STATISTICS WITH R PROGRAMMING

UNIT-4

Graphics

Creating Graphs

The Workhorse of R Base Graphics, the plot() Function

A. plot() function

B. Starting New Graph- x11() function

C. points() function

D. legend() function

E. text() function

F. locator() function

barplot() function

boxplot() function

Histogram - hist() function

PIE-CHART : pie() function

Strip- Chart : stripchart() function

Customizing Graphs

Saving Graphs to Files

R- Colors

Graphics

R has a very rich set of graphics facilities. The R home page (*http://www.r-project.org/*) has a few colorful examples, but to really appreciate R's graphical power, browse through the R Graph Gallery at *http://addictedtor.free.fr/graphiques*.

Creating Graphs

The Workhorse of R Base Graphics, the plot() Function

A. plot() function

A Plot is a graph that connects a series of points by drawing line segments between them. These points are ordered in one of their coordinate (usually the x-coordinate) value. Plots are usually used in identifying the trends in data.

Plots are also called as Line Charts

The **plot()** function in R is used to creates the plot.

Syntax

```
plot(x, y, type, main, sub, xlab, ylab, asp , ...)
```

Arguments

`x` The x coordinates of points in the plot.

`y` the y coordinates of points in the plot,

`type`

what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both '**o**verplotted',
- `"h"` for '**h**istogram' like (or 'high-density') vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see 'Details' below,
- `"n"` for no plotting.

`main`

an overall title for the plot:

`sub`

a sub title for the plot: see.

`xlab`

a title for the x axis:

`ylab`

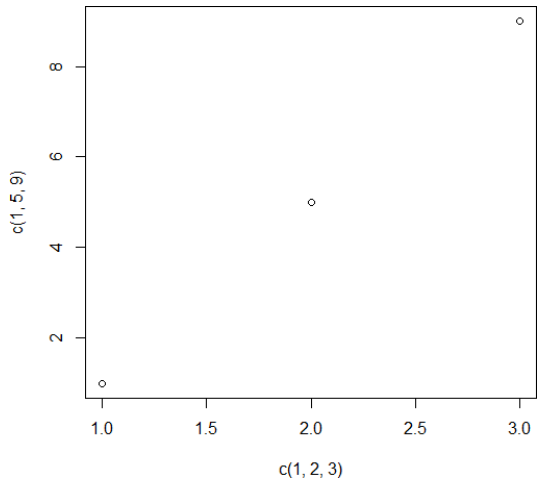
a title for the y axis:

`asp`

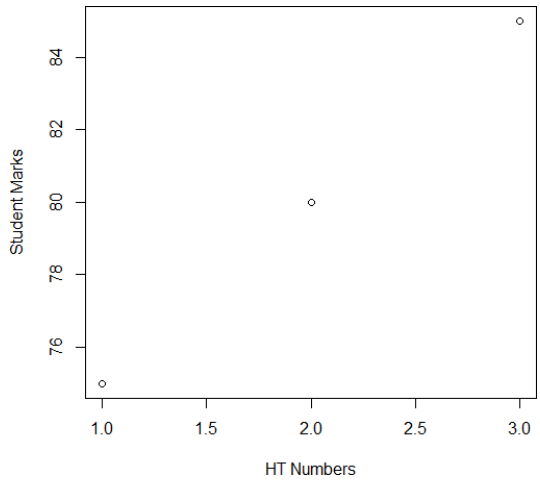
the *y/x* aspect ratio.

Example:

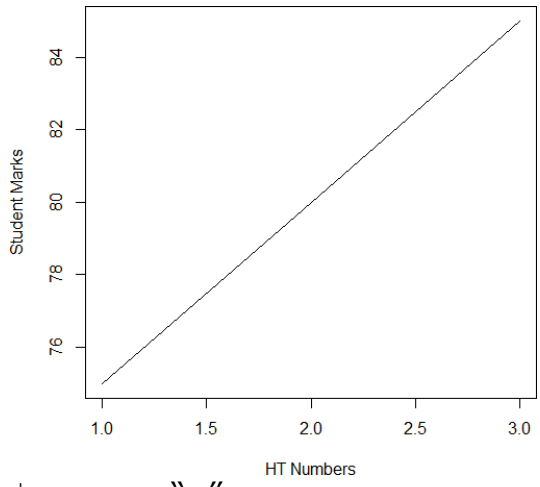
```
>plot(c(1,2,3),c(1,5,9))
```



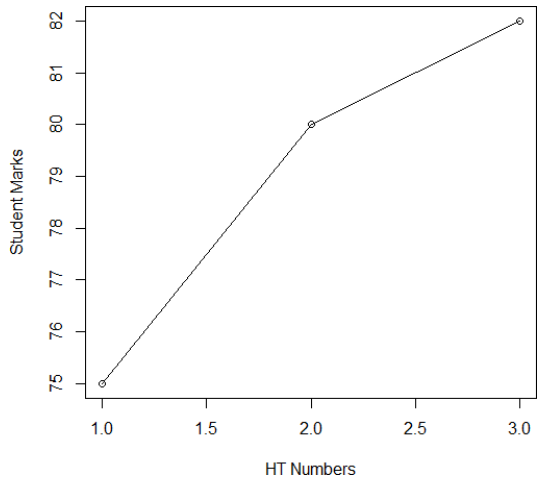

```
>#  Changing X axis and Y axis Names
>plot(c(1,2,3),c(75,80,85),xlab="HT Numbers",ylab="Student
Marks")
```



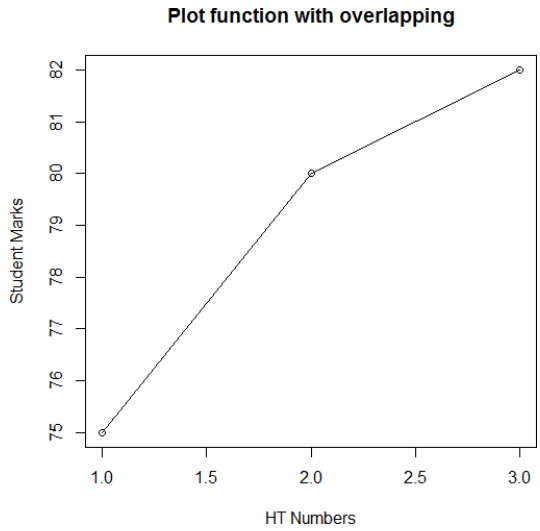
```
># Changing plot type as “l”
>plot(c(1,2,3),c(75,80,85),xlab="HT Numbers",ylab="Student
Marks", type=”l”)
```




```
># Changing plot type as “o”
>plot(c(1,2,3),c(75,80,82),xlab="HT Numbers",ylab="Student
Marks", type=”o”)
```

