Practical No. 10 Date: / / 201

Aim: Program to visualize given data using graphics functions.

Objectives:

- To study R Graphics Functions.
- Implement a program to visualize given data using same.

Theory:

R Graphics

One of the main reason of using R for data science is its graphics functionalities. R Programming language has numerous libraries to create plots, charts and graphs. This is done using R code which makes replication & modifications easier. Some of the common packages of R includes ggplot2, ggvis, lattice.

There are many functions in these packages that can be used to visualize data, we will see plot() (scatterplot)& hist() (histogram). We consider following contents of state_info data frame for visualization:

```
             name region   area population
1      Maharashtra   West 307.71      11.23
2              Goa   West   3.70       0.14
3    Uttar Pradesh  North 243.29      19.98
4 Himachal Pradesh  North  55.67       0.68
5            Assam   East  78.43       3.11
6      West Bengal   East  88.75       9.13
7           Kerala  South  38.86       3.33
8       Tamil Nadu  South 130.06       7.21
```

plot() Function

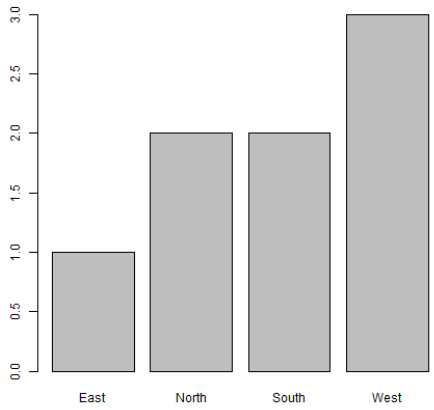
Syntax

```
plot(x, y, main, xlab, ylab, xlim, ylim, axes)
```

- x: data set whose values are the horizontal coordinates.
- y: data set whose values are the vertical coordinates.
- main: tile of the graph.
- xlab: label in the horizontal axis.
- ylab: label in the vertical axis.
- xlim: limits of the values of x used for plotting.
- ylim: limits of the values of y used for plotting.
- axes: indicates whether both axes should be drawn on the plot.

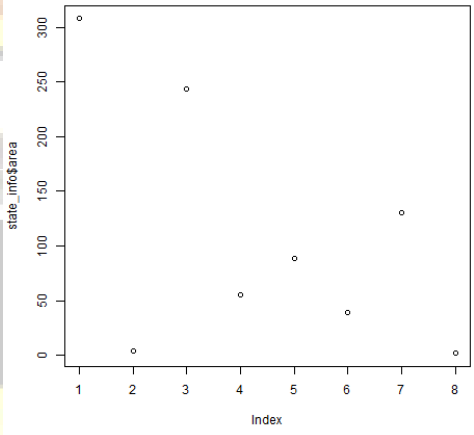
Categorical Data

```
png(file = "cat_info.jpg")
plot(state_info$region)
```



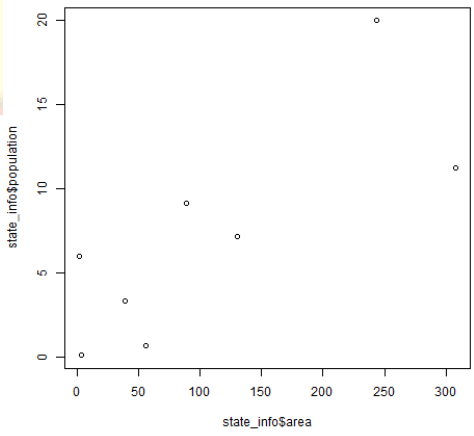
Numerical Data

```
png(file = "num_info.jpg")
plot(state_info$area)
```



