Practical 2 (Data importing and Visualization in R)

Base packages:

Packages that come with r natively like the datasets packages. You still have to load and unload packages manually. Use the command- library(datasets)

Other frequently used packages:

dplyr- for manipulating data frames

tidyr-for cleaning up information

Stringr-for working with strings or text information

Lubridate-for manipulating date information

Httr- for working with website data

- One package to load them all- Packman (stands for package manager). With pacman, specific packages can be unloaded separately. Command for installation of packages – install.packages("nameof the package")
- Eg.:run install.packages("pacman")
- This will make it available in the hardware but loading means actually making it accessible for use.
- Command for loading is library("nameof the package")
- Eg.: library("pacman")
-

Clear packages command- p_unload(dplyr, tidyr)
Or unload all using the command- p_unload(all) , this command unloads all the third party packages.

For base packages use command- detach("package:datasets",unload=TRUE)

To clear console, command- cat("\014")

Data Importing and using inbuilt data:

Inbuilt data:

Load datasets use command library(datasets). Use command head(datasets) to see the first few lines. At the beginning of your script, simply add library(readxl) to the list of libraries you are loading. And tidyverse package by using install.package(tidyverse) command.

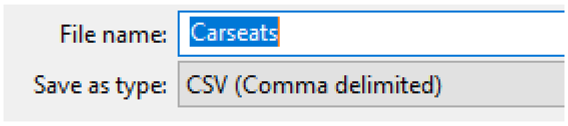
Data Importing

data = read.csv("nameof thefile", header = TRUE)

View(data)

dat<-read.csv(file="Carseats.csv",header=TRUE,sep=",",dec=".")

dat


File name: Carseats

Save as type: CSV (Comma delimited)

Data Cleaning and Preprocessing with R

Introduction

Data cleaning and preprocessing are crucial steps in the data analysis process. They involve identifying and rectifying errors, inconsistencies, and missing values in the dataset to ensure accurate and reliable results.

R, a popular programming language for statistical computing and data analysis, offers a wide range of tools and packages to effectively clean and preprocess data.

In this article, we will explore various techniques and methodologies in R for data cleaning and preprocessing.

Importance of Data Cleaning

Data cleaning is an essential step before conducting any analysis as it helps in improving data quality, reliability, and overall accuracy of the results. Unclean data may contain errors, outliers, or missing values, which can lead to biased or incorrect conclusions. Cleaning the data ensures that subsequent analyses are based on accurate and trustworthy information.

Common Data Cleaning Tasks

- **Handling Missing Data** − Missing data can significantly impact the analysis and interpretation of results. R provides functions like **is.na()** and **complete.cases()** to identify and handle missing values. Techniques such as imputation, where missing values are replaced with estimated values, can be performed using packages like **mice** or **missForest**.
- **Outlier Detection and Treatment** − Outliers are extreme values that deviate significantly from the rest of the data. R offers various methods, such as the use of **boxplots, z-scores**,

or the **Mahalanobis distance** to detect outliers. Once identified, outliers can be treated by removing them or transforming them to more reasonable values.

- **Removing Duplicates** − Duplicate records in a dataset can introduce bias and affect the integrity of the analysis. R provides functions like **duplicated()** and **distinct()** to identify and remove duplicates based on specific columns or combinations of columns.
- **Data Validation** − Validating the integrity and consistency of data is crucial. R offers validation techniques like **cross-tabulation, data profiling**, and **summary statistics** to ensure data accuracy.

For example:

```
data=data.frame(column1=c(1,2,NA,34),
                column2=c(NA,34,56,NA),
                column3=c(NA,NA,32,56))
# display
print(data)
# get NA values
print(is.na(data))
# count NA values
print(sum(is.na(data)))
# get the NA index positions
print(which(is.na(data)))

# Create a sample vector with duplicate elements
vector_data <- c(1, 2, 3, 4, 4, 5)

# Identify duplicate elements
duplicated(vector_data)

# count of duplicated data
sum(duplicated(vector_data))

# Create a sample vector with duplicate elements
vector_data <- c(1, 2, 3, 4, 4, 5)

# Remove duplicate elements
unique(vector_data)
```

Feature Selection − Feature selection aims to identify the most relevant variables for analysis. R offers techniques like correlation analysis, stepwise regression, or regularization methods (e.g., Lasso or Ridge regression) to select informative features and avoid overfitting.

Encoding Categorical Variables − Categorical variables often require encoding to numerical representations for analysis. R offers functions like **factor()** or **dummyVars()** to convert categorical variables into binary or numerical representations. This process enables the inclusion of categorical variables in statistical models.