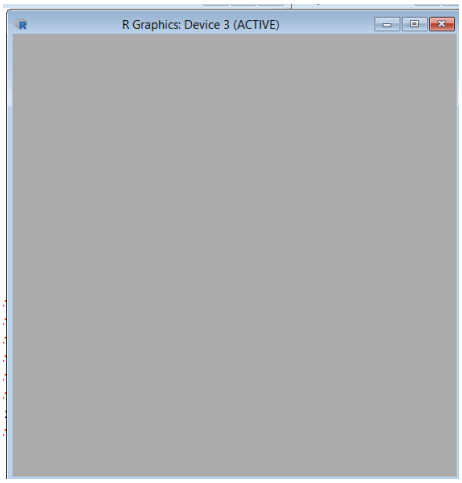


```
>#  Changing title of the page- main

>plot(c(1,2,3),c(75,80,82),xlab="HT Numbers",ylab="Student
Marks", type=”o”, main=” Plot function with overlapping”)
```



B. Starting New Graph- x11() function

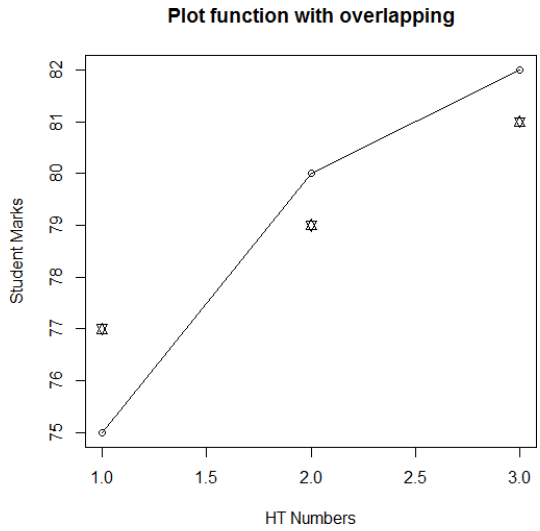
Each time when we call plot(), directly or indirectly, the current graph window will be replaced by the new one. To create new window or new graph just use the x11() as follows
>x11()



C. points() function

The points() function adds a set of (*x*,*y*) points, with labels for each, to the currently displayed graph.
Example:

```
>points(c(1,2,3),c(77,79,81),pch=11)
```



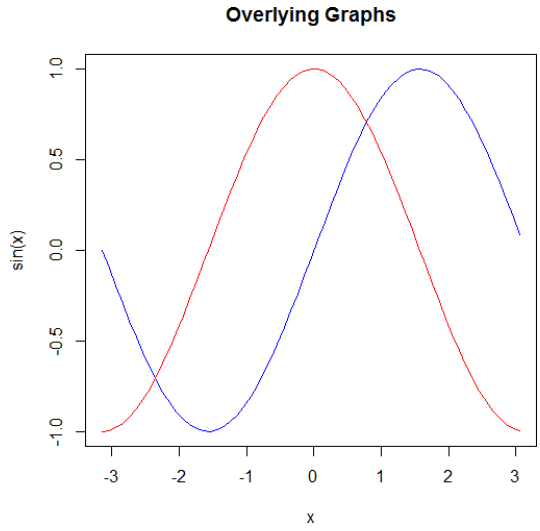
Here pch means Plot Character. pch=11 means, the character ✳ will be printed in plot diagram based on x and y axis coordinates.

D. legend() function

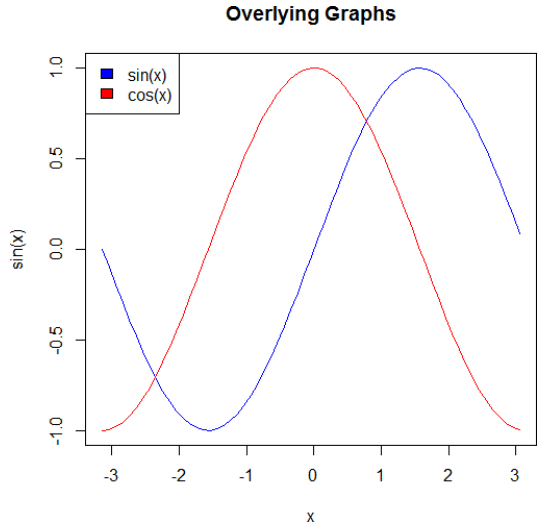
The legend() function is used to add extra information to the plot.
The following plot has two curved lines in the plot diagram. sin(x) and cos(x) depicted as blue and red colors in the plot diagram. The legend function gives an explanation about curved lines in the rectangle box with respective colors.

Example:

```
>x<-seq(-pi,pi,0.1)
>plot(x,sin(x), main= “Overlying Graphs”, type= “l”, col=“blue”)
>lines(x,cos(x), col=“red”)
```



>legend("topleft", c("sin(x)", "cos(x)"), fill=c("blue", "red"))

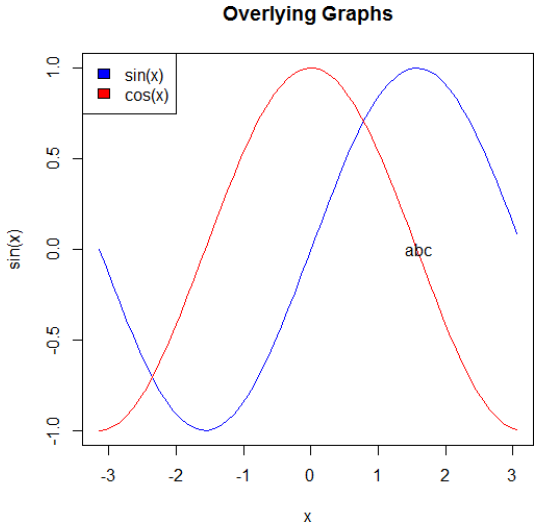


E. text() function

The text() function is used to place some text anywhere in the current graph.
Example:

```
>text(1.6, 0, "abc")
```

This writes the text “abc” at the point (1.6, 0) in the graph. The center of the string, in this case “b,” would go at that point.



