Exploratory analysis part -1

library(tidyverse)

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.4 ✔ readr 2.1.5  
✔ forcats 1.0.0 ✔ stringr 1.5.1  
✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
✔ purrr 1.0.2   
── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag() masks stats::lag()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(renv)

Attaching package: 'renv'  
  
The following object is masked from 'package:purrr':  
  
 modify  
  
The following objects are masked from 'package:stats':  
  
 embed, update  
  
The following objects are masked from 'package:utils':  
  
 history, upgrade  
  
The following objects are masked from 'package:base':  
  
 autoload, load, remove, use

library(devtools)

Loading required package: usethis  
  
Attaching package: 'devtools'  
  
The following object is masked from 'package:renv':  
  
 install

library(usethis)  
library(gitcreds)  
library(magrittr)

Attaching package: 'magrittr'  
  
The following object is masked from 'package:purrr':  
  
 set\_names  
  
The following object is masked from 'package:tidyr':  
  
 extract

library(httr) webAPI  
library(base)  
library(dplyr) data manipulation  
library(ggplot2) data graphics of visualization  
# library(pacman) load all data

# DPLYR package needed for doing below functions

column select , mutate row filter, distinct, arrange GROUP Group \_by summarise count

# Exploratory Analysis

Using some visual techniques to get statistical Summary of the data. Deals with missing values,outliers, trends or patterns of the data.

###Load the data and created subset of the data df

data(iris)  
  
df <- iris

### Dimension function returns the Total number of columns and rows of the data

dim(df)

[1] 150 5

# we got 150 rows and 5 coumns

# function length return the number of columns

length(iris) # returns number of columns in the data

[1] 5

ncol(iris) # returns number of columns in the data

[1] 5

nrow(iris) # returns number of rows in the data

[1] 150

# sample data viewed by head() and View() functions

head(iris) # returns 5 rows of the data as sample

Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
1 5.1 3.5 1.4 0.2 setosa  
2 4.9 3.0 1.4 0.2 setosa  
3 4.7 3.2 1.3 0.2 setosa  
4 4.6 3.1 1.5 0.2 setosa  
5 5.0 3.6 1.4 0.2 setosa  
6 5.4 3.9 1.7 0.4 setosa

View(iris) # returns the dataset

# the function summary() returns statistical summaries likely max,min,mean, median, outliers,standard dviations,parttern or trends

summary(iris)

Sepal.Length Sepal.Width Petal.Length Petal.Width   
 Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300   
 Median :5.800 Median :3.000 Median :4.350 Median :1.300   
 Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199   
 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800   
 Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500   
 Species   
 setosa :50   
 versicolor:50   
 virginica :50

### Find the columns /Variable names and its data types using structure function

str(df)

'data.frame': 150 obs. of 5 variables:  
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
 $ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

# it returns each five variable names and datatypes as numeric and Factor respectively.

# Alternate method for finding data types using by sapply function will return as vector.

sapply(df,class)

Sepal.Length Sepal.Width Petal.Length Petal.Width Species   
 "numeric" "numeric" "numeric" "numeric" "factor"

# the out put shows earch variables data types   
# numeric and factor respectively as vector Horizontal rows

Alternate method if using lapply function will return list type.

lapply(df,class)

$Sepal.Length  
[1] "numeric"  
  
$Sepal.Width  
[1] "numeric"  
  
$Petal.Length  
[1] "numeric"  
  
$Petal.Width  
[1] "numeric"  
  
$Species  
[1] "factor"

# the out put shows earch variables data types   
# numeric and factor respectively as list vertical rows

## Finding missing values NA of each variables if NA values present it will return TRUE otherwise FALSE

is.na(df)

Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
 [1,] FALSE FALSE FALSE FALSE FALSE  
 [2,] FALSE FALSE FALSE FALSE FALSE  
 [3,] FALSE FALSE FALSE FALSE FALSE  
 [4,] FALSE FALSE FALSE FALSE FALSE  
 [5,] FALSE FALSE FALSE FALSE FALSE  
 [6,] FALSE FALSE FALSE FALSE FALSE  
 [7,] FALSE FALSE FALSE FALSE FALSE  
 [8,] FALSE FALSE FALSE FALSE FALSE  
 [9,] FALSE FALSE FALSE FALSE FALSE  
 [10,] FALSE FALSE FALSE FALSE FALSE  
 [11,] FALSE FALSE FALSE FALSE FALSE  
 [12,] FALSE FALSE FALSE FALSE FALSE  
 [13,] FALSE FALSE FALSE FALSE FALSE  
 [14,] FALSE FALSE FALSE FALSE FALSE  
 [15,] FALSE FALSE FALSE FALSE FALSE  
 [16,] FALSE FALSE FALSE FALSE FALSE  
 [17,] FALSE FALSE FALSE FALSE FALSE  
 [18,] FALSE FALSE FALSE FALSE FALSE  
 [19,] FALSE FALSE FALSE FALSE FALSE  
 [20,] FALSE FALSE FALSE FALSE FALSE  
 [21,] FALSE FALSE FALSE FALSE FALSE  
 [22,] FALSE FALSE FALSE FALSE FALSE  
 [23,] FALSE FALSE FALSE FALSE FALSE  
 [24,] FALSE FALSE FALSE FALSE FALSE  
 [25,] FALSE FALSE FALSE FALSE FALSE  
 [26,] FALSE FALSE FALSE FALSE FALSE  
 [27,] FALSE FALSE FALSE FALSE FALSE  
 [28,] FALSE FALSE FALSE FALSE FALSE  
 [29,] FALSE FALSE FALSE FALSE FALSE  
 [30,] FALSE FALSE FALSE FALSE FALSE  
 [31,] FALSE FALSE FALSE FALSE FALSE  
 [32,] FALSE FALSE FALSE FALSE FALSE  
 [33,] FALSE FALSE FALSE FALSE FALSE  
 [34,] FALSE FALSE FALSE FALSE FALSE  
 [35,] FALSE FALSE FALSE FALSE FALSE  
 [36,] FALSE FALSE FALSE FALSE FALSE  
 [37,] FALSE FALSE FALSE FALSE FALSE  
 [38,] FALSE FALSE FALSE FALSE FALSE  
 [39,] FALSE FALSE FALSE FALSE FALSE  
 [40,] FALSE FALSE FALSE FALSE FALSE  
 [41,] FALSE FALSE FALSE FALSE FALSE  
 [42,] FALSE FALSE FALSE FALSE FALSE  
 [43,] FALSE FALSE FALSE FALSE FALSE  
 [44,] FALSE FALSE FALSE FALSE FALSE  
 [45,] FALSE FALSE FALSE FALSE FALSE  
 [46,] FALSE FALSE FALSE FALSE FALSE  
 [47,] FALSE FALSE FALSE FALSE FALSE  
 [48,] FALSE FALSE FALSE FALSE FALSE  
 [49,] FALSE FALSE FALSE FALSE FALSE  
