` -> ...2
* `` -> ...4
* `` -> ...5
* `` -> ...7
* `` -> ...8
```

```
> dim(data_excel)                              # dimensions of xlsx file
[1] 174   8
> head(data_excel)                            # showing first 5 entries
# A tibble: 6 x 8
  `Anadarko Region` ...2   `Oil (bbl/d)`  ...4     ...5    `Natural gas (M~ ...7    ...8
  <chr>             <chr>  <chr>          <chr>    <chr>   <chr>            <chr>   <chr>
1 Month             Rig c~ Production pe~ Legacy ~ Total ~ Production per ~ Legacy~ Total~
2 39083             176    20.5188174     -520.28~ 126398~ 839.27955599999~ -11168~ 40312~
3 39114             180    21.3920670999~ -873.53~ 134177~ 847.03119800000~ -11495~ 40533~
4 39142             166    22.1683562000~ -1187.6~ 137515~ 853.06840399999~ -11787~ 41653~
5 39173             157    23.0175471000~ -1530.8~ 141635~ 858.62531100000~ -12107~ 41215~
6 39203             170    23.8427818999~ -1859.6~ 144397~ 863.21038699999~ -12415~ 42012~
> # part (b)
> # url for csv file
> url <- "http://gattonweb.uky.edu/sheather/book/docs/datasets/magazines.csv"
> data_csv <- read.csv(url)                # reading csv file in data_csvl variable
> dim(data_csv)                               # dimensions of xlsx file
[1] 204   5
> head(data_csv)                              # showing first 5 entries
                      Magazine AdRevenue AdPages SubRevenue NewsRevenue
1               Weekly World News      2280  300.00        854       16568
2               National Examiner      3382  380.00        968       27215
3                            J-14      4218  250.00       2206       12453
4               Soap Opera Weekly      4622  439.00       5555       24282
5                     Easyriders      5121  523.69       4155        9929
6 Mary Engelbreit's Home Companion      5259  189.00       9048        4363
```

Question 4 : Write an R program to create a vector of a specified type and length. Create a vector of numeric, complex, logical and character types of length 6.

Solution :

Code :

```
x = vector("numeric", 5)
print("Numeric Vector:")
print(x)
c = vector("complex", 5)
print("Complex Vector:")
print(c)
l = vector("logical", 5)
print("Logical Vector:")
print(l)
chr = vector("character", 5)
print("Character Vector:")
print(chr)
```

Output :

```
> x = vector("numeric", 5)
> print("Numeric Vector:")
[1] "Numeric Vector:"
> print(x)
[1] 0 0 0 0 0
> c = vector("complex", 5)
> print("Complex Vector:")
[1] "Complex Vector:"
> print(c)
[1] 0+0i 0+0i 0+0i 0+0i 0+0i
> l = vector("logical", 5)
> print("Logical Vector:")
[1] "Logical Vector:"
> print(l)
[1] FALSE FALSE FALSE FALSE FALSE
> chr = vector("character", 5)
> print("Character Vector:")
[1] "Character Vector:"
> print(chr)
[1] "" "" "" "" ""
```

Question 5 : Write an R program to append value to a given empty vector.

Solution :

Code :

```
vector = c()
values = c('a','b','c','d','e','f','g')
for (i in 1:length(values))
  vector[i] <- values[i]
vector
```

Output :

```
> vector = c()
> values = c('a','b','c','d','e','f','g')
> for (i in 1:length(values))
+   vector[i] <- values[i]
> vector
[1] "a" "b" "c" "d" "e" "f" "g"
```

Question 6 : Write an R program to reverse the order of given vector.

Solution :

Code :

```
v = c(0, 10, 10, 10, 20, 30, 40, 40, 40, 50, 60)
print("Original vector-:")
print(v)
```

```
rv = rev(v)
print("The vector in reverse order:")
print(rv)
```

Output :

```
> v = c(0, 10, 10, 10, 20, 30, 40, 40, 40, 50, 60)
> print("Original vector-:")
[1] "Original vector-:"
> print(v)
 [1]  0 10 10 10 20 30 40 40 40 50 60
> rv = rev(v)
> print("The vector in reverse order:")
[1] "The vector in reverse order:"
> print(rv)
 [1] 60 50 40 40 40 30 20 10 10 10  0
```

Question 7 : Write an R program to convert given data frame column(s) to a vector.

Solution :

Code :

```
dfc1 = c(1, 2, 3, 4, 5)
dfc2 = c(6, 7, 8, 9, 10)
dfc3 = c(11, 12, 13, 14, 15)
dfc4 = c(16, 17, 18, 19, 20)
v <- data.frame(dfc1, dfc2, dfc3, dfc4)
print(v)
```

Output :

```
> dfc1 = c(1, 2, 3, 4, 5)
> dfc2 = c(6, 7, 8, 9, 10)
> dfc3 = c(11, 12, 13, 14, 15)
> dfc4 = c(16, 17, 18, 19, 20)
> v <- data.frame(dfc1, dfc2, dfc3, dfc4)
> print(v)
  dfc1 dfc2 dfc3 dfc4
1    1    6   11   16
2    2    7   12   17
3    3    8   13   18
4    4    9   14   19
5    5   10   15   20
```

Question 8 :

a) Write an R program to create a list containing strings, numbers, vectors and a logical value.

b) Write an R program to convert a given list to a vector.

Solution :

Code :

```
# part (a)
list_data<-list(3,7,12.5,34.7,"hello","world",TRUE,FALSE)
print("List Data is :")
print(list_data)

# part (b)
l1<-list(1,3,"Hii",TRUE)
print("Original list :")
print(l1)
print("List in vector form :")
v1=unlist(l1)
print(v1)
```

Output :

```
> # part (a)
> list_data<-list(3,7,12.5,34.7,"hello","world",TRUE,FALSE)
> print("List Data is :")
[1] "List Data is :"
> print(list_data)
[[1]]
[1] 3

[[2]]
[1] 7

[[3]]
[1] 12.5

[[4]]
[1] 34.7

[[5]]
[1] "hello"

[[6]]
[1] "world"

[[7]]
[1] TRUE

[[8]]
[1] FALSE
```

```
> # part (b)
> l1<-list(1,3,"Hii",TRUE)
> print("Original list :")
[1] "Original list :"
> print(l1)
[[1]]
[1] 1

[[2]]
[1] 3

[[3]]
[1] "Hii"

[[4]]
[1] TRUE

> print("List in vector form :")
[1] "List in vector form :"
> v1=unlist(l1)
> print(v1)
[1] "1"    "3"    "Hii"  "TRUE"
```

Question 9 : Write an R program to create a list containing a vector, a matrix and a list and give names to the elements in the list. Access the first and second element of the list.

Solution :

Code :

```
list_data <- list(c("Red","Green","Black"), matrix(c(1,3,5,7,9,11),
                                nrow = 2), list("Python", "PHP", "Java"))
print("List:")
print(list_data)
names(list_data) = c("Color", "Odd numbers", "Language(s)")
print("List with column names:")
print(list_data)
print('1st element:')
print(list_data$Color)
print('2nd element:')
print(list_data$`Odd numbers`)
```

Output :

```
> list_data <- list(c("Red","Green","Black"), matrix(c(1,3,5,7,9,11), nrow = 2),
+                   list("Python", "PHP", "Java"))
> print("List:")
[1] "List:"
> print(list_data)
[[1]]
[1] "Red"   "Green" "Black"

[[2]]
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11

[[3]]
[[3]][[1]]
[1] "Python"

[[3]][[2]]
[1] "PHP"

[[3]][[3]]
[1] "Java"
```

```
> names(list_data) = c("Color", "odd numbers", "Language(s)")
> print("List with column names:")
[1] "List with column names:"
> print(list_data)
$Color
[1] "Red"   "Green" "Black"

$`odd numbers`
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11

$`Language(s)`
$`Language(s)`[[1]]
[1] "Python"

$`Language(s)`[[2]]
[1] "PHP"

$`Language(s)`[[3]]
[1] "Java"


> print('1st element:')
[1] "1st element:"
> print(list_data$Color)
[1] "Red"   "Green" "Black"
> print('2nd element:')
[1] "2nd element:"
> print(list_data$`odd numbers`)
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11
```

Question 10 : Write an R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from given two vectors.

Solution :

Code :

```
print("Two vectors of different lengths:")
v1 =  c(1,3,4,5)
v2 =  c(10,11,12,13,14,15)
print(v1)
print(v2)
```

```
result = array(c(v1,v2),dim = c(3,3,2))
print("New array:")
print(result)
```

Output :

```
> print("Two vectors of different lengths:")
[1] "Two vectors of different lengths:"
> v1 =  c(1,3,4,5)
> v2 =  c(10,11,12,13,14,15)
> print(v1)
[1] 1 3 4 5
> print(v2)
[1] 10 11 12 13 14 15
> result = array(c(v1,v2),dim = c(3,3,2))
> print("New array:")
[1] "New array:"
> print(result)
, , 1

     [,1] [,2] [,3]
[1,]    1    5   12
[2,]    3   10   13
[3,]    4   11   14

, , 2

     [,1] [,2] [,3]
[1,]   15    4   11
[2,]    1    5   12
[3,]    3   10   13
```

Question 11 :

(a) Write an R program to find the maximum and the minimum value of a given vector.

(b) Write a R program to create a list of heterogeneous data, which include character, numeric and logical vectors. Print the lists.

Solution :

Code :

```
# part (a)
nums = c(10, 20, 30, 40, 50, 60)
print('Original vector:')
print(nums)
print(paste("Maximum value of the said vector:",max(nums)))
print(paste("Minimum value of the said vector:",min(nums)))

# part (b)
my_list = list(Chr=c("Hello","World"), nums = 1:15,
                                    flag=c(TRUE,FALSE))
print(my_list)
```

Output :

```
> # part (a)
> nums = c(10, 20, 30, 40, 50, 60)
> print('Original vector:')
[1] "Original vector:"
> print(nums)
[1] 10 20 30 40 50 60
> print(paste("Maximum value of the said vector:",max(nums)))
[1] "Maximum value of the said vector: 60"
> print(paste("Minimum value of the said vector:",min(nums)))
[1] "Minimum value of the said vector: 10"
> # part (b)
> my_list = list(Chr=c("Hello","World"), nums = 1:15, flag=c(TRUE,FALSE))
> print(my_list)
$Chr
[1] "Hello" "World"

