Práctica 2 - Limpieza y análisis de datos

Tipología y ciclo de vida de los datos, UOC

Carmelo León Suárez y Vanesa Navarro Oronoz

Mayo 2021

Contents


```
#Importamos las librearías necesarias
library(ggplot2)
library(skimr)
library(knitr)
#library(kableExtra)
library(pROC)
library(corrplot)
```

1 Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

Se utiliza la función `read.csv()` para la lectura del archivo ya que está delimitados por comas. Se guardan los datos en el dataframe `data`.

```
# Leemos el archivo de datos
data <- read.csv("./Pokemon.csv",header=T)
```

Mostramos una muestra del dataset

```
# Muestra dedataset
head(data)
```

```
##   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
```

Con la función `str()` examinamos los valores resumen de cada tipo de variable. Así podemos ver que los datos están compuestos por 800 registros y 13 variables o atributos. Estos atributos, son de tipo enteros y character.

```
# Verificamos la estructura del conjunto de datos
str(data)
```

```
## '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 : chr  "False" "False" "False" "False" ...
```

Aplicamos la función `summary()` para obtener una estadística descriptiva simple de cada variable:

```
#Estadísticas básicas
summary(data)
```

```
##        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
##  Length:800
##  Class :character
##  Mode  :character
##
##
##
```

Con la siguiente función obtenemos más información de dataset como valores nulos e histogramas de las variables.

```
skim(data)
```

Table 1: Data summary

Name	data
Number of rows	800
Number of columns	13
Column type frequency:	
character	4

Table 1: Data summary

numeric	9
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Name	0	1	3	25	0	800	0
Type.1	0	1	3	8	0	18	0
Type.2	0	1	0	8	386	19	0
Legendary	0	1	4	5	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
X.	0	1	362.81	208.34	1	184.75	364.5	539.25	721	
Total	0	1	435.10	119.96	180	330.00	450.0	515.00	780	
HP	0	1	69.26	25.53	1	50.00	65.0	80.00	255	
Attack	0	1	79.00	32.46	5	55.00	75.0	100.00	190	
Defense	0	1	73.84	31.18	5	50.00	70.0	90.00	230	
Sp..Atk	0	1	72.82	32.72	10	49.75	65.0	95.00	194	
Sp..Def	0	1	71.90	27.83	20	50.00	70.0	90.00	230	
Speed	0	1	68.28	29.06	5	45.00	65.0	90.00	180	
Generation	0	1	3.32	1.66	1	2.00	3.0	5.00	6	

Tabla con el resumen y descripción de las variables:

VARIABLE	TIPO	DESCRIPCIÓN
X.	integer	ID de cada Pokémon
Name	character	Nombre de cada Pokémon
Type.1	character	El tipo principal de Pokémon, esto determina la debilidad / resistencia a los ataques
Type.2	character	El tipo secundario de Pokémon si lo tiene
Total	integer	Suma de todas las estadísticas, una guía general de qué tan fuerte es un Pokémon
HP	integer	Puntos de golpe, o salud, define cuánto daño puede soportar un Pokémon antes de desmayarse
Attack	integer	El ataque base de los Pokémon para ataques normales
Defense	integer	La defensa base de los Pokémon contra ataques normales
Sp..Atk	integer	Ataque especial, el modificador base para ataques especiales
Sp..Def	integer	La resistencia base al daño contra ataques especiales
Speed	integer	Determina qué Pokémon ataca primero en cada ronda
Generation	integer	La generación numerada en la que se introdujo por primera vez el Pokémon
Legendary	character	Indica si el Pokémon es legendario

El dataset y el resumen lo hemos obtenido del siguiente repositorio:

https://www.kaggle.com/abcsds/pokemon

La razón por la que hemos cogido este dataset es para salirnos un poco de lo habitual y además trabajar con un dataset que aunque no tenga una acogida muy grande si creo que hay un sector de niños y no tan niños que han jugado a este juego de realidad aumentada o son apasionados de los distintos productos que existen de ellos. Este conjunto de datos puede ser de gran utilidad para enseñar estadística a los niños y jóvenes con un tema de interés para ellos.