 [50,] FALSE FALSE FALSE FALSE FALSE  
 [51,] FALSE FALSE FALSE FALSE FALSE  
 [52,] FALSE FALSE FALSE FALSE FALSE  
 [53,] FALSE FALSE FALSE FALSE FALSE  
 [54,] FALSE FALSE FALSE FALSE FALSE  
 [55,] FALSE FALSE FALSE FALSE FALSE  
 [56,] FALSE FALSE FALSE FALSE FALSE  
 [57,] FALSE FALSE FALSE FALSE FALSE  
 [58,] FALSE FALSE FALSE FALSE FALSE  
 [59,] FALSE FALSE FALSE FALSE FALSE  
 [60,] FALSE FALSE FALSE FALSE FALSE  
 [61,] FALSE FALSE FALSE FALSE FALSE  
 [62,] FALSE FALSE FALSE FALSE FALSE  
 [63,] FALSE FALSE FALSE FALSE FALSE  
 [64,] FALSE FALSE FALSE FALSE FALSE  
 [65,] FALSE FALSE FALSE FALSE FALSE  
 [66,] FALSE FALSE FALSE FALSE FALSE  
 [67,] FALSE FALSE FALSE FALSE FALSE  
 [68,] FALSE FALSE FALSE FALSE FALSE  
 [69,] FALSE FALSE FALSE FALSE FALSE  
 [70,] FALSE FALSE FALSE FALSE FALSE  
 [71,] FALSE FALSE FALSE FALSE FALSE  
 [72,] FALSE FALSE FALSE FALSE FALSE  
 [73,] FALSE FALSE FALSE FALSE FALSE  
 [74,] FALSE FALSE FALSE FALSE FALSE  
 [75,] FALSE FALSE FALSE FALSE FALSE  
 [76,] FALSE FALSE FALSE FALSE FALSE  
 [77,] FALSE FALSE FALSE FALSE FALSE  
 [78,] FALSE FALSE FALSE FALSE FALSE  
 [79,] FALSE FALSE FALSE FALSE FALSE  
 [80,] FALSE FALSE FALSE FALSE FALSE  
 [81,] FALSE FALSE FALSE FALSE FALSE  
 [82,] FALSE FALSE FALSE FALSE FALSE  
 [83,] FALSE FALSE FALSE FALSE FALSE  
 [84,] FALSE FALSE FALSE FALSE FALSE  
 [85,] FALSE FALSE FALSE FALSE FALSE  
 [86,] FALSE FALSE FALSE FALSE FALSE  
 [87,] FALSE FALSE FALSE FALSE FALSE  
 [88,] FALSE FALSE FALSE FALSE FALSE  
 [89,] FALSE FALSE FALSE FALSE FALSE  
 [90,] FALSE FALSE FALSE FALSE FALSE  
 [91,] FALSE FALSE FALSE FALSE FALSE  
 [92,] FALSE FALSE FALSE FALSE FALSE  
 [93,] FALSE FALSE FALSE FALSE FALSE  
 [94,] FALSE FALSE FALSE FALSE FALSE  
 [95,] FALSE FALSE FALSE FALSE FALSE  
 [96,] FALSE FALSE FALSE FALSE FALSE  
 [97,] FALSE FALSE FALSE FALSE FALSE  
 [98,] FALSE FALSE FALSE FALSE FALSE  
 [99,] FALSE FALSE FALSE FALSE FALSE  
[100,] FALSE FALSE FALSE FALSE FALSE  
