DATA VISUALIZATION

(USING R LANGUAGE)

DATA VISUALIZATION

Data visualization is a general term that describes any effort to help people understand the significance of data by placing it in a visual context. **Patterns, trends and correlations** that might go undetected in text-based data can be exposed and recognized easier with data visualization.

Why visualize data?

- Explore: Visualization helps in exploring and explaining patterns and trends.
- Detect: Patterns and anomalies in data can be detected by looking at graphs.
- Make sense: Possible to make sense of amount of data efficiently and in time.
- Communicate: Easy to communicate and share insights from the data.

DATA EXPLORATION

Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a data set, including its size, accuracy, initial patterns in the data and other attributes. It is commonly conducted by data analysts using visual analytics tools, but it can also be done in more advanced statistical software, such as **R**.

DATA EXPLORATION: Outcomes

- How many cases are in the dataset.
- What variables are included.
- Relationship between different variables.
- How many missing values are there.
- What general hypothesis the data is likely to support.

etc…

Three main packages in R for visualization

- Graphics: Fundamental package for visualizing data used to create all basic plots.
- ggplot2: Based on Grammar of Graphics. Structured approach to data visualization and builds upon the features available on Graphics and Lattice packages.
- Lattice: Data visualization system with an emphasis on multivariate data.

Univariate Visualization

Univariate is a term commonly used in statistics to describe a type of data which consists of observations on only a single characteristic or attribute.

- Line charts
- Bar charts
- Stacked bar charts
- Scatter plots
- Percentages
- Candle chart

Bi and multivariate Visualization

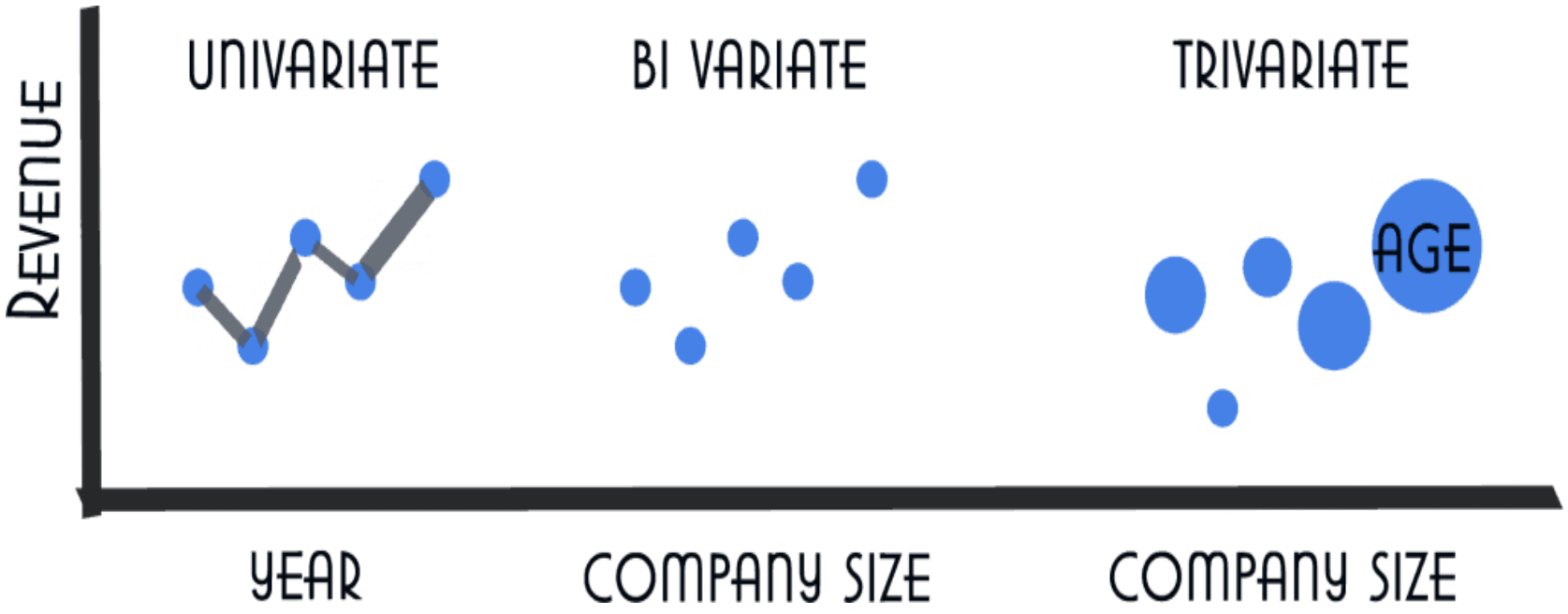
Bivariate involves the analysis of two variables (often denoted as *X*, *Y*), for the purpose of determining the empirical relationship between them.

