UNIT -IV

IV Data Definitions and Analysis Techniques   Data Definitions: Introduction to statistical learning and R-Programming, Elements, Variables, Data structures, Data categorization, Levels of Measurement, Data management and indexing.
Analysis Techniques: Introduction to statistical hypothesis generation and its types.

R Programming

- R is a popular programming language used for statistical computing and graphical presentation. Its most common use is to analyze and visualize data.
- **R is an interpreted computer programming language which was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand."**
- R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.
- It works on different platforms (Windows, Mac, Linux)
- It is a great resource for data analysis, data visualization, data science and machine learning
- It has many packages (libraries of functions) that can be used to solve different problems

Features of R programming

- It provides effective data handling and storage facility
- It is an interpreted language
- It provides highly extensible graphical techniques.
- It is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- It's a platform-independent language. This means it can be applied to all operating system.
- It has a consistent and incorporated set of tools which are used for data analysis.
- For different types of calculation on arrays, lists and vectors, R contains a suite of operators.

Applications of R

- There are several-applications available in real-time. Some of the popular applications are as follows:

Facebook,Google,Twitter,Sunlight Foundation.

Environments in R Programming

- The environment is a virtual space that is triggered when an interpreter of a programming language is launched
- The environment is a collection of all the objects, variables, and functions. Or, Environment can be assumed as a top-level object
- An environment in R programming can be created using **new.env()** function
- The variables can be accessed using **$ or [[ ]]** operator. But, each variable is stored in different memory locations
- There are four special environments: **globalenv(), baseenv(), emptyenv() and environment()**
- A variable in an environment is deleted using **rm()** function

- A variable or a function can be searched in R programming by using **where()** function among all the environments and packages present

R Packages

- R packages are the collection of R functions, sample data, and compile codes
- In the R environment, these packages are stored under a directory called "**library**."
- During installation, R installs a set of packages
- We can add packages later when they are needed for some specific purpose. Only the default packages will be available when we start the R console
- R provides libPaths() function to find the library locations
- R provides library() function, which allows us to get the list of all the installed packages.
- R provides search() function to get all packages currently loaded in the R environment.
- There are two techniques to add new R packages.

(i) installing package directly from the CRAN directory,

(ii)install it manually after downloading the package to our local system

Ex. install.packages("XML")

Data Types in R Programming

- Variables are the reserved memory location to store values
- The operating system allocates memory based on the data type of the variable and decides what can be stored in the reserved memory.
- **R supports three ways of variable assignment:**

```
variable_name = value
#using leftward operator
variable_name <- value
#using rightward operator
value -> variable_name
```

Data Types in R Programming

Data type	Example	Description
Logical	True, False	two possible values either true/false.
Numeric	12,32,112,5432	default computational data type.
Integer	3L, 66L, 2346L	Here, L tells R to store the value as an integer,
Complex	Z=1+2i, t=7+3i	A complex value in R is defined as the pure imaginary value i.
Character	'a', '"good'", "TRUE", '35.4'	In R programming, a character is used to represent string values
Raw		A raw data type is used to holds raw bytes.

Logical
Complex
Raw
Data Types
Numeric
Integer
Character

R Operators

R divides the operators in the following groups:
i) Arithmetic operators ii) Assignment operators iii) Comparison operators iv)Logical operators v) Miscellaneous operators

i)R Arithmetic Operators

-	**Subtraction**	**x - y**
*****	**Multiplication**	**x * y**
/	**Division**	**x / y**
^	**Exponent**	**x ^ y**
%%	**Modulus (Remainder from division)**	**x %% y**
%/%	**Integer Division**	**x%/%y**

ii)R Assignment Operators
Assignment operators are used to assign values to variables
variable_name = value
#using leftward operator
variable_name <- value
#using rightward operator
value -> variable_name

iii)R Comparison Operators
Logical operators are used to combine conditional statements

Operator	Description
&	It returns TRUE if both elements are TRUE
&&	Returns TRUE if both statements are TRUE
|	It returns TRUE if one of the statement is TRUE
||	It returns TRUE if one of the statement is TRUE.
!	returns FALSE if statement is TRUE

iv)R Miscellaneous Operators

Operator	Description	Example

:	Creates a series of numbers in a sequence	x <- 1:10
%in%	Find out if an element belongs to a vector	x %in% y
%*%	Matrix Multiplication	x <- Matrix1 %*% Matrix2

v)R supports the usual logical conditions

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Conditional Statements:
If statement" is written with the if keyword, and it is used to specify a block of code to be executed if a condition is TRUE
Ex1:
```
a <- 33
b <- 200

if (b > a) {
  print("b is greater than a")
}
```
Ex2:
```
a <- 33
b <- 33
if (b > a) {
  print("b is greater than a")
} else if (a == b) {
  print ("a and b are equal")
}
```
Loops
Ex1:
```
i <- 1
while (i < 6) {
  print(i)
  i <- i + 1
}
```
Output:
```
1 2 3 4 5
```

Ex2:

```
i <- 1
while (i < 6) {
  print(i)
  i <- i + 1
  if (i == 4) {
    break
  }
}
```

Output: 1 2 3

Ex3:

```
i <- 0
while (i < 6) {
  i <- i + 1
  if (i == 3) {
            next
  }
  print(i)
}
```

Output: 1 2 4 5 6

-
  ```
  for (x in 1:10) {
    print(x)
  }
  ```

-
  ```
  fruits <- list("apple", "banana", "cherry")

  for (x in fruits) {
    print(x)
  }
  ```

Creating a Function:

FUNCTION DEFENINTION

-
  ```
  fun1 <- function() {      # create a function with the name fun1
     print("Hello World!")
     }
  ```

FUNCTION CALL:

```
 fun1()
```

Function with arguments :

```
fun1 <- function(fname, lname) {
  paste(fname, lname)
}

fun1("Peter“,”Paul”)
fun1("Lois“,”Jack”)
fun1("Stewie“,”Star”)
```

DATA STRUCTURES:

- **VECTORS**
- **LISTS**
- **MATRICES**
- **ARRAYS**
- **DATA FRAMES**

Vector:
Vectors are the most basic R data objects and there are six types of atomic vectors. They are logical, integer, double, complex, character and raw
Single Element Vector
# Atomic vector of type character.
print("abc");
# Atomic vector of type double.
print(12.5)
Multiple Elements Vector
v <- 5:13
Print(v) #5 6 7 8 9 10 11 12 13
# If the final element specified does not belong to the sequence then it is discarded. v <- 3.8:11.4
print(v)#3.8 4.8 5.8 6.8 7.8 8.8 9.8 10.8
Seq operator
# Create vector with elements from 5 to 9 incrementing by 0.4.
print(seq(5, 9, by=0.4)) #5.0 5.4 5.8 6.2 6.6 7.0 7.4 7.8 8.2 8.6 9.0
c() function
The non-character values are coerced to character type if one of the elements is a character
# The logical and numeric values are converted to characters.
s <- c('apple','red',5,TRUE)
print(s) #"apple" "red" "5" "TRUE“
Accessing Vector Elements :
The [ ] brackets are used for indexing. Indexing starts with position 1. Giving a negative value in the index drops that element from result. TRUE, FALSE or 0 and 1 can also be used for indexing.
# Accessing vector elements using position.
t <- c("Sun","Mon","Tue","Wed","Thurs","Fri","Sat")
u<- t[c(2,3,6)]
print(u) # "Mon" "Tue" "Fri“
# Accessing vector elements using logical indexing.
v <- t[c(TRUE,TRUE,FALSE,FALSE,FALSE,TRUE,FALSE)]
print(v)#"Sun" "Mon" "Fri“
# Accessing vector elements using negative indexing.
x <- t[c(-2,-3)]
print(x) #"Sun"   "Wed"   "Thurs" "Fri"   "Sat"
Vector Manipulation
Vector Arithmetic
Two vectors of same length can be added, subtracted, multiplied or divided giving the result as a vector output.
# Create two vectors.
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11,0,8,1,2)
# Vector addition.
add.result <- v1+v2
print(add.result)   #7 19 4 13 1 13
# Vector substraction.
sub.result <- v1-v2
print(sub.result)   #-1 -3 4 -3 -1 9
# Vector multiplication.
multi.result <- v1*v2
print(multi.result)   #12 88 0 40 0 22
# Vector division.

divi.result <- v1/v2
print(divi.result)   #0.7500000 0.7272727 Inf 0.6250000 0.0000000 5.5000000
Vector Element Recycling
If we apply arithmetic operations to two vectors of unequal length, then the elements of the shorter vector are recycled to complete the operations.
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11) # V2 becomes c(4,11,4,11,4,11)
add.result <- v1+v2
print(add.result)   #7 19 8 16 4 22
Elements in a vector can be sorted using the sort() function.
v <- c(3,8,4,5,0,11, -9, 304)
 # Sort the elements of the vector.
sort.result <- sort(v)
print(sort.result) #-9   0   3   4   5   8 11 304
# Sorting character vectors.
v <- c("Red","Blue","yellow","violet")
sort.result <- sort(v)
print(sort.result) #"Blue" "Red" "violet" "yellow"
# Sorting character vectors in reverse order.
revsort.result <- sort(v, decreasing = TRUE)
print(revsort.result) #"yellow" "violet" "Red" "Blue"
Lists:
A list in R can contain many different data types inside it. A list is a collection of data which is ordered and changeable.The list() function is used to create a list.
# List of strings
thislist <- list(“ABC", “PQR", “XYZ")
# Print the list
thislist   #[[1]] [1] "ABC”   [[2]] [1] "PQR“ [[3]][1] "XYZ“
thislist[1]   #[[1]] [1] "ABC“
Change Item Value
thislist[1] <- “WXY“
Check if Item Exists
Matrices:
Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout
A Matrix is created using the **matrix()** function.
matrix(data, nrow, ncol, byrow, dimnames)
data is the input vector which becomes the data elements of the matrix.
nrow is the number of rows to be created.
ncol is the number of columns to be created.
byrow is a logical clue. If TRUE then the input vector elements are arranged by row.
dimname is the names assigned to the rows and columns.
M <- matrix(c(3:14), nrow = 4, byrow = TRUE)
print(M)
O/P

```
      [,1] [,2] [,3]
[1,]    3    4    5
[2,]    6    7    8
[3,]    9   10   11
[4,]   12   13   14
```

