week-4

4. To import data in R and performing time series data and Google trends in R

Import data in R Reading datasets using R:

1. Reading data from txt or csv files using R base function

read.csv()   read.txt() read.csv2() read.table()

Syntax:

```
read.csv(filename,header=TRUE,sep=",",dec="."..)
Ex: data<-read.csv("sample1.csv")
    print(data)
```

Output:

```
      student.percentages
1 rollno  name  percentage
2  501      xx     50
3  502      yy     70
```

--reading data from internet

Syntax: read.delim("URL")

Ex: data<-read.delim("URL")
     print(data)

2. Reading data from excel files into R:

  read_excel(), read.xlsx()

Syntax:

read.xlsx(filename,sheetIndex,header=TRUE)

Ex:

Note: install the package installer.packages(“xlsx”) if required.

```
library(“xlsx”)
data<-read.xlsx("one.xlsx", sheetIndex=1)
print(data)
```

output:

```
  rollno name percentage
1     1  ss        30
2     2  ff        40
```

EX:

data<-read_excel("filename.xlsx")

Time series:

Is a series of data points in which each data point is associated with a time stamp.

time-series object: the data for the time series is stored in this object it is like a vector or data frame.

Syntax:

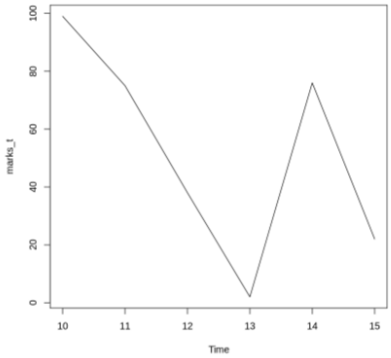
ts_object_name=ts(data,start,end,frequency)

Ex:

```
marks=c(99,75,38,2,76,22)
marks_t=ts(marks,start=c(10,1),frequency=1)
print(marks_t)
plot(marks_t)
```

output:

```
Time Series:
Start = 10
End = 15
Frequency = 1
[1] 99 75 38  2 76 22
```



Google Trends in R:

syntax:

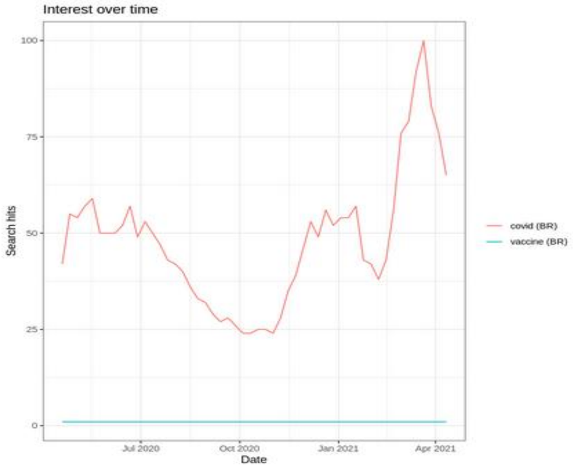
```
gtrends(keyword.geo=" ", time=" " ..)
```

Ex:

Note: install gtrendsR in workspace if required by using installer.packages(“gtrendsR”).

```
library (gtrendsR)
res=gtrends(keyword=c("covid", "vaccine"), time="today 12-m", geo="BR")
plot(res)
```

output:



WEEK-5

5. A Data Science project –a first look

DATATYPES: -

#Integer Datatype

```
variable_integer <- 122L

cat(variable_integer,“\n”)
```

Output:

```
[1] 122
```

```
cat(“the datatype of variable_integer is”, class(variable_integer),“\n\n”)
```

Output:

```
[1] the datatype of variable_integer is integer
```

#Numeric Datatype

```
variable_numeric<-5432
```

```
cat(variable_numeric,“\n”)
```

Output:

```
[1] 5432
```

```
cat(“the datatype of variable_numeric is”, class(variable_numeric),“\n\n”)
```

Output:

```
[1] the datatype of variable_numeric is numeric
```

#Complex Datatype

```
variable_complex <- 7+3i
```

```
cat(variable_complex,“\n”)
```

Output:

```
[1] 7+3i
```

```
cat(“the datatype of variable_complex is”, class(variable_complex), “\n\n”)
```