- Area chart
- Histogram

Multivariate involves observation and analysis of more than one statistical outcome variable at a time. In design and analysis, the technique is used to perform trade studies across multiple dimensions while taking into account the effects of all variables on the responses of interest.

- Bubble chart
- Geo Map combined with one of the above
- Tree Map

Example:


UNIVARIATE
BI VARIATE
TRIVARIATE
REVENUE
AGE
YEAR
COMPANY SIZE
COMPANY SIZE

ggplot2

Histogram

- R creates histogram using **hist()** function which takes a vector as an input and uses some more parameters

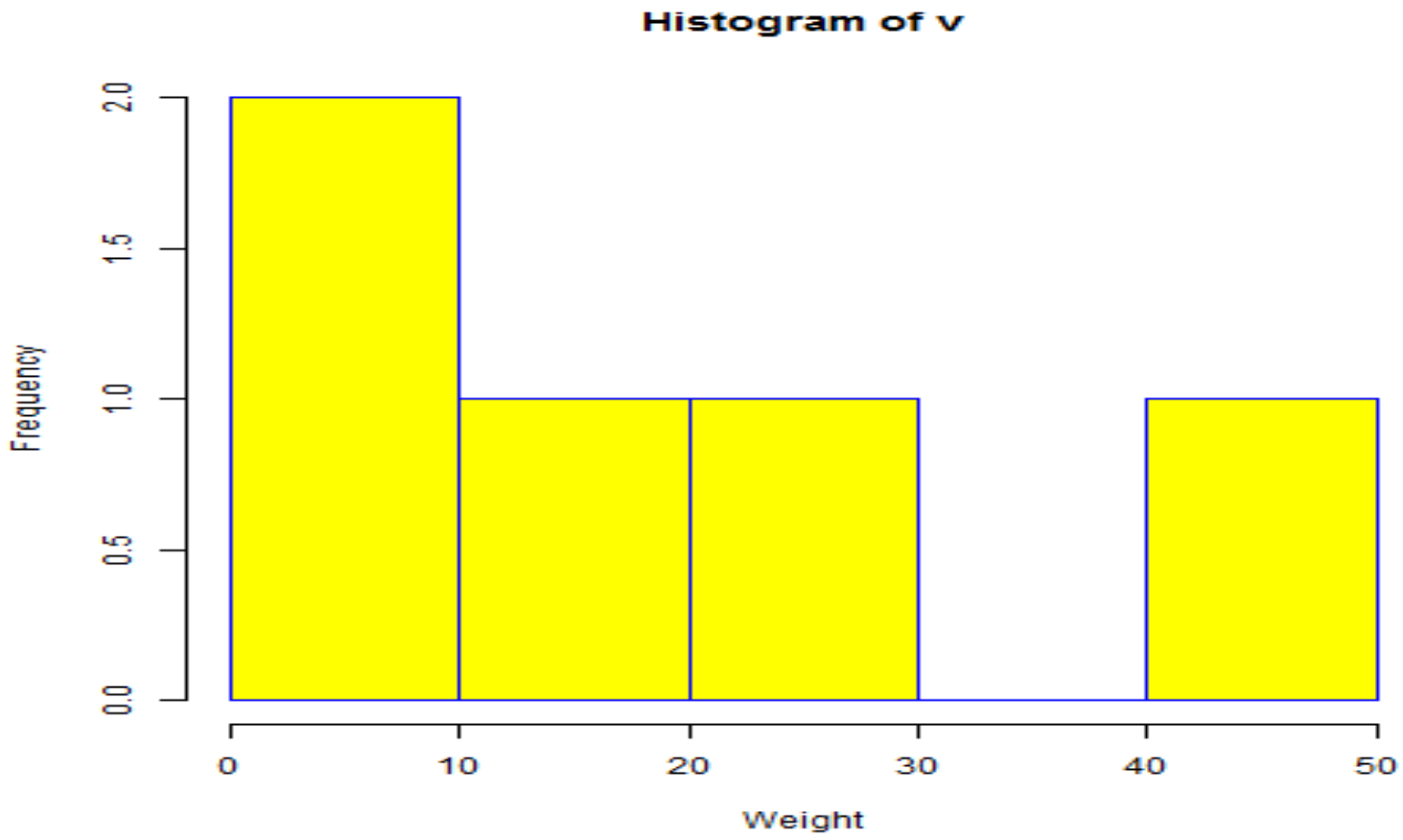
Syntax:

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

- **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

```
> v <- c(9,13,21,8,36,22,12,41,31,33,19)  # Create data for the graph.
> png(file = "histogram.png")             # Give the chart file a name.
> hist(v,xlab = "Weight",col = "yellow",border = "blue") # Create the histogram.
> dev.off()                               # Save the file.
```



Boxplot

- It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.
- Created in R by using **boxplot()** function.