Identificar patrones con distintos estadísticos puede ser interesante para los amantes de juego, dibujos y cómics.

Con este conjunto de datos, podríamos responder las siguientes preguntas:

¿Siguen algún patrón las características de los Pokémon? ¿Están relacionados los atributos de los Pokémon? *¿Los Pokémon de fuego tienen mejor ataque que los de agua?* ¿Es posible construir un modelo predictor para identificar Pokémon legendarios?

---

2 Integración y selección de los datos de interés a analizar.

Debido a las características del dataset no es necesario hacer ninguna integración y nos quedamos con todos los datos

---

3 Limpieza de los datos.

3.1 ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

Para tener una información global y completa del dataset, se van a buscar los registros vacíos o Na existentes para conocimiento y posterior tratamiento. A pesar de que la función skim() ya nos ha dado parte de esta información hacemos pruebas especificas.

```
#Estadísticas de valores vacíos
colSums(is.na(data))
```

```
##         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
```

No hay registros con NA.

```
#Estadísticas de variables con cadenas vacías
colSums(data=="")
```

```
##         X.       Name     Type.1     Type.2      Total         HP     Attack
##          0          0          0        386          0          0          0
##    Defense    Sp..Atk    Sp..Def      Speed Generation  Legendary
##          0          0          0          0          0          0
```

En este resumen vemos que en la columna Type.2 hay 386 registros que están vacíos. Algo que ya nos indicaba la descripción del dataset.

Convertimos a datos de tipo factor los atributos: Type.1, Type.2 y Legendary.

```
#Creamos factor
data$Type.1 <- as.factor(data$Type.1)
data$Type.2 <- as.factor(data$Type.2)
data$Legendary <- as.factor(data$Legendary)

#Mostramos los valores de los factores
table(data$Type.1)
```

```
##
##      Bug     Dark   Dragon Electric    Fairy Fighting     Fire   Flying
##       69       31       32       44       17       27       52        4
##    Ghost    Grass   Ground      Ice   Normal   Poison  Psychic     Rock
##       32       70       32       24       98       28       57       44
##    Steel    Water
##       27      112
```

```
table(data$Type.2)
```

```
##
##                Bug     Dark   Dragon Electric    Fairy Fighting     Fire
##      386         3       20       18        6       23       26       12
##   Flying    Ghost    Grass   Ground      Ice   Normal   Poison  Psychic
##       97       14       25       35       14        4       34       33
##     Rock    Steel    Water
##       14       22       14
```

```
table(data$Legendary)
```

```
##
## False  True
##   735    65
```

