Análisis Exploratorio de Datos y gráficos depurados

Portada

CI-0131 Diseño de Experimentos

Laboratorio 2: Análisis Exploratorio de Datos y gráficos depurados

Estudiantes:

- [Milton]
- [Archibald Emmanuel Carrion Claeys, C01736]

Para el Profesor:

Nota: _____

Comentarios Generales:

_____ _____

A. Primera parte

A.1. Información general

Empezamos realizando un lectura de los datos. En este caso, el dataset es un archivo CSV que contiene información sobre Pokémon.

```
df <- (read.csv(file.choose(), header = TRUE, encoding = "UTF-8"))
attach(df)

# Resumen informativo de los datos - tendencias
summary(df)
```

Podemos conseguir información general sobre el dataset usando str() y glimpse().

```
# Información básica
str(df)
# glimpse() es una función del paquete dplyr que proporciona una
# vista rápida de los datos
library(dplyr)
glimpse(df)

# adicionalmente tambien existe summary() que nos da un resumen de las
# variables, como  cuartiles  y  datos  máximos  y mínimos
summary(df)
```

No se agregaron las salidas de los 2 chunks anteriores, ya que son muy extensas, y pueden facilmente ser consultadas en el archivo csv adjunto. Algunos de los datos mas valiosos que se agregara al reporte son los siguientes - attack - defense - hp - weight_kg - height_m

```
summary(df$attack)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   55.00   75.00   77.86  100.00  185.00
```

```
summary(df$defense)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   50.00   70.00   73.01   90.00  230.00
```

```r
summary(df$hp)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   50.00   65.00   68.96   80.00  255.00
```

```r
summary(df$weight_kg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.10    9.00   27.30   61.38   64.80  999.90      20
```

```r
summary(df$height_m)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.100   0.600   1.000   1.164   1.500  14.500      20
```

Las variables categoricas se pueden obtener usando la función table() o count(). En R, una variable categórica es aquella que puede tomar un número limitado de valores distintos, representando categorías o grupos.

```r
# Variables categoricas
table(df$type1)
```

```
##
##      bug     dark   dragon electric    fairy fighting     fire   flying
##       72       29       27       39       18       28       52        3
##    ghost    grass   ground      ice   normal   poison  psychic     rock
##       27       78       32       23      105       32       53       45
##    steel    water
##       24      114
```

```r
table(df$type2)
```

```
##
##                bug     dark   dragon electric    fairy fighting     fire
##      384         5       21       17        9       29       25       13
##   flying    ghost    grass   ground      ice   normal   poison  psychic
##       95       14       20       34       15        4       34       29
##     rock    steel    water
##       14       22       17
```

```r
count(df, type1)
```

```
##       type1   n
## 1       bug  72
## 2      dark  29
## 3    dragon  27
## 4  electric  39
## 5     fairy  18
## 6  fighting  28
## 7      fire  52
## 8    flying   3
## 9     ghost  27
## 10    grass  78
## 11   ground  32
## 12      ice  23
## 13   normal 105
## 14   poison  32
## 15  psychic  53
## 16     rock  45
## 17    steel  24
## 18    water 114
```

```r
count(df, type2)
```

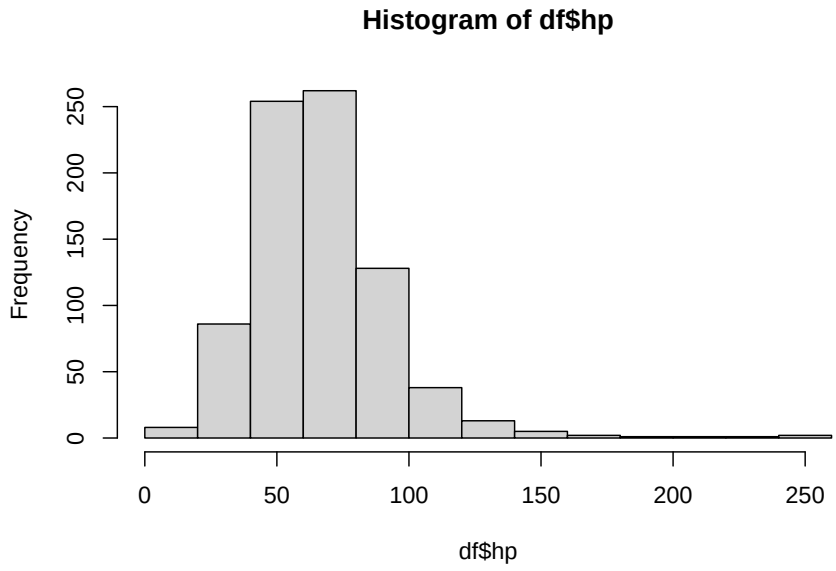
```
##       type2   n
## 1           384
## 2       bug   5
## 3      dark  21
## 4    dragon  17
## 5  electric   9
## 6     fairy  29
## 7  fighting  25
## 8      fire  13
## 9    flying  95
## 10    ghost  14
## 11    grass  20
## 12   ground  34
## 13      ice  15
## 14   normal   4
## 15   poison  34
## 16  psychic  29
## 17     rock  14
## 18    steel  22
## 19    water  17
```

A.2. Histogramas

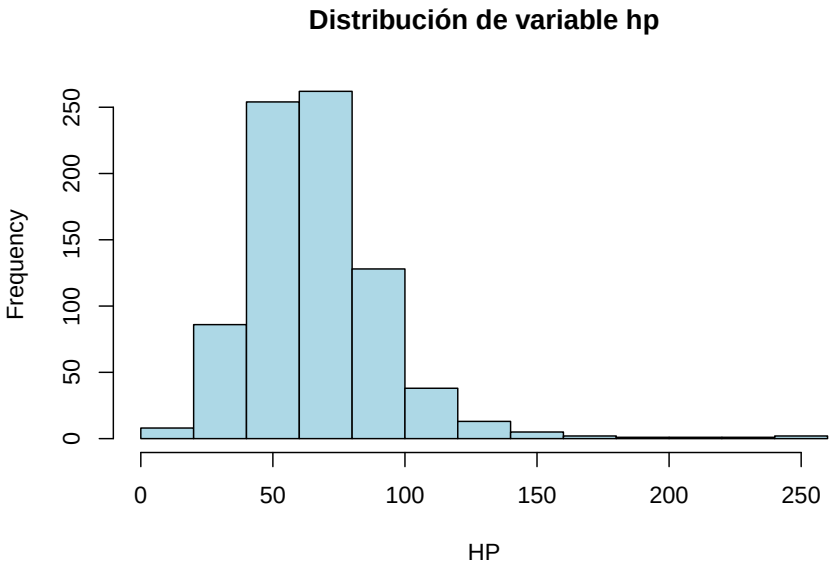
Un histograma es una representación gráfica de la distribución de un conjunto de datos que muestra la frecuencia de los valores en intervalos o "bins". Se usa para analizar la distribución de una variable continua, como la altura o el peso de los Pokémon.

Histograma de la variable hp

```r
# Por ejemplo, el siguiente código construye un histograma para la variable hp

hist(df$hp)
```

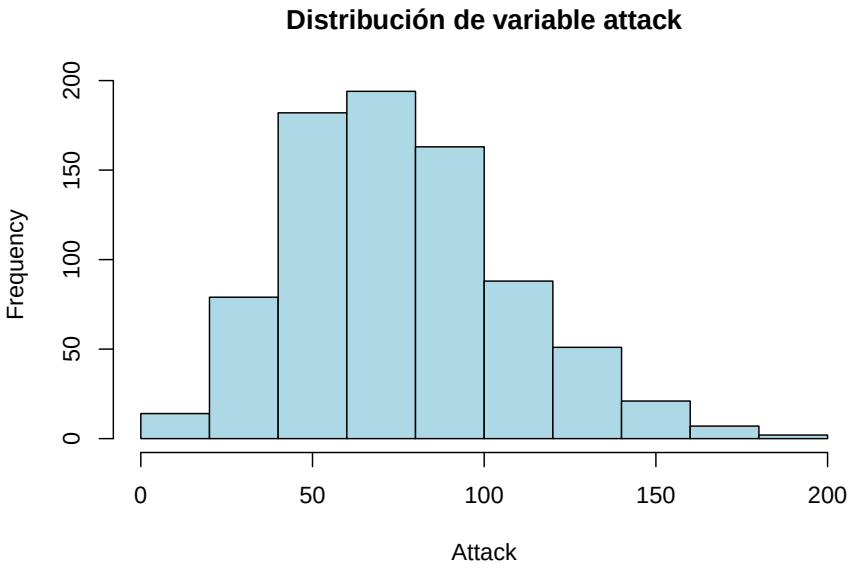



```r
# Se puede hacer un poco más claro agregando título y etiquetas:
hist(df$hp,
     main = "Distribución de variable hp",
     xlab = "HP",
     col = "lightblue",
     border = "black")
```