Syntax: **boxplot**(x, data, notch, varwidth, names, main)

- **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.

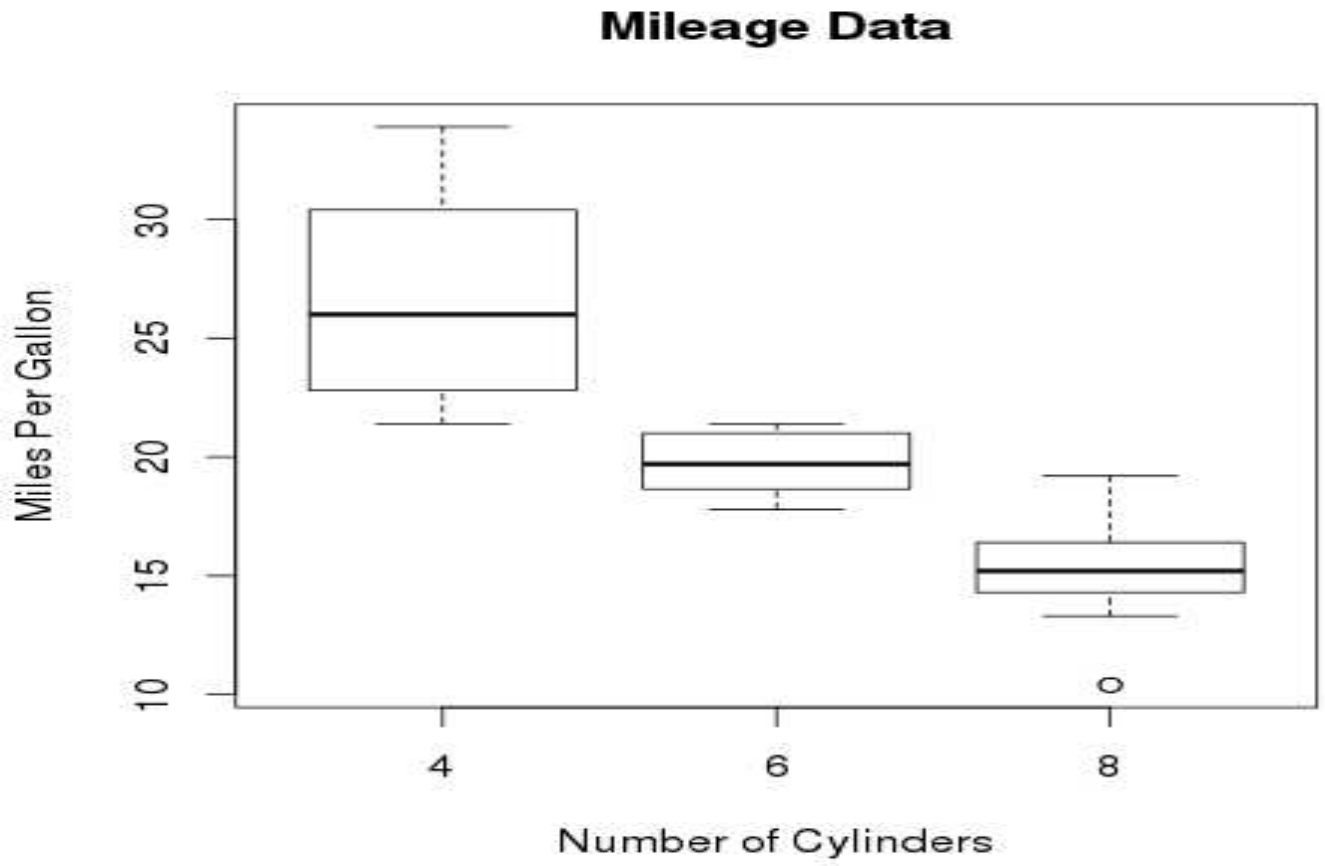
Example:

- Suppose we are using the dataset **mtcars** available in R environment which is,

	mpg	cyl
Mazda RX4	21.0	6
Mazda RX4 Wag	21.0	6
Datsun 710	22.8	4
Hornet 4 Drive	21.4	6
Hornet Sportabout	18.7	8
Valiant	18.1	6

Contd.

```
> png(file = "boxplot.png")
> boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders", ylab = "Miles Per Gallon", main = "Mileage Data")  # Plot the chart.
> dev.off()  # Save the file.
```