[101,] FALSE FALSE FALSE FALSE FALSE  
[102,] FALSE FALSE FALSE FALSE FALSE  
[103,] FALSE FALSE FALSE FALSE FALSE  
[104,] FALSE FALSE FALSE FALSE FALSE  
[105,] FALSE FALSE FALSE FALSE FALSE  
[106,] FALSE FALSE FALSE FALSE FALSE  
[107,] FALSE FALSE FALSE FALSE FALSE  
[108,] FALSE FALSE FALSE FALSE FALSE  
[109,] FALSE FALSE FALSE FALSE FALSE  
[110,] FALSE FALSE FALSE FALSE FALSE  
[111,] FALSE FALSE FALSE FALSE FALSE  
[112,] FALSE FALSE FALSE FALSE FALSE  
[113,] FALSE FALSE FALSE FALSE FALSE  
[114,] FALSE FALSE FALSE FALSE FALSE  
[115,] FALSE FALSE FALSE FALSE FALSE  
[116,] FALSE FALSE FALSE FALSE FALSE  
[117,] FALSE FALSE FALSE FALSE FALSE  
[118,] FALSE FALSE FALSE FALSE FALSE  
[119,] FALSE FALSE FALSE FALSE FALSE  
[120,] FALSE FALSE FALSE FALSE FALSE  
[121,] FALSE FALSE FALSE FALSE FALSE  
[122,] FALSE FALSE FALSE FALSE FALSE  
[123,] FALSE FALSE FALSE FALSE FALSE  
[124,] FALSE FALSE FALSE FALSE FALSE  
[125,] FALSE FALSE FALSE FALSE FALSE  
[126,] FALSE FALSE FALSE FALSE FALSE  
[127,] FALSE FALSE FALSE FALSE FALSE  
[128,] FALSE FALSE FALSE FALSE FALSE  
[129,] FALSE FALSE FALSE FALSE FALSE  
[130,] FALSE FALSE FALSE FALSE FALSE  
[131,] FALSE FALSE FALSE FALSE FALSE  
[132,] FALSE FALSE FALSE FALSE FALSE  
[133,] FALSE FALSE FALSE FALSE FALSE  
[134,] FALSE FALSE FALSE FALSE FALSE  
[135,] FALSE FALSE FALSE FALSE FALSE  
[136,] FALSE FALSE FALSE FALSE FALSE  
[137,] FALSE FALSE FALSE FALSE FALSE  
[138,] FALSE FALSE FALSE FALSE FALSE  
[139,] FALSE FALSE FALSE FALSE FALSE  
[140,] FALSE FALSE FALSE FALSE FALSE  
[141,] FALSE FALSE FALSE FALSE FALSE  
[142,] FALSE FALSE FALSE FALSE FALSE  
[143,] FALSE FALSE FALSE FALSE FALSE  
[144,] FALSE FALSE FALSE FALSE FALSE  
[145,] FALSE FALSE FALSE FALSE FALSE  
[146,] FALSE FALSE FALSE FALSE FALSE  
[147,] FALSE FALSE FALSE FALSE FALSE  
[148,] FALSE FALSE FALSE FALSE FALSE  
[149,] FALSE FALSE FALSE FALSE FALSE  
[150,] FALSE FALSE FALSE FALSE FALSE