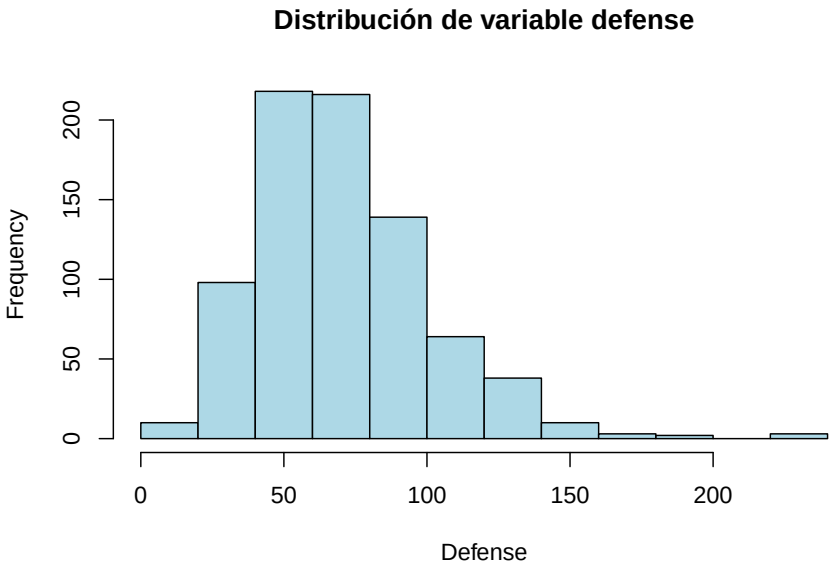
Histograma de la variable attack

```
# Histograma de la variable attack
hist(df$attack,
     main = "Distribución de variable attack",
     xlab = "Attack",
     col = "lightblue",
     border = "black")
```



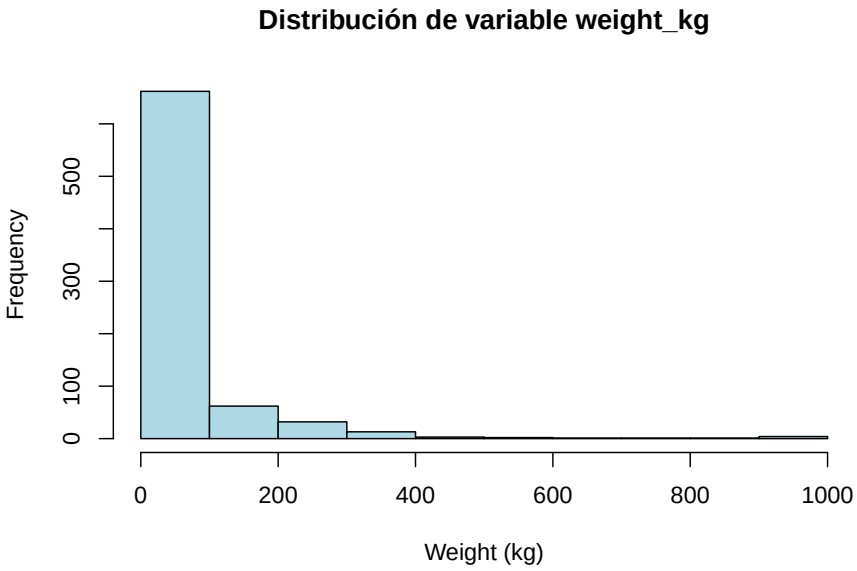
Histograma de la variable defense

```
# Histograma de la variable defense
hist(df$defense,
     main = "Distribución de variable defense",
     xlab = "Defense",
     col = "lightblue",
     border = "black")
```



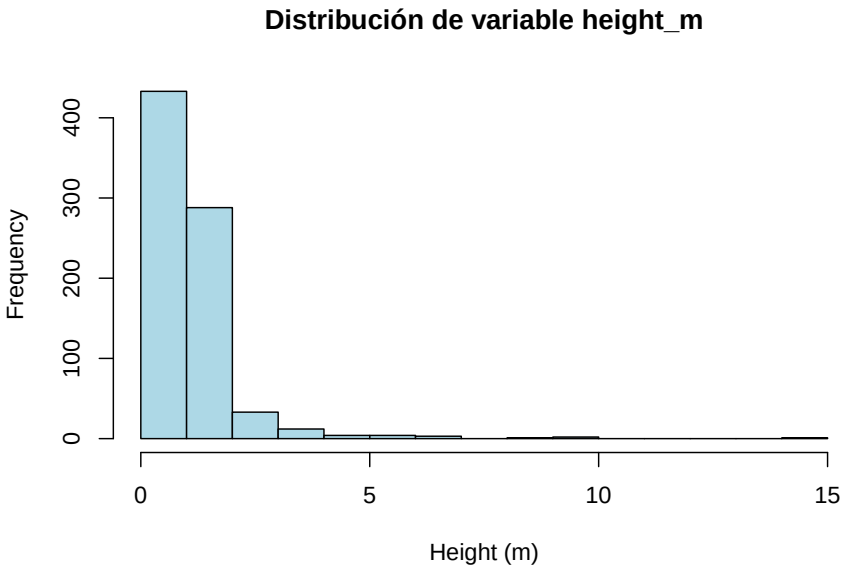
Histograma de la variable weight_kg

```
# Histograma de la variable weight_kg
hist(df$weight_kg,
     main = "Distribución de variable weight_kg",
     xlab = "Weight (kg)",
     col = "lightblue",
     border = "black")
```



Histograma de la variable height_m

```
# Histograma de la variable height_m
hist(df$height_m,
     main = "Distribución de variable height_m",
     xlab = "Height (m)",
     col = "lightblue",
     border = "black")
```

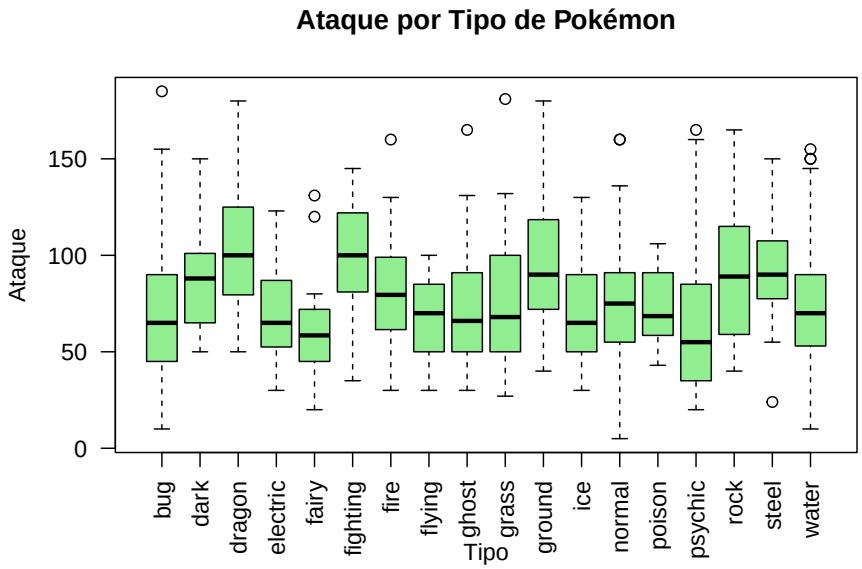


A.3. Boxplots (cajas y bigotes)

Un boxplot es una representación gráfica que muestra la distribución de un conjunto de datos a través de sus cuartiles. Los boxplots son útiles para identificar la presencia de valores atípicos (outliers) y para comparar la distribución de diferentes grupos.

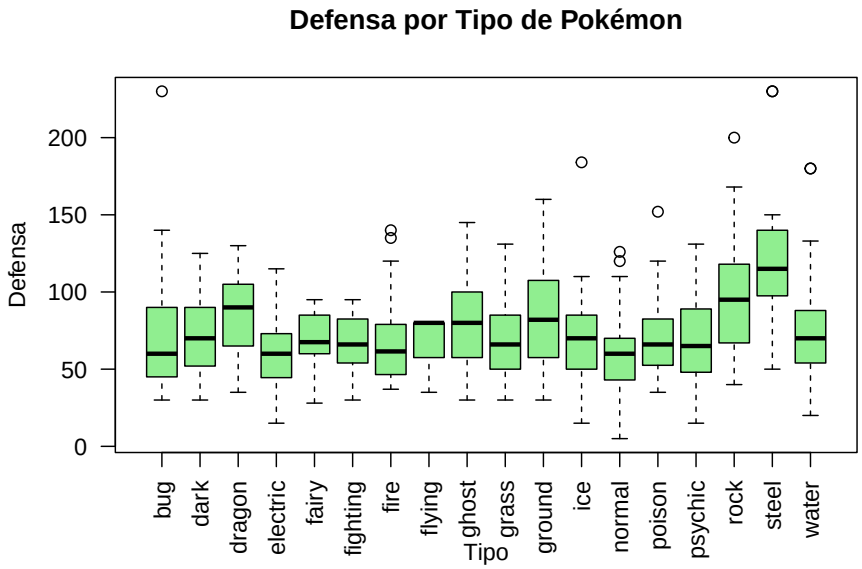
Boxplot de la variable attack

```
boxplot(attack ~ type1,
        data = df,
        main = "Ataque por Tipo de Pokémon",
        xlab = "Tipo",
        ylab = "Ataque",
        las = 2,
        col = "lightgreen")
```



