Pokemon

Aman

20/02/2021

```
getwd()
```

```
## [1] "D:/IISER BHOPAL/SEM 8/DSE 401/Revision"
```

Exploring Pokemon dataset:

Finding more details about our dataset:

- Correlations
- Normality tests
- some intersting patterns

Finding patterns:

I'll be dealing with following Hypothesises

- *Hypothesis1:* legenaday pokemons are better ?
- *Hypothesis2:* normality check of height , defence and attack
- *Hypothesis3:* hypothesis small pokemon has greater speed ?
- *Hypothesis4:* Do Pokemons improve with generations?
- *Hypothesis5:* Is Bigger the better, always?

Loading the libraries

```
library(tidyverse)
library(dplyr)
library(tidyr)
library(Hmisc)
library(ggplot2)
```

Goal: Using R explore the pokemon dataset

Importing dataset

```
pokemon <- read.csv("pokemon.csv", header = T)
```

```
dim(pokemon)
```

```
## [1] 801  41
```

This pokemon dataset has 801 rows and 41 cols

```
head(pokemon)
```

```
##                      abilities against_bug against_dark against_dragon
## 1 ['Overgrow', 'Chlorophyll']        1.00            1              1
## 2 ['Overgrow', 'Chlorophyll']        1.00            1              1
## 3 ['Overgrow', 'Chlorophyll']        1.00            1              1
## 4    ['Blaze', 'Solar Power']        0.50            1              1
## 5    ['Blaze', 'Solar Power']        0.50            1              1
## 6    ['Blaze', 'Solar Power']        0.25            1              1
##   against_electric against_fairy against_fight against_fire against_flying
## 1              0.5           0.5           0.5          2.0              2
## 2              0.5           0.5           0.5          2.0              2
## 3              0.5           0.5           0.5          2.0              2
## 4              1.0           0.5           1.0          0.5              1
## 5              1.0           0.5           1.0          0.5              1
## 6              2.0           0.5           0.5          0.5              1
##   against_ghost against_grass against_ground against_ice against_normal
## 1             1          0.25              1         2.0              1
## 2             1          0.25              1         2.0              1
## 3             1          0.25              1         2.0              1
## 4             1          0.50              2         0.5              1
## 5             1          0.50              2         0.5              1
## 6             1          0.25              0         1.0              1
##   against_poison against_psychic against_rock against_steel against_water
## 1              1               2            1           1.0           0.5
## 2              1               2            1           1.0           0.5
## 3              1               2            1           1.0           0.5
## 4              1               1            2           0.5           2.0
## 5              1               1            2           0.5           2.0
## 6              1               1            4           0.5           2.0
##   attack base_egg_steps base_happiness base_total capture_rate   classfication
## 1     49           5120             70        318           45   Seed PokÃ©mon
## 2     62           5120             70        405           45   Seed PokÃ©mon
## 3    100           5120             70        625           45   Seed PokÃ©mon
## 4     52           5120             70        309           45 Lizard PokÃ©mon
## 5     64           5120             70        405           45  Flame PokÃ©mon
## 6    104           5120             70        634           45  Flame PokÃ©mon
##   defense experience_growth height_m hp              japanese_name       name
## 1      49           1059860      0.7 45 Fushigidaneãƒ•ã‚·ã‚®ãƒ\200ãƒ\215  Bulbasaur
## 2      63           1059860      1.0 60  Fushigisouãƒ•ã‚·ã‚®ã‚½ã‚¦    Ivysaur
## 3     123           1059860      2.0 80 Fushigibanaãƒ•ã‚·ã‚®ãƒ\220ãƒŠ   Venusaur
## 4      43           1059860      0.6 39       Hitokageãƒ’ãƒ\210ã‚«ã‚² Charmander
## 5      58           1059860      1.1 58        Lizardoãƒªã‚¶ãƒ¼ãƒ‰ Charmeleon
## 6      78           1059860      1.7 78    Lizardonãƒªã‚¶ãƒ¼ãƒ‰ãƒ³  Charizard
##   percentage_male pokedex_number sp_attack sp_defense speed type1  type2
## 1            88.1              1        65         65    45 grass poison
## 2            88.1              2        80         80    60 grass poison
## 3            88.1              3       122        120    80 grass poison
## 4            88.1              4        60         50    65  fire
## 5            88.1              5        80         65    80  fire
## 6            88.1              6       159        115   100  fire flying
##   weight_kg generation is_legendary
## 1       6.9          1            0
## 2      13.0          1            0
## 3     100.0          1            0
```

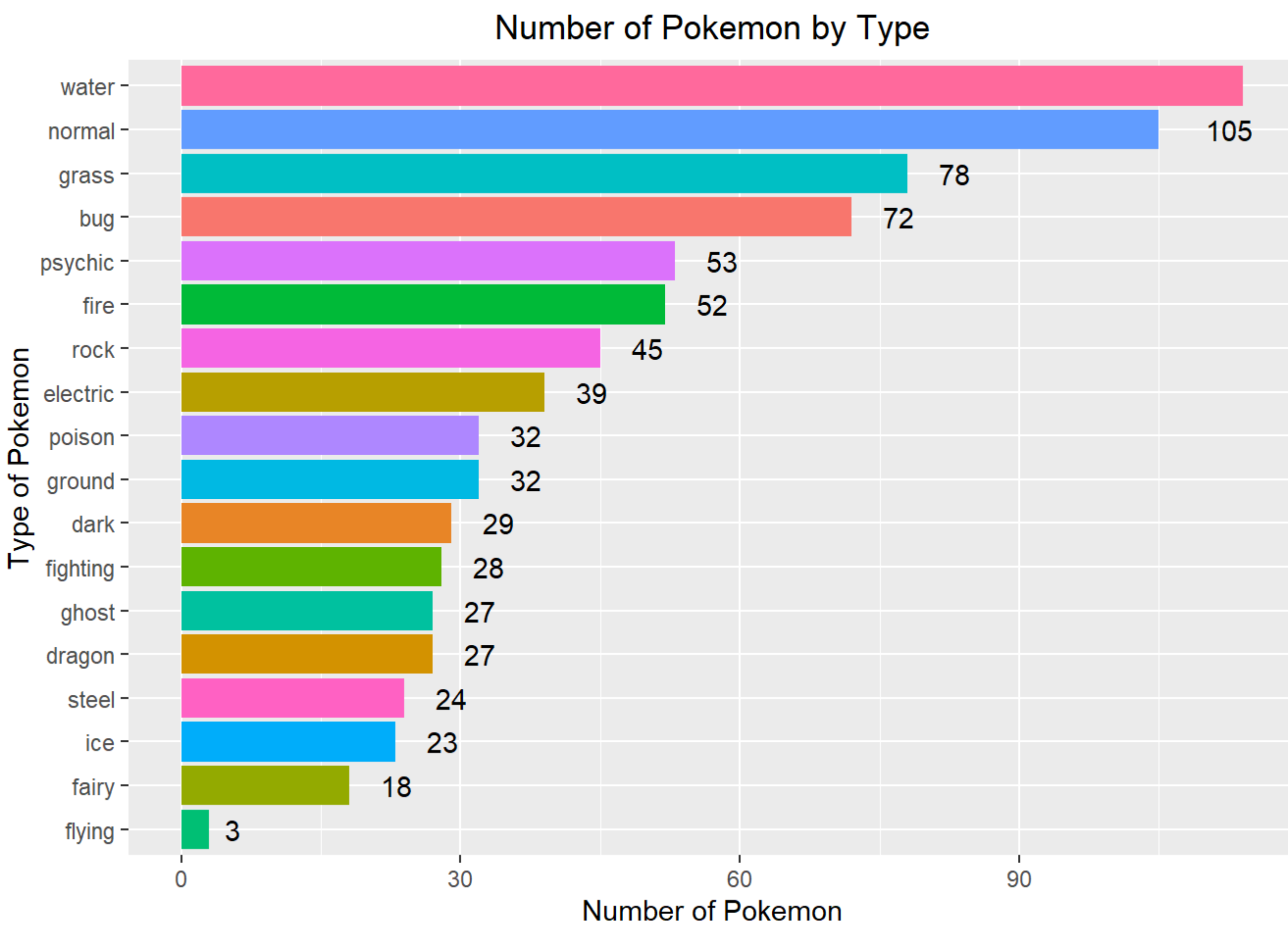
```
## 4       8.5          1           0
## 5      19.0          1           0
## 6      90.5          1           0
```

```
str(pokemon)
```

```
## 'data.frame':    801 obs. of  41 variables:
##  $ abilities        : chr  "['Overgrow', 'Chlorophyll']" "['Overgrow', 'Chlorophyll']" "['Ove
rgrow', 'Chlorophyll']" "['Blaze', 'Solar Power']" ...
##  $ against_bug      : num  1 1 1 0.5 0.5 0.25 1 1 1 1 ...
##  $ against_dark     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ against_dragon   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ against_electric : num  0.5 0.5 0.5 1 1 2 2 2 2 1 ...
##  $ against_fairy    : num  0.5 0.5 0.5 0.5 0.5 0.5 1 1 1 1 ...
##  $ against_fight    : num  0.5 0.5 0.5 1 1 0.5 1 1 1 0.5 ...
##  $ against_fire     : num  2 2 2 0.5 0.5 0.5 0.5 0.5 0.5 2 ...
##  $ against_flying   : num  2 2 2 1 1 1 1 1 1 2 ...
##  $ against_ghost    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ against_grass    : num  0.25 0.25 0.25 0.5 0.5 0.25 2 2 2 0.5 ...
##  $ against_ground   : num  1 1 1 2 2 0 1 1 1 0.5 ...
##  $ against_ice      : num  2 2 2 0.5 0.5 1 0.5 0.5 0.5 1 ...
##  $ against_normal   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ against_poison   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ against_psychic  : num  2 2 2 1 1 1 1 1 1 1 ...
##  $ against_rock     : num  1 1 1 2 2 4 1 1 1 2 ...
##  $ against_steel    : num  1 1 1 0.5 0.5 0.5 0.5 0.5 0.5 1 ...
##  $ against_water    : num  0.5 0.5 0.5 2 2 2 0.5 0.5 0.5 1 ...
##  $ attack           : int  49 62 100 52 64 104 48 63 103 30 ...
##  $ base_egg_steps   : int  5120 5120 5120 5120 5120 5120 5120 5120 5120 3840 ...
##  $ base_happiness   : int  70 70 70 70 70 70 70 70 70 70 ...
##  $ base_total       : int  318 405 625 309 405 634 314 405 630 195 ...
##  $ capture_rate     : chr  "45" "45" "45" "45" ...
##  $ classfication    : chr  "Seed PokÃ©mon" "Seed PokÃ©mon" "Seed PokÃ©mon" "Lizard PokÃ©mon"
...
##  $ defense          : int  49 63 123 43 58 78 65 80 120 35 ...
##  $ experience_growth: int  1059860 1059860 1059860 1059860 1059860 1059860 1059860 1059860 10
59860 1000000 ...
##  $ height_m         : num  0.7 1 2 0.6 1.1 1.7 0.5 1 1.6 0.3 ...
##  $ hp               : int  45 60 80 39 58 78 44 59 79 45 ...
##  $ japanese_name    : chr  "Fushigidaneãƒ•ã,·ã,®ãƒ\200ãƒ\215" "Fushigisouãƒ•ã,·ã,®ã,½ã,¦" "Fu
shigibanaãƒ•ã,·ã,®ãƒ\220ãƒŠ" "Hitokageãƒ'ãƒ\210ã,«ã,²" ...
##  $ name             : chr  "Bulbasaur" "Ivysaur" "Venusaur" "Charmander" ...
##  $ percentage_male  : num  88.1 88.1 88.1 88.1 88.1 88.1 88.1 88.1 88.1 50 ...
##  $ pokedex_number   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sp_attack        : int  65 80 122 60 80 159 50 65 135 20 ...
##  $ sp_defense       : int  65 80 120 50 65 115 64 80 115 20 ...
##  $ speed            : int  45 60 80 65 80 100 43 58 78 45 ...
##  $ type1            : chr  "grass" "grass" "grass" "fire" ...
##  $ type2            : chr  "poison" "poison" "poison" "" ...
##  $ weight_kg        : num  6.9 13 100 8.5 19 90.5 9 22.5 85.5 2.9 ...
##  $ generation       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ is_legendary     : int  0 0 0 0 0 0 0 0 0 0 ...
```

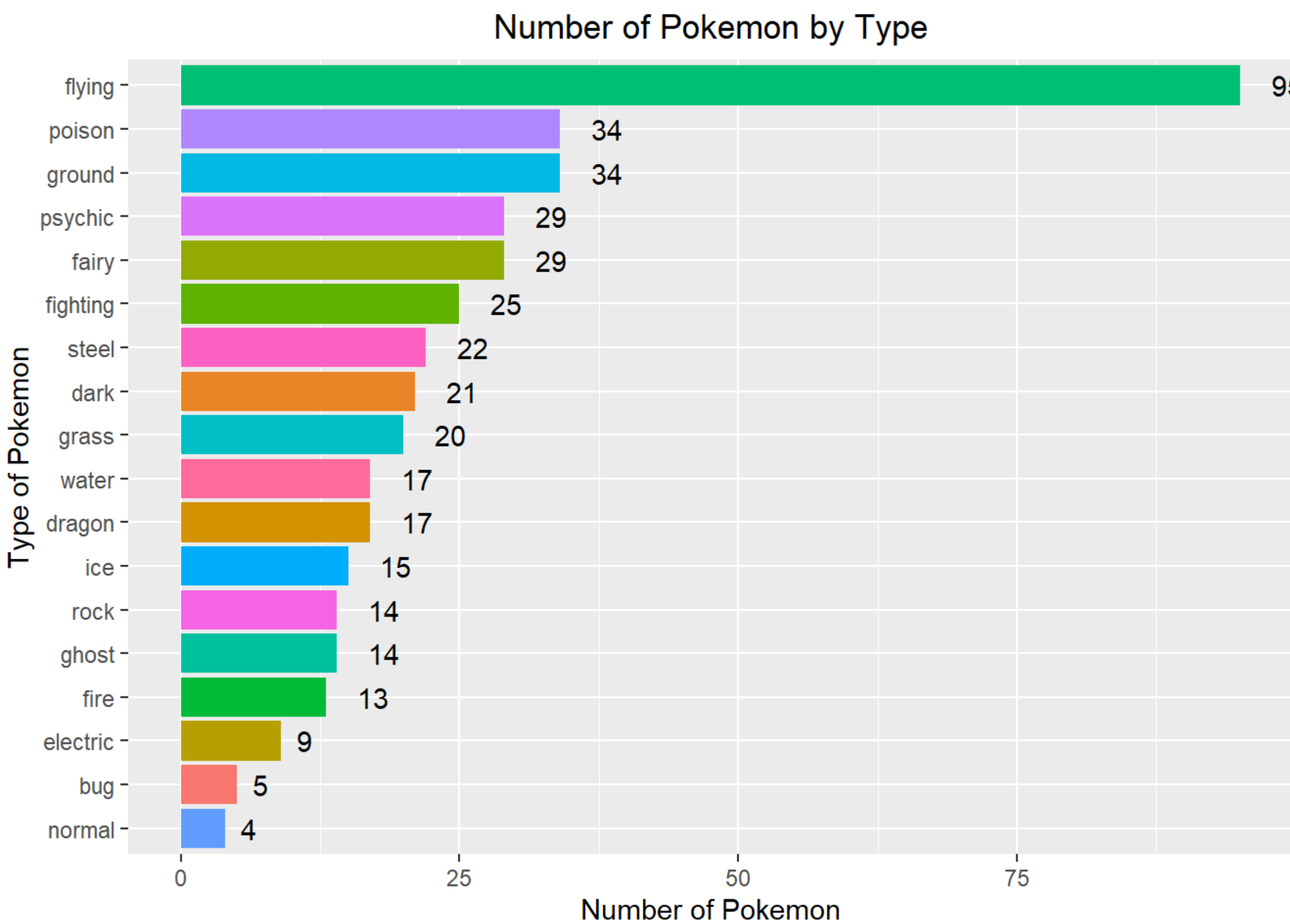
distribution of pokemon based on type 1

```
pokemon %>%
  group_by(type1) %>%
  summarise(number = n()) %>%
  ggplot(aes(x = reorder(type1, number), y = number , fill = type1)) +
  geom_bar(stat = 'identity') +
  xlab(label = "Type of Pokemon") +
  ylab(label = "Number of Pokemon") +
  ggtitle(label = "Number of Pokemon by Type") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.position="none") +
  coord_flip() +
  geom_text(aes(label = number), hjust = -1.0)
```



distribution of pokemon based on type 2

```
pokemon %>%
  filter(type2 != '') %>%
  group_by(type2) %>%
  summarise(number = n()) %>%
  ggplot(aes(x = reorder(type2, number), y = number , fill = type2)) +
  geom_bar(stat = 'identity') +
  xlab(label = "Type of Pokemon") +
  ylab(label = "Number of Pokemon") +
  ggtitle(label = "Number of Pokemon by Type") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.position="none") +
  coord_flip() +
  geom_text(aes(label = number), hjust = -1.0)
```



subset of numeric columns

```
#install.packages("dplyr")
```

Subseting the numeric columns

```
library(dplyr)
num_pokemon <- select_if(pokemon, is.numeric)
dim(num_pokemon)
```

```
## [1] 801  34
```