# finding the number of missing values

sum(is.na(df))

[1] 0

# Returns total number of missing values NA, here it is zero no NA value.

# unique function remove the duplicate values

unique(iris)

Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
1 5.1 3.5 1.4 0.2 setosa  
2 4.9 3.0 1.4 0.2 setosa  
3 4.7 3.2 1.3 0.2 setosa  
4 4.6 3.1 1.5 0.2 setosa  
5 5.0 3.6 1.4 0.2 setosa  
6 5.4 3.9 1.7 0.4 setosa  
7 4.6 3.4 1.4 0.3 setosa  
8 5.0 3.4 1.5 0.2 setosa  
9 4.4 2.9 1.4 0.2 setosa  
10 4.9 3.1 1.5 0.1 setosa  
11 5.4 3.7 1.5 0.2 setosa  
12 4.8 3.4 1.6 0.2 setosa  
13 4.8 3.0 1.4 0.1 setosa  
14 4.3 3.0 1.1 0.1 setosa  
15 5.8 4.0 1.2 0.2 setosa  
16 5.7 4.4 1.5 0.4 setosa  
17 5.4 3.9 1.3 0.4 setosa  
18 5.1 3.5 1.4 0.3 setosa  
19 5.7 3.8 1.7 0.3 setosa  
20 5.1 3.8 1.5 0.3 setosa  
21 5.4 3.4 1.7 0.2 setosa  
22 5.1 3.7 1.5 0.4 setosa  
23 4.6 3.6 1.0 0.2 setosa  
24 5.1 3.3 1.7 0.5 setosa  
25 4.8 3.4 1.9 0.2 setosa  
26 5.0 3.0 1.6 0.2 setosa  
27 5.0 3.4 1.6 0.4 setosa  
28 5.2 3.5 1.5 0.2 setosa  
29 5.2 3.4 1.4 0.2 setosa  
30 4.7 3.2 1.6 0.2 setosa  
31 4.8 3.1 1.6 0.2 setosa  
32 5.4 3.4 1.5 0.4 setosa  
33 5.2 4.1 1.5 0.1 setosa  
34 5.5 4.2 1.4 0.2 setosa  
35 4.9 3.1 1.5 0.2 setosa  
36 5.0 3.2 1.2 0.2 setosa  
37 5.5 3.5 1.3 0.2 setosa  
38 4.9 3.6 1.4 0.1 setosa  
39 4.4 3.0 1.3 0.2 setosa  
40 5.1 3.4 1.5 0.2 setosa  
41 5.0 3.5 1.3 0.3 setosa  
42 4.5 2.3 1.3 0.3 setosa  
43 4.4 3.2 1.3 0.2 setosa  
44 5.0 3.5 1.6 0.6 setosa  
45 5.1 3.8 1.9 0.4 setosa  
46 4.8 3.0 1.4 0.3 setosa  
47 5.1 3.8 1.6 0.2 setosa  
48 4.6 3.2 1.4 0.2 setosa  
49 5.3 3.7 1.5 0.2 setosa  
50 5.0 3.3 1.4 0.2 setosa  
51 7.0 3.2 4.7 1.4 versicolor  
52 6.4 3.2 4.5 1.5 versicolor  
53 6.9 3.1 4.9 1.5 versicolor  
54 5.5 2.3 4.0 1.3 versicolor  
55 6.5 2.8 4.6 1.5 versicolor  
56 5.7 2.8 4.5 1.3 versicolor  
57 6.3 3.3 4.7 1.6 versicolor  
58 4.9 2.4 3.3 1.0 versicolor  
59 6.6 2.9 4.6 1.3 versicolor  
60 5.2 2.7 3.9 1.4 versicolor  
61 5.0 2.0 3.5 1.0 versicolor  
62 5.9 3.0 4.2 1.5 versicolor  
63 6.0 2.2 4.0 1.0 versicolor  
64 6.1 2.9 4.7 1.4 versicolor  
65 5.6 2.9 3.6 1.3 versicolor  
66 6.7 3.1 4.4 1.4 versicolor  
67 5.6 3.0 4.5 1.5 versicolor  
68 5.8 2.7 4.1 1.0 versicolor  
69 6.2 2.2 4.5 1.5 versicolor  
70 5.6 2.5 3.9 1.1 versicolor  
71 5.9 3.2 4.8 1.8 versicolor  
72 6.1 2.8 4.0 1.3 versicolor  
73 6.3 2.5 4.9 1.5 versicolor  
74 6.1 2.8 4.7 1.2 versicolor  
75 6.4 2.9 4.3 1.3 versicolor  
76 6.6 3.0 4.4 1.4 versicolor  
77 6.8 2.8 4.8 1.4 versicolor  
78 6.7 3.0 5.0 1.7 versicolor  
79 6.0 2.9 4.5 1.5 versicolor  
80 5.7 2.6 3.5 1.0 versicolor  
81 5.5 2.4 3.8 1.1 versicolor  
82 5.5 2.4 3.7 1.0 versicolor  
83 5.8 2.7 3.9 1.2 versicolor  
84 6.0 2.7 5.1 1.6 versicolor  
85 5.4 3.0 4.5 1.5 versicolor  
86 6.0 3.4 4.5 1.6 versicolor  