F. locator() function

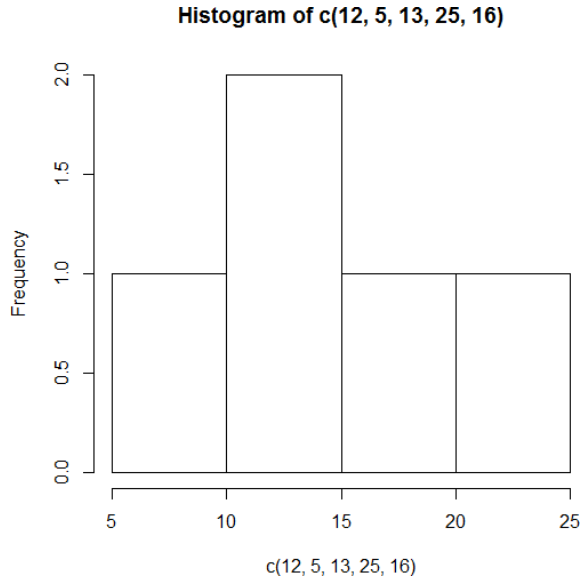
The locator() function retrieves the *x*- and *y*-coordinates where mouse pointer is selected in the graph. Simply call the function locator(1)and then click the mouse at the desired spot in the graph. The function returns the *x* and *y*-coordinates of your click point. Specifically, typing the following will tell R that you will click in one place in the graph:

```
>locator(1)
```

Once we click, R will tell us the exact coordinates of the point mouse clicked.
Call locator(2) to get the locations of two places, and so on.

Example:

```
> hist(c(12,5,13,25,16))
```




```
> locator(1)
$x
[1] 6.239237
$y
[1] 1.221038
```

barplot() function

A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function **barplot()** to create bar charts. R can draw both vertical and horizontal bars in the bar chart. In bar chart each of the bars can be given different colors.

Syntax

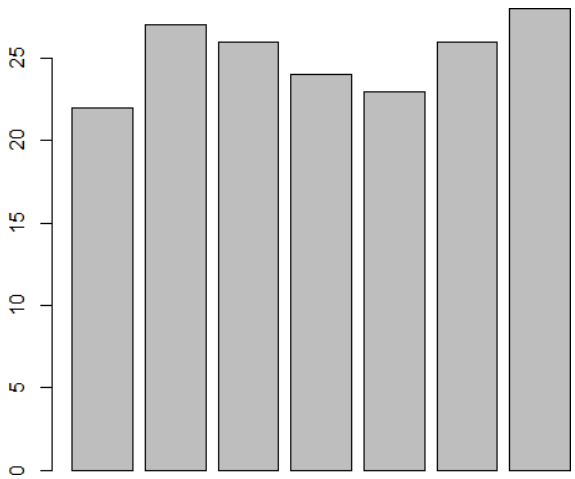
The basic syntax to create a bar-chart in R is:

```
>barplot(H,xlab,ylab,main, names.arg,col)
```

Following is the description of the parameters used:

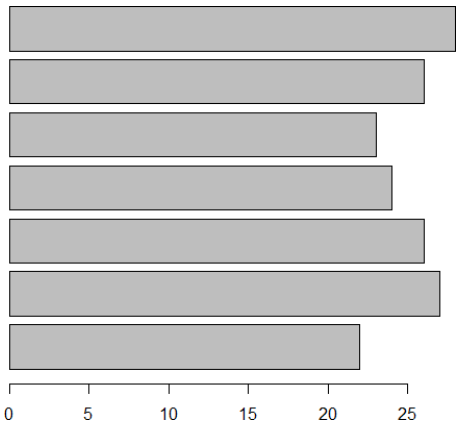
- **H** is a vector or matrix containing numeric values used in bar chart.
- **xlab** is the label for x axis.
- **ylab** is the label for y axis.
- **main** is the title of the bar chart.
- **names.arg** is a vector of names appearing under each bar.
- **col** is used to give colors to the bars in the graph.

```
> temp<-c(22,27,26,24,23,26,28)

> temp

[1] 22 27 26 24 23 26 28

> barplot(temp)
```

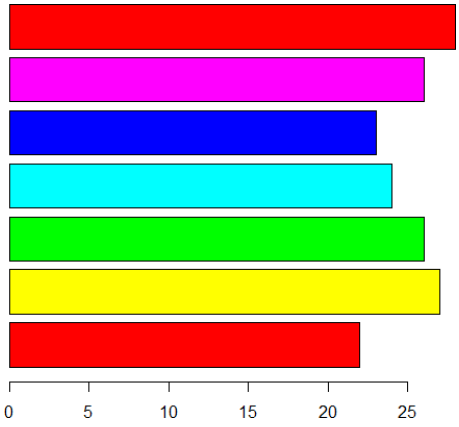


The bars can be pictured horizontally in the graph using the argument horiz=TRUE

> barplot(temp,horiz=TRUE)



Adding colors to the bars through rainbow() function
>barplot(temp,horiz=TRUE,col=rainbow(6))



boxplot() function

Boxplots are a measure of how well distributed is the data in a data set.
It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set.
It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.

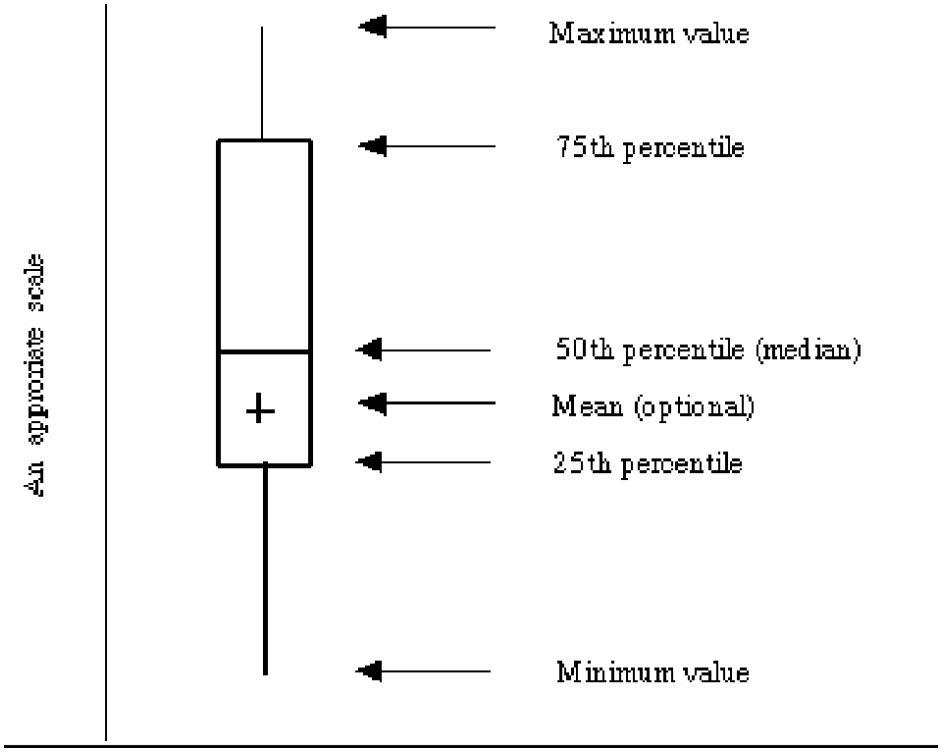
Boxplots are created in R by using the **boxplot()** function.