Pie charts

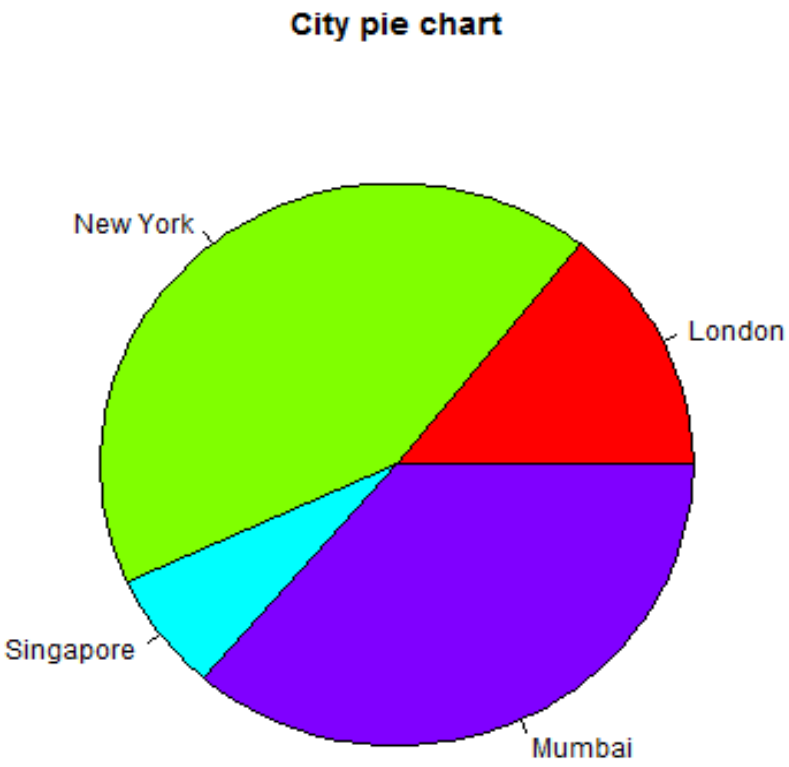
- A representation of values as slices of a circle with different colours.
- Also done with **pie()** function.

Syntax: **pie**(x, labels, radius, main, col, clockwise)

- **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

```
> x <- c(21, 62, 10, 53)
> labels <- c("London", "New York", "Singapore", "Mumbai") # Create data for the graph.
> png(file = "city_title_colours.jpg")  # Give the chart file a name.
> pie(x, labels, main = "City pie chart", col = rainbow(length(x))) # Plot the chart with title and rainbow colour pallet.
> dev.off() # Save the file.
```



Bar charts

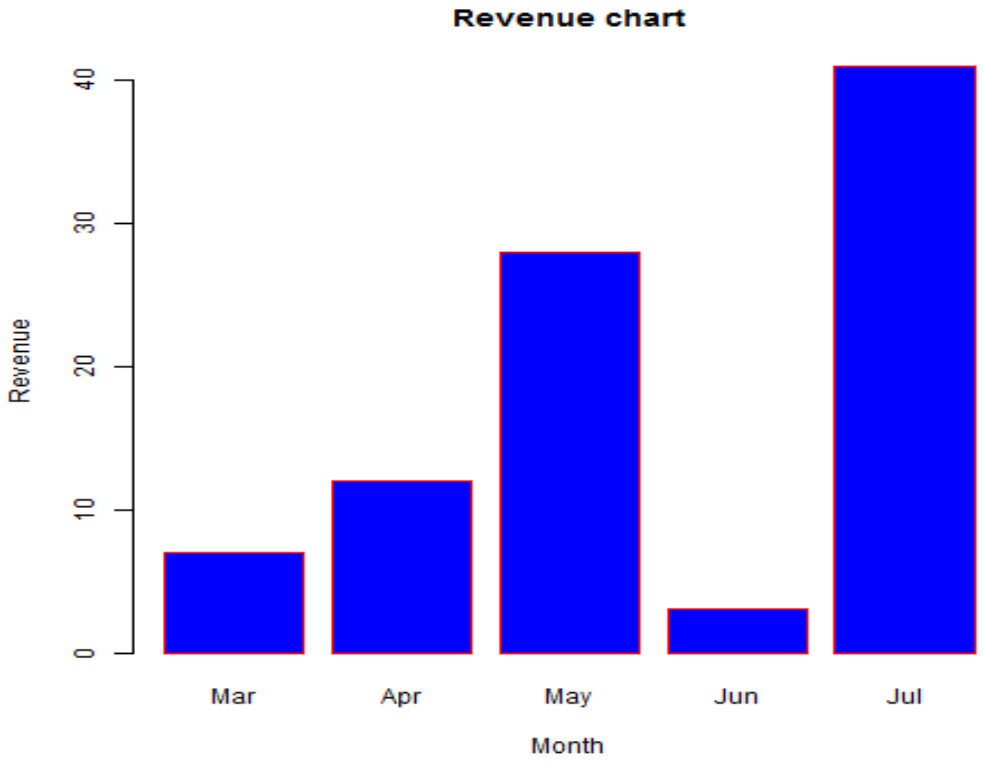
- Represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function **barplot().**

Syntax: **barplot**(H,xlab,ylab,main, names.arg,col)

- **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.