Output:

```
[1] the datatype of variable_complex is complex
```

#Character Datatype

```
variable_char <- “Learning R programming”
```

```
cat(variable_char, “\n”)
```

Output:

```
[1] Learning R programming
```

```
cat(“the datatype of variable_char is”, class(variable_char), “\n\n”)
```

Output:

```
[1] the datatype of variable_char is character
```

#Raw datatype

```
variable_raw<- charToRaw (“Learning R programming”)

cat (variable_raw, “\n”)
```

Output:

```
[1] 4c 65 61 72 6e 69 6e 67 20 72 20 70 72 6f 67 72 61 6d 6d 69 6e 67
```

```
cat(“the datatype of variable_raw is”, class(variable_raw), “\n\n”)
```

Output:

```
[1] the datatype of variable_raw is raw
```

WEEK-6

6. Data Manipulation with tidyr

```
library(tidyr)
n=2
t_d=data.frame(s.no=c(1:n),group.1=c(10,20),group.2=c(30,40),group.3=c(50,60))
print(t_d)
long=t_d%>% gather(group,frequency,group.1:group.3)
sp=long%>% separate(group,c("allotment","number"))
print(sp)
```

Output:

```
  s.no group.1 group.2 group.3
1    1      10      30      50
2    2      20      40      60
  s.no allotment number frequency
1    1     group      1        10
2    2     group      1        20
```

```
3   1    group     2       30
4   2    group     2       40
5   1    group     3       50
6   2    group     3       60
```

Description:

tidyr is a package it makes easy to tidy our data. Data is said to be tidy when each column represents a variable, and each row represents an observation.

It simplify the process of creating tidy data(data in order).

tidyr package provides various important functions that can be used for data cleaning.

Functions:

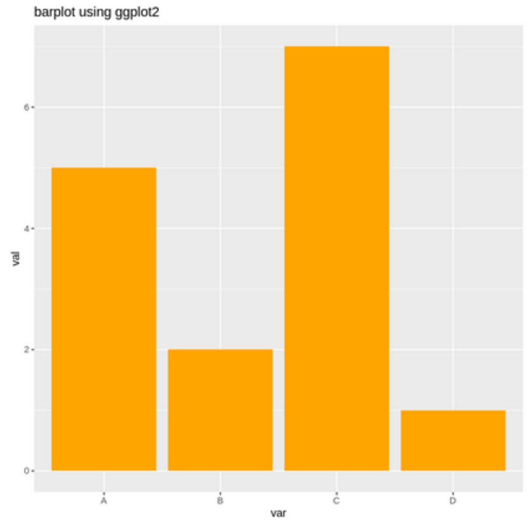
1. **gather():** It takes multiple columns and gather them into key value pairs.
2. **separate():** It converts longer data into a wide format.
3. **unite():** it merges two columns into one column.
4. **split()** – splits one column into two or more columns.

WEEK-7

Aim: plotting with ggplot2.

```
library(ggplot2)
dataframe<-data.frame(var=c("A","B","C","D"), val=c(5,2,7,1))
ggplot(data=dataframe, aes(x=var, y=val))+geom_bar(stat="identity", fill="orange")+ggtitle(
"barplot using ggplot2")
```

Output:



Description:

ggplot2 is based on the grammar of graphics. It is a package, offers a powerful graphics language for creating elegant and complex plots. ggplot2 allows you to create graphs that represent both univariate and multivariate numerical and categorical data in a straightforward manner.

Producing a plot with ggplot2, we must give three things:

1. A data frame containing our data.
2. How the columns of the data frame can be translated into positions, colors, sizes, and shapes of graphical elements (“aesthetics”).
3. The actual graphical elements to display (“geometric objects”).

   To display data values, map variables in the dataset to aesthetic(aes), properties of the geom like size, color and x and y location.