$nums
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15

$flag
[1]  TRUE FALSE
```

Question : 12 Import data set Midwest and do the following :

a) By using ggplot() plot a graph for area v/s poptotal (total population).
b) Plot a bar plot for area v/s poptotal (total population).
c) Plot a pie chart for state, area, poptotal, popdensity, popwhite, popblack.

Solution :

Code :

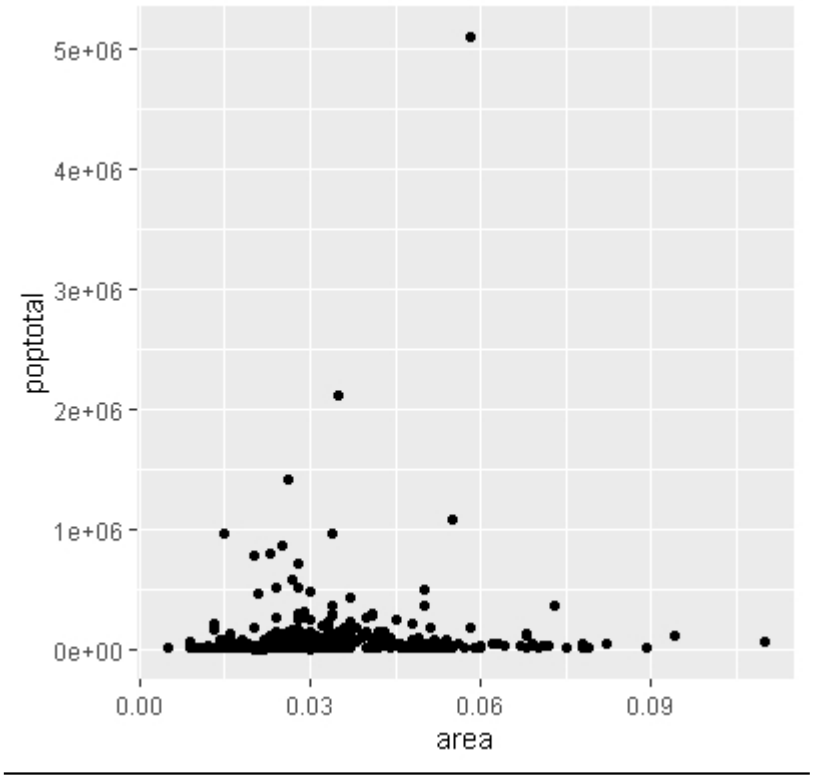
```
install.packages("ggplot2")
library(ggplot2)

# part (a)
data("midwest")
ggplot(data = midwest,mapping = aes(x = area,
                                    y =poptotal))+geom_point()

# part (b)
data("midwest")
ggplot(data = midwest, aes(x=area, y=poptotal)) +
  geom_bar(stat="identity", width=.001, fill="tomato3")

# part (c)
midarea <- aggregate(area~state,midwest,FUN = 'sum')
pie(midarea$area, midarea$state, main = "Area Per State")

midpopt <- aggregate(poptotal~state,midwest,FUN = 'sum')
pie(midpopt$poptotal, midpopt$state,
                              main = "Total Population Per State")
```

```
middnsty <- aggregate(popdensity~state,midwest,FUN = 'sum')
pie(middnsty$popdensity, middnsty$state,
                                main = "Population density Per State")

midpopw <- aggregate(popwhite~state,midwest,FUN = 'sum')
pie(midpopw$popwhite, midpopw$state,
                                main = "White Population Per State")

midpopb <- aggregate(popblack~state,midwest,FUN = 'sum')
pie(midpopb$popblack, midpopb$state,
                                main = "Black Population Per State")
```

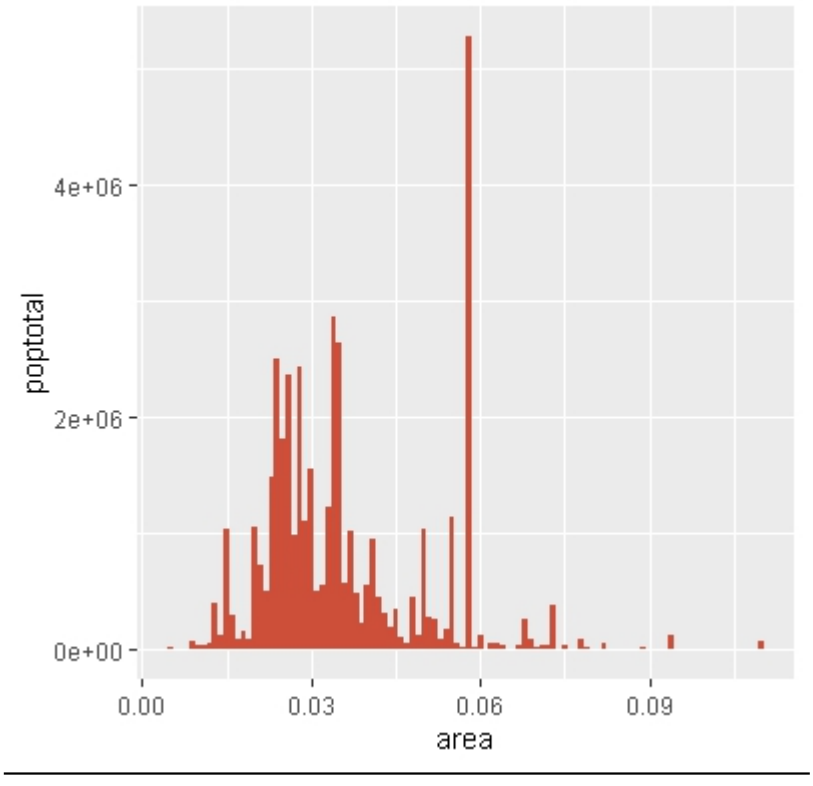
Output :

```
> library(ggplot2)
Warning message:
package 'ggplot2' was built under R version 4.0.5
> data("midwest")
> ggplot(data = midwest,mapping = aes(x = area,y =poptotal))+geom_point()
> # part (b)
> data("midwest")
> ggplot(data = midwest, aes(x=area, y=poptotal)) +
+   geom_bar(stat="identity", width=.001, fill="tomato3")
>
```

(a)

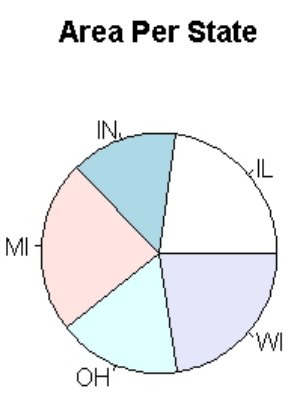

5e+06
4e+06
3e+06
2e+06
1e+06
0e+00
poptotal
0.00
0.03
0.06
0.09
area

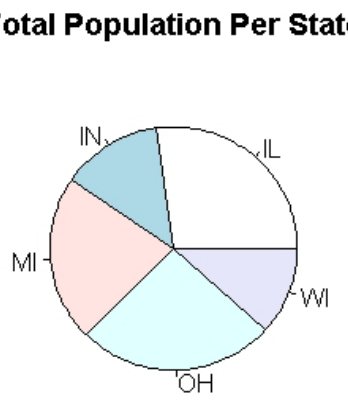
(b)


4e+06
2e+06
0e+00
poptotal
0.00
0.03
0.06
0.09
area

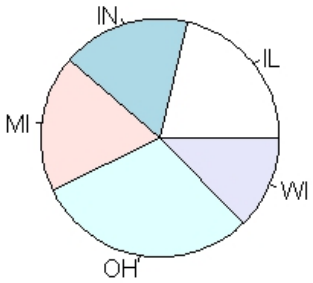
(c)

```
> # part (c)
> midarea <- aggregate(area~state,midwest,FUN = 'sum')
> pie(midarea$area, midarea$state, main = "Area Per State")
> midpopt <- aggregate(poptotal~state,midwest,FUN = 'sum')
> pie(midpopt$poptotal, midpopt$state, main = "Total Population Per State")
> middnsty <- aggregate(popdensity~state,midwest,FUN = 'sum')
> pie(middnsty$popdensity, middnsty$state, main = "Population density Per State")
> midpopw <- aggregate(popwhite~state,midwest,FUN = 'sum')
> pie(midpopw$popwhite, midpopw$state, main = "white Population Per State")
> midpopb <- aggregate(popblack~state,midwest,FUN = 'sum')
> pie(midpopb$popblack, midpopb$state, main = "Black Population Per State")
```

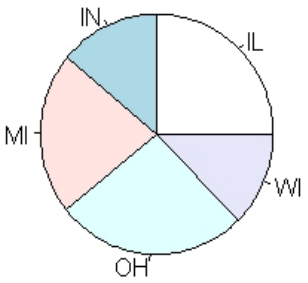




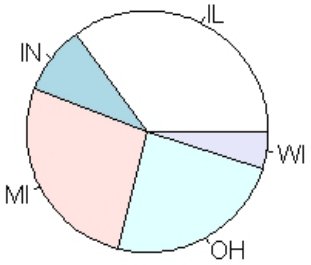
Population density Per State



White Population Per State



Black Population Per State



Question 13 : Create an employe table having :

id	name	salary	start_date	dept
1	Rick	623.30	2012-01-01	IT
2	Dan	515.20	2013-09-23	Operations
3	Michelle	611.00	2014-11-15	IT
4	Ryan	729.00	2014-05-11	HR
5	Gary	843.25	2015-03-27	Finance
6	Nina	578.00	2013-05-21	IT
7	Simon	632.80	2013-07-30	Operations
8	Guru	722.50	2014-06-17	Finance

a) Get the max salary from the data frame.
b) Get the person details having max salary.
c) Get all the people working in the IT department.
d) Get the persons in the IT department whose salary is greater than 600.
e) Get the people who joined on or after 2014.

Solution :

Code :

```
# creating employee table
id <- 1:8
name <- c("Rick", "Dan", "Michelle", "Ryan", "Gary", "Nina",
                                                "Simon", "Guru")

salary <- c(623.30, 515.20, 611.00, 729.00, 843.25, 578.00, 632.80,
                                                    722.50)

start_date <- c( as.Date('2012-01-01'), as.Date('2013-09-23'),
                 as.Date('2014-11-15'), as.Date('2014-05-11'),
                 as.Date('2015-03-27'), as.Date('2013-05-21'),
                 as.Date('2013-07-30'), as.Date('2014-06-17'))

dept <- c("IT", "Operations", "IT", "HR", "Finance", "IT",
                                            "Operations", "Finance")

employee <- data.frame(id, name, salary, start_date, dept)
names(employee) <- c("ID", "Name", "Salary", "Start_Date",
                                                    "Dept")
```

```
# part (a)
max_salary<-max(employee["Salary"])
max_salary
employee

install.packages("tidyverse")
library(tidyverse)

# part (b)
employee %>% filter(employee$Salary==max_salary)

# part (c)
employee %>% filter(employee$Dept=="IT")

# part (d)
employee %>% filter(employee$Dept == "IT", employee$Salary >
                                                        600)