Syntax
The basic syntax to create a boxplot in R is :

boxplot(x,data,notch,varwidth,names,main)

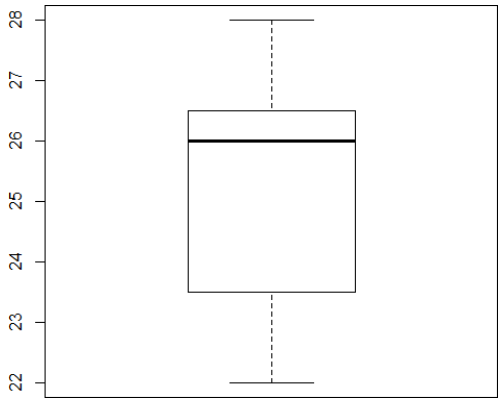
Following is the description of the parameters used:

- **x** is a vector or a formula.
- **data** is the data frame.
- **notch** is a logical value. Set as TRUE to draw a notch.
- **varwidth** is a logical value. Set as true to draw width of the box proportionate to
- the sample size.
- **names** are the group labels which will be printed under each boxplot.
- **main** is used to give a title to the graph.



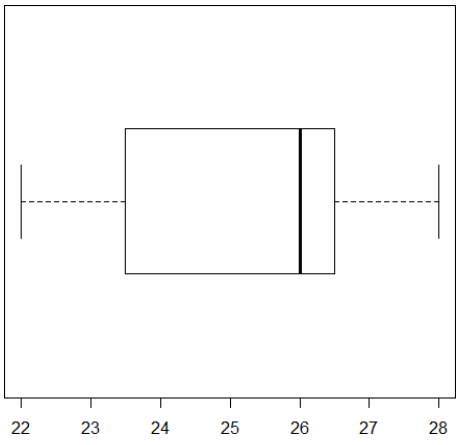
Example

```
> temp<-c(22,27,26,24,23,26,28)
> temp
[1] 22 27 26 24 23 26 28
> boxplot(temp)
```



Placing boxplot as horizontal

```
> boxplot(temp, horizontal=TRUE)
```



Histogram - hist() function

A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.
R creates histogram using **hist**() function. This function takes a vector as an input and uses some more parameters to plot histograms.

Syntax
The basic syntax for creating a histogram using R is:

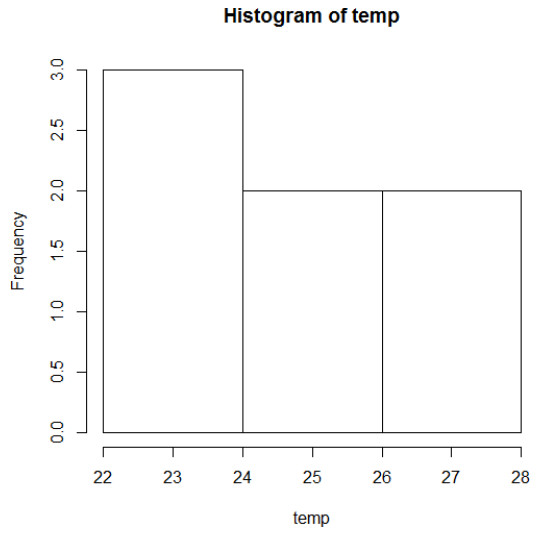
```
hist(v,main,xlab,xlim,ylim,breaks,col,border)
```

Following is the description of the parameters used:

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

Example

```
> temp<-c(22,27,26,24,23,26,28)
> temp
[1] 22 27 26 24 23 26 28
> hist(temp)
>
```



PIE-CHART : pie() function

R Programming language has numerous libraries to create charts and graphs. A pie-chart is a representation of values as slices of a circle with different colors. The slices are labeled and the numbers corresponding to each slice is also represented in the chart.

In R the pie chart is created using the **pie()** function which takes positive numbers as a vector input. The additional parameters are used to control labels, color, title etc.

Syntax

The basic syntax for creating a pie-chart using the R is:

```
pie(x, labels, radius, main, col, clockwise)
```

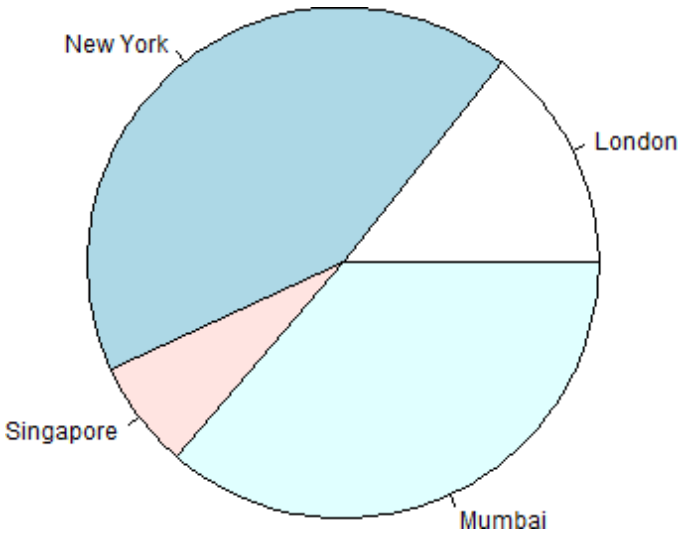
Following is the description of the parameters used:

- **x** is a vector containing the numeric values used in the pie chart.
- **labels** is used to give description to the slices.
- **radius** indicates the radius of the circle of the pie chart.(value between -1 and +1).
- **main** indicates the title of the chart.
- **col** indicates the color palette.
- **clockwise** is a logical value indicating if the slices are drawn clockwise or anti clockwise.

Example

A very simple pie-chart is created using just the input vector and labels. The below script will create and save the pie chart in the current R working directory.

```
# Create data for the graph.
x <- c(21, 62, 10, 53)
labels <- c("London", "New York", "Singapore", "Mumbai")

# Give the chart file a name.
png(file = "city.jpg")

# Plot the chart.
pie(x,labels)

# Save the file.
dev.off()
```



Strip- Chart : stripchart() function

stripchart produces one dimensional scatter plots (or dot plots) of the given data. These plots are a good alternative to boxplots when sample sizes are small.