# Access the element at 3rd column and 1st row.
print(M[1,3])   #5

Matrix Computations

```
# Create two 2x3 matrices.
matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
print(matrix1)
matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
print(matrix2)
# Multiply the matrices.
result <- matrix1 * matrix2
cat("Result of multiplication","\n")
print(result)
Result of multiplication
     [,1] [,2] [,3]
[1,]   15    0    6
[2,]   18   36   24
```

Arrays:

- Arrays can have more than two dimensions
- If we create an array of dimension (2, 3, 4) then it creates 4 rectangular matrices each with 2 rows and 3 columns. Arrays can store only data type.
- array_name **<-** array(data, dim= (row_size, column_size, matrices, dim_names))

```
vec1 <-c(1,3,5)
vec2 <-c(10,11,12,13,14,15)
 #Taking these vectors as input to the array
res <- array(c(vec1,vec2),dim=c(3,3,2))
print(res)
     [,1] [,2] [,3]
[1,]    1   10   13
[2,]    3   11   14
[3,]    5   12   15
, , 2
     [,1] [,2] [,3]
[1,]    1   10   13
[2,]    3   11   14
[3,]    5   12   15
```

Data Frame:

- A data frame is a two-dimensional array-like structure or a table in which a column contains values of one variable, and rows contains one set of values from each column
- A data frame is used to store data table and the vectors which are present in the form of a list in a data frame, are of equal length.
- matrix can contain one type of data, but a data frame can contain different data types such as numeric, character, factor, etc.

Characteristics of a data frame:

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factor or character type.
- Each column should contain same number of data items.
- # Creating the data frame.
- emp.data<- data.frame(
- employee_id = c (1:4),
- employee_name = c("Shyam","Arjun","Neha","Guru")
-)
- # Printing the data frame.

- print(emp.data)
- # Extract first 1 rows.
- result <- emp.data[1:2,]
- print(result)
- # Extract 1st and 2nd row with 1st and 2nd column.
- result <- emp.data[c(1,2),c(1,2)]
- print(result)

```
      employee_id employee_name
1         1        Shyam
2         2        Arjun
3         3         Neha
4         4         Guru
employee_id     employee_name
1         1          Shyam
2         2          Arjun
employee_id     employee_name
1         1           Shyam
2         2          Arjun
```

Data Categorization:

- Factor in R is also known as a categorical variable that stores both string and integer data values as levels
- Factor is used in Statistical Modeling And data analysis
- In a dataset, we can have two types of variables:

categorical and **continuous**.

- **Categorical Variable** is limited and usually based on a particular finite group

for example countries, year, gender, occupation etc.

- **Continuous Variable** can take any values like revenue,

price of a share

- Factors are data structures that are implemented to categorize the data or represent categorical data and store it on multiple levels
- Factors are the data objects which are used to categorize the data and store it as levels. They can store both strings and integers
- Factors are created using the **factor ()** function by taking a vector as input.
- Factor's levels will always be character values
- An example of a categorical variable is gender.
- In most cases, you can limit the categories to “Male” or “Female”
- The factor function is used to encode a vector as a factor
- Categorical variables in R are stored into a factor
- gender_vector <- c("Male", "Female", "Female", "Male", "Male")
- There are two categories (**factor levels**): “Male” and “Female”.
- levels() is used to print the levels
- We can also set the levels, by adding the levels

argument inside the factor() function

- factors are stored as integer vectors
- **Accessing components of a factor:**

>music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"))

```
>music_genre
  [1] Jazz    Rock    Classic Classic Pop     Jazz    Rock    Jazz
  Levels: Classic Jazz Pop Rock
>music_genre[2]
```

```
[1] Rock
Levels: Classic Jazz Pop Rock
```

- To add new level

```
levels(music_genre) <- c(levels(music_genre), "Other")
>music_genre
[1] Jazz    Rock    Classic Classic Pop     Jazz    Rock    Jazz
Levels: Classic Jazz Pop Rock Other
```

- We can classify R factors as **ordered or unordered**
- By default, the levels are arranged in alphabetical order and are all considered equal irrespective of their arrangement.
- Ex: **unordered** R factor

```
>sizes <- c("s","m","s","l","m","xs","l","m","xl","xxl","s", "l","xs","xl","m","l")
>sizef <- factor(sizes)
>sizef
[1] s   m   s   l   m   xs  l   m   xl  xxl s   l   xs  xl  m   l
Levels: l m s xl xs xxl
>sizefnew <- factor(sizes,levels=c("xs","s","m","l","xl","xxl"))
>sizefnew
[1] s   m   s   l   m   xs  l   m   xl  xxl s   l   xs  xl  m   l
Levels: xs s m l xl xxl
>as.ordered(sizefnew)
[1] s   m   s   l   m   xs  l   m   xl  xxl s   l   xs  xl  m   l
Levels: xs < s < m < l < xl < xxl
```

Levels of Measurement:

- In 1946, Harvard University psychologist “Stanley Smith Stevens” developed the theory of the four levels of measurement.
- There are four main levels of measurement: **nominal**, **ordinal**, **interval**, and **ratio**
- There are various levels of measurement you could use for this variable.
- Ex: (a) 10-19k, (b) 20-29k, (c) 30-39k, and so on. They can be categorized as “high,” “medium,” or “low.”
- Nominal scale simply categorizes variables according to qualitative labels (or names)
- These labels and groupings don’t have any order or hierarchy to them, nor do they convey any numerical value
- Nominal scale is a naming scale, where variables are simply “named” or labeled, with no specific order.
- Nominal scale is often used in research surveys and questionnaires where only variable labels hold significance

Nominal Scale Examples

· Gender

- Political preferences
- Place of residence

What is your Gender?	What is your Political preference?	Where do you live?
•M- Male •F- Female	•1- Independent •2- Democrat •3- Republican	•1- Suburbs •2- City •3- Town

- **Ordinal Scale** is defined as a variable measurement scale used to simply depict the order of variables and not the difference between each of the variables
- These scales are generally used to depict non-mathematical ideas such as frequency, satisfaction, happiness, a degree of pain, etc
- Ex: Academic grades (A, B, C, and so on)
  Happiness on a scale of 1-10
  Satisfaction (extremely satisfied, quite satisfied, slightly dissatisfied, extremely dissatisfied)
  Income (high, medium, or low)

- Interval Scale is defined as a numerical scale where the order of the variables is known as well as the difference between these variables.
- Variables that have familiar, constant, and computable differences are classified using the Interval scale
- Interval scale contains all the properties of the ordinal scale.
- It's a numerical scale in which the order is known and the difference between the values has meaning

- :
- Ratio Scale is defined as a variable measurement scale that not only produces the order of variables but also makes the difference between variables known along with information on the value of true zero
- A value of zero on a ratio scale means that the variable you're measuring is absent
- There is no negative point as lowest score is 0 point

Ex:

- Height
- Income
- Distance travelled

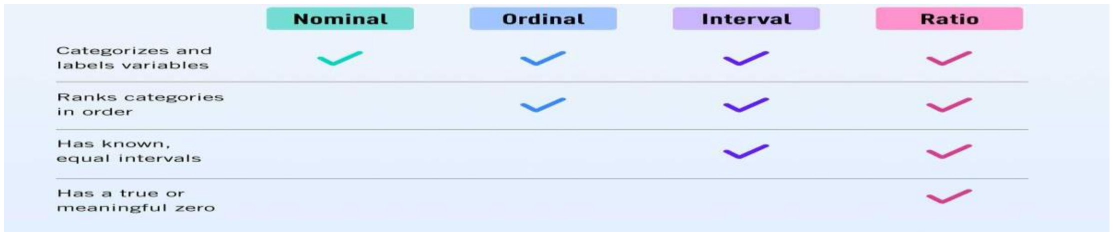
- Time elapsed or time remaining
- Money in your bank account, wallet, or pocket

What is your weight in pounds?

- Less than 70
- 70 – 120
- 121 – 150
- More than 150

What is your age?

- Less than 20
- 20 – 30
- 31 – 40
- 41 – 50
- More than 50



	Nominal	Ordinal	Interval	Ratio
Categorizes and labels variables	✓	✓	✓	✓
Ranks categories in order		✓	✓	✓
Has known, equal intervals			✓	✓
Has a true or meaningful zero				✓

Operation	**Nominal**	**Ordinal**	**Interval**	**Ratio**
The sequence of variables is established	–	Yes	Yes	Yes
Mode	Yes	Yes	Yes	Yes
Median	–	Yes	Yes	Yes
Mean	–	–	Yes	Yes
Difference between variables can be evaluated	–	–	Yes	Yes
Addition and Subtraction of variables	–	–	Yes	Yes
Multiplication and Division of variables	–	–	–	Yes
Absolute zero	–	–	–	Yes

File Handling:

- To create a new file
- >file.create(“sum.txt")

- To write content to the created file
- >write.table(x = mtcars[1:10],file = "sum.txt")
- # Rename file
- file.rename(" sum.txt ", " sum1.txt ")
- # Reading txt file
- new.sum <- read.table(file = " sum1.txt ")
- print(new.sum)
- #to list all files and directories
- list.files()
- #TO CREATE DIRECTORY
- dir.create(“NEWDIR”)
- # to read a file in table format
- myCSV = read.csv(“TEST.csv")
- print(myData)

install.packages("xlsx")

Reading the Excel File

read.xlsx(file_name,sheet_index) #syntax

excel_data<- read.xlsx("employee.xlsx", sheetIndex = 1)

print(excel_data)

Reading JSON File(JavaScript Object Notation)

# The JSON file contains the data as text in a human-readable format

install.packages("rjson")

library("rjson")

new.json<-fromJSON(file="new.json")

print(new.json)

$ID

[1] "1" "2" "3" "4" "5" "6" "7" "8"