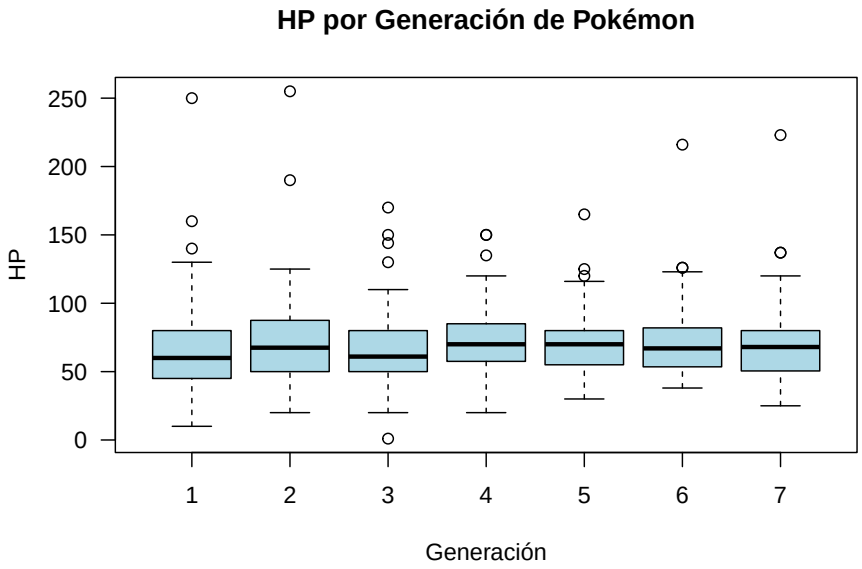
Boxplot de la variable defense por tipo de Pokémon

```
boxplot(defense ~ type1,
        data = df,
        main = "Defensa por Tipo de Pokémon",
        xlab = "Tipo",
        ylab = "Defensa",
        las = 2,
        col = "lightgreen")
```



Boxplot de la variable hp por generación de Pokémon

```
boxplot(hp ~ generation,
        data = df,
        main = "HP por Generación de Pokémon",
        xlab = "Generación",
        ylab = "HP",
        las = 1,
        col = "lightblue")
```

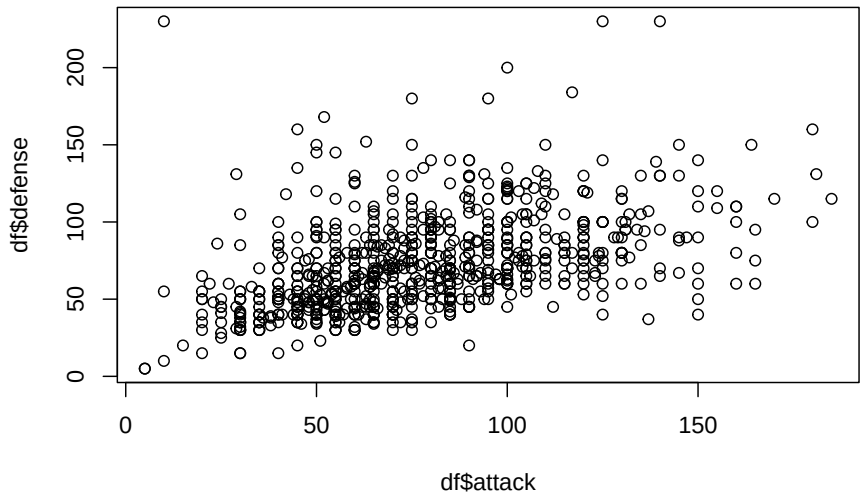


A.4. Gráficos de dispersión

Un gráfico de dispersión (scatter plot) es una representación gráfica que muestra la relación entre dos variables cuantitativas. Los puntos en el gráfico representan pares de valores de las dos variables, lo que permite observar patrones, tendencias y posibles correlaciones entre ellas, o en el caso contrario, si no existe relación entre las variables.

Gráfico de dispersión entre ataque y defensa

```
# Por ejemplo, el siguiente código construye un gráfico de dispersión
# entre attack y defense.
plot(df$attack,
     df$defense)
```




```r
# Se puede hacer un poco más claro agregando título y etiquetas:
plot(df$attack,
     df$defense,
     main = "Ataque vs Defensa",
     xlab = "Ataque",
     ylab = "Defensa",
     col = "darkred")
```

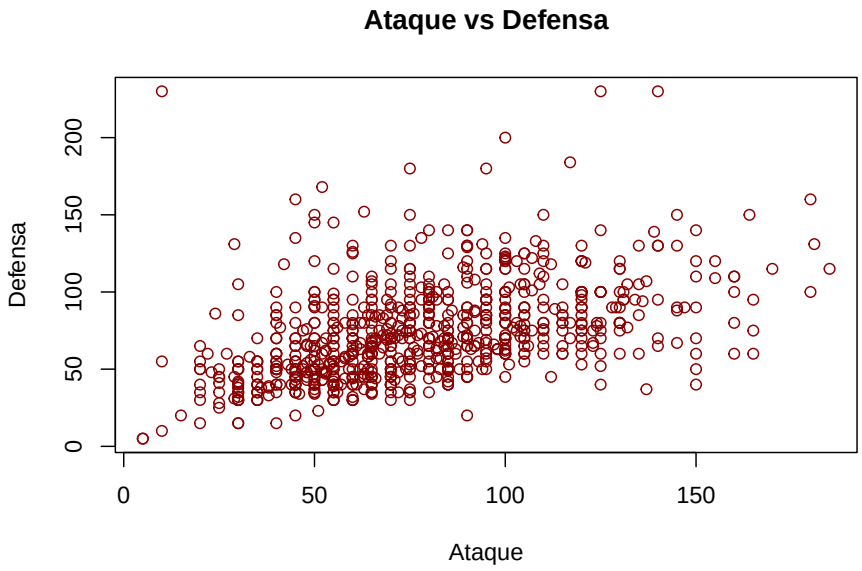


Gráfico de dispersión entre attack y hp

```
# Gráfico de dispersión entre attack y hp
plot(df$attack,
     df$hp,
     main = "Ataque vs HP",
     xlab = "Ataque",
     ylab = "HP",
     col = "darkblue")
```

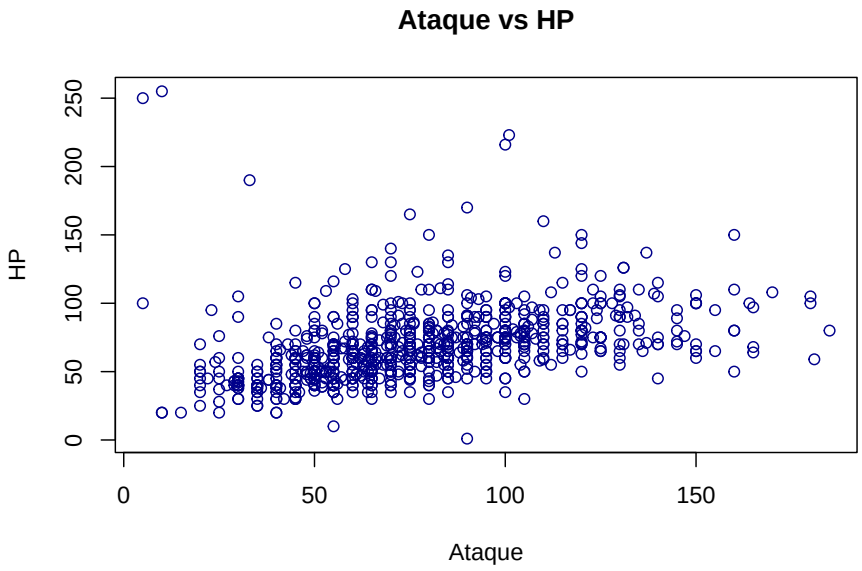
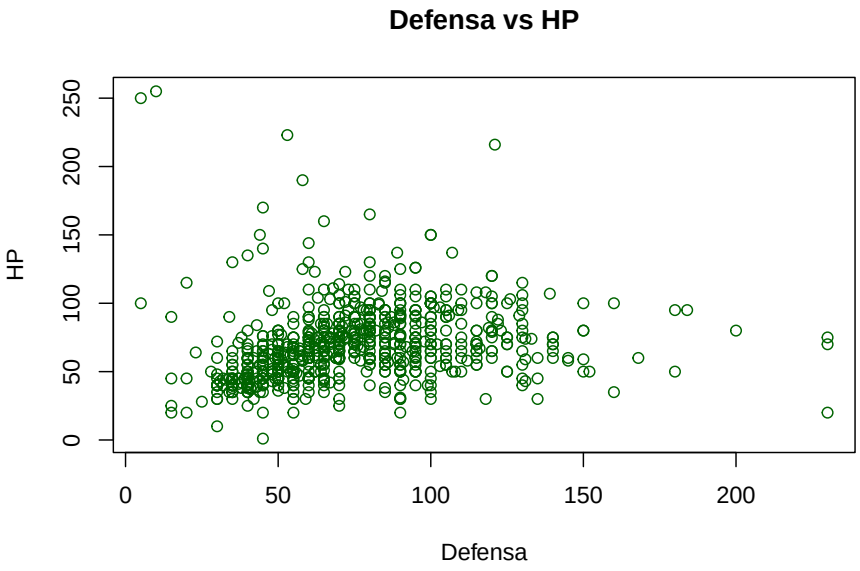