R Packages for Data Cleaning and Preprocessing

Tidyverse − Tidyverse is a collection of R packages, including **dplyr, tidyr**, and **stringr**, that provide powerful tools for data manipulation, cleaning, and tidying. These packages offer a consistent and intuitive syntax for transforming and cleaning data. For example:

```
library(dplyr)
# Sample data frame
df <- data.frame(
  age = c(23, 25, 30, 22, 24, 29),
  name = c("Alice", "Bob", "Carol", "David", "Eve", "Frank"),
  score = c(85, 90, 88, 95, 92, 87))
filter(df,age>25)
#to find range
max(vector)-min(vector)
# to exclude NA values
print(max(data, na.rm=TRUE)-min(data, na.rm=TRUE))
```

Data visualisation

A picture is worth a thousand words. A good data scientist is able to communicate findings and persuade stakeholders through effective data visualisations.
Following built-in functions are available in R:

- **Scatter plot or line plot:** plot( )
- **Add graph on top of existing plot:** points( )
- **Draw straight lines on existing plot:** abline( )
- **Box plot:** boxplot( )
- **Histogram:** hist( )
- **Column graph:** barplot( )
- **Pie chart:** pie( )

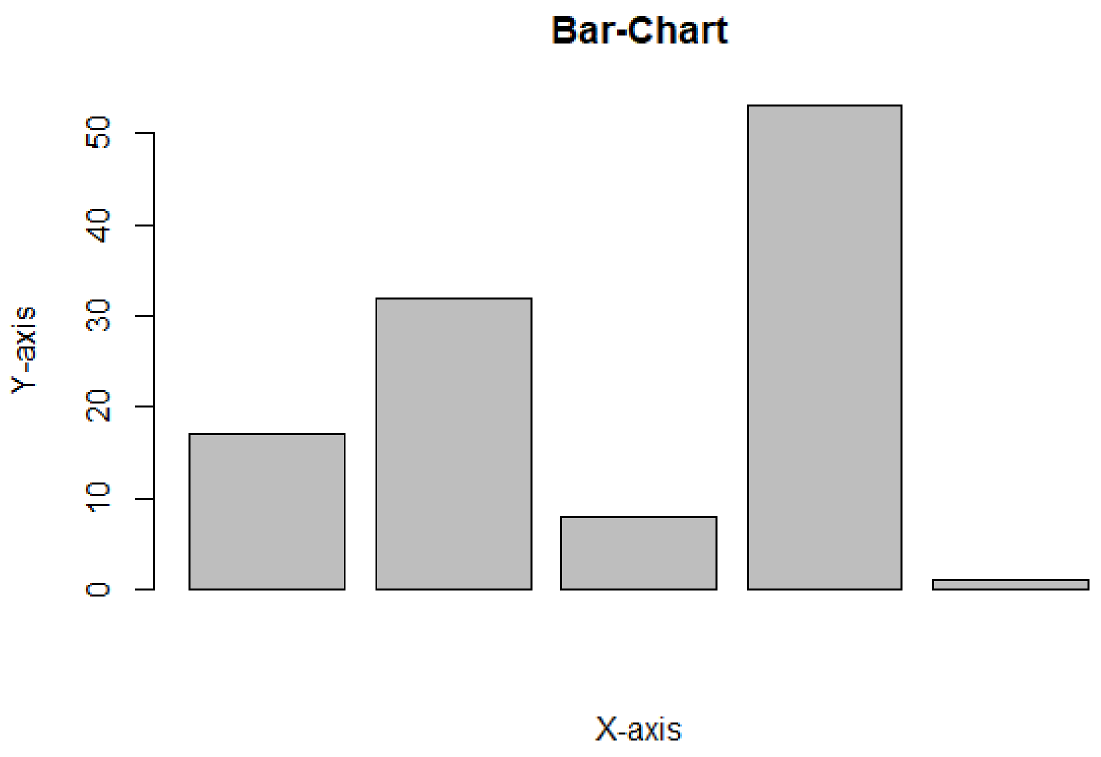
1.Bar graph

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

Parameters:

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

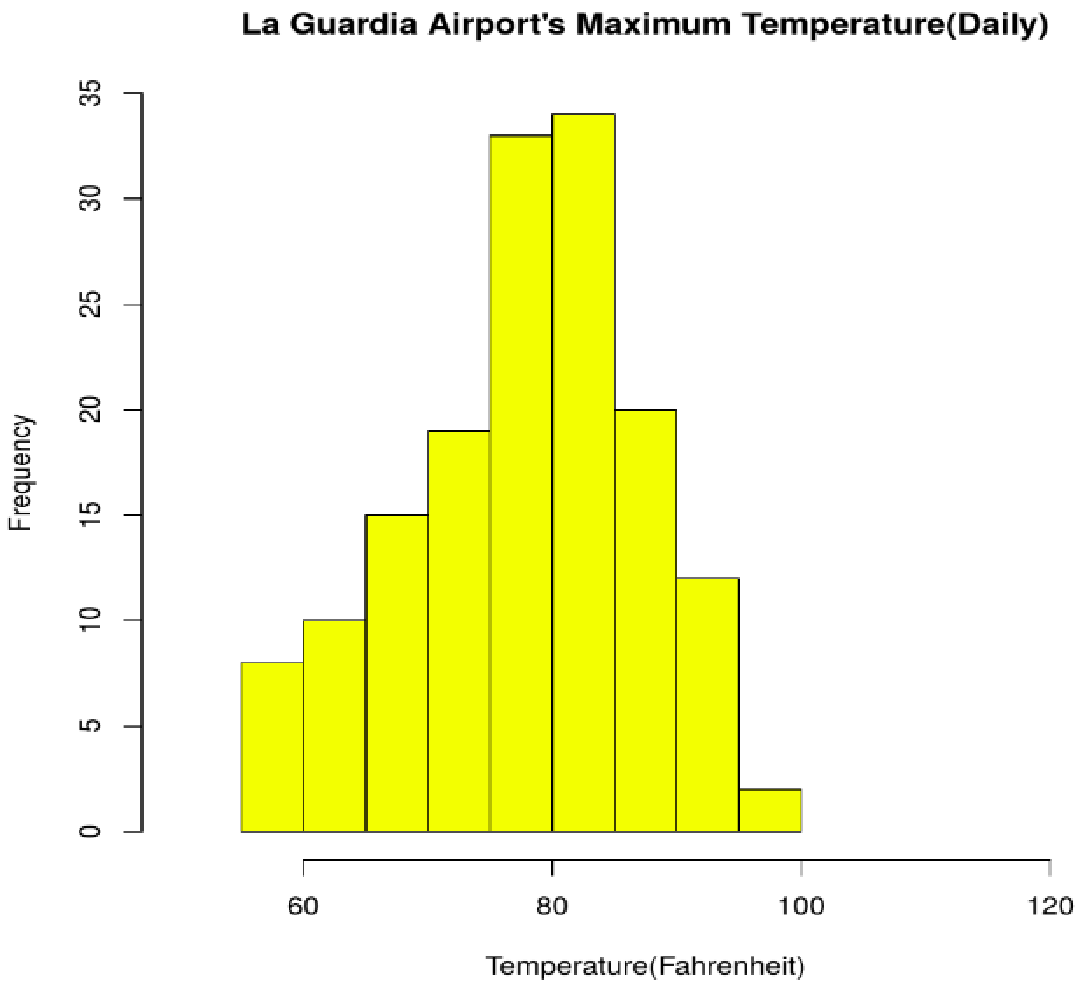
```
# Create the data for the chart
A <- c(17, 32, 8, 53, 1)

# Plot the bar chart
barplot(A, xlab = "X-axis", ylab = "Y-axis", main ="Bar-Chart")
```



1.2Histogram

A histogram is like a bar chart as it uses bars of varying height to represent data distribution. However, in a histogram values are grouped into consecutive intervals called bins. In a Histogram, continuous values are grouped and displayed in these bins whose size can be varied.

```
# Histogram for Maximum Daily Temperature
data(airquality)
hist(airquality$Temp, main ="La Guardia Airport's\
Maximum Temperature(Daily)",
    xlab ="Temperature(Fahrenheit)",
    xlim = c(50, 125), col ="yellow",
    freq = TRUE)
```



Another parameter **freq** when set to *TRUE* denotes the frequency of the various values in the histogram and when set to *FALSE*, the probability densities are represented on the y-axis such that they are of the histogram adds up to one.

Histograms are used in the following scenarios:

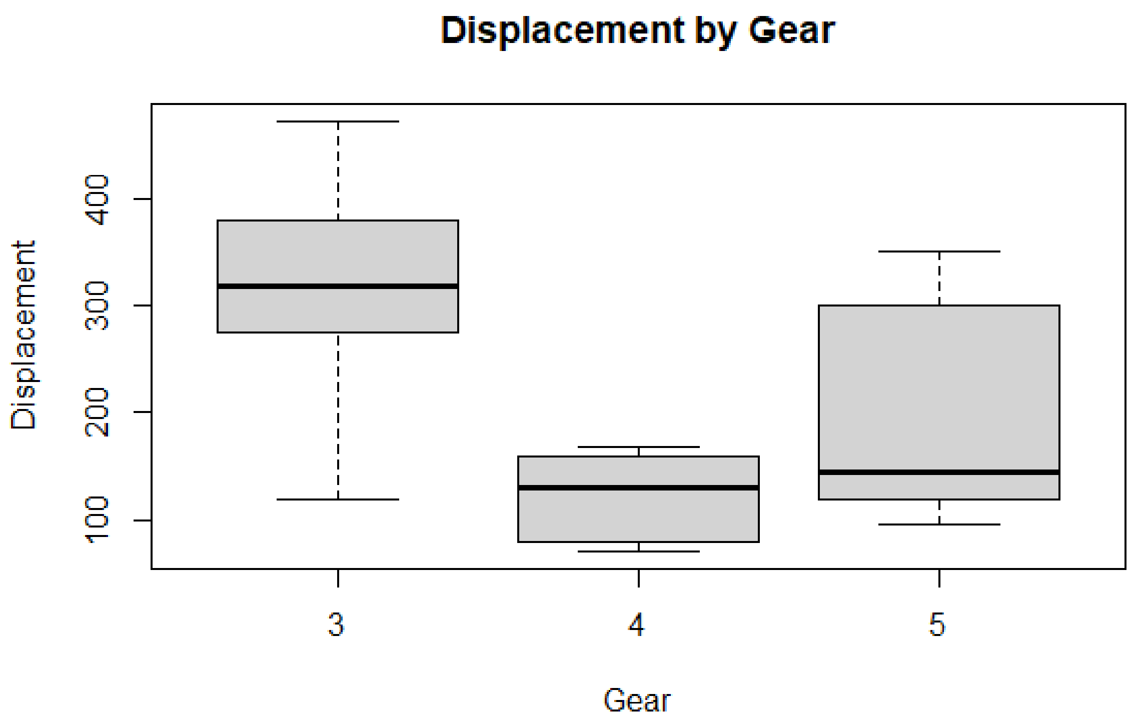
- To verify an equal and symmetric distribution of the data.
- To identify deviations from expected values.