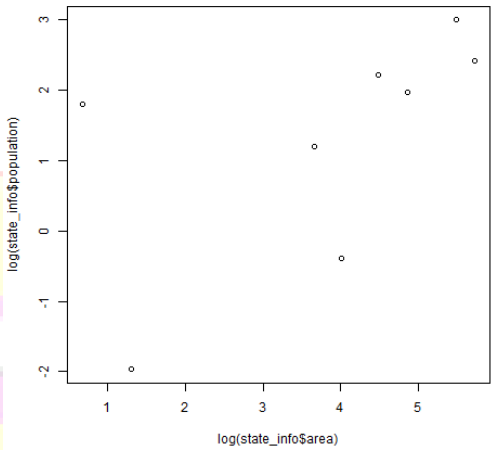
2 x Numerical Data

```
png(file = "num2_info.jpg")
plot(state_info$area, state_info$population)
```



2 x Numerical Data with log()

```
png(file = "numlog_info.jpg")
plot(log(state_info$area), log(state_info$population))
```



hist() Function

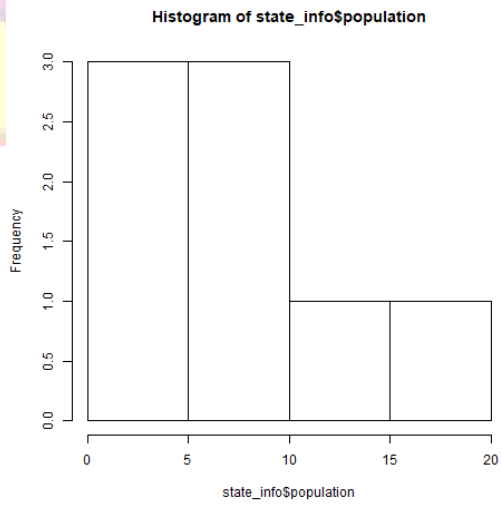
Syntax

```
hist(v, main, xlab, xlim, ylim, breaks, col, border)
```

- v: vector containing numeric values used in histogram.
- main: indicates title of the chart.
- col: used to set color of the bars.
- border: used to set border color of each bar.
- xlab: used to give description of x-axis.
- xlim: used to specify the range of values on the x-axis.
- ylim: used to specify the range of values on the y-axis.
- breaks: used to mention the width of each bar.

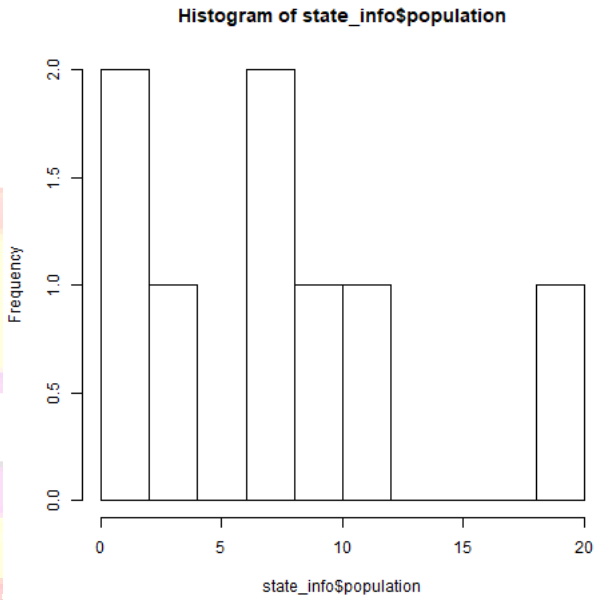
With Default bins

```
png(file = "hist_pop.jpg")
hist(state_info$population)
```



With 10 bins

```
png(file = "hist_10.jpg")
hist(state_info$population, breaks = 10)
```



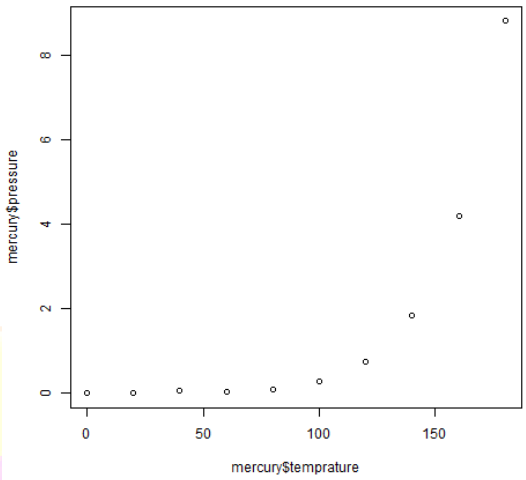
plot() Function (Customization)