Usage

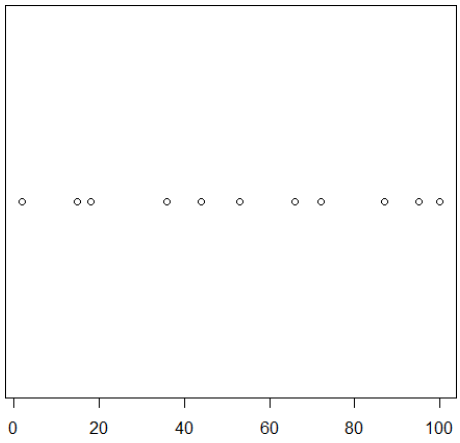
stripchart(x, ...)

Arguments

x the data from which the plots are to be produced

Example

```
>temp<-c(2,100,95,15,72,87,66,44,53,36,18)
> stripchart(temp,pch=21)
```

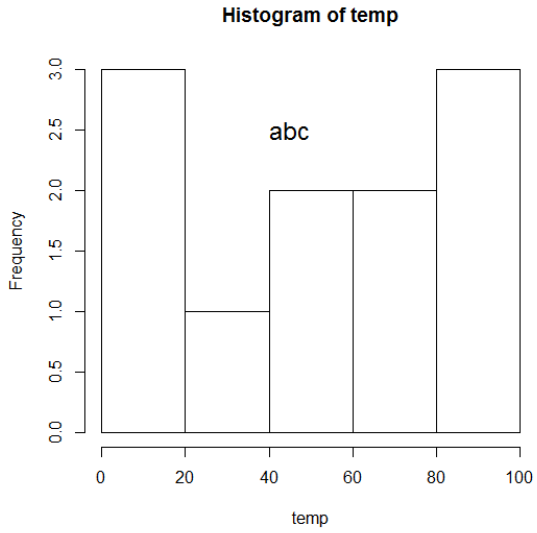


Customizing Graphs

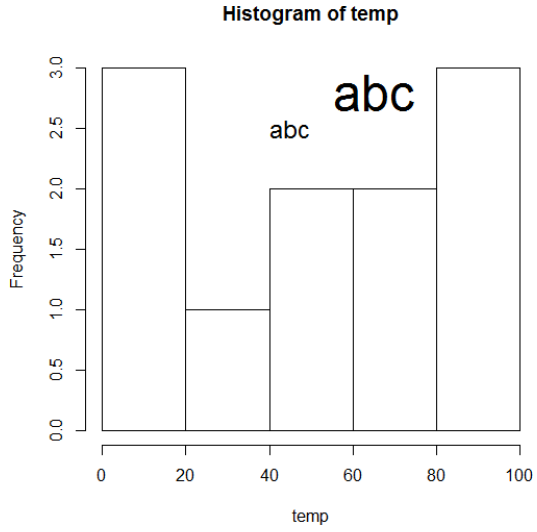
Changing Character Sizes: The cex Option

The cex (for *character expand*) function allows you to expand or shrink characters within a graph, which can be very useful. For instance, you may wish to draw the text "abc" at some point, say (2.5,4), in your graph but with a larger font, in order to call attention to this particular text. You could do this by typing the following:

```
>text(2.5,4,"abc",cex = 1.5)
```




```
>text(65,2.8,"abc",cex = 3.0)
```



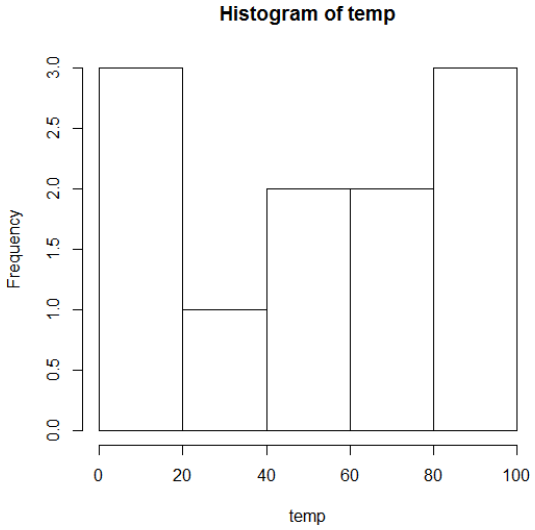
This prints the same text as in our earlier example but with characters 1.5 times the normal size.

Changing the Range of Axes: The xlim and ylim Options

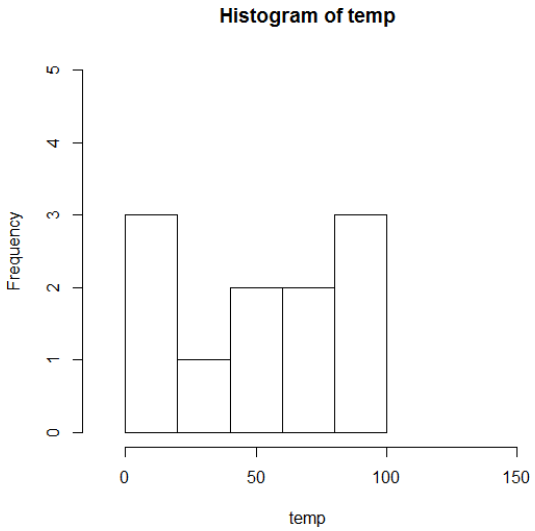
Xlim and ylim is used to set the ranges of x- and y- axis of plot.

Example:

Before using xlim and ylim
>hist(temp)



After using xlim and ylim
> hist(temp, xlim=c(-10,150),ylim=c(0,5))



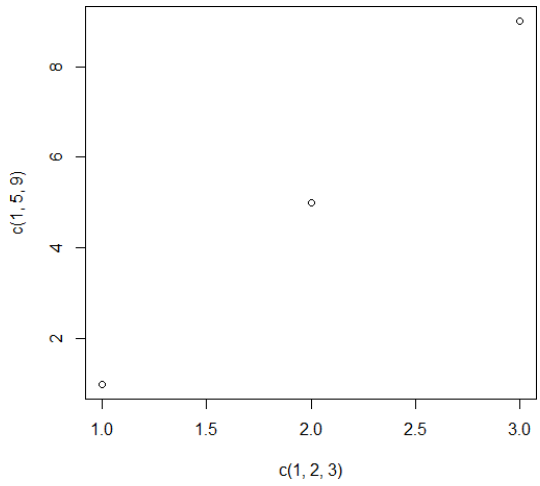
12.2.3 Adding a Polygon: The polygon() Function

We can use polygon() to draw arbitrary polygonal objects.
Syntax
>polygon(x,y,…)