$Name

[1] "Rick" "Dan" "Michelle"

[4] "Ryan" "Gary" "Nina"

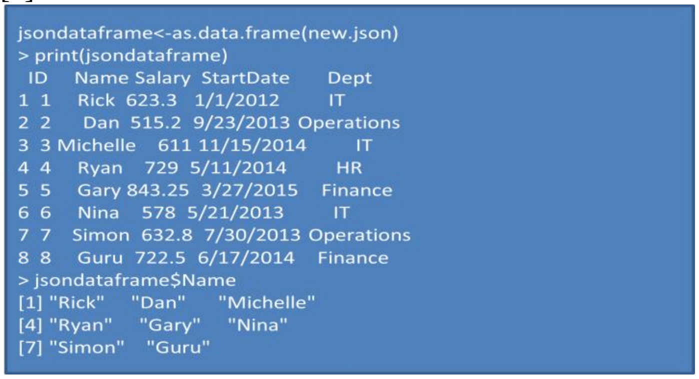
[7] "Simon" "Guru"

$Salary

[1] "623.3" "515.2" "611"

[4] "729" "843.25" "578"

[7] "632.8" "722.5"

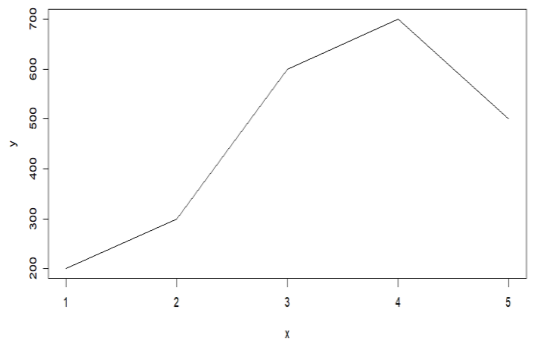


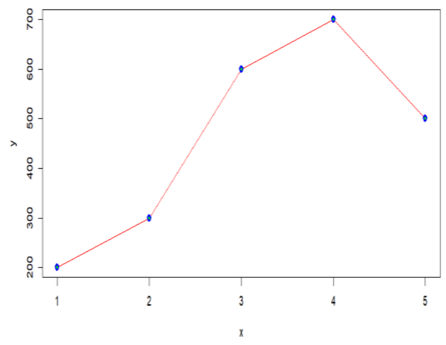
```
jsondataframe<-as.data.frame(new.json)
> print(jsondataframe)
  ID     Name Salary  StartDate       Dept
1  1     Rick  623.3   1/1/2012         IT
2  2      Dan  515.2  9/23/2013 Operations
3  3 Michelle    611 11/15/2014         IT
4  4     Ryan    729  5/11/2014         HR
5  5     Gary 843.25  3/27/2015    Finance
6  6     Nina    578  5/21/2013         IT
7  7    Simon  632.8  7/30/2013 Operations
8  8     Guru  722.5  6/17/2014    Finance
> jsondataframe$Name
[1] "Rick"     "Dan"      "Michelle"
[4] "Ryan"     "Gary"     "Nina"
[7] "Simon"    "Guru"
```

Graphs in R

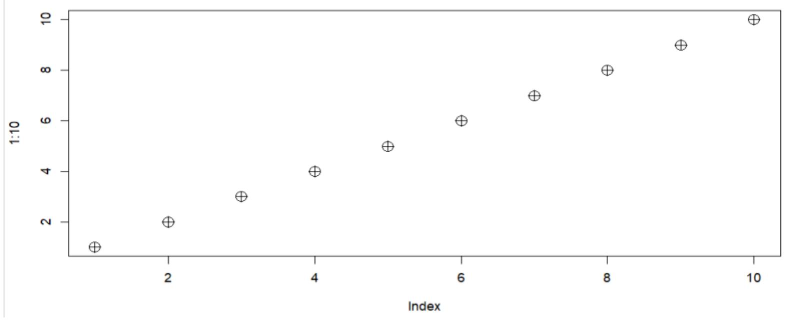
- A line chart can be created in base R with the plot function
- # Data
- x <- c(1, 2, 3, 4, 5)
- y <- c(200, 300, 600, 700, 500)
- # Vectors
- plot(x, y, type = "l") # l for line
- # Data frame
- plot(data.frame(x, y), type = "l") # Equivalent
- # Formula
- plot(y ~ x, type = "l") # Equivalent
- points(x, y,   # Coordinates

```
pch = 21,      # Symbol
bg = "green",  # Background color of the symbol
col = "blue",  # Border color of the symbol
lwd = 3)
```





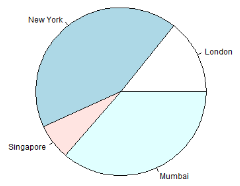
- Plot :The plot() function is used to draw points (markers) in a diagram.
- plot(1:10, pch=10, cex=2) # point shape –pch ,cex-size of symbol



- **Pie Charts**
- pie() function which takes positive numbers as a vector input.

The parameters are used to control labels, color, title etc.
pie(x, labels, radius, main, col, clockwise

- >x <- c(21, 62, 10, 53)
- >labels <- c("London", "New York", "Singapore", "Mumbai")
- # Give the chart file a name.
- >png(file = "city.png")
- # Plot the chart.
- >pie(x,labels)
- # Save the file.
- >dev.off()



- A bar chart represents data in rectangular bars

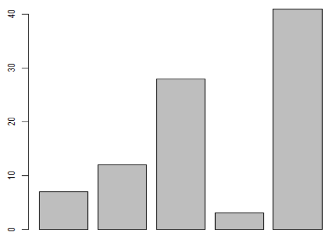
with length of the bar proportional to the value
of the variable. R uses the function **barplot()**

to create bar charts

- 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.
- # Create the data for the chart
- H <- c(7,12,28,3,41)
- # Give the chart file a name
- png(file = "barchart.png")
- # Plot the bar chart
- barplot(H)
- # Save the file
- dev.off()

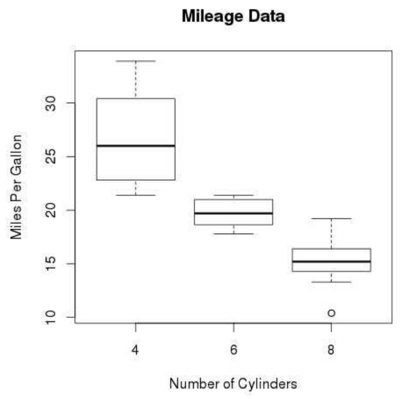


- Boxplots are a measure of how well distributed is the data in a data set.

It divides the data set into three quartiles.
This graph represents the minimum, maximum, median

- 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
- # Give the chart file a name.
- png(file = "boxplot.png")
- # Plot the chart.
- boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders",
-   ylab = "Miles Per Gallon", main = "Mileage Data")
- # Save the file.

dev.off()

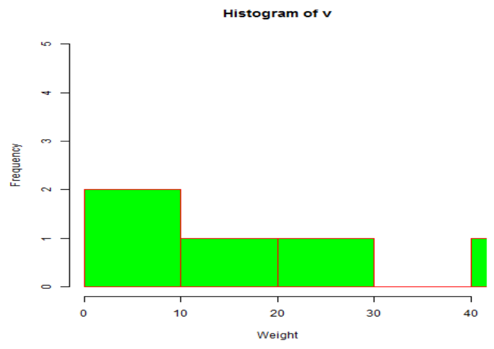


- A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range hist(v,main,xlab,xlim,ylim,breaks,col,border)

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.

```
# Create data for the graph.
v <- c(9,13,21,8,36,22,12,41,31,33,19)
# Give the chart file a name.
png(file = "histogram_lim_breaks.png")
# Create the histogram.
hist(v,xlab = "Weight",col = "green",border = "red", xlim = c(0,40), ylim = c(0,5),
  breaks = 5)
# Save the file.
dev.off()
```



Data Management:

- Data management is the practice of collecting, organizing, protecting, and storing an organization's data so it can be analyzed for business decisions
- The software helps with everything from data preparation to cataloging, search, and governance, allowing people to quickly find the information they need for analysis.
- Effective data management is a crucial piece of deploying the IT systems.
- The data management process includes a combination of different functions that collectively aim to make sure that the data in corporate systems is accurate, available and accessible
- The principles of Data Management are:
    - ➢ Creating, accessing, and regularly updating data across diverse data tiers
    - ➢ Storing data both on-premises and across multiple clouds
    - ➢ Providing both high availability and rapid disaster recovery
    - ➢ Using data in an increasing number of algorithms, analytics, and applications
    - ➢ Ensuring effective data privacy and data security
    - ➢ Archiving and destroying data in compliance with established retention schedules and compliance guidelines
- The data management process includes a wide range of tasks and procedures, such as:

➢ Collecting, processing, validating, and storing data
➢ Integrating different types of data from disparate sources, including structured and unstructured data
➢ Ensuring high data availability and disaster recovery
➢ Governing how data is used and accessed by people and apps
➢ Protecting and securing data and ensuring data privacy

Types of Data Management:

Data preparation is used to clean and transform raw data into the right shape and format for analysis, including making corrections and combining data sets.

Data pipelines enable the automated transfer of data from one system to another.

ETLs (Extract, Transform, Load) are built to take the data from one system, transform it, and load it into the organization's data warehouse.

Data catalogs help manage metadata to create a complete picture of the data, providing a summary of its changes, locations, and quality while also making the data easy to find.

Data warehouses are places to consolidate various data sources, contend with the many data types businesses store, and provide a clear route for data analysis.

Data governance defines standards, processes, and policies to maintain data security and integrity.

Data architecture provides a formal approach for creating and managing data flow.

Data security protects data from unauthorized access and corruption.

Data modeling documents the flow of data through an application or organization.

