Lab 5A

Nehal Ur Rahman

2023-03-16

NAME: NEHAL UR RAHMNAN STUDENT ID: 991691259

Introduction: In lab 5A we are going to analyze Pokemon dataset and perform the given instructions to answer the questions. This dataset is about the different types of pokemon and its strengths in terms of HP, Attack, Defense, Special attack, Special Defense and Speed.

#We are loading the below library to perform various functions.

library(sets)  
library(ggplot2)  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ tibble 3.1.8 ✔ dplyr 1.1.0  
## ✔ tidyr 1.3.0 ✔ stringr 1.5.0  
## ✔ readr 2.1.3 ✔ forcats 1.0.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ forcats::%>%() masks stringr::%>%(), dplyr::%>%(), purrr::%>%(), tidyr::%>%(), tibble::%>%(), sets::%>%()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(ISLR)  
library(moments)  
library(dplyr)   
library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2  
##   
## Attaching package: 'GGally'  
##   
## The following object is masked from 'package:sets':  
##   
## %>%

The Pokemon data is loaded by setting the session directory.

poke <- read.csv("Pokemon.csv")  
head(poke)

## X. Name Type.1 Type.2 Total HP Attack Defense Sp..Atk  
## 1 1 Bulbasaur Grass Poison 318 45 49 49 65  
## 2 2 Ivysaur Grass Poison 405 60 62 63 80  
## 3 3 Venusaur Grass Poison 525 80 82 83 100  
## 4 3 VenusaurMega Venusaur Grass Poison 625 80 100 123 122  
## 5 4 Charmander Fire 309 39 52 43 60  
## 6 5 Charmeleon Fire 405 58 64 58 80  
## Sp..Def Speed Generation Legendary  
## 1 65 45 1 FALSE  
## 2 80 60 1 FALSE  
## 3 100 80 1 FALSE  
## 4 120 80 1 FALSE  
## 5 50 65 1 FALSE  
## 6 65 80 1 FALSE

We are using the cat function to print the column names of the pokemon dataset with the help of colnames function.

cat("Column Names : ", colnames(poke))

## Column Names : X. Name Type.1 Type.2 Total HP Attack Defense Sp..Atk Sp..Def Speed Generation Legendary

Using Summary function`we display the statistical information about the numerical variables in the dataset.

cat("\nSummary of data: \n \n")

##   
## Summary of data:   
##

summary(poke)

## X. Name Type.1 Type.2   
## Min. : 1.0 Length:800 Length:800 Length:800   
## 1st Qu.:184.8 Class :character Class :character Class :character   
## Median :364.5 Mode :character Mode :character Mode :character   
## Mean :362.8   
## 3rd Qu.:539.2   
## Max. :721.0   
## Total HP Attack Defense   
## Min. :180.0 Min. : 1.00 Min. : 5 Min. : 5.00   
## 1st Qu.:330.0 1st Qu.: 50.00 1st Qu.: 55 1st Qu.: 50.00   
## Median :450.0 Median : 65.00 Median : 75 Median : 70.00   
## Mean :435.1 Mean : 69.26 Mean : 79 Mean : 73.84   
## 3rd Qu.:515.0 3rd Qu.: 80.00 3rd Qu.:100 3rd Qu.: 90.00   
## Max. :780.0 Max. :255.00 Max. :190 Max. :230.00   
## Sp..Atk Sp..Def Speed Generation   
## Min. : 10.00 Min. : 20.0 Min. : 5.00 Min. :1.000   
## 1st Qu.: 49.75 1st Qu.: 50.0 1st Qu.: 45.00 1st Qu.:2.000   
## Median : 65.00 Median : 70.0 Median : 65.00 Median :3.000   
## Mean : 72.82 Mean : 71.9 Mean : 68.28 Mean :3.324   
## 3rd Qu.: 95.00 3rd Qu.: 90.0 3rd Qu.: 90.00 3rd Qu.:5.000   
## Max. :194.00 Max. :230.0 Max. :180.00 Max. :6.000   
## Legendary   
## Mode :logical   
## FALSE:735   
## TRUE :65   
##   
##   
##

I have checked if there are any NA values present in the dataset to see if cleaning is required.

colSums(is.na(poke))

## X. Name Type.1 Type.2 Total HP Attack   
## 0 0 0 0 0 0 0   
## Defense Sp..Atk Sp..Def Speed Generation Legendary   
## 0 0 0 0 0 0

After going through the dataset I have noticed that there are some blank spaces in Type.2 variable. I have decided not to clean this variable as only some pokemon have its type 2 and thus it doesn’t need to be cleaned. Cleaning it will lead to loss of a large chunk of data unnecessarily.

#Deleting any duplicates if present.

poke <- poke[!duplicated(poke), ]

str(poke)

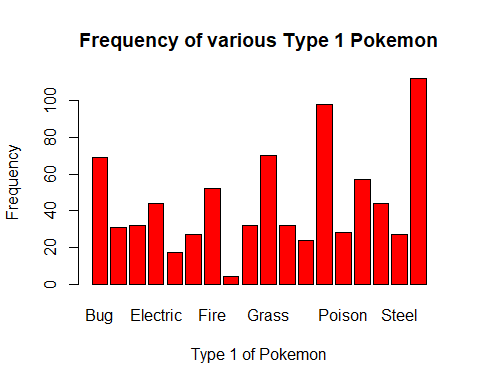
## 'data.frame': 800 obs. of 13 variables:  
## $ X. : int 1 2 3 3 4 5 6 6 6 7 ...  
## $ Name : chr "Bulbasaur" "Ivysaur" "Venusaur" "VenusaurMega Venusaur" ...  
## $ Type.1 : chr "Grass" "Grass" "Grass" "Grass" ...  
## $ Type.2 : chr "Poison" "Poison" "Poison" "Poison" ...  
## $ Total : int 318 405 525 625 309 405 534 634 634 314 ...  
## $ HP : int 45 60 80 80 39 58 78 78 78 44 ...  
## $ Attack : int 49 62 82 100 52 64 84 130 104 48 ...  
## $ Defense : int 49 63 83 123 43 58 78 111 78 65 ...  
## $ Sp..Atk : int 65 80 100 122 60 80 109 130 159 50 ...  
## $ Sp..Def : int 65 80 100 120 50 65 85 85 115 64 ...  
## $ Speed : int 45 60 80 80 65 80 100 100 100 43 ...  
## $ Generation: int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Legendary : logi FALSE FALSE FALSE FALSE FALSE FALSE ...

The Pokemon dataset consists of 800 observations and 13 variables. 3 of the variables are of character type, 9 are of integer type and 1 of logical type.

##Question 1

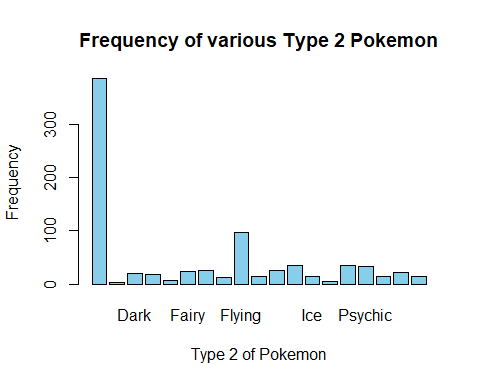
Exploration

#PLOT 1  
barplot(table(poke$Type.1), main="Frequency of various Type 1 Pokemon", col="red" , xlab="Type 1 of Pokemon", ylab = "Frequency")



From the above barplot we can see the frequency of Type 1 pokemon in the dataset. The graph got is not a normal distribution graph and has many peaks.

#PLOT 2  
barplot(table(poke$Type.2), main="Frequency of various Type 2 Pokemon", col="skyblue" , xlab="Type 2 of Pokemon", ylab = "Frequency")

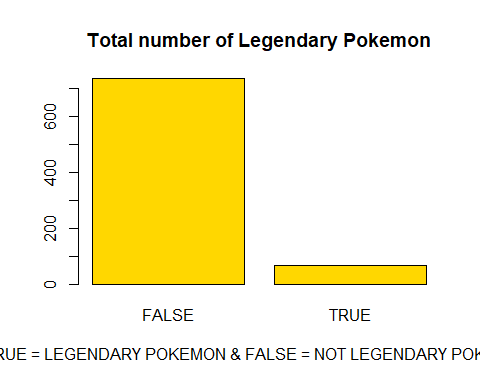


From the above barplot we can see the frequency of Type 2 pokemon in the dataset. The graph got is skewed towards the right and is uni-modal. The plot is unimodal as it has only one peak

#PLOT 3  
as.factor(poke$Legendary)

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [157] TRUE TRUE TRUE FALSE FALSE FALSE TRUE TRUE TRUE FALSE FALSE FALSE  
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [253] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE   
## [265] TRUE FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE  
## [277] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [289] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [301] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [313] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [325] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [337] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [349] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [361] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [373] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [385] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [397] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [409] FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE   
## [421] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE   
## [433] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [445] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [457] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [469] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [481] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [493] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [505] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [517] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [529] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE   
## [541] TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE TRUE TRUE TRUE   
## [553] TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [565] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [577] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [589] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [601] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [613] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [625] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [637] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [649] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [661] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [673] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [685] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [697] FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE   
## [709] TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [721] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [733] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [745] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [757] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [769] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [781] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [793] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE   
## Levels: FALSE TRUE

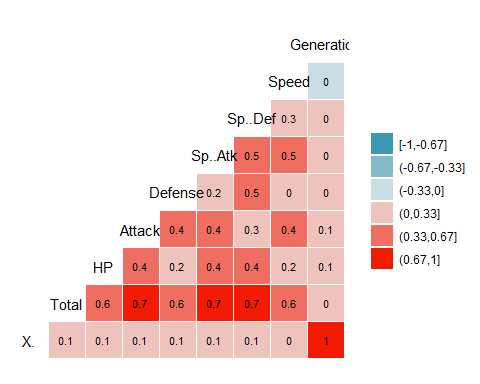
barplot(table(poke$Legendary), main="Total number of Legendary Pokemon", col="gold" , xlab="TRUE = LEGENDARY POKEMON & FALSE = NOT LEGENDARY POKEMON")



The plot shown above gives us information of how many legendary pokemons are there in the dataset. Therefore, most of the pokemons are not legendary in the given dataset and a very small number of pokemons are legendary.

#PLOT 4  
ggcorr(poke, nbreaks = 6, label = TRUE, label\_size = 3)

## Warning in ggcorr(poke, nbreaks = 6, label = TRUE, label\_size = 3): data in  
## column(s) 'Name', 'Type.1', 'Type.2', 'Legendary' are not numeric and were  
## ignored



With the help correlation matrix given above, we can see how different numerical variables are strongly or weakly related to each other. From the correlation matrix we get to know that the TOTAL variable has strong correlation with HP, Attack, Defense, Special attack, Special Defense and Speed. The variable X and Generation are strongly correlated as well.

##Question 2

#Step 1 First we create another variable called HP\_category which replicates the values of HP. The HP\_category is the stored in the poke dataset along with other variables. So now we have 14 variables.

HP\_category <- rep(poke$HP ,)  
poke <- data.frame(poke , HP\_category)

str(poke)

## 'data.frame': 800 obs. of 14 variables:  
## $ X. : int 1 2 3 3 4 5 6 6 6 7 ...  
## $ Name : chr "Bulbasaur" "Ivysaur" "Venusaur" "VenusaurMega Venusaur" ...  
## $ Type.1 : chr "Grass" "Grass" "Grass" "Grass" ...  
## $ Type.2 : chr "Poison" "Poison" "Poison" "Poison" ...  
## $ Total : int 318 405 525 625 309 405 534 634 634 314 ...  
## $ HP : int 45 60 80 80 39 58 78 78 78 44 ...  
## $ Attack : int 49 62 82 100 52 64 84 130 104 48 ...  
## $ Defense : int 49 63 83 123 43 58 78 111 78 65 ...  
## $ Sp..Atk : int 65 80 100 122 60 80 109 130 159 50 ...  
## $ Sp..Def : int 65 80 100 120 50 65 85 85 115 64 ...  
## $ Speed : int 45 60 80 80 65 80 100 100 100 43 ...  
## $ Generation : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Legendary : logi FALSE FALSE FALSE FALSE FALSE FALSE ...  
## $ HP\_category: int 45 60 80 80 39 58 78 78 78 44 ...

#Step 2 Now, we have categorized the Pokemon based on its HP. Therefore, 6 levels of HP\_category have been created for the given sets of HP using ifelse condition.

poke$HP\_category<- as.factor(ifelse(poke$HP\_category <= 49, 'Level 1: (-∞, 49]',   
 ifelse(poke$HP\_category >= 50 & poke$HP\_category <= 99,'Level 2: [50, 99]',  
 ifelse(poke$HP\_category >= 100 & poke$HP\_category <= 149,'Level 3: [100, 149]',  
 ifelse(poke$HP\_category >= 150 & poke$HP\_category <= 199, 'Level 4: [150, 199]',  
 ifelse(poke$HP\_category >= 200 & poke$HP\_category <= 249, 'Level 5: [200, 249]',  
'Level 6: [250, +∞)'))))))

##Question 3

Using tapply function the average Attack strength for each HP\_category is found.

tapply(poke$Attack, poke$HP\_category, FUN = mean)

## Level 1: (-∞, 49] Level 2: [50, 99] Level 3: [100, 149] Level 4: [150, 199]   
## 50.05405 82.62749 104.68539 96.00000   
## Level 6: [250, +∞)   
## 7.50000

From the above output we camn infer that the mean attack strength is the highest for Level 3 Pokemon and the least for level 6 pokemon

##Question 4

#Step 1 First we see the total number of legendary pokemon in the dataset(TRUE = LEGENDARY POKEMON & FALSE = NOT LEGENDARY POKEMON)

table(poke$Legendary)

##   
## FALSE TRUE   
## 735 65

There are total of 65 legendary pokemon and 735 non-legendary pokemon in the dataset.

#Step 2 Now we check if Mega is present in the name of the Pokemon using grepl function and store its value in the poke dataset.

mega\_poke<-grepl("Mega",poke$Name)  
poke<-data.frame(poke,mega\_poke)  
table(mega\_poke)

## mega\_poke  
## FALSE TRUE   
## 751 49

The grepl function returns a logical value(TRUE or FALSE) depending on the word pattern given. Therefore, there are a total of 49 Mega-Evolved Pokémon and 751 non-Mega-Evolved Pokémon.

#Step 3 Now we make another dataset as the subset of poke in which there are only legendary pokemons.

legendary\_poke <- subset(poke,poke$Legendary=='TRUE')  
str(legendary\_poke)

## 'data.frame': 65 obs. of 15 variables:  
## $ X. : int 144 145 146 150 150 150 243 244 245 249 ...  
## $ Name : chr "Articuno" "Zapdos" "Moltres" "Mewtwo" ...  
## $ Type.1 : chr "Ice" "Electric" "Fire" "Psychic" ...  
## $ Type.2 : chr "Flying" "Flying" "Flying" "" ...  
## $ Total : int 580 580 580 680 780 780 580 580 580 680 ...  
## $ HP : int 90 90 90 106 106 106 90 115 100 106 ...  
## $ Attack : int 85 90 100 110 190 150 85 115 75 90 ...  
## $ Defense : int 100 85 90 90 100 70 75 85 115 130 ...  
## $ Sp..Atk : int 95 125 125 154 154 194 115 90 90 90 ...  
## $ Sp..Def : int 125 90 85 90 100 120 100 75 115 154 ...  
## $ Speed : int 85 100 90 130 130 140 115 100 85 110 ...  
## $ Generation : int 1 1 1 1 1 1 2 2 2 2 ...  
## $ Legendary : logi TRUE TRUE TRUE TRUE TRUE TRUE ...  
## $ HP\_category: Factor w/ 5 levels "Level 1: (-∞, 49]",..: 2 2 2 3 3 3 2 3 3 3 ...  
## $ mega\_poke : logi FALSE FALSE FALSE FALSE TRUE TRUE ...

Now we have 15 variables in the dataset after adding the variable mega\_poke.

#Step 4 We substitute the True with Mega Poke and False with Non Mega to get a better idea of the proportion.

legendary\_poke$mega\_poke<-gsub("TRUE","Mega Poke",legendary\_poke$mega\_poke)  
legendary\_poke$mega\_poke<-gsub("FALSE","NON-Mega Poke",legendary\_poke$mega\_poke)

Using the proportion table we are able to find out proportion of the Mega-Evolved Pokémon and non-Mega-Evolved Pokémon that have a Legendary value(Legendary = TRUE).

prop.table(table(legendary\_poke$mega\_poke))

##   
## Mega Poke NON-Mega Poke   
## 0.09230769 0.90769231

With the output above, we understand that the proportion of Mega-Evolved Pokémon and non-Mega-Evolved Pokémon are: Mega-Evolved Pokémon = 9.23% Non-Mega-Evolved Pokémon = 90.76%