INDEX

EX NO: 01 DATE:	**Download WhatsApp data – integrate with geo spatial data and provide timeline trained analysis**

AIM:

To download WhatsApp data – integrate with geo spatial data and provide timeline trained analysis

ABOUT DATASET :

This dataset is taken from Kaggle .This dataset show about the rise in the demand of online delivery in the metropolitan cities such as Bangalore in India.

ALGORITHM:

Step 1: Start the program by including all the required packages and call them using library().

Step 2: Attach the dataset by using attach() function

Step 3: Subset the dataset by selecting the necessary columns in such a way that it contains latitude and longitude coordinates

Step 4: Define a function that assigns color to each coordinates using a factor attribute

Step 5: Assign leaflet() function to R object and respective parameters such as dataframe name, latitude, longitude, icons, etc. ,

Step 6: Run the code and View the Geo spatial map created.

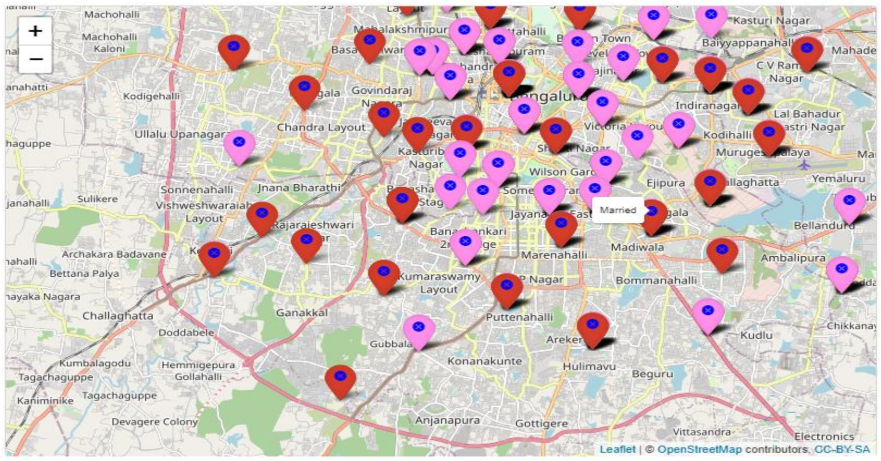
Step 7: Stop the Program

CODE:

```
# overview of dataset

spatialData=read.csv(file.choose(), header = T)

View(spatialData)

attach(spatialData)

summary(spatialData)


#selecting particular features

spatial_df = spatialData[c(1,3,4,5,8,9)]

spatial_df

unique(Pin.code)


#adding new column (vectorization)


label_id = as.numeric(as.factor(Gender))

unique(spatial_df[c(2,3)])

unique(label_id)

names(spatial_df)


library(leaflet)


#applying colors

getcolor = function(spatial_df)

{

  sapply(label_id,function(label_id) #call by reference [reference = label_id]

  {

    if(label_id == 1)

    {

      "pink"

    }

    else if(label_id == 2)
    {
      "red"
    }
    else if(label_id == 3)
    {
      "green"
    }
    else
    {
      "blue"
    }
  }
  )
}

icons = awesomeIcons(icon="ios-close",iconColor = "blue", library = "ion",markerColor
=getcolor(spatial_df))
icons

#calling leaflet functing to a r object

admin_name_map <- leaflet() %>%
  addAwesomeMarkers(data = spatial_df, lat = latitude,  lng = longitude,icon = icons,
                  popup = Gender, label = Marital.Status)%>%

  #this is the intergeration part with Open Street Map using pipeline
  addTiles(group ="OSM")
admin_name_map
```

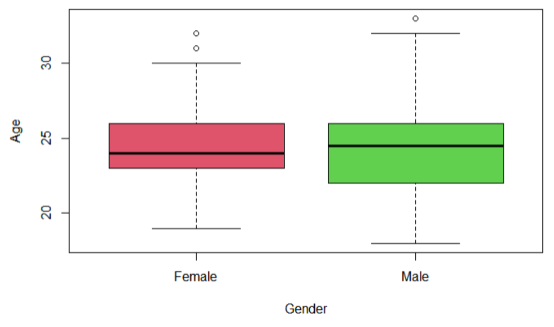
```
# EDA
mean(spatialData$Age)
mean(spatialData$Family.size)
boxplot(Age~Gender)
boxplot(Age~Occupation)
boxplot(Age~Marital.Status)
hist(Pin.code)
```

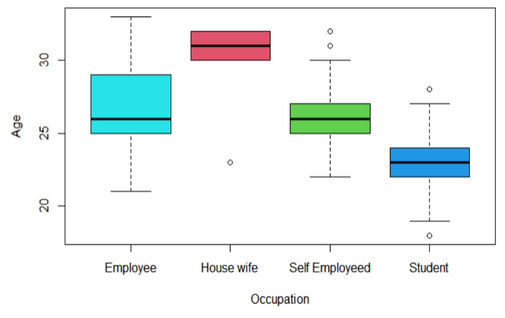
OUTPUT

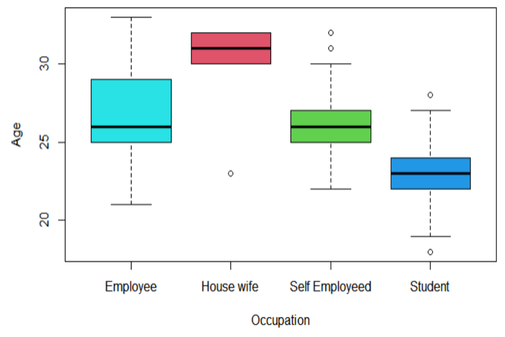


```
[1] 24.62887
```

```
[1] 24.62887
```







INSIGHTS

- From this data set we understand that In Bangalore most of the people who order food through online are married and the average age limit of students who order food through online are 24 and the average family size of people who order food is 3 and the gender who order more food through online is Male. Mostly the people who order food are educated people.

- Here people who order food through online who's age limit between 25 to 30 are employees, and Above 30 are house wife and few self employed people and age limit between 20 to 25 are students. From this we come to know people who mostly order foods are Employed and they are male. Here most of the people prefer not to say their marital status so we can assume from the data set

- The most people order food are married employed and they are male.

EX NO: 02 DATE:	**Visualize any education dataset and scrap with frequent terms and do all the pre-processing techniques of Text Analytics**

AIM:

To visualize any education dataset and scrap with frequent terms and do all the pre-processing techniques of text Analytics

ABOUT DATASET:

This education data set is taken from Kaggle . This dataset contains 3.682 records of courses from 4 subjects (Business Finance, Graphic Design, Musical Instruments and Web Design) taken from Udemy.

ALGORITHM:

Step 1: Start the program by including all the required packages and call them using library().

Step 2: Subset the data frame and select a particular column which we have to tokenize

Step 3: Apply unnest_tokens() from tidy text package and tokenize particular column value

Step 4: Apply all the pre-processing techniques like stop word removal, gram separation, frequency count, word separation, etc

Step 5: Find the word associations using igraph() function

Step 6: Visualize the plot of token association using ggraph() function

Step 7: Visualize word cloud of tokens and its count

Step 8: Run the code and View the analysis and visualizations created.

Step 9: Stop the Program

CODING

~~~
---
title: "Text Preprocessing"
author: "AKSHAYA.N"
date: '2022-05-30'
---
```{r setup, include=FALSE}
education= read.csv(file.choose(),header = TRUE)
View(education)
attach(education)
attach(education)
library(tidytext)
library(tidyr)
library(dplyr)
library(stopwords)
library(igraph)
library(ggraph)
library(wordcloud2)
~~~

#The head() function in R is used to display the first n rows present in the input data frame. Here I
have choosen 50 rows so 50 rows have been displayed

```{r}
head(education,50)
str(education)
summary(education)
```
```{r}
str(education)
```
```
```

Text Preprocessing

# Tokenization

```{r}
data_tokens = unnest_tokens(education, word,subject)
head(data_tokens,20)
```

# Stop Words

Stopwords are certain words that are present in high proportion but rarely contribute anything of sense towards serving analytical purpose. These include words such as "a","the","if" etc words are removed.

```{r}
token_stop=education_tokens %>% filter(!word %in% stop_words$word)
head(token_stop,100)
```

# N-grams

I have used unnest_tokens function to tokenize by word which is useful for the frequency analyses, the ngrams function to use tokenize into consecutive sequences of words like bigram,trigram etc., I have use bigram function and separated sequence of words.

```{r}
ng = education %>% unnest_tokens(word,course_title, token = "ngrams", n=2)%>%
  separate(word, c("word1","word2"),sep = " ") %>% filter(!word1 %in% stop_words$word)%>%
  filter(!word2 %in% stop_words$word)%>% unite(word, word1,word2,sep=" ")%>%
  count(word, sort =TRUE)
head(ng,20)
```

#Seperate

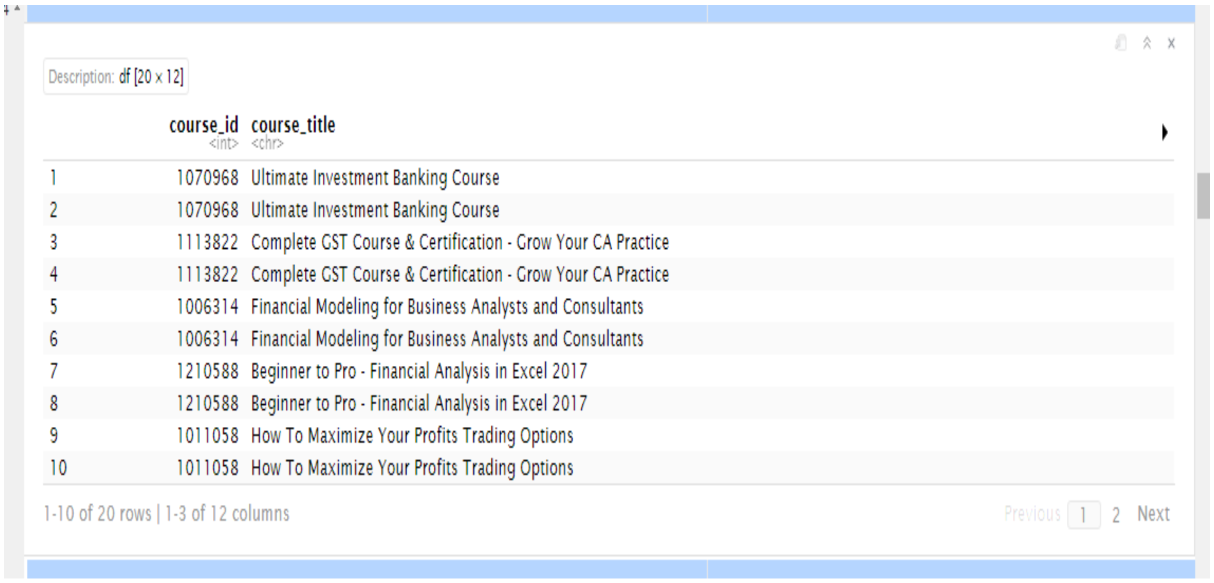
After ngrams,separate bigram words like word1 and word2 which is helpful to use the igraph represention

~~~
```{r}
s= ng %>%separate(word,c("word1","word2"),sep = " ")
x=head(s %>% filter(n>=2))
x
```
#Igraph
```{r}
i=x %>% count(word1,word2,directed=TRUE) %>% graph_from_data_frame()
i
```
#GG graph
   GGgraph to find the link words of root word in graphical represention, it is more useful to find
the root word and related words.
```{r}
set.seed(20181005)
a=arrow(angle=20,length=unit(0.1,"inches"),ends="last",type="open")


ggraph(i,layout="fr")+geom_edge_link(aes(color=n,width=n),arrow=a)+
  geom_node_point()+ geom_node_text(aes(label=name),vjust=1,hjust=1)
```
#Word frequency
    I have to find the frequency words in this dataset, the count operation is heplful to find the
which word is more frequnecy in this.
```{r}
w = token_stop %>% count(word,sort = TRUE)
head(w)
```


#word_cloud
      Word Cloud provides an excellent option to analyze the text data through visualization in the
form of tags, or words, where the importance of a word is explained by its frequency.
~~~