Example

```
> H <- c(7,12,28,3,41)
> M <- c("Mar","Apr","May","Jun","Jul") # Create the data for the chart
> png(file = "barchart_months_revenue.png") # Give the chart file a name
> barplot(H,names.arg=M,xlab="Month",ylab="Revenue",col="blue",
main="Revenue chart",border="red") # Plot the bar chart
> dev.off()      # Save the file
```



Line Charts

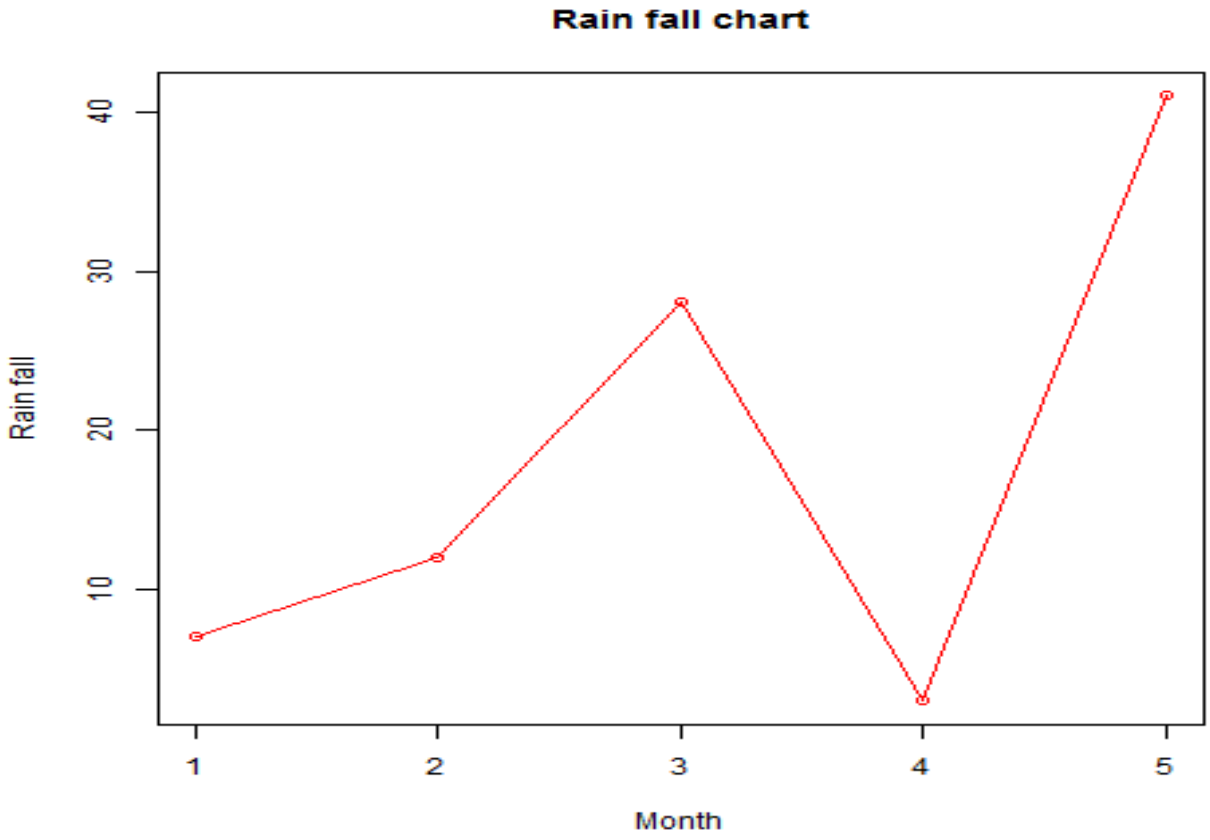
- A line chart is a graph that connects a series of points by drawing line segments between them.
- The **plot()** function in R is used.

Syntax: **plot**(v, type,col,xlab,ylab)

- **v** is a vector containing the numeric values.
- **type** takes the value "p" to draw only the points, "l" to draw only the lines and "o" to draw both points and lines.
- **xlab** is the label for x axis.
- **ylab** is the label for y axis.
- **main** is the Title of the chart.
- **col** is used to give colours to both the points and lines.