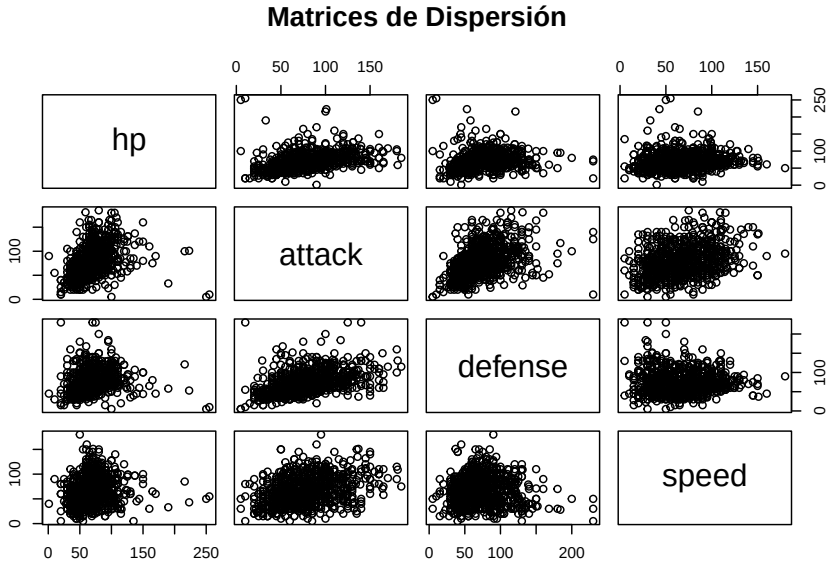
Gráfico de dispersión entre defense y hp

```
# Gráfico de dispersión entre defense y hp
plot(df$defense,
     df$hp,
     main = "Defensa vs HP",
     xlab = "Defensa",
     ylab = "HP",
     col = "darkgreen")
```



De la misma manera se pueden generar varios gráficos simultáneamente

```
# Matrices de dispersión dos a dos para hp, attack, defense y speed.
pairs(df[, c("hp", "attack", "defense", "speed")],
      main = "Matrices de Dispersión")
```



B. Segunda parte

instalación de librerías necesarias

Instalar los paquetes:

```
# instalar las siguientes librerías si no están instaladas
# install.packages(c("dplyr",
#                    "ggplot2",
#                    "gridExtra",
#                    "tidyr",
#                    "reshape2",
#                    "RColorBrewer",
#                    "ggrepel"))
```

Cargas los paquetes:

```
library(dplyr)
library(ggplot2)
library(gridExtra)
library(tidyr)
library(reshape2)
```

```r
library(RColorBrewer)
library(ggrepel)
```

Una timble en R es una versión mejorada de un data frame, que proporciona una forma más legible y fácil de trabajar con los datos. Las timbles son parte del paquete dplyr y están diseñadas para ser más eficientes y amigables para el usuario.

```r
df <- tibble::as_tibble(df)
colnames(df)[25] <- "classification"
df$capture_rate <- as.numeric(df$capture_rate)
head(df)
```

```
## # A tibble: 6 x 40
##   abilities             against_bug against_dark against_dragon against_elect
##   <chr>                       <dbl>        <dbl>          <dbl>            <d
## 1 ['Overgrow', 'Chloro~         1              1              1
## 2 ['Overgrow', 'Chloro~         1              1              1
## 3 ['Overgrow', 'Chloro~         1              1              1
## 4 ['Blaze', 'Solar Pow~         0.5            1              1
## 5 ['Blaze', 'Solar Pow~         0.5            1              1
## 6 ['Blaze', 'Solar Pow~         0.25           1              1
## # i 35 more variables: against_fairy <dbl>, against_fight <dbl>,
## #   against_fire <dbl>, against_flying <dbl>, against_ghost <dbl>,
## #   against_grass <dbl>, against_ground <dbl>, against_ice <dbl>,
## #   against_normal <dbl>, against_poison <dbl>, against_psychic <dbl>,
## #   against_rock <dbl>, against_steel <dbl>, against_water <dbl>, attack <int
## #   base_egg_steps <int>, base_happiness <int>, base_total <int>,
## #   capture_rate <dbl>, classification <chr>, defense <int>, ...
```

Como ese timble contiene muchos datos, solo usaramos un subconjunto de las columnas pra realizar nuestra EDA.

```r
df <- select(df, name, classification, hp, weight_kg,
             height_m, speed, attack, defense,
             sp_attack, sp_defense, type1, type2,
             abilities, generation, is_legendary,
             capture_rate)
head(df)
```

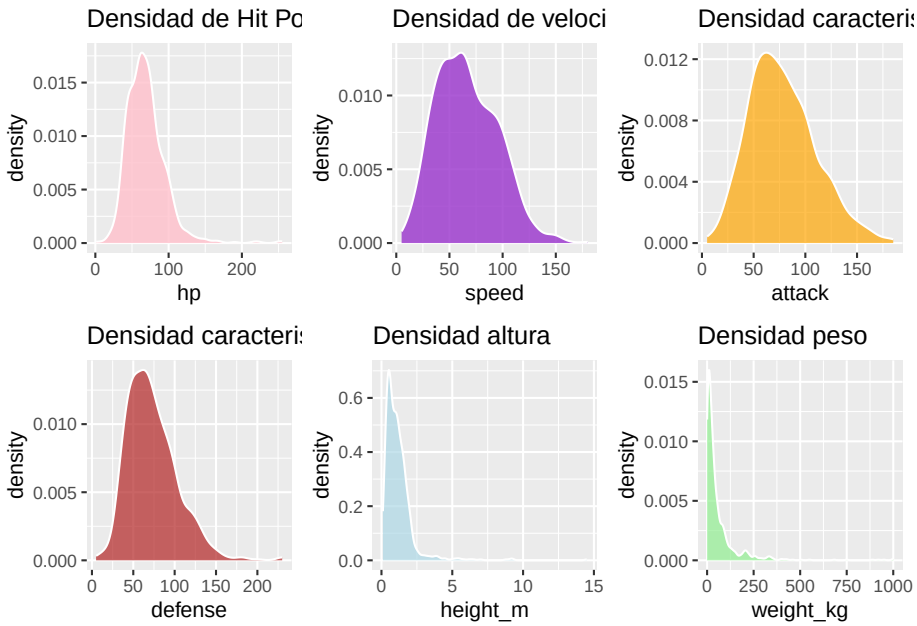
```
## # A tibble: 6 x 16
##   name    classification    hp weight_kg height_m speed attack defense sp_att
##   <chr>   <chr>          <int>     <dbl>    <dbl> <int>  <int>   <int>     <i
## 1 Bulbas~ Seed Pokémon      45       6.9      0.7    45     49      49
## 2 Ivysaur Seed Pokémon      60      13        1      60     62      63
```

```
## 3 Venusa~ Seed Pokémon      80     100        2      80    100     123
## 4 Charma~ Lizard Pokémon    39       8.5      0.6    65     52      43
## 5 Charme~ Flame Pokémon     58      19        1.1    80     64      58
## 6 Chariz~ Flame Pokémon     78      90.5      1.7   100    104      78
## # i 7 more variables: sp_defense <int>, type1 <chr>, type2 <chr>,
## #   abilities <chr>, generation <int>, is_legendary <int>, capture_rate <dbl>
```

B.1. Gráficos de densidad de varios atributos de Pokémon.

Un diagrama de densidad es una representación gráfica que muestra la distribución de una variable continua a través de una curva suave (suavizado del nucleo). Es útil para visualizar la forma de la distribución y la concentración de los datos en diferentes rangos de valores.

```
density_hp <- ggplot(data = df, aes(hp)) +
  geom_density(col = "white", fill = "pink", alpha = 0.8) +
  ggtitle("Densidad de Hit Points o Vida")

density_speed  <-  ggplot(data = df,  aes(speed)) +
  geom_density(col = "white",  fill = "darkorchid",  alpha = 0.8) +
  ggtitle("Densidad de velocidad")

density_attack  <-  ggplot(data = df,  aes(attack)) +
  geom_density(col = "white",  fill = "orange",  alpha = 0.7) +
  ggtitle("Densidad caracteristicas ofensivas")

density_defense  <- ggplot(data = df, aes(defense)) +
  geom_density(col = "white",  fill = "firebrick",  alpha = 0.7) +
  ggtitle("Densidad caracteristicas defensivas")

density_height  <- ggplot(data = df, aes(height_m)) +
  geom_density(col = "white",  fill = "lightblue",  alpha = 0.7) +
  ggtitle("Densidad altura")

density_weight  <- ggplot(data = df, aes(weight_kg)) +
  geom_density(col = "white",  fill = "lightgreen",  alpha = 0.7) +
  ggtitle("Densidad peso")


grid.arrange(density_hp, density_speed, density_attack,
             density_defense, density_height, density_weight,
             ncol = 3, nrow = 2)
```