We can also use other parameter(s) of plot() functions to make the plot more informative. We consider following contents of mercury data frame for visualization:

```
   temprature pressure
1           0   0.0002
2          20   0.0012
3          40   0.0600
4          60   0.0300
5          80   0.0900
6         100   0.2700
7         120   0.7500
8         140   1.8500
9         160   4.2000
10        180   8.8000
```

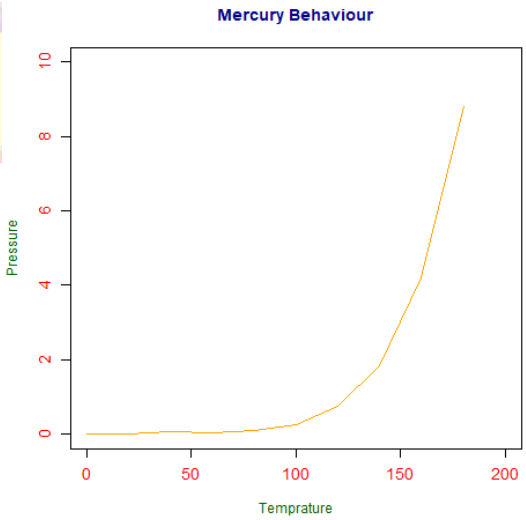
Basic Plot

```
png(file = "mer_basic.jpg")
plot(mercury$temprature, mercury$pressure)
```



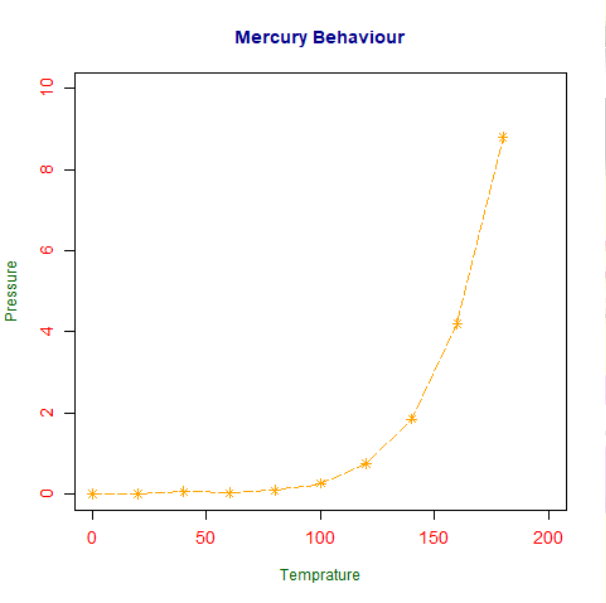
Plot with simple line

```
png(file = "mer_line.jpg")
plot(mercury$temprature,
     mercury$pressure,
     main = "Mercury Behaviour",
     xlab = "Temprature",
     ylab = "Pressure",
     xlim = c(0,200),
     ylim = c(0,10),
     type = "l",
     col = "orange",
     col.main = "darkblue",
     col.lab = "darkgreen",
     col.axis = "red",
     cex.axis = 1.2)
```