Example

```
> v <- c(7,12,28,3,41) # Create the data for the chart.
> png(file = "line_chart_label_colored.jpg") # Give the chart file a name.
> plot(v,type = "o", col = "red", xlab = "Month", ylab = "Rain fall", main =
"Rain fall chart")    # Plot the bar chart.
> dev.off()    # Save the file.
```



Scatter plots

- Scatterplots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.

Syntax: **plot**(x, y, main, xlab, ylab, xlim, ylim, axes)

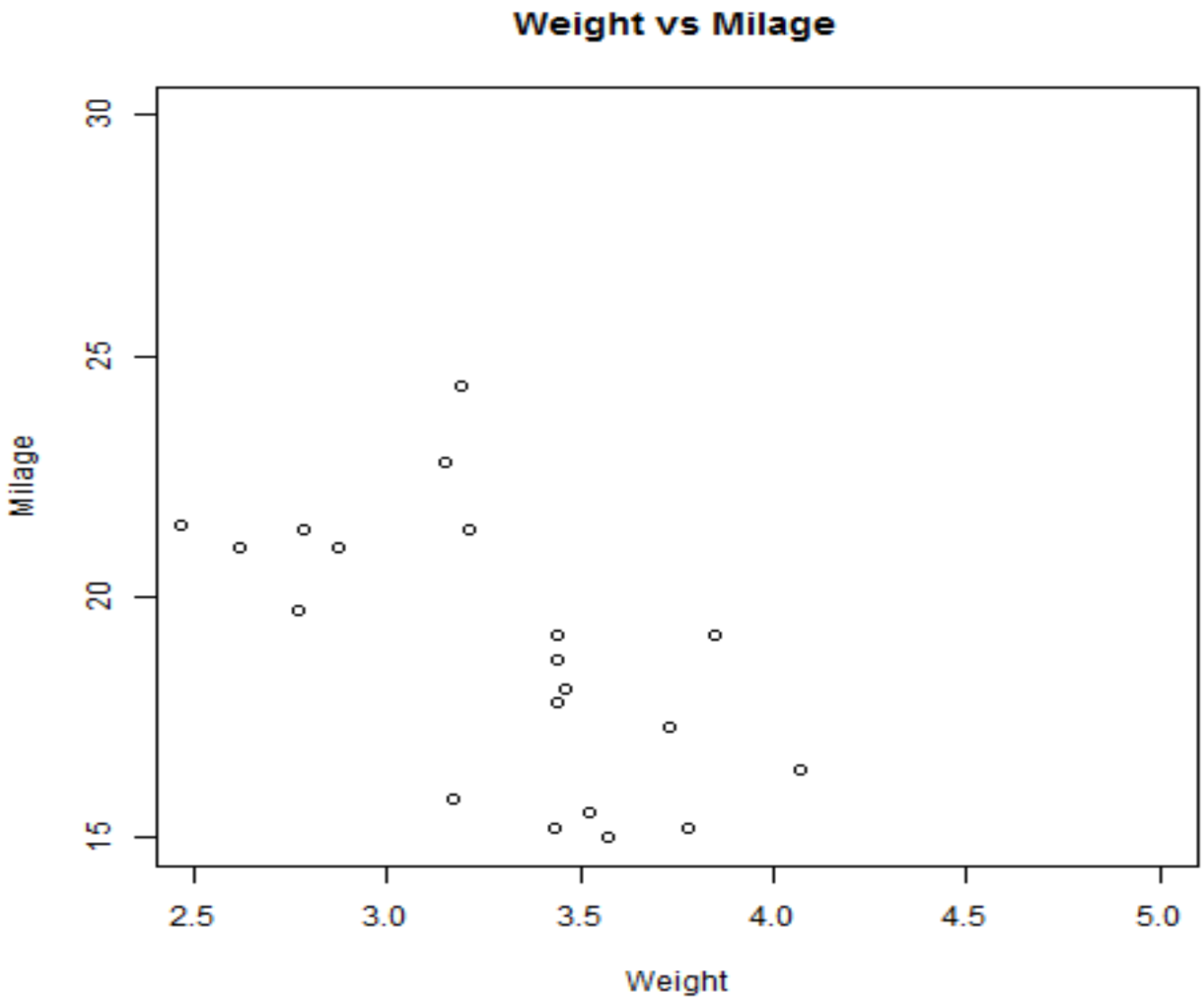
- **x** is the data set whose values are the horizontal coordinates.
- **y** is the data set whose values are the vertical coordinates.
- **main** is the tile of the graph.
- **xlab** is the label in the horizontal axis.
- **ylab** is the label in the vertical axis.
- **xlim** is the limits of the values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

Example

We are again using the **mtcars** data set form R environment.

```
> input <- mtcars[,c('wt','mpg')] # Get the input values.
> png(file = "scatterplot.png") # Give the chart file a name.
> plot(x = input$wt,
        y = input$mpg,
        xlab = "Weight",
         ylab = "Milage",
         xlim = c(2.5,5),
         ylim = c(15,30),
         main = "Weight vs Milage" )
 # Plot the chart for cars with weight between 2.5 to 5 and mileage between 15 and 30.
> dev.off() # Save the file.
```

Contd.