# part (e)
employee %>% filter(employee$Start_Date>"2014-12-31")
```

Output :

```
> # creating employee table
> id<-1:8
> name<-c("Rick","Dan","Michelle","Ryan","Gary","Nina","Simon","Guru")
> salary<-c(623.30,515.20,611.00,729.00,843.25,578.00,632.80,722.50)
> start_date<-c(as.Date('2012-01-01'),as.Date('2013-09-23'),as.Date('2014-11-15'),as.Date('20
14-05-11'),as.Date('2015-03-27'),as.Date('2013-05-21'),as.Date('2013-07-30'),as.Date('2014-06
-17'))
> dept<-c("IT","Operations","IT","HR","Finance","IT","Operations","Finance")
> employee<-data.frame(id,name,salary,start_date,dept)
> names(employee)<-c("ID","Name","Salary","Start_Date","Dept")
> employee
  ID     Name Salary Start_Date       Dept
1  1     Rick 623.30 2012-01-01         IT
2  2      Dan 515.20 2013-09-23 Operations
3  3 Michelle 611.00 2014-11-15         IT
4  4     Ryan 729.00 2014-05-11         HR
5  5     Gary 843.25 2015-03-27    Finance
6  6     Nina 578.00 2013-05-21         IT
7  7    Simon 632.80 2013-07-30 Operations
8  8     Guru 722.50 2014-06-17    Finance
```

```
> # part (a)
> max_salary<-max(employee["Salary"])
> max_salary
[1] 843.25
```

```
> library(tidyverse)
-- Attaching packages --------------------------------------------------- tidyverse 1.3.1 --
v ggplot2 3.3.3     v purrr   0.3.4
v tibble  3.1.1     v dplyr   1.0.5
v tidyr   1.1.3     v stringr 1.4.0
v readr   1.4.0     v forcats 0.5.1
-- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
Warning messages:
1: package 'tidyverse' was built under R version 4.0.5
2: package 'ggplot2' was built under R version 4.0.5
3: package 'tibble' was built under R version 4.0.5
4: package 'tidyr' was built under R version 4.0.5
5: package 'readr' was built under R version 4.0.5
6: package 'purrr' was built under R version 4.0.5
7: package 'dplyr' was built under R version 4.0.5
8: package 'stringr' was built under R version 4.0.5
9: package 'forcats' was built under R version 4.0.5
> # part (b)
> employee %>% filter(employee$Salary==max_salary)
  ID Name Salary Start_Date    Dept
1  5 Gary 843.25 2015-03-27 Finance
> # part (c)
> employee %>% filter(employee$Dept=="IT")
  ID     Name Salary Start_Date Dept
1  1     Rick  623.3 2012-01-01   IT
2  3 Michelle  611.0 2014-11-15   IT
3  6     Nina  578.0 2013-05-21   IT
> # part (d)
> employee %>% filter(employee$Dept=="IT",employee$Salary>600)
  ID     Name Salary Start_Date Dept
1  1     Rick  623.3 2012-01-01   IT
2  3 Michelle  611.0 2014-11-15   IT
> # part (e)
> employee %>% filter(employee$Start_Date>"2014-12-31")
  ID Name Salary Start_Date    Dept
1  5 Gary 843.25 2015-03-27 Finance
```

Question 14 : In the library MASS is a dataset UScereal which contains information about popular breakfast cereals. Attach the data set and use different kinds of plots to investigate the following relationships :

(a) Relationship between manufacturer and shelf.
(b) Relationship between fat and vitamins.
(c) Relationship between fat and shelf.
(d) Relationship between carbohydrates and sugars.
(e) Relationship between fibre and manufacturer.
(f) Relationship between sodium and sugars.

Solution :

Code :

```
install.packages("MASS")
library(MASS)
library(ggplot2)
library(lattice)

data("UScereal")
attach(UScereal)

#   a) relationship between manufacturer and shelf
table(mfr, shelf)

#   b) relationship between fat and vitamins
xyplot(vitamins~fat, UScereal)
xyplot(vitamins~jitter(fat,100), UScereal)
stripchart(fat~vitamins, pch = 19, method = "jitter")
```

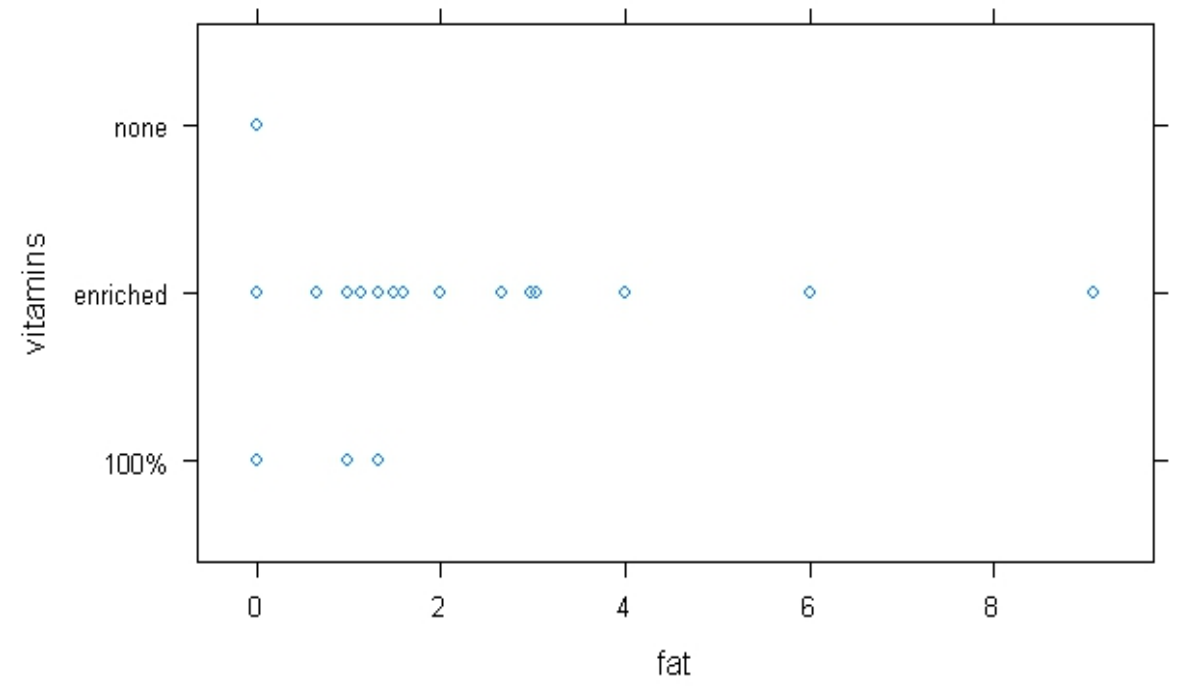
```
stripplot(vitamins~jitter(fat,100))
ggplot(UScereal, aes(fat, vitamins)) +
                                        geom_point(position = "jitter")