2.Box plot

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

- ***x:*** *This parameter sets as a vector or a formula.*
- ***data:*** *This parameter sets the data frame.*
- ***varwidth:*** *This parameter is a logical value. Set as true to draw width of the box proportionate to the sample size.*
- ***main:*** *This parameter is the title of the chart.*
- ***names:*** *This parameter are the group labels that will be showed under each boxplot.*

```
# Load the dataset
data(mtcars)

# Create the box plot
boxplot(disp ~ gear, data = mtcars, main = "Displacement by Gear",
        xlab = "Gear", ylab = "Displacement")
```



Box Plots are used for:

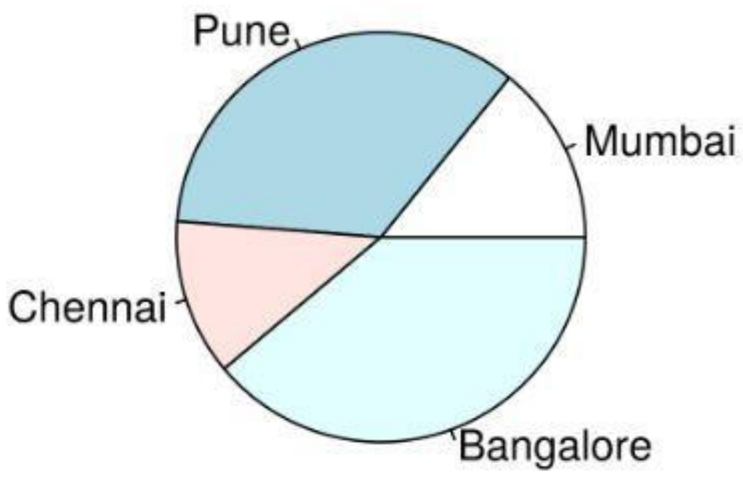
- To give a comprehensive statistical description of the data through a visual cue.
- To identify the outlier points that do not lie in the inter-quartile range of data.

3.Pie chart

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

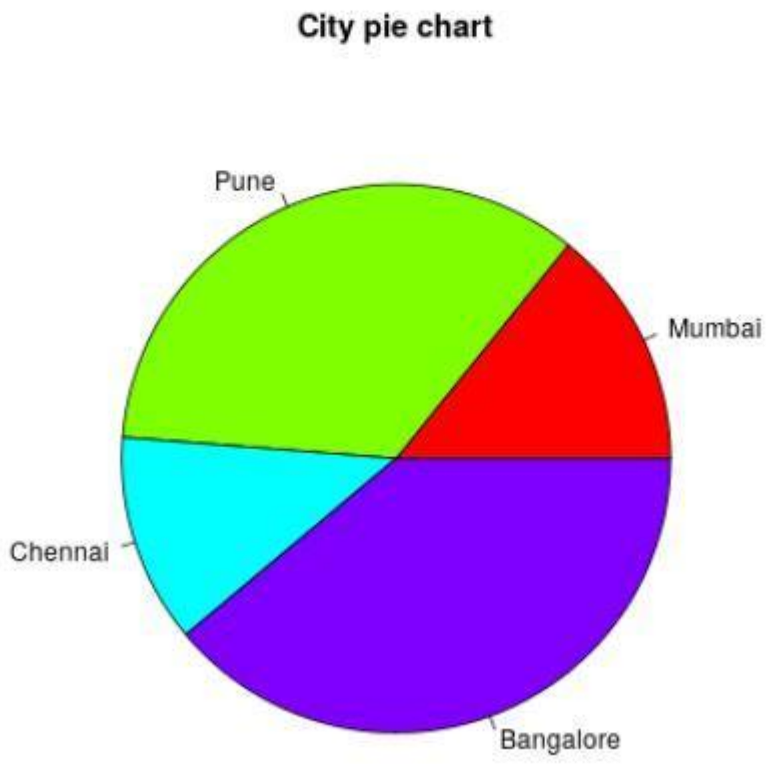
- ***x:*** *This parameter is a vector that contains the numeric values which are used in the pie chart.*
- ***labels:*** *This parameter gives the description to the slices in pie chart.*
- ***radius:*** *This parameter is used to indicate the radius of the circle of the pie chart.(value between -1 and +1).*
- ***main:*** *This parameter is represents title of the pie chart.*
- ***clockwise:*** *This parameter contains the logical value which indicates whether the slices are drawn clockwise or in anti clockwise direction.*
- ***col:*** *This parameter give colors to the pie in the graph.*

```
# Create data for the graph.
geeks<- c(23, 56, 20, 63)
labels <- c("Mumbai", "Pune", "Chennai", "Bangalore")

# Plot the chart.
pie(geeks, labels)
```