B.2. Diagramas de Barras

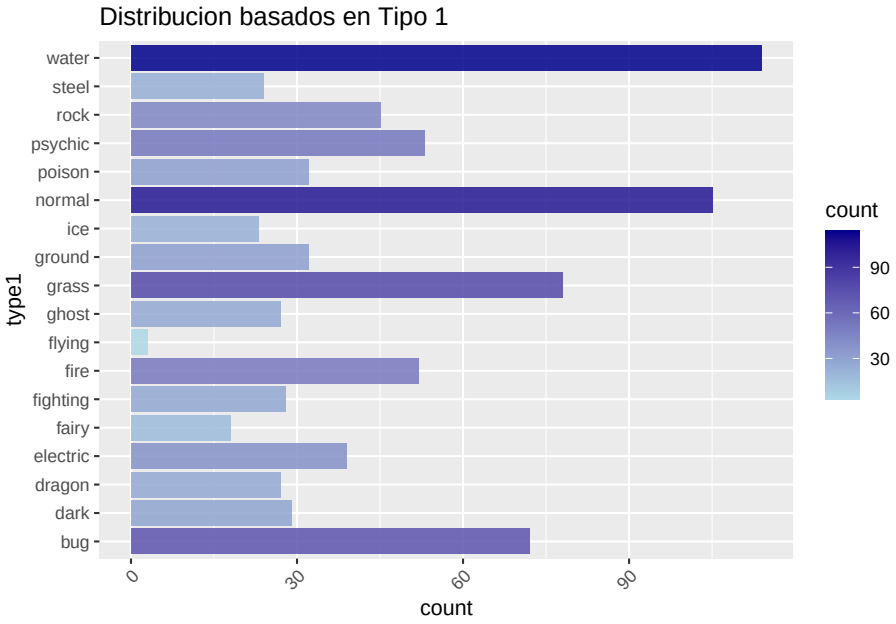
Un diagrama de barras es una representación gráfica que muestra la frecuencia o cantidad de diferentes categorías o grupos. Cada barra representa una categoría y su altura (o longitud) indica la cantidad o frecuencia de esa categoría. Es muy similar a un histograma, pero en lugar de mostrar la distribución de una variable continua, muestra la frecuencia de categorías discretas.

Número de Pokémon basado en su tipo primario (type1) y secundario (type2)

```
ggplot(data=df, aes(type1)) +
  geom_bar(aes(fill = ..count..), alpha = 0.85) +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  # Gradiente de colores
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Distribucion basados en Tipo 1") +
  coord_flip()
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
```

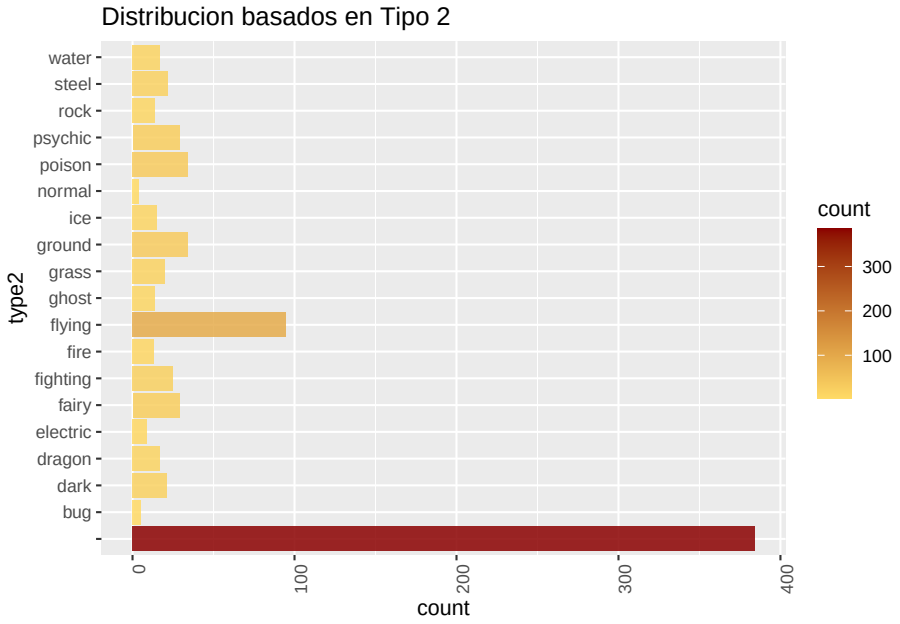
```
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Vemos que el tipo mas commun es Water, Normal, Grass y Bug.

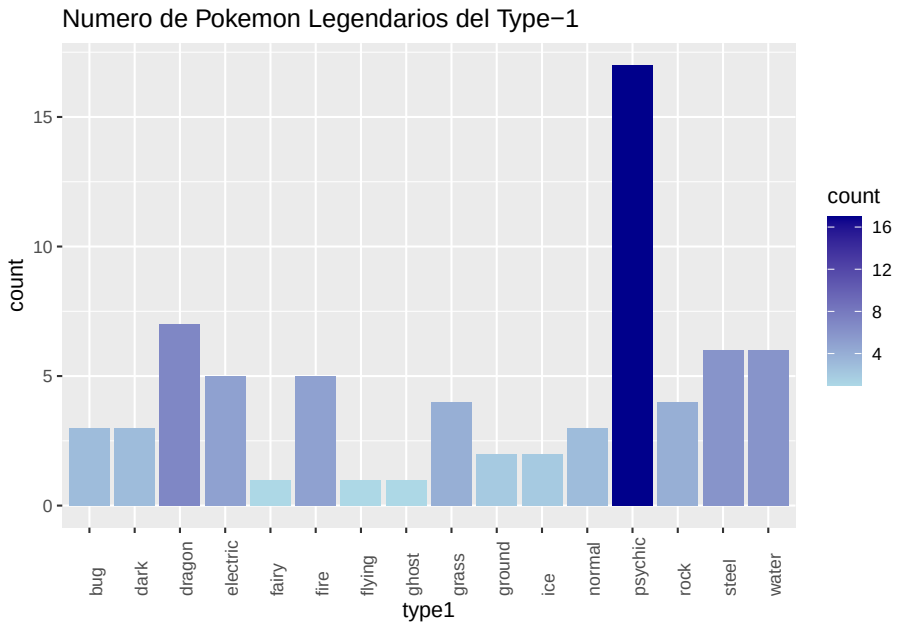
Diagrama de barras para el tipo secundario (type2)

```
ggplot(data = df, aes(type2)) +
  geom_bar(aes(fill = ..count..), alpha = 0.85) +
  scale_fill_gradient(low = "#ffdb67", high = "#8b0000") +
  # Gradiente de colores
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Distribucion basados en Tipo 2") +
  coord_flip()
```



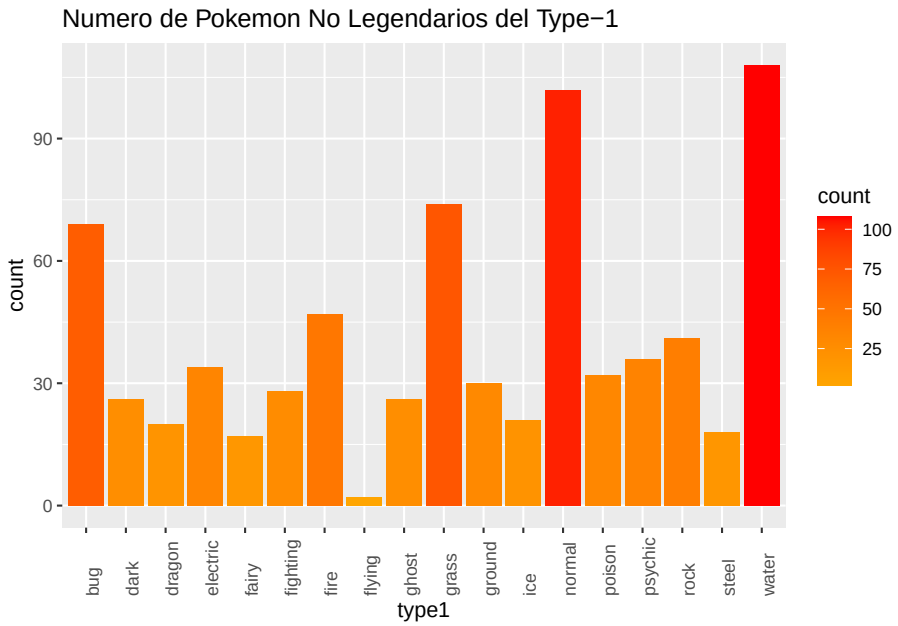
Número de Pokemon legendarios según su tipo primario (type1)

```
df %>%
  filter(is_legendary == 1) %>%
  ggplot(aes(type1)) +
  geom_bar(aes(fill = ..count..)) +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  # Gradiente de colores
  theme(axis.text.x = element_text(angle = 90, hjust = 0)) +
  ggtitle("Numero de Pokemon Legendarios del Type-1")
```



El siguiente gráfico muestra la cantidad de Pokémon legendarios según su tipo primario (type1).