Categories of Data Management:

- **Master Data Management (MDM):** Master data management ensures that the company operates on the most precise and reliable information and helps eliminate needless data processing

- **Data Steward:** The function of the data steward is to regulate the nature of the information the company collects and ensure it conforms to its policies. A data steward monitors the data gathering processes and ensures that the data collated is in the right format.
- **Data Quality Management (DQM):** DQM preserves data quality. It does this by looking for hidden errors or inconsistencies in the collated data, thus maintaining high-quality information
- **Data Security:** This is the most crucial part of data management. Data security experts require the knowledge of encryption management to prevent data breaches and track any suspicious activity in the enterprise's database.
- **Data Governance:** Data governance, as the name implies, is the control of data in an organization. It must establish the policies and regulations for the quality of the information in the company
- **Big Data Management:**Big data management is gathering and processing huge data sets and is carried out using big data tools.
- **Data Warehousing:** This is the collation and storage of enormous amounts of data in the businesses' data warehouse. A data warehouse comprises many databases, each containing rows and columns.

Advantages to Data Management:

- **They can handle any volume.** No matter how much your database grows, data management systems can easily manage unlimited quantities of data and fields.
- **They facilitate global work**. Any permitted user can access the data management tools from any device and location, and continue to work under any circumstance. This centralized functionality of data management programs is perfectly suited to the needs of working remotely and synchronizing international teams.
- **They eliminate data redundancies and omissions**. Such errors are difficult to detect manually, but automated data management techniques can boost your accuracy to close to 100%.
- **They guarantee a high level of security, efficiency and privacy**. This is essential if you are leaving all the company's information in the hands of a single tool. In addition, they include backup generation, a history of changes and options for recovery of past data.
- **They save on storage**. Thousands of Excel files, photos in different formats and Word docs with translations can become complicated and overwhelming for any team. A data management system can help save on space since it will include its own storage system (better still if it’s in the cloud) and also save on the cost of subscribing to accounts like Dropbox.
- **They optimize data access**. Your team will save a lot of time by having access to the data they need, and can rely on, at all times. Data can also be exported to any format you need to share with third parties or across other digital channels.

Limitations of Data Management:

- When there is no established data management practice, corporations will find it difficult to access accurate and relevant data since they are fragmented and dispersed widely
- Conversion of raw data into organized and useful data is another limitation of data management

- Employees should understand the merits of applying data management processes and their impact on their careers and personal lives.

Indexing:

- There are multiple ways to access or replace values in vectors or other data structures. The most common approach is to use “indexing”. This is also referred to as “slicing”.
- R has main 3 indexing operators
- They are as follows :

[ ] = always returns a list with a single element.

[[ ]] = returns a object of the class of item contained in the list.

$ = returns elements from list that have names associated with it, not necessarily same class

- 1.Brackets[ ] are used for indexing
- Ex:

b <- 10:15

- To get 2 and 3 element >b[2:3]
- To print all elements except the second >b[c(1,3:6)] or b[-2]
- We can use an index to change values
- >b[1]<-11 # 11 11 12 13 14 15
- >b[3:6] <- -22 # 11 11 -22 -22 -22 -22
- b[1:3] <- 3 # 3 3 3 -22 -22 -22
- [[ ]] returns a object of the class of item contained in the list
- > dat <- list(str='R', vec=c(1,2,3), bool=TRUE)
- >b=dat[["str"]]
- > class(b) # "character“
- b=dat["str"]
- class(b) # "list"
- $ operator returns elements from list that have names associated with it. It is used to extract or subset a specific part of a data object in R.
- data <- data.frame(x1 = 1:5, x2 = letters[1:5], x3 = 9)
- data$x2 ## [1] "a" "b" "c" "d" "e"
- which(): The which function returns the position of the values in the logical vector.
- which(x,arr.ind = F,useNames = F)

X = An input logical vector.

arr.ind = Returns the array indices if x is an array.

useNames = Indicates the dimension names of an array.

- which(letters=="p,") #return 16, lettters” is a built-in constant with all the 26 letters
- df<- c(5,4,3,2,1)
- #Postion of 5
- which(df==5) #1
- #Position of 1
- which(df==1) #5
- #Position of values greater than 2
- which(df>2) #1 2 3

- match() function in R Language is used to return the positions of the first match of the elements of the first vector in the second vector. If the element is not found, it returns NA.
- match(x1, x2, nomatch)
- # R program to match the vectors
- # Creating vectors
- x1 <- c("a", "b", "c", "d", "e")
- x2 <- c("d", "f", "g", "a", "e", "k")
- # Calling match function
- match(x1, x2, nomatch = "-1") #4 -1 -1 1 5
-

Analysis Techniques:

- A hypothesis is made by the researchers about the data collected for any experiment or data set.
- It a supposition or proposed explanation made on the basis of limited evidence as a starting point for further investigation.
- A hypothesis is an assumption made by the researchers that are not mandatory true.
- A hypothesis is a decision taken by the researchers based on the data of the population collected.
- Hypothesis Testing in R Programming is a process of testing the hypothesis made by the researcher or to validate the hypothesis.
- To perform hypothesis testing, a random sample of data from the population is taken and testing is performed.
- Based on the results of testing, the hypothesis is either selected or rejected. This concept is known as “Statistical Inference”.
- Hypothesis testing can be defined as a statistical tool that is used to identify if the results of an experiment are meaningful or not.
- Hypothesis generation is an educated “guess” of various factors that are impacting the business problem that needs to be solved using machine learnin*g*
- For Ex:It is claimed that a 500 gm sugar packet for a particular brand, say XYZA, contains sugar of less than 500 gm, say around 480gm. Can this claim be taken as truth? How do we know that this claim is true? This is a hypothesis until proved.

Null hypothesis	The weight of the sugar packet is 500 gm. (A well-established fact)
Alternate hypothesis	The weight of the sugar packet is **less than** 500 gm.

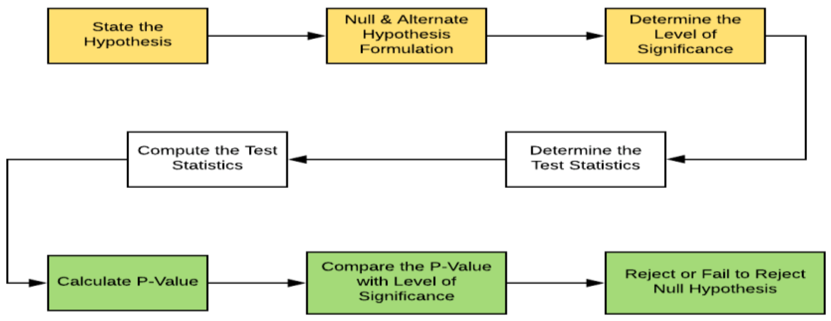
- The four-step process of hypothesis testing,i) **State the Hypotheses ii) Formulate an Analysis Plan** iii) **Analyze Sample Data iv) Interpret Results**
-   **State the Hypotheses** – Stating the null and alternative hypotheses.

i)Alternative Hypothesis (H1) or the research hypothesis states that there is a relationship between two variables (where one variable affects the other)
ii) The Null Hypothesis (H0 OR HA) aims to nullify the alternative hypothesis by implying that there exists no relation between two variables in statistics

These two hypotheses will always be mutually exclusive

H_0 must always contain equality(=).

H_a always contains difference($\neq$, >, <).


State the Hypothesis
Null & Alternate Hypothesis Formulation
Determine the Level of Significance
Determine the Test Statistics
Compute the Test Statistics
Calculate P-Value
Compare the P-Value with Level of Significance
Reject or Fail to Reject Null Hypothesis

- **Formulate an Analysis Plan** – The formulation of an analysis plan is a crucial step
  in this stage.
- **Analyze Sample Data** – Calculation and interpretation of the test statistic, as described in the analysis plan.
- **Interpret Results** – Application of the decision rule described in the analysis plan.
- The level of significance is defined as the criteria or threshold value based on which one can reject the null hypothesis or fail to reject the null hypothesis
- Level of confidence is calculated based on the input samples it might be 95% or 99%
- The **level of significance(alpha)** can take values such as **0.1**, **0.05**, **0.01**. The **most common value** of the level of significance is **0.05. α=1-c where c is** Level of confidence (**loc)**

α=1-0.95=0.05

- Suppose that is alpha = 0.10. You then collect the data and calculate the p-value. If the p-value is greater than alpha, you assume that the null hypothesis is true. If the p-value is less than alpha, you assume that null hypothesis is false.
- The p-value determines whether there is enough evidence to retain the alternative hypothesis or retain the null hypothesis
- alpha is the probability of rejecting the null hypothesis when it was in fact true
- For example, if the significance level(alpha) is set to 0.1 probability, then the sample mean less than 10% will be rejected. Otherwise, the hypothesis is retained to be true.
- A p-value is used in hypothesis testing to help you support or reject the null hypothesis.
- A P-value is the probability of obtaining a sample "more extreme" than the one's observed in your data assuming H_0 is true
- we consider p-value based upon the level of significance of 10%.

  If $p > 0.1$, the null hypothesis will not be considered as an assumption

  If $p > 0.05$ and ≤ 0.1, the null hypothesis has a chance of low assumption.

  If $p > 0.01$ and ≤ 0.05, the null hypothesis is strongly assumed

  If $p \leq 0.01$, the null hypothesis is very significantly assumed.

- The rejection rule of the null hypothesis is as follows:
  If $p < \alpha$, then one must reject the null hypothesis
  If $p > \alpha$, then one should not reject the null hypothesis

UNIT –V

V Testing Techniques Testing: Chi-Square test, t-Test, Z-test, Analysis of variance, Maximum likelihood test, regression, Practice and analysis with R.

Testing Techniques:

Regression:

- **Regression** is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength and character of the relationship between **one dependent variable** (usually denoted by Y) and **a series of other variables** (known as independent variables).
- Statistical learning refers to a vast set of tools for understanding data. These tools can be classified as supervised or unsupervised.
- Supervised statistical learning involves building a statistical model for predicting, or estimating, an output based on one or more inputs.
- Problems of this nature occur in fields as diverse as business, medicine, astrophysics, and public policy
- With unsupervised statistical learning, there are inputs but no supervising output
- It is free ,comparatively expensive statistical tools, such as SAS, STATA, and SPSS.