Create data for the graph.

```
geeks<- c(23, 56, 20, 63)
labels <- c("Mumbai", "Pune", "Chennai", "Bangalore")

# Plot the chart with title and rainbow
# color pallet.
pie(geeks, labels, main = "City pie chart",
          col = rainbow(length(geeks)))
```



4. Scatter Plot

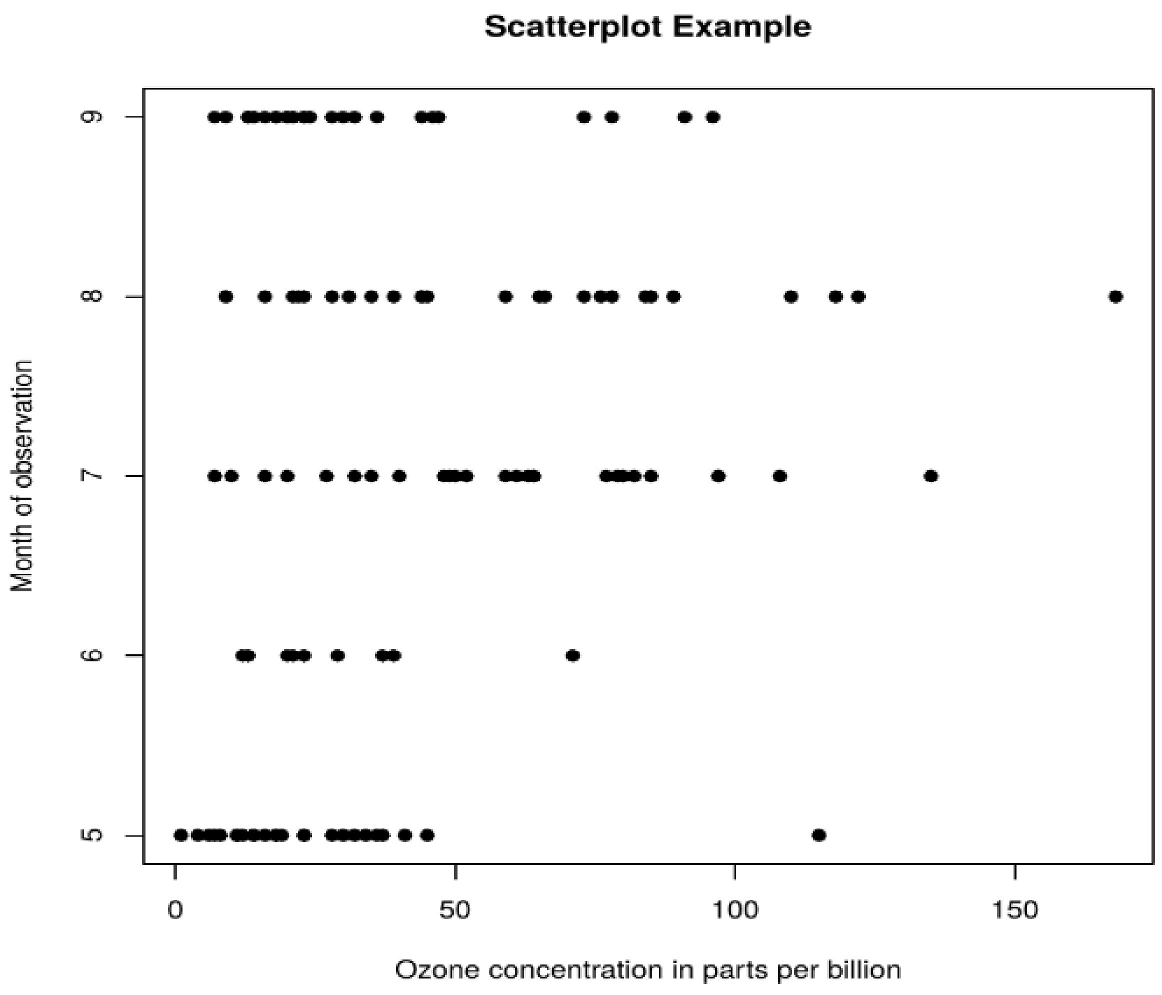
Syntax: *plot(x, y, main, xlab, ylab, xlim, ylim, axes)*
Parameters:

- ***x:*** *This parameter sets the horizontal coordinates.*
- ***y:*** *This parameter sets the vertical coordinates.*
- ***xlab:*** *This parameter is the label for horizontal axis.*
- ***ylab:*** *This parameter is the label for vertical axis.*
- ***main:*** *This parameter main is the title of the chart.*
- ***xlim:*** *This parameter is used for plotting values of x.*
- ***ylim:*** *This parameter is used for plotting values of y.*
- ***axes:*** *This parameter indicates whether both axes should be drawn on the plot.*

```
# Scatter plot for Ozone Concentration per month
data(airquality)

plot(airquality$Ozone, airquality$Month,
     main ="Scatterplot Example",
    xlab ="Ozone Concentration in parts per billion",
    ylab =" Month of observation ", pch = 19)
```

Note:

- pch = 0,square
- pch = 1,circle
- pch = 2,triangle point up
- pch = 3,plus
- pch = 4,cross
- pch = 5,diamond
- pch = 6,triangle point down
- pch = 7,square cross
- pch = 8,star
- pch = 9,diamond plus
- pch = 10,circle plus
- pch = 11,triangles up and down
- pch = 12,square plus
- pch = 13,circle cross
- pch = 14,square and triangle down
- pch = 15, filled square
- pch = 16, filled circle
- pch = 17, filled triangle point-up
- pch = 18, filled diamond
- pch = 19, solid circle
- pch = 20,bullet (smaller circle)
- pch = 21, filled circle blue
- pch = 22, filled square blue
- pch = 23, filled diamond blue
- pch = 24, filled triangle point-up blue
- pch = 25, filled triangle point down blue



Scatter Plots are used in the following scenarios:

- To show whether an association exists between bivariate data.
- To measure the strength and direction of such a relationship.

5.Line Graphs

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

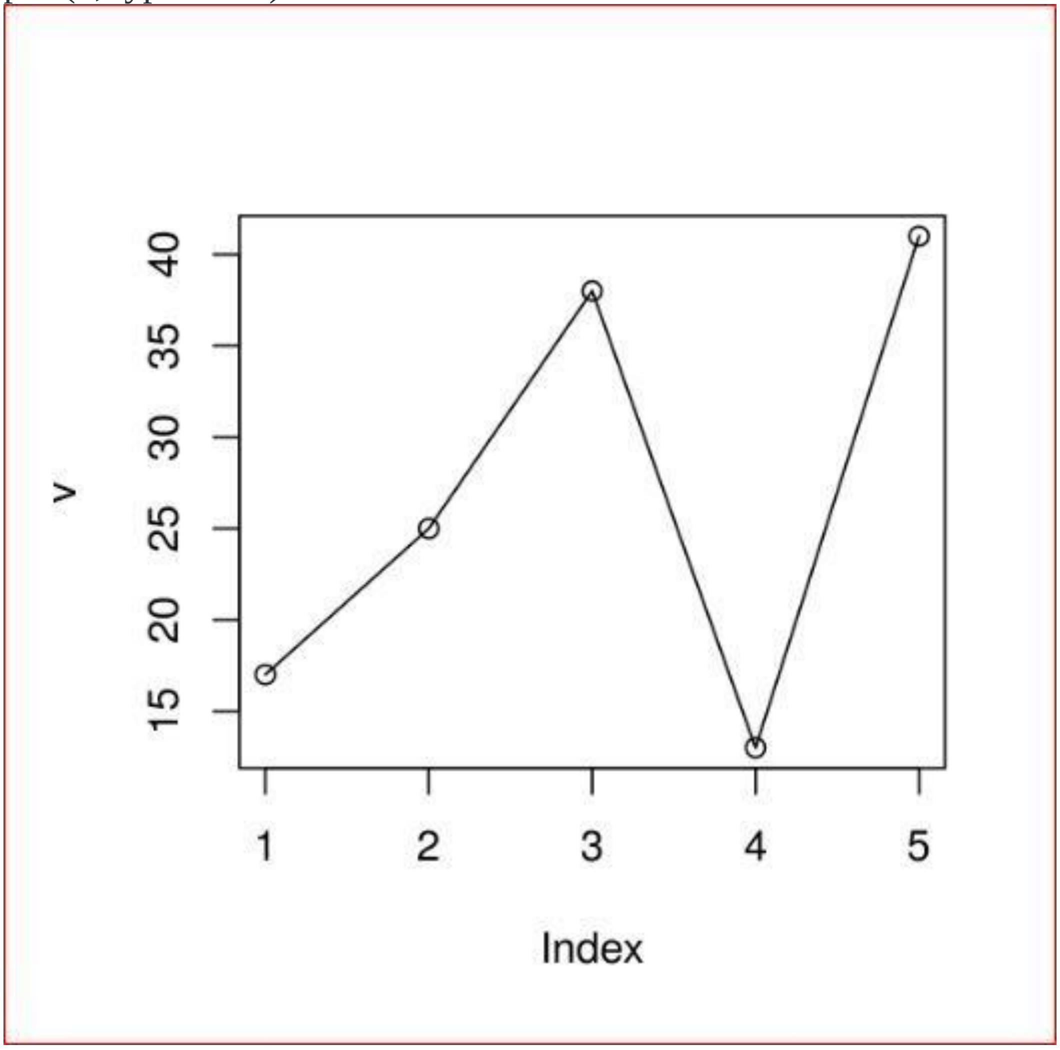
Parameters:

- ***v:*** *This parameter is a contains only the numeric values*
- ***type:*** *This parameter has the following value:*

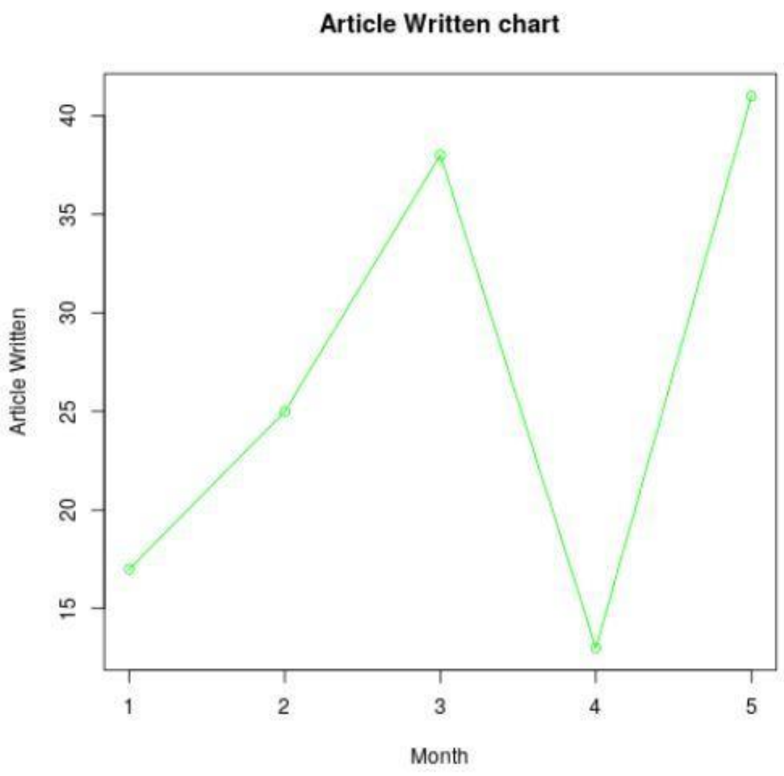
  1. ***"p" :*** *This value is used to draw only the points.*
  2. ***"l" :*** *This value is used to draw only the lines.*
  3. ***"o":*** *This value is used to draw both points and lines*
- ***xlab:*** *This parameter is the label for x axis in the chart.*
- ***ylab:*** *This parameter is the label for y axis in the chart.*
- ***main:*** *This parameter main is the title of the chart.*
- ***col:*** *This parameter is used to give colors to both the points and lines.*

Create the data for the chart.
v <- c(17, 25, 38, 13, 41)

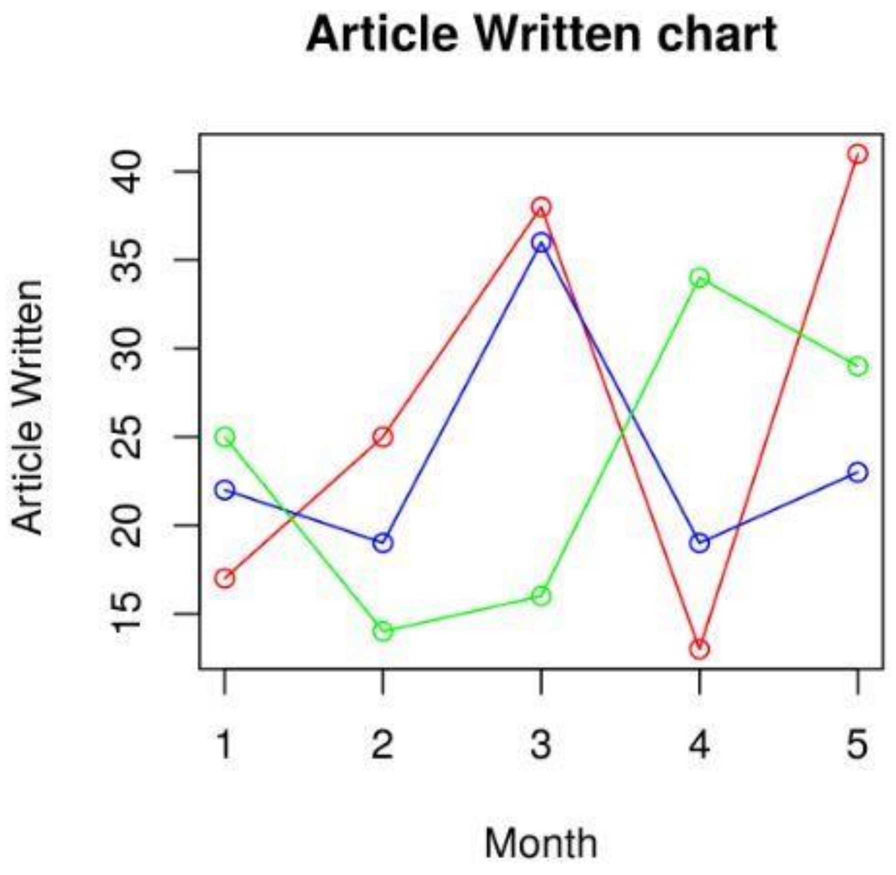
# Plot the bar chart.
plot(v, type = "o")



```
Create the data for the chart.
v <- c(17, 25, 38, 13, 41)

# Plot the bar chart.
plot(v, type = "o", col = "green",
   xlab = "Month", ylab = "Article Written",
   main = "Article Written chart")
```



```
# Create the data for the chart.
v <- c(17, 25, 38, 13, 41)
t <- c(22, 19, 36, 19, 23)
m <- c(25, 14, 16, 34, 29)

# Plot the bar chart.
plot(v, type = "o", col = "red",
   xlab = "Month", ylab = "Article Written ",
   main = "Article Written chart")

lines(t, type = "o", col = "blue")
lines(m, type = "o", col = "green")
```