Radar chart

- **Spider** or **Web** or **Polar** charts. They are drawn in R using the **fmsb** library.

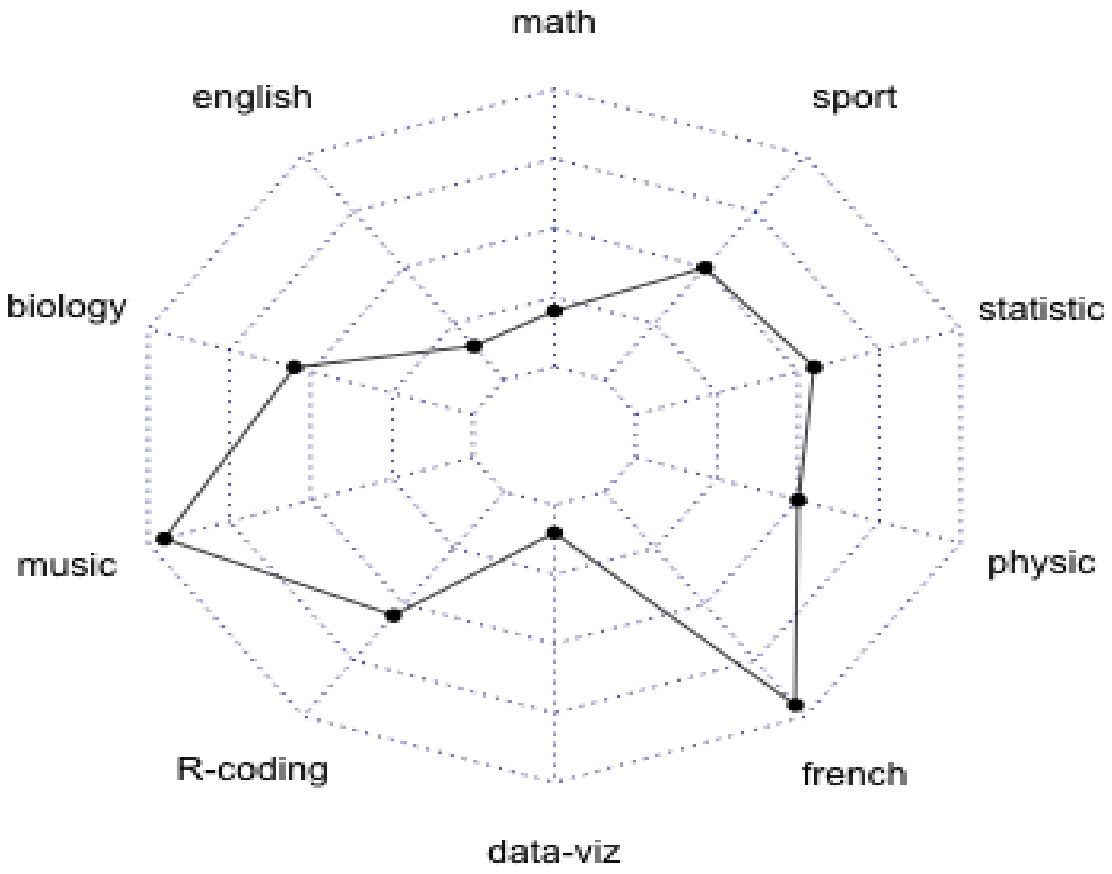
- Way of comparing multiple quantitative variables. This makes them useful for seeing which variables have similar values or if there are any outliers amongst each variable.

- Radar Charts are also useful for seeing which variables are scoring high or low within a dataset, making them ideal for displaying performance.