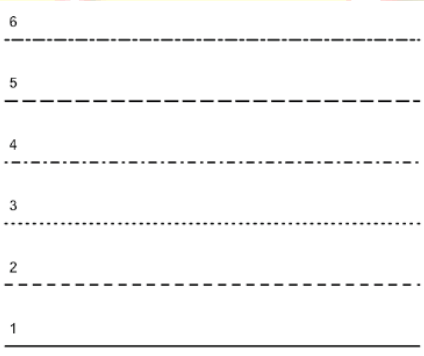


Plot with * symbol & -- line

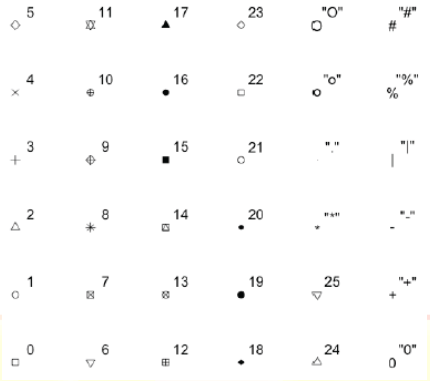
```
png(file = "mer_line.jpg")
plot(mercury$temprature,
     mercury$pressure,
     main = "Mercury Behaviour",
     xlab = "Temprature",
     ylab = "Pressure",
     xlim = c(0,200),
     ylim = c(0,10),
     type = "o",
     col = "orange",
     col.main = "darkblue",
     col.lab = "darkgreen",
     col.axis = "red",
     cex.axis = 1.2,
     lty = 5,
     pch = 8)
```



lty (Line Types)



pch (Symbol Types)

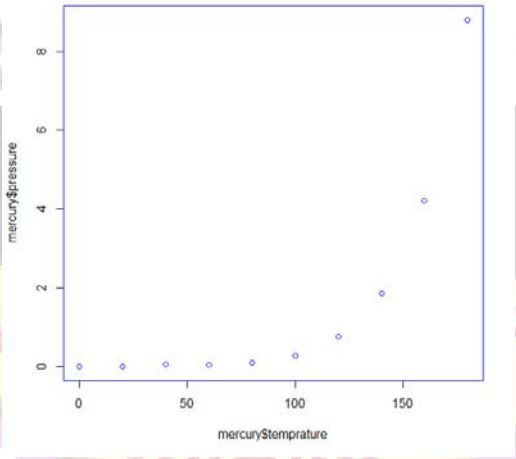


par() Function

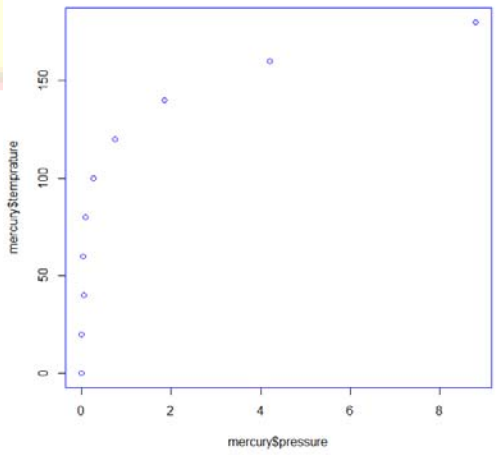
The graphical parameters can also be checked or specified by par() function. By calling par() function current values of these parameters can be checked. The par() function can also be used to set values for these parameters for current session.

Example

```
> par(col = "blue")
> plot(mercury$temprature, mercury$pressure)
```




```
> plot(mercury$pressure, mercury$temprature)
```



```
> par()$col
 [1] blue
```