Imputación de valores vacíos en atributo Type.2

Como se ha visto anteriormente, la columna Type.2, que describe el tipo secundario del pokémon, contiene 386 registros vacíos. Vamos a imputar un tipo secundario a esos pokémon basándonos en el tipo secundario más frecuente para cada tipo primario. Para ello, creamos la tabla de contingencia entre las variables Type.1 y Type.2 y posteriormente la tabla de frecuencias relativas.

```r
#Tablas contingencia y frecuencia
ctable <- table(data$Type.1, data$Type.2)
tabla_frel <- prop.table(ctable, margin=1)
tabla_frel
```

```
##
##                              Bug        Dark      Dragon    Electric
##   Bug      0.246376812 0.000000000 0.000000000 0.000000000 0.028985507
##   Dark     0.322580645 0.000000000 0.000000000 0.096774194 0.000000000
##   Dragon   0.343750000 0.000000000 0.000000000 0.000000000 0.031250000
##   Electric 0.613636364 0.000000000 0.000000000 0.022727273 0.000000000
##   Fairy    0.882352941 0.000000000 0.000000000 0.000000000 0.000000000
##   Fighting 0.740740741 0.000000000 0.037037037 0.000000000 0.000000000
##   Fire     0.538461538 0.000000000 0.000000000 0.019230769 0.000000000
##   Flying   0.500000000 0.000000000 0.000000000 0.500000000 0.000000000
##   Ghost    0.312500000 0.000000000 0.031250000 0.062500000 0.000000000
##   Grass    0.471428571 0.000000000 0.042857143 0.014285714 0.000000000
##   Ground   0.406250000 0.000000000 0.093750000 0.062500000 0.031250000
##   Ice      0.541666667 0.000000000 0.000000000 0.000000000 0.000000000
##   Normal   0.622448980 0.000000000 0.000000000 0.000000000 0.000000000
##   Poison   0.535714286 0.035714286 0.107142857 0.035714286 0.000000000
##   Psychic  0.666666667 0.000000000 0.017543860 0.000000000 0.000000000
##   Rock     0.204545455 0.045454545 0.045454545 0.045454545 0.000000000
##   Steel    0.185185185 0.000000000 0.000000000 0.037037037 0.000000000
##   Water    0.526785714 0.000000000 0.053571429 0.017857143 0.017857143
##
##                   Fairy    Fighting        Fire      Flying       Ghost
##   Bug      0.000000000 0.028985507 0.028985507 0.202898551 0.014492754
##   Dark     0.000000000 0.064516129 0.096774194 0.161290323 0.064516129
##   Dragon   0.031250000 0.000000000 0.031250000 0.187500000 0.000000000
##   Electric 0.022727273 0.000000000 0.022727273 0.113636364 0.022727273
##   Fairy    0.000000000 0.000000000 0.000000000 0.117647059 0.000000000
##   Fighting 0.000000000 0.000000000 0.000000000 0.037037037 0.000000000
##   Fire     0.000000000 0.134615385 0.000000000 0.115384615 0.000000000
##   Flying   0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
##   Ghost    0.000000000 0.000000000 0.093750000 0.062500000 0.000000000
##   Grass    0.028571429 0.042857143 0.000000000 0.071428571 0.000000000
##   Ground   0.000000000 0.000000000 0.031250000 0.125000000 0.062500000
##   Ice      0.000000000 0.000000000 0.000000000 0.083333333 0.041666667
##   Normal   0.051020408 0.020408163 