#   c) relationship between fat and shelf
ggplot(UScereal,aes(fat, shelf)) + geom_point(position = "jitter")

#   d) relationship between carbohydrates and sugars
ggplot(UScereal,aes(carbo,sugars)) + geom_point()

#   e) relationship between fibre and manufacturer
ggplot(UScereal,aes(mfr,fibre)) + geom_point(position = "jitter")

#   f) relationship between sodium and sugars
ggplot(UScereal,aes(sodium,sugars)) + geom_point()
```

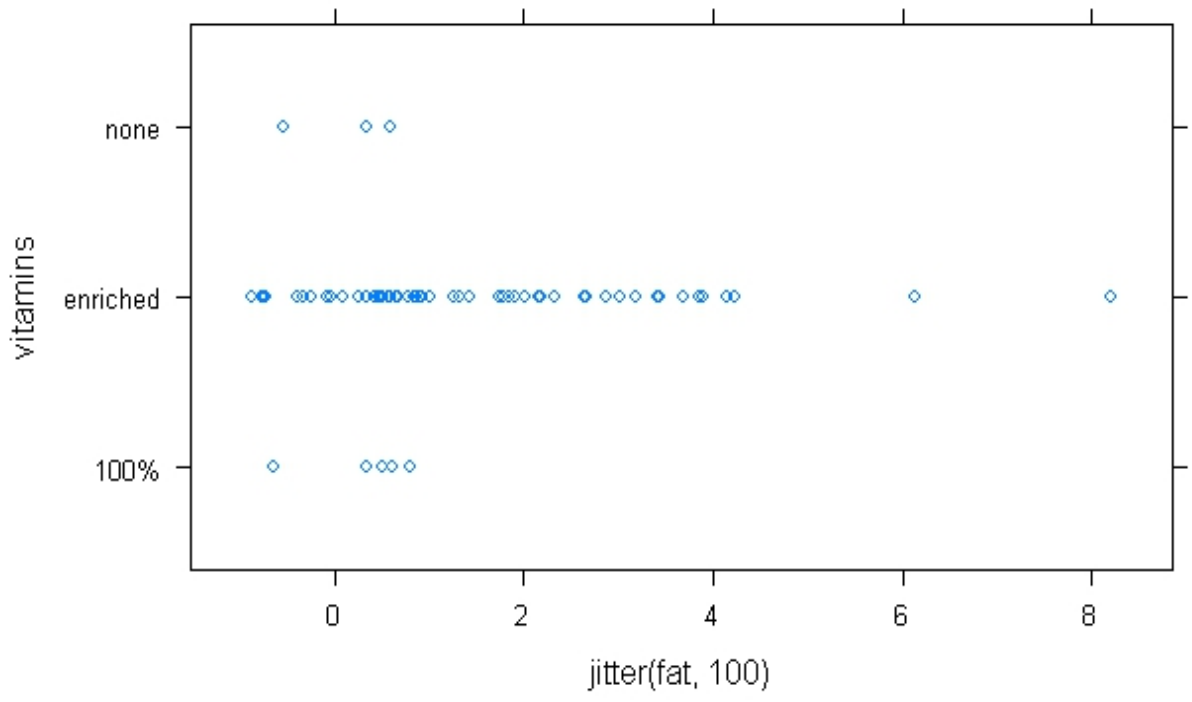
Output :

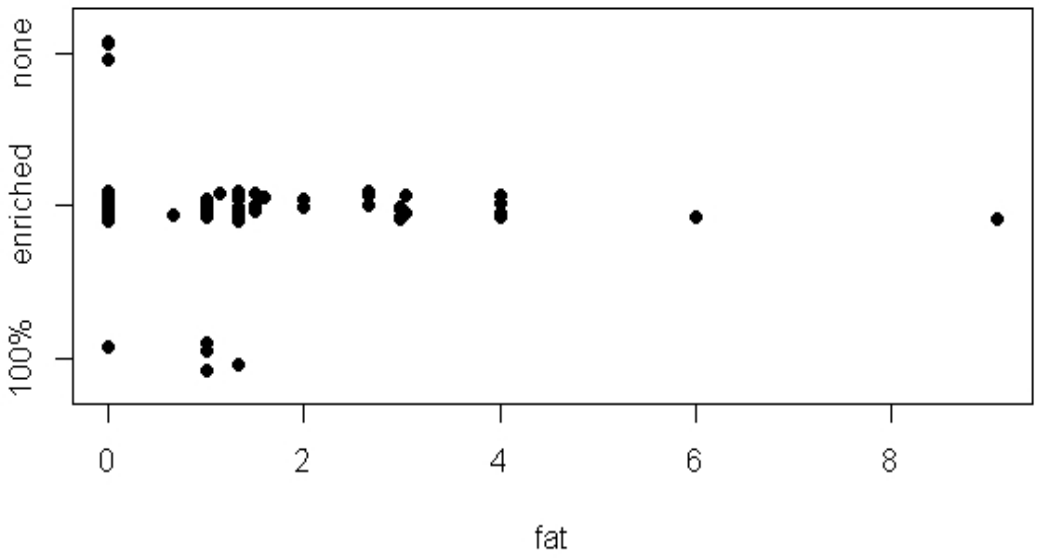
(a)

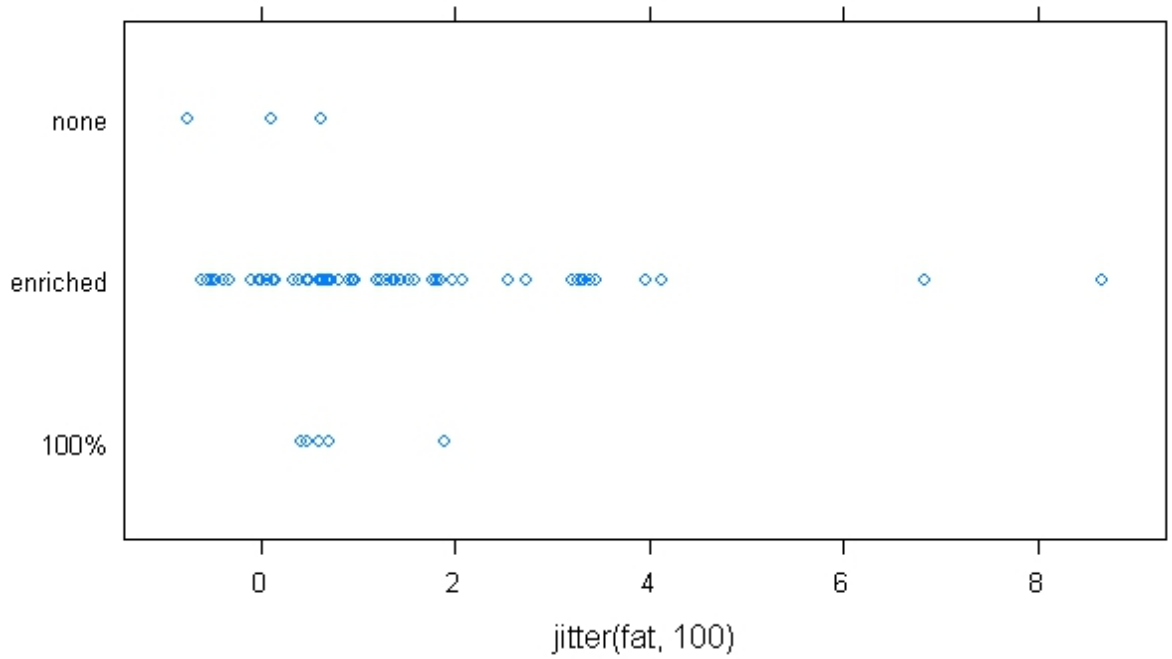
```
> #   a) relationship between manufacturer and shelf
> table(mfr,shelf)
   shelf
mfr  1  2  3
  G  6  7  9
  K  4  7 10
  N  2  0  1
  P  2  1  6
  Q  0  3  2
  R  4  0  1
```

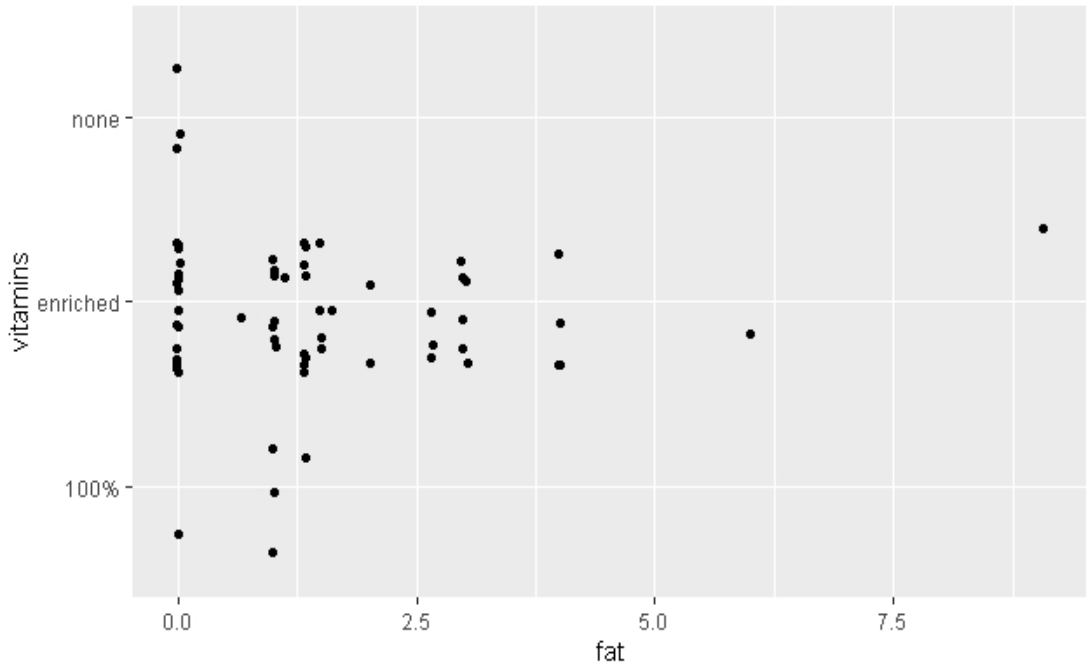
(b)


none
enriched
100%
vitamins
0
2
4
6
8
fat

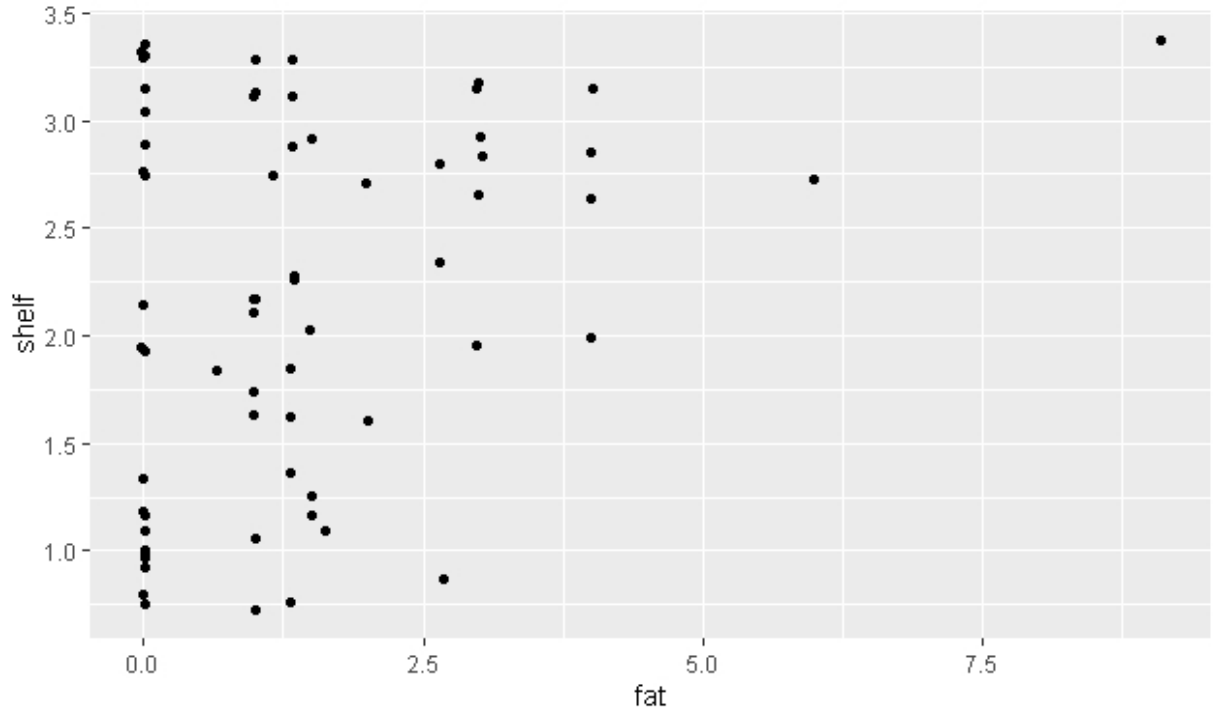

none
enriched
100%
vitamins
0
2
4
6
8
jitter(fat, 100)


none
enriched
100%
0
2
4
6
8
fat

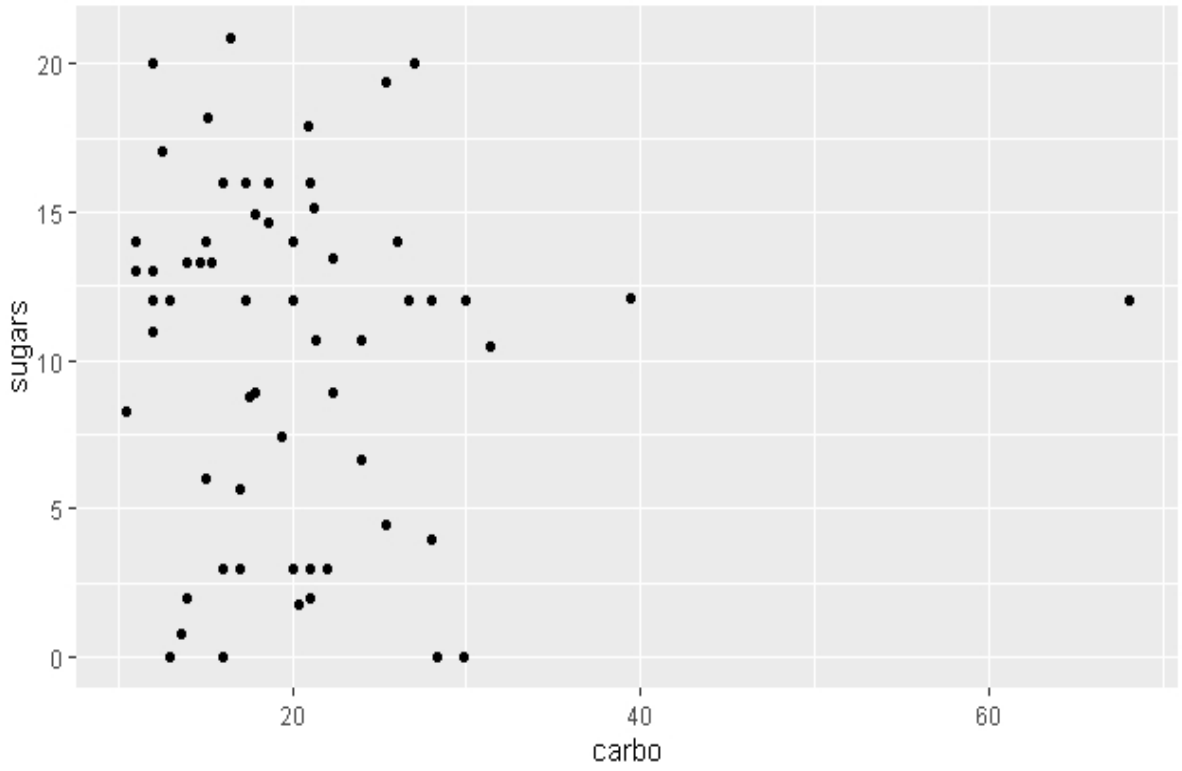

none
enriched
100%
0
2
4
6
8
jitter(fat, 100)


none
enriched
100%
vitamins
0.0
2.5
5.0
7.5
fat

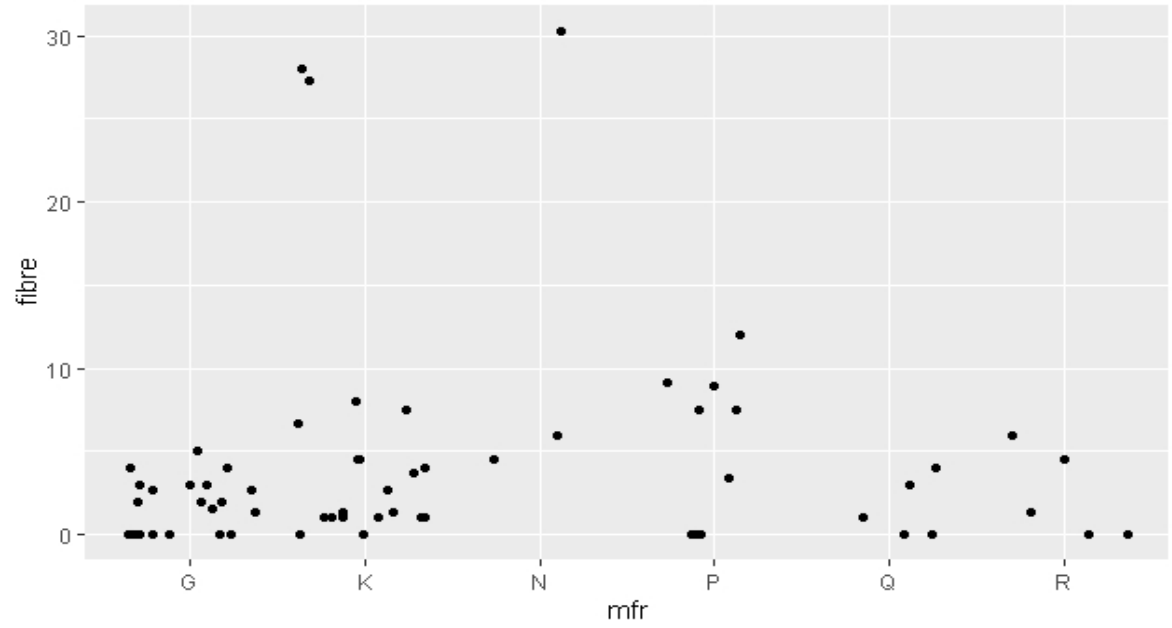
(c)


3.5
3.0
2.5
2.0
1.5
1.0
shelf
0.0
2.5
5.0
7.5
fat

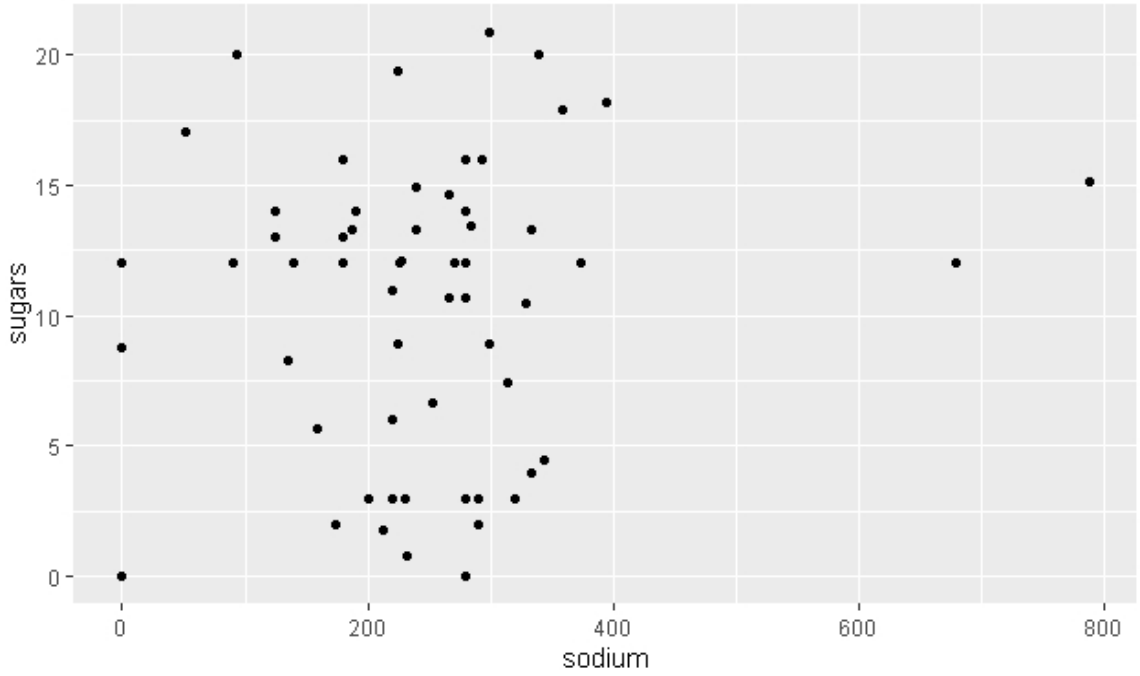
(d)



(e)



(f)



Question 15 : Write an R script to do the following:

a) simulate a sample of 100 random data points from a normal distribution with mean 100 & standard deviation 5 and store the result in a vector.

b) visualize the vector created above using different plots.

c) test the hypothesis that the mean equals 100.

d) use wilcox test to test the hypothesis that mean equals 90.

Solution :

Code :