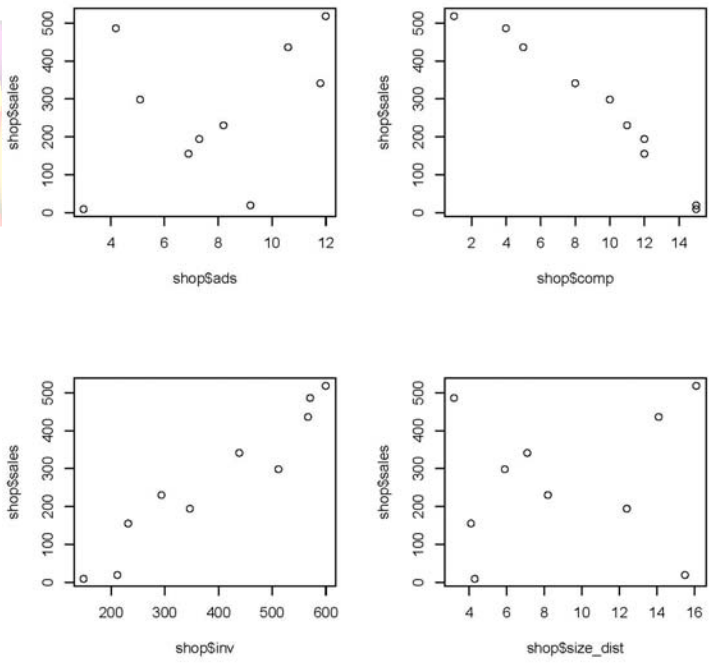
par() Function: Multiple Plots

Many times it is useful to display information using combination of multiple plots in same grid. Arguments of par() function like mfrows & mfcol, or layout() function can be used for the same. Consider following contents of shop data frame:

```
   sales  ads comp inv size_dist
1    231  8.2   11 294       8.2
2    156  6.9   12 232       4.1
3     10  3.0   15 149       4.3
4    519 12.0    1 600      16.1
5    437 10.6    5 567      14.1
6    487  4.2    4 571       3.2
7    299  5.1   10 512       5.9
8    195  7.3   12 347      12.4
9     20  9.2   15 212      15.5
10   342 11.8    8 439       7.1
```

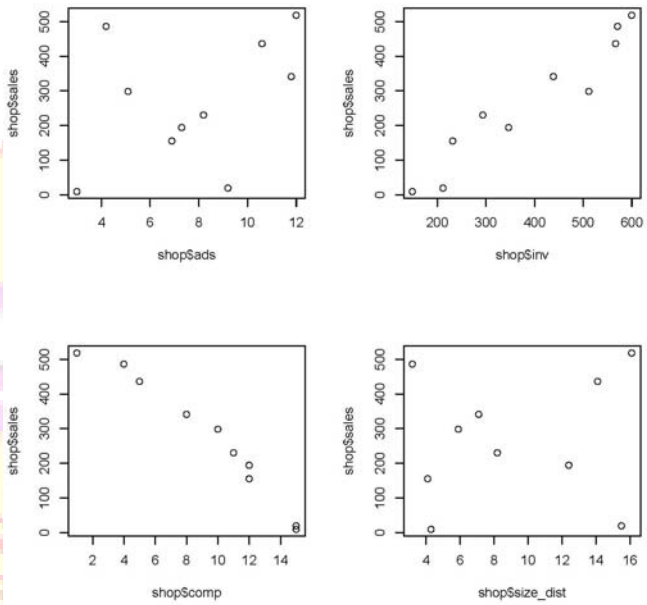
mfrow parameter

```
par(mfrow = c(2,2))
plot(shop$ads, shop$sales)
plot(shop$comp, shop$sales)
plot(shop$inv, shop$sales)
plot(shop$size_dist, shop$sales)
```



mfcol parameter

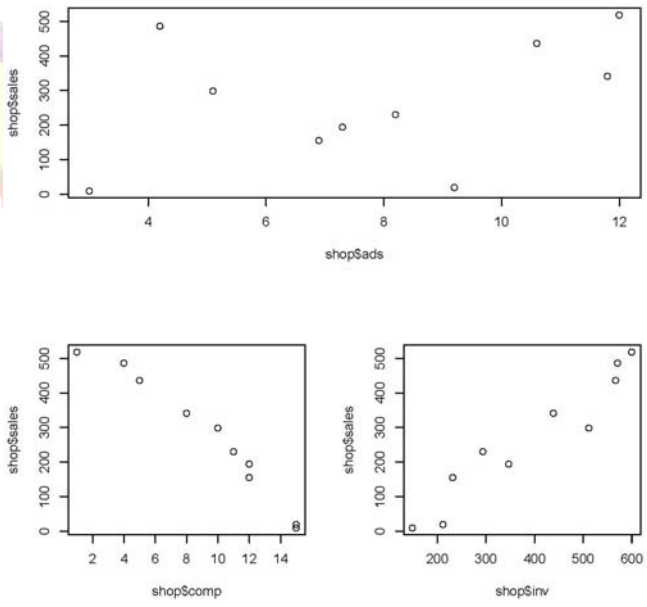
```
par(mfcol = c(2,2))
plot(shop$ads, shop$sales)
plot(shop$comp, shop$sales)
plot(shop$inv, shop$sales)
plot(shop$size_dist, shop$sales)
```



layout() Function (Multiple Plots)