```
df %>%
  filter(is_legendary == 0) %>%
  ggplot(aes(type1)) +
  geom_bar(aes(fill = ..count..)) +
  scale_fill_gradient(low = "orange", high = "red") +  # Gradiente de colores
  theme(axis.text.x = element_text(angle = 90, hjust = 0)) +
  ggtitle("Numero de Pokemon No Legendarios del Type-1")
```

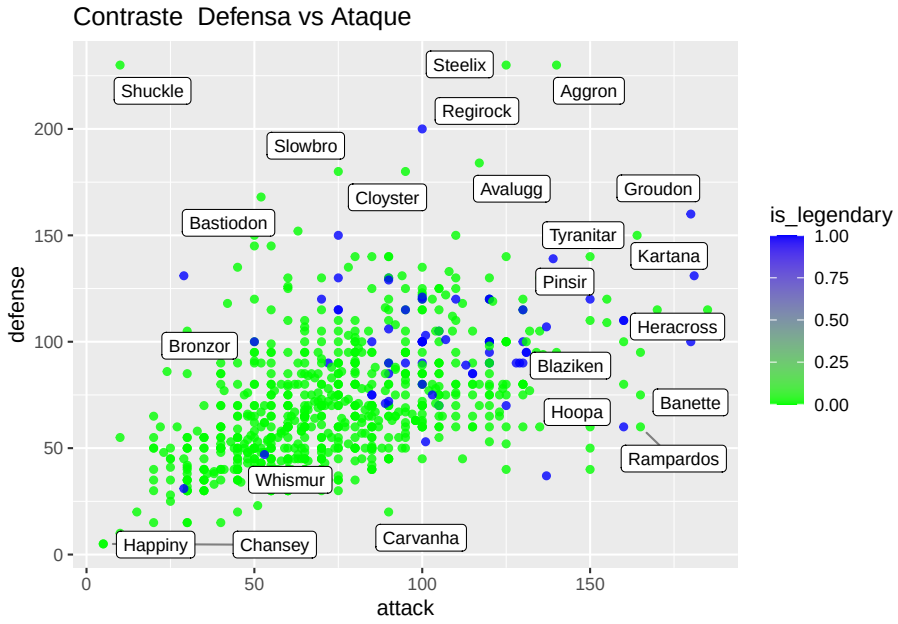


Este nuevo gráfico muestra la cantidad de Pokémon no legendarios según su tipo primario (type1). Vemos que el tipo más común es Water, Normal, Grass y Bug.

B.3. Gráfico de dispersión – Scatterplots

Ejemplo dado de un gráfico de dispersión entre ataque y defensa.

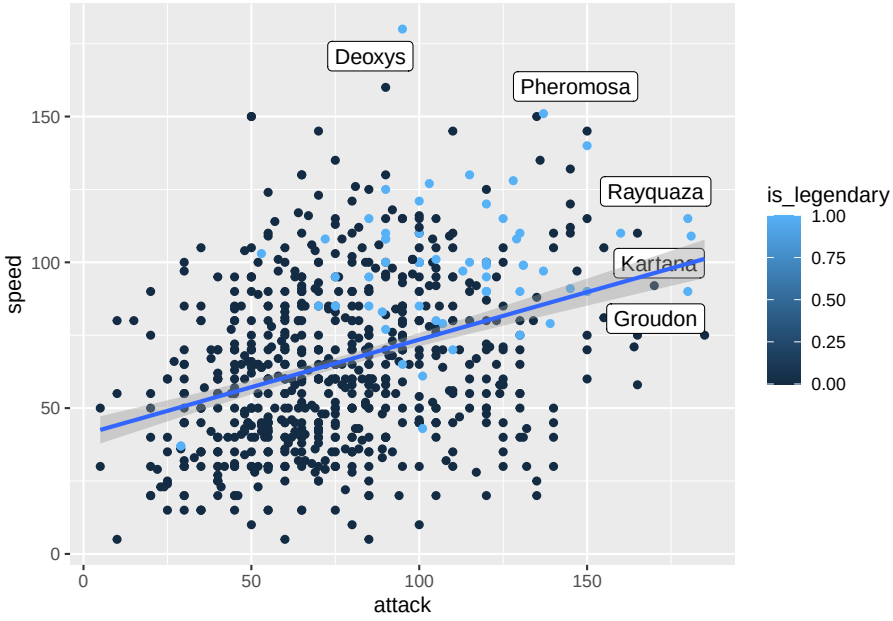
```
ggplot(data = df, aes(attack, defense)) +
  geom_point(aes(color=is_legendary), alpha = 0.8) +
  scale_color_gradient(low="green", high = "blue") +
  ggtitle("Contraste  Defensa vs Ataque") +
  geom_label_repel(data=subset(df,attack > 150 | defense > 160 | attack < 25 | d
  aes(label=name),
  box.padding = 0.35, point.padding = 0.5, size = 3,
  segment.color = 'grey50')
```



Ahora graficamos la dispersión entre attack y speed para los legendarios:

```
speed_attack_legendary <- ggplot(na.omit(df), aes(attack, speed)) +
  geom_point(aes(color=is_legendary)) +
  geom_label_repel(data=subset(df, (attack > 170 | speed > 150) & is_legendary =
                   box.padding = 0.35, point.padding = 0.5, segment.color = 'gre
  geom_smooth(method = "lm")

print(speed_attack_legendary)
```

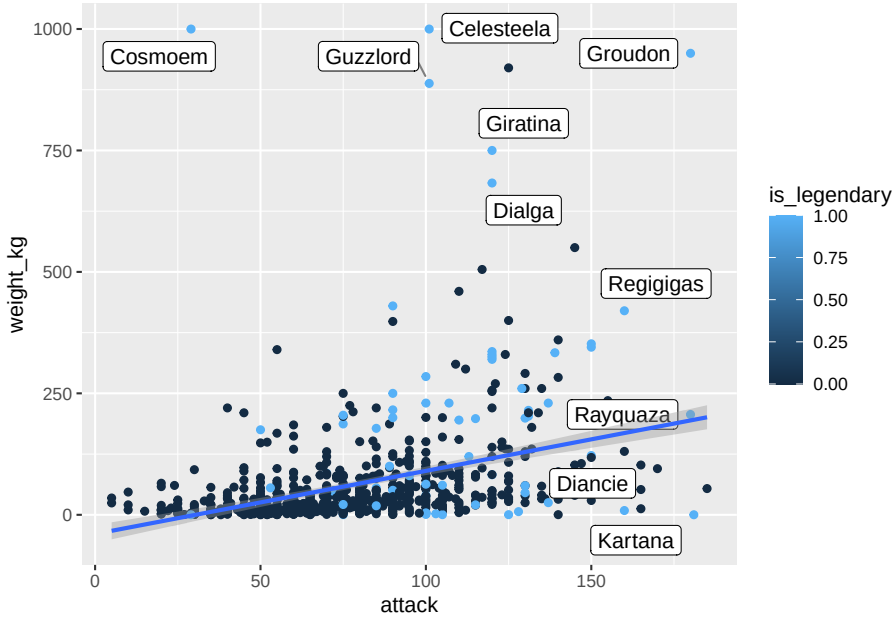
```
## `geom_smooth()` using formula = 'y ~ x'
```



Ahora graficamos un grafico similar pero para attack y weight_kg para los legendarios y etiquetamos a los pokemon con attack > 150 o weight_kg > 650:

```
weight_attack_legendary <- ggplot(na.omit(df), aes(attack, weight_kg)) +
  geom_point(aes(color=is_legendary)) +
  geom_label_repel(data=subset(df, (attack > 150 | weight_kg > 650) & is_legenda
                   box.padding = 0.35, point.padding = 0.5, segment.color = 'gre
  geom_smooth(method = "lm")

print(weight_attack_legendary)
```

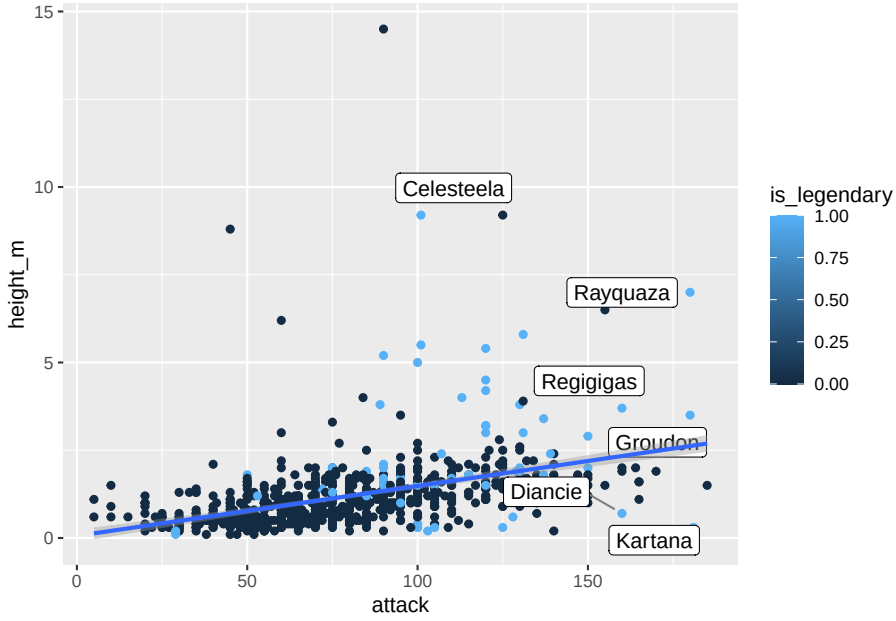
```
## `geom_smooth()` using formula = 'y ~ x'
```



Ahora graficamos un grafico similar pero para attack y height_m para los legendarios y etiquetamos a los pokemon con attack > 150 o height_m > 7.5:

```
height_attack_legendary <- ggplot(na.omit(df), aes(attack, height_m)) +
  geom_point(aes(color=is_legendary)) +
  geom_label_repel(data=subset(df, (attack > 150 | height_m > 7.5) & is_legendar
                   box.padding = 0.35, point.padding = 0.5, segment.color = 'gre
  geom_smooth(method = "lm")

print(height_attack_legendary)
```

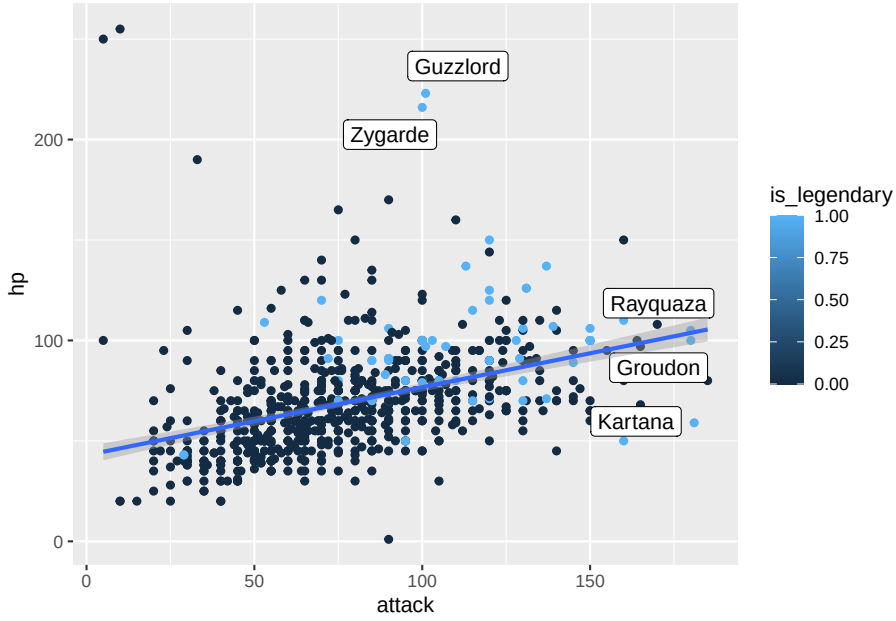
```
## `geom_smooth()` using formula = 'y ~ x'
```



Ahora graficamos un grafico similar pero para attack y hp para los legendarios y etiquetamos a los pokemon con attack > 170 o hp > 190:

```
hp_attack_legendary <- ggplot(na.omit(df), aes(attack, hp)) +
  geom_point(aes(color=is_legendary)) +
  geom_label_repel(data=subset(df, (attack > 170 | hp > 190) & is_legendary == 1
                   box.padding = 0.35, point.padding = 0.5, segment.color = 'gre
  geom_smooth(method = "lm")

print(hp_attack_legendary)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



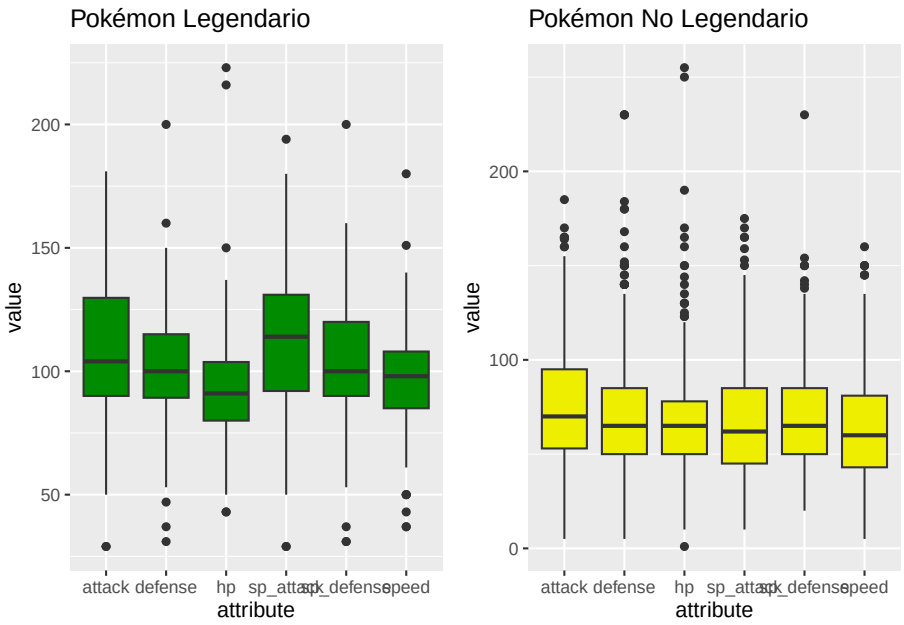
B.4. Diagramas de Caja o Boxplots

Un boxplot es una representación gráfica que muestra la distribución de un conjunto de datos a través de sus cuartiles. Los boxplots son útiles para identificar la presencia de valores atípicos (outliers) y para comparar la distribución de diferentes grupos.

Al analizar los boxplots vemos que exista mas valores atipicos en cuantos a atributos dentro de los pokemon no-leyendarios que en los legendarios. En amarillo tenemos los pokemon no legendarios y en verde los legendarios.

```
# Select relevant columns
box_plot_attr <- select(df, type1, is_legendary, hp, defense, attack, sp_attack,

# Separate legendary and non-legendary Pokémon
box_plot_attr_leg <- filter(box_plot_attr, is_legendary == 1)
box_plot_attr_nor <- filter(box_plot_attr, is_legendary == 0)

# Reshape to long format
box_plot_attr_leg_long <- gather(box_plot_attr_leg, attribute, value, -c(type1,
box_plot_attr_nor_long <- gather(box_plot_attr_nor, attribute, value, -c(type1,

# Create boxplots
bp_leg <- ggplot(data = box_plot_attr_leg_long, aes(attribute, value)) +
  geom_boxplot(fill = "green4") +
```

```
  ggtitle("Pokémon Legendario")

bp_nor <- ggplot(data = box_plot_attr_nor_long, aes(attribute, value)) +
  geom_boxplot(fill = "yellow2") +
  ggtitle("Pokémon No Legendario")

# Display plots side by side
grid.arrange(bp_leg, bp_nor, ncol = 2)
```



B.5. Mapas de calor

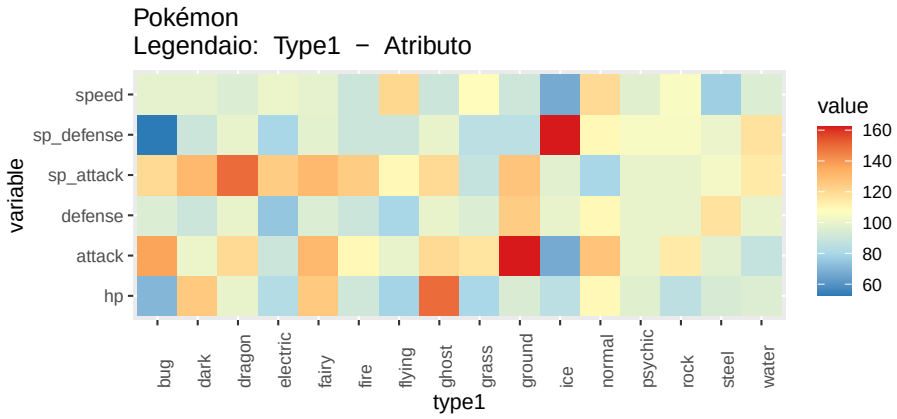
Un mapa de calor es una representación gráfica de datos en la que los valores se representan mediante colores. Los mapas de calor son útiles para visualizar patrones, tendencias y correlaciones en grandes conjuntos de datos.