```
library('stats')
options(warn = -1) #For disable any kind of warnings !!!
```

```
#a. Sampling of Normal Distribution with mean = 100 & S.D. = 5
x <- c(rnorm(100, mean = 100L, sd = 5L))
cat("\n\t\t Practical 1 \n\n A. Random Sample of Normal
Distribution (x) :\n\n")
print(x)
cat('\n\n')

#b. Plotting of the Samples in a 3 * 2 matrix
par(mfrow = c(3, 2), col = 'black') # Matrix Designing
plot(x, ylab = "Samples", main = 'Point Plotting')

plot(x,
     ylab = "Samples",
     main = 'Line Plotting',
     type = 'l')

plot(x,
     ylab = "Samples",
     main = 'Points with Line Plotting',
     type = 'b')

plot(x,
     ylab = "Samples",
     main = 'Line without Points Plotting',
     type = 'c')
```

```
plot(x,
     ylab = "Samples",
     main = 'Stair Plotting',
     type = 's')

plot(x,
     ylab = "Samples",
     main = 'Histogram Plotting',
     type = 'h')

mtext(
  'B. Random Sample in Different Plots',
  line = -1.5 ,
  outer = TRUE,
  col = 6
)

#c. Hypothesis Testing for mean=100
result <- t.test(x, mu = 100)
print('C. Hypothesis testing for Mean = 100')
print(result)

#d. Wilcox Testing for mean=90
result <- wilcox.test(x, mu = 90)
print('D. Wilcox test for Mean/Location = 90')
print(result)
```

Output :

```
               Practical 1

A. Random Sample of Normal Distribution (x) :

 [1] 110.40815  96.67767 103.94387 101.99050 105.11878 110.86306 101.86215
 [8]  98.19763 109.90217  96.72286  96.15182 105.07477  93.40556  95.85809
[15] 106.83570 100.21838  96.61373  97.75356 104.20270 102.63547 111.08113
[22]  98.70791 100.05532  87.62350 106.04729 104.34495 106.02933 103.18900
[29]  87.45021 102.08527  99.24826  96.14832  88.07398  98.45858  93.78041
[36]  98.00982  94.42703 101.84031  99.59488 103.36294  97.54067 108.35563
[43] 101.66600  91.45751  98.74982  91.32278  96.17024 104.91791  95.37440
[50] 107.18314  99.30737 101.77657  99.55503  98.75090 108.36331 105.19859
[57] 108.03375 101.83959 100.29325 102.70360  96.43348  96.30976  99.47207
[64] 105.85451 101.42610 103.72824  93.34641  97.93295 107.64041  98.11233
[71] 102.99606  94.06394 109.50026 102.46131 104.78327 101.29754 101.96030
[78]  97.33364  97.17207 105.50548 102.24483  96.83740 100.66057  86.92272
[85] 101.99855 107.88961  96.43964  92.75042  93.60224 105.52426  98.16329
[92]  96.37903 112.81633  94.78781 106.81434 101.58959 103.03307 101.22210
[99]  99.10401  96.84271
```

Click here for Q 15. (b)

```
[1] "C. Hypothesis testing for Mean = 100"

        One Sample t-test

data:  x
t = 0.83445, df = 99, p-value = 0.406
alternative hypothesis: true mean is not equal to 100
95 percent confidence interval:
  99.37297 101.53719
sample estimates:
mean of x
 100.4551

[1] "D. Wilcox test for Mean/Location = 90"