87 6.7 3.1 4.7 1.5 versicolor  
88 6.3 2.3 4.4 1.3 versicolor  
89 5.6 3.0 4.1 1.3 versicolor  
90 5.5 2.5 4.0 1.3 versicolor  
91 5.5 2.6 4.4 1.2 versicolor  
92 6.1 3.0 4.6 1.4 versicolor  
93 5.8 2.6 4.0 1.2 versicolor  
94 5.0 2.3 3.3 1.0 versicolor  
95 5.6 2.7 4.2 1.3 versicolor  
96 5.7 3.0 4.2 1.2 versicolor  
97 5.7 2.9 4.2 1.3 versicolor  
98 6.2 2.9 4.3 1.3 versicolor  
99 5.1 2.5 3.0 1.1 versicolor  
100 5.7 2.8 4.1 1.3 versicolor  
101 6.3 3.3 6.0 2.5 virginica  
102 5.8 2.7 5.1 1.9 virginica  
103 7.1 3.0 5.9 2.1 virginica  
104 6.3 2.9 5.6 1.8 virginica  
105 6.5 3.0 5.8 2.2 virginica  
106 7.6 3.0 6.6 2.1 virginica  
107 4.9 2.5 4.5 1.7 virginica  
108 7.3 2.9 6.3 1.8 virginica  
109 6.7 2.5 5.8 1.8 virginica  
110 7.2 3.6 6.1 2.5 virginica  
111 6.5 3.2 5.1 2.0 virginica  
112 6.4 2.7 5.3 1.9 virginica  
113 6.8 3.0 5.5 2.1 virginica  
114 5.7 2.5 5.0 2.0 virginica  
115 5.8 2.8 5.1 2.4 virginica  
116 6.4 3.2 5.3 2.3 virginica  
117 6.5 3.0 5.5 1.8 virginica  
118 7.7 3.8 6.7 2.2 virginica  
119 7.7 2.6 6.9 2.3 virginica  
120 6.0 2.2 5.0 1.5 virginica  
121 6.9 3.2 5.7 2.3 virginica  
122 5.6 2.8 4.9 2.0 virginica  
123 7.7 2.8 6.7 2.0 virginica  
124 6.3 2.7 4.9 1.8 virginica  
125 6.7 3.3 5.7 2.1 virginica  
126 7.2 3.2 6.0 1.8 virginica  
127 6.2 2.8 4.8 1.8 virginica  
128 6.1 3.0 4.9 1.8 virginica  
129 6.4 2.8 5.6 2.1 virginica  
130 7.2 3.0 5.8 1.6 virginica  
131 7.4 2.8 6.1 1.9 virginica  
132 7.9 3.8 6.4 2.0 virginica  
133 6.4 2.8 5.6 2.2 virginica  
134 6.3 2.8 5.1 1.5 virginica  
135 6.1 2.6 5.6 1.4 virginica  
136 7.7 3.0 6.1 2.3 virginica  
137 6.3 3.4 5.6 2.4 virginica  
138 6.4 3.1 5.5 1.8 virginica  
139 6.0 3.0 4.8 1.8 virginica  
140 6.9 3.1 5.4 2.1 virginica  
141 6.7 3.1 5.6 2.4 virginica  
142 6.9 3.1 5.1 2.3 virginica  
144 6.8 3.2 5.9 2.3 virginica  
145 6.7 3.3 5.7 2.5 virginica  
146 6.7 3.0 5.2 2.3 virginica  
147 6.3 2.5 5.0 1.9 virginica  
148 6.5 3.0 5.2 2.0 virginica  
149 6.2 3.4 5.4 2.3 virginica  
150 5.9 3.0 5.1 1.8 virginica

# lets, create vector x using function c()

x<- c(1,2,NA,4,5,NA,NA,8)  
print(x)

[1] 1 2 NA 4 5 NA NA 8

## Finding missing value of each variables

#if NA values present it will regturn TRUE otherwise FALSE

is.na(x)

[1] FALSE FALSE TRUE FALSE FALSE TRUE TRUE FALSE

# the output returns FALSE - no NA value to that variable.   
 # TRUE - yes the variable contains NA vaLues

# finding the number of missing values NA

sum(is.na(x))

[1] 3

# using which() functions we can find the index positions of NA values

which(is.na(x))

[1] 3 6 7

# returns 3rd,6th index positions has NA values

length(iris) # returns number of columns in the data

[1] 5

ncol(iris) # returns number of columns in the data

[1] 5

nrow(iris) # returns number of rows in the data

[1] 150