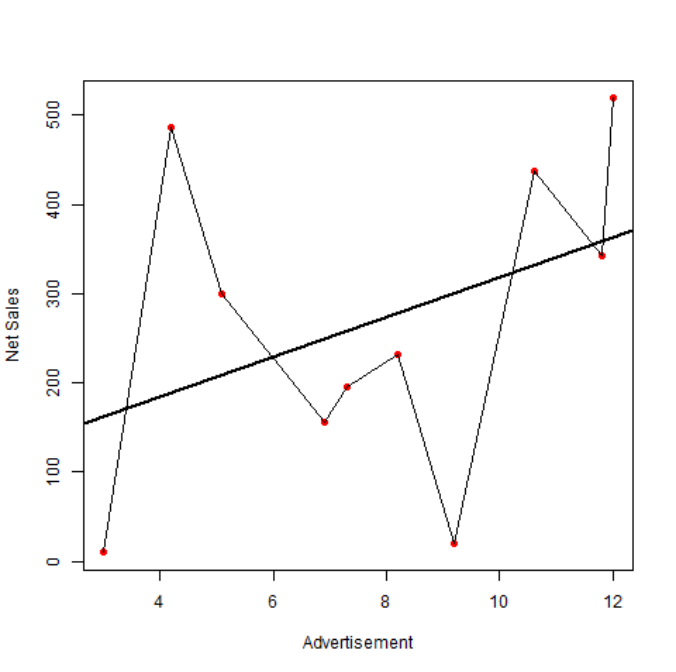
requires matrix to specify locations of figures

```
grid <- matrix(c(1,1,2,3), nrow = 2, byrow = TRUE)
layout(grid)
plot(shop$ads, shop$sales)
plot(shop$comp, shop$sales)
plot(shop$inv, shop$sales)
```



Adding Layers to Plot

```
png(file = "add_info.jpg")
plot(shop$ads, shop$sales,
      pch = 16,
      col = 2,
      xlab = "Advertisement",
      ylab = "Net Sales")
lm_sales <- lm(shop$sales ~ shop$ads)
                      #returns linear fit coefficients
abline(coef(lm_sales), lwd = 2)
                      #plots a straight line, lwd (line width)
ranks <- order(shop$ads)
lines(shop$ads[ranks], shop$sales[ranks])
                  #connect points in order using lines
```



Resetting to Default

Grid

```
par(mfrow = c(1,1))
par(mfcol = c(1,1))
layout(1)
```

All Parameters

```
old_par <- par()
#Perform desired graphics operations
par(old_par)
```

Other Functions

```
barplot()
boxplot()
pairs()
points()
segments()
text()
```

Algorithm

1. Start.
2. Create data frames “state_info”, “mercury” & “shop”.
3. Name columns of data frames using any one method.
4. Read choice to visualize data frame contents from menu as
    a. Plot Categorical Data
    b. Plot 2xNumeric Data
    c. Create Histogram
    d. Create Customized Plot
    e. Create Multiple Plots
5. As per choice perform visualizations as
    a. If choice is “a”, visualize categorical data column using plot().
    b. If choice is “b”, visualize two numerical data columns using plot().
    c. If choice is “c”, visualize numerical data column using hist().
    d. If choice is “d”, visualize two numerical data columns with additional graphic parameter using plot() or par().
    e. If choice is “e”, visualize three/four pairs numerical data columns using par() or layout().
6. Stop.