```{r}

wordcloud2(data = w, size=0.4, color = "random-light",minRotation = -
pi/6,backgroundColor=rainbow(2), maxRotation = -pi/6, rotateRatio = 1,shape="circle")

wordcloud2(data=w,size=1.6,color='random-dark')

```

OUTPUT

Tokenization:



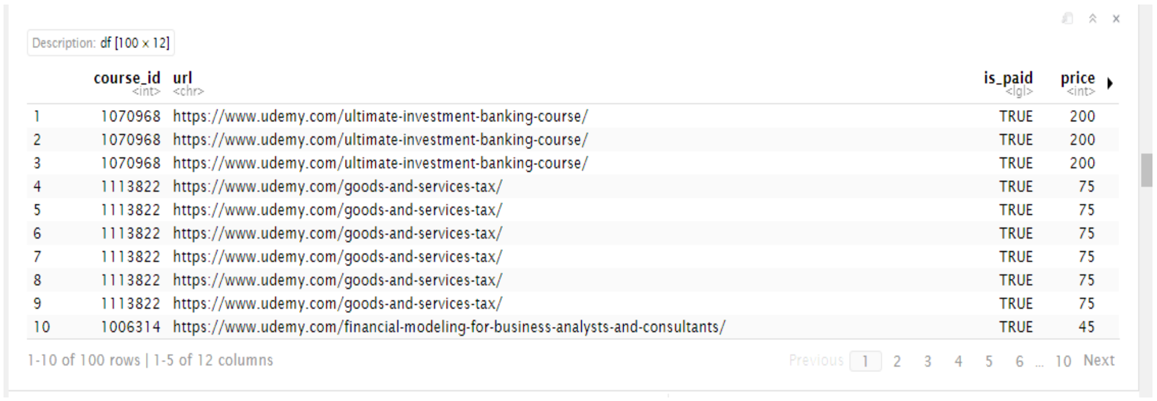
Description: df [20 x 12]

	course_id <int>	course_title <chr>
1	1070968	Ultimate Investment Banking Course
2	1070968	Ultimate Investment Banking Course
3	1113822	Complete GST Course & Certification - Grow Your CA Practice
4	1113822	Complete GST Course & Certification - Grow Your CA Practice
5	1006314	Financial Modeling for Business Analysts and Consultants
6	1006314	Financial Modeling for Business Analysts and Consultants
7	1210588	Beginner to Pro - Financial Analysis in Excel 2017
8	1210588	Beginner to Pro - Financial Analysis in Excel 2017
9	1011058	How To Maximize Your Profits Trading Options
10	1011058	How To Maximize Your Profits Trading Options

1-10 of 20 rows | 1-3 of 12 columns

Previous 1 2 Next

Stop Words:

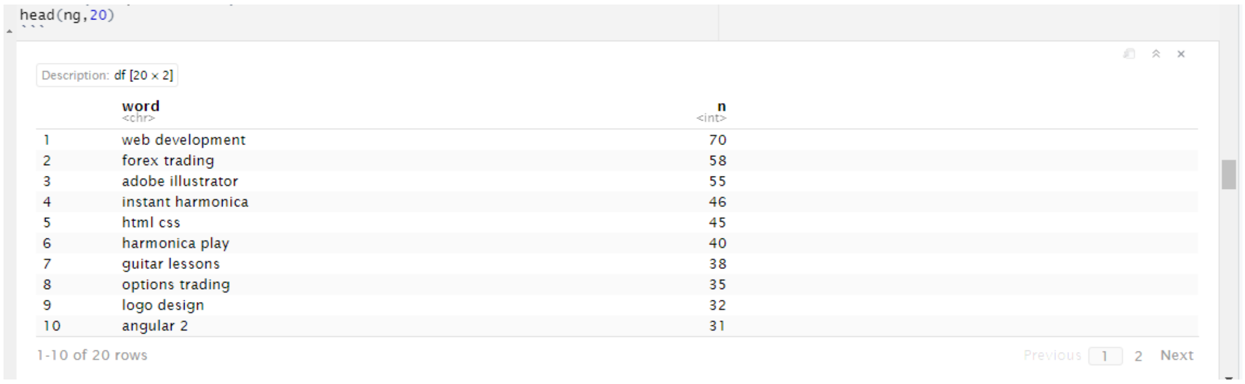


Description: df [100 × 12]

	course_id <int>	url <chr>	is_paid <lgl>	price <int>
1	1070968	https://www.udemy.com/ultimate-investment-banking-course/	TRUE	200
2	1070968	https://www.udemy.com/ultimate-investment-banking-course/	TRUE	200
3	1070968	https://www.udemy.com/ultimate-investment-banking-course/	TRUE	200
4	1113822	https://www.udemy.com/goods-and-services-tax/	TRUE	75
5	1113822	https://www.udemy.com/goods-and-services-tax/	TRUE	75
6	1113822	https://www.udemy.com/goods-and-services-tax/	TRUE	75
7	1113822	https://www.udemy.com/goods-and-services-tax/	TRUE	75
8	1113822	https://www.udemy.com/goods-and-services-tax/	TRUE	75
9	1113822	https://www.udemy.com/goods-and-services-tax/	TRUE	75
10	1006314	https://www.udemy.com/financial-modeling-for-business-analysts-and-consultants/	TRUE	45

1-10 of 100 rows | 1-5 of 12 columns   Previous 1 2 3 4 5 6 ... 10 Next

N-grams:



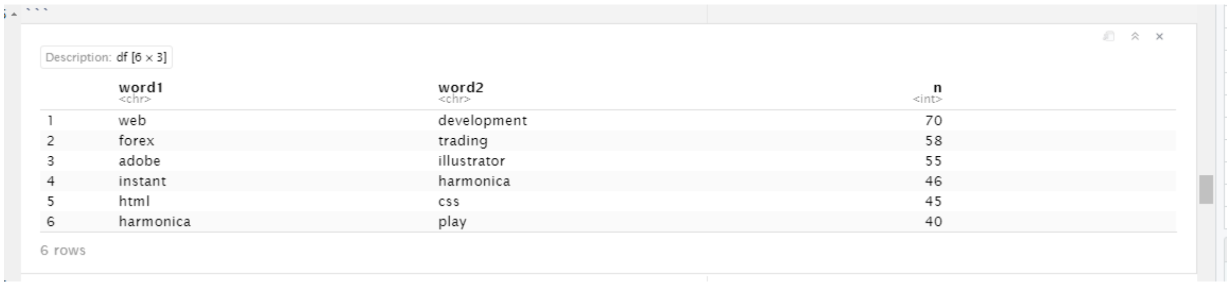
```
head(ng,20)
```

Description: df [20 × 2]

	word <chr>	n <int>
1	web development	70
2	forex trading	58
3	adobe illustrator	55
4	instant harmonica	46
5	html css	45
6	harmonica play	40
7	guitar lessons	38
8	options trading	35
9	logo design	32
10	angular 2	31

1-10 of 20 rows   Previous 1 2 Next

Separate:

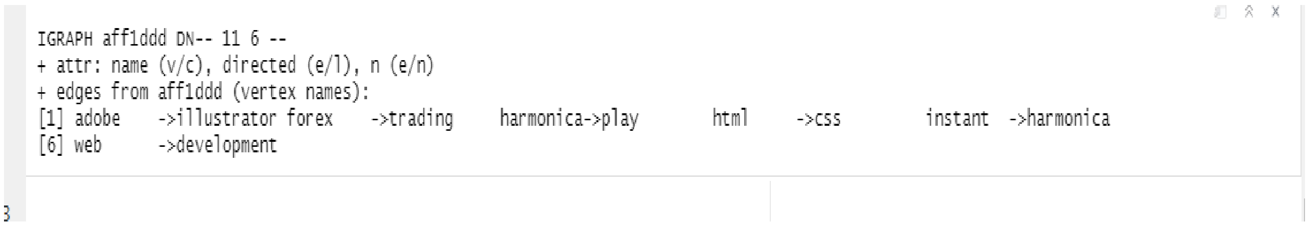


Description: df [6 × 3]

	word1 <chr>	word2 <chr>	n <int>
1	web	development	70
2	forex	trading	58
3	adobe	illustrator	55
4	instant	harmonica	46
5	html	css	45
6	harmonica	play	40

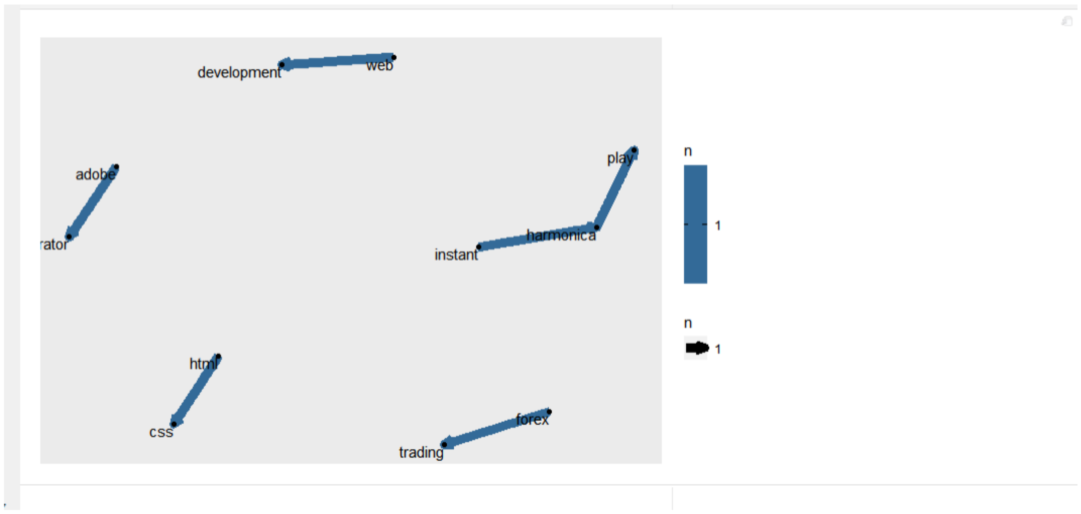
6 rows

Igraph:

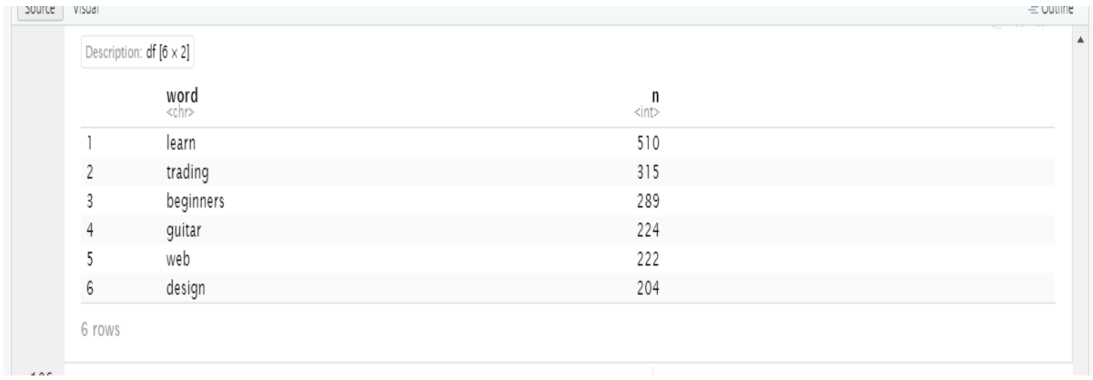


```
IGRAPH aff1ddd DN-- 11 6 --
+ attr: name (v/c), directed (e/l), n (e/n)
+ edges from aff1ddd (vertex names):
[1] adobe    ->illustrator forex    ->trading    harmonica->play       html     ->css         instant  ->harmonica
[6] web      ->development
```

GG graph:



Word frequency:

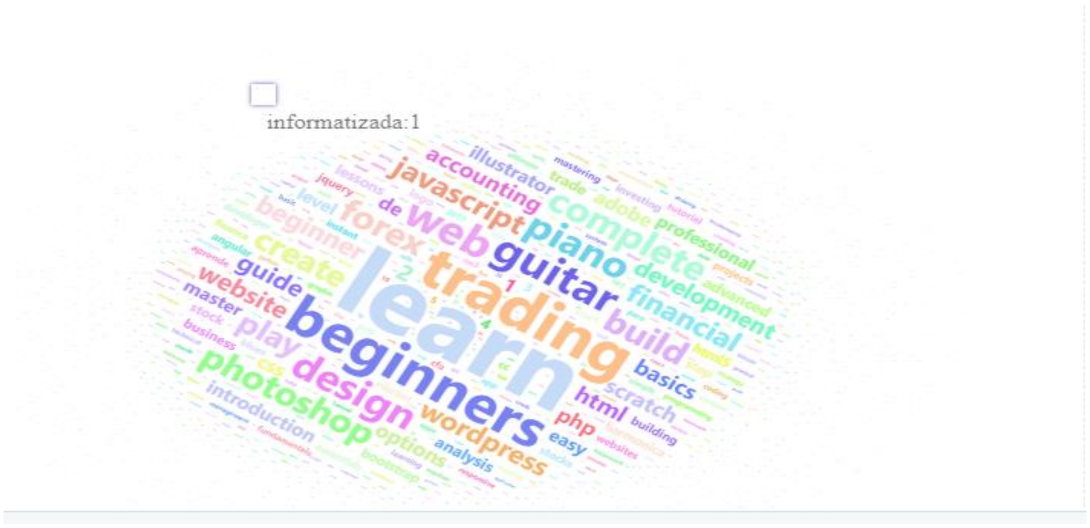


Description: df [6 x 2]

	word <chr>	n <int>
1	learn	510
2	trading	315
3	beginners	289
4	guitar	224
5	web	222
6	design	204

6 rows

word cloud :



INSIGHTS:

This dataset contains 3.682 records of courses from 4 subjects (Business Finance, Graphic Design, Musical Instruments and Web Design) taken from Udemy. Here we come to know about the price of the course number of lectures in the course .We used this data to do Data pre-processing . Tokenization gives text analytic software the ability to provide more structure for analysis. Stop words are certain words that are present in high proportion but rarely contribute anything of sense towards serving analytical purpose. These include words such as "a","the","if" etc words are removed. I have used unnest_tokens function to tokenize by word which is useful for the frequency analyses, the ngrams function to use tokenize into consecutive sequences of words like bigram,trigram etc., I have use bigram function and separated sequence of words. I have to find the frequency words in this dataset, the count operation is heplful to find the which word is more frequnecy in this. Word Cloud provides an excellent option to analyze the text data through visualization in the form of tags, or words, where the importance of a word is explained by its frequency.

EX NO: 3 DATE:	**Download a dataset from Kaggle and create a shiny dashboard**

AIM:

To download a dataset from Kaggle and create a shiny dashboard.

ALGORITHM:

Step 1: Start the program by including all the required packages and call them using library().

Step 2: Add necessary attributes in shiny UI section like title panel, column selection, buttons, etc.,

Step 3: Manipulate the dataset using the filter function of dplyr

Step 4: Define server logic in the server tab and include all the plots

Step 5: Visualize all the plots dynamically

Step 6: Run the code and view the visualizations created.

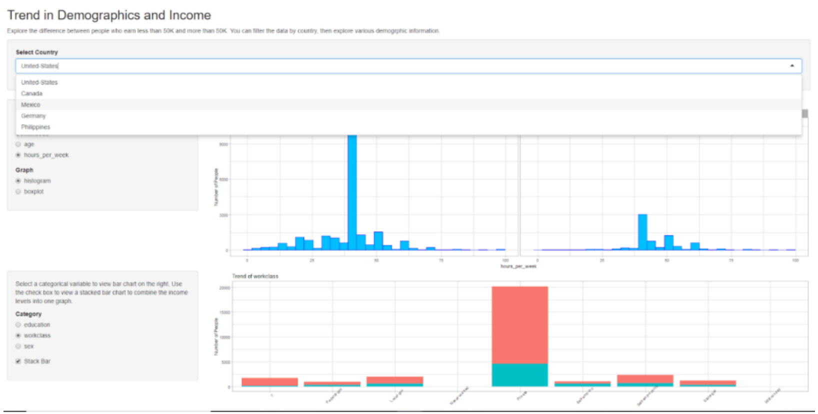
Step 7: Stop the Program

CODING