0.000000000 0.244897959 0.000000000
##   Poison   0.000000000 0.071428571 0.000000000 0.107142857 0.000000000
##   Psychic  0.105263158 0.052631579 0.017543860 0.105263158 0.017543860
##   Rock     0.068181818 0.022727273 0.000000000 0.090909091 0.000000000
##   Steel    0.111111111 0.037037037 0.000000000 0.037037037 0.148148148
##   Water    0.017857143 0.026785714 0.000000000 0.062500000 0.017857143
##
##                   Grass      Ground         Ice      Normal      Poison
##   Bug      0.086956522 0.028985507 0.000000000 0.000000000 0.173913043
##   Dark     0.000000000 0.000000000 0.064516129 0.000000000 0.000000000
##   Dragon   0.000000000 0.156250000 0.093750000 0.000000000 0.000000000
##   Electric 0.022727273 0.000000000 0.022727273 0.045454545 0.000000000
##   Fairy    0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
##   Fighting 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
```

```
##   Fire     0.000000000 0.057692308 0.000000000 0.038461538 0.000000000
##   Flying   0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
##   Ghost    0.312500000 0.000000000 0.000000000 0.000000000 0.125000000
##   Grass    0.000000000 0.014285714 0.042857143 0.000000000 0.214285714
##   Ground   0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
##   Ice      0.000000000 0.125000000 0.000000000 0.000000000 0.000000000
##   Normal   0.020408163 0.010204082 0.000000000 0.000000000 0.000000000
##   Poison   0.000000000 0.071428571 0.000000000 0.000000000 0.000000000
##   Psychic  0.017543860 0.000000000 0.000000000 0.000000000 0.000000000
##   Rock     0.045454545 0.136363636 0.045454545 0.000000000 0.000000000
##   Steel    0.000000000 0.074074074 0.000000000 0.000000000 0.000000000
##   Water    0.026785714 0.089285714 0.026785714 0.000000000 0.026785714
##
##                Psychic        Rock       Steel       Water
##   Bug      0.000000000 0.043478261 0.101449275 0.014492754
##   Dark     0.064516129 0.000000000 0.064516129 0.000000000
##   Dragon   0.125000000 0.000000000 0.000000000 0.000000000
##   Electric 0.000000000 0.000000000 0.068181818 0.022727273
##   Fairy    0.000000000 0.000000000 0.000000000 0.000000000
##   Fighting 0.111111111 0.000000000 0.074074074 0.000000000
##   Fire     0.038461538 0.019230769 0.019230769 0.019230769
##   Flying   0.000000000 0.000000000 0.000000000 0.000000000
##   Ghost    0.000000000 0.000000000 0.000000000 0.000000000
##   Grass    0.028571429 0.000000000 0.028571429 0.000000000
##   Ground   0.062500000 0.093750000 0.031250000 0.000000000
##   Ice      0.083333333 0.000000000 0.000000000 0.125000000
##   Normal   0.020408163 0.000000000 0.000000000 0.010204082
##   Poison   0.000000000 0.000000000 0.000000000 0.035714286
##   Psychic  0.000000000 0.000000000 0.000000000 0.000000000
##   Rock     0.045454545 0.000000000 0.068181818 0.136363636
##   Steel    0.259259259 0.111111111 0.000000000 0.000000000
##   Water    0.044642857 0.035714286 0.008928571 0.000000000
```