Example

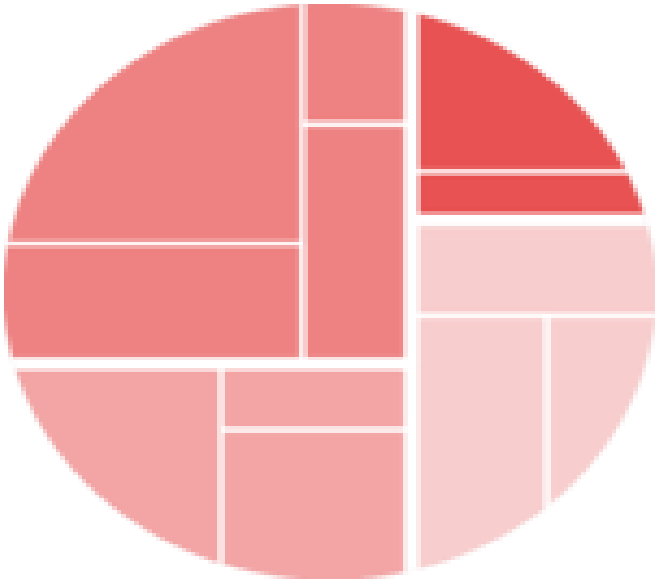
```
> library(fmsb)
> data=as.data.frame(matrix( sample( 2:20 , 10 , replace=T) , ncol=10)) # Create data
> colnames(data)=c("math" , "english" , "biology" , "music" , "R-coding", "data-viz" , "french" , "physic", "statistic", "sport" )

# To use the fmsb package, 2 lines have to added to the dataframe: the max and min of each topic to show on the plot.
> data=rbind(rep(20,10) , rep(0,10) , data)
# The default radar chart proposed by the library:
> radarchart(data)
```

Contd.

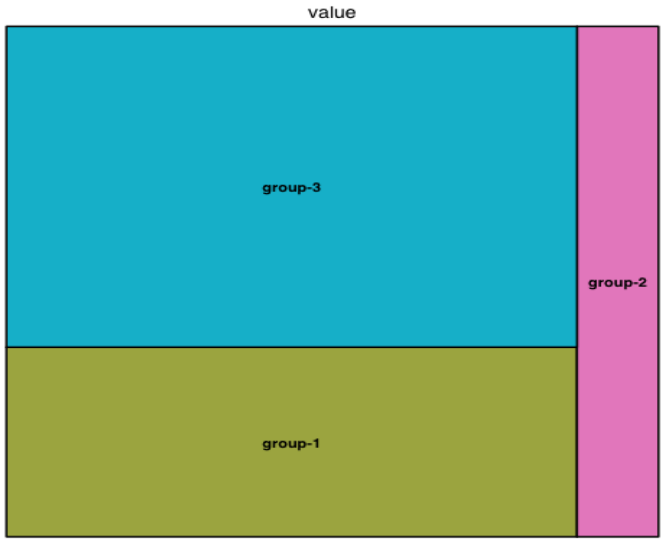

math
sport
statistic
physic
french
data-viz
R-coding
music
biology
english

Tree Map:

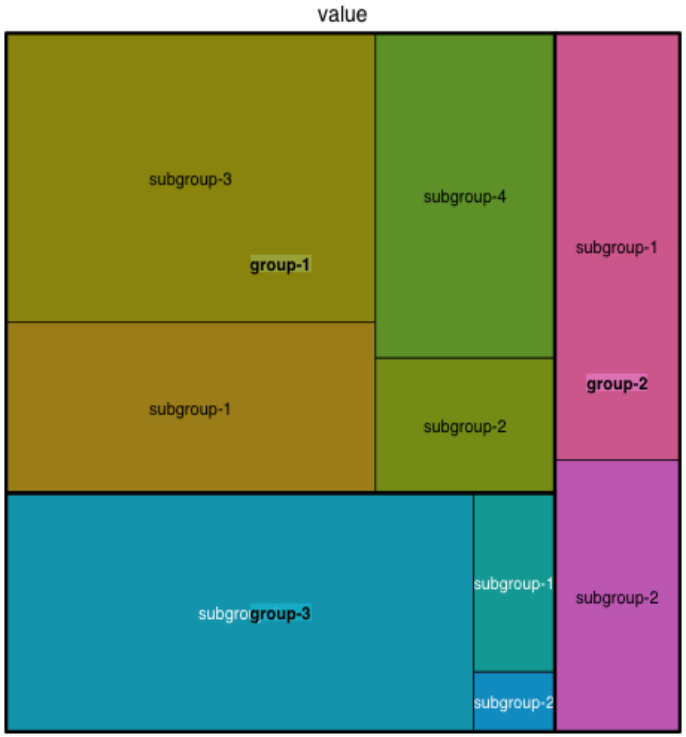
- It is an alternative way of visualizing the structure of a tree diagram.
- Treemaps display hierarchical data as a set of nested rectangles. Each branch of the tree is given a rectangle, which is then tiled with smaller rectangles representing sub-branches.
- A leaf node's rectangle has an area proportional to a specified dimension of the data.



- The most basic treemap.



- Treemap with subgroups



We can transform static treemaps in **interactive treemaps** using the **d3TreeR library**.