```
# This is a Shiny web application. You can run the application by clicking
# the 'Run App' button above.
# Find out more about building applications with Shiny here:
#    http://shiny.rstudio.com/
library(shiny)
library(tidyverse)
library(rio)
library(ggthemes)
```

```
adult_df=read.csv(file.choose(),header=TRUE)

head(adult_df)

# Define UI for application that draws a histogram

# Application UI Layout

ui=shinyUI(fluidPage(

   br(),

   # TASK 1: Application title

   titlePanel("Trend in Demographics and Income"),

   p("Explore the difference between people who earn less than 50K and more than 50K. You can
filter the data by country, then explore various demogrphic information."),

# TASK 2: Add first fluidRow to select input for country

   fluidRow(

      column(12,

            wellPanel(selectInput("country","Select Country",choices=c("United-
States","Canada","Mexico","Germany","Philippines")))

      )  # add select input

   ),

   # TASK 3: Add second fluidRow to control how to plot the continuous variables

   fluidRow(

      column(3,

            wellPanel(

               p("Select a continuous variable and graph type (histogram or boxplot) to view on the
right."),

               radioButtons("continuous_variable","Continuous",choices=c("age","hours_per_week")),
# add radio buttons for continuous variables

               radioButtons("graph_type","Graph",choices=c("histogram","boxplot"))    # add radio
buttons for chart type

            )

      ),column(9, plotOutput("p1"))  # add plot output

   ),


   # TASK 4: Add third fluidRow to control how to plot the categorical variables

   fluidRow(
        column(3,
            wellPanel(
              p("Select a categorical variable to view bar chart on the right. Use the check box to view
a stacked bar chart to combine the income levels into one graph. "),

radioButtons("categorical_variable","Category",choices=c("education","workclass","sex")),    # add
radio buttons for categorical variables
              checkboxInput("is_stacked","Stack Bar",value=FALSE)     # add check box input for
stacked bar chart option
            )
        ),
        column(9, plotOutput("p2"))  # add plot output
    )
)
)
# Define server logic required to draw a histogram
server=shinyServer(function(input, output) {
    adult=import("D:/College Materials/Sem 2/R programming/datasets/adult.csv")              # Read in
data
    names(adult)=tolower(names(adult))      # Convert column names to lowercase for convenience


    df_country <- reactive({
        adult %>% filter(native_country == input$country)
    })


    # TASK 5: Create logic to plot histogram or boxplot
    output$p1 <- renderPlot({
        if (input$graph_type == "histogram") {
            # Histogram
            ggplot(df_country(), aes_string(x =input$continuous_variable)) +
                geom_histogram(color="blue",fill="deepskyblue") +  # histogram geom
                labs(y="Number of People", title=paste("Trend of ",input$continuous_variable)) +  # labels
                facet_wrap(~prediction)+ # facet by prediction
            theme_light()
        }
        else {
          # Boxplot
          ggplot(df_country(), aes_string(y = input$continuous_variable)) +
            geom_boxplot(color="chocolate",fill="darkgoldenrod1") +  # boxplot geom
            coord_flip() +  # flip coordinates
            labs(x="Number of People", title=paste("Boxplot of",input$continuous_variable)) +  #
labels
            facet_wrap(~prediction)+    # facet by prediction
            theme_light()
        }

    })

    # TASK 6: Create logic to plot faceted bar chart or stacked bar chart
    output$p2 <- renderPlot({
        # Bar chart
        p <- ggplot(df_country(), aes_string(x =input$categorical_variable)) +
          labs(y="Number of People",title=(paste("Trend of",input$categorical_variable))) +  # labels
          theme_light()+
          theme(axis.text.x=element_text(angle=45),legend.position="bottom")    # modify theme to
change text angle and legend position

        if (input$is_stacked) {
          p + geom_bar(aes(fill=prediction))  # add bar geom and use prediction as fill
        }
        else{
          p +
            geom_bar(aes_string(fill=input$categorical_variable)) + # add bar geom and use
input$categorical_variables as fill
            facet_wrap(~prediction)   # facet by prediction
        }
      })
    })

    # Run the application
    shinyApp(ui = ui, server = server)
```

OUTPUT:



Trend in Demographics and Income

Explore the difference between people who earn less than 50K and more than 50K. You can filter the data by country, then explore various demogrphic information.

Select Country

United-States

United-States
Canada
Mexico
Germany
Philippines

- ○ age
- ● hours_per_week

Graph

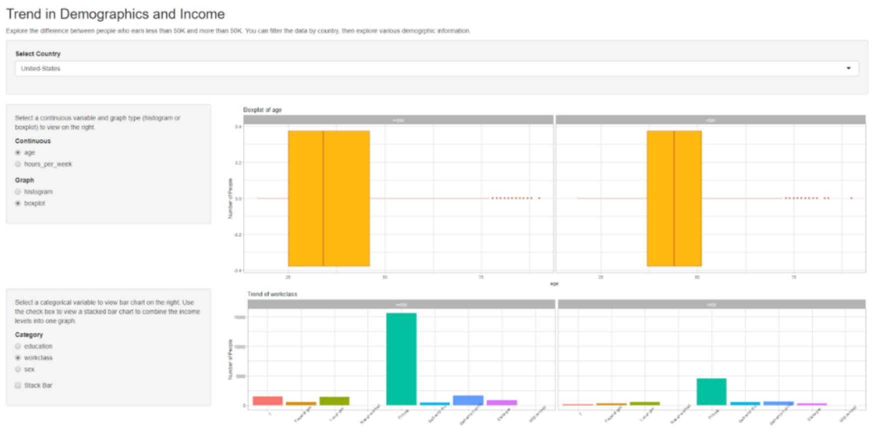
- ● histogram
- ○ boxplot

Select a categorical variable to view bar chart on the right. Use the check box to view a stacked bar chart to combine the income levels into one graph.

Category

- ○ education
- ● workclass
- ○ sex

☑ Stack Bar



Trend in Demographics and Income

Explore the difference between people who earn less than 50K and more than 50K. You can filter the data by country, then explore various demogrphic information.

Select Country

United-States

Select a continuous variable and graph type (histogram or boxplot) to view on the right.

Continuous

- ● age
- ○ hours_per_week

Graph

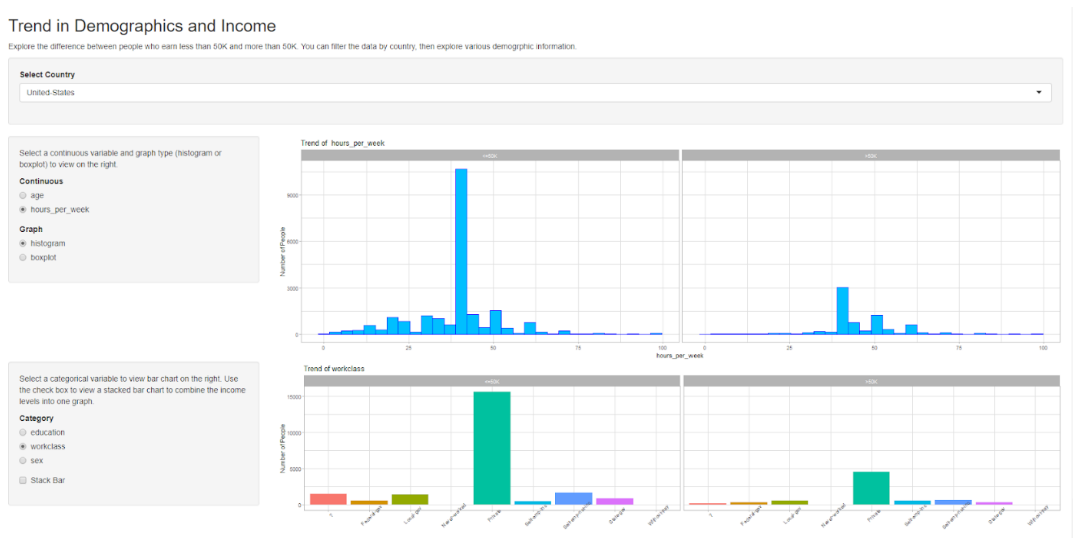
- ○ histogram
- ● boxplot

Select a categorical variable to view bar chart on the right. Use the check box to view a stacked bar chart to combine the income levels into one graph.

Category

- ○ education
- ● workclass
- ○ sex

☐ Stack Bar


Trend in Demographics and Income

Explore the difference between people who earn less than 50K and more than 50K. You can filter the data by country, then explore various demogrphic information.

Select Country

United-States

Select a continuous variable and graph type (histogram or boxplot) to view on the right.

Continuous

- age
- [✓ hours_per_week]

Graph

- [✓ histogram]
- boxplot

Select a categorical variable to view bar chart on the right. Use the check box to view a stacked bar chart to combine the income levels into one graph.

Category

- education
- [✓ workclass]
- sex

☐ Stack Bar

INSIGHTS

1. Self employed workers salary range are not that much appreciable and hence focus should be given on their growth
2. On an average people tend to work for 40 hours per week only.
3. It is inferred from histogram that people working for 40-50 hours are more and the salary for both cases are more in private class
4. The average age lies between 30-35 years for the salary range <= 50k. The average lies around 40 years of age for the salary range >= 50k
5. In salary range <= 50k more number of people are found to be working 40 hours per week
6. IN salary range >= 50k more number of people are found to be working between 40-50 hours per week
7. Private class workers are more among both the salary cases
8. In all workspaces male tend to dominate more than women.

EX NO: 4 DATE:	**Extract text data by using RScrapee based on the term 'A Free Online Data Science Courses' and perform EDA**

AIM: To extract text data by using Rscrapee based on the term 'A Free Online Data Science Courses'

ABOUT DATASET:

This dataset is scrapped using WebScrappe - Rvest funcion from www.greatlearning.com and it contains columns such as Courses, Duration, Level, Enrollments, Ratings, etc. Duration of the course will be in hours, Level will be in beginner or intermediate, Enrollments contains number of people enrolled, Ratings contain the rating of the course in floating. The dataset contains 57 rows and 6 columns.

ALGORITHM:

Step 1: Start the program by including all the required packages and call them using library().

Step 2: Create an R object that hold the url of the page from which the data will be scrapped.

Step 3: Using map_df() create a loop that should iterate one by one till to the last page of the url.

Step 4: Create a Data frame with required fields.

Step 5: Using a Selector gadget extension, copy the required element's Xpath/css and pass it inside the "html_nodes () as an argument.

Step 6: Run the code and View the Data frame created.

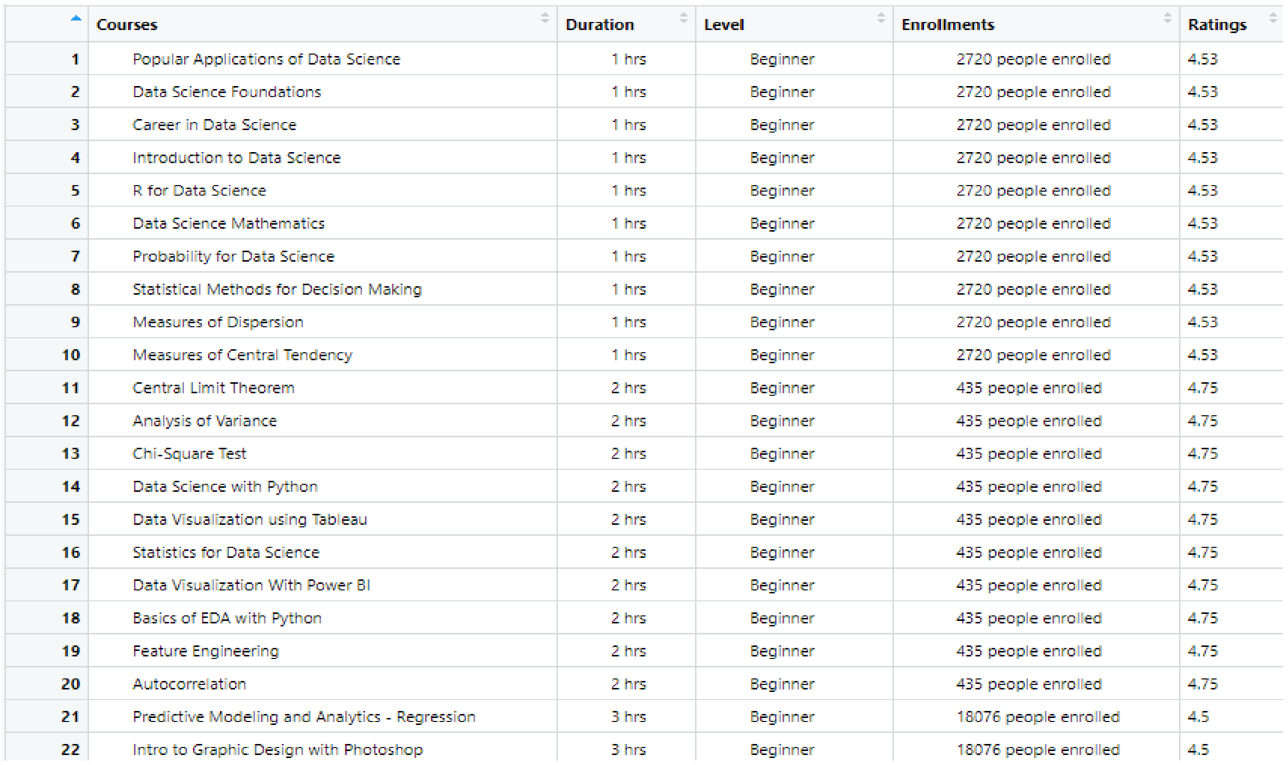
Step 7: Stop the Program

CODE:

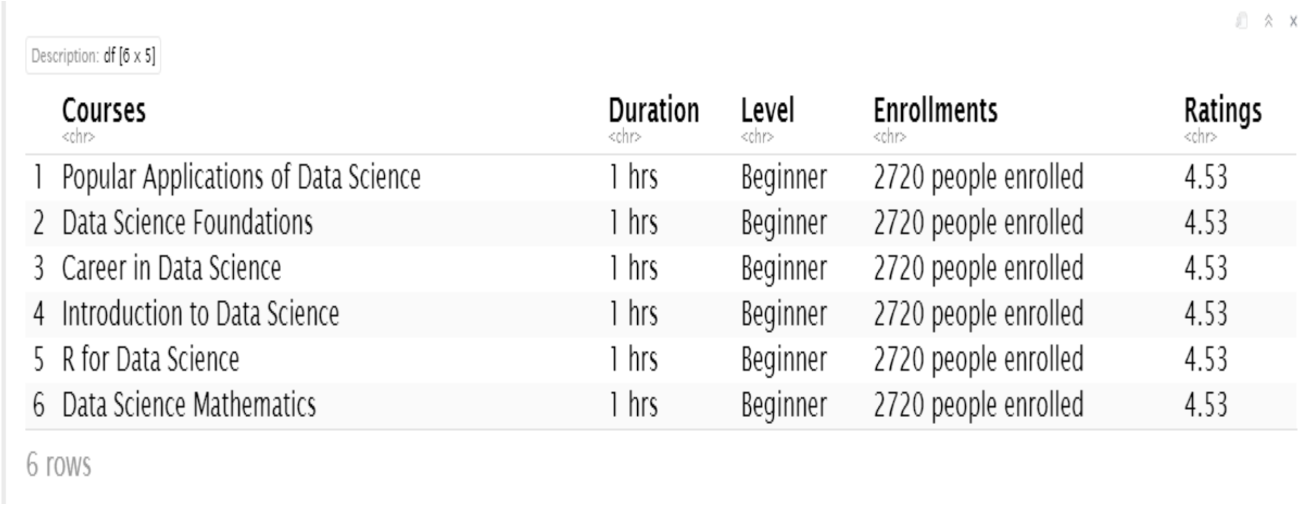
~~~
---
title: "WebScrappe"
author: "Akshaya"
output:
  html_document:
    df_print: paged
---
```{r load libraries, include=FALSE}
library(rvest)
library(purrr)
library(xml2)
library(stringr)
library(knitr)
library(skimr)
~~~
```{r}
url1 = "https://www.mygreatlearning.com/data-science/free-courses?p=%d"

map_df(1:6, function(i){
  page <- read_html(sprintf(url1, i))

  data.frame(Courses = html_text(html_nodes(page, ".course-name")),
             Duration = html_text(html_node(page, ".course-info div:nth-child(1)")),
             Level = html_text(html_node(page, ".dot:nth-child(2)")),
             Enrollments = html_text(html_node(page, ".dot+ .dot")),
             Ratings = html_text(html_node(page, ".course-ratings-label"))
  )
}) -> course
View(course)
colnames(course)
str(course)
```

~~~
```