        Wilcoxon signed rank test with continuity correction

data:  x
V = 5031, p-value < 2.2e-16
alternative hypothesis: true location is not equal to 90
```

Question 16 : Using the Algae data set from package DMwR2 to complete the following tasks :

a) Create a graph that you find adequate to show the distribution of the values of algae a6.

b) Show the distribution of the values of size 3.

c) Check visually if oPO4 follows a normal distribution.

d) Produce a graph that allows you to understand how the values of NO3 are distributed across the sizes of river.

e) Using a graph check if the distribution of algae a1 varies with the speed of the river.

f) Visualize the relationship between the frequencies of algae a1 and a6. Give the appropriate graph title, x-axis and y-axis title.

Solution :

Code :

```
library('DMwR2')

#import & set up database for user
attach(algae)
color = c('red', 'green', 'orange', 'blue')

#A. Plotting a multiple graph to show the distribution of algae a6
par(mfrow = c(2, 2)) #plot graphs in a grid
plot(
  x = algae$a6,
  xlab = 'Index',
  ylab = 'algae a6',
  type = 'l'
)
```

```
plot(
  x = algae$season,
  y = algae$a6,
  xlab = 'Season',
  ylab = 'algae a6',
  col = color
)

plot(
  x = algae$size,
  y = algae$a6,
  xlab = 'River Size',
  ylab = 'algae a6',
  col = color
)

plot(
  x = algae$speed,
  y = algae$a6,
  xlab = 'River Speed',
  ylab = 'algae a6',
  col = color
)

mtext(
  text = 'A. Graph of distribution of Algae a6',
  line = -2,
```

```
  outer = TRUE,
  col = 'dark blue',
  font = 2,
  cex = 1.5
)

#B. Showing the distribution of values of size = 3
algae.size3 <- algae[algae$size == 'small', ]
View(x = algae.size3,
     title = 'B. Distribution of Algae values for size = 3')

#c. Checking visually that algae$oPO4 is normally distributed or not
par(mfrow = c(1, 2))

boxplot(x = algae$oPO4, ylab = 'algae$oPO4')
mtext(
     text = 'Here,\n Values evenly distributed\t: No\n Median is
               in centre\t\t: No\n Normally Distributed\t\t: No',
      side = 1,
      line = 4
)
hist(x = algae$oPO4, main = '')
mtext(
  text = 'Here,\n Bell-shaped Histogram : No\n Normally
                                          Distributed : No',
  adj = 1,
  line = -3
)
```

```
mtext(
  text = 'C. Visual Normality Check of Algae oPO4',
  line = -2,
  outer = TRUE,
  col = 6,
  font = 2,
  cex = 1.5
)

# D. NO3 distribution regarding river size
par(mfrow = c(1, 1))

plot(
  x = algae$size,
  y = algae$NO3,
  xlab = 'River Size',
  ylab = 'algae NO3',
  col = color
)

mtext(
  text = 'D. NO3 Distribution across River Sizes',
  line = -2,
  outer = TRUE,
  col = 6,
  font = 2,
  cex = 1.2
)
```

```
# E. Algae A1 distribution across River Speed
plot(
  x = algae$speed,
  y = algae$a1,
  xlab = 'River Speed',
  ylab = 'Algae a1',
  col = color
)

mtext(
  text = 'E. Algae A1 distribution across River Speed',
  line = -2,
  outer = TRUE,
  col = 20,
  font = 2,
  cex = 1.2
)

# F. Algae A1 relationship with Algae A6
par(mfrow = c(1,2))

plot(
  x = algae$a1,
  y = algae$a6,
  xlab = 'Algae a1',
  ylab = 'Algae a6',
)
```

```
boxplot(
  x = algae$a1,algae$a6,
  names = c('Algae a1', 'Algae a6'),
  col = color[2:3]
)

mtext(
  text = 'F. Relationship of Algae A1 with Algae A6',
  line = -2,
  outer = TRUE,
  col = 20,
  font = 2,
  cex = 1.2
)
```

Output :

```
B. Distribution of Algae values for size = 3
tibble [71 × 18] (S3: tbl_df/tbl/data.frame)
 $ season: Factor w/ 4 levels "autumn","spring",..: 4 2 1 2 1 4 3 1 4 4 ...
 $ size  : Factor w/ 3 levels "large","medium",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ speed : Factor w/ 3 levels "high","low","medium": 3 3 3 3 3 1 1 1 3 1 ...
 $ mxPH  : num [1:71] 8 8.35 8.1 8.07 8.06 8.25 8.15 8.05 8.7 7.93 ...
 $ mnO2  : num [1:71] 9.8 8 11.4 4.8 9 13.1 10.3 10.6 3.4 9.9 ...
 $ Cl    : num [1:71] 60.8 57.8 40 77.4 55.4 ...
 $ NO3   : num [1:71] 6.24 1.29 5.33 2.3 10.42 ...
 $ NH4   : num [1:71] 578 370 346.7 98.2 233.7 ...
 $ oPO4  : num [1:71] 105 428.8 125.7 61.2 58.2 ...
 $ PO4   : num [1:71] 170 558.8 187.1 138.7 97.6 ...
 $ Chla  : num [1:71] 50 1.3 15.6 1.4 10.5 ...
 $ a1    : num [1:71] 0 1.4 3.3 3.1 9.2 15.1 2.4 18.2 25.4 17 ...
 $ a2    : num [1:71] 0 7.6 53.6 41 2.9 14.6 1.2 1.6 5.4 0 ...
 $ a3    : num [1:71] 0 4.8 1.9 18.9 7.5 1.4 3.2 0 2.5 0 ...
 $ a4    : num [1:71] 0 1.9 0 0 0 0 3.9 0 0 2.9 ...
 $ a5    : num [1:71] 34.2 6.7 0 1.4 7.5 22.5 5.8 5.5 0 0 ...
 $ a6    : num [1:71] 8.3 0 0 0 4.1 12.6 6.8 8.7 0 0 ...
 $ a7    : num [1:71] 0 2.1 9.7 1.4 1 2.9 0 0 0 1.7 ...
>
```

Click Here for Q.16 Graphs

Question 17 : The built-in data set mammals contain data on body weight versus brain weight. Write R commands to:

a) Find the Pearson and Spearman correlation coefficients. Are they similar?

b) Plot the data using the plot command .

c) Plot the logarithm (log) of each variable and see if that makes a difference.

Solution :

Code :

```
library('MASS', 'ggpubr')
options(warn = -1)

attach(mammals)

pearson <- cor(x = mammals$body,
               y = mammals$brain,
               method = 'pearson')

spearman <- cor(x = mammals$body,
                y = mammals$brain,
                method = 'spearman')
cat(
  "\n\t\t\t Practical 4 \n\n\n A. Pearson & Spearman Correlation
  Coefficients\n",  '\n\n Pearson Correlation Coefficient (cor)  \t :
  ',  pearson,  '\n\n Spearman Correltaion Coefficient (rho) \t : ',
```

```
  spearman, '\n\n Pearson & Spearman coefficients are
  equal ?\t : ',  spearman == pearson,  '\n\n Difference of
  Spearman & Pearson coefficients : ',  spearman - pearson,
  '\n\n Any other Similarities : \n\t\t Both Correlation coefficients
    are showing a strong positive realtionship & association
    between Mammals\' Body & Brain !!!\n'
)

# B. Plotting the mammals data.frame
par(mfrow = c(2, 1))

plot(
  x = mammals$brain,
  y = mammals$body,
  xlab = 'Mammals\'s Brain',
  ylab = 'Mammals\' Body'
)

boxplot(
  x = mammals$brain,
  mammals$body,
  horizontal = TRUE,
  names = c('Brain', 'Body'),
  col = c('orange', 'dark green')
)

mtext(
  text = 'B. Plotting the values of \nMammals\' Brain v/s
                                                    Mammals\' Body',
```

```
  outer = TRUE,
  col = 6,
  line = -3,
  font = 2
)

# C. Plotting the log values of mammals data.frame
plot(
  x = mammals$brain,
  y = mammals$body,
  xlab = 'Mammals\'s Brain',
  ylab = 'Mammals\' Body',
  log = 'xy'
)

boxplot(
  x = mammals$brain,
  mammals$body,
  horizontal = TRUE,
  names = c('Brain', 'Body'),
  col = c('orange', 'dark green'),
  log = 'x'
)

mtext(
  text = 'C. Plotting the logarithm values of \nMammals\' Brain
                                          v/s Mammals\' Body',
  outer = TRUE,
```

```
  col = 6,
  line = -3,
  font = 2
)

# Practice
plot.ts(mammals)
plot.ts(mammals, log = 'y')
```

Output :

```
                        Practical 4


A. Pearson & Spearman Correlation Coefficients


Pearson Correlation Coefficient (cor)         :  0.9341638

Spearman Correltaion Coefficient (rho)        :  0.9534986

Pearson & Spearman coefficients are equal ?   :  FALSE

Difference of Spearman & Pearson coefficients :  0.01933478

Any other Similarities :
                Both Correlation coefficients are showing a
        strong positive realtionship & association between
        Mammals' Body & Brain !!!
>
```

Click Here for Q.17 Graphs

Question 18 : In the library MASS is a dataset UScereal which contains information about popular breakfast cereals. Attach the data set and use different kinds of plots to investigate the following relationships:

a) relationship between manufacturer and shelf
b) relationship between fat and vitamins
c) relationship between fat and shelf
d) relationship between carbohydrates and sugars
e) relationship between fibre and manufacturer
f) relationship between sodium and sugars

Solution :

Code :

```
library('MASS')

attach(UScereal)

# A. Relation of Manufacturer & Shelf
plot(
  x = UScereal$mfr,
  y = UScereal$shelf,
  xlab = 'Manufacturer',
  ylab = 'Shelf',
  main = 'A. Relation of Manufacturer & Shelf of UScereal',
  col = c(3:7)
)
```

```
# B. Relation of Fat & Vitamins
plot(
  x = UScereal$vitamins,
  y = UScereal$fat,
  xlab = 'Vitamins',
  ylab = 'Fat',
  main = 'B. Relation of Fat & Vitamins of UScereal',
  col = 'purple'
)

# C. Relation of Fat & Shelf
boxplot(
  x = UScereal$fat,
  UScereal$shelf,
  names = c('Fat', 'Shelf'),
  main = 'C. Relation of Fat & Shelf of UScereal',
  col = c('orange','brown'),
  horizontal = TRUE,
  xlab = 'Range of Values'
)

# D. Relation of Carbohydrates & Sugars
plot(
  x = UScereal$carbo,
  y = UScereal$sugars,
  xlab = 'Carbs',
  ylab = 'Sugars',
  main = 'D. Relation of Carbohydrates & Sugars of UScereal',
  col = 'dark green'
```