De esta forma podemos obtener los valores más frecuentes para el Type.2 en función del Type.1. Siendo esto así, procedemos a la imputación de esos valores en el dataset.

```
#Imputación de valores
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Bug", "Flying", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Dark", "Flying", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Dragon", "Flying", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Electric", "Flying", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Fairy", "Flying", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Fighting", "Psychic", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Fire", "Ground", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Flying", "Dragon", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Ghost", "Grass", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Grass", "Poison", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Ground", "Rock", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Ice", "Water", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Normal", "Flying", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Poison", "Dark", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Psychic", "Fairy", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Rock", "Ground", data$Type.2)
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Steel", "Psychic", data$Type.2)
```

```
data$Type.2 <- ifelse(data$Type.2=="" & data$Type.1=="Water", "Ground", data$Type.2)
```

Volvemos a realizar el análisis de cadenas vacías para confirmar que ya no existen.

```
#Estadísticas de variables con cadenas vacías
colSums(data=="")
```

```
##          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
```

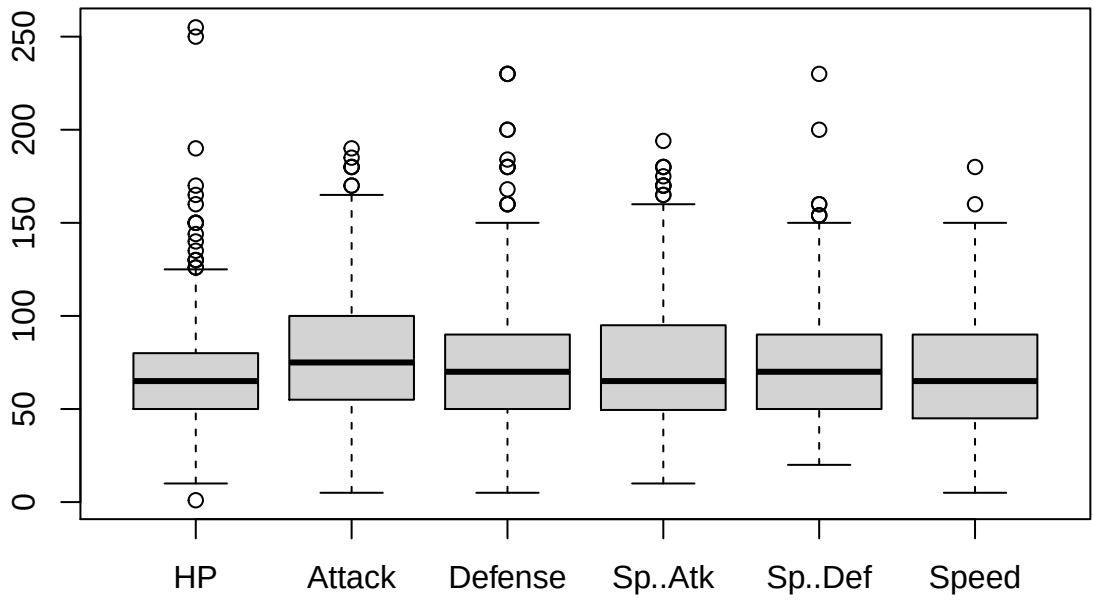
Análisis de variables cuantitativas del dataset

```
#Creamos dataset para la variables cuantitativas
temp <- data[c(6:11)]
```

3.2 Identificación y tratamiento de valores extremos.

Mostramos los boxplot de estas variables que nos ayudaran a detectar la existencia de valores atípicos o outliers.

```
#Mostramos boxplot
boxplot(temp)
```



```r
#Valores extremos y sus posiciones en HP:
values <- boxplot.stats(temp$HP)$out
idx <- which( temp$HP %in% values)

cat("\nValores extremos en HP:", toString(values), "\n" )
```

```
##
## Valores extremos en HP: 140, 250, 130, 130, 160, 190, 255, 150, 1, 144, 130, 170, 150, 135, 150, 150
```

```r
HP_outliers <- temp[idx,]
HP_outliers %>% kable( caption="Valores atipicos de HP")
```

Table 5: Valores atipicos de HP

	HP	Attack	Defense	Sp..Atk	Sp..Def	Speed
46	140	70	45	85	50	45
122	250	5	5	35	105	50
143	130	85	80	85	95	60
146	130	65	60	110	95	65
156	160	110	65	65	110	30
218	190	33	58	33	58	33
262	255	10	10	75	135	55
314	150	160	100	95	65	100
317	1	90	45	30	30	40
322	144	120	60	40	60	50
351	130	70	35	70	35	60
352	170	90	45	90	45	60
474	150	80	44	90	54	80
496	135	85	40	40	85	5
545	150	100	120	100	120	90
546	150	120	100	120	100	90
656	165	75	80	40	45	65
793	126	131	95	131	98	99
794	126	131	95	131	98	99

```r
#Valores extremos y sus posiciones en Attack:
values <- boxplot.stats(temp$Attack)$out
idx <- which( temp$Attack %in% values)

cat("\nValores extremos en Attack:", toString(values), "\n" )
```

```
##
## Valores extremos en Attack: 190, 185, 180, 180, 180, 170, 170
```

```r
HP_outliers <- temp[idx,]
HP_outliers %>% kable( caption="Valores atipicos de Attack")
```

Table 6: Valores atipicos de Attack

	HP	Attack	Defense	Sp..Atk	Sp..Def	Speed
164	106	190	100	154	100	130
233	80	185	115	40	105	75
425	100	180	160	150	90	90
427	105	180	100	180	100	115
430	50	180	20	180	20	150
495	108	170	115	120	95	92
712	125	170	100	120	90	95

```
#Valores extremos y sus posiciones en Defense:
values <- boxplot.stats(temp$Defense)$out
idx <- which( temp$Defense %in% values)

cat("\nValores extremos en Defense:", toString(values), "\n" )
```

```
##
## Valores extremos en Defense: 180, 180, 160, 200, 230, 230, 180, 230, 200, 160, 160, 168, 184
```

```
HP_outliers <- temp[idx,]
HP_outliers %>% kable( caption="Valores atipicos de Defense")
```