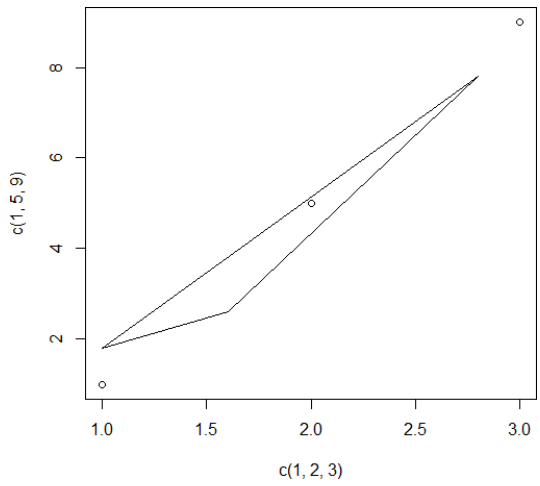
X – x axis coordinates
Y – y axis coordinates

For example, the following code draws the polygon in the plot diagram

Before Polygon()
>plot(c(1,2,3),c(1,5,9))



>polygon(c(1,1.6,2.8),c(1.8,2.6,7.8))



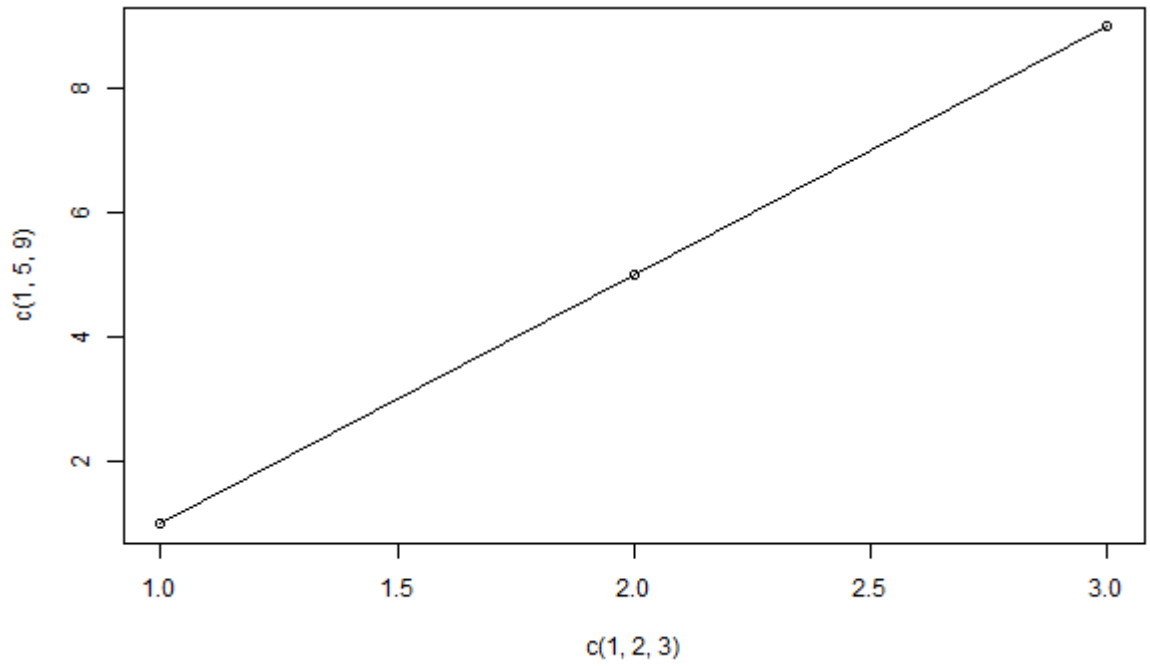
Saving Graphs to Files

The R graphics display can consist of various graphics devices. The default device is the screen. If you want to save a graph to a file, you must set up another device.

By default, all the graphs are displayed on the screen. We can save the graphs with special functions in R.

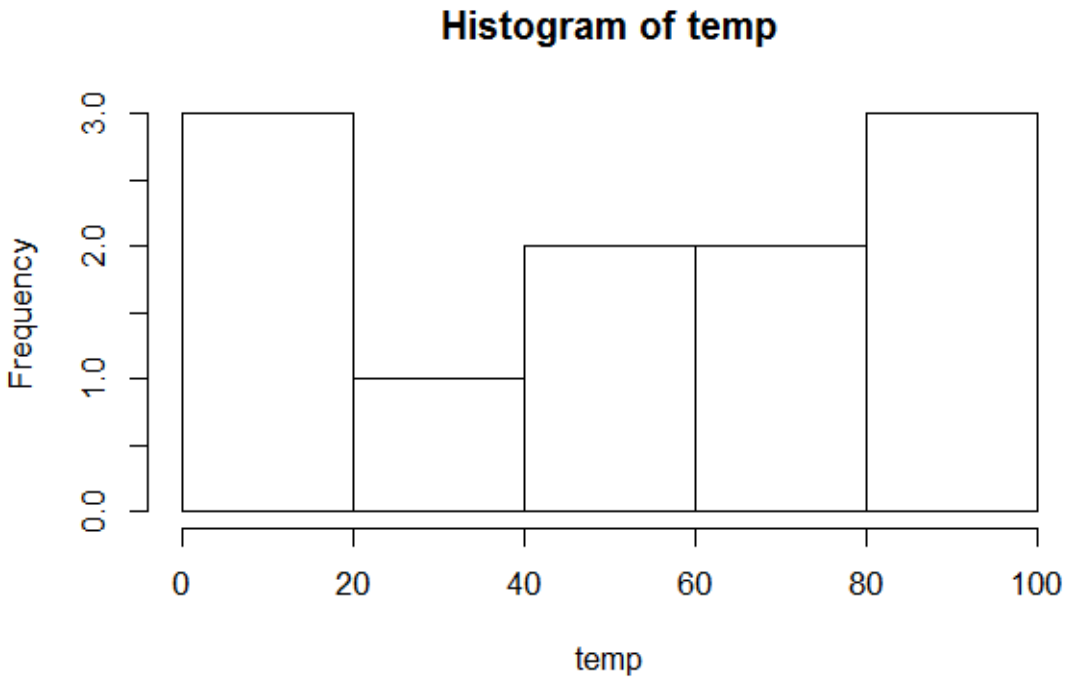
1. To save graph as .png format

```
> png(file="plot1.png", width=600, height=400)
> plot(c(1,2,3),c(1,5,9), type="o")
> dev.off()
null device
          1
>
```



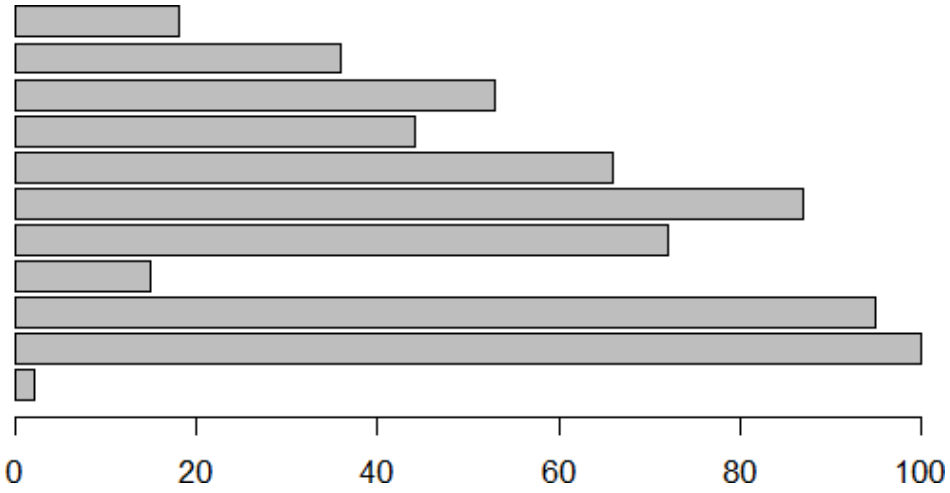
2. To save graph as .bmp format

```
> bmp(file="hist1.bmp", width=6, height=4, units="in",
res=100)
> hist(temp)
> dev.off()
null device
          1
>
```



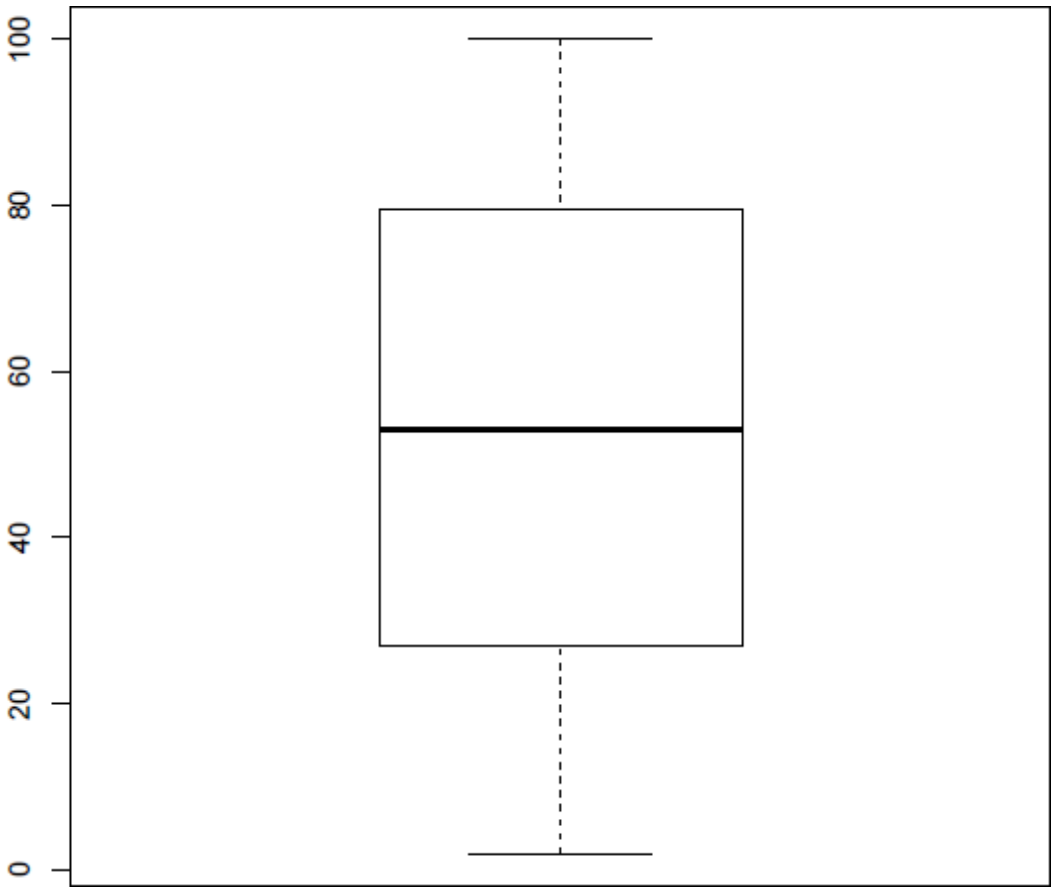
3. To save graph as .tiff format

```
> tiff(file="barplot1.tiff", width=6, height=4, units="in",
res=100)
> barplot(temp,horiz=TRUE)
> dev.off()
tiff
   2
>
```



4. To save graph as .pdf format

```
> pdf(file="boxplot.pdf")
> boxplot(temp)
> dev.off()
tiff
   2
```



R – Colors

Colors in R can be specified in three ways

1. Specifying through the names
2. Specifying through the Hexa Decimal values
3. Specifying through the rainbow() function

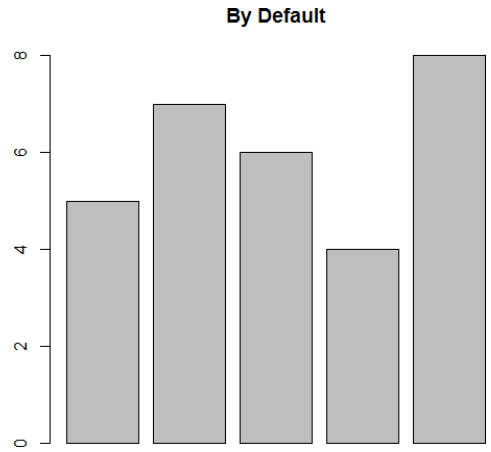
1. Specifying through the names

We can specify the color through color name. R has 657 colors to specify color name
We can get the color names through colors() function

> colors()

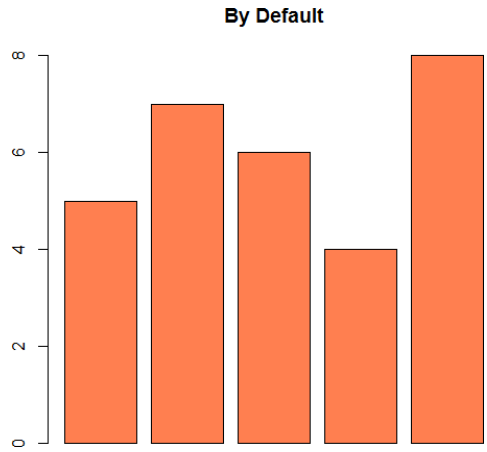
```
 [1] "white"           "aliceblue"       "antiquewhite"
 [4] "antiquewhite1"   "antiquewhite2"   "antiquewhite3"
 [7] "antiquewhite4"   "aquamarine"      "aquamarine1"
[10] "aquamarine2"     "aquamarine3"     "aquamarine4"
[13] "azure"           "azure1"          "azure2"
[16] "azure3"          "azure4"          "beige"
[19] "bisque"          "bisque1"         "bisque2"
[22] "bisque3"         "bisque4"         "black"
```

```
> temp<-c(5,7,6,4,8)
> barplot(temp,main="By Default")
```