```
)

# E. Relation of Manufacturer & Fibre
plot(
  x = UScereal$mfr,
  y = UScereal$fibre,
  xlab = 'Manufacturer',
  ylab = 'Fibre',
  main = 'E. Relation of Manufacturer & Fibre of UScereal',
  col = c(3:8)
)

# F. Relation of Sodium & Sugars
plot(
  x = UScereal$carbo,
  y = UScereal$sugars,
  xlab = 'Sodium',
  ylab = 'Sugars',
  main = 'F. Relation of Sodium & Sugars of UScereal',
  col = 'dark blue'
)
```

Output :

Click Here for Q.18 Graphs

Question 19 : Write R script to:

a) Do two simulations of a binomial number with n = 100 and p = .5. Do you get the same results each time? What is different? What is similar?

b) Do a simulation of the normal two times. Once with n = 10, μ = 10 and σ = 10, the other with n = 10, μ = 100 and σ = 100. How are they different? How are they similar? Are both approximately normal?

Solution :

Code :

```
library('stats')

nvars <- data.frame(n1 = rnorm(10, 10L, 10L),
                    n2 = rnorm(10, 100L, 100L))

bvars <- data.frame(b1 = rbinom(n = 100, size = 10L, p = .5),
                    b2 = rbinom(n = 100, size = 10L, p = .5))

cat("\nA.\n Correlation Coefficient : ", cor(bvars$b1, bvars$b2),
'\n\n Summary :\n' )
print(summary(nvars))
cat('\n Data : \n')
str(nvars)

cat("\nB. Correlation Coefficient : ",cor(nvars$n1, nvars$n2,
method = 'spearman'), '\n\n Summary :\n')
print(summary(bvars))
cat('\n Data : \n')
str(bvars)

par(mfrow = c(2,2))
```

```
plot(nvars, main = 'A. Random Normally Distributed Variables')
boxplot(nvars, col = c('red', 'green'))
hist(bvars$b1, main = 'B. Random Binomially Distributed
Variables')
hist(bvars$b2)
```

Output :

```
> source("~/Desktop/B.SC./IV/R programming/Practicals/Q6.R")

A.
 Correlation Coefficient :  -0.00926523

 Summary :
       n1               n2
 Min.   : 4.780   Min.   :-58.87
 1st Qu.: 6.424   1st Qu.:174.53
 Median : 9.954   Median :229.18
 Mean   :11.763   Mean   :225.97
 3rd Qu.:14.116   3rd Qu.:328.76
 Max.   :28.130   Max.   :364.91

 Data :
'data.frame':   10 obs. of  2 variables:
 $ n1: num  19.66 9.57 15.25 10.72 8.46 ...
 $ n2: num  259 165 365 232 226 ...
```

```
B. Correlation Coefficient :  0.769697

 Summary :
       b1              b2
 Min.   :1.00   Min.   :1.0
 1st Qu.:4.00   1st Qu.:4.0
 Median :5.00   Median :5.0
 Mean   :4.92   Mean   :5.2
 3rd Qu.:6.00   3rd Qu.:6.0
 Max.   :8.00   Max.   :8.0

 Data :
'data.frame':   100 obs. of  2 variables:
 $ b1: int  5 2 6 6 6 6 6 5 2 4 ...
 $ b2: int  5 4 5 7 6 1 5 4 7 4 ...
> ``
```

[Click Here for Q.19 Graphs]

Question 20 : Create a package in R to perform certain basic statistics functions.

Solution :

Code :

```
install.packages(c("devtools", "roxygen2"))
library("devtools", "roxygen2")
```

linearTrend.R

```
#' Linear Trend
#'
#' This function uses time-series data to create a linear trend line plot.
#'
#' @author Avi
#' @param t  A time vector from the time-series.
#' @param yt An entity data which is mapped according to t.
#' @param ... Extra arguments that will be used in plot().
#' @return ye A trend vector calculated from t & yt.
#'
#' @import graphics
#' @export
#' @examples
#' linearTrend(t = c(1997:2002), yt = c(10,12,15,16,18,19))
#'

linearTrend <- function(t, yt, ...)
{
  #Error Checking
  if (!is.vector(t) | !is.numeric(t))
    stop('t must be a numeric vector !!!')

  if (!is.vector(yt) | !is.numeric(yt))
    stop('yt must be a numeric vector !!!')

  if (length(t) != length(yt))
    stop('t & yt must be of same length !!!')
```

```
  #Calculations
  n = length(t)
  x <- t - mean(t)
  x.2 <- x ^ 2
  x.yt <- x * yt

  if (as.integer(sum(x)) != 0)
    stop('Please share more accurate data !!!')

  a = sum(yt) / n
  b = sum(x.yt) / sum(x.2)
  ye = a + b*x

  plot(
    x = t,
    y = yt,
    ylim = c(min(ye,yt), max(ye,yt)),
    ... = ...
  )
  lines(
    x = t,
    y = ye,
  )
  return(ye)
}

package.skeleton("astats", c("linearTrend"))
```

Description

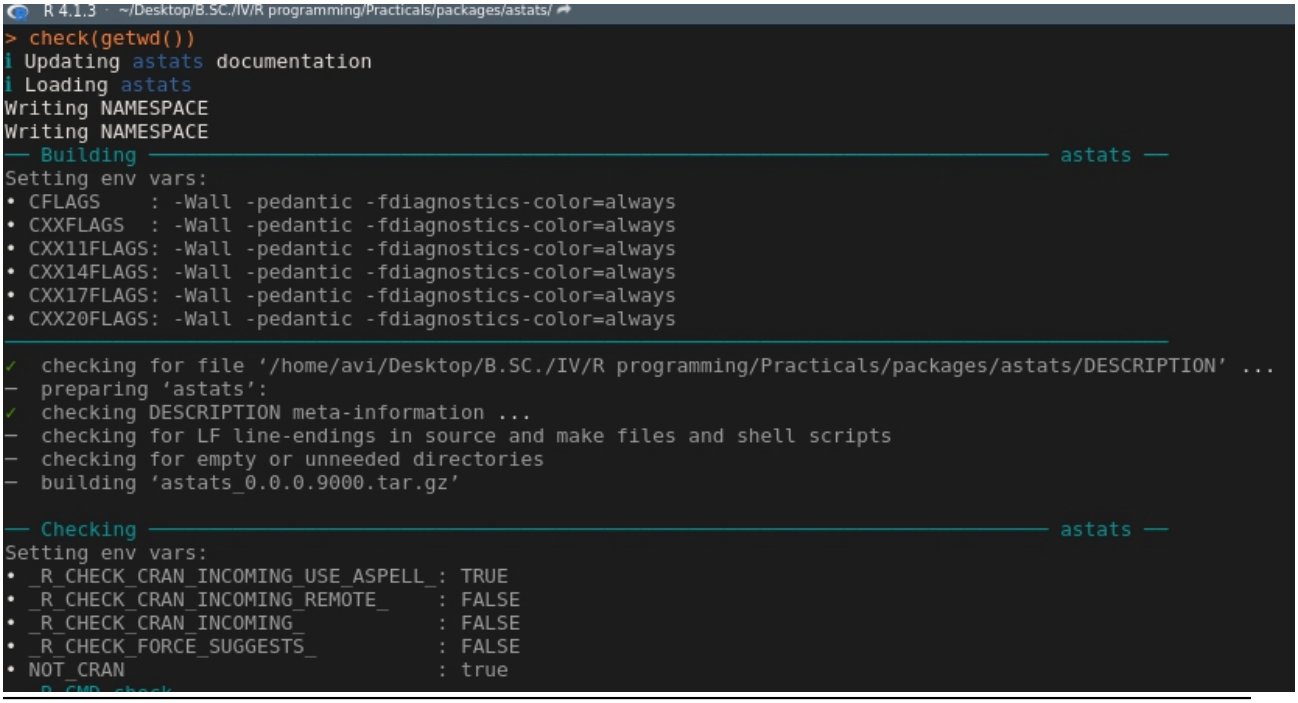
Package: astats
Title: Package for Applied Statistics Tools
Version: 0.0.0.9000
Author: Avi

Authors@R:
    person("Avinash", "Gautam", , "boss.avinashg@gmail.com",
           role = c("aut", "cre"),
           comment = c(GitHub = "Avinash-msc"))

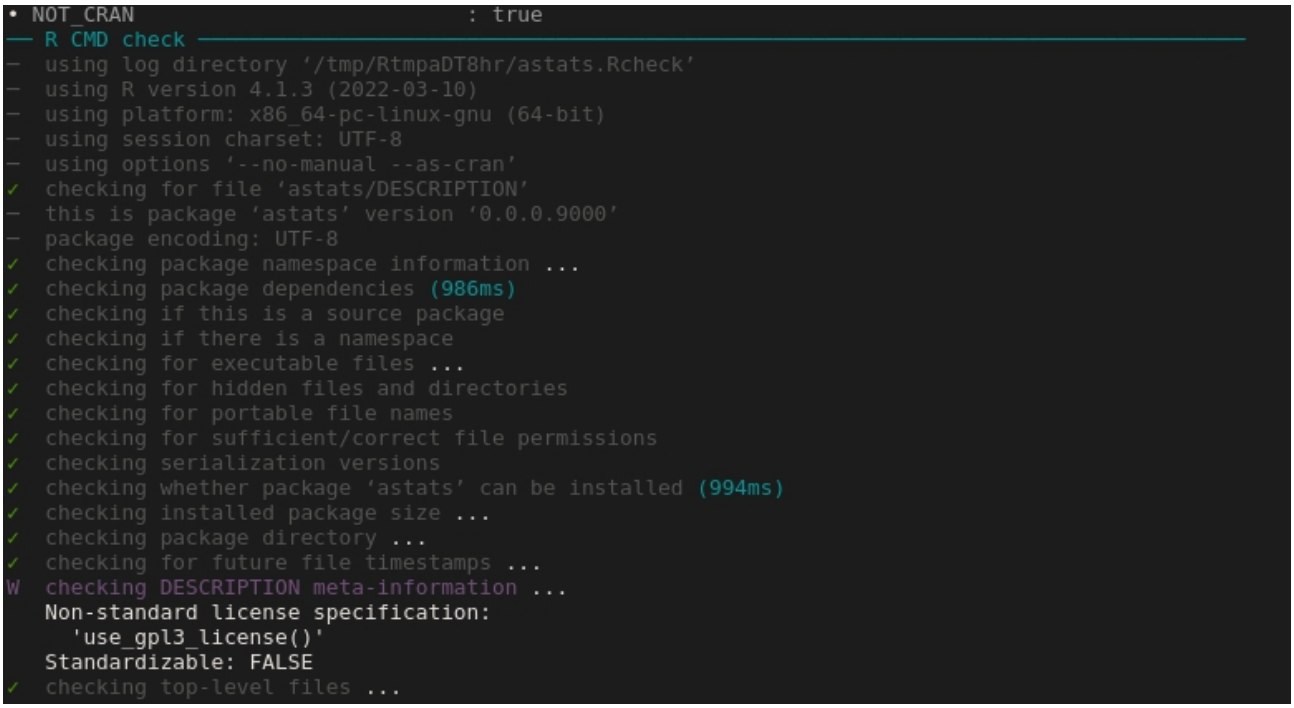
Description:
    In this astats package, my goal is to create function tools capable of doing statistical analysis.
For example:
    linearTrend() -> returns the linear trend of the time-series & plot the trendline
    Xbar.Plot() -> returns the vector of control limits & plot the control chart for mean
    Range.Plot() -> returns the vector of control limits & plot the control chart for range & so on.

License: 'use_gpl3_license()'
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.1.2
Imports:
    graphics

Output :