subset pokemon with generation

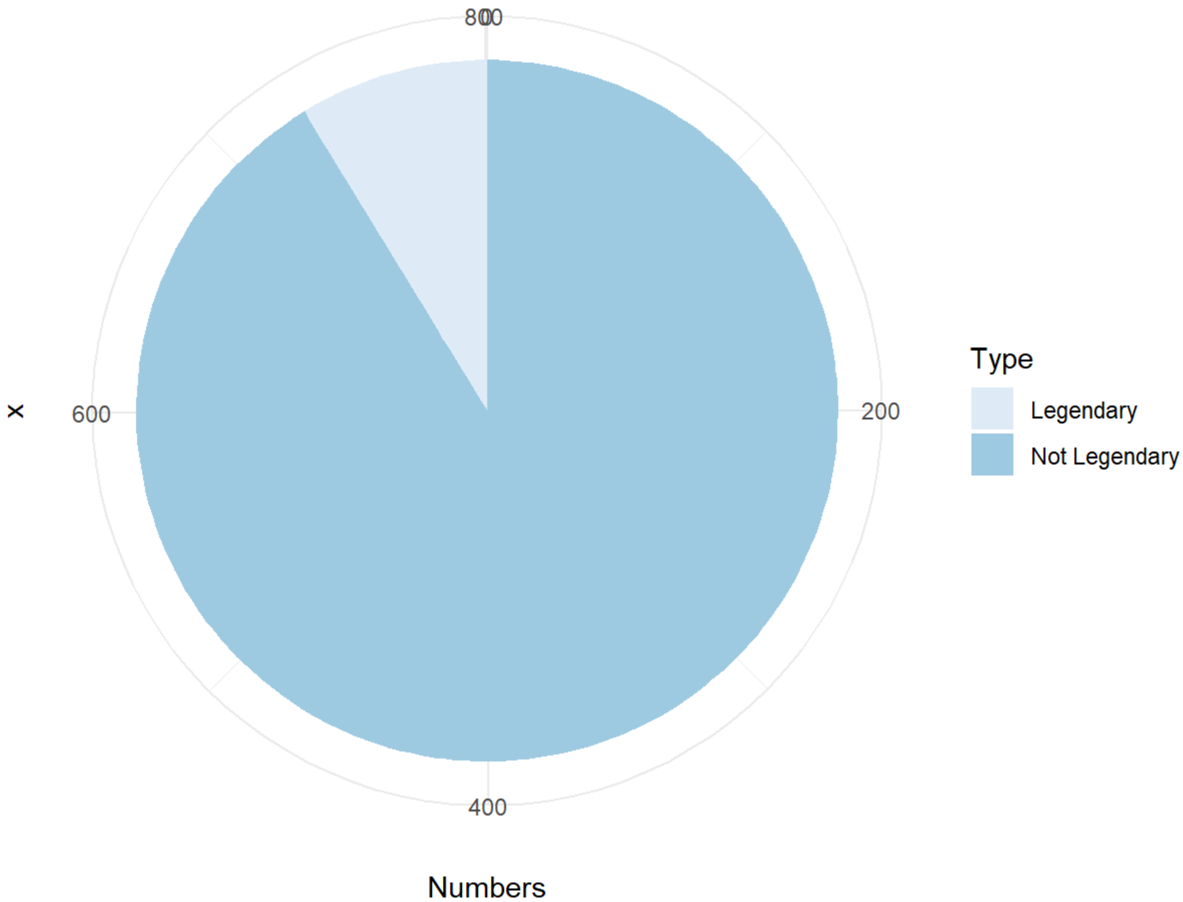
```
generation1<- pokemon[pokemon$generation== 1,]
generation2<- pokemon[pokemon$generation== 2,]
generation3<- pokemon[pokemon$generation== 3,]
generation4<- pokemon[pokemon$generation== 4,]
generation5<- pokemon[pokemon$generation== 5,]
generation6<- pokemon[pokemon$generation== 6,]
generation7<- pokemon[pokemon$generation== 7,]
```

Creating subset of legendary and non legendary pokemons

```
is_legendary<- pokemon[pokemon$is_legendary==1, ]
not_legendary<- pokemon[pokemon$is_legendary==0, ]
#conveted to data frame
legendary <- data.frame(
 Type = c("Legendary", "Not Legendary"),
 Numbers = c(nrow(is_legendary), nrow(not_legendary))
 )
```

Pie chat to visualise the pokemon based on legendary (ggplot)

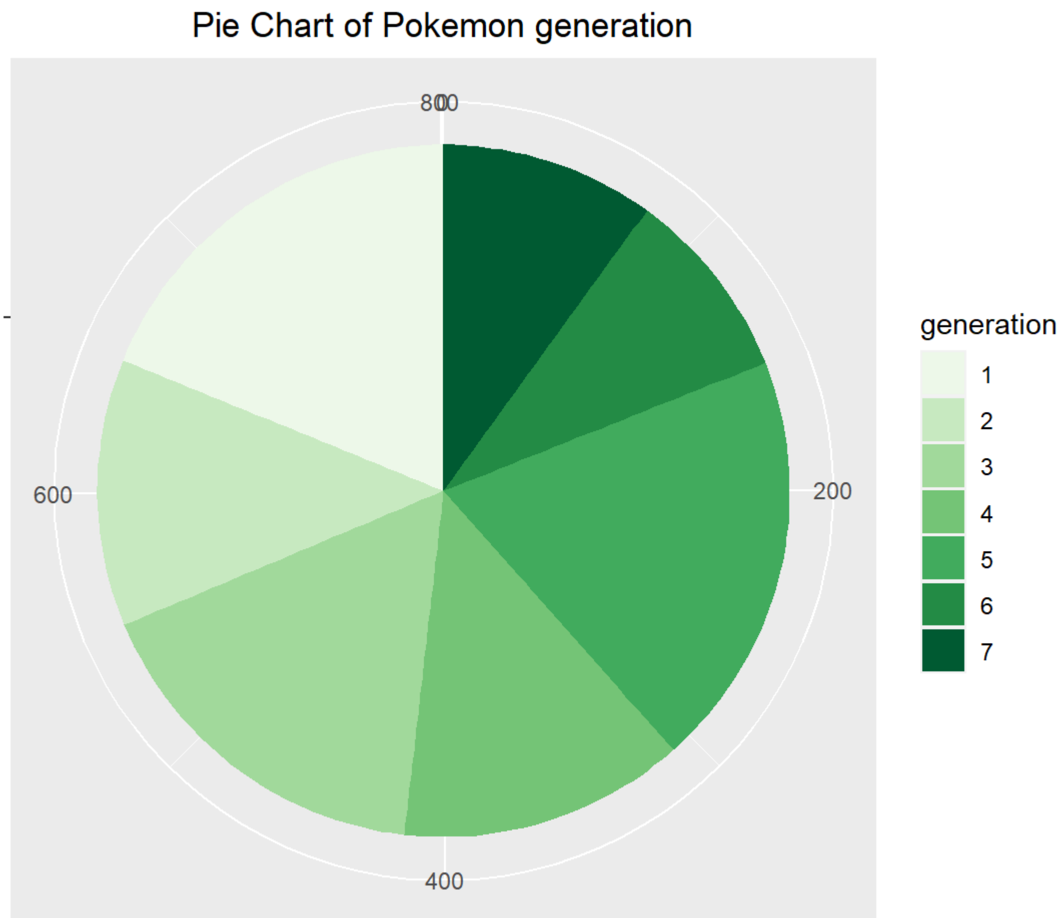
```
ggplot(legendary, aes(x="", y= Numbers, fill= Type))+
  geom_bar(width = 1, stat = "identity")+
  coord_polar("y", start=0)+
  scale_fill_brewer(palette="Blues")+
  theme_minimal()
```



Pie chat to visualise the pokemon based on generation

```
pie <- ggplot(pokemon, aes(x = "", fill = factor(generation))) +
  geom_bar(width = 1) +
  theme(axis.line = element_blank(),
        plot.title = element_text(hjust=0.5)) +
  labs(fill="generation",
       x=NULL,
       y=NULL,
       title="Pie Chart of Pokemon generation")

pie + coord_polar(theta = "y") + scale_fill_brewer(palette = "Green")
```

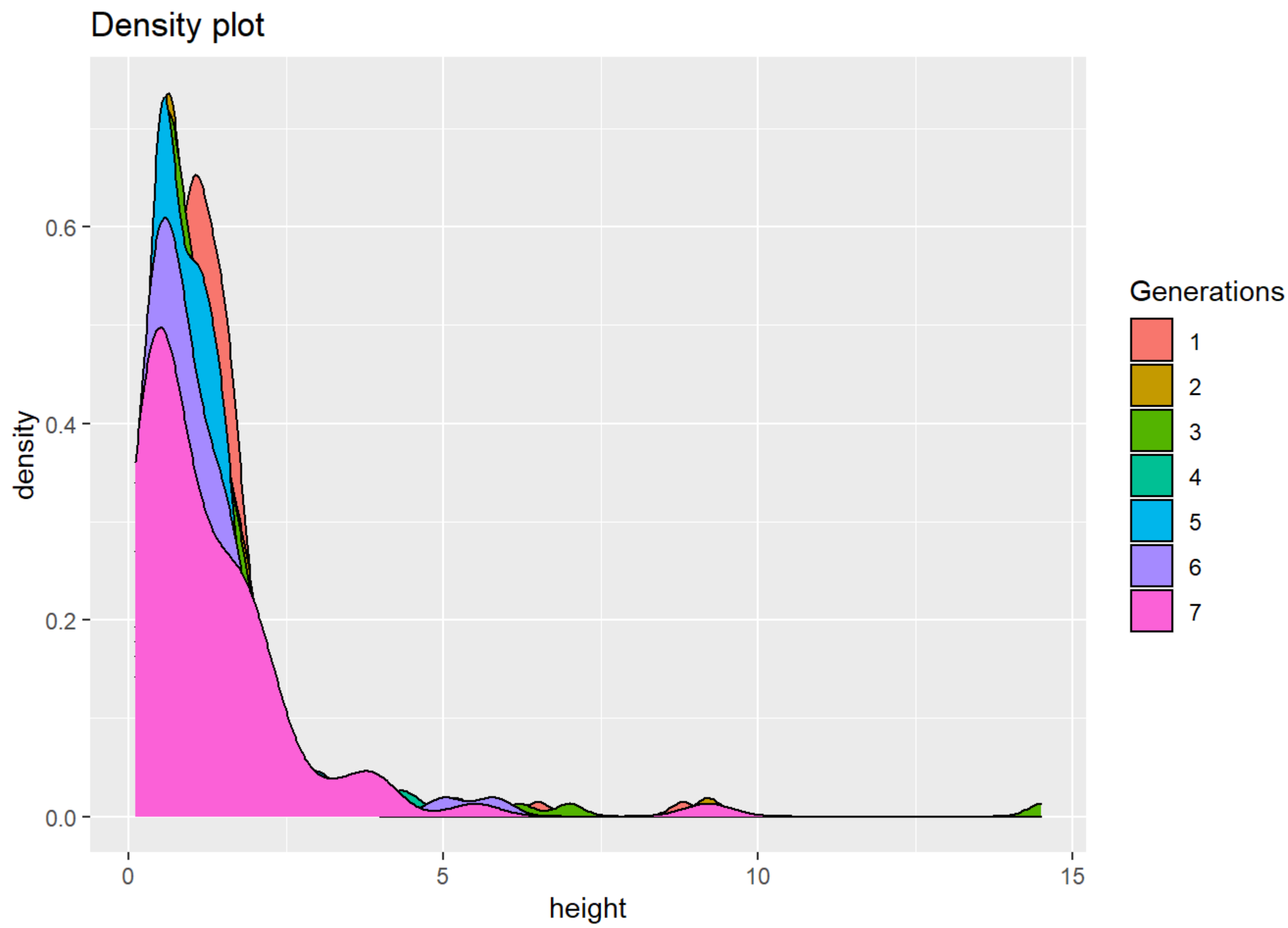
```
## Warning in pal_name(palette, type): Unknown palette Green
```



Density plot to visualise height of the pokemon based on generation

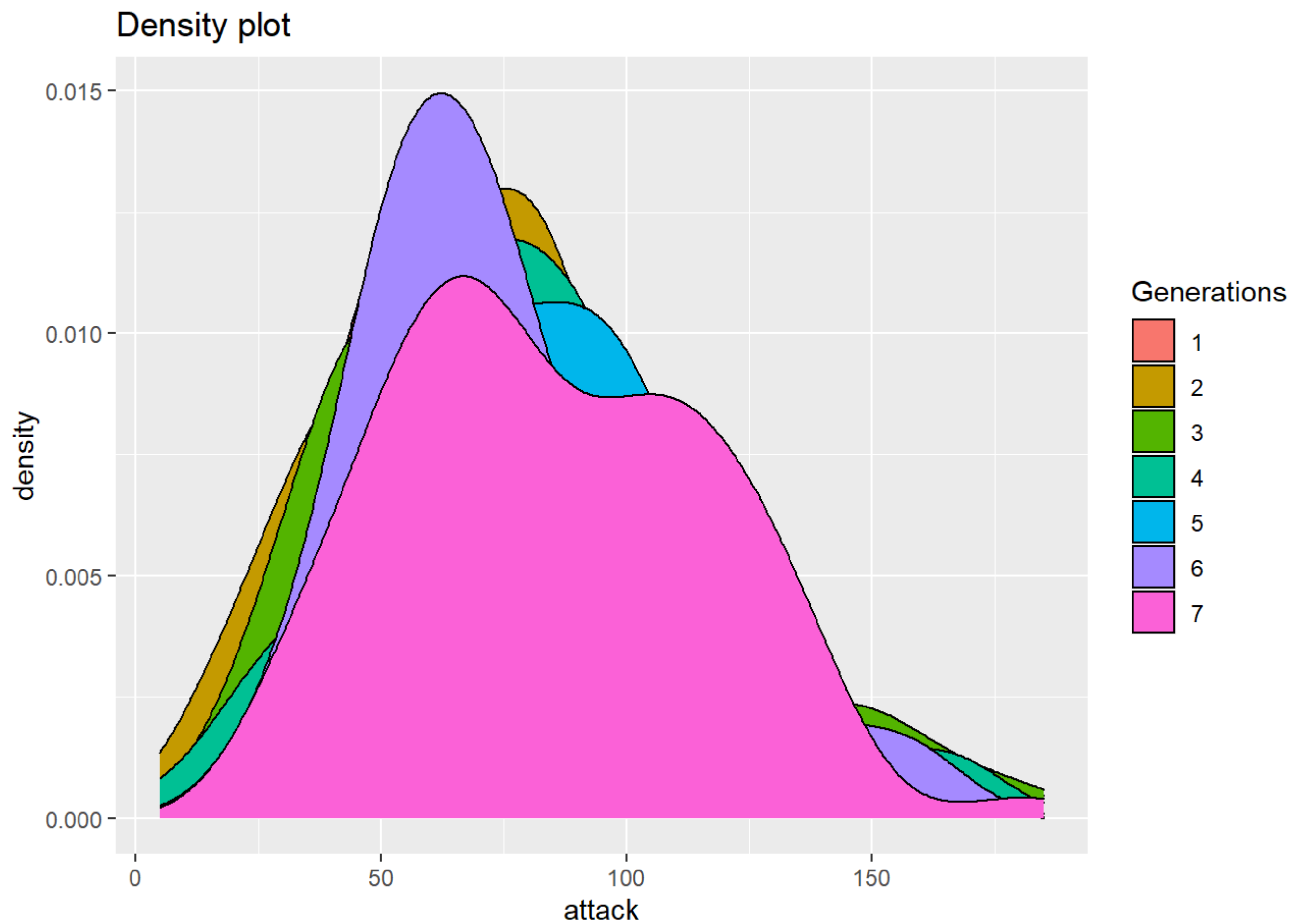
```
g <- ggplot(pokemon, aes(height_m))
g + geom_density(aes(fill=factor(generation))) +
    labs(title="Density plot",
         x="height",
         fill="Generations")
```

```
## Warning: Removed 20 rows containing non-finite values (stat_density).
```



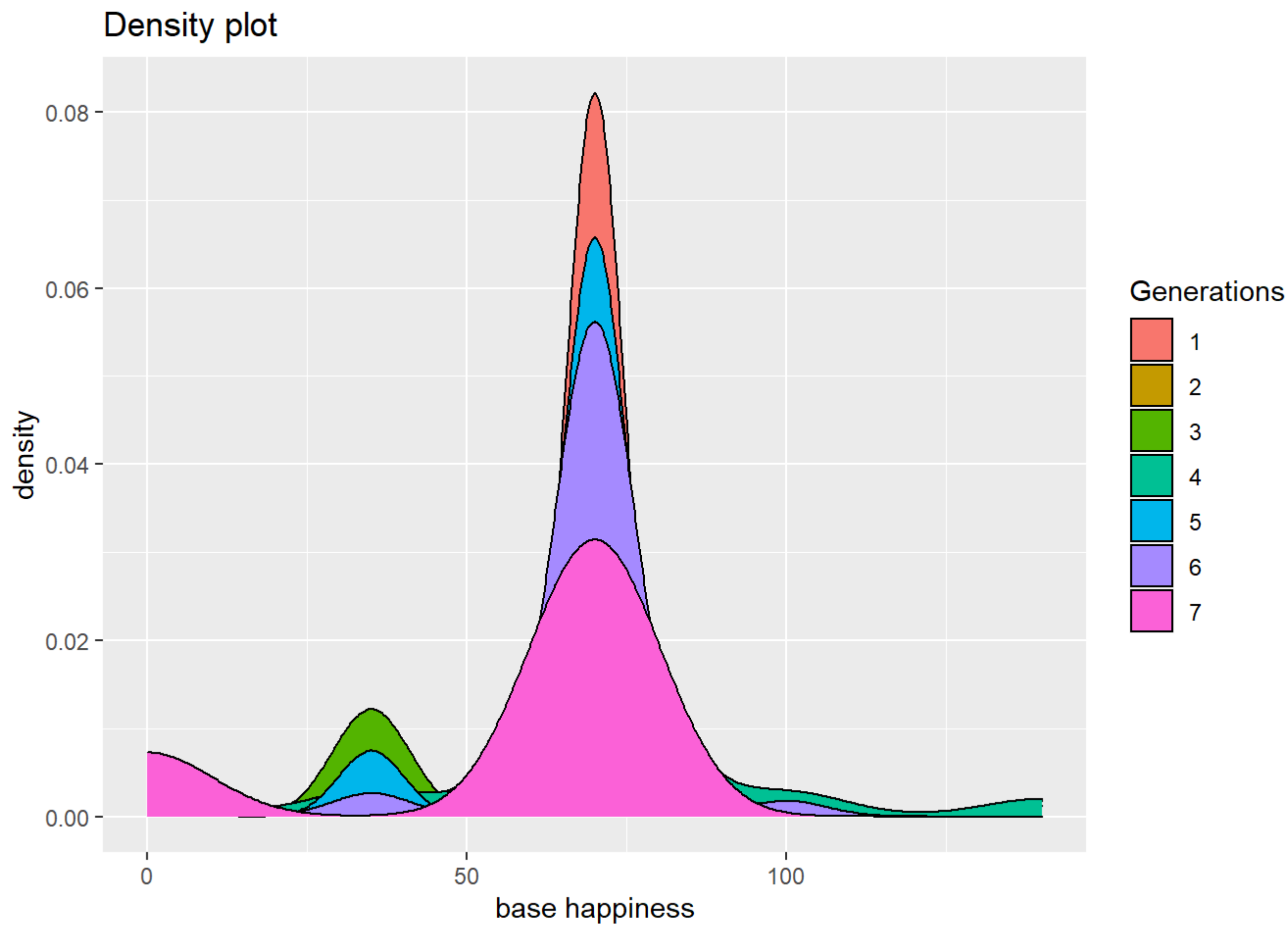
Density plot to visualise attack of the pokemon based on generation

```
g <- ggplot(pokemon, aes(attack))
g + geom_density(aes(fill=factor(generation))) +
    labs(title="Density plot",
         x="attack",
         fill="Generations")
```