Nuestro primer ejemplo se basa en analizar mas en detalle los pokemon tipo Ice:

```
hmap_attr <- select(df, type1, is_legendary, hp, defense, attack, sp_attack, sp_
hmap_attr_leg <- filter(hmap_attr, is_legendary == 1)
hmap_attr_leg <- group_by(hmap_attr_leg, type1)
hmap_attr_leg <- summarise(hmap_attr_leg, hp=median(hp), attack=median(attack),
defense=median(defense), sp_attack=median(sp_attack), sp_defense=median(sp_defen
speed=median(speed))
```

```
hmap_attr_leg_m <- melt(hmap_attr_leg)


## Using type1 as id variables
```

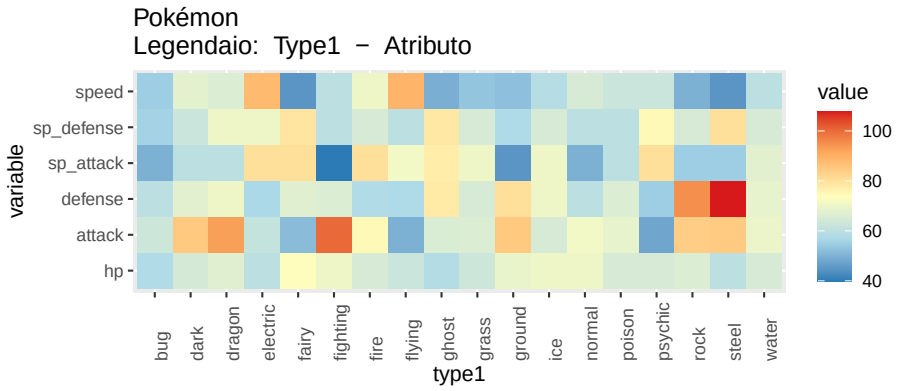
```
hm.palette <- colorRampPalette(rev(brewer.pal(5, 'RdYlBu')), space='Lab')

ggplot(data=hmap_attr_leg_m, aes(type1, variable)) + geom_tile(aes(fill=valu
Legendaio: Type1 - Atributo") + scale_fill_gradientn(colours = hm.palette
element_text(angle=90, hjust=0)) + coord_equal()
```



Ahora realizamos el mismo mapa de calor pero para los pokemon no legendarios:

```
hmap_attr <- select(df, type1, is_legendary, hp, defense, attack, sp_attack, sp_
hmap_attr_norm <- filter(hmap_attr, is_legendary == 0)
hmap_attr_norm <- group_by(hmap_attr_norm, type1)
hmap_attr_norm <- summarise(hmap_attr_norm, hp=median(hp), attack=median(attack)
defense=median(defense), sp_attack=median(sp_attack), sp_defense=median(sp_defen
speed=median(speed))

hmap_attr_norm_m <- melt(hmap_attr_norm)


## Using type1 as id variables
```

```
hm.palette <- colorRampPalette(rev(brewer.pal(5, 'RdYlBu')), space='Lab')

ggplot(data=hmap_attr_norm_m, aes(type1, variable)) + geom_tile(aes(fill=val
Legendaio: Type1 - Atributo") + scale_fill_gradientn(colours = hm.palette
element_text(angle=90, hjust=0)) + coord_equal()
```

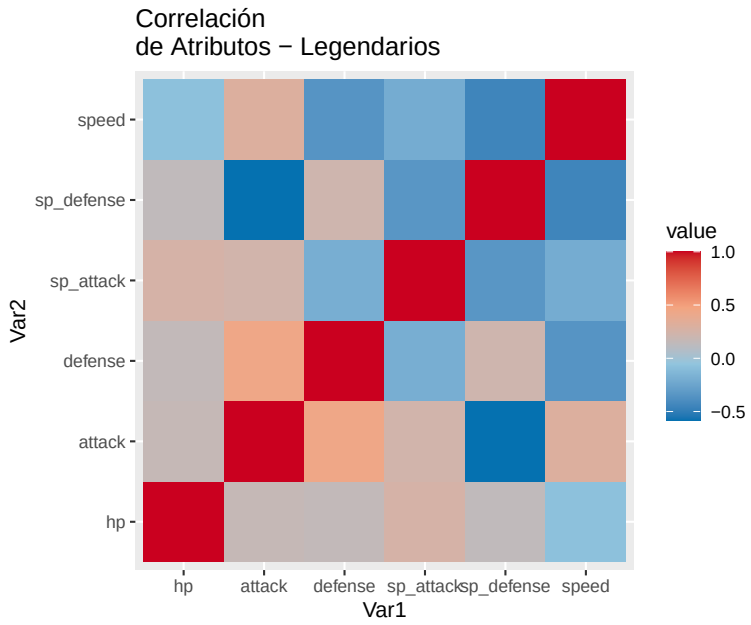


Al analizar el segundo mapa de calor vemos que los pokemon no legendarios tienden a ser mas "dispersos" en cuanto a sus atributos, ya que no se concentran en un solo tipo de pokemon, salvo un par de puntos mas notables como steel y defense, o fighting y attack, aparte de eso no presentan tanta concentración. Pero parecen que igual se nivalan en cuanto a sus atributos, ya que la mayoría de los pokemon con una estadistica alta tienen otras mas bajas.

B.6. Matriz de Correlación

```
hmap_attr <- select(df, type1, is_legendary, hp, defense, attack, sp_attack, sp_
hmap_attr_leg <- filter(hmap_attr, is_legendary == 1)
hmap_attr_leg <- group_by(hmap_attr_leg, type1)
hmap_attr_leg <- summarise(hmap_attr_leg, hp=median(hp), attack=median(attack),
defense=median(defense), sp_attack=median(sp_attack), sp_defense=median(sp_defen
speed=median(speed))

row.names(hmap_attr_leg) <- hmap_attr_leg$type1
```

```
hmap_attr_leg$type1 <- NULL
hmap_attr_leg$is_legendary <- NULL

hmap_attr_leg_cor <- cor(hmap_attr_leg)
hmap_attr_leg_cor_m <- melt(hmap_attr_leg_cor)
hm.palette <- colorRampPalette(rev(brewer.pal(4, 'RdBu')), space='Lab')
ggplot(data=hmap_attr_leg_cor_m, aes(Var1, Var2)) + geom_tile(aes(fill=value)) +
de Atributos - Legendarios") + scale_fill_gradientn(colours = hm.palette(100)) +
```



Análisis de la efectividad de los Pokémon basados en su tipo de ataque primario.

Se quiere encontrar la efectividad de los Pokémon basados en su tipo de ataque primario contra otros pokemones.

```
# se vuelve a cargar el dataset
df= (read.csv(file.choose(), header=T, encoding = "UTF-8"))
attach(df)
```

```
## The following objects are masked from df (pos = 10):
##
##     abilities, against_bug, against_dark, against_dragon,
##     against_electric, against_fairy, against_fight, against_fire,
##     against_flying, against_ghost, against_grass, against_ground,
```

```
##     against_ice, against_normal, against_poison, against_psychic,
##     against_rock, against_steel, against_water, attack, base_egg_steps,
##     base_happiness, base_total, capture_rate, classfication, defense,
##     experience_growth, generation, height_m, hp, is_legendary, name,
##     percentage_male, pokedex_number, sp_attack, sp_defense, speed,
##     type1, type2, weight_kg
```

```
df = tibble::as_tibble(df)
colnames(df)[25] <- "classification"
df$capture_rate <- as.numeric(df$capture_rate)


# Para esto seleccionamos un sub-conjunto de los datos:

df_fight_against <- select(df, type1, against_bug:against_water)
head(df_fight_against)
```

```
## # A tibble: 6 x 19
##   type1 against_bug against_dark against_dragon against_electric against_fair
##   <chr>       <dbl>        <dbl>          <dbl>             <dbl>         <dbl
## 1 grass         1               1              1               0.5           0.
## 2 grass         1               1              1               0.5           0.
## 3 grass         1               1              1               0.5           0.
## 4 fire          0.5             1              1               1             0.
## 5 fire          0.5             1              1               1             0.
## 6 fire          0.25            1              1               2             0.
## # i 13 more variables: against_fight <dbl>, against_fire <dbl>,
## #   against_flying <dbl>, against_ghost <dbl>, against_grass <dbl>,
## #   against_ground <dbl>, against_ice <dbl>, against_normal <dbl>,
## #   against_poison <dbl>, against_psychic <dbl>, against_rock <dbl>,
## #   against_steel <dbl>, against_water <dbl>
```

```
# Se deben encontrar la mediana de todas las columnas against_type

df_fight_against_g <- group_by(df_fight_against, type1)

df_fight_against_summ <- summarise(df_fight_against_g,
                                   against_bug = median(against_bug),
                                   against_dark = median(against_dark),
                                   against_dragon = median(against_dragon),
                                   against_electric = median(against_electric),
                                   against_fairy = median(against_fairy),
                                   against_fight = median(against_fight),
                                   against_fire = median(against_fire),
                                   against_flying = median(against_flying),
```

```
                                    against_ghost = median(against_ghost),
                                    against_grass = median(against_grass),
                                    against_ground = median(against_ground),
                                    against_ice = median(against_ice),
                                    against_normal = median(against_normal),
                                    against_poison = median(against_poison),
                                    against_psychic = median(against_psychic),
                                    against_rock = median(against_rock),
                                    against_steel = median(against_steel),
                                    against_water = median(against_water))

df_fight_against_long <- melt(df_fight_against_summ)


## Using type1 as id variables
```

```
hm.palette <- colorRampPalette(rev(brewer.pal(9, 'RdYlBu')), space='Lab')

ggplot(data=df_fight_against_long, aes(type1, variable)) + geom_tile(aes(fill=va
scale_fill_gradientn(colours = hm.palette(100)) + coord_equal() +
theme(axis.text.x=element_text(angle=90, hjust=0)) + ggtitle("Efectividad por ti
```