Summary:

Plot adapts to whatever datasets is given accordingly give a scatter diagram or a box plot,etc.

To get help in r just run ?and write whatever help u need.

For eg: ?plot

1. barplot(A, xlab = "X-axis", ylab = "Y-axis", main ="Bar-Chart")

2. hist(airquality$Temp,
   main ="La Guardia Airport's Maximum Temperature(Daily)",
   xlab ="Temperature(Fahrenheit)",
   xlim = c(50, 125), col ="yellow",
   freq = TRUE)

3. boxplot(disp ~ gear,
   data = mtcars,
   main = "Displacement by Gear",

```
xlab = "Gear",
ylab = "Displacement")
```

4.
   ```
   pie(geeks,
   labels,
   main = "City pie chart",
    col = rainbow(length(geeks)))
   ```

5.
   ```
   plot(airquality$Ozone, airquality$Month,
   main ="Scatterplot Example",
   xlab ="Ozone Concentration in parts per billion",
   ylab =" Month of observation ", pch = 19)
   ```

Categorical variables in R are stored into a factor.

Syntax: factor(x = character(), levels, labels = levels, ordered = is.ordered(x))
R does not use the terms nominal, ordinal, and interval/ratio for types of variables. In R, nominal variables can be coded as variables with factor or character classes. Continuous variable/Interval/ratio data can be coded as variables with numeric or integer classes. An L used with values to tell R to store the data as an integer class. We can code ordinal data as either numeric or factor variables, depending on how we will be summarizing, plotting, and analyzing it.

```
sex <- factor(c("male", "female", "female", "male"))
levels(sex)
nlevels(sex)
```
Nominal Categorical Variable
```
# Create a color vector
color_vector <- c('blue', 'red', 'green', 'white', 'black', 'yellow')
# Convert the vector to factor
factor_color <- factor(color_vector)
factor_color
```

```
x<-c("yes","no","may be later")
fx<-factor(x)
levels(fx)
nlevels(fx)
```

Practical 2 (Data importing and Visualization in R)

Ordinal Categorical Variable

```
# Create Ordinal categorical vector
day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight',
'evening')
# Convert `day_vector` to a factor with ordered level
factor_day <- factor(day_vector, order = TRUE, levels =c('morning', 'midday',
'afternoon', 'evening', 'midnight'))
# Print the new variable
factor_day
## Levels: morning < midday < afternoon < evening < midnight
# Append the line to above code
# Count the number of occurence of each level
summary(factor_day)
```

```
##Continuous Data
dataset <- mtcars
class(dataset$mpg)
```

Questions:

1. Draw a histogram of sepal width from dataset iris. How would you describe this?
2. Create a csv file for the following data

```
## Source Cu
## 1 Site1 19.700
## 2 Site2 10.643
## 3 Site1 33.792
## 4 Site2 5.353
## 5 Site2 19.890
## 6 Site2 26.966
```

Draw a boxplot to show atmospheric copper concentration by sites.

3. Create a csv file for the following data

```
## City ProductA ProductB ProductC
## 1 Seattle 23 11 12
## 2 London 89 6 56
## 3 Tokyo 24 7 13
## 4 Berlin 36 34 44
## 5 Mumbai 3 78 14
```

Draw a bar graph to show total sales across different cities.

4. The dataset swiss contains a standardized fertility measure and various socioeconomic indicators for each of 47 French-speaking provinces of Switzerland in about 1888.

a. Draw a scatterplot of Fertility against %Catholic. Which kind of areas have the lowest fertility rates?
b. Discuss the relationship between the variables Education and Agriculture. 5

5. Write a R program to change the first level of a factor with another level of a given factor.
6. Write a R program to create an ordered factor from data of minimum 20 elements consisting of names of months.

For the following vectors, plot a pie chart one with labels (labels<-c("A","B","C","D")) and other with no labels
x<-c(75,58,95,26,34,56,32,78,90,33,21,11,45,67)
y<-c(56,58,74,89,34,12,32,78,90,22,21,11,33,67)

hint: pie(x,labels)
to use help, syntax is ?pie

7. For mtcars data, draw a boxplot of mpg against cyl.