Density plot to visualise base happiness offthe pokemon based on generation

```
g <- ggplot(pokemon, aes(base_happiness))
g + geom_density(aes(fill=factor(generation))) +
    labs(title="Density plot",
         x="base happiness",
         fill="Generations")
```




```r
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.0.4
```

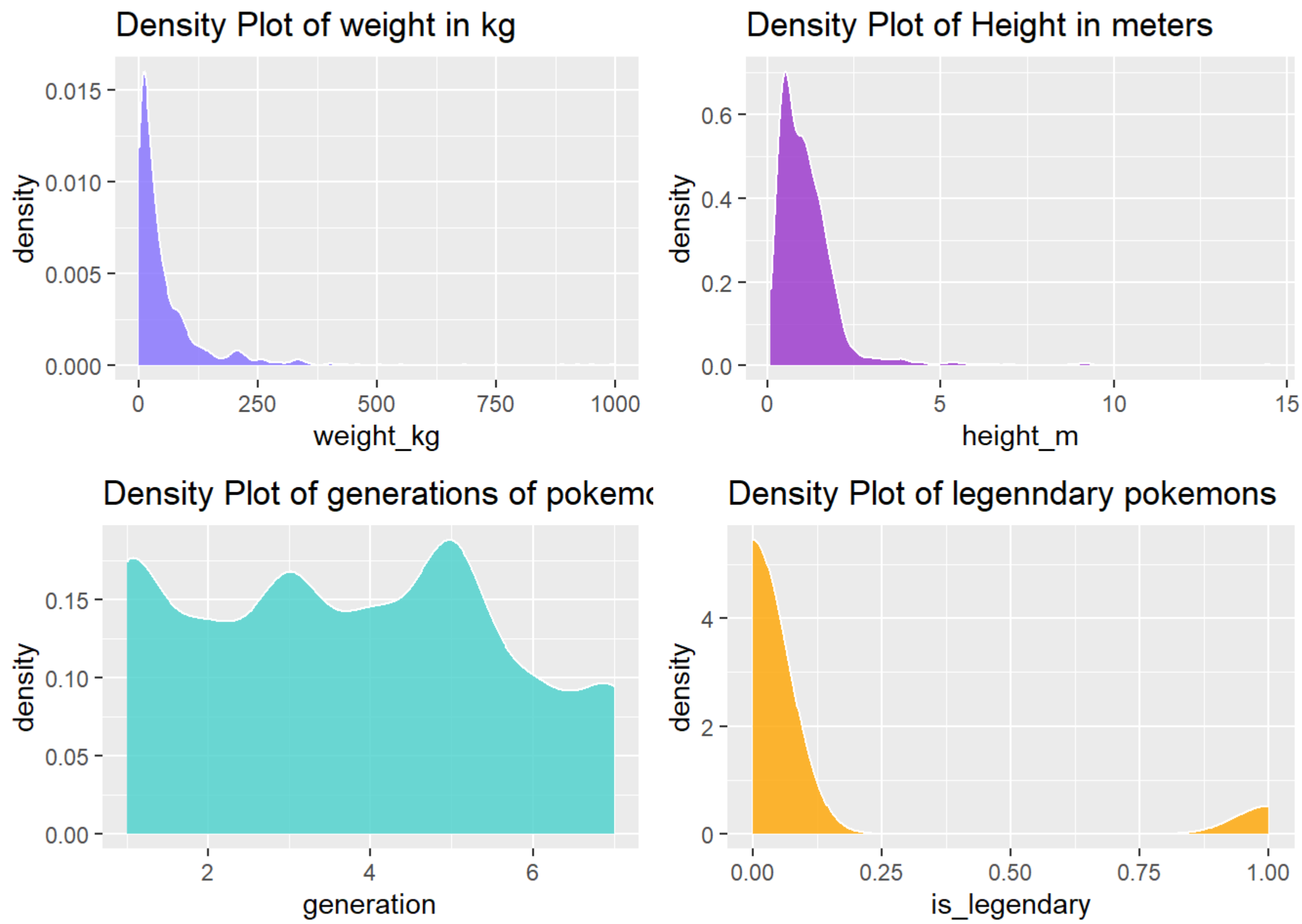
```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(grid)
```

```r
density_weight <- ggplot(data=pokemon, aes(weight_kg)) + geom_density(col="white",fill="slateblu
e1", alpha=0.8) + ggtitle("Density Plot of weight in kg")
density_height <- ggplot(data=pokemon, aes(height_m)) + geom_density(col="white",fill="darkorchi
d", alpha=0.8) + ggtitle("Density Plot of Height in meters")
density_generation <- ggplot(data=pokemon, aes(generation)) + geom_density(col="white",fill="med
iumturquoise", alpha=0.8) + ggtitle("Density Plot of generations of pokemons")
density_legendary <- ggplot(data=pokemon, aes(is_legendary)) + geom_density(col="white",fill="or
ange", alpha=0.8) + ggtitle("Density Plot of legenndary pokemons")
grid.arrange(density_weight, density_height, density_generation, density_legendary)
```

```
## Warning: Removed 20 rows containing non-finite values (stat_density).

## Warning: Removed 20 rows containing non-finite values (stat_density).
```



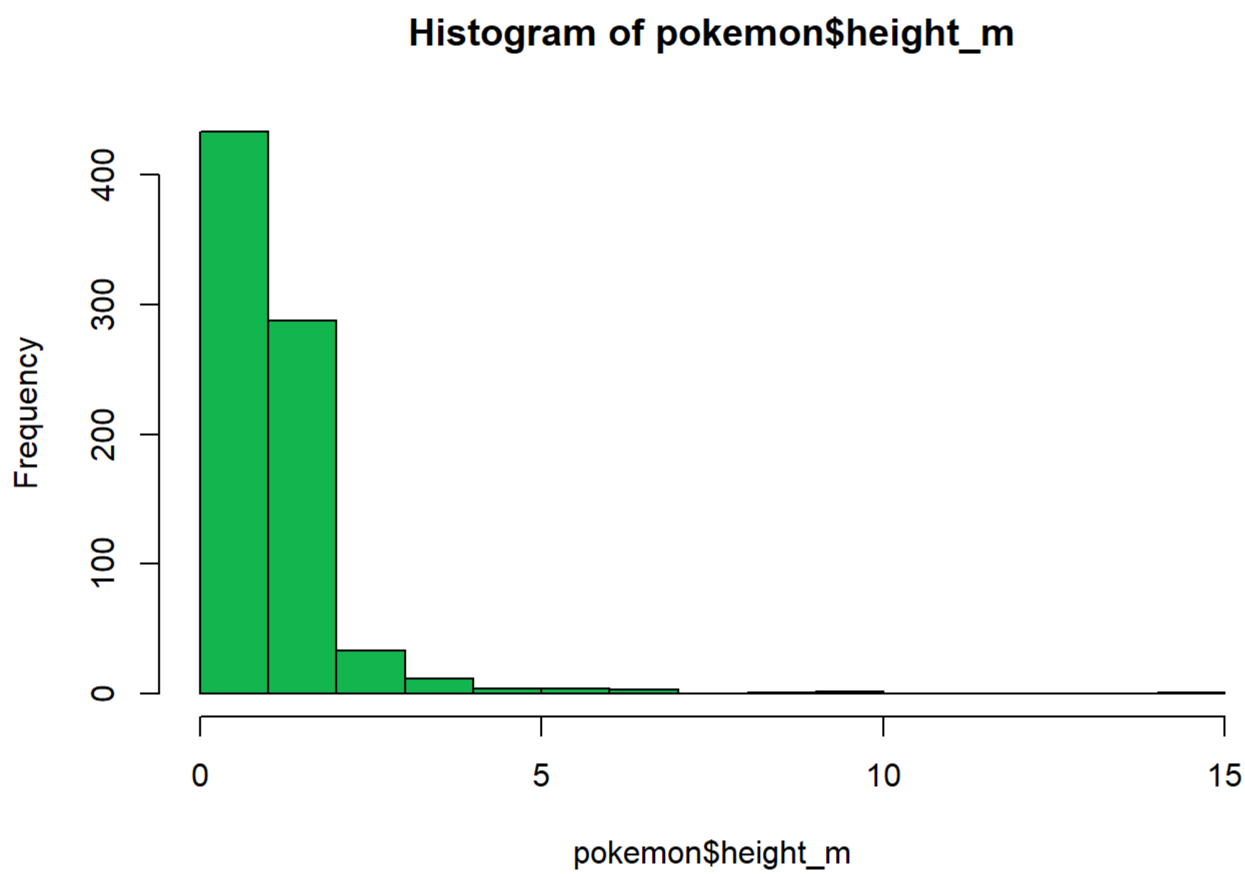

```
#install.packages("plotrix")
```

Subset dataset into 3 categories based on height

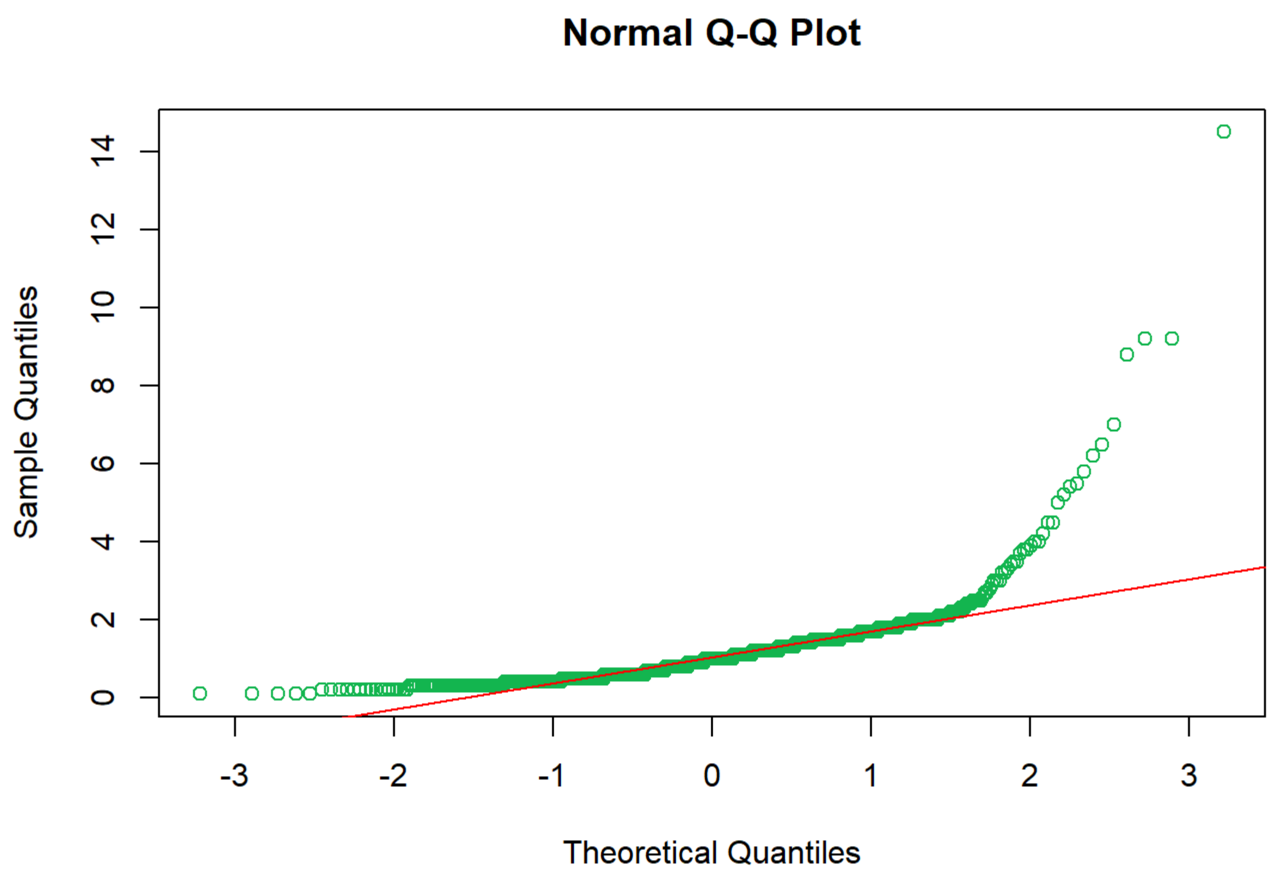
```
small <- na.omit(num_pokemon[num_pokemon$height_m <=0.7, ])
mid <- na.omit(num_pokemon[num_pokemon$height_m >0.7 & pokemon$height_m <=1.4, ])
big <- na.omit(num_pokemon[num_pokemon$height_m >1.4, ])
```

normality check of hieght of pokemon

```
hist(pokemon$height_m, col = "#13b54f")
```




```
#quntile quantile plot
qqnorm(pokemon$height_m, col = "#13b54f")
#qqline for normal distribution
qqline(pokemon$height_m,col='red')
```



Checking normality

First we will perform **Shapiro-Wilk Normality Test** to check whether the height of the pokemon has normal distribution.

- Null Hypothesis, $H_0 :=$ height is normally distributed
- Alternate Hypothesis, $H_a :=$ height is **NOT** normally distributed

```
#normaltiy test
shapiro.test(pokemon$height_m)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  pokemon$height_m
## W = 0.62103, p-value < 2.2e-16
```

p value is soo low , so we can reject null hypothesis , hence distribution is not normal

Checking normality for defence of pokemon

First we will perform **Shapiro-Wilk Normality Test** to check whether the attack of the pokemon has normal distribution.

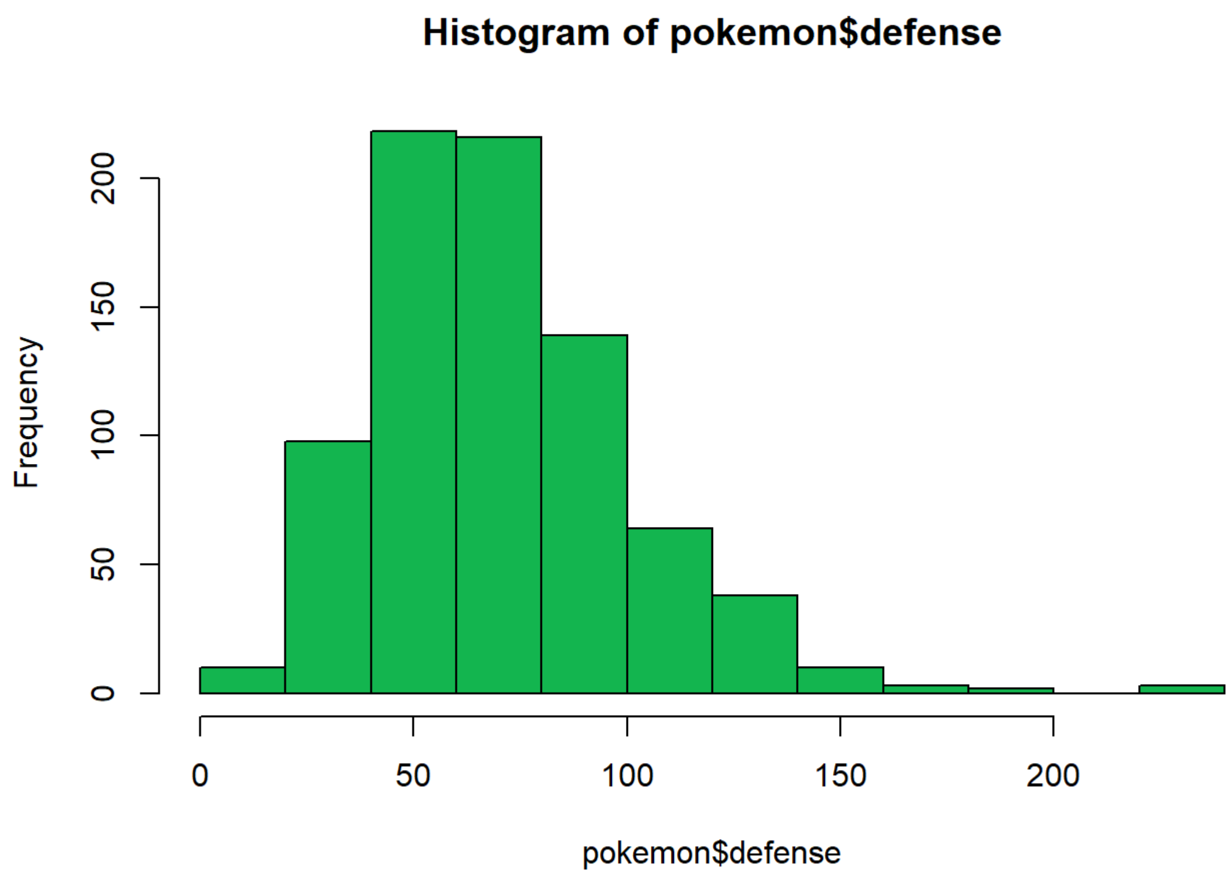
- Null Hypothesis, $H_0 :=$ defense is normally distributed

- Alternate Hypothesis, $H_a :=$ defense is **NOT** normally distributed

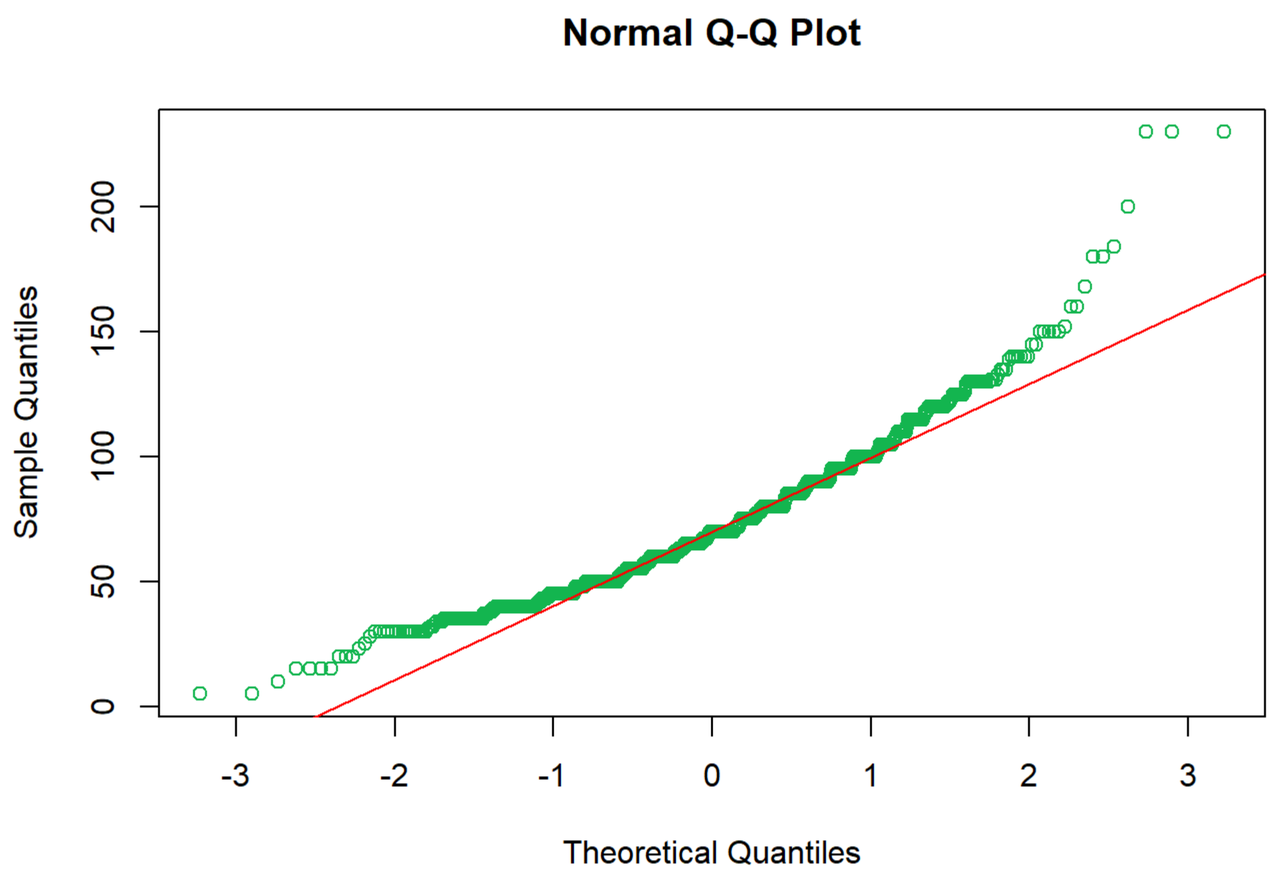
```
shapiro.test(pokemon$defense)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  pokemon$defense
## W = 0.93984, p-value < 2.2e-16
```

```
hist(pokemon$defense, col = "#13b54f")
```




```
#quntile quantile plot
qqnorm(pokemon$defense, col = "#13b54f")
#qqline for normal distribution
qqline(pokemon$defense,col='red')
```



Checking normality for attack of pokemon

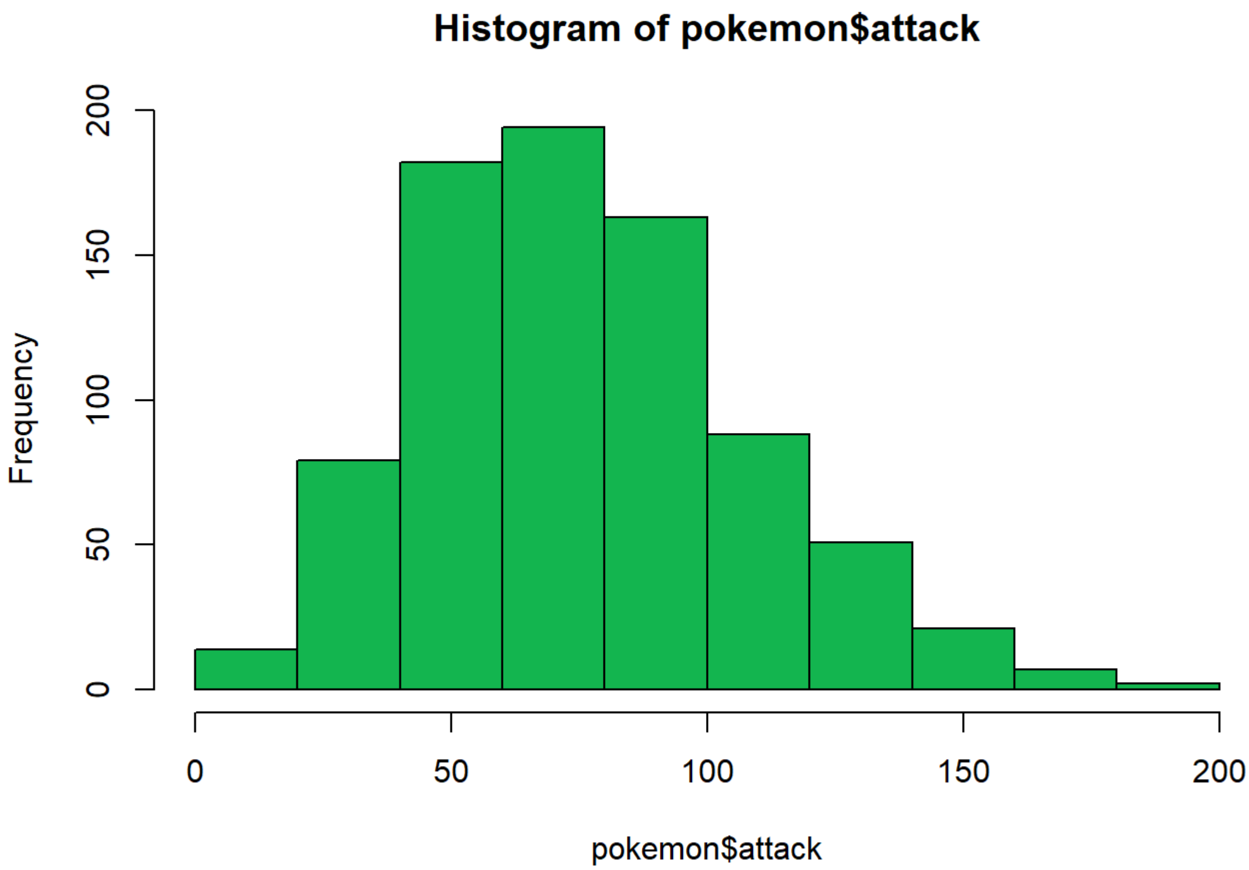
First we will perform **Shapiro-Wilk Normality Test** to check whether the attack of the pokemon has normal distribution.

- Null Hypothesis, $H_0 :=$ attack is normally distributed
- Alternate Hypothesis, $H_a :=$ attack is **NOT** normally distributed

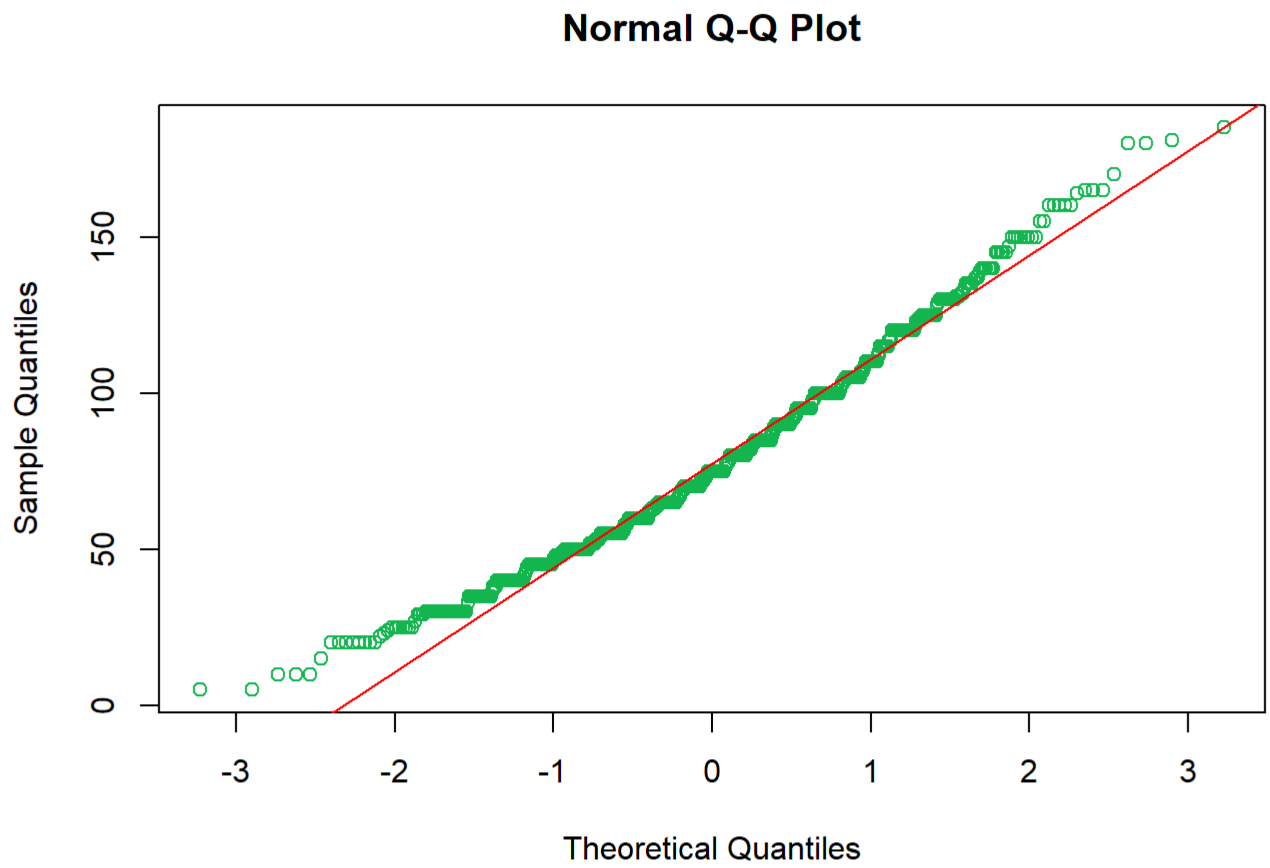
```
shapiro.test(pokemon$attack)
```

```
## 
##  Shapiro-Wilk normality test
## 
## data:  pokemon$attack
## W = 0.97948, p-value = 3.581e-09
```

```
hist(pokemon$attack, col = "#13b54f")
```




```
#quntile quantile plot
qqnorm(pokemon$attack, col = "#13b54f")
#qqline for normal distribution
qqline(pokemon$attack,col='red')
```



For each type of height and attack we have the p-value less than 0.05. Hence, we reject our Null Hypothesis (that the data vectors are normally distributed) and conclude that the set of mean values forheight and attack of pokemon is not normally distributed. Thus we cannot use **t test** or **ANOVA** to compare means of our datasets.

Now we will need a non parametric test to compare the means of our datasets pairwise. We will use **Wilcoxon test** to compare the mean.

Wilcoxon test between defense and attack of pokemon:

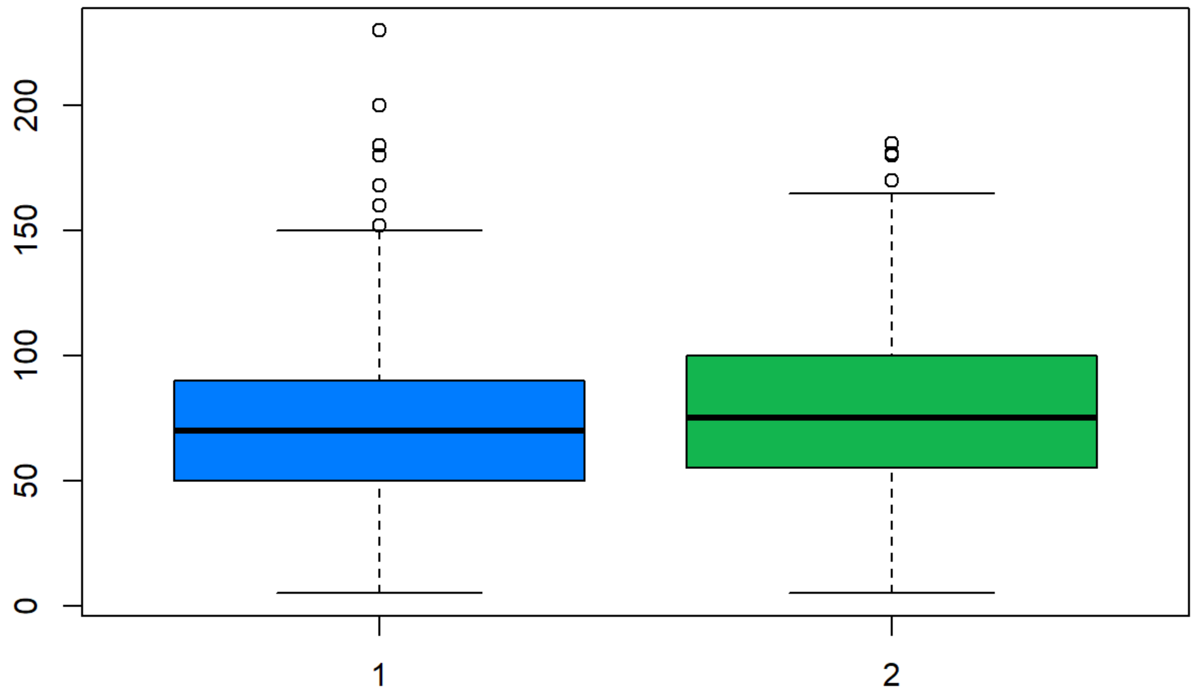
- Null Hypothesis, H_0 := The difference between median value of defense and attack of pokemon is zero.
- Alternate Hypothesis, H_a := The difference between median value of defense and attack of pokemon is less than zero

```
wilcox.test(pokemon$defense,pokemon$attack
            ,paired=TRUE,alternative = "less")
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  pokemon$defense and pokemon$attack
## V = 101335, p-value = 3.01e-08
## alternative hypothesis: true location shift is less than 0
```

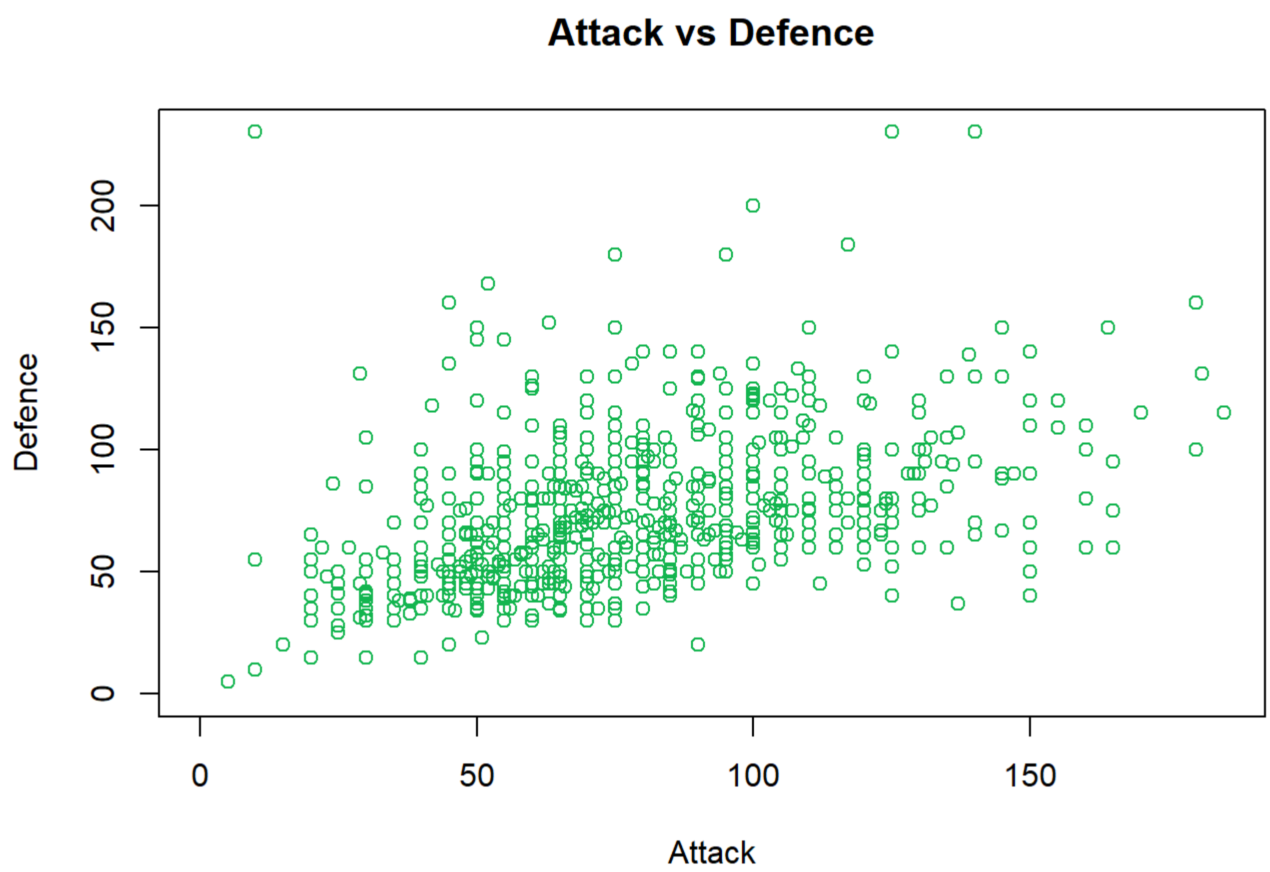
Since we get the p-value is less than 0.05 hence we reject our Null hypothesis and get that **median value of defense and attack of pokemon is less than zero.**

```
boxplot(pokemon$defense,pokemon$attack
        , col = c("#007cff","#13b54f"))
```



correlation in attack and defence for pokemon

```
plot(pokemon$attack, pokemon$defense, main="Attack vs Defence", xlab = "Attack", ylab = "Defenc
e", xlim = c(0, max(pokemon$attack)), ylim= c(0, max(pokemon$defense)), col="#13b54f")
```



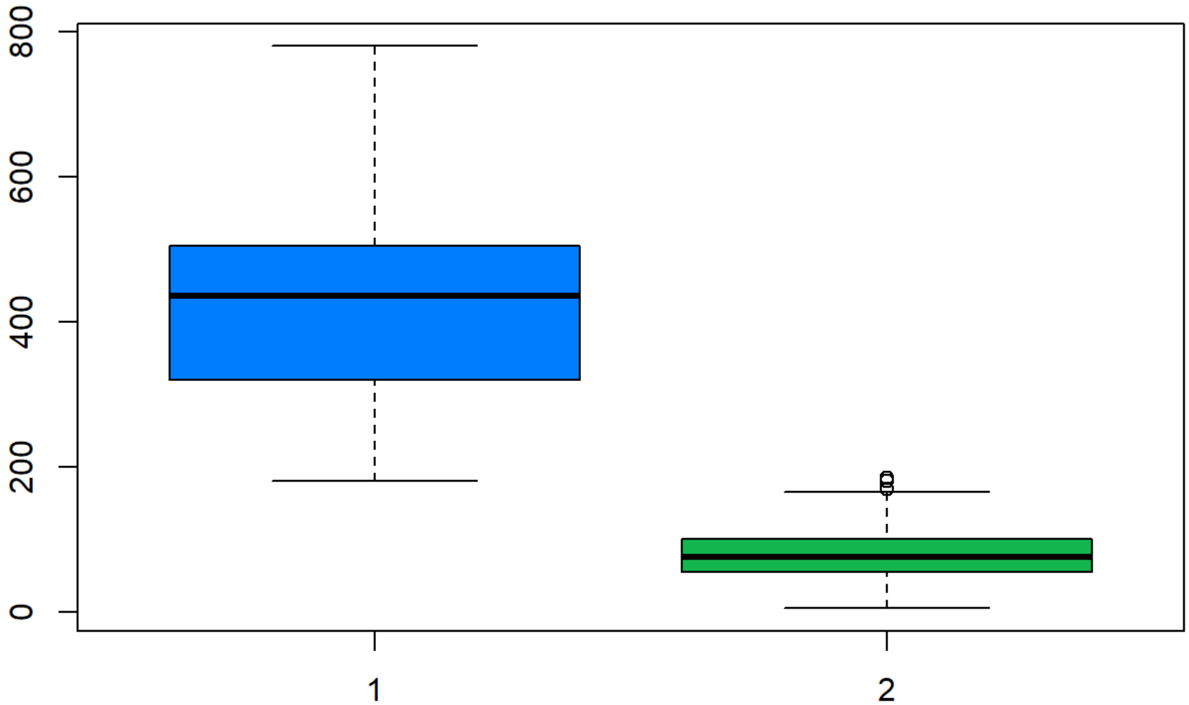

```
cor.test(pokemon$defense,pokemon$attack)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pokemon$defense and pokemon$attack
## t = 15.007, df = 799, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4130612 0.5212543
## sample estimates:
##       cor
## 0.4689149
```

The attact and defence of pokemon are a bit correlated

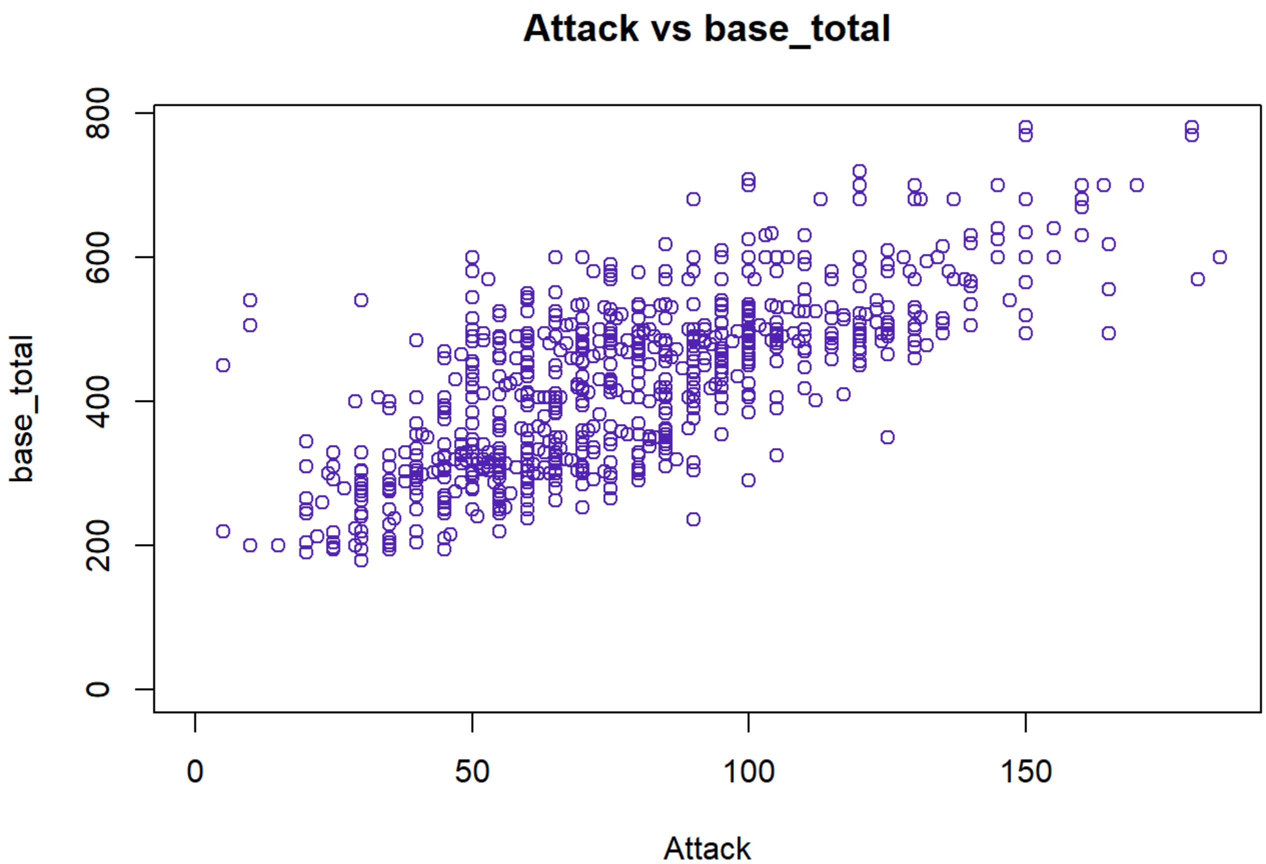
box plot for attack and base total of the pokemon

```
boxplot(pokemon$base_total,pokemon$attack, col = c("#007cff","#13b54f"))
```



corelation btw attavck and base total

```
plot(pokemon$attack, pokemon$base_total, main="Attack vs base_total", xlab = "Attack", ylab = "b
ase_total", xlim = c(0, max(pokemon$attack)), ylim= c(0, max(pokemon$base_total)), col="#5122b1"
)
```



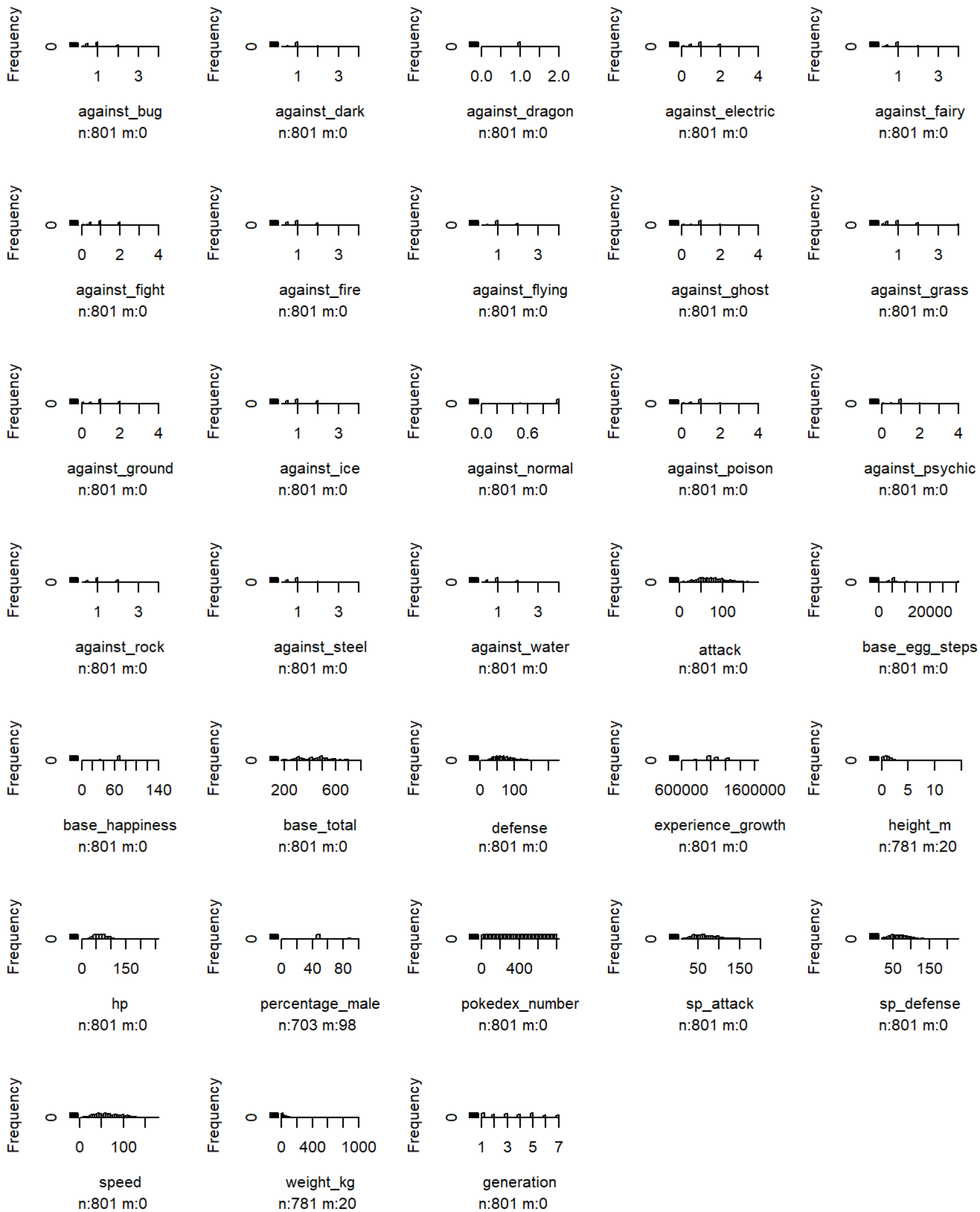

```
cor.test(pokemon$attack, pokemon$base_total)
```

```
## 
##  Pearson's product-moment correlation
## 
## data:  pokemon$attack and pokemon$base_total
## t = 30.204, df = 799, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6960683 0.7609199
## sample estimates:
##       cor 
## 0.7301341
```

The attack and base_total of pokemon are correlated

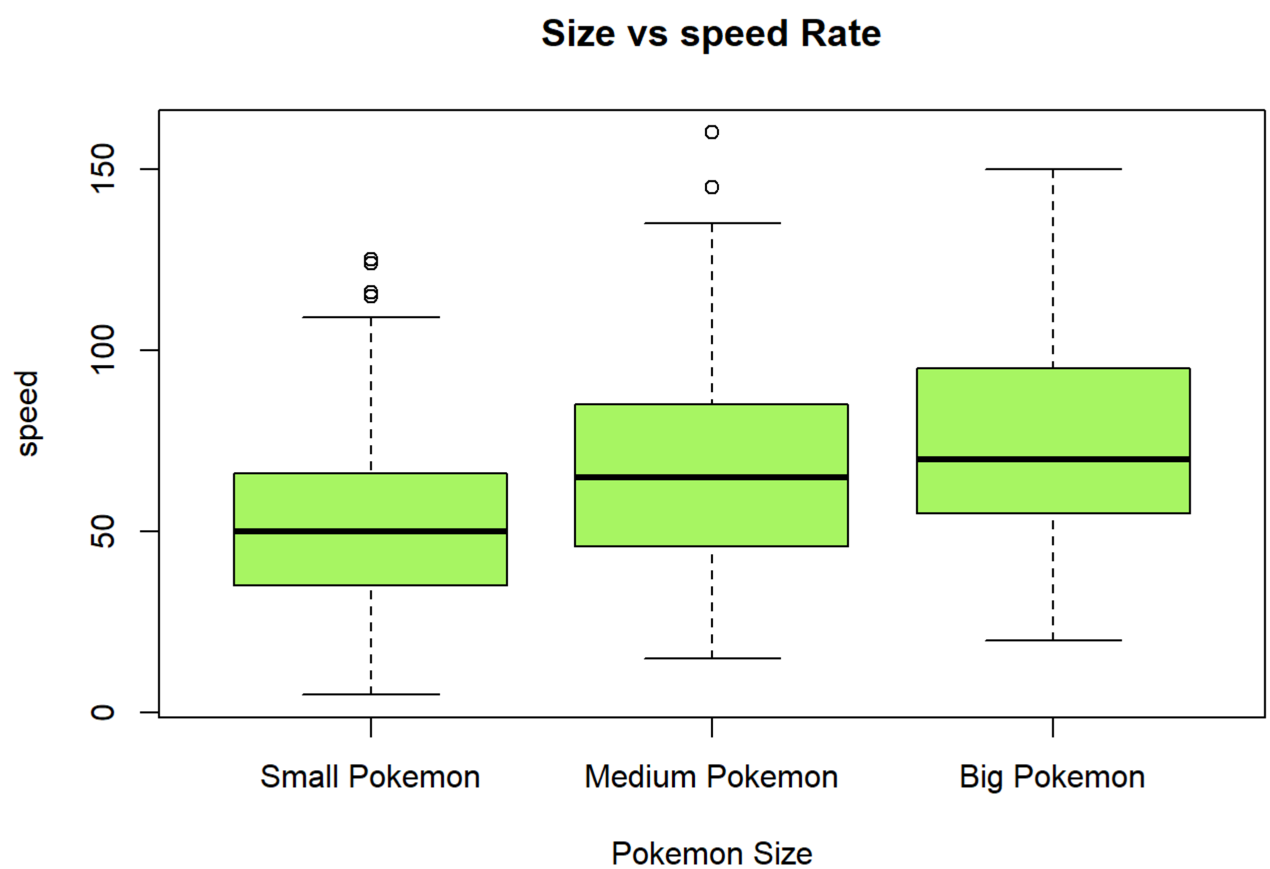
histogram plot for all feature vector of pokemon

```
hist.data.frame(num_pokemon, col = c("#007cff","#13b54f"))
```


Frequency
against_bug
n:801 m:0
against_dark
n:801 m:0
against_dragon
n:801 m:0
against_electric
n:801 m:0
against_fairy
n:801 m:0
against_fight
n:801 m:0
against_fire
n:801 m:0
against_flying
n:801 m:0
against_ghost
n:801 m:0
against_grass
n:801 m:0
against_ground
n:801 m:0
against_ice
n:801 m:0
against_normal
n:801 m:0
against_poison
n:801 m:0
against_psychic
n:801 m:0
against_rock
n:801 m:0
against_steel
n:801 m:0
against_water
n:801 m:0
attack
n:801 m:0
base_egg_steps
n:801 m:0
base_happiness
n:801 m:0
base_total
n:801 m:0
defense
n:801 m:0
experience_growth
n:801 m:0
height_m
n:781 m:20
hp
n:801 m:0
percentage_male
n:703 m:98
pokedex_number
n:801 m:0
sp_attack
n:801 m:0
sp_defense
n:801 m:0
speed
n:801 m:0
weight_kg
n:781 m:20
generation
n:801 m:0

hypothesis small pokemon has greater speed

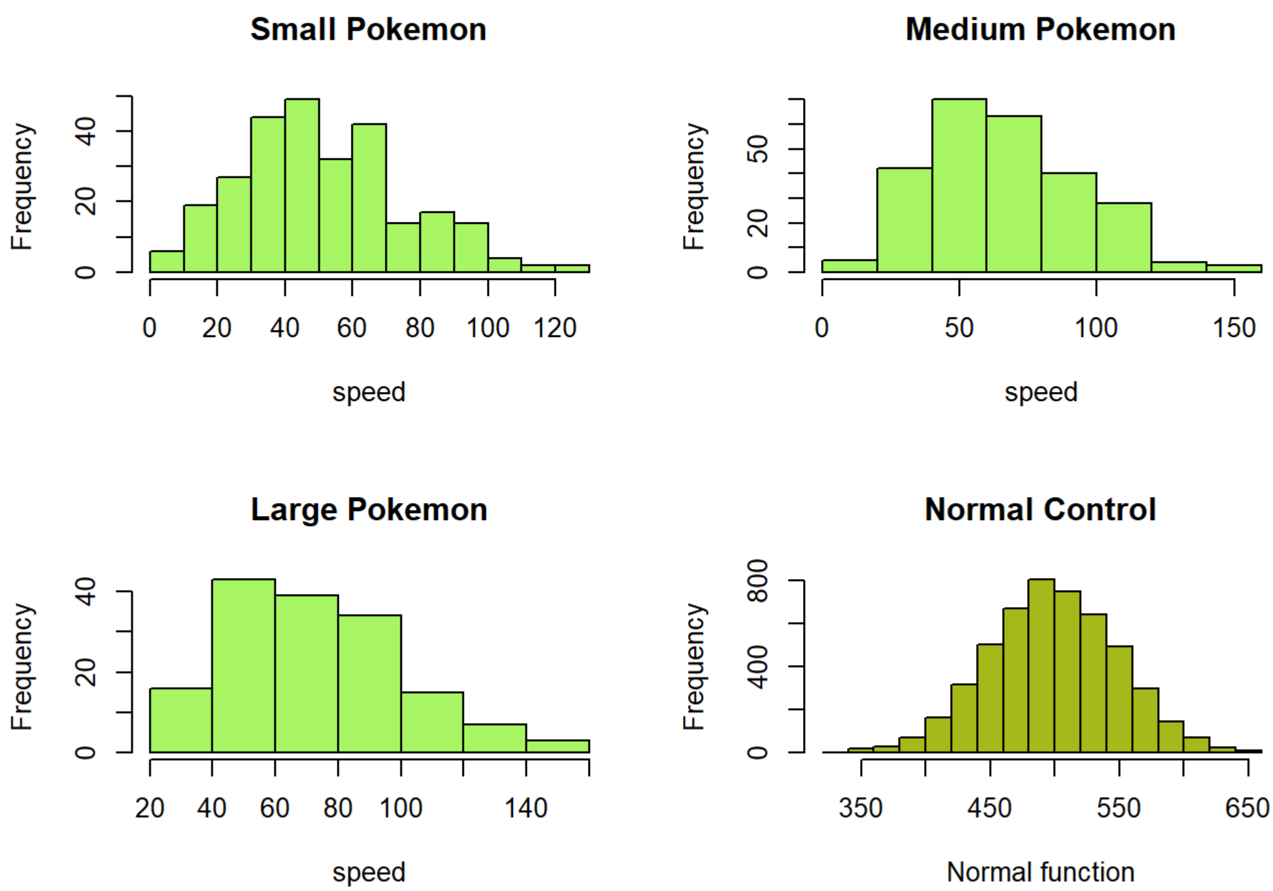
```
boxplot(as.numeric(small$speed), as.numeric(mid$speed), as.numeric(big$speed),col = "#a7f562", m
ain="Size vs speed Rate", names = c("Small Pokemon", "Medium Pokemon", "Big Pokemon"), xlab="Pok
emon Size", ylab="speed")
```



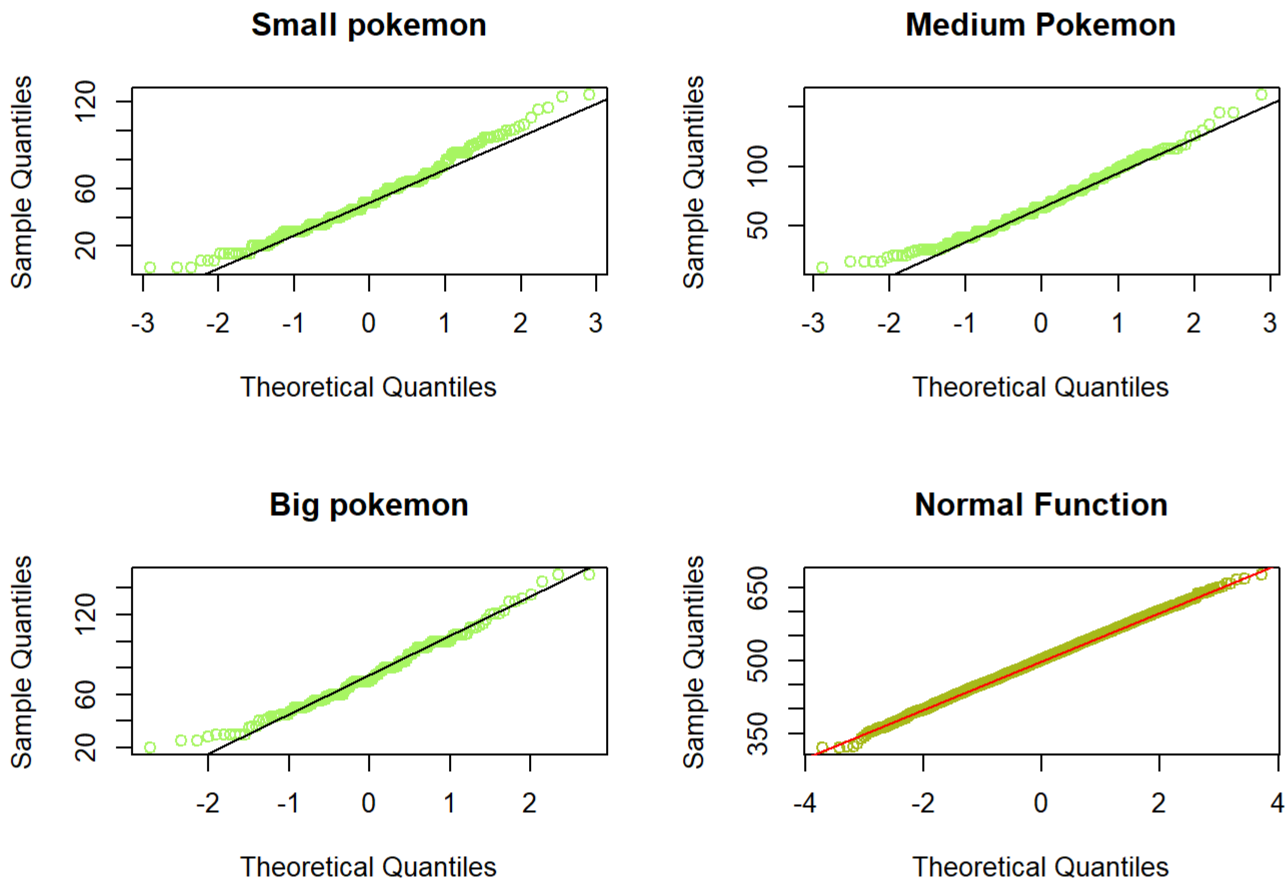
so the speed of big and medium pokemon has greater speed hence my hypothesis is wrong

normality testing

```
par(mfrow=c(2,2))
hist(as.numeric(small$speed), main = "Small Pokemon", col = "#a7f562", xlab = "speed")
hist(as.numeric(mid$speed), main = "Medium Pokemon", col = "#a7f562", xlab = "speed")
hist(as.numeric(big$speed), main = "Large Pokemon", col = "#a7f562", xlab = "speed")
hist(rnorm(5000,mean=500,sd=50), main = "Normal Control",  col="#A6B91A", xlab = "Normal functio
n")
```




```
par(mfrow=c(2,2))
qqnorm(as.numeric(small$speed), col = "#a7f562", main = "Small pokemon" )
qqline(as.numeric(small$speed))

qqnorm(as.numeric(mid$speed), col="#a7f562", main = "Medium Pokemon")
qqline(as.numeric(mid$speed))

qqnorm(as.numeric(big$speed), col = "#a7f562", main = "Big pokemon")
qqline(as.numeric(big$speed))

qqnorm(rnorm(5000,mean=500,sd=50),  col="#A6B91A", main = "Normal Function")
qqline(rnorm(5000,mean=500,sd=50), col= "red")
```



shaphiro test

First we will perform **Shapiro-Wilk Normality Test** to check whether the height of the pokemon has normal distribution.

- Null Hypothesis, H_0 := height is normally distributed
- Alternate Hypothesis, H_a := height is **NOT** normally distributed

```
shapiro.test(small$speed)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  small$speed
## W = 0.97935, p-value = 0.000559
```

```
shapiro.test(mid$speed)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  mid$speed
## W = 0.97892, p-value = 0.0007782
```

```
shapiro.test(big$speed)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  big$speed
## W = 0.98171, p-value = 0.03553
```

- small pokemon is not normally distributed
- mid pokemon is not normally distributed
- big pokemon is not normally distributed

mean compare with medium and big pokemon

For each type of height and attack we have the p-value less than 0.05. Hence, we reject our Null Hypothesis (that the data vectors are normally distributed) and conclude that the set of mean values forheight and attack of pokemon is not normally distributed. Thus we cannot use **t test** or **ANOVA** to compare means of our datasets.

Now we will need a non parametric test to compare the means of our datasets pairwise. We will use **Wilcoxon test** to compare the mean.

Wilcoxon test between defense and attack of pokemon:

- Null Hypothesis, $H_0 :=$ The difference between median value of speed of mid and speed of big is zero.
- Alternate Hypothesis, $H_a :=$ The difference between median value of speed of mid and speed of big is less than zero

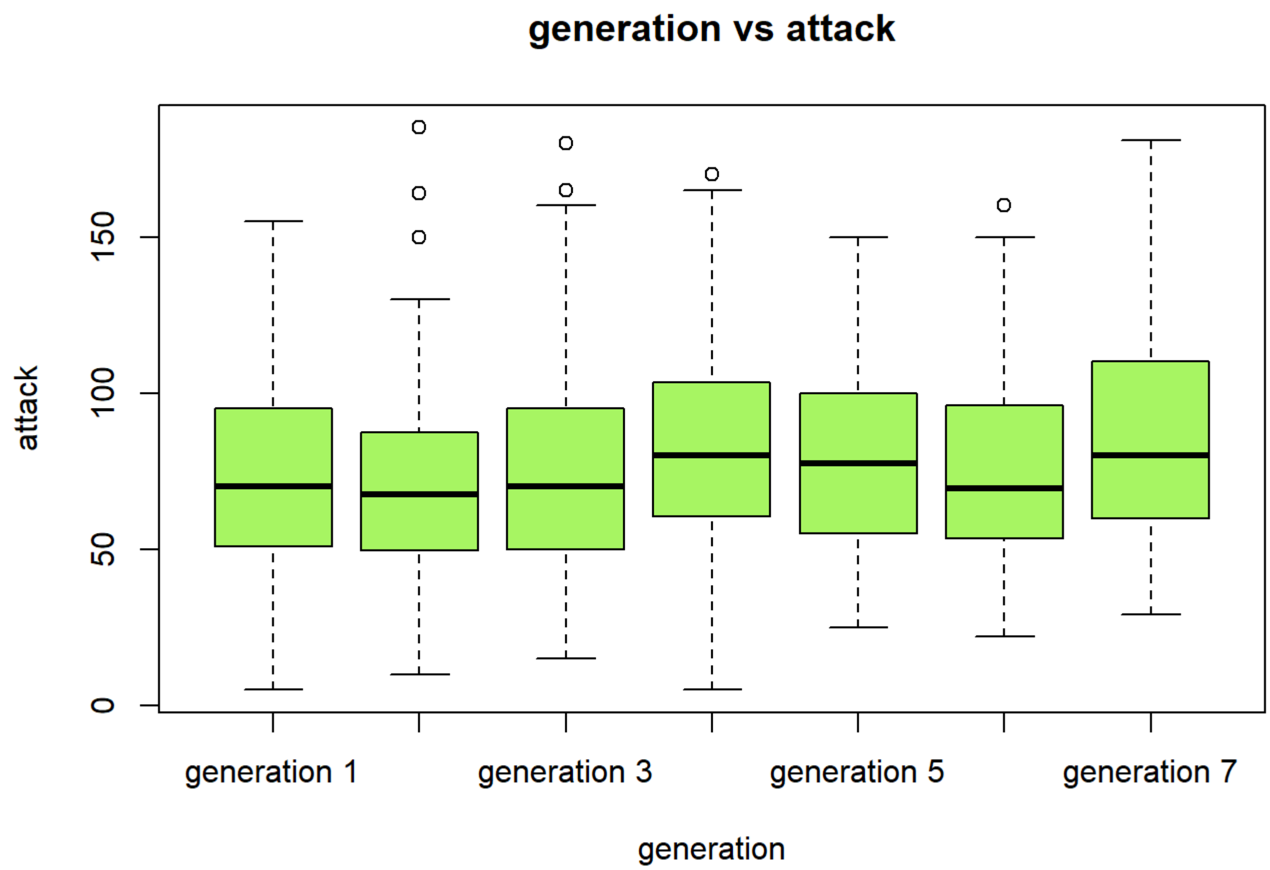
```
wilcox.test(big$speed, mid$speed)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  big$speed and mid$speed
## W = 22726, p-value = 0.02098
## alternative hypothesis: true location shift is not equal to 0
```

- there is small difference in median of big and mid pokemon speed

hypothesis attact increase with generation

```
boxplot(as.numeric(generation1$attack), as.numeric(generation2$attack), as.numeric(generation3$a
ttack),as.numeric(generation4$attack), as.numeric(generation5$attack), as.numeric(generation6$at
tack), as.numeric(generation7$attack),col = "#a7f562", main="generation vs attack", names = c("g
eneration 1","generation 2","generation 3","generation 4","generation 5","generation 6","generat
ion 7" ), xlab="generation", ylab="attack")
```



- we cant see any good varation of attack with increase in generation

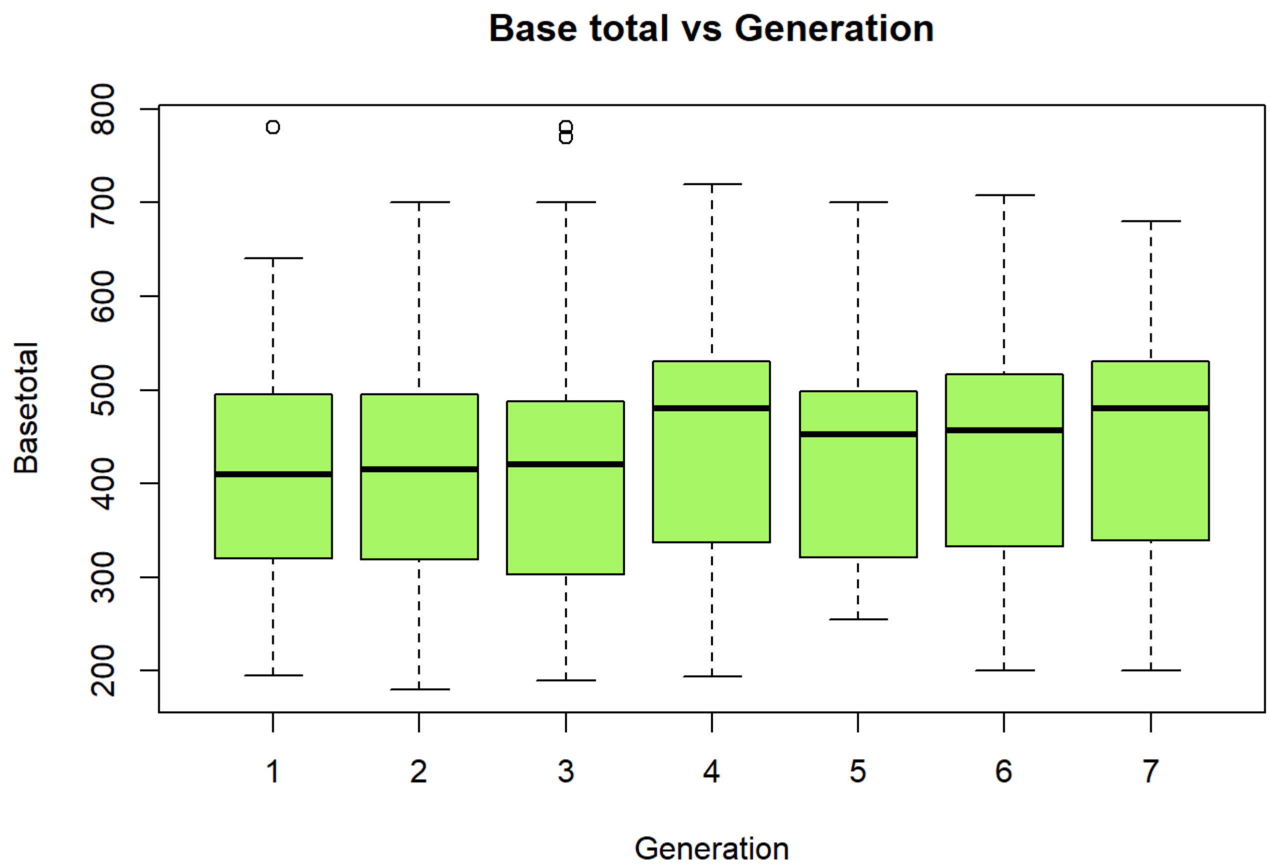
kuskal test

```
kruskal.test(pokemon$generation~pokemon$attack)
```

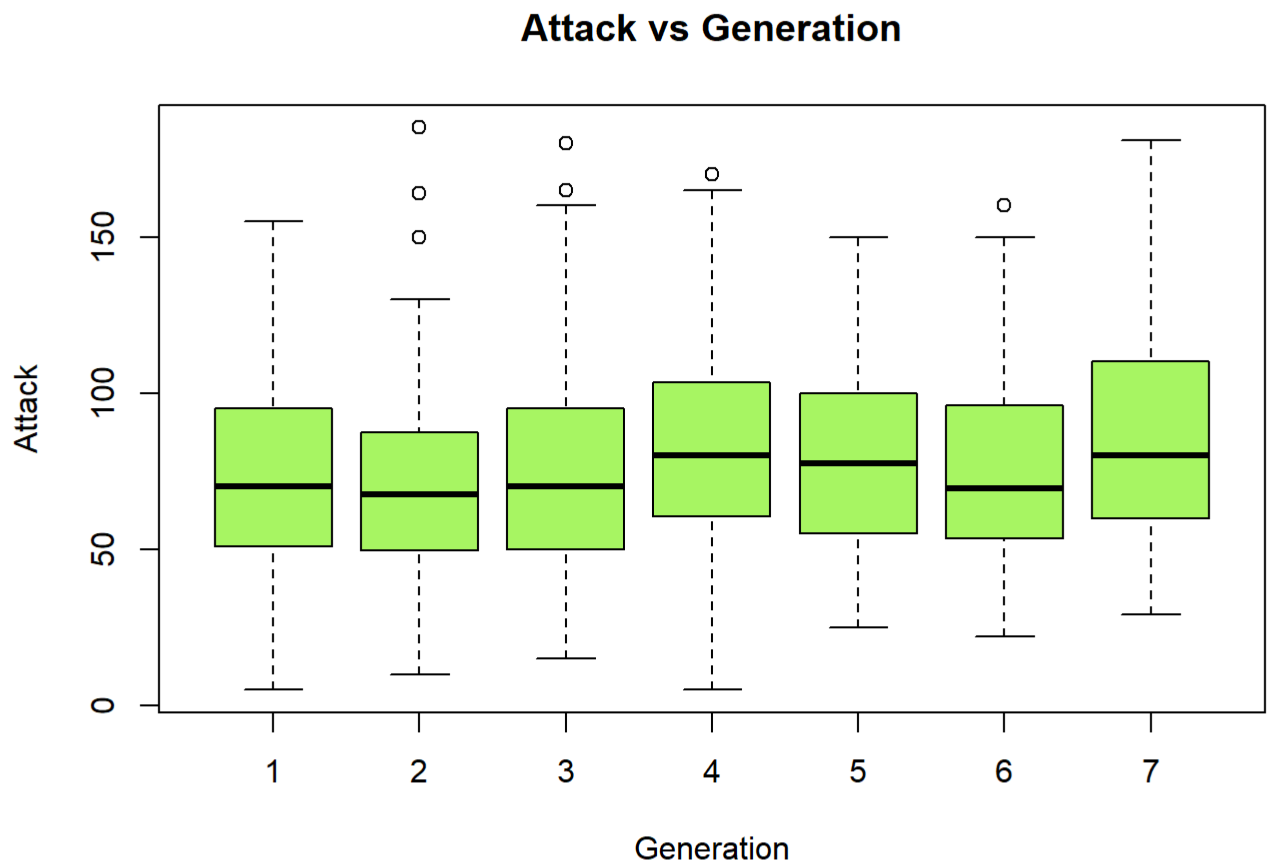
```
##
##  Kruskal-Wallis rank sum test
##
## data:  pokemon$generation by pokemon$attack
## Kruskal-Wallis chi-squared = 162.94, df = 113, p-value = 0.001476
```

- note here p value is very less which implies there is significantly no difference in attach with increase in generations

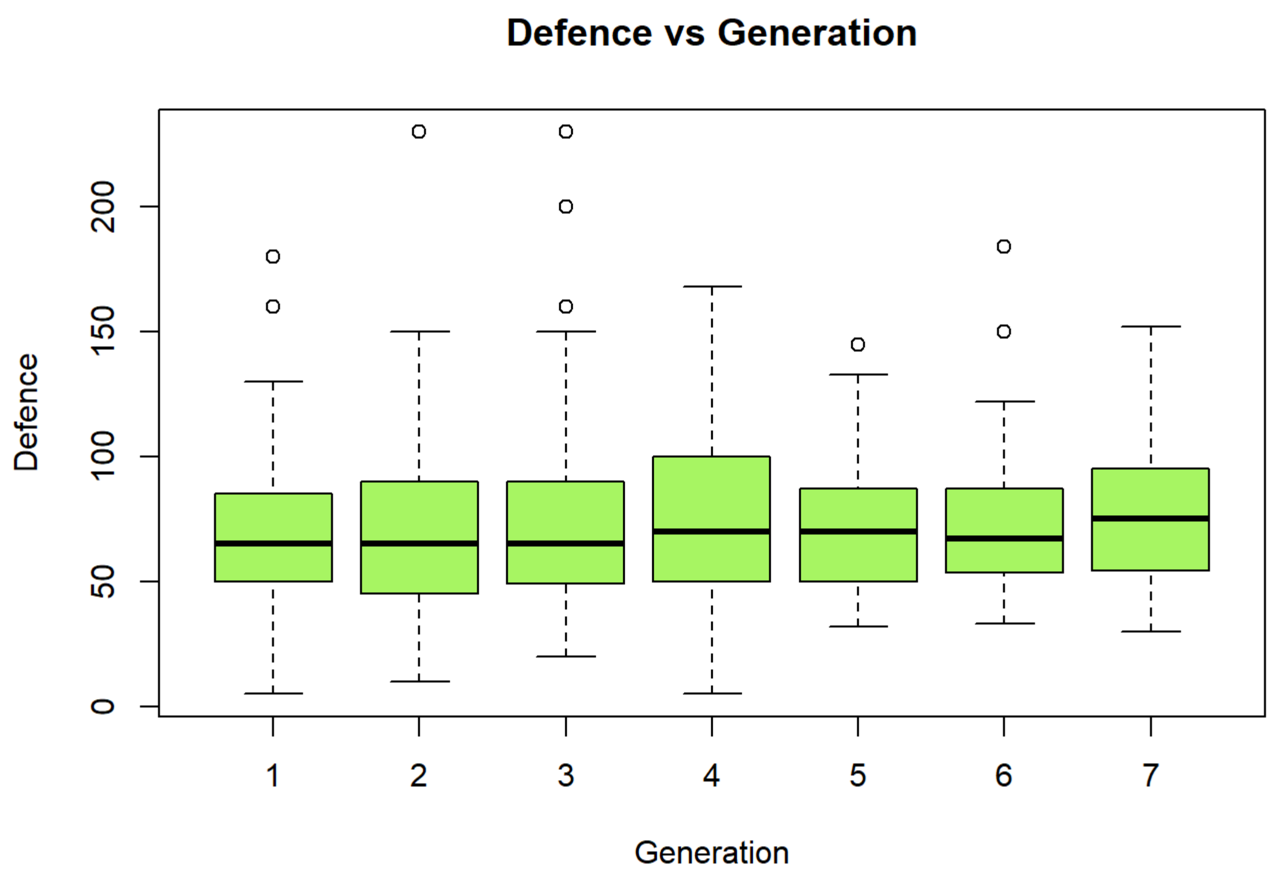
```
par(mfrow=c(1,1))
boxplot(pokemon$base_total~pokemon$generation, main="Base total vs Generation", xlab = "Generati
on", ylab = "Basetotal", col= "#a7f562")
```



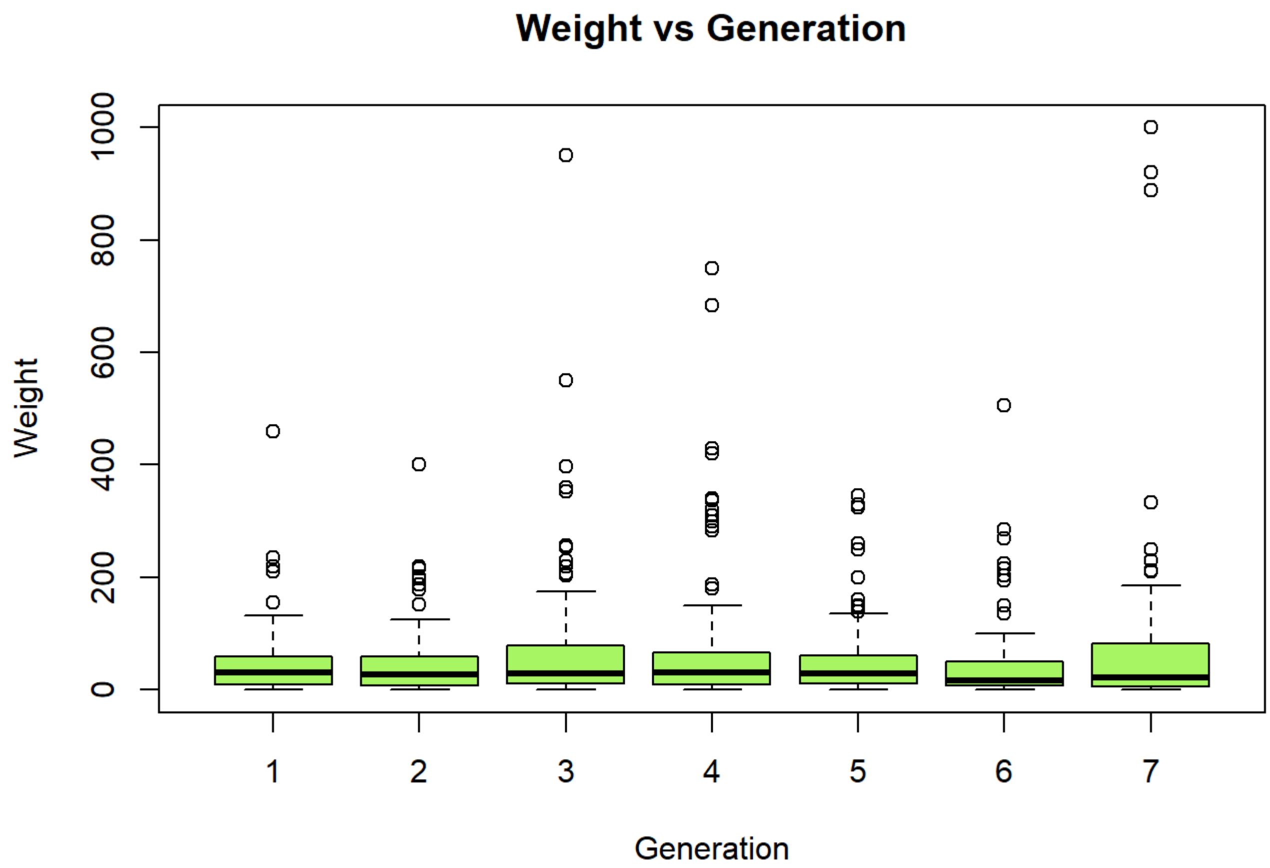

```
boxplot(pokemon$attack~pokemon$generation, main="Attack vs Generation", xlab = "Generation", yla
b = "Attack", col= "#a7f562")
```



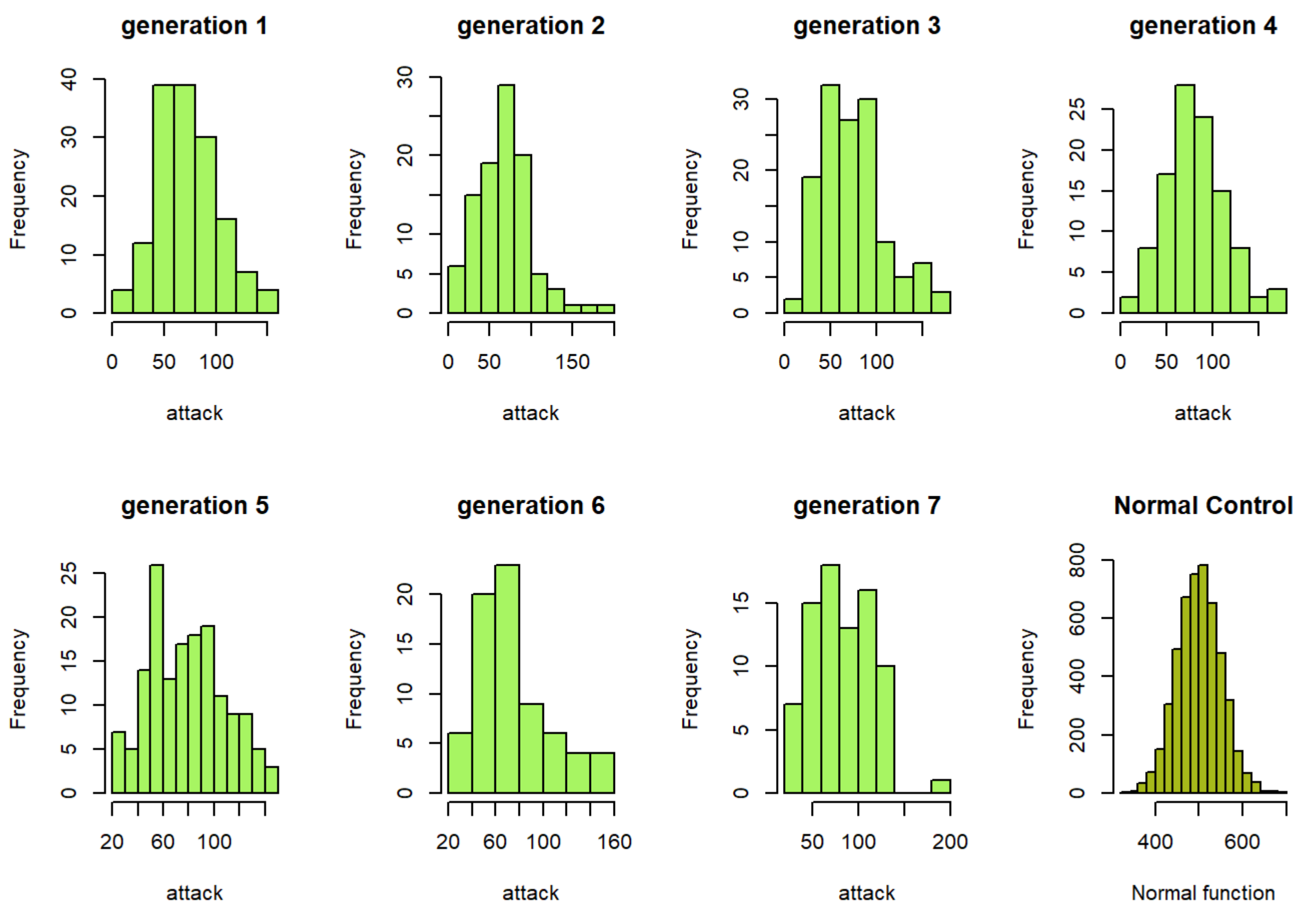

```
boxplot(pokemon$defense~pokemon$generation, main="Defence vs Generation", xlab = "Generation", y
lab = "Defence", col= "#a7f562")
```



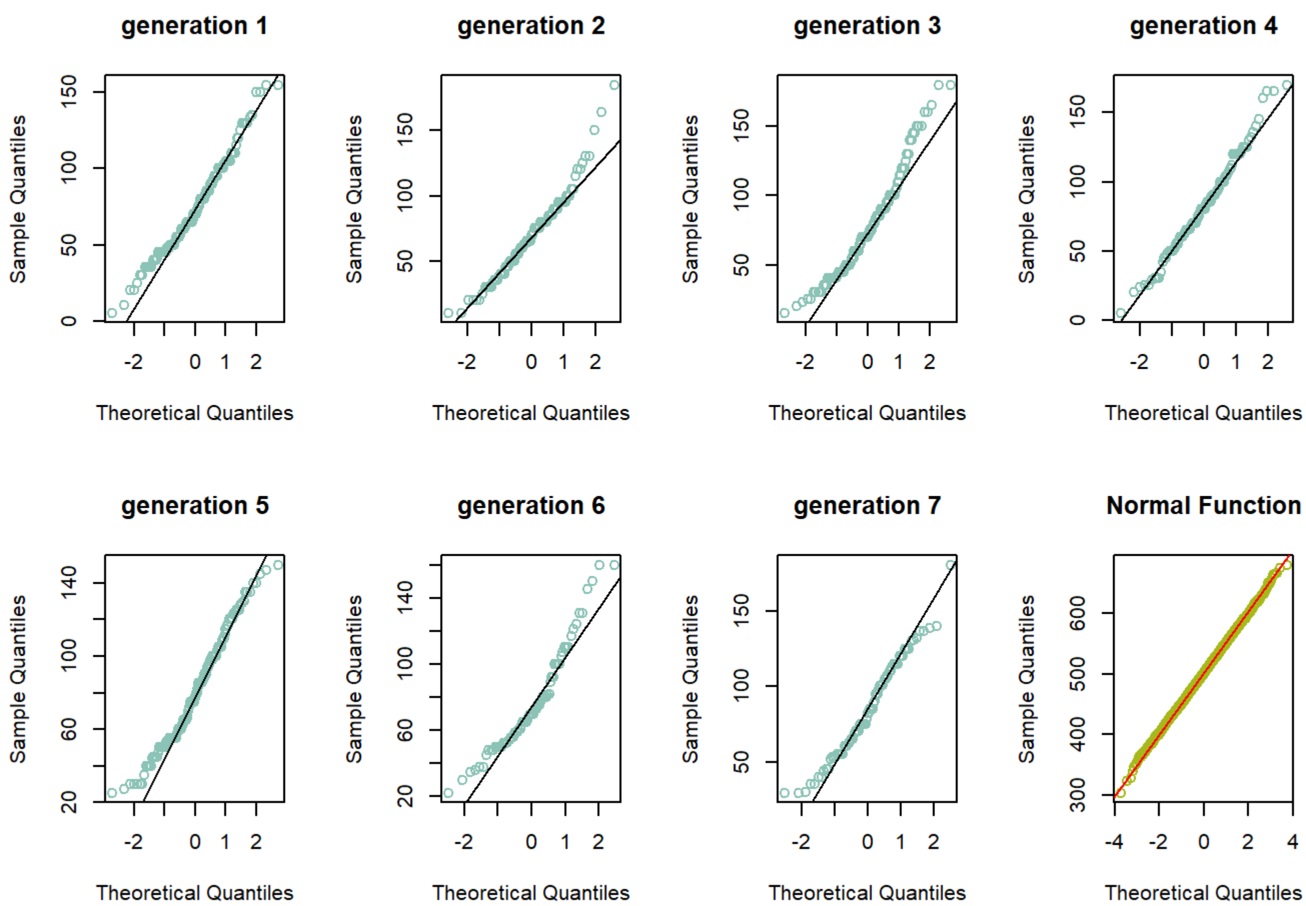

```
boxplot(pokemon$weight_kg~pokemon$generation, main="Weight vs Generation", xlab = "Generation",
 ylab = "Weight", col= "#a7f562")
```




```
par(mfrow=c(2,4))
hist(as.numeric(generation1$attack), main = "generation 1", col = "#a7f562", xlab = "attack")

hist(as.numeric(generation2$attack), main = "generation 2", col = "#a7f562", xlab = "attack")

hist(as.numeric(generation3$attack), main = "generation 3", col = "#a7f562", xlab = "attack")

hist(as.numeric(generation4$attack), main = "generation 4", col = "#a7f562", xlab = "attack")

hist(as.numeric(generation5$attack), main = "generation 5", col = "#a7f562", xlab = "attack")

hist(as.numeric(generation6$attack), main = "generation 6", col = "#a7f562", xlab = "attack")

hist(as.numeric(generation7$attack), main = "generation 7", col = "#a7f562", xlab = "attack")

hist(rnorm(5000,mean=500,sd=50), main = "Normal Control",  col="#A6B91A", xlab = "Normal functio
n")
```




```
par(mfrow=c(2,4))
qqnorm(generation1$attack, col = "#8BC3B6", main = "generation 1" )
qqline(generation1$attack)

qqnorm(generation2$attack, col = "#8BC3B6", main = "generation 2" )
qqline(generation2$attack)

qqnorm(generation3$attack, col = "#8BC3B6", main = "generation 3" )
qqline(generation3$attack)

qqnorm(generation4$attack, col = "#8BC3B6", main = "generation 4" )
qqline(generation4$attack)

qqnorm(generation5$attack, col = "#8BC3B6", main = "generation 5" )
qqline(generation5$attack)

qqnorm(generation6$attack, col = "#8BC3B6", main = "generation 6" )
qqline(generation6$attack)

qqnorm(generation7$attack, col = "#8BC3B6", main = "generation 7" )
qqline(generation7$attack)

qqnorm(rnorm(5000,mean=500,sd=50),  col="#A6B91A", main = "Normal Function")
qqline(rnorm(5000,mean=500,sd=50), col= "red")
```



shaphiro test

First we will perform **Shapiro-Wilk Normality Test** to check whether the attack of the pokemon has normal distribution in each generation.

- Null Hypothesis, $H_0 :=$ attack is normally distributed
- Alternate Hypothesis, $H_a :=$ attack is **NOT** normally distributed

```
shapiro.test(generation1$attack)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  generation1$attack
## W = 0.9796, p-value = 0.0242
```

```
shapiro.test(generation2$attack)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  generation2$attack
## W = 0.96352, p-value = 0.007225
```

```
shapiro.test(generation3$attack)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  generation3$attack
## W = 0.95102, p-value = 9.94e-05
```

```
shapiro.test(generation4$attack)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  generation4$attack
## W = 0.98645, p-value = 0.3537
```

```
shapiro.test(generation5$attack)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  generation5$attack
## W = 0.97522, p-value = 0.006524
```

```
shapiro.test(generation6$attack)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  generation6$attack
## W = 0.93364, p-value = 0.00092
```

```
shapiro.test(generation7$attack)
```

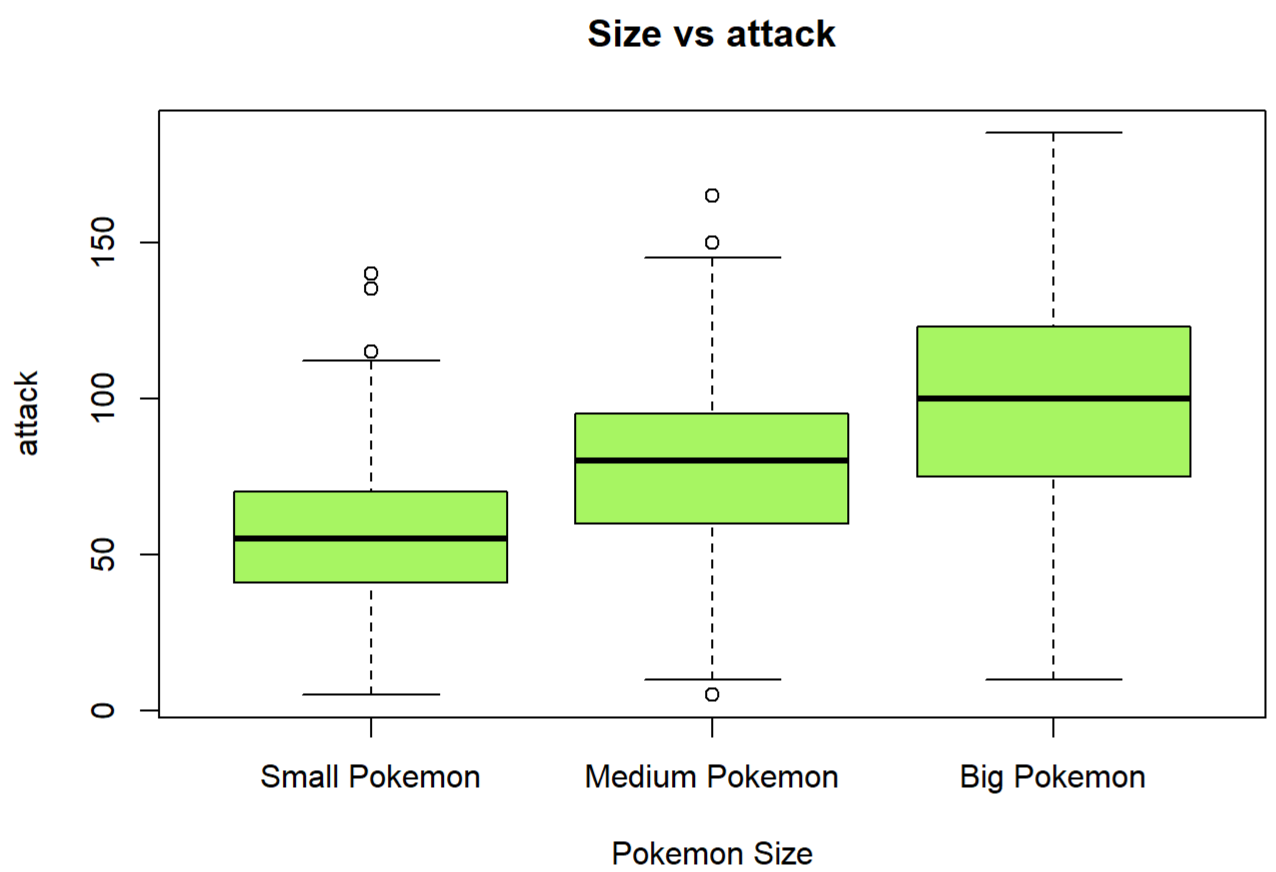
```
##
##  Shapiro-Wilk normality test
##
## data:  generation7$attack
## W = 0.97275, p-value = 0.08506
```

- attack of generation 1 is normally distributed
- attack of generation 2 is not normally distributed
- attack of generation 3 is not normally distributed
- attack of generation 4 is normally distributed
- attack of generation 5 is not normally distributed
- attack of generation 6 is not normally distributed

- attack of generation 7 is not normally distributed

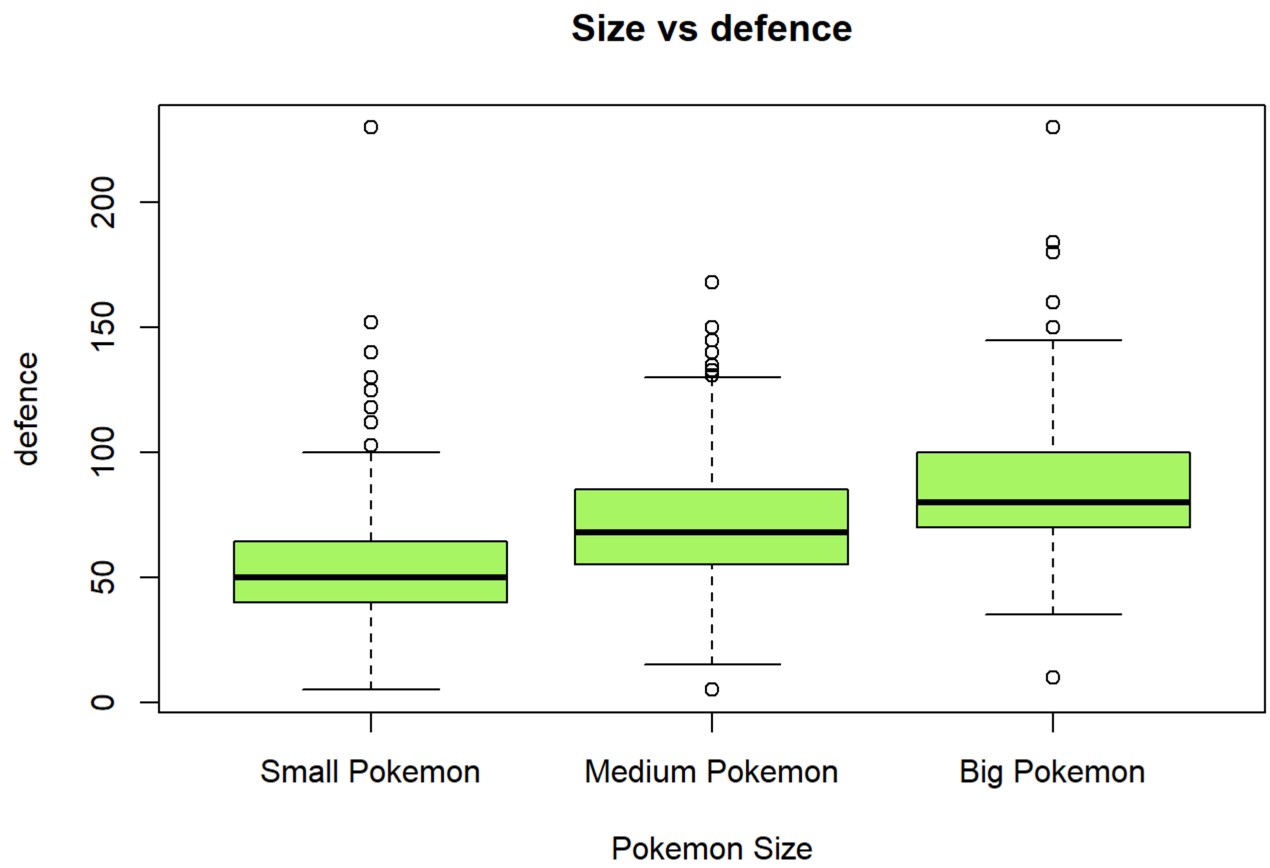
hypothesis correlation in size with attack and defence

```
boxplot(as.numeric(small$attack), as.numeric(mid$attack), as.numeric(big$attack),col = "#a7f562"
, main="Size vs attack", names = c("Small Pokemon", "Medium Pokemon", "Big Pokemon"), xlab="Poke
mon Size", ylab="attack")
```



- bigger pokemon has more attack

```
boxplot(as.numeric(small$defense), as.numeric(mid$defense), as.numeric(big$defense),col = "#a7f5
62", main="Size vs defence", names = c("Small Pokemon", "Medium Pokemon", "Big Pokemon"), xlab=
"Pokemon Size", ylab="defence")
```



- bigger pokemon has more defense

Wilcoxon test

Null Hypothesis: median of attack of Big pokemons is equal to mid. *Alternate Hypothesis:* median of attack of Big pokemons is greaterthan mid.

```
wilcox.test(mid$attack ,big$attack ,alternative = "greater")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  mid$attack and big$attack
## W = 12047, p-value = 1
## alternative hypothesis: true location shift is greater than 0
```

- attack of big pokemon is more

Wilcoxon test

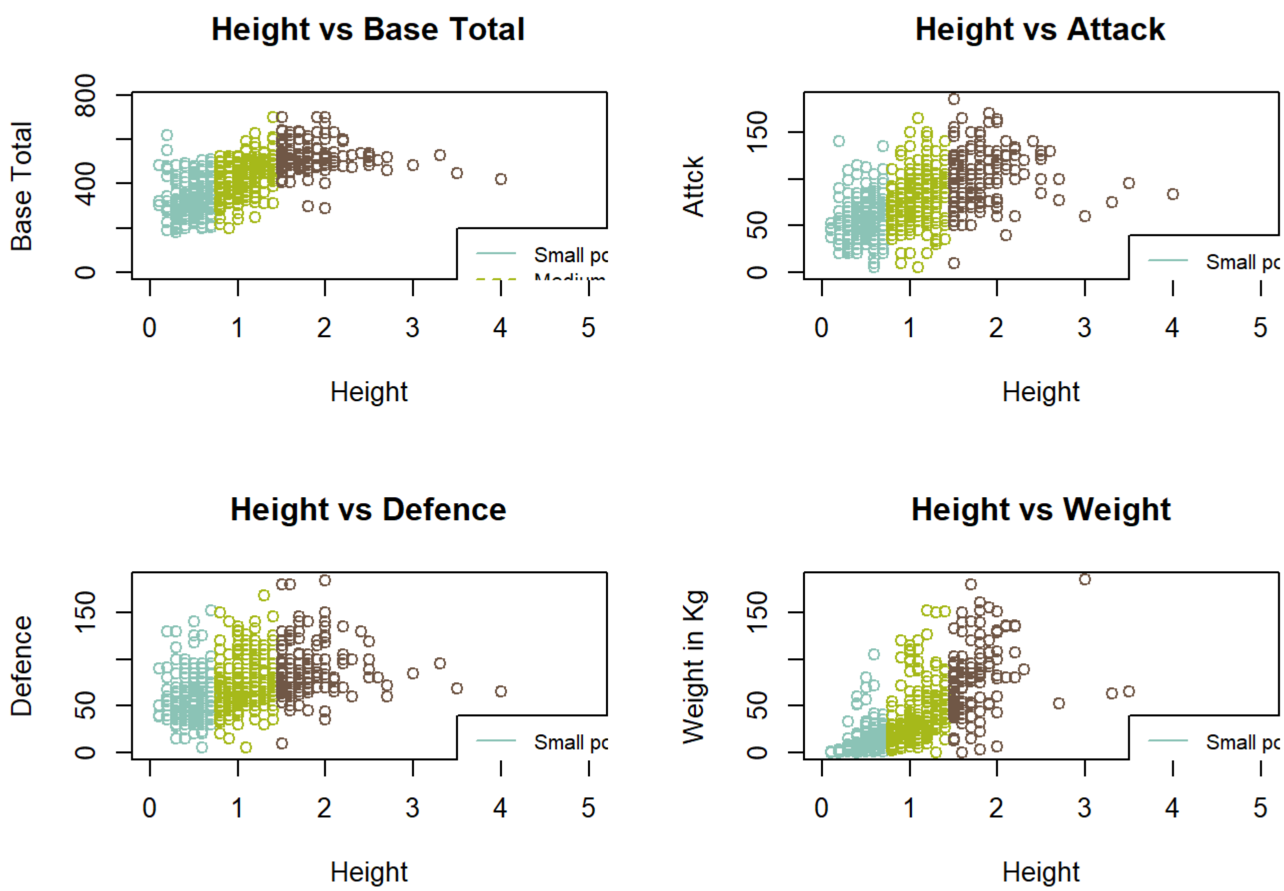
Null Hypothesis: median of defense of Big pokemons is equal to mid. *Alternate Hypothesis:* median of defense of Big pokemons is greater than mid.

```
wilcox.test(mid$defense ,big$defense ,alternative = "greater")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  mid$defense and big$defense
## W = 12492, p-value = 1
## alternative hypothesis: true location shift is greater than 0
```

- defence of big pokemon is more

```
par(mfrow=c(2,2))

plot(small$height_m, small$base_total,col="#8BC3B6", ylim = c(0, max(pokemon$base_total)), xlim
 = c(0, 5), , xlab = "Height", ylab = "Base Total", main = "Height vs Base Total")
points(mid$height_m, mid$base_total, col="#A6B91A")
points(big$height_m, big$base_total, col="#705746")
legend(3.5, 200, legend=c("Small pokemons", "Medium pokemon", "Big pokemon"),
       col=c("#8BC3B6", "#A6B91A","#705746"), lty=1:2, cex=0.8)

plot(small$height_m, small$attack,col="#8BC3B6", ylim = c(0, max(pokemon$attack)), xlim = c(0, 5
), xlab = "Height", ylab = "Attck", main = "Height vs Attack")
points(mid$height_m, mid$attack, col="#A6B91A")
points(big$height_m, big$attack, col="#705746")
legend(3.5, 40, legend=c("Small pokemons", "Medium pokemon", "Big pokemon"),
       col=c("#8BC3B6", "#A6B91A","#705746"), lty=1:2, cex=0.8)

plot(small$height_m, small$defense ,col="#8BC3B6", ylim = c(0, max(pokemon$attack)), xlim = c(0,
5),, xlab = "Height", ylab = "Defence", main = "Height vs Defence")
points(mid$height_m, mid$defense, col="#A6B91A")
points(big$height_m, big$defense, col="#705746")
legend(3.5, 40, legend=c("Small pokemons", "Medium pokemon", "Big pokemon"),
       col=c("#8BC3B6", "#A6B91A","#705746"), lty=1:2, cex=0.8)


plot(small$height_m, small$weight_kg,col="#8BC3B6", ylim = c(0, max(pokemon$attack)), xlim = c(0
, 5), xlab = "Height", ylab = "Weight in Kg", main = "Height vs Weight")
points(mid$height_m, mid$weight_kg, col="#A6B91A")
points(big$height_m, big$weight_kg, col="#705746")
legend(3.5, 40, legend=c("Small pokemons", "Medium pokemon", "Big pokemon"),
       col=c("#8BC3B6", "#A6B91A","#705746"), lty=1:2, cex=0.8)
```



correlation test

```
cor.test(pokemon$height_m, pokemon$attack)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pokemon$height_m and pokemon$attack
## t = 13.035, df = 779, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3638075 0.4790907
## sample estimates:
##       cor
## 0.4231602
```

```
cor.test(pokemon$height_m, pokemon$weight_kg)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pokemon$height_m and pokemon$weight_kg
## t = 22.438, df = 779, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5819789 0.6673701
## sample estimates:
##       cor
## 0.6265511
```

```
cor.test(pokemon$height_m, pokemon$defense)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pokemon$height_m and pokemon$defense
## t = 10.837, df = 779, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2993865 0.4213907
## sample estimates:
##       cor
## 0.3619375
```

```
cor.test(pokemon$height_m, pokemon$base_total)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pokemon$height_m and pokemon$base_total
## t = 17.677, df = 779, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4830419 0.5833202
## sample estimates:
##       cor
## 0.5350631
```

- Cor Values are positive for all the data so we can say With Increase in height Base total, Attack, Defence and Weight also increases