Table 7: Valores atipicos de Defense

	HP	Attack	Defense	Sp..Atk	Sp..Def	Speed
88	95	75	180	130	80	30
99	50	95	180	85	45	70
104	35	45	160	30	45	70
224	75	85	200	55	65	30
225	75	125	230	55	95	30
231	20	10	230	10	230	5
333	70	110	180	60	60	50
334	70	140	230	60	80	50
415	80	100	200	50	100	50
425	100	180	160	150	90	90
431	50	70	160	70	160	90
457	60	52	168	47	138	30
790	95	117	184	44	46	28

```
#Valores extremos y sus posiciones en Sp..Atk:
values <- boxplot.stats(temp$Sp..Atk)$out
idx <- which( temp$Sp..Atk %in% values)

cat("\nValores extremos en Sp..Atk:", toString(values), "\n" )
```

```
##
## Valores extremos en Sp..Atk: 175, 170, 194, 165, 165, 180, 180, 180, 170, 170
```

```
HP_outliers <- temp[idx,]
HP_outliers %>% kable( caption="Valores atipicos de Sp..Atk")
```

Table 8: Valores atipicos de Sp..Atk

	HP	Attack	Defense	Sp..Atk	Sp..Def	Speed
72	55	50	65	175	95	150
103	60	65	80	170	95	130
165	106	150	70	194	120	140
197	90	95	105	165	110	45
307	68	85	65	165	135	100
423	100	150	90	180	160	90
427	105	180	100	180	100	115
430	50	180	20	180	20	150
713	125	120	90	170	100	95
799	80	160	60	170	130	80

```
#Valores extremos y sus posiciones en Sp..Def:
values <- boxplot.stats(temp$Sp..Def)$out
idx <- which( temp$Sp..Def %in% values)

cat("\nValores extremos en Sp..Def:", toString(values), "\n" )
```

```
##
## Valores extremos en Sp..Def: 230, 154, 154, 200, 160, 160, 154
```

```
HP_outliers <- temp[idx,]
HP_outliers %>% kable( caption="Valores atipicos de Sp..Def")
```

Table 9: Valores atipicos de Sp..Def

	HP	Attack	Defense	Sp..Atk	Sp..Def	Speed
231	20	10	230	10	230	5
270	106	90	130	90	154	110
271	106	130	90	110	154	90
416	80	50	100	100	200	50
423	100	150	90	180	160	90
431	50	70	160	70	160	90
740	78	65	68	112	154	75

```
#Valores extremos y sus posiciones en Speed:
values <- boxplot.stats(temp$Speed)$out
idx <- which( temp$Speed %in% values)

cat("\nValores extremos en Speed:", toString(values), "\n" )
```

```
##
## Valores extremos en Speed: 160, 180
```

```
HP_outliers <- temp[idx,]
HP_outliers %>% kable( caption="Valores atipicos de Speed")
```

Table 10: Valores atipicos de Speed

	HP	Attack	Defense	Sp..Atk	Sp..Def	Speed
316	61	90	45	50	50	160
432	50	95	90	95	90	180

El boxplot nos confirma la presencia de valores atípicos pero que consideramos posibles y por tanto no creemos que haya que realizar ningún tipo de acción.

---

4 Análisis de los datos.

##Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

La idea es analizar los datos cuantitativos que ya hemos trabajado anteriormente en primera estancia. A partir de aquí estudiaremos su normalidad, revisaremos si hay alguna correlación entre ellas y para finalizar predeciremos los valores de Tipo2 que están vacíos y daremos respuesta a la siguiente pregunta:

