**R Programming**

**Day 1 [26/06/2023]**

Data Structures –

1. Vectors
2. Metrics
3. Data frames
4. Lists
5. Arrays

Variables –

It can be a number, string, logical also.

Numbers 🡪

1. Num
2. Int

String 🡪

1. Char

Priority 🡪

1. Brackets
2. Exponential
3. Division
4. Multiplication
5. Addition and Subtraction

Note 🡪 If you have to get remainder, you have to use modulus operator twice.

Ex. 12 %% 4 = 1

If you mix the different types of data into a list, it will be converted to a string.

**Vectors 🡪**

1. To create a vector 🡪

x = c(5,9,8,3,6)

c -> stands for concatenation.

1. Index value in vectors starts with 1 and not with 0.
2. To append the value into a vector.

x = append(x,10)

y = c(4,8,12)

y

[1] 4 8 12

x = c(x,y)

x

[1] 5 9 8 3 6 10 4 8 12

1. If you want to add a value at a specific location ->

> x = append(x,15,6)

append(vector, value, position)

position - > It will append the value after the position which you have mentioned.

> x

[1] 5 9 8 3 6 10 15 4 8 12

> x = append(x,c(1,2,3),5)

> x

[1] 5 9 8 3 6 1 2 3 10 15 4 8 12

1. To modify the value from a vector ->

> x[3] = 4

> x

[1] 5 9 4 3 6 1 2 3 10 15 4 8 12

1. To delete a value from a vector ->

> x = x[-2]

> x

[1] 5 4 3 6 1 2 3 10 15 4 8 12

x[-value] 🡪

Whatever value is mentioned, it will delete a value at that particular index and other elements will be left shifted to 1.

1. To print limited values from the vector 🡪

> x[c(7,3,4)]

[1] 3 3 6

1. To modify the multiple values at a time 🡪

> x[c(2,5)] = c(9,2)

> x

[1] 5 9 3 6 2 2 3 10 15 4 8 12

1. Aggregate function 🡪

> sum(x)

[1] 79

> prod(x)

[1] 559872000

> min(x)

[1] 2

> max(x)

[1] 15

> cummin(x)

[1] 5 5 3 3 2 2 2 2 2 2 2 2

> cummax(x)

[1] 5 9 9 9 9 9 9 10 15 15 15 15

> cumsum(x)

[1] 5 14 17 23 25 27 30 40 55 59 67 79

1. Factorial 🡪 It will return factorial of vector or any particular number.

> factorial(5)

[1] 120

1. Some important cases

> a = c(4,7,6,2)

> b = c(2,1,4,9)

> c = c(1,2)

> d = c(3,4,5)

**Case no 1 ->**

Operating vector with a scaler quantity.

Same operation will be done across the vector.

> a + 5

[1] 9 12 11 7

**Case no 2 ->**

when vectors with equal length added together, it will get added with same index value.

> a + b

[1] 6 8 10 11

**Case no 3 ->**

When you operate with 2 vectors, with different length, then smaller size of vector must be required to be multiple of bigger vector size.

> a + c

[1] 5 9 7 4

It is also called as recycling a vector.

**Case no 4 ->**

When you operate with 2 vectors with different length and smaller size of vector is not the multiple of bigger size of vector. Then it will generate error.

> a + d

[1] 7 11 11 5

Warning message:

In a + d : longer object length is not a multiple of shorter object length.

1. If you want to print the values from the given range

Method 1 –

> x = 8:15

> x

[1] 8 9 10 11 12 13 14 15

> 6:20

[1] 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Method 2 –

> seq(4,20)

[1] 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

1. Print the values from the given range with any particular step–

> seq(4,20,3)

[1] 4 7 10 13 16 19

> seq(4,20,4)

[1] 4 8 12 16 20

> seq(40,6,-3)

[1] 40 37 34 31 28 25 22 19 16 13 10 7

1. Print the values from the given range with the given length size ->

Here, gap / step is done by R automatically.

> z = seq(4,20,length.out=8)

> z

[1] 4.000000 6.285714 8.571429 10.857143 13.142857 15.428571

[7] 17.714286 20.000000

> z[4] - z[3]

[1] 2.285714

> z = seq(4,20,length.out=9)

> z

[1] 4 6 8 10 12 14 16 18 20

> z[4] - z[3]

[1] 2

1. To sort elements in ascending or descending order.

> sort(z)

[1] 4 6 8 10 12 14 16 18 20

> sort(z, decreasing = T)

[1] 20 18 16 14 12 10 8 6 4

1. To repeat values in a vector. ->

> rep(c(5,4,7),times = 5)

[1] 5 4 7 5 4 7 5 4 7 5 4 7 5 4 7

> rep(c(1,2,3),each = 5)

[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3

> rep(c(1,2,3),times = c(5,6,7))

[1] 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 3

1. Salary example ->

Method 1🡪

> hours = c(5,4,3,6,5,4,6,2,4,5,6,5,6,4)

> #1000 per hour for weekdays and 1500 for weekend

> salary = c(1000,1000,1000,1000,1000,1500,1500)

> hours \* salary

[1] 5000 4000 3000 6000 5000 6000 9000 2000 4000 5000 6000 5000

[13] 9000 6000

> sum(hours \* salary)

[1] 75000

Method 2 🡪

> pay = rep(c(1000,1500),times = c(5,2))

> pay

[1] 1000 1000 1000 1000 1000 1500 1500

> sum(hours\*pay)

[1] 75000

Method 3 🡪

> sum(hours \* rep(c(1000,1500),times=c(5,2)))

[1] 75000

1. To print the length of the characters from all elements from any particular vector.

> nchar(places)

1. To find the index of given substring from vector

> grep('bad',places)

[1] 4 6

1. To print the values having mentioned substring into the vector.

> grep('bad',places,value = T)

[1] "Hydrabad" "Allahabad"

1. Substring – extract a string from a given location.

> substr(places,3,5)

[1] "mba" "ne" "vi " "dra" "ngl" "lah"

1. To print the values having mentioned substring / char / blank space into the vector.

In this following command we want value with blank space.

> grep(' ',places, value = T)

[1] "Navi Mumbai"

1. To change the values from the elements with particular elements.

> gsub('bad','baad',places)

[1] "Mumbai" "Pune" "Navi Mumbai" "Hydrabaad" "Banglore"

[6] "Allahabaad"

1. Indexes get printed where condition gets true 🡪

This command will return the **index**, where value is greater than or equal to 4.

> x = c(4,8,7,3,25,9,1,6,3)

> x = c(4,8,7,3,25,9,1,6,3)

> which(x>=4)

[1] 1 2 3 5 6 8

1. Values will get printed where condition gets true🡪

This command will return the **values**, where value is greater than or equal to 4.

> x = c(4,8,7,3,25,9,1,6,3)

> x[which(x>=4)]

[1] 4 8 7 25 9 6

1. **AND** and **OR** operations
2. AND operation will return TRUE, only if there are both the TRUE values, otherwise it will return false.

> a & b

[1] TRUE

> a & c

[1] FALSE

1. OR operation will return TRUE, only if there are at least one of the value is TRUE value, otherwise it will return false.

> a | c

[1] TRUE

1. Find the values which are greater than 4 and not divisible by 2.

So there are 3 different methods are as follows 🡪

> x[which(x>=4 & x%%2!=0)]

[1] 7 25 9

> x[which(x>=4 & x%%2==1)]

[1] 7 25 9

> x[which(x>=4 & !x%%2==0)]

[1] 7 25 9

1. How you can break a string.

> strsplit(places[1],'')

[[1]]

[1] "M" "u" "m" "b" "a" "i"

1. Splitting a element into two parts 🡪

> strsplit(places[3],' ')[[1]]

[1] "Navi" "Mumbai"

> strsplit(places[3],' ')

[[1]]

[1] "Navi" "Mumbai"

1. Collapse 🡪

> x1 = c('A','B','C')

> x2 = c(1,2,3)

> paste(x1)

[1] "A" "B" "C"

> paste(x1, collapse = '')

[1] "ABC"

> d = strsplit('vaibhav','')

> d = strsplit('vaibhav','')[[1]]

> d

[1] "v" "a" "i" "b" "h" "a" "v"

> paste(d,collapse = '')

[1] "vaibhav"

1. Collapse with separator🡪

> paste(x1,x2,sep = '-')

[1] "A-1" "B-2" "C-3"

> paste(x1,x2,sep = '')

[1] "A1" "B2" "C3"

> paste(x1,x2,sep = '',collapse = '\*\*\*')

[1] "A1\*\*\*B2\*\*\*C3"

> paste(x1,x2,sep = '',collapse = '')

[1] "A1B2C3"

> paste(x1,x2,sep = '-',collapse = '')

[1] "A-1B-2C-3"

1. Fd

**Matrix 🡪**

1. **To create a matrix 🡪**

It will create a matrix with elements 1 to 24 and columns will be 6 as mentioned.

> mat1 = matrix(1:24, ncol = 6)

> mat1

[,1] [,2] [,3] [,4] [,5] [,6]

[1,] 1 5 9 13 17 21

[2,] 2 6 10 14 18 22

[3,] 3 7 11 15 19 23

[4,] 4 8 12 16 20 24

1. It will create a matrix with elements 1 to 24 and columns will be 3 as mentioned.

> mat1 = matrix(1:24, ncol = 3)

> mat1

[,1] [,2] [,3]

[1,] 1 9 17

[2,] 2 10 18

[3,] 3 11 19

[4,] 4 12 20

[5,] 5 13 21

[6,] 6 14 22

[7,] 7 15 23

[8,] 8 16 24

1. It will create a matrix with elements 1 to 24 and rows will be 3 as mentioned.

> mat1 = matrix(1:24, nrow = 3)

> mat1

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

[1,] 1 4 7 10 13 16 19 22

[2,] 2 5 8 11 14 17 20 23

[3,] 3 6 9 12 15 18 21 24

1. Create a matrix using vectors.

> a = c(5,8,7,6)

> b = c(1,4,3,9)

> c = c(4,1,9,2)

1. It will create a matrix as row bind

> mat1 = rbind(a,b,c)

> mat1

[,1] [,2] [,3] [,4]

a 5 8 7 6

b 1 4 3 9

c 4 1 9 2

1. It will create a matrix as column bind

> mat1 = cbind(a,b,c)

> mat1

a b c

[1,] 5 1 4

[2,] 8 4 1

[3,] 7 3 9

[4,] 6 9 2

1. To print the particular element from a matrix

> mat1[2,3]

c

1

1. To modify the particular element from a matrix

> mat1[2,3] = 7

> mat1

a b c

[1,] 5 1 4

[2,] 8 4 7

[3,] 7 3 9

[4,] 6 9 2

1. Print second row from given matrix

> mat1[2,]

a b c

8 4 7

1. Print second column from given matrix

> mat1[,2]

[1] 1 4 3 9

1. To print the multiple rows from a given matrix.

It will print first and third rows from a given matrix.

> mat1[c(2,4),]

a b c

[1,] 8 4 7

[2,] 6 9 2

1. To print the multiple columns from a given matrix.

It will print first and third columns from a given matrix.

> mat1[,c(1,3)]

a c

[1,] 5 4

[2,] 8 7

[3,] 7 9

[4,] 6 2

1. You can print the columns of matrix by using their name also.

> mat1[,c('a','b')]

a b

[1,] 5 1

[2,] 8 4

[3,] 7 3

[4,] 6 9

1. You can also add characters into a vector.

> mat1 = cbind(mat1,d = c('a','b','c','d'))

> mat1

a b c d

[1,] "5" "1" "4" "a"

[2,] "8" "4" "7" "b"

[3,] "7" "3" "9" "c"

[4,] "6" "9" "2" "d"

1. Create data frame

> names = c('Akhil','Pankaj','Rahul')

> inst = c('IITM','MU','SPPU')

> marks = c(69,79,89)

> student\_df = data.frame(names,inst,marks)

> student\_df

names inst marks

1 Akhil IITM 69

2 Pankaj MU 79

3 Rahul SPPU 89

Printing third column of marks ->

> str(student\_df[,3])

num [1:3] 69 79 89

1. You can print and modify single or multiple values of rows and columns etc.

> student\_df[2,3]

[1] 79

> student\_df[2,3] = 99

> student\_df[2,3]

[1] 99

> student\_df

names inst marks

1 Akhil IITM 69

2 Pankaj MU 99

3 Rahul SPPU 89

1. You can modify single / multiple column names at a time.

> student\_df = data.frame(names,college=inst,marks)

> student\_df

names college marks

1 Akhil IITM 69

2 Pankaj MU 79

3 Rahul SPPU 89

1. You can print the column names.

> colnames(student\_df)

[1] "names" "college" "marks"

1. You can print any of the column in different ways.

> student\_df[,2]

[1] "IITM" "MU" "SPPU"

> student\_df[,'college']

[1] "IITM" "MU" "SPPU"

> student\_df$college

[1] "IITM" "MU" "SPPU"

> student\_df$names

[1] "Akhil" "Pankaj" "Rahul"

> student\_df$marks

[1] 69 79 89

1. To install the readxl package.

> install.packages('readxl')

1. Reading library of readxl packages.

> library(readxl)

1. Create a new data frame.

> new\_df = data.frame(a = c('Sachin','Virat'), b = c('UMB','IITM'), c = c(76,82))

> new\_df

a b c

1 Sachin UMB 76

2 Virat IITM 82

1. Change the column names of data frame same as another data frame.

> colnames(new\_df) = colnames(student\_df)

> new\_df

names college marks

1 Sachin UMB 76

2 Virat IITM 82

1. Merge this 2 data frames by using row bind.

> student\_df = rbind(student\_df,new\_df)

> student\_df

names college marks

1 Akhil IITM 69

2 Pankaj MU 79

3 Rahul SPPU 89

4 Sachin UMB 76

5 Virat IITM 82

1. Create another vector.

> d = c('Infosys','Wipro','TCS','AmDocs','TATA')

1. Append this vector to the data frame.

> temp = student\_df

> temp[,4] = d

> temp

names college marks V4

1 Akhil IITM 69 Infosys

2 Pankaj MU 79 Wipro

3 Rahul SPPU 89 TCS

4 Sachin UMB 76 AmDocs

5 Virat IITM 82 TATA

1. Change the column name from V4 to the ‘company’.

> colnames(temp)[4] = 'company'

> temp

names college marks company

1 Akhil IITM 69 Infosys

2 Pankaj MU 79 Wipro

3 Rahul SPPU 89 TCS

4 Sachin UMB 76 AmDocs

5 Virat IITM 82 TATA

1. Delete ‘company’ column from the temp data frame.

> temp = temp[,-4]

> temp

names college marks

1 Akhil IITM 69

2 Pankaj MU 79

3 Rahul SPPU 89

4 Sachin UMB 76

5 Virat IITM 82

1. Print the vector of company.

> d

[1] "Infosys" "Wipro" "TCS" "AmDocs" "TATA"

1. Again appending company vector to the temp data frame by using cbind().

> temp = cbind(temp,company = d)

> temp

names college marks company

1 Akhil IITM 69 Infosys

2 Pankaj MU 79 Wipro

3 Rahul SPPU 89 TCS

4 Sachin UMB 76 AmDocs

5 Virat IITM 82 TATA

**Day 2 [27/06/2023]**

1. While loop ex – credit card

Till your credit limit hits to the maximum limit you can use it for any kind of purposes.

1. Local variable is not allowed to use outside of the block.
2. Global variable can be use inside of the block.
3. By using return statement, you can make local variable to global variable.
4. Load data set.

> getwd()

[1] "C:/Users/vaibh/OneDrive/Documents"

> setwd('D:/DBDA/R Programming')

> df = read\_excel('ERPData.xlsx')

> df.head()

1. Printing ‘Quantity’ column where location is ‘MWH-2’.

> df[which(df$Location=='MWH-2'),3]

# A tibble: 9 × 1

Quantity

*<dbl>*

1 27

2 29

3 10

4 87

5 31

6 28

7 31

8 69

9 85

1. Printing whole data frame where location is ‘MWH-2’.

> df[which(df$Location=='MWH-2'),]

# A tibble: 9 × 3

MaterialID Location Quantity

*<chr>* *<chr>* *<dbl>*

1 LXCV-21 MWH-2 27

2 TMI-43T MWH-2 29

3 GCVB-79 MWH-2 10

4 GCVB-79 MWH-2 87

5 GCVB-79 MWH-2 31

6 GCVB-79 MWH-2 28

7 SDRT-67 MWH-2 31

8 DDBN-89 MWH-2 69

9 AXCP-78 MWH-2 85

1. Printing columns where location is ‘MWH-2’ except location.

> df[which(df$Location=='MWH-2'),1,3]

[1] "LXCV-21" "TMI-43T" "GCVB-79" "GCVB-79" "GCVB-79" "GCVB-79"

[7] "SDRT-67" "DDBN-89" "AXCP-78"

1. Printing data frame where location is ‘MWH-2’ except location.

> df[which(df$Location=='MWH-2'),-2]

# A tibble: 9 × 2

MaterialID Quantity

*<chr>* *<dbl>*

1 LXCV-21 27

2 TMI-43T 29

3 GCVB-79 10

4 GCVB-79 87

5 GCVB-79 31

6 GCVB-79 28

7 SDRT-67 31

8 DDBN-89 69

9 AXCP-78 85

1. Prints data where location is ‘NWH-2’ and quantity greater than equal to 50.

Method 1

> subset(df,df$Location=='MWH-2' & df$Quantity>=50)

# A tibble: 3 × 3

MaterialID Location Quantity

*<chr>* *<chr>* *<dbl>*

1 GCVB-79 MWH-2 87

2 DDBN-89 MWH-2 69

3 AXCP-78 MWH-2 85

Method 2 –

> df[which((df$Location=='MWH-2') & (df$Quantity >= 50)),]

# A tibble: 3 × 3

MaterialID Location Quantity

*<chr>* *<chr>* *<dbl>*

1 GCVB-79 MWH-2 87

2 DDBN-89 MWH-2 69

3 AXCP-78 MWH-2 85

1. Adding one extra column where quantity is greater than equal to 50 are scrap and rest are useful.

mydf = function() {

x = df$Quantity

y = c()

for (ctr in 1 : nrow(df)) {

if (x[ctr] < 50) {

y[ctr] = 'scrap'

}

else {

y[ctr] = 'useful'

}

}

df$status = y

print(head(df))

print(sum(df$Quantity[df$status=='scrap']))

print(sum(df$Quantity[df$status=='useful']))

}

1. Sorting elements in vector.

> x = c(4,8,7,2,3,6,9,6)

> sort(x)

[1] 2 3 4 6 6 7 8 9

1. Sorting elements in vector and return its index (index of element).

> order(x)

[1] 4 5 1 6 8 3 2 7

1. Sorting elements in descending order in vector and return its index (index of element).

> order(x, decreasing = T)

[1] 7 2 3 6 8 1 5 4

1. V
2. To install packages

> install.packages('dplyr')

> install.packages('tidyverse')

1. S
2. D

Day 3 [28/06/2023]

1. Create one vector having grade values.

> grades = c('A','B','A','C','C','B','A','A','B','D','B','A','A','A','B','C','B','A','A','B')

> length(grades)

[1] 20

> unique(grades)

[1] "A" "B" "C" "D"

> grd\_count = table(grades)

> grd\_count

grades

A B C D

9 7 3 1

> grd\_prop = grd\_count / length(grades)

> grd\_prop

grades

A B C D

0.45 0.35 0.15 0.05

1. Plot graph using available data.

> grd\_pct = grd\_prop\*100

> grd\_pct

grades

A B C D

45 35 15 5

> cumsum(grd\_pct)

A B C D

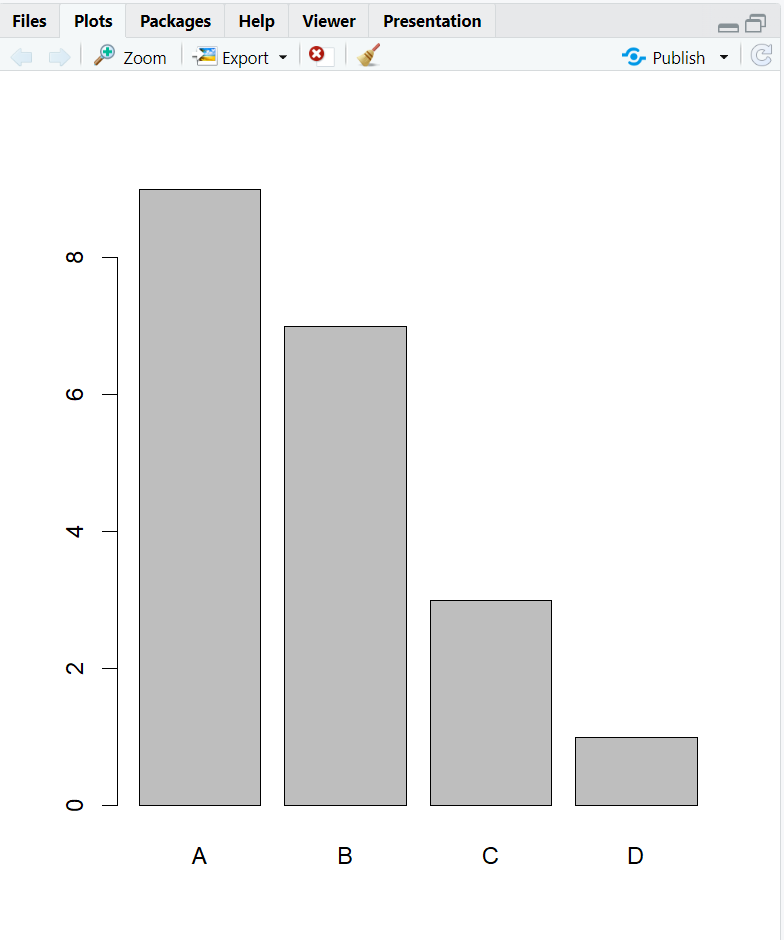
45 80 95 100

> cumsum(grd\_count)

A B C D

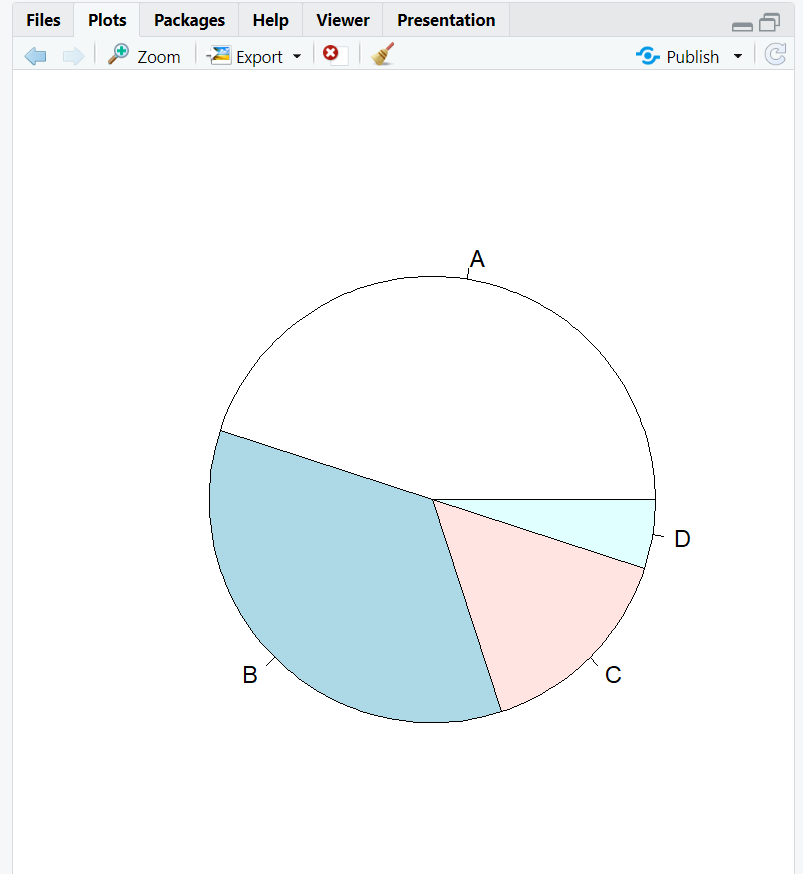
9 16 19 20

> barplot(grd\_count)



1. Plot graph using available data.

> pie(grd\_count)



1. D
2. d

> barplot(table(erupt\_cat\_name))



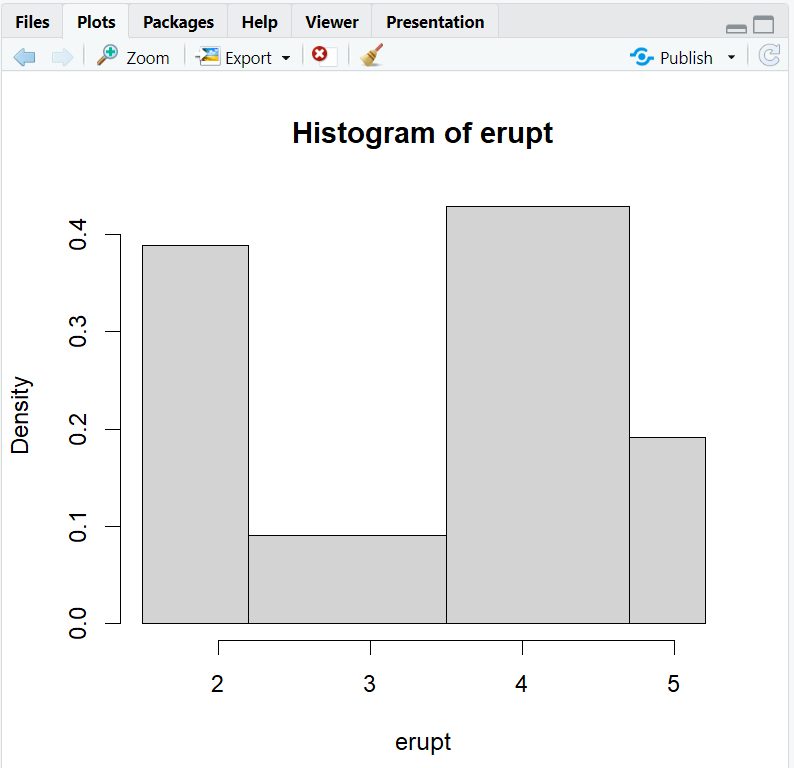
1. d

> barplot(sort(table(erupt\_cat\_name),decreasing = T))



1. Histogram

> hist(erupt,b)



1. D
2. Mean, Median, Mode.

> mean(erupt)

[1] 3.487783

> median(erupt)

[1] 4

> erupt\_count = table(erupt)

> head(erupt\_count)

erupt

1.6 1.667 1.7 1.733 1.75 1.783

1 1 1 1 6 2

> which(erupt\_count == max(erupt\_count))

1.867 4.5

11 99

> erupt\_count[c(11,99)]

erupt

1.867 4.5

8 8

1. D
2. Installing ‘moments’ packages.

> install.packages('moments')

> library(moments)

1. Calculating skewness and kurtosis.

> skewness(erupt)

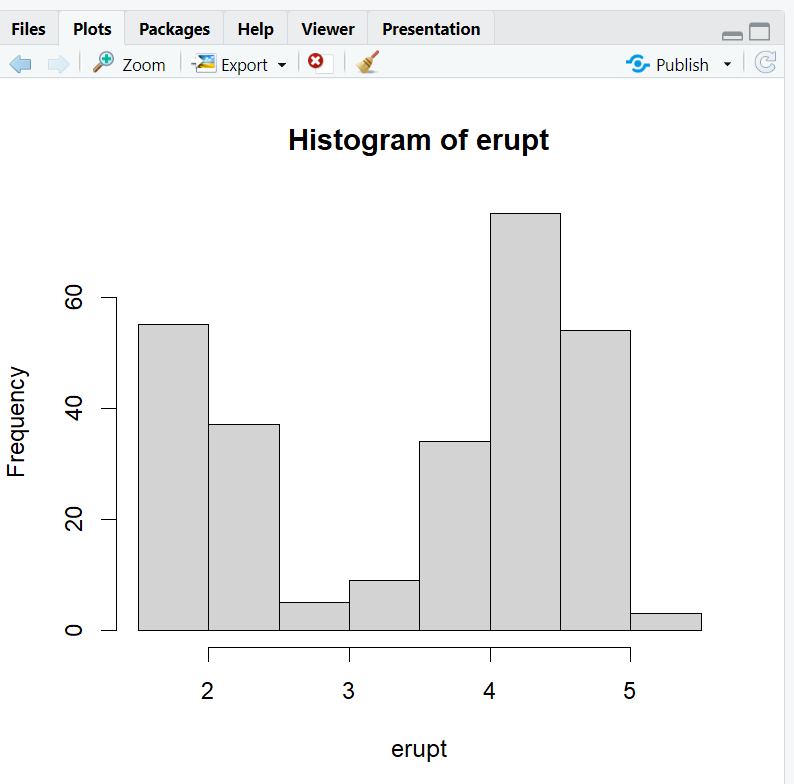
[1] -0.415841

> kurtosis(erupt)

[1] 1.4994

1. Histogram.

> hist(erupt)



1. Probability distribution 🡪 Binomial distribution

> dbinom(7,10,0.7) #exact 7

[1] 0.2668279

> pbinom(7,10,0.7) #max 7

[1] 0.6172172

> 1 - pbinom(6,10,0.7) #min 7

[1] 0.6496107

> pbinom(6,10,0.7,lower.tail = F) #min 7

[1] 0.6496107

1. Probability Distribution 🡪 Poisson Distribution

> #avg occurances for accide is 25 per day

> #prob of exactly 17 accidents, upto 17 and min 17

> dpois(17,25) #exact 17

[1] 0.02272739

> ppois(17,25) #upto 17

[1] 0.06047504

> 1 - ppois(17,25) #min 17

[1] 0.939525

1. Probability distribution 🡪 hypergeometric

> #in the box of 100 LEDs, we will allow 10 to be defective

> #take a sample of 10 LEDs, what probability upto 1 will be defective LED

> phyper(1,100,10,10)

[1] 2.134436e-11

1. Exponential distribution

> #200 break downs are reported in a year

> (24\*365) / 200

[1] 43.8

> #average time between two break downs

> #client is in the my office for 60 hours. what is the probability that he sees the breakdown

> pexp(60,1/43.8)

[1] 0.7458582

> pexp(80,1/43.8)

[1] 0.8390214

1. D
2. Sample T Test -> Left Tail Test

> erupt = faithful$eruptions

> s1 = erupt[35:80]

> #mean of s1 is atleast 3.8

> #Ho : mean(s1) >= 3.8

> #Ha : mean(s1) < 3.8 --> left tail test

> t.test(s1,mu=3.8,alternative = 'less')

One Sample t-test

data: s1

t = -1.7325, df = 45, p-value = 0.04502

alternative hypothesis: true mean is less than 3.8

95 percent confidence interval:

-Inf 3.790116

sample estimates:

mean of x

3.477152

> t.test(s1,mu=3.6,alternative = 'less')

One Sample t-test

data: s1

t = -0.65923, df = 45, p-value = 0.2566

alternative hypothesis: true mean is less than 3.6

95 percent confidence interval:

-Inf 3.790116

sample estimates:

mean of x

3.477152

1. Sample T Test -> Right Tail Test

> t.test(s1,mu=3.3,alternative = 'greater')

One Sample t-test

data: s1

t = 0.95064, df = 45, p-value = 0.1734

alternative hypothesis: true mean is greater than 3.3

95 percent confidence interval:

3.164189 Inf

sample estimates:

mean of x

3.477152

1. Sample T Test -> Two Sided Test

> #Ho : mean(s1) = 2.8

> #Ha : mean(s1) != 2.8 --> two-sided test

> t.test(s1,mu=2.8,alternative = 'two.sided')

One Sample t-test

data: s1

t = 3.6337, df = 45, p-value = 0.0007138

alternative hypothesis: true mean is not equal to 2.8

95 percent confidence interval:

3.101821 3.852483

sample estimates:

mean of x

3.477152

1. Two Sample T Test

> s2 = erupt[130:160]

> length(s1)

[1] 46

> length(s2)

[1] 31

> mean(s1)

[1] 3.477152

> mean(s2)

[1] 3.556355

> #Ho : mean(s2) - mean(s1) >= 0.25

> #Ha : mean(s2) - mean(s1) < 0.25 --> left tail test

> t.test(s2,s1,mu=0.25,alternative = 'less')

Welch Two Sample t-test

data: s2 and s1

t = -0.60207, df = 67.067, p-value = 0.2746

alternative hypothesis: true difference in means is less than 0.25

95 percent confidence interval:

-Inf 0.5523578

sample estimates:

mean of x mean of y

3.556355 3.477152

> #Ho is not rejected, because p-value is greater than 0.05

> #so no action will be taken

> #the difference between both mean values and claim value, it is just a matter of chance variation.

1. Paired Test

> before = c(7,9,8,6,7,8)

> after = c(6,8,9,5,6,8)

> #since people started working from home, their time of delivery has increased by an average of 0.2 min

> #Ho : avg(after-before) >= 0.2

> #Ha : avg(after-before) < 0.2 --> left tail test

> t.test(after,before,mu=0.2,alternative = 'less',paired = T)

Paired t-test

data: after and before

t = -2.0494, df = 5, p-value = 0.04786

alternative hypothesis: true mean difference is less than 0.2

95 percent confidence interval:

-Inf 0.18827

sample estimates:

mean difference

-0.5

1. One Way Anova

> head(InsectSprays)

count spray

1 10 A

2 7 A

3 20 A

4 14 A

5 14 A

6 12 A

> str(InsectSprays)

'data.frame': 72 obs. of 2 variables:

$ count: num 10 7 20 14 14 12 10 23 17 20 ...

$ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...

> unique(InsectSprays$spray)

[1] A B C D E F

Levels: A B C D E F

> InsectSprays$spray = as.factor(InsectSprays$spray) #--> dummies(python)

> #Ho : counts are the same for all the pairs of sprays

> mod = aov(count ~ spray, data = InsectSprays)

> summary(mod)

Df Sum Sq Mean Sq F value Pr(>F)

spray 5 2669 533.8 34.7 <2e-16 \*\*\*

Residuals 66 1015 15.4

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> TukeyHSD(mod)

Tukey multiple comparisons of means

95% family-wise confidence level

Fit: aov(formula = count ~ spray, data = InsectSprays)

$spray

diff lwr upr p adj

B-A 0.8333333 -3.866075 5.532742 0.9951810

C-A -12.4166667 -17.116075 -7.717258 0.0000000

D-A -9.5833333 -14.282742 -4.883925 0.0000014

E-A -11.0000000 -15.699409 -6.300591 0.0000000

F-A 2.1666667 -2.532742 6.866075 0.7542147

C-B -13.2500000 -17.949409 -8.550591 0.0000000

D-B -10.4166667 -15.116075 -5.717258 0.0000002

E-B -11.8333333 -16.532742 -7.133925 0.0000000

F-B 1.3333333 -3.366075 6.032742 0.9603075

D-C 2.8333333 -1.866075 7.532742 0.4920707

E-C 1.4166667 -3.282742 6.116075 0.9488669

F-C 14.5833333 9.883925 19.282742 0.0000000

E-D -1.4166667 -6.116075 3.282742 0.9488669

F-D 11.7500000 7.050591 16.449409 0.0000000

F-E 13.1666667 8.467258 17.866075 0.0000000

1. Two Way Anova
2. 1 Sample variance Test

#1 sample variance test

> erupt = faithful$eruptions

> s1 = erupt[30:80]

> var(s1)

[1] 1.491471

> #Ho : var(s1) <= 1

> #Ha : var(s1) > 1

> varTest(s1,sigma.squared = 1,alternative = 'greater')

$statistic

Chi-Squared

74.57355

$parameters

df

50

$p.value

[1] 0.01370224

$estimate

variance

1.491471

$null.value

variance

1

$alternative

[1] "greater"

$method

[1] "Chi-Squared Test on Variance"

$data.name

[1] "s1"

$conf.int

LCL UCL

1.104715 Inf

attr(,"conf.level")

[1] 0.95

attr(,"class")

[1] "htestEnvStats"

1. 2 Sample Variance Test

> #2 sample variance test

> s2 = erupt[210:240]

> var(s1)

[1] 1.491471

> var(s2)

[1] 1.15639

>

> #Ho : var(s2)/var(s1) >= 1.1

> #Ha : var(s2)/var(s1) < 1.1

>

> var(s2)/var(s1)

[1] 0.7753354

> var.test(s2,s1,ratio = 1.1,alternative = 'less')

F test to compare two variances

data: s2 and s1

F = 0.70485, num df = 30, denom df = 50, p-value = 0.1541

alternative hypothesis: true ratio of variances is less than 1.1

95 percent confidence interval:

0.000000 1.365272

sample estimates:

ratio of variances

0.7753354

1. One Sample Proportion Test

> #1 sample proportion test

>

> #Ho : atleast 70% of the people in the batch are males

> #Ha : prop(males) < 0.7

>

> #from a sample of 84 people, 65 are males

> 65/80

[1] 0.8125

>

> #from a sample of 84 people, 50 are males

> 50/80

[1] 0.625

>

> prop.test(50,84,p=0.7,alternative = 'less')

1-sample proportions test with continuity correction

data: 50 out of 84, null probability 0.7

X-squared = 3.9053, df = 1, p-value = 0.02407

alternative hypothesis: true p is less than 0.7

95 percent confidence interval:

0.0000000 0.6845859

sample estimates:

p

0.5952381

> prop.test(50,84,p=0.65,alternative = 'less')

1-sample proportions test with continuity correction

data: 50 out of 84, null probability 0.65

X-squared = 0.87964, df = 1, p-value = 0.1741

alternative hypothesis: true p is less than 0.65

95 percent confidence interval:

0.0000000 0.6845859

sample estimates:

p

0.5952381

1. 2 Sample Proportion Test

> #2 proportion test

>

> #MH has atleast 12% more covid infected people as compared to GJ

> #from MH, in a sample of 150 people, 115 were infected

> #from GJ, in a sample of 100 people, 80 were infected

>

> 115/150 - 80/100

[1] -0.03333333

>

> #Ho : prop(MH) - prop(GJ) >= 0.12

> #Ha : prop(MH) - prop(GJ) < 0.12

>

> prop.test(c(115,80),c(150,100),alternative = 'less')

2-sample test for equality of proportions with continuity

correction

data: c(115, 80) out of c(150, 100)

X-squared = 0.21853, df = 1, p-value = 0.3201

alternative hypothesis: less

95 percent confidence interval:

-1.00000000 0.06192227

sample estimates:

prop 1 prop 2

0.7666667 0.8000000

1. Chi-square 🡪 Goodness of Fit

> #chi-square

> #1. goodness of fit

> #When we have to check any count is following the particular pattern or not

>

> pattern = c(0.6,0.2,0.15,0.05)

> sum(pattern)

[1] 1

> cts = c(120,50,40,30)

> chisq.test(cts,p=pattern)

Chi-squared test for given probabilities

data: cts

X-squared = 31.528, df = 3, p-value = 6.581e-07

1. Chi-square 🡪 Degree of Association

> #degree of association

> train = c(50,80,30)

> flight = c(20,50,25)

> bus = c(80,30,50)

> travel = rbind(train,flight,bus)

> colnames(travel) = c('summer','rain','winter')

> travel

summer rain winter

train 50 80 30

flight 20 50 25

bus 80 30 50

> chisq.test(travel)

Pearson's Chi-squared test

data: travel

X-squared = 47.132, df = 4, p-value = 1.431e-09

1. One Sample Poisson Rate

> #one sample poisson rate

> # - to compare the counts obtained from a sample against the claim value.

> # - claimed that fewer than 3 accidents take place on the local every day.

> # - Ho : n <= 3 and Ha : n > 3

> # - In a sample of 20 days, I saw 75 accidents were reported

> poisson.test(75,T = 20,r = 3,alternative = 'greater')

Exact Poisson test

data: 75 time base: 20

number of events = 75, time base = 20, p-value = 0.03407

alternative hypothesis: true event rate is greater than 3

95 percent confidence interval:

3.067294 Inf

sample estimates:

event rate

3.75

1. 2 Sample Poisson Rate

> #2 sample poisson rate

> #ratio of rate of occurances in 2 diffrent samples is compared against the claim value

> #in a sample of 30 days, I saw 100 accidents were reported in Banglore

>

> (100/30)/(75/20)

[1] 0.8888889

>

> #Ho : rate(BLR) / rate(MUM) = 1

> #Ha : rate(BLR) / rate(MUM) != 1

>

> poisson.test(c(100,75),T = c(30,20),r = 1,alternative = 'two.sided')

Comparison of Poisson rates

data: c(100, 75) time base: c(30, 20)

count1 = 100, expected count1 = 105, p-value = 0.4414

alternative hypothesis: true rate ratio is not equal to 1

95 percent confidence interval:

0.6523343 1.2156621

sample estimates:

rate ratio

0.8888889

1. Regression Models
2. Simple Regression Model
3. Multiple Regression Model

> #2. Multiple Linear Regression

>

> head(stackloss)

Air.Flow Water.Temp Acid.Conc. stack.loss

1 80 27 89 42

2 80 27 88 37

3 75 25 90 37

4 62 24 87 28

5 62 22 87 18

6 62 23 87 18

> str(stackloss)

'data.frame': 21 obs. of 4 variables:

$ Air.Flow : num 80 80 75 62 62 62 62 62 58 58 ...

$ Water.Temp: num 27 27 25 24 22 23 24 24 23 18 ...

$ Acid.Conc.: num 89 88 90 87 87 87 93 93 87 80 ...

$ stack.loss: num 42 37 37 28 18 18 19 20 15 14 ...

> mod1 = lm(stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,data = stackloss)

> summary(mod1)

Call:

lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,

data = stackloss)

Residuals:

Min 1Q Median 3Q Max

-7.2377 -1.7117 -0.4551 2.3614 5.6978

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -39.9197 11.8960 -3.356 0.00375 \*\*

Air.Flow 0.7156 0.1349 5.307 5.8e-05 \*\*\*

Water.Temp 1.2953 0.3680 3.520 0.00263 \*\*

Acid.Conc. -0.1521 0.1563 -0.973 0.34405

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.243 on 17 degrees of freedom

Multiple R-squared: 0.9136, Adjusted R-squared: 0.8983

F-statistic: 59.9 on 3 and 17 DF, p-value: 3.016e-09

> mod2 = lm(stack.loss ~ .,data = stackloss)

> summary(mod2)

Call:

lm(formula = stack.loss ~ ., data = stackloss)

Residuals:

Min 1Q Median 3Q Max

-7.2377 -1.7117 -0.4551 2.3614 5.6978

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -39.9197 11.8960 -3.356 0.00375 \*\*

Air.Flow 0.7156 0.1349 5.307 5.8e-05 \*\*\*

Water.Temp 1.2953 0.3680 3.520 0.00263 \*\*

Acid.Conc. -0.1521 0.1563 -0.973 0.34405

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.243 on 17 degrees of freedom

Multiple R-squared: 0.9136, Adjusted R-squared: 0.8983

F-statistic: 59.9 on 3 and 17 DF, p-value: 3.016e-09

1. Categorical Regression Model
2. Installing packages from ‘car’

> install.packages('car')

> library(car)

> head(Salaries)

rank discipline yrs.since.phd yrs.service sex salary

1 Prof B 19 18 Male 139750

2 Prof B 20 16 Male 173200

3 AsstProf B 4 3 Male 79750

4 Prof B 45 39 Male 115000

5 Prof B 40 41 Male 141500

6 AssocProf B 6 6 Male 97000

> levels(df$rank)

[1] "AsstProf" "AssocProf" "Prof"

>

> #to change the reference

> df$rank = relevel(df$rank, ref = 'AsstProf')

>

> mod4 = lm(salary ~ .,data = df)

> summary(mod4)

Call:

lm(formula = salary ~ ., data = df)

Residuals:

Min 1Q Median 3Q Max

-64515 -16180 -1234 12181 107174

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 81151.3 2896.9 28.013 < 2e-16 \*\*\*

rankAssocProf 14615.4 4270.6 3.422 0.000686 \*\*\*

rankProf 49228.8 3991.9 12.332 < 2e-16 \*\*\*

yrs.service -158.1 115.0 -1.376 0.169708

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 23610 on 393 degrees of freedom

Multiple R-squared: 0.3972, Adjusted R-squared: 0.3926

F-statistic: 86.3 on 3 and 393 DF, p-value: < 2.2e-16

# for every year of service wil decrease in the salary by 150 rs.

# change in the level of rank from assistant proff. to associate proff. would increase your salary by 14615.

# change in the level of rank form assistant proff. to professor it would increase your salary by 49228.

1. Logistic Regressions
2. Binary Logistic Regressions
3. Nominal Logistic Regressions

> #Nominal / Multi-Nominal Regression

>

> df = read\_excel('CDAC\_DataBook.xlsx',sheet = 'nominal')

> head(df)

# A tibble: 6 × 4

ses write math prog

*<dbl>* *<dbl>* *<dbl>* *<dbl>*

1 1 35 41 1

2 2 33 41 2

3 3 39 44 3

4 1 37 42 1

5 2 31 40 2

6 3 36 42 2

> df = df[,-2]

> head(df)

# A tibble: 6 × 3

ses math prog

*<dbl>* *<dbl>* *<dbl>*

1 1 41 1

2 2 41 2

3 3 44 3

4 1 42 1

5 2 40 2

6 3 42 2

> str(df)

tibble [200 × 3] (S3: tbl\_df/tbl/data.frame)

$ ses : num [1:200] 1 2 3 1 2 3 2 2 2 2 ...

$ math: num [1:200] 41 41 44 42 40 42 46 40 33 46 ...

$ prog: num [1:200] 1 2 3 1 2 2 2 2 2 2 ...

> df$ses = as.factor(df$ses)

> df$prog = as.factor(df$prog)

> str(df)

tibble [200 × 3] (S3: tbl\_df/tbl/data.frame)

$ ses : Factor w/ 3 levels "1","2","3": 1 2 3 1 2 3 2 2 2 2 ...

$ math: num [1:200] 41 41 44 42 40 42 46 40 33 46 ...

$ prog: Factor w/ 3 levels "1","2","3": 1 2 3 1 2 2 2 2 2 2 ...

> install.packages('nnet')

> library(nnet)

> mod = multinom(prog ~ ., data = df)

# weights: 15 (8 variable)

initial value 219.722458

iter 10 value 136.243354

final value 136.060483

converged

> summary(mod)

Call:

multinom(formula = prog ~ ., data = df)

Coefficients:

(Intercept) ses2 ses3 math

2 -0.7311753 2.734445 2.790327 0.002800878

3 -8.9284639 4.653155 6.592924 0.116028359

Std. Errors:

(Intercept) ses2 ses3 math

2 1.456837 0.5322456 1.099542 0.02919595

3 2.082252 0.9077981 1.292542 0.03422413

Residual Deviance: 272.121

AIC: 288.121

> z = summary(mod)$coefficients/summary(mod)$standard.errors

> p = (1 - pnorm(abs(z)))\*2

> p

(Intercept) ses2 ses3 math

2 6.157432e-01 2.783241e-07 1.115778e-02 0.9235731474

3 1.803799e-05 2.963410e-07 3.383215e-07 0.0006982878

1. Ordinal Logistic Regressions