```
R 4.1.3 · ~/Desktop/B.SC./IV/R programming/Practicals/packages/astats/
> check(getwd())
i Updating astats documentation
i Loading astats
Writing NAMESPACE
Writing NAMESPACE
── Building ──────────────────────────────────────────────────────────────── astats ──
Setting env vars:
• CFLAGS    : -Wall -pedantic -fdiagnostics-color=always
• CXXFLAGS  : -Wall -pedantic -fdiagnostics-color=always
• CXX11FLAGS: -Wall -pedantic -fdiagnostics-color=always
• CXX14FLAGS: -Wall -pedantic -fdiagnostics-color=always
• CXX17FLAGS: -Wall -pedantic -fdiagnostics-color=always
• CXX20FLAGS: -Wall -pedantic -fdiagnostics-color=always
──────────────────────────────────────────────────────────────────────────────
✓  checking for file '/home/avi/Desktop/B.SC./IV/R programming/Practicals/packages/astats/DESCRIPTION' ...
─  preparing 'astats':
✓  checking DESCRIPTION meta-information ...
─  checking for LF line-endings in source and make files and shell scripts
─  checking for empty or unneeded directories
─  building 'astats_0.0.0.9000.tar.gz'

── Checking ──────────────────────────────────────────────────────────────── astats ──
Setting env vars:
• _R_CHECK_CRAN_INCOMING_USE_ASPELL_: TRUE
• _R_CHECK_CRAN_INCOMING_REMOTE_    : FALSE
• _R_CHECK_CRAN_INCOMING_           : FALSE
• _R_CHECK_FORCE_SUGGESTS_          : FALSE
• NOT_CRAN                          : true
```



```
• NOT_CRAN                          : true
── R CMD check ─────────────────────────────────────────────────────────────
─  using log directory '/tmp/RtmpaDT8hr/astats.Rcheck'
─  using R version 4.1.3 (2022-03-10)
─  using platform: x86_64-pc-linux-gnu (64-bit)
─  using session charset: UTF-8
─  using options '--no-manual --as-cran'
✓  checking for file 'astats/DESCRIPTION'
─  this is package 'astats' version '0.0.0.9000'
─  package encoding: UTF-8
✓  checking package namespace information ...
✓  checking package dependencies (986ms)
✓  checking if this is a source package
✓  checking if there is a namespace
✓  checking for executable files ...
✓  checking for hidden files and directories
✓  checking for portable file names
✓  checking for sufficient/correct file permissions
✓  checking serialization versions
✓  checking whether package 'astats' can be installed (994ms)
✓  checking installed package size ...
✓  checking package directory ...
✓  checking for future file timestamps ...
W  checking DESCRIPTION meta-information ...
   Non-standard license specification:
     'use_gpl3_license()'
   Standardizable: FALSE
✓  checking top-level files ...
```

```
  'use_gpl3_license()'
Standardizable: FALSE
✓  checking top-level files ...
✓  checking for left-over files
✓  checking index information
✓  checking package subdirectories ...
✓  checking R files for non-ASCII characters ...
✓  checking R files for syntax errors ...
✓  checking whether the package can be loaded ...
✓  checking whether the package can be loaded with stated dependencies ...
✓  checking whether the package can be unloaded cleanly ...
✓  checking whether the namespace can be loaded with stated dependencies ...
✓  checking whether the namespace can be unloaded cleanly ...
✓  checking loading without being on the library search path ...
✓  checking dependencies in R code ...
✓  checking S3 generic/method consistency (486ms)
✓  checking replacement functions ...
✓  checking foreign function calls ...
✓  checking R code for possible problems (1.8s)
✓  checking Rd files ...
✓  checking Rd metadata ...
✓  checking Rd line widths ...
✓  checking Rd cross-references ...
✓  checking for missing documentation entries ...
✓  checking for code/documentation mismatches ...
✓  checking Rd \usage sections (542ms)
✓  checking Rd contents ...
✓  checking for unstated dependencies in examples ...
✓  checking examples (491ms)
✓  checking for non-standard things in the check directory
✓  checking for detritus in the temp directory

See
  '/tmp/RtmpaDT8hr/astats.Rcheck/00check.log'
for details.
```

```
── R CMD check results ──────────────────────────────────────── astats 0.0.0.9000 ────
Duration: 8.6s

> checking DESCRIPTION meta-information ... WARNING
  Non-standard license specification:
    'use_gpl3_license()'
  Standardizable: FALSE

0 errors ✓ | 1 warning x | 0 notes ✓
> build(getwd())
✓  checking for file '/home/avi/Desktop/B.SC./IV/R programming/Practicals/packages/astats/DESCRIPTION' ...
−  preparing 'astats':
✓  checking DESCRIPTION meta-information ...
−  checking for LF line-endings in source and make files and shell scripts
−  checking for empty or unneeded directories
−  building 'astats_0.0.0.9000.tar.gz'

[1] "/home/avi/Desktop/B.SC./IV/R programming/Practicals/packages/astats_0.0.0.9000.tar.gz"
> release(getwd())
```

R: Linear Trend Find in Topic

linearTrend {astats} R Documentation

Linear Trend

Description

This function uses time-series data to create a linear trend line plot.

Usage

```
linearTrend(t, yt, ...)
```

Arguments

t	A time vector from the time-series.
yt	An entity data which is mapped according to t.
...	Extra arguments that will be used in plot().

Value

ye A trend vector calculated from t & yt.

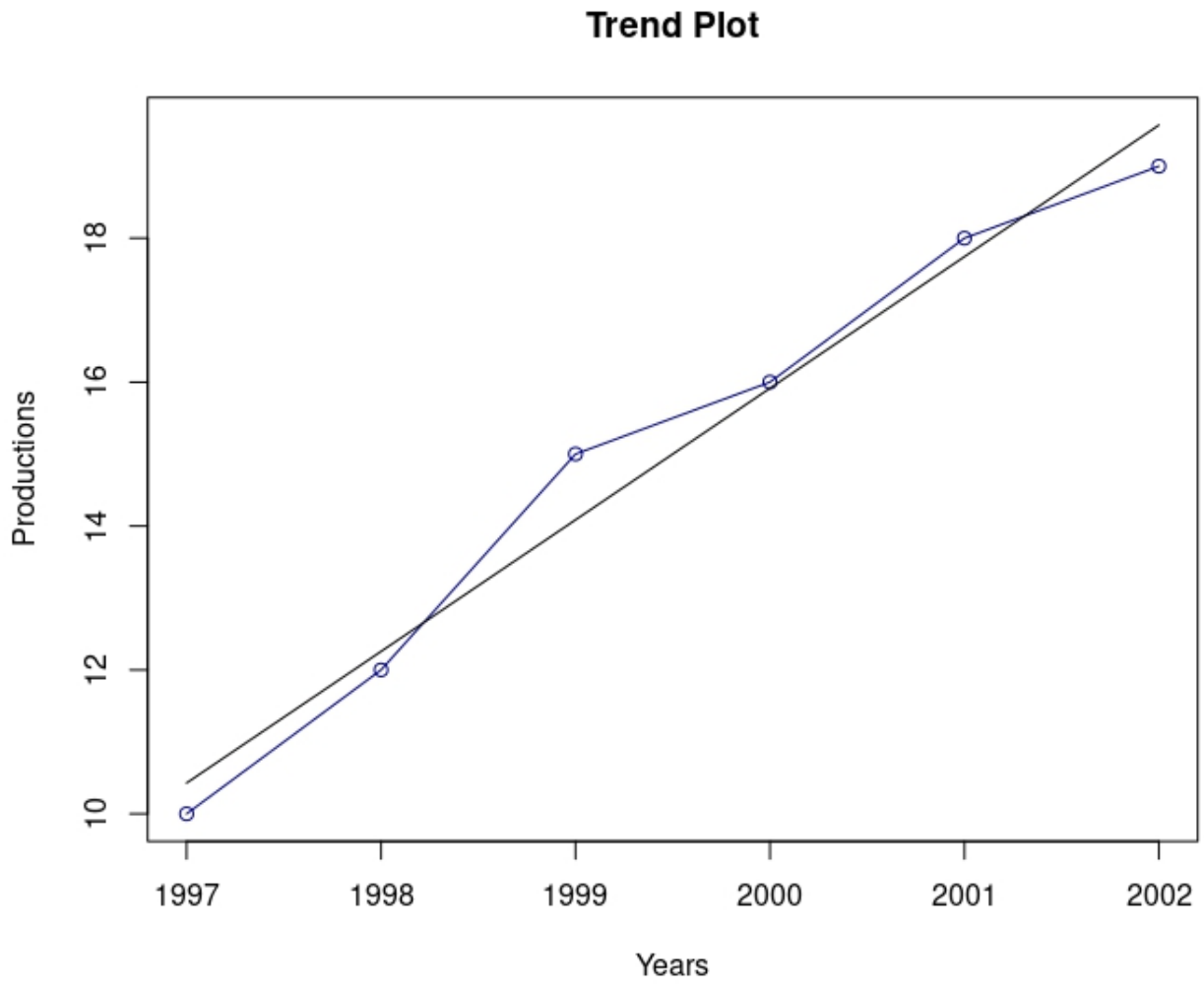
Author(s)

Avi

Examples

```
linearTrend(t = c(1997:2002), yt = c(10,12,15,16,18,19))
```

```
>
>
> cat('\n Linear Trend : ',
+ linearTrend(t = c(1997:2002), yt = c(10,12,15,16,18,19),
+ xlab = 'Years', ylab = 'Productions',
+ main = 'Trend Plot', col = 'dark blue', type = 'o')
+ )

 Linear Trend :  10.42857 12.25714 14.08571 15.91429 17.74286 19.57143
>
```



The End