Specifying color through col option in barplot() function. Here color name is “coral”

```
>barplot(temp,main="By Default", col="coral")
```

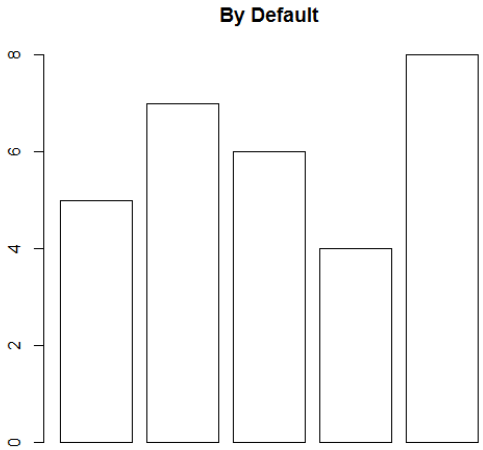


2. Specifying through the Hexa Decimal values

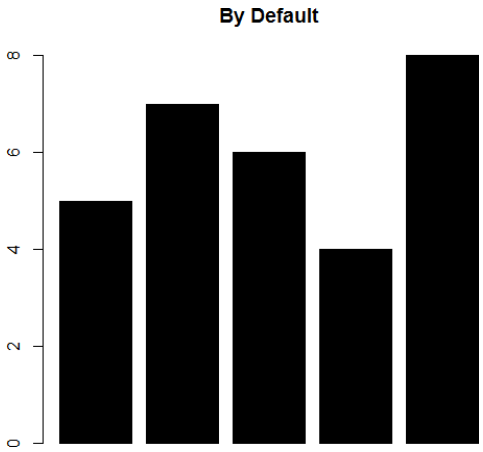
We can also specify the color name through hexadecimal value. The hexadecimal digit number has 6 numerical values followed by #. It is a six hexadecimal digit number of the form #RRGGBB. Here RR stands for RED, GG for GREEN and BB for BLUE.
The permissible range of value is from 00 to FF.
Sample hexadecimal codes for colors
Red = #FF0000
Green = #00FF00
Blue=#0000FF
Black=#000000
White=#FFFFFF

Example

```
> barplot(temp,main="By Default", col="#FFFFFF") #White
```




```
> barplot(temp,main="By Default", col="#000000") # Black
```



3. Specifying through the Color Palette

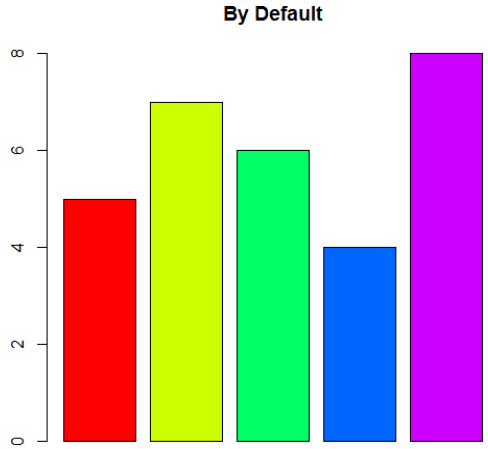
We can apply colors to the diagram through Color Palette
Color Palette has five built- in functions. They are

> rainbow() function
> terrain.colors()
> cm.colors()
> heat.colors()
> colors()

Example

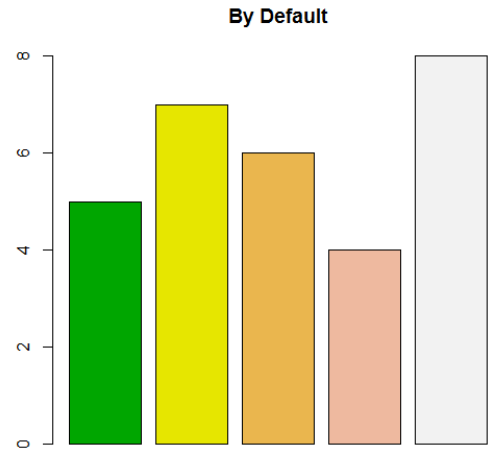
1. rainbow() function

```
>barplot(temp,main="By Default", col=rainbow(5))
```



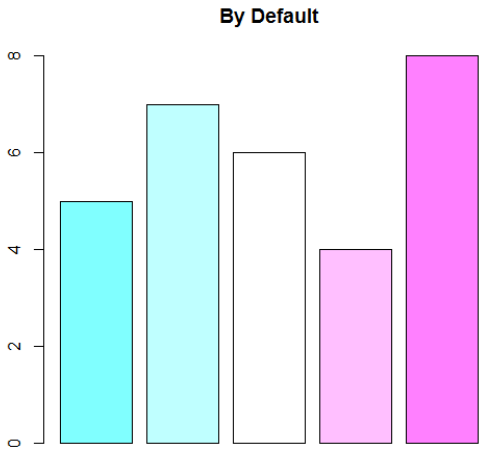
2. terrain.colors()

```
>barplot(temp,main="By Default",
col=terrain.colors(5))
```



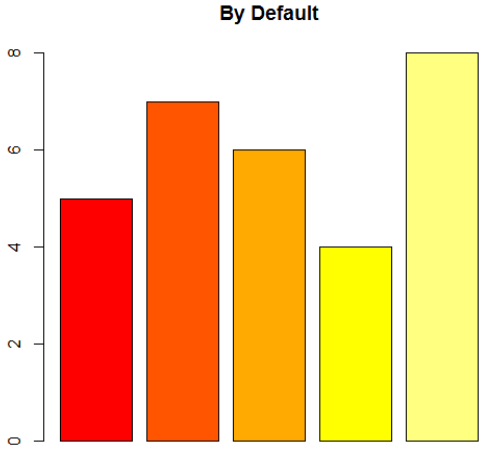
3. cm.colors()

```
>barplot(temp,main="By Default", col= cm.colors
(5))
```



4. heat.colors()

```
>barplot(temp,main="By Default",
col=heat.colors(5))
```



5. colors()

```
>barplot(temp,main="By Default", col=colors(5))
```