Regression analysis:

- Sales(in thousands of units) for a particular product as a function of advertising budgets (in thousands of dollars) for TV, radio, and newspaper media
- The predictions made through this plots
- i)Is the relationship linear?
- ii)Which media contribute to sales?
- iii) How strong is the relationship between
  advertising budget and sales?

iv) How accurately can we predict future sales?

- Linear Regression:
- General mathematical equation for a linear regression is
- $y = ax + b$

y is the response variable.
x is the predictor variable.
a and **b** are constants which are called the coefficients.

- y ~ x, it is defining a relationship between input x and output y(formula used in R)
- it is similar to $y = a * x + b$, where 'a' is the slope, and 'b' is the intercept
- ***Regression shows a line or curve that passes through all the data points on a target-predictor graph in such a way that the vertical distance between the data points and the regression line is minimum.***
- It is used principally for prediction, forecasting, time series modeling, and

determining the causal-effect relationship between variables

- Regression analysis is used as statistical tool to establish a relationship model between two variables

    (i) predictor variable whose value is gathered through experiments

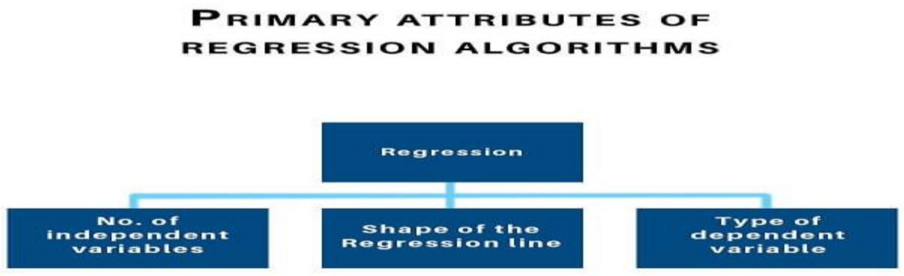
    (ii) response variable whose value is derived from the predictor variable

    In Linear Regression these two variables are related through an equation,
where exponent of both these variables is 1

- linear regression, a very simple approach for supervised learning.
- a linear relationship represents a straight line when plotted as a graph
- A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve
- linear regression is a way for predicting a quantitative response Y on the basis of a sin- regression single predictor variable X.
- The mathematical representation of linear regression is $Y \approx \beta 0 + \beta 1X$
- Where , β0 and β1 are two unknown constants that represent the **intercept** and **slope** terms in the linear model
- β0 and β1 are intercept slope known as the model **coefficients** or **parameters**

Types of Regression:

- ➢ Linear Regression
- ➢ Multiple Linear Regression
- ➢ Logistic Regression
- ➢ Poisson Regression


PRIMARY ATTRIBUTES OF REGRESSION ALGORITHMS
Regression
No. of independent variables
Shape of the Regression line
Type of dependent variable

Linear Regression:

This model consists of a single parameter and a dependent variable ,which is having a linear relationship. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables

Multiple Linear Regression:

Multiple regression analysis is a statistical technique that analyzes the relationship between **two or more variables** and uses the information to estimate the **value of the dependent variables**.

In multiple regression, the objective is to develop a model that describes **a dependent variable y** to **more than one independent** variable.

- The multiple regression equation is given by

$y = a + b\ _1x_1 + b_2 \times_2 + \ldots\ldots + b_k x_k$

- The objective of regression analysis is to model the relationship between a dependent variable and one or more independent variables
- Ex:A gym trainer has collected the data of his client that are coming to his gym and want to observe some things of client that are **health, eating habits**, **the**

weight of the client. This wants to find a relation between these variables.

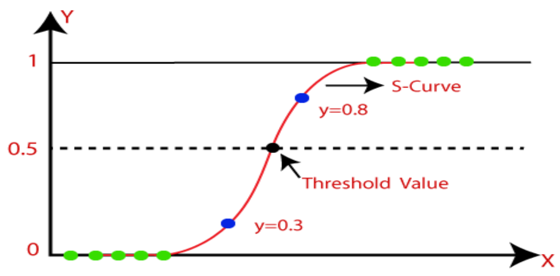
Logistic Regression:

- Logistic Regression is used when the dependent variable(target) is categorical.
- For example, To predict whether an email is spam (1) or (0),Whether the tumor is malignant (1) or not (0)
- Logistic regression predicts the output of a categorical dependent variable.
- Therefore the outcome must be a categorical or discrete value.
- It can be either Yes or No, 0 or 1, true or False, etc.

but instead of giving the exact value as 0 and 1,
it gives the probabilistic values which
lie between 0 and 1.

- The sigmoid function is a mathematical function

used to map the predicted values to probabilities
e=Euler's number(2.71828)



Poisson Regression:

- Poisson Regression involves regression models in which the response variable is in the form of counts and not fractional numbers.
- It is a best method which is used for modeling events where the outcomes are counts
- For example,i) the count of number of births or number of wins in a football match series.ii) the number of people in line at the grocery store iii) The number of phone calls at a call center per minute
- The values of the response variables follow a Poisson distribution.
- The general mathematical equation for Poisson regression is:
- $\log(y) = a + b_1x_1 + b_2x_2 + b_nx_n$

Maximum Likelihood Estimation in R

Maximum Likelihood Estimation (MLE) is a key method in statistical modeling, used to estimate parameters by finding the best fit to the observed data. By looking closely at the data we have, MLE calculates the parameter values that make our observed results most likely based on our model. In this article, we explore how to use MLE with the R Programming Language.

What is Likelihood Estimation?

Likelihood Estimation is a statistical method used to estimate the parameters of a probability distribution or a statistical model based on observed data. Unlike traditional estimation methods that focus on finding the "best-fitting" parameters, likelihood estimation frames the problem in terms of the likelihood function.

What is Maximum Likelihood Estimation (MLE)?

The goal of likelihood estimation is to find the parameter values that maximize the likelihood function. These parameter values are called Maximum Likelihood Estimates (MLEs). Once the MLEs are obtained, they provide estimates of the parameters that best explain the observed data. These estimates can be used for inference, prediction, or further analysis depending on the context of the problem.

θ^MLE=arg⁡max⁡θL(θ|x)*θ*^MLE=argmax*θL*(*θ*|*x*)

Where:

- **?^MLE:** represents the Maximum Likelihood Estimator.
- **?(?|?):** is the likelihood function, which measures the likelihood of the observed data ? for different values of the parameter ?.
- **arg max? :** denotes the value of θ that maximizes the likelihood function.

Now we will Calculate Maximum Likelihood Estimation using optim function in R Programming Language.

```
# Load necessary libraries
library(ggplot2)


# Generate synthetic data from an exponential distribution
set.seed(123)
data <- rexp(100, rate = 0.1)


# Likelihood function for the exponential distribution
log_likelihood <- function(lambda) {
  sum(dexp(data, rate = lambda, log = TRUE))
}


# Maximum Likelihood Estimation using optim
result <- optim(par = 0.1, log_likelihood, method = "Brent", lower = 0.001, upper = 10)
mle_lambda <- result$par


# Print the MLE estimate
cat("Maximum Likelihood Estimate for Lambda:", mle_lambda, "\n")
```

Output:

```
Maximum Likelihood Estimate for Lambda: 10
```

First, we generate synthetic data representing the time taken for a task to complete from an exponential distribution with a known rate parameter of 0.1.

- We define a likelihood function that computes the log likelihood of the data given a lambda value.
- We use the optim() function to find the value of lambda that maximizes the log likelihood.
- We print the Maximum Likelihood Estimate (MLE) for the parameter lambda.

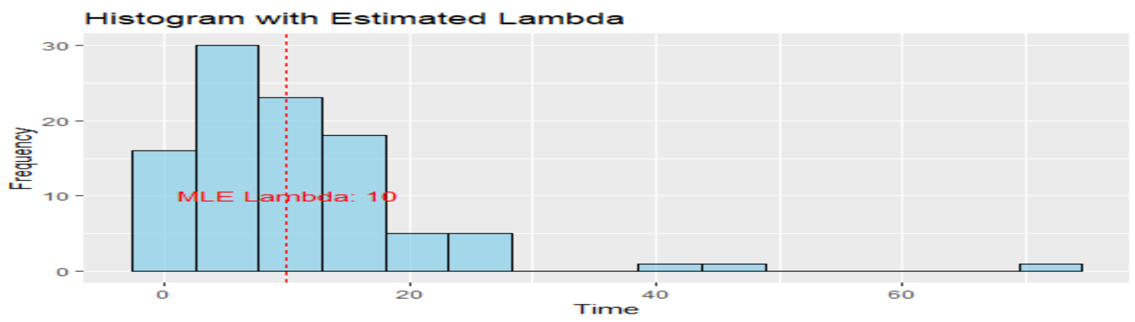
Visualize the Maximum Likelihood Estimate (MLE) on a plot

Now we can plot the histogram of the data along with the estimated data points for Maximum Likelihood Estimate.

R

```
# Plot the histogram of the data with the estimated lambda
ggplot(data = NULL, aes(x = data)) +
  geom_histogram(bins = 15, fill = "skyblue", color = "black", alpha = 0.7) +
  geom_vline(xintercept = mle_lambda, color = "red", linetype = "dashed") +
  annotate("text", x = mle_lambda + 0.1, y = 10,
           label = paste("MLE Lambda:", round(mle_lambda, 2)), color = "red") +
  labs(title = "Histogram with Estimated Lambda", x = "Time", y = "Frequency")
```

Output:



Maximum Likelihood Estimation in R

We generate the histogram of the synthetic data.

- We add a vertical dashed line representing the Maximum Likelihood Estimate (MLE) of the parameter lambda.
- We annotate the plot with the MLE estimate value.
- This visualization allows us to see the MLE estimate visually on the histogram plot of the data.

Chi-Square Test in R

The chi-square test of independence evaluates whether there is an association between the categories of the two variables. There are basically two types of random variables and they yield two types of data: numerical and categorical. In R Programming Language Chi-square statistics is used to investigate whether distributions of categorical variables differ from one another. The chi-square test is also useful while comparing the tallies or counts of categorical responses between two(or more) independent groups.
In R Programming Language, the function used for performing a chi-square test is **chisq.test()**.
Syntax:
chisq.test(data)
Parameters:
data*: data is a table containing count values of the variables in the table.*
We will take the survey data in the **MASS** library which represents the data from a survey conducted on students.

```
# load the MASS package

library(MASS)

print(str(survey))
```

Output:

```
'data.frame':    237 obs. of  12 variables:
 $ Sex   : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 1 2 1 2 2 ...
 $ Wr.Hnd: num  18.5 19.5 18 18.8 20 18 17.7 17 20 18.5 ...
 $ NW.Hnd: num  18 20.5 13.3 18.9 20 17.7 17.7 17.3 19.5 18.5 ...
 $ W.Hnd : Factor w/ 2 levels "Left","Right": 2 1 2 2 2 2 2 2 2 2 ...
 $ Fold  : Factor w/ 3 levels "L on R","Neither",..: 3 3 1 3 2 1 1 3 3 3 ...
 $ Pulse : int  92 104 87 NA 35 64 83 74 72 90 ...
 $ Clap  : Factor w/ 3 levels "Left","Neither",..: 1 1 2 2 3 3 3 3 3 3 ...
 $ Exer  : Factor w/ 3 levels "Freq","None",..: 3 2 2 2 3 3 1 1 3 3 ...
 $ Smoke : Factor w/ 4 levels "Heavy","Never",..: 2 4 3 2 2 2 2 2 2 2 ...
 $ Height: num  173 178 NA 160 165 ...
 $ M.I   : Factor w/ 2 levels "Imperial","Metric": 2 1 NA 2 2 1 1 2 2 2 ...
 $ Age   : num  18.2 17.6 16.9 20.3 23.7 ...
NULL
```

The above result shows the dataset has many Factor variables which can be considered as categorical variables. For our model, we will consider the variables "**Exer**" and "**Smoke**".The Smoke column records the students smoking habits while the Exer column records their exercise level. **Our aim is to test the hypothesis whether the students smoking habit is independent of their exercise level at .05 significance level.**

```
# Create a data frame from the main data set.
stu_data = data.frame(survey$Smoke,survey$Exer)

# Create a contingency table with the needed variables.
stu_data = table(survey$Smoke,survey$Exer)

print(stu_data)
```

Output:

```
      Freq None Some
 Heavy    7    1    3
 Never   87   18   84
 Occas   12    3    4
 Regul    9    1    7
```

And finally we apply the **chisq.test()** function to the contingency table stu_data.

```
# applying chisq.test() function
print(chisq.test(stu_data))
```

Output:

```
     Pearson's Chi-squared test


data:  stu_data
X-squared = 5.4885, df = 6, p-value = 0.4828
```

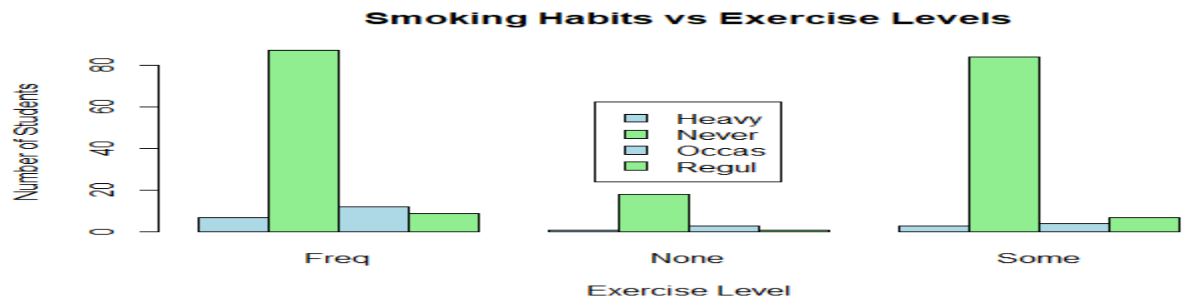
As the p-value 0.4828 is greater than the .05, we conclude that the smoking habit is independent of the exercise level of the student and hence there is a weak or no correlation between the two variables. The complete R code is given below.
So, in summary, it can be said that it is very easy to perform a Chi-square test using R. One can perform this task using **chisq.test()** function in R.

Visualize the Chi-Square Test data

```
# Load required library
library(MASS)

# Print structure of the survey dataset
print(str(survey))

# Create a data frame for smoking and exercise columns
stu_data <- data.frame(survey$Smoke, survey$Exer)
stu_data <- table(survey$Smoke, survey$Exer)

# Print the table
print(stu_data)

# Perform the Chi-Square Test
chi_result <- chisq.test(stu_data)
print(chi_result)

# Visualize the data with a bar plot
barplot(stu_data, beside = TRUE, col = c("lightblue", "lightgreen"),
        main = "Smoking Habits vs Exercise Levels",
        xlab = "Exercise Level", ylab = "Number of Students")
```

```
# Add legend separately
legend("center", legend = rownames(stu_data), fill = c("lightblue", "lightgreen"))
```

Output:



Chi-Square Test in R

In this code we use the MASS library to conduct a Chi-Square Test on the 'survey' dataset, focusing on the relationship between smoking habits and exercise levels.
It creates a contingency table, performs the statistical test, and visualizes the data using a bar plot. The legend is added separately to the top-left corner, distinguishing between different smoking habits with distinct colors.
The code aims to explore and communicate the associations between smoking behavior and exercise practices within the dataset.

Z test in R

Hypothesis testing, also known as significance testing, is a statistical test that is used to conclude the population based on assumption. Here two hypotheses are proposed. One is the null hypothesis, and the other is the alternate hypothesis. For hypothesis testing different tests are used. The tests have been categorized in two ways:

- Parametric Test: These tests make assumptions about the population parameters. Some of the tests are Z Test, F test etc.
- Non-Parametric Tests: These tests do not make any assumptions about the population parameters.

What is Z Test?

Z test is a popular parametric test used for hypothesis testing. Z test is a statistical method used to determine if there is a significant difference between sample and population means or between the means of two samples. It is used when there is a large sample size and the population. It is to be noted that Z Test follows normal distribution. The Z value acts as a threshold. Based on its value it is decided whether to accept the hypothesis or reject the hypothesis. This test is applicable where the sample size is greater than 30.
There are two types of Z tests based on samples:

- One Sample Z-test
- Two Sample Z-test

One Sample Z test

Here Z Test is applicable on one sample that has been taken from the population. The formula is as follows:
Z = \frac{{\bar{X} - \mu}}{{\frac{\sigma}{\sqrt{n}}}}

Here,

- Z denotes the Z value
- \bar{X} is the sample mean
- \mu denotes mean of the population
- \sigma denotes population standard deviation
- n denotes sample size.

Two sample Z test

Here Z Test is applicable on two samples that has been taken from the population. The formula is as follows:
Z = \frac{{\bar{X}_1 - \bar{X}_2}}{{\sqrt{\frac{{s_1^2}}{{n_1}} + \frac{{s_2^2}}{{n_2}}}}}

Here,

- {\bar X_1} and {\bar X_2} are the sample means.
- s1 and s2 are standard deviations of the two samples.
- n1 and n2 are sample sizes of two samples.

Application of Z-test

Z-test is applied when:

1. Population Standard Deviation is Known:
   - We use z-test when, we know the standard deviation of the population and are comparing a sample mean to a population mean or comparing means of two independent samples.
   - If you know the average height of a population and you want to test whether a sample of individuals has a significantly different average height.
2. Large Sample Size:
   - The Z-test is most reliable when dealing with large sample sizes (typically, n > 30 is considered "large").
   - As the sample size increases, the sampling distribution of the sample mean becomes approximately normal, according to the Central Limit Theorem. Therefore, the Z-test becomes more appropriate as the sample size increases.

Z test in R

R is a popular high level programming language used for statistical analysis. It is open-source programming language as it has a huge community and users can contribute to the development as well. It has vast number of packages which allows the data miners to perform statistical analysis and data visualizations in an interactive manner.

The syntax of z- test in R is:

```
z.test(x, y, alternative='two.sided', mu=0, sigma.x=NULL, sigma.y=NULL,conf.level=.95)
```

Now we can conduct one sample test and two sample tests in R.

Here we provide the vector(s) and also provide the value of standard deviation and population mean whose hypothesis is to be tested against. Then we use **z.test** to calculate the z value. This method provides a complete summary of the output.

The one sample test is as follows:

Here,

- **mu** is the population mean under the null hypothesis.
- **sigma.x** is the known population standard deviation.

```
library(BSDA)

# Sample data
sample_data <- c(26, 25, 10, 34, 30, 23, 28, 29, 25, 27)

# One-sample Z-test
z_test <- z.test(sample_data, mu = 24,sigma.x=10)

# Print the result
print(z_test)
```

Output:

```
    One-sample z-Test
data:  sample_data
z = 0.53759, p-value = 0.5909
alternative hypothesis: true mean is not equal to 24
95 percent confidence interval:
 19.50205 31.89795
sample estimates:
mean of x
     25.7
```

The **z.test** function returns a test result object that includes the test statistic, p-value, and other relevant information.

The output of the z test is:

- **Test Statistics (z):** 0.53759
- **P-value:** 0.5909
- **Alternative Hypothesis:** The true mean is not equal to 24.

- **95% Confidence Interval:** The confidence interval for the true mean is given as (19.50205, 31.89795).
- **Sample Estimate (mean of x):** 25.7

The p-value is 0.5909 and the value is greater than the chosen significance level, hence, we will fail to reject the null hypothesis. There is not enough evidence to suggest that the true mean is different from 24 based on your sample data. The 95% confidence interval provides a range of plausible values for the true mean.
Based on the above output it is said that there is not much evidence to reject null hypothesis. So, the null hypothesis is accepted, and the alternate hypothesis is rejected.
Now we will perform two sample Z-Test

```
# Two vectors of sample data
data1 <- c(27, 24, 18, 29, 30,27)
data2 <- c(23, 28, 20, 19, 35,23)

# Two-sample Z-test
z_test_result <- z.test(data1,data2,mu=26,sigma.x=10,sigma.y=15)

# Print the result
print(z_test_result)
```

Output:

```
   Two-sample z-Test
data:  data1 and data2
z = -3.3742, p-value = 0.0007403
alternative hypothesis: true difference in means is not equal to 26
95 percent confidence interval:
 -13.25828  15.59161
sample estimates:
mean of x mean of y
 25.83333  24.66667
```

The output of the two-sample z-test comparing two independent samples:

- **Test Statistic (z):** -3.3742
- **P-value:** 0.0007403
- **Alternative Hypothesis:** The true difference in means is not equal to 26.
- **95% Confidence Interval:** (-13.25828, 15.59161)
- **Sample Estimates:**
    - Mean of Group 1 (data1): 25.83333
    - Mean of Group 2 (data2): 24.66667

From the above output we can see that the z-value is negative, and the p value is very small. So based on the above calculations we can say that there is sufficient evidence to accept null hypothesis. In this case we have to accept alternate hypothesis.

ANOVA (Analysis of Variance) Test in R Programming

ANOVA also known as Analysis of variance is used to investigate relations between categorical variables and continuous variables in the R Programming Language. It is a type of hypothesis testing for population variance. It enables us to assess whether observed variations in means are statistically significant or merely the result of chance by comparing the variation within groups to the variation between groups. The ANOVA test is frequently used in many disciplines, including business, social sciences, biology, and experimental research.

R – ANOVA Test

ANOVA tests may be run in R programming, and there are a number of functions and packages available to do so.
ANOVA test involves setting up:

- **Null Hypothesis:** The default assumption, or null hypothesis, is that there is no meaningful relationship or impact between the variables. It stands for the absence of a population-wide link, difference, or effect. The statement that two or more groups are equal or that the effect size is zero is sometimes expressed as the null hypothesis. The null hypothesis is commonly written as H0.

- **Alternate Hypothesis:** The opposite of the null hypothesis is the alternative hypothesis. It implies that there is a significant relationship, difference, or link among the population's variables. Depending on the study question or the nature of the issue under investigation, it may take several forms. Alternative hypotheses are sometimes referred to as H1 or HA.

ANOVA tests are of two types:

- **One-way ANOVA:** One-way When there is a single categorical independent variable (also known as a factor) and a single continuous dependent variable, an ANOVA is employed. It seeks to ascertain whether there are any notable variations in the dependent variable's means across the levels of the independent variable.

- **Two-way ANOVA:** When there are two categorical independent variables (factors) and one continuous dependent variable, two-way ANOVA is used as an extension of one-way ANOVA. You can evaluate both the direct impacts of each independent variable and how they interact with one another on the dependent variable.

The Dataset

The mtcars(motor trend car road test) dataset is used which consist of 32 car brands and 11 attributes. The dataset comes preinstalled in **dplyr** package in R.
To get started with ANOVA, we need to install and load the **dplyr** package.

Performing One Way ANOVA test in R language

One-way ANOVA test is performed using mtcars dataset which comes preinstalled with dplyr package between disp attribute, a continuous attribute and gear attribute, a categorical attribute.here are some steps.

- Setup Null Hypothesis and Alternate Hypothesis
- H0 = mu = mu01 = mu02(There is no differencebetween average displacement for different gears)
- H1 = Not all means are equal.

```
# Installing the package

install.packages("dplyr")

# Loading the package

library(dplyr)

head(mtcars)
```

Output:

```
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Here we will print top 5 record of our dataset to get an idea about our dataset.

Perform the ANOVA test using aov function.

```
mtcars_aov <- aov(mtcars$disp~factor(mtcars$gear))
summary(mtcars_aov)
```

Output:

```
                    Df Sum Sq Mean Sq F value   Pr(>F)    
factor(mtcars$gear)  2 280221  140110   20.73 2.56e-06 ***
Residuals           29 195964    6757                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- **Df:** The model's degrees of freedom.
- **Sum Sq:** The sums of squares, which represent the variability that the model is able to account for.
- **Mean Sq:** The variance explained by each component is represented by the mean squares.
- **F-value:** It is the measure used to compare the mean squares both within and between groups.

- **Pr(>F):** The F-statistics p-value, which denotes the factors' statistical significance.
- **Residuals:** Relative deviations from the group mean, are often known as residuals and their summary statistics.

Identifier codes: Asterisks (*) are used to show the degree of significance; they stand for p 0.05, p 0.01, and p 0.001, respectively.

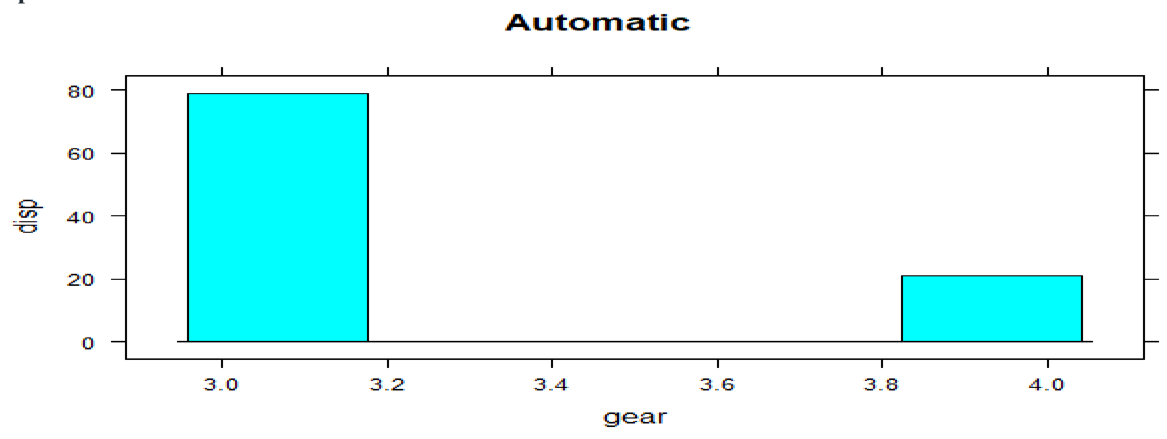
Performing Two Way ANOVA test in R

A two-way ANOVA test is performed using mtcars dataset which comes preinstalled with dplyr package between disp attribute, a continuous attribute and gear attribute, a categorical attribute, am attribute, a categorical attribute.

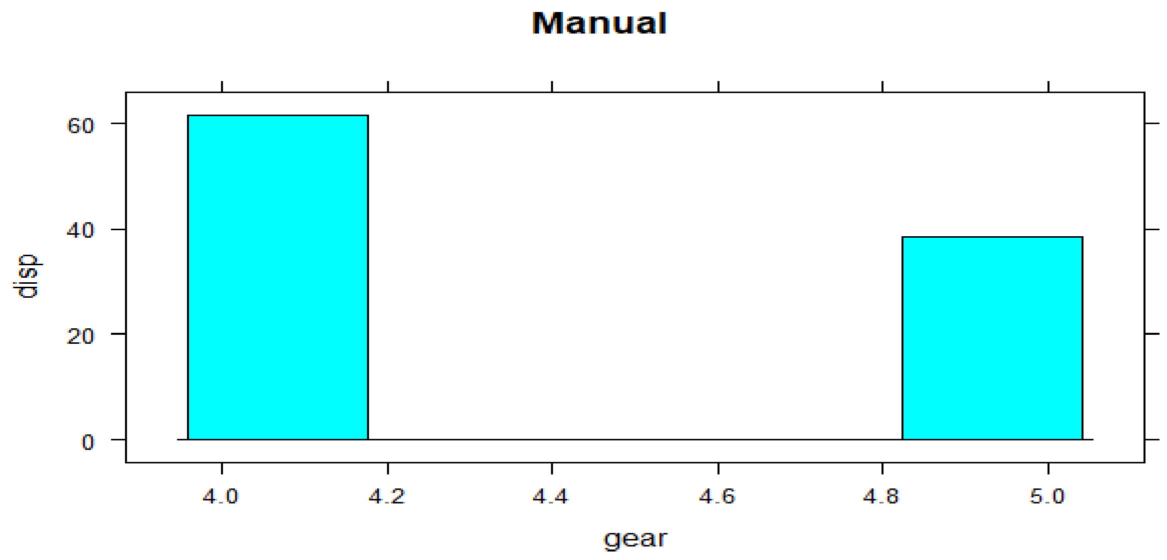
- Setup Null Hypothesis and Alternate Hypothesis
- H0 = mu0 = mu01 = mu02(There is no difference between average displacement for different gear)
- H1 = Not all means are equal

```
# Installing the package
install.packages("dplyr")

# Loading the package
library(dplyr)

# Variance in mean within group and between group
histogram(mtcars$disp~mtcars$gear, subset = (mtcars$am == 0),
          xlab = "gear", ylab = "disp", main = "Automatic")
histogram(mtcars$disp~mtcars$gear, subset = (mtcars$am == 1),
          xlab = "gear", ylab = "disp", main = "Manual")
```

Output:



ANOVA Test in R Programming



ANOVA Test in R Programming

The histogram shows the mean values of gear with respect to displacement. Hear categorical variables are gear and am on which factor function is used and continuous variable is disp.

Calculate test statistics using aov function

```
mtcars_aov2 <- aov(mtcars$disp~factor(mtcars$gear) *
                                        factor(mtcars$am))
summary(mtcars_aov2)
```

Output:

```
                 Df Sum Sq Mean Sq F value   Pr(>F)
factor(mtcars$gear)  2 280221  140110  20.695 3.03e-06 ***
factor(mtcars$am)    1   6399    6399   0.945    0.339
Residuals           28 189565    6770
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The summary shows that the gear attribute is very significant to displacement(Three stars denoting it) and am attribute is not much significant to displacement. P-value of gear is less than 0.05, so it proves that gear is significant to displacement i.e related to each other. P-value of am is greater than 0.05, am is not significant to displacement i.e not related to each other.

Find the best-fit model

We have two different anova models and we will try to find the best fit model based on their AIC score.
The Akaike Information Criterion (AIC), which accounts for the number of predictors, is a gauge of a model’s goodness of fit. It penalizes more intricate models in order to prevent overfitting. Better-fitting models are indicated by lower AIC values.

```
library(AICcmodavg)

model.set <- list(mtcars_aov, mtcars_aov2)
model.names <- c("mtcars_aov", "mtcars_aov2")

aictab(model.set, modnames = model.names)
```

Output:

```
Model selection based on AICc:


            K   AICc Delta_AICc AICcWt Cum.Wt      LL
mtcars_aov  4 379.33       0.00   0.71   0.71 -184.93
mtcars_aov2 5 381.10       1.76   0.29   1.00 -184.39
```

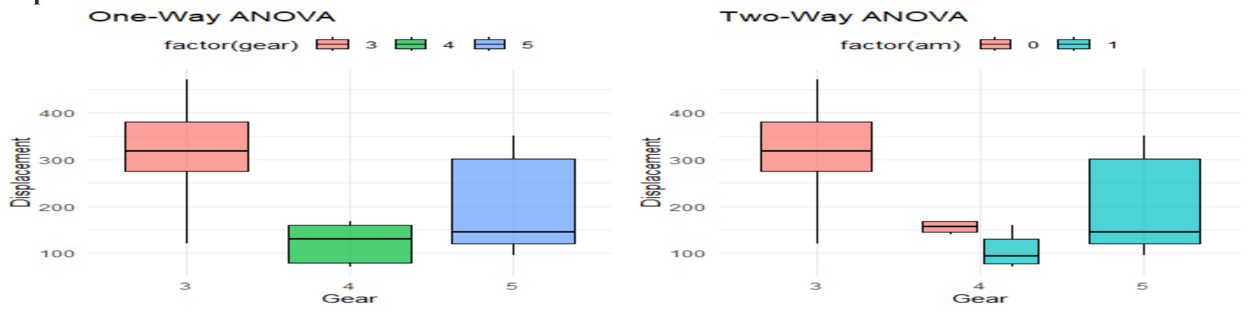
- **AICc (Corrected AIC):** AICc is a measure of how well a statistical model fits the data. Lower AICc values indicate better-fitting models.
- **Delta_AICc (Difference in AICc):** This column represents the difference in AICc between each model and the best-fitting model. Smaller values are better, and a difference of 2 or more is considered significant.
- **AICcWt (AICc Weight):** AICc weight indicates the probability that a given model is the best among the ones considered. In your table, the model with the highest AICc weight (0.71) is considered the most likely best model.
- **Cum.Wt (Cumulative AICc Weight):** This shows the cumulative probability that any model up to a particular row is the best-fitting model.
- **LL (Log-Likelihood):** Log-likelihood measures how well a model explains the observed data. Higher values mean a better fit.

Plot the results in a graph

We will plot both the model together so find out the compression of both the models. here we will use ggplot library and plot the box plot and visualize our both models.

```
# Load required packages
install.packages("ggplot2")
library(ggplot2)

# One-way ANOVA visualization
plot1 <- ggplot(mtcars, aes(x = factor(gear), y = disp, fill = factor(gear))) +
geom_boxplot(color = "black", alpha = 0.7) +
labs(title = "One-Way ANOVA",
              x = "Gear",
              y = "Displacement") +
theme_minimal() +
theme(legend.position = "top")

# Two-way ANOVA visualization
plot2 <- ggplot(mtcars, aes(x = factor(gear), y = disp, fill = factor(am))) +
geom_boxplot(color = "black", alpha = 0.7) +
labs(title = "Two-Way ANOVA",
              x = "Gear",
              y = "Displacement") +
theme_minimal() +
theme(legend.position = "top")

# Combine the plots for comparison
library(gridExtra)
grid.arrange(plot1, plot2, ncol = 2)
```

Output:



The box plots visually compare the displacement (disp) distribution across different gear levels for both one-way and two-way ANOVA models. In the one-way ANOVA, each box represents a gear level, showcasing the variability in displacements.
The two-way ANOVA extends this comparison, incorporating the additional factor (am), providing a more detailed insight into how both factors collectively influence displacement. The plots help discern any notable differences or patterns in dispersion, aiding in the interpretation of model effects on the response variable.

Results

We see significant results from boxplots and summaries.

- Displacement is strongly related to Gears in cars i.e displacement is dependent on gears with p < 0.05.
- Displacement is strongly related to Gears but not related to transmission mode in cars with p 0.05 with am.

T-Test Approach in R Programming

We will be trying to understand the T-Test in R Programming with the help of an example. Suppose a businessman with two sweet shops in a town wants to check if the average number of sweets sold in a day in both stores is the same or not.
So, the businessman takes the average number of sweets sold to 15 random people in the respective shops. He found out that the first shop sold 30 sweets on average whereas the second shop sold 40. So, from the owner's point of view, the second shop was doing better business than the former. But the thing to notice is that the data set is based on a mere number of random people and they cannot represent all the customers. This is where T-testing comes into play it helps us to understand whether the difference between the two means is real or simply by chance.

Mathematically, what the t-test does is, take a sample from both sets and establish the problem assuming a null hypothesis that the two means are the same.

Classification of T-tests

- One Sample T-test
- Two sample T-test
- Paired sample T-test

One Sample T – Test Approach

The One-Sample T-Test is used to test the statistical difference between a sample mean and a known or assumed/hypothesized value of the mean in the population.

So, for performing a one-sample t-test in R, we would use the syntax t.test(y, mu = 0) where x is the name of the variable of interest and mu is set equal to the mean specified by the null hypothesis.

For Example:

```
set.seed(0)
sweetSold <- c(rnorm(50, mean = 140, sd = 5))

# mu=The hypothesized mean difference between the two groups.
t.test(sweetSold, mu = 150)
```

Output:

```
   One Sample t-test


data:  sweetSold

t = -15.249, df = 49, p-value < 2.2e-16

alternative hypothesis: true mean is not equal to 150

95 percent confidence interval:

 138.8176 141.4217
```

```
sample estimates:
mean of x
 140.1197
```

- t = -15.249, df = 49, and a 2.2e-16 p-value: provides the p-value, degrees of freedom (df), and test statistic (t). The computed t-value in this instance is -15.249, there are 49 degrees of freedom, and the p-value is very small (2.2e-16), indicating strong evidence that the null hypothesis is false.
- The true mean is not equal to 150, as an alternative explains the alternative theory, which contends that the population's actual mean is not 150.
- The confidence interval, which ranges from 138.8176 to 141.4217, shows that there is a 95% chance that the genuine population mean is located between those two numbers.
- provides the sample estimate, in this example the sample mean (x) of 140.1197, or "sample estimates: mean of x 140.1197."

Two sample T-Test Approach

It is used to help us to understand whether the difference between the two means is real or simply by chance.

The general form of the test is t.test(y1, y2, paired=FALSE). By default, R assumes that the variances of y1 and y2 are unequal, thus defaulting to Welch's test. To toggle this, we use the flag var.equal=TRUE.

For Example:

```
set.seed(0)

shopOne <- rnorm(50, mean = 140, sd = 4.5)
shopTwo <- rnorm(50, mean = 150, sd = 4)

t.test(shopOne, shopTwo, var.equal = TRUE)
```

Output:

```
   Two Sample t-test


data:  shopOne and shopTwo
t = -13.158, df = 98, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -11.482807  -8.473061
sample estimates:
mean of x mean of y
 140.1077  150.0856
```

- Sample estimates: 140.1077 for the mean of x and 150.0856 for the mean of y the sample means (x and y), which are the sample estimates. In this instance, shopOne's mean is 140.1077, whereas shopTwo's mean is 150.0856.

Paired Sample T-test

This is a statistical procedure that is used to determine whether the mean difference between two sets of observations is zero. In a paired sample t-test, each subject is measured two times, resulting in pairs of observations.

The test is run using the syntax t.test(y1, y2, paired=TRUE)

For Example:

```
set.seed(2820)

sweetOne <- c(rnorm(100, mean = 14, sd = 0.3))
sweetTwo <- c(rnorm(100, mean = 13, sd = 0.2))

t.test(sweetOne, sweetTwo, paired = TRUE)
```

Output:

```
   Paired t-test


data:  sweetOne and sweetTwo
t = 29.31, df = 99, p-value < 2.2e-16
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 0.9892738 1.1329434
sample estimates:
mean difference
      1.061109
```

estimations from samples: mean difference The sample estimate, in this case, the mean difference between the paired samples, is given by the number 1.061109. A mean difference of 1.061109 is estimated.

Differences between one-sample, two-sample, and paired-sample t-tests:

One-sample t-test	Two-sample t-test	Paired sample t-test
Purpose: Determines whether a single sample's mean deviates considerably from a given population mean.	**Purpose:** Determines whether there is a substantial difference between the means of two independent groups.	**Purpose:** Determines whether the means of two connected or paired samples differ significantly from one another.
Data: Analyses a single set of measurements or observations.	**Data:** Compares two distinct groups' or samples' means.	**Data:** Analyses the identical group or set of observations made under two distinct situations or at two different times.
Hypotheses: Tests whether a population mean is hypothesized to be significantly different from	**Hypotheses:** Determines whether there is a significant difference between the two groups' means.	**Hypotheses:** Tests hypotheses to determine if the mean difference between the paired samples differs noticeably from

One-sample t-test	**Two-sample t-test**	**Paired sample t-test**
the sample mean.		zero.
Assumptions: Assumes that observations are independent and that the data is regularly distributed.	**Assumptions:** Assumes that observations are unrelated to one another, that data in each group is normally distributed, and that the variances of the two groups may or may not be equal.	**Assumptions**: Assumes that the paired observations are dependent or matched pairs, that the differences have a fixed variance, and that the differences are normally distributed.
Example: Examining whether a class's average test scores considerably deviate from the average test score for the country, for instance.	**Example:** Using the average heights of male and female people to determine whether there is a noticeable difference between the two groups.	**Example:** Comparing measures taken from the same group of people before and after a new treatment can help determine whether it has a discernible impact.