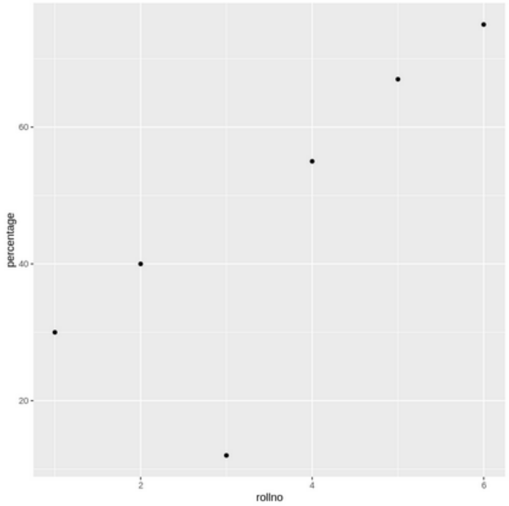
Using ggplot2 with data frame:

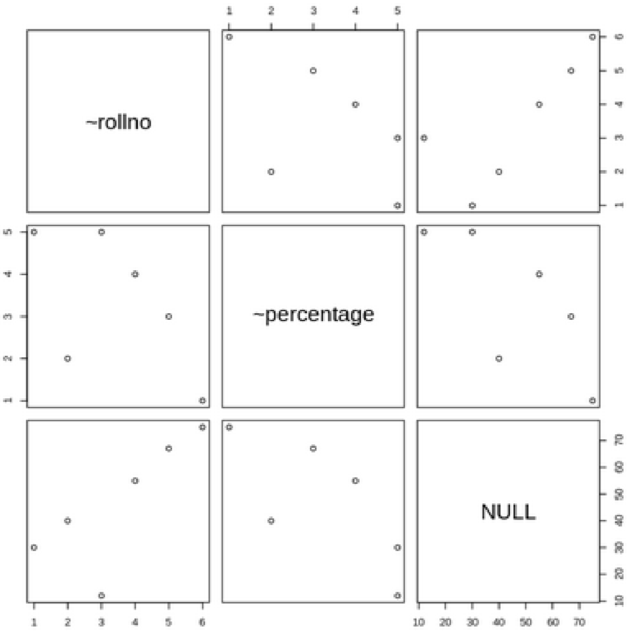
```
gap <- read.csv("ex1.csv")
head(gap)
ggplot(gap,aes(x=rollno, y=percentage))+geom_point()
plot(gap,aes(x=rollno, y=percentage))+geom_point()
gap_t=ts(gap,start=c(2,1), frequency=1)
plot(gap_t)
```

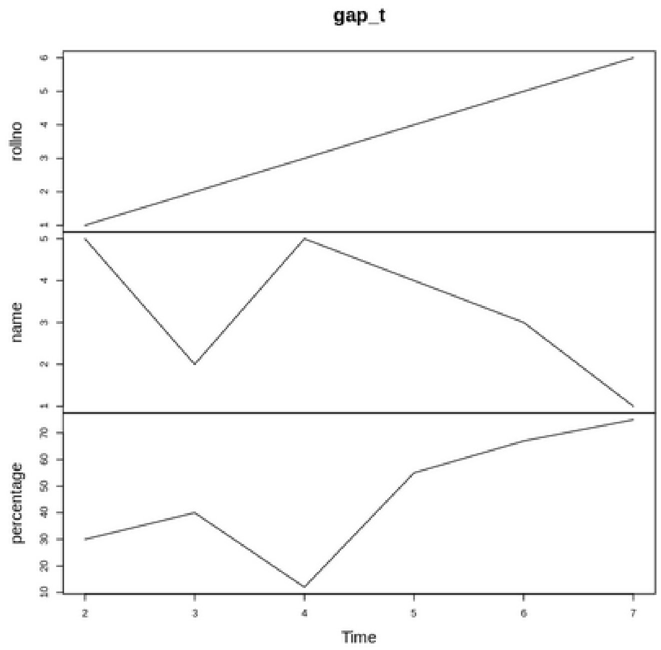
output:

A data.frame: 6 × 3

	rollno	name	percentage
	`<int>`	`<chr>`	`<int>`
1	1	ss	30
2	2	ff	40
3	3	ss	12
4	4	rr	55
5	5	hh	67
6	6	bb	75




~rollno
~percentage
NULL
1 2 3 4 5
1 2 3 4 5 6
10 20 30 40 50 60 70


gap_t
rollno
name
percentage
Time
2 3 4 5 6 7
1 2 3 4 5 6
10 20 30 40 50 60 70