### Statistical inference

```{r, include=FALSE}

skim(course)

```

```{r}

course$Courses <- trimws(course$Courses, which = c("both"))

course$Duration <- trimws(course$Duration, which = c("both"))

course$Level <- trimws(course$Level, which = c("both"))

course$Enrollments <- trimws(course$Enrollments, which = c("both"))

head(course)

```

### Type conversion

```{r}

course$Ratings <- as.numeric(course$Ratings)

```

Ratings column data type has been converted from character to numeric for analysis

### Filtering conditions

```{r}

table(course$Level)

```

### Histogram

```{r}

hist(course$Ratings, col = rainbow(5), main = "Histogram of Course Ratings")

```

Many courses is rated from 4.4 to 4.6
~~~

OUTPUT:



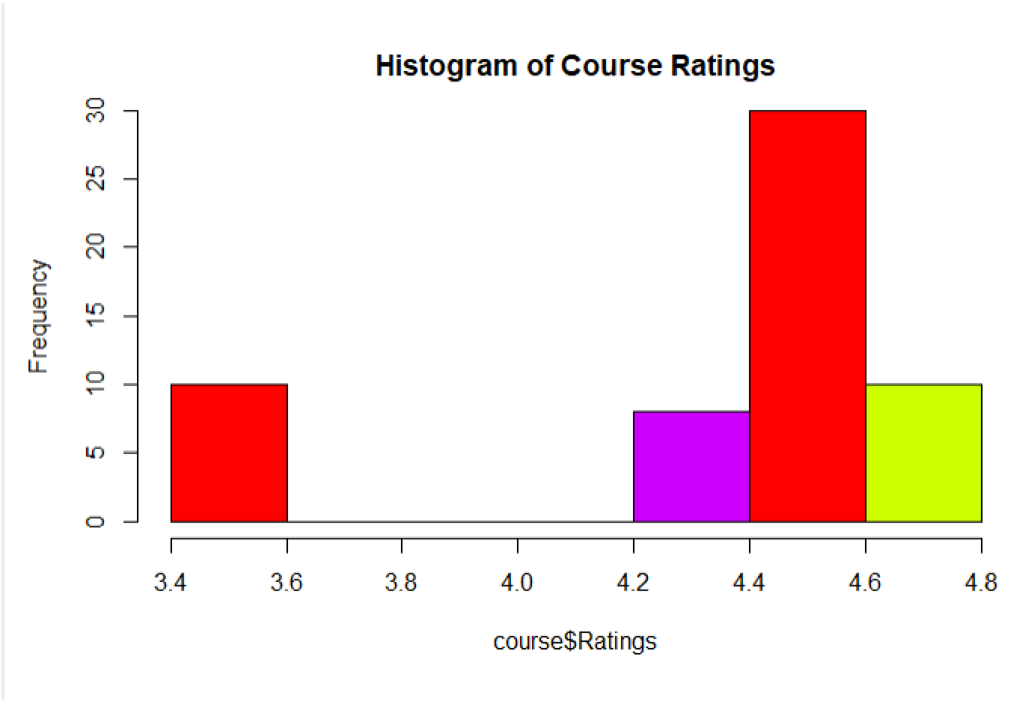
	Courses	Duration	Level	Enrollments	Ratings
1	Popular Applications of Data Science	1 hrs	Beginner	2720 people enrolled	4.53
2	Data Science Foundations	1 hrs	Beginner	2720 people enrolled	4.53
3	Career in Data Science	1 hrs	Beginner	2720 people enrolled	4.53
4	Introduction to Data Science	1 hrs	Beginner	2720 people enrolled	4.53
5	R for Data Science	1 hrs	Beginner	2720 people enrolled	4.53
6	Data Science Mathematics	1 hrs	Beginner	2720 people enrolled	4.53
7	Probability for Data Science	1 hrs	Beginner	2720 people enrolled	4.53
8	Statistical Methods for Decision Making	1 hrs	Beginner	2720 people enrolled	4.53
9	Measures of Dispersion	1 hrs	Beginner	2720 people enrolled	4.53
10	Measures of Central Tendency	1 hrs	Beginner	2720 people enrolled	4.53
11	Central Limit Theorem	2 hrs	Beginner	435 people enrolled	4.75
12	Analysis of Variance	2 hrs	Beginner	435 people enrolled	4.75
13	Chi-Square Test	2 hrs	Beginner	435 people enrolled	4.75
14	Data Science with Python	2 hrs	Beginner	435 people enrolled	4.75
15	Data Visualization using Tableau	2 hrs	Beginner	435 people enrolled	4.75
16	Statistics for Data Science	2 hrs	Beginner	435 people enrolled	4.75
17	Data Visualization With Power BI	2 hrs	Beginner	435 people enrolled	4.75
18	Basics of EDA with Python	2 hrs	Beginner	435 people enrolled	4.75
19	Feature Engineering	2 hrs	Beginner	435 people enrolled	4.75
20	Autocorrelation	2 hrs	Beginner	435 people enrolled	4.75
21	Predictive Modeling and Analytics - Regression	3 hrs	Beginner	18076 people enrolled	4.5
22	Intro to Graphic Design with Photoshop	3 hrs	Beginner	18076 people enrolled	4.5



Description: df [6 x 5]

	Courses <chr>	Duration <chr>	Level <chr>	Enrollments <chr>	Ratings <chr>
1	Popular Applications of Data Science	1 hrs	Beginner	2720 people enrolled	4.53
2	Data Science Foundations	1 hrs	Beginner	2720 people enrolled	4.53
3	Career in Data Science	1 hrs	Beginner	2720 people enrolled	4.53
4	Introduction to Data Science	1 hrs	Beginner	2720 people enrolled	4.53
5	R for Data Science	1 hrs	Beginner	2720 people enrolled	4.53
6	Data Science Mathematics	1 hrs	Beginner	2720 people enrolled	4.53

6 rows



INSIGHTS:

From the data set we can find beginner level courses are more when compared to intermediate level. More number of courses contain the word 'data' in it as the dataset has been scrapped using the sentence 'data science'. Many courses were offered 4.4-4.6 rating by the users. Majority of the courses are 2 hours.

From this we can infer there are 50 Beginner and 7 intermediate courses.

From this we can infer there are 50 Beginner and 7 intermediate courses

From this we can infer that many courses is rated from 4.4 to 4.

EX NO: 05 DATE:	**Download a data set from data.gov.in to do all data manipulations**

AIM: To download a dataset from data.gov.in to do all data manipulations

ABOUT DATASET:

This dataset is taken from data.gov.in. In this Data set we come to know about the Total number of cases till 26.5.2022 District wise.

ALGORITHM:

Step 1: Start the program by including all the required packages and call them using library().

Step 2: Summarize the dataset using summary() function to know about the statistical inference.

Step 3: Find unique values in each column and subset the dataframe as per the requirement

Step 4: Apply dplyr functions such as select, filter, group_by, arrange, rename, sort to get insights from the data

Step 5: Mutate the columns to get the desired values and append it to the dataset

Step 6: Run the code, get inference and insights from the manipulated data

Step 7: Stop the Program

CODE:

~~~
        ---
        title: "Data Manipulation"
        author: "AKSHAYA.N"
        date: "2022/30/05"
        ---
### DISCREPTION:
        Coronavirus disease (COVID-19) is an infectious disease caused by the SARS-CoV-2 virus. Most
people infected with the virus will experience mild to moderate respiratory illness and recover without
requiring special treatment.
        This data set shows the Details of Statistics showing the Indigenous.Cases.till.25.05.2022,
Indigenous.Cases.on.26.05.2022 , Imported.Cases.till.25.05.2022, Imported.Cases.on.26.05.2022  and
Total.cases.till.26.05.2022.


```{r setup, include=FALSE}
library(tidyverse)
data()
covid19=read.csv(file.choose(),header = TRUE)
View(covid19)
attach(covid19)
summary(covid19)
```
   # R glimpse glimpse () function get a glimpse of your data. dplyr package is needed to run this function.
```{r}
library(dplyr)
glimpse(covid19)
```
### The head () function returns the first part of a vector, Matrix, table, data frame, or function
```{r pressure, echo=FALSE}
head(covid19)
```
~~~

# The skimr returns a summary with the name of the dataset and the number of rows and columns

```{r}
library(skimr)

library(janitor)

skim_without_charts(covid19)
```

# selecting one coloumn

### Here we are selecting only one coloumn(District) so We get a dataframe with only the “District” column.

```{r}
covid19%>%select(District)
```

# selecting all columns other than district

### Here we are selecting all coloumn expect district so, We get a dataframe without the “Districts” column.

```{r}
covid19%>%select(-District)
```

#rename a variable

rename () function in R Language is used to rename the column names of a data frame, based on the older names, Here we have changed the name of the District into Districts

```{r}
covid19

covid19 %>%

  rename("Districts" ="District") %>%

  glimpse()
```

# clean

The easiest option to replace spaces in column names is with the clean.names () function. This R function creates syntactically correct column names by replacing blanks with an underscore. Here the coloumn names are replaced with Underscores

```{r}
clean_names(covid19)
```

# unique

The unique () function in R is used to eliminate or delete the duplicate values or the rows present in the vector, data frame, or matrix as well. In Kancheepuram and Perambalur Imported.Cases.till.25.05.2022 is 3 duplicate value is deleted using unique function.

```{r}
unique(covid19$Imported.Cases.till.25.05.2022)
```

###ORGANISING THE DATA (filter(),group_by(),arrange(),summarise())

### filter

The filter () function is used to produce a subset of the data frame, retaining all rows that satisfy the specified conditions. In this data set in column Indigenous.Cases.till.25.05.2022 we filter the cases with 0. There are three districts with 0 cases.

```{r}
covid19%>%filter(Indigenous.Cases.till.25.05.2022 =="0")
```

### In this data set in column Indigenous.Cases.on.26.05.2022 we filter the cases with 3. There are two districts with 3 cases.

```{r}
covid19%>%filter(Indigenous.Cases.on.26.05.2022 == "3")
```

### In this data set in column Imported.Cases.till.25.05.2022 we filter the cases with 3. There are two districts with 3 cases.

```{r}
covid19%>%filter(Imported.Cases.till.25.05.2022=="3")
```

### Like wise we filter Total.cases.till.26.05.2022 as 14461, There are two districts with 14458 cases.

```{r}
covid19%>%filter(Total.cases.till.26.05.2022=="14461")
```

### group_by & summarise

group_by allows you to group by a one or more variables. summarize: summarize creates a new data.frame containing calculated summary information about a grouped variable. Here the data set is grouped by district and summarize imformation about Imported.Cases.till.25.05.2022

```{r}
covid19%>%group_by(District)%>%drop_na()%>%summarise(Imported.Cases.till.25.05.2022)
```

### Here the data set is grouped by district and summarize imformation about Total.cases.till.26.05.2022

```{r}
covid19%>%group_by(District)%>%drop_na()%>%summarise(Total.cases.till.26.05.2022)
```

### arrange

The arrange () function is used to rearrange rows in ascending or descending order. In this dataset District column is arranged in decending order.

```{r}
covid19%>%arrange(desc(District))
```

### arrange by group

R arrange () orders the rows of a data frame by the values of selected columns. , In this dataset Indigenous.Cases.on.26.05.2022 arranged in decending order.

```{r}
covid19%>%arrange(desc(Indigenous.Cases.on.26.05.2022), .by_group = TRUE)
```

```{r}
covid19%>%arrange(across(starts_with("T")))
```

~~~~
### slice

```{r}

covid19%>%slice(1L)

```

```{r}

covid19%>%slice(n())

```

```{r}

covid19%>%slice_min(District,n=15)

```

#TRANSFORMING DATA  mutate(),unite()


### mutate

        The mutate () function is a function for creating new variables.

Here I have added New column named Result to show the result if Total.cases.till.26.05.2022 > 5000 then
"Many cases", if not more then 5000 then "Not many cases"

```{r}

covid19 %>%

mutate(Result= if_else(Total.cases.till.26.05.2022 > 5000, "Many cases", "Not many cases"))

```

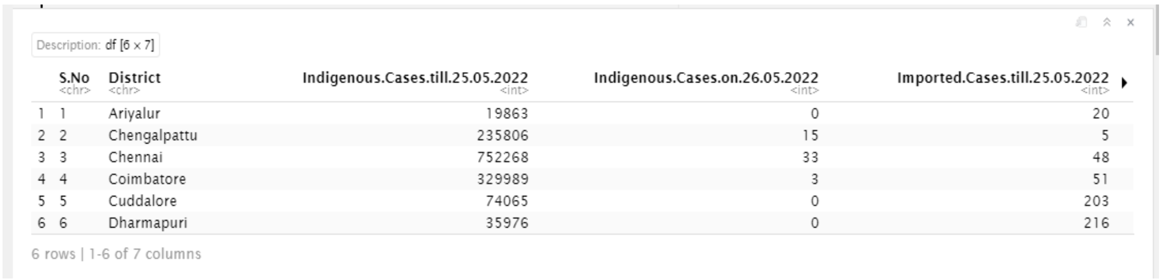
### unite

         The unite () function from the tidyr package can be used to unite multiple data frame columns into a
single column. Here in this data set i have choosed columns District and Total.cases.till.26.05.2022.

```{r}

covid19%>%unite("a",District:Total.cases.till.26.05.2022,remove=FALSE)

```
~~~~

OUTPUT

Head():

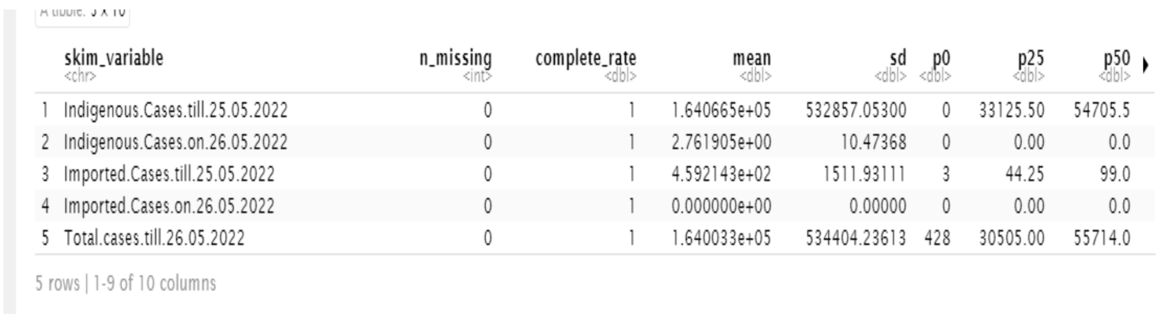


Description: df [6 × 7]

	S.No <chr>	District <chr>	Indigenous.Cases.till.25.05.2022 <int>	Indigenous.Cases.on.26.05.2022 <int>	Imported.Cases.till.25.05.2022 <int>
1	1	Ariyalur	19863	0	20
2	2	Chengalpattu	235806	15	5
3	3	Chennai	752268	33	48
4	4	Coimbatore	329989	3	51
5	5	Cuddalore	74065	0	203
6	6	Dharmapuri	35976	0	216

6 rows | 1-6 of 7 columns

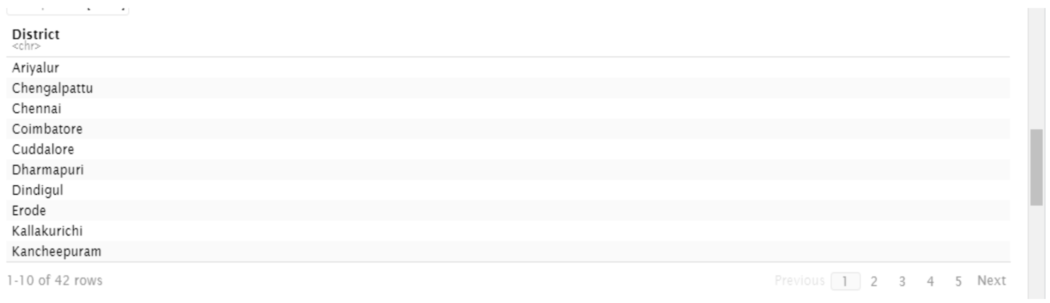
Skimmer:



	skim_variable <chr>	n_missing <int>	complete_rate <dbl>	mean <dbl>	sd <dbl>	p0 <dbl>	p25 <dbl>	p50 <dbl>
1	Indigenous.Cases.till.25.05.2022	0	1	1.640665e+05	532857.05300	0	33125.50	54705.5
2	Indigenous.Cases.on.26.05.2022	0	1	2.761905e+00	10.47368	0	0.00	0.0
3	Imported.Cases.till.25.05.2022	0	1	4.592143e+02	1511.93111	3	44.25	99.0
4	Imported.Cases.on.26.05.2022	0	1	0.000000e+00	0.00000	0	0.00	0.0
5	Total.cases.till.26.05.2022	0	1	1.640033e+05	534404.23613	428	30505.00	55714.0

5 rows | 1-9 of 10 columns

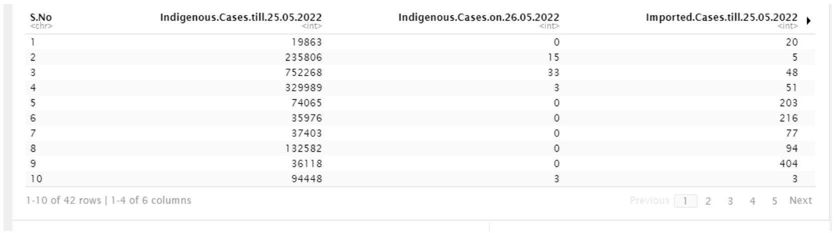
selecting one coloumn:



District <chr>
Ariyalur
Chengalpattu
Chennai
Coimbatore
Cuddalore
Dharmapuri
Dindigul
Erode
Kallakurichi
Kancheepuram

1-10 of 42 rows Previous 1 2 3 4 5 Next

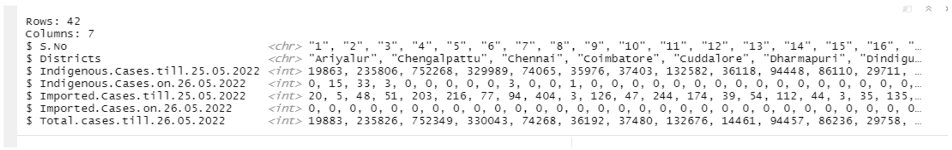
selecting all columns other than district:



S.No <chr>	Indigenous.Cases.till.25.05.2022 <int>	Indigenous.Cases.on.26.05.2022 <int>	Imported.Cases.till.25.05.2022 <int>
1	19863	0	20
2	235806	15	5
3	752268	33	48
4	329989	3	51
5	74065	0	203
6	35976	0	216
7	37403	0	77
8	132582	0	94
9	36118	0	404
10	94448	3	3

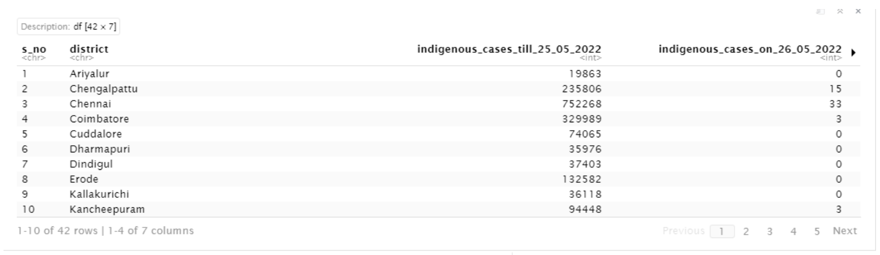
1-10 of 42 rows | 1-4 of 6 columns Previous 1 2 3 4 5 Next

rename a variable:



```
Rows: 42
Columns: 7
$ S.No                             <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "…
$ Districts                        <chr> "Ariyalur", "Chengalpattu", "Chennai", "Coimbatore", "Cuddalore", "Dharmapuri", "Dindigu…
$ Indigenous.Cases.till.25.05.2022 <int> 19863, 235806, 752268, 329989, 74065, 35976, 37403, 132582, 36118, 94448, 86110, 29711, …
$ Indigenous.Cases.on.26.05.2022   <int> 0, 15, 33, 3, 0, 0, 0, 0, 0, 3, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
$ Imported.Cases.till.25.05.2022   <int> 20, 5, 48, 51, 203, 216, 77, 94, 404, 3, 126, 47, 244, 174, 39, 54, 112, 44, 3, 35, 135,…
$ Imported.Cases.on.26.05.2022     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ Total.cases.till.26.05.2022      <int> 19883, 235826, 752349, 330043, 74268, 36192, 37480, 132676, 14461, 94457, 86236, 29758, …
```

clean:

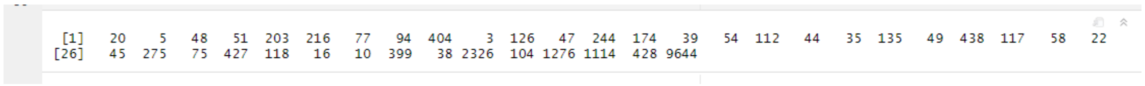


Description: df [42 × 7]

s_no <chr>	district <chr>	indigenous_cases_till_25_05_2022 <int>	indigenous_cases_on_26_05_2022 <int>
1	Ariyalur	19863	0
2	Chengalpattu	235806	15
3	Chennai	752268	33
4	Coimbatore	329989	3
5	Cuddalore	74065	0
6	Dharmapuri	35976	0
7	Dindigul	37403	0
8	Erode	132582	0
9	Kallakurichi	36118	0
10	Kancheepuram	94448	3

1-10 of 42 rows | 1-4 of 7 columns Previous 1 2 3 4 5 Next

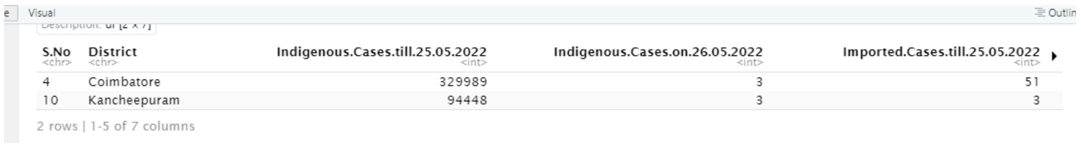
unique:



```
 [1]   20    5   48   51  203  216   77   94  404    3  126   47  244  174   39   54  112   44   35  135   49  438  117   58   22
[26]   45  275   75  427  118   16   10  399   38 2326  104 1276 1114  428 9644
```

ORGANISING THE DATA:

1. Filter

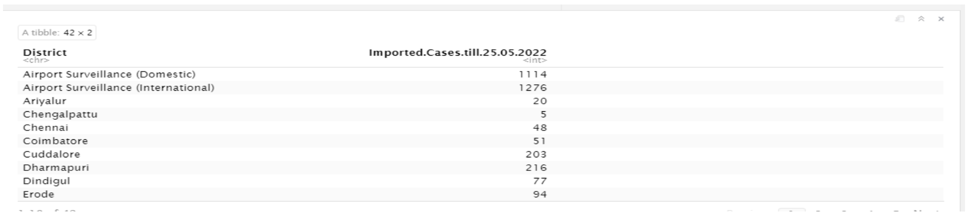


Description: df [2 × 7]

S.No <chr>	District <chr>	Indigenous.Cases.till.25.05.2022 <int>	Indigenous.Cases.on.26.05.2022 <int>	Imported.Cases.till.25.05.2022 <int>
4	Coimbatore	329989	3	51
10	Kancheepuram	94448	3	3

2 rows | 1-5 of 7 columns

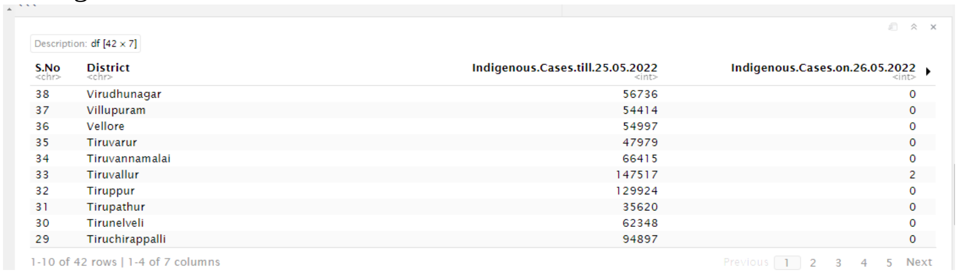
2. group_by & summarise



A tibble: 42 × 2

District <chr>	Imported.Cases.till.25.05.2022 <int>
Airport Surveillance (Domestic)	1114
Airport Surveillance (International)	1276
Ariyalur	20
Chengalpattu	5
Chennai	48
Coimbatore	51
Cuddalore	203
Dharmapuri	216
Dindigul	77
Erode	94

3. Arrange

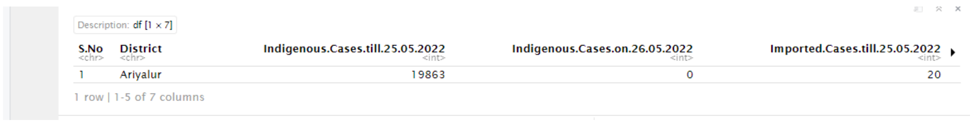


Description: df [42 × 7]

S.No <chr>	District <chr>	Indigenous.Cases.till.25.05.2022 <int>	Indigenous.Cases.on.26.05.2022 <int>
38	Virudhunagar	56736	0
37	Villupuram	54414	0
36	Vellore	54997	0
35	Tiruvarur	47979	0
34	Tiruvannamalai	66415	0
33	Tiruvallur	147517	2
32	Tiruppur	129924	0
31	Tirupathur	35620	0
30	Tirunelveli	62348	0
29	Tiruchirappalli	94897	0

1-10 of 42 rows | 1-4 of 7 columns   Previous 1 2 3 4 5 Next

4. Slice



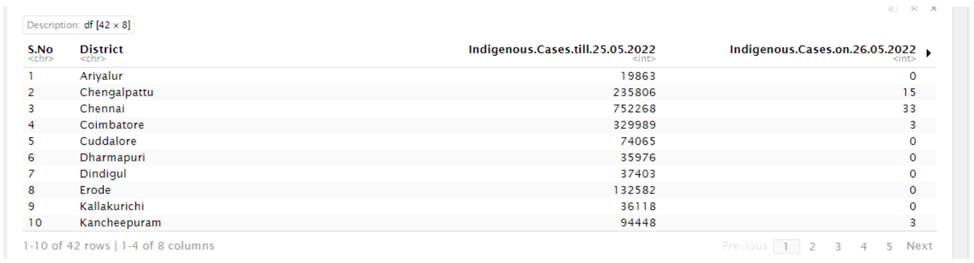
Description: df [1 × 7]

S.No <chr>	District <chr>	Indigenous.Cases.till.25.05.2022 <int>	Indigenous.Cases.on.26.05.2022 <int>	Imported.Cases.till.25.05.2022 <int>
1	Ariyalur	19863	0	20

1 row | 1-5 of 7 columns

TRANSFORMING DATA:

1. Mutate

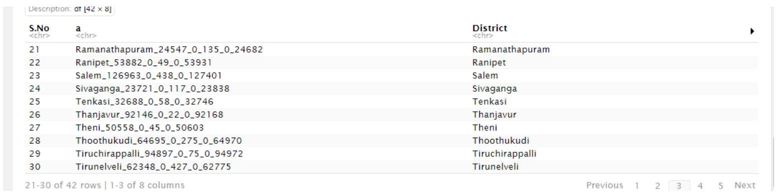


Description: df [42 × 8]

S.No <chr>	District <chr>	Indigenous.Cases.till.25.05.2022 <int>	Indigenous.Cases.on.26.05.2022 <int>
1	Ariyalur	19863	0
2	Chengalpattu	235806	15
3	Chennai	752268	33
4	Coimbatore	329989	3
5	Cuddalore	74065	0
6	Dharmapuri	35976	0
7	Dindigul	37403	0
8	Erode	132582	0
9	Kallakurichi	36118	0
10	Kancheepuram	94448	3

1-10 of 42 rows | 1-4 of 8 columns   Previous 1 2 3 4 5 Next

2. Unite



Description: df [42 × 8]

S.No <chr>	a <chr>	District <chr>
21	Ramanathapuram_24547_0_135_0_24682	Ramanathapuram
22	Ranipet_53882_0_49_0_53931	Ranipet
23	Salem_126963_0_438_0_127401	Salem
24	Sivaganga_23721_0_117_0_23838	Sivaganga
25	Tenkasi_32688_0_58_0_32746	Tenkasi
26	Thanjavur_92146_0_22_0_92168	Thanjavur
27	Theni_50558_0_45_0_50603	Theni
28	Thoothukudi_64695_0_275_0_64970	Thoothukudi
29	Tiruchirappalli_94897_0_75_0_94972	Tiruchirappalli
30	Tirunelveli_62348_0_427_0_62775	Tirunelveli

21-30 of 42 rows | 1-3 of 8 columns   Previous 1 2 3 4 5 Next

INSIGHTS :

In this Data set we come to know about the Total number of cases till 26.5.2022 District wise by using this dataset we have performed Data manipulation. Through Data manipulation we can perform various functions using that function we can filter the particular column we need arrange them in Ascending and descending. By filter we can find similar number of cases in Different Districts.

EX NO: 6 DATE:	**Download a data set from Kaggle, analyse with all plots and visualize all plots in flex dashboard.**

AIM:

To Download a Data set from Kaggle, analyse with all plots and visualize all plots in flex dashboard.

ABOUT DATA:

This dataset is taken from Kaggle. This data set was created to list all shows available on Amazon Prime streaming, and analyze the data to find interesting facts. This data was acquired in May 2022 containing data available in the United States.

ALGORITHM:

Step 1: Start the program by creating a markdown with a Flex dashboard template.

Step 2: Include all necessary libraries.

Step 3: Read the data and look for all possible information about the data using various data manipulation methods in R.

Step 4: Analyse the data and visualise all possible plots in different individual menu in Dashboard

Step 5: Once completed all visualisations snippets, knit the code to visualise as a Dashboard

Step 6: Stop the Program

CODE:

~~~
---
title: "FLEXDASHBOARD"
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: scroll
    theme: darkly
    storyboard: True
    social: menu
    source_code: embed
---
```{r setup, include=FALSE}
library(flexdashboard)
library(dplyr)
library(DT)
library(ggplot2)
library(tidyverse)
library(lattice)
library(skimr)
library(janitor)
~~~

# OVER VIEW OF DATASET
```{r}
amazon <- read.csv(file.choose(), na.strings = c("", "NA"), stringsAsFactors =FALSE)
summary(amazon)
attach(amazon)
view(amazon)
skim_without_charts(amazon)
```
```
```

# DESCRIPTION

Here we can see the short Description about the Coloumn Names given:

~~~
```{r}

library(plotly)

values_table1 <- rbind(c('id', 'type', 'title', 'age_certificate', 'release_year', 'production_country',
'runtime', 'release_year', 'season' ,
'genres','imdb_id','imdb_score','imdb_votes','tmdb_popularity','tmdb_score', 'description'),
c("Unique ID for every Movie / TV Show",

     "Identifier - A Movie or TV Show",

     "Title of the Movie or TV Show",

     "Meeting the proof of age requirement",

    "Year on which movie is released",

    "Country where the movie / show was produced",

    "The duration of a motion picture",

    "Actual release year of the Movie / TV Show",

    "Number of parts",

    "Category of movie",

    "Internet Movie Database ID",

    " average score is determined through the scores given by the top-rated critics","Number of votes
for imdb"," movie and TV database popularity"," average score is determined through the scores
given by the top-rated critics for tv show"))

fig_table1 <- plot_ly(

  type = 'table',

  columnorder = c(1,2),

  columnwidth = c(12,12),

  header = list(

    values = c('<b>VARIABLES</b><br>', '<b>DESCRIPTION</b>'),

    line = list(color = '#506784'),

    fill = list(color = '#119DFF'),

    align = c('left','center'),

    font = list(color = 'white', size = 12),

    height = 40

  ),
~~~

~~~~
  cells = list(
    values = values_table1,
    line = list(color = '#506784'),
    fill = list(color = c('#25FEFD', 'white')),
    align = c('left', 'left'),
    font = list(color = c('#506784'), size = 12),
    height = 30
    ))
fig_table1
```
# Histogram Representation
```{r}
hist(amazon$release_year, col = "green", main = 'RELEASE YEAR')
```
     #  The release of TV shows and movies were high duing the year of 2000 to 2020


```{r}
hist(amazon$runtime , col = "blue" , main = 'RUNTIME')
```
     # The maximum run time of Movies and Tv show is 200 but most of TV show and movies runtime
lies between 50 to 100


# plot Representation


```{r}
ggplot(data=amazon ,aes(x=type,y=release_year))+
  geom_point(color='green')
```
     # From this plot we come to know that Early stage of amazon prime there were only movies, after
1950 only Tv shows were introduced
~~~~

```{r}
ggplot(data=amazon ,aes(x=release_year,y=imdb_score))+
  geom_point(color='green')
```

    # Internet Movie Database score is high in 2020's

```{r}
ggplot(data=amazon ,aes(x=release_year,y=tmdb_score))+
  geom_point(color='green')
```

     # In TMDB (community build movie and Tv database) also high during the year 2020 onwards

```{r}
ggplot(data=amazon ,aes(x=release_year,y=tmdb_popularity))+
  geom_point(color='green')
```

     # Initially there were no tv shows in amazon and now the TMDB propularity gradually became high in 2020's

# TYPE OF CONTENT

Here we can see the amount of content in Movies and Tv shows, we come to know that Movie content is high in amazon prime

```{r}
library(tibble)
library(dplyr)
library(ggplot2)
amount_by_type = amazon %>% group_by(type) %>% summarise(count = n())
ggplot(data = amount_by_type, aes(x= type, y= count, fill= type))+
  geom_bar(colour ="black", size= 0.8, fill = "dark green" ,stat = "identity")+
  guides(fill= none)+
  xlab("amazon Content by Type") + ylab("Amount of amazon Content")+
  ggtitle("Amount of amazon Content By Type")
```

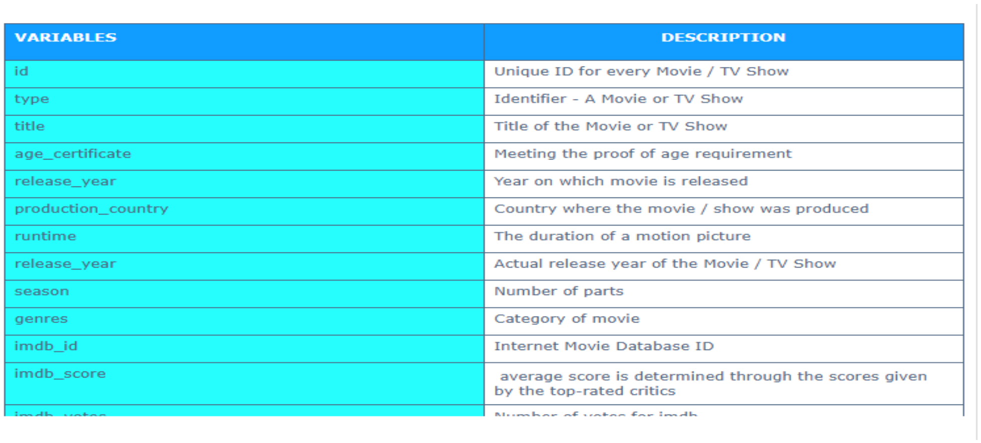
~~~
### Plot for GENERS
```{r}
k <- strsplit(amazon$genres, split = ", ")
amazon_genres = data.frame(type = rep(amazon$type, sapply(k, length)), genres = unlist(k))
amazon_genres$genres <- as.character(amazon_genres$genres)
amount_by_genres <- na.omit(amazon_genres) %>%
  group_by(genres, type) %>%
  summarise(count = n())
u <- reshape(data=data.frame(amount_by_genres),idvar="genres",
                          v.names = "count",
                          timevar = "type",
                          direction="wide") %>%
                          top_n(10)
names(u)[2] <- "Number_of_Movies"
names(u)[3] <- "Number_of_TV_Shows"
u <- u[order(desc(u$Number_of_Movies +u$Number_of_TV_Shows)),]
library(ggplot2)
ggplot(u, aes(Number_of_Movies, Number_of_TV_Shows, colour=genres))+
  geom_point(size=5)+
  xlab("Number of Movies") + ylab("Number of TV Shows")+
  ggtitle("Amount of amazon Content By GENERS")
```
     #  From this Visual representation we come to know in which genres more number of movies and
Tv shows released: In drama genres more number of movies and tv shows are released and In
Animation less number of Movies and Tv shows are released
```{r}
amazon$imdb_votes[is.na(amazon$imdb_votes)]=mean(amazon$imdb_votes,na.rm = TRUE)
amazon$imdb_score[is.na(amazon$imdb_score)]=mean(amazon$imdb_score,na.rm = TRUE)
amazon$tmdb_popularity[is.na(amazon$tmdb_popularity)]=mean(amazon$tmdb_popularity,na.rm =
TRUE)
amazon$tmdb_score[is.na(amazon$tmdb_score)]=mean(amazon$tmdb_score,na.rm = TRUE)
```
~~~

~~~~
# correlation
```{r}
library(corrplot)
corplot =amazon[c(12,15)]
cor(corplot)
```
   # IMDB and TMDB scores are highly correlated
### corrplot
```{r}
corrplot(cor(corplot))
```
   # corrplot showing correlation between IMDB and TMDB scores

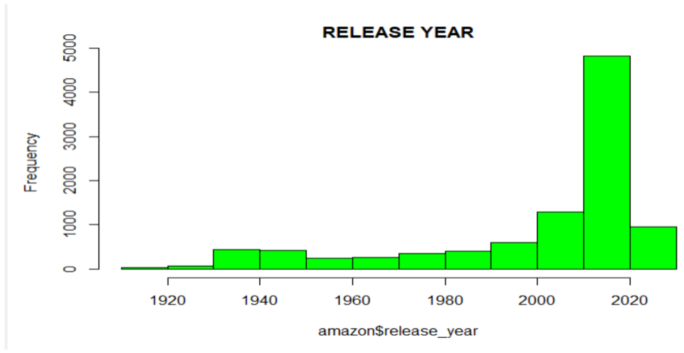

```{r}
ggplot(data=amazon,aes(imdb_score,tmdb_score))+
  geom_point(color='red')
```

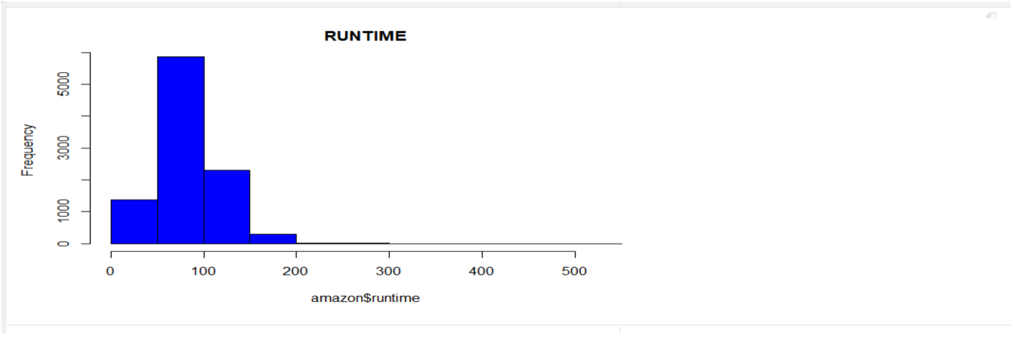

# download
```{r}
datatable(amazon,extensions="Buttons",options=list(dom="Bftrip",Buttons=c('copy','print','csv','pd
f')))
```
~~~~

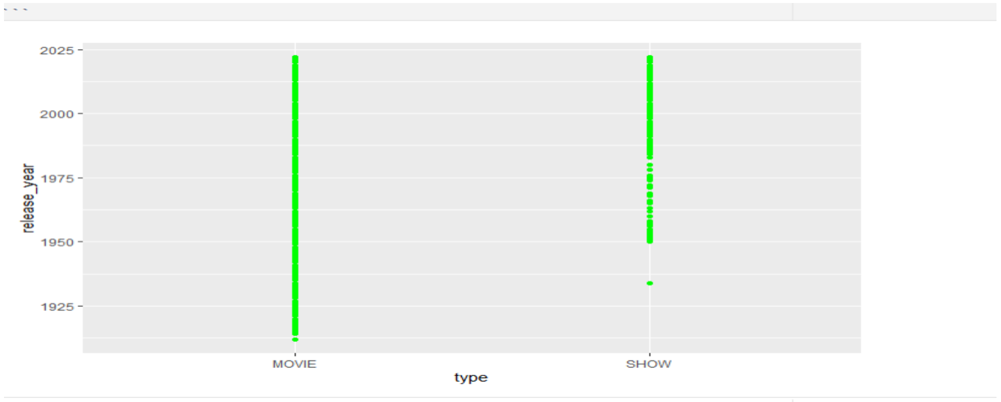
OUTPUT:

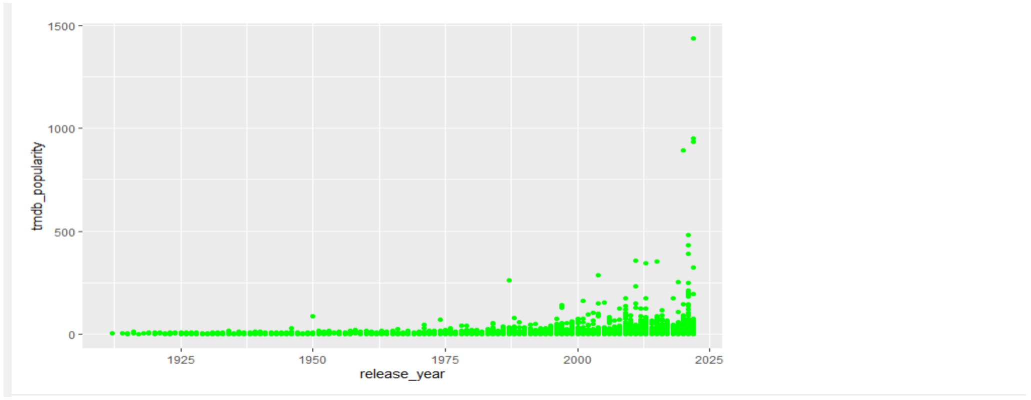


VARIABLES	DESCRIPTION
id	Unique ID for every Movie / TV Show
type	Identifier - A Movie or TV Show
title	Title of the Movie or TV Show
age_certificate	Meeting the proof of age requirement
release_year	Year on which movie is released
production_country	Country where the movie / show was produced
runtime	The duration of a motion picture
release_year	Actual release year of the Movie / TV Show
season	Number of parts
genres	Category of movie
imdb_id	Internet Movie Database ID
imdb_score	average score is determined through the scores given by the top-rated critics

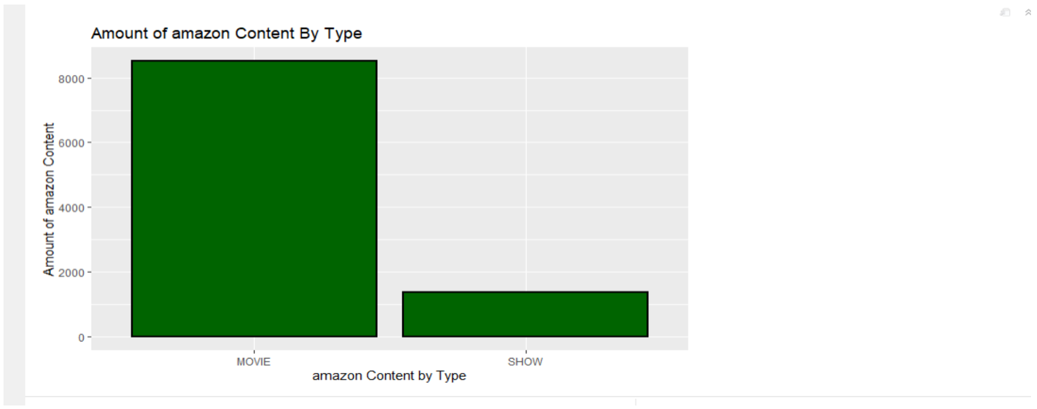




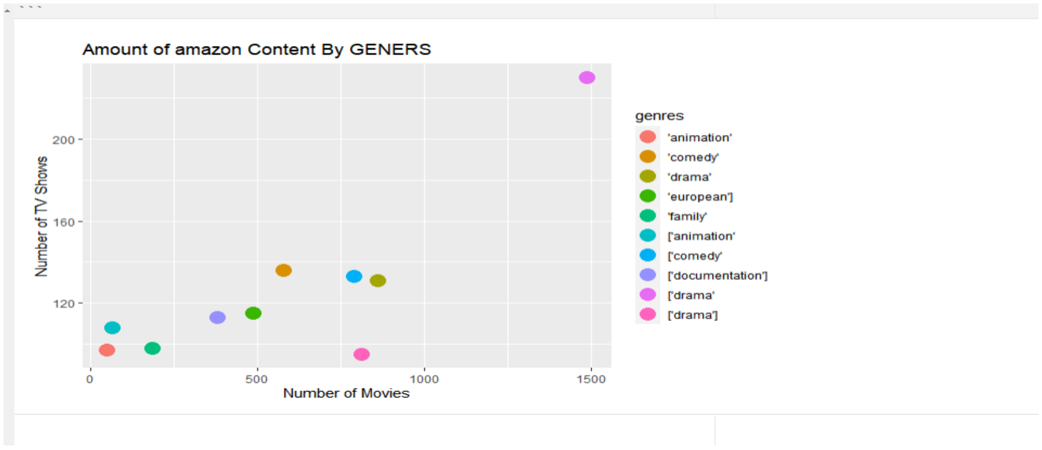




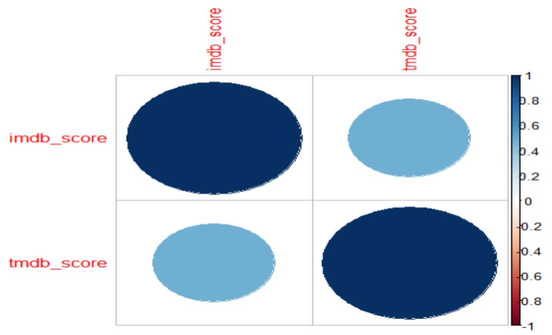
tmdb_popularity
1500
1000
500
0
1925
1950
1975
2000
2025
release_year



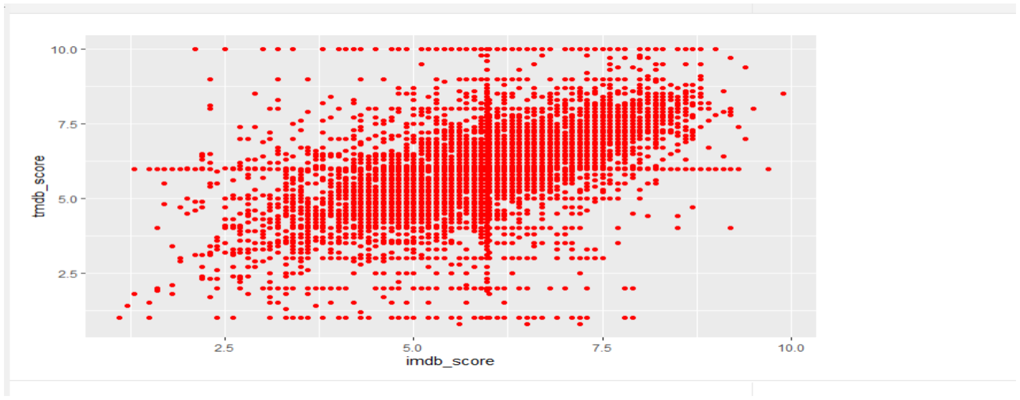
Amount of amazon Content By Type
Amount of amazon Content
8000
6000
4000
2000
0
MOVIE
SHOW
amazon Content by Type



Amount of amazon Content By GENERS
Number of TV Shows
200
160
120
0
500
1000
1500
Number of Movies
genres
'animation'
'comedy'
'drama'
'european']
'family'
['animation'
['comedy'
['documentation']
['drama'
['drama']



imdb_score
tmdb_score
imdb_score
tmdb_score
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1



INSIGHTS

This data set was created to list all shows available on Amazon Prime streaming, and analyze the data to find interesting facts. This data was acquired in May 2022 containing data available in the United States. The release of TV shows and movies were high duing the year of 2000 to 2020. The maximum run time of Movies and Tv show is 200 but most of TV show and movies runtime lies between 50 to 100. From this plot we come to know that Early stage of amazon prime there were only movies, after 1950 only Tv shows were introduced. Movie content is high in amazon prime. In drama genres more number of movies and tv shows are released and In Animation less number of Movies and Tv shows are released.

EX NO: 7 DATE:	**Download a data set from Kaggle, and do a Statistical Analysis –** Mean, SD, Variance, Confidence Interval.

AIM:

To Download a Data set from Kaggle, and do a Statistical Analysis –Mean, SD, Variance, Confidence Interval.

ABOUT DATASET:

This dataset is taken from Kaggle**.** This dataset tells details about students name and the marks scored by them in each subjects.

ALGORITHM:

Step 1: Start the program by opening a new script.

Step 2: Include all necessary libraries.

Step 3: Read the data and describe all statistical summary.

Step 4: Visualize the summary using density plot.

Step 5: Once Visualisation completed, frame hypothesis and run T-Test.

Step 6: Stop the Program

CODE:

~~~
    ---
title: "STATISTICAL ANALYSIS"
author: "AKSHAYA.N"
date : 31/05/2022
---
```{r setup, include=FALSE}
library(dplyr)
```


### This dataset tells details about students name and the marks scored by them in each subjects


```{r}
marks = read.csv(file.choose(),header = TRUE)
View(marks)
summary(marks)
```


### here we select the columns with numerics to perform Statistical Analysis


```{r}
ma = select(marks,c(3:8))
summary(ma)
View(ma)
attach(ma)
```



# Hypothesis testing
~~~

```{r}

mean(marks$English)
sd(marks$English)
hist(ma$English,prob=TRUE,main = "Perecentage of the marks")

mean(marks$Bengali)
sd(marks$Bengali)
hist(ma$Bengali,prob=TRUE,main = "Perecentage of the marks")

mean(marks$Hindi)
sd(marks$Hinhi)
hist(ma$Hindi,prob=TRUE,main = "Perecentage of the marks")

mean(marks$Maths)
sd(marks$Maths)
hist(ma$Maths,prob=TRUE,main = "Perecentage of the marks")

mean(marks$History)
sd(marks$History)
hist(ma$History,prob=TRUE,main = "Perecentage of the marks")

mean(marks$Geography)
sd(marks$Geography)
hist(ma$Geography,prob=TRUE,main = "Perecentage of the marks")
```

```
# Highest percentage of marks scored by students in English is in 80'S.
# Highest percentage of the marks scored by students in Bengali is in 75'S.
# Highest percentage of the marks scored by students in Hindhi is between 70 to 90.
# Highest percentage of the marks scored by students in Maths between 90 to 100
# Highest percentage of the marks scored by students in History is between 90 to 95
```

# Highest percentage of the marks scored by students in Geography is between 90 to 95

# one sample t-test

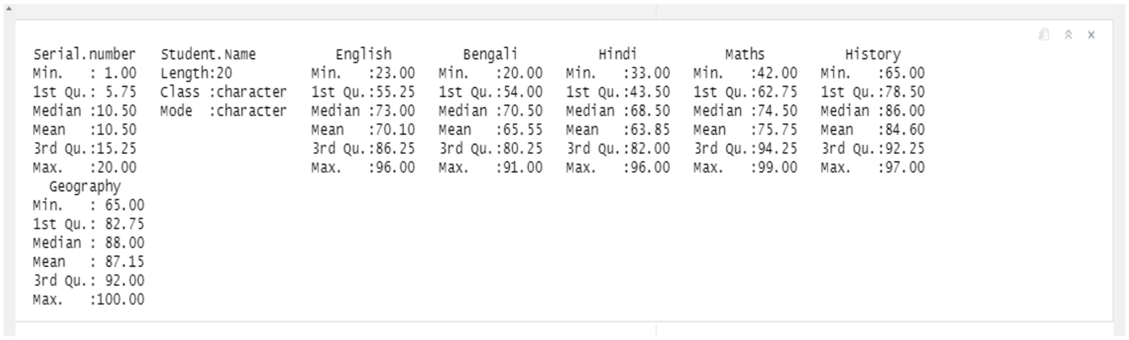
~~~
```{r}
eng = sample(marks$English)
t.test(eng,mu=70.1)
x1=data.frame(eng)

x1%>%
 ggplot(data=x1,mapping=aes(x))+geom_density(fill="red")+
  geom_vline(aes(xintercept=mean(x)),color="black",linetype="dashed",size=0.5)



ben = sample(marks$Bengali)
t.test(ben,mu=65.55)
x2=data.frame(ben)

x2%>%
 ggplot(data=x1,mapping=aes(x))+geom_density(fill="blue")+
  geom_vline(aes(xintercept=mean(x)),color="black",linetype="dashed",size=0.5)



hin = sample(marks$Hindi)
t.test(hin,mu=63.85)
x3=data.frame(hin)

x3%>%
 ggplot(data=x1,mapping=aes(x))+geom_density(fill="red")+
  geom_vline(aes(xintercept=mean(x)),color="black",linetype="dashed",size=0.5)

mat = sample(marks$Maths)
~~~

```
t.test(mat,mu=75.75)

x4=data.frame(mat)


x4%>%

 ggplot(data=x1,mapping=aes(x))+geom_density(fill="red")+

  geom_vline(aes(xintercept=mean(x)),color="black",linetype="dashed",size=0.5)



his = sample(marks$History)

t.test(his,mu=84.6)

x5=data.frame(his)


x5%>%

 ggplot(data=x1,mapping=aes(x))+geom_density(fill="red")+

  geom_vline(aes(xintercept=mean(x)),color="black",linetype="dashed",size=0.5)



geo = sample(marks$Geography)

t.test(geo,mu=87.15)

x6=data.frame(mat)


x6%>%

 ggplot(data=x1,mapping=aes(x))+geom_density(fill="red")+

  geom_vline(aes(xintercept=mean(x)),color="black",linetype="dashed",size=0.5)



```

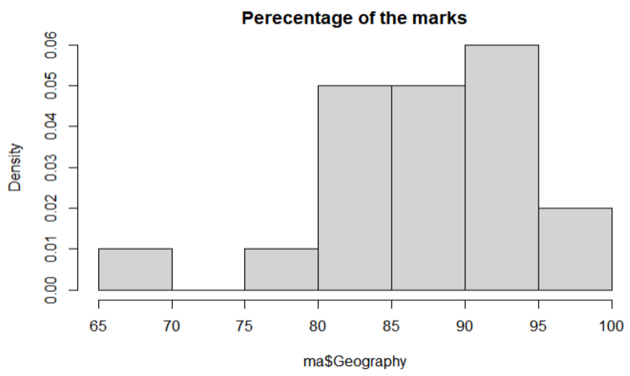
From the above result it is observed that p-value for all subject is more than the significant value i.e(1>0.05) the null hypothesis is accepted at 95%,65.55%,53.97%,67.51%,80.20%,83.29% level .

~~~
# Two sample t-test

```{r}

x<-sample(ma$Maths)

mean(x)


y<-sample(ma$Maths)

mean(y)


x1<-data.frame(x)

y1<-data.frame(y)

x1%>%

 ggplot(data=x1,mapping=aes(x))+geom_density(fill="green")+

  geom_vline(aes(xintercept=mean(x)),color="black",linetype="dashed",size=1)

t.test(x,y,var.equal = TRUE)


```

# From the above result it is observed that p-value is more than the significant value i.e(1>0.05) the null
hypothesis is accepted at 95% level. Hence there is no significant difference average of two samples.



# correlation test


```{r}

cor.test(ma$English,ma$Maths)

```


From the above result it is observed that p-value is greater than the significant value i.e(0.4492>0.05) the
null hypothesis is accepted at 5% level.The correlation value is 0.1793893.
~~~

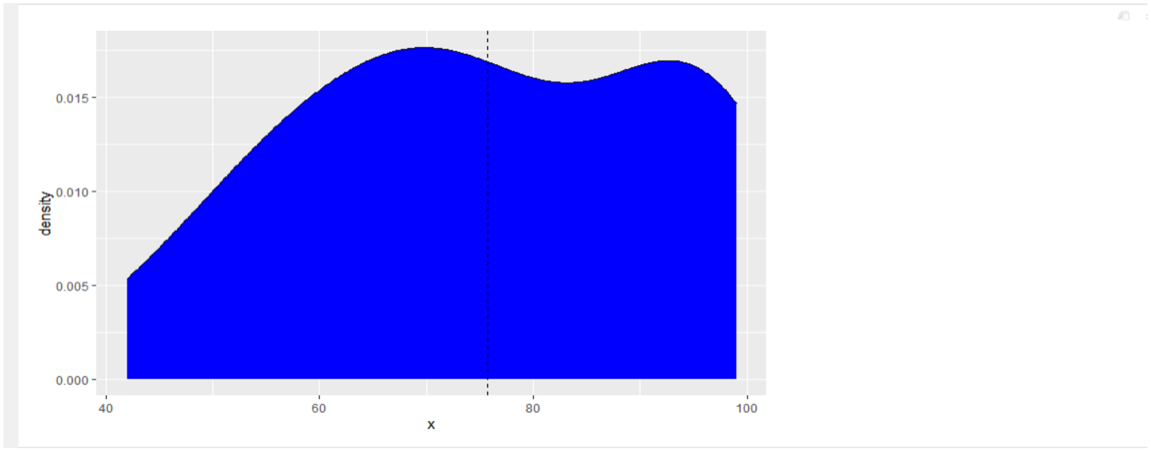
OUTPUT:

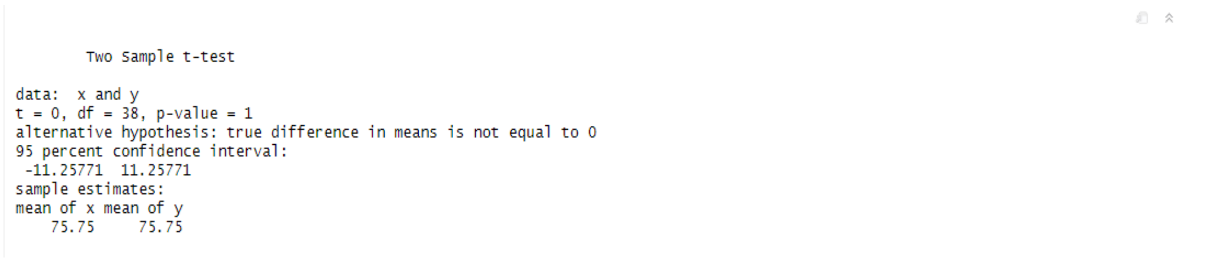


```
 Serial.number    Student.Name          English         Bengali          Hindi           Maths          History     
 Min.   : 1.00   Length:20          Min.   :23.00   Min.   :20.00   Min.   :33.00   Min.   :42.00   Min.   :65.00  
 1st Qu.: 5.75   Class :character   1st Qu.:55.25   1st Qu.:54.00   1st Qu.:43.50   1st Qu.:62.75   1st Qu.:78.50  
 Median :10.50   Mode  :character   Median :73.00   Median :70.50   Median :68.50   Median :74.50   Median :86.00  
 Mean   :10.50                      Mean   :70.10   Mean   :65.55   Mean   :63.85   Mean   :75.75   Mean   :84.60  
 3rd Qu.:15.25                      3rd Qu.:86.25   3rd Qu.:80.25   3rd Qu.:82.00   3rd Qu.:94.25   3rd Qu.:92.25  
 Max.   :20.00                      Max.   :96.00   Max.   :91.00   Max.   :96.00   Max.   :99.00   Max.   :97.00  
   Geography     
 Min.   : 65.00  
 1st Qu.: 82.75  
 Median : 88.00  
 Mean   : 87.15  
 3rd Qu.: 92.00  
 Max.   :100.00  
```

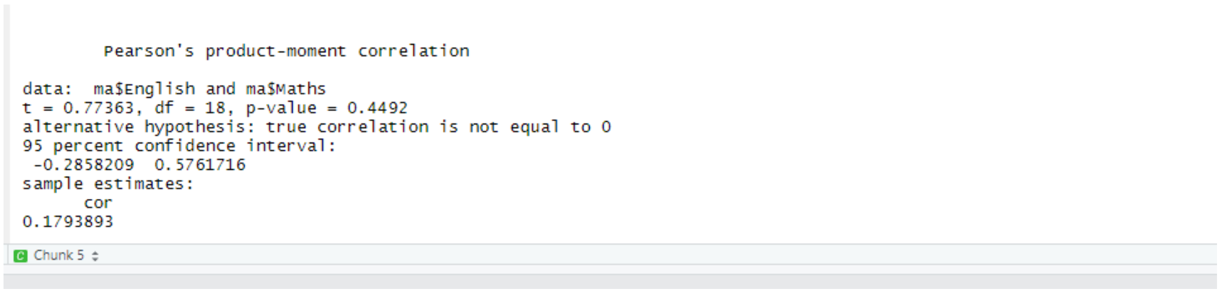


```
	One Sample t-test

data:  geo
t = 0, df = 19, p-value = 1
alternative hypothesis: true mean is not equal to 87.15
95 percent confidence interval:
 83.39889 90.90111
sample estimates:
mean of x 
    87.15 
```





```
        Two Sample t-test

data:  x and y
t = 0, df = 38, p-value = 1
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -11.25771  11.25771
sample estimates:
mean of x mean of y
    75.75     75.75
```



```
        Pearson's product-moment correlation

data:  ma$English and ma$Maths
t = 0.77363, df = 18, p-value = 0.4492
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.2858209  0.5761716
sample estimates:
      cor
0.1793893
```

INSIGHTS:

This dataset tells details about students name and the marks scored by them in each subjects

From this data set performing Statistical Analysis we find that

# Highest percentage of marks scored by students in English is in 80'S.

# Highest percentage of the marks scored by students in Bengali is in 75'S.

# Highest percentage of the marks scored by students in Hindhi is between 70 to 90.

# Highest percentage of the marks scored by students in Maths between 90 to 100

# Highest percentage of the marks scored by students in History is between 90 to 95

# Highest percentage of the marks scored by students in Geography is between 90 to 95

From the above one sample test result it is observed that p-value for all subject is more than the significant value i.e(1>0.05) the null hypothesis is accepted at 95%,65.55%,53.97%,67.51%,80.20%,83.29% level From the above two sample test result it is observed that p-value is more than the significant value i.e(1>0.05) the null hypothesis is accepted at 95% level. Hence there is no significant difference average of two samples. From the above correlation test result it is observed that p-value is greater than the significant value i.e(0.4492>0.05) the null hypothesis is accepted at 5% level. The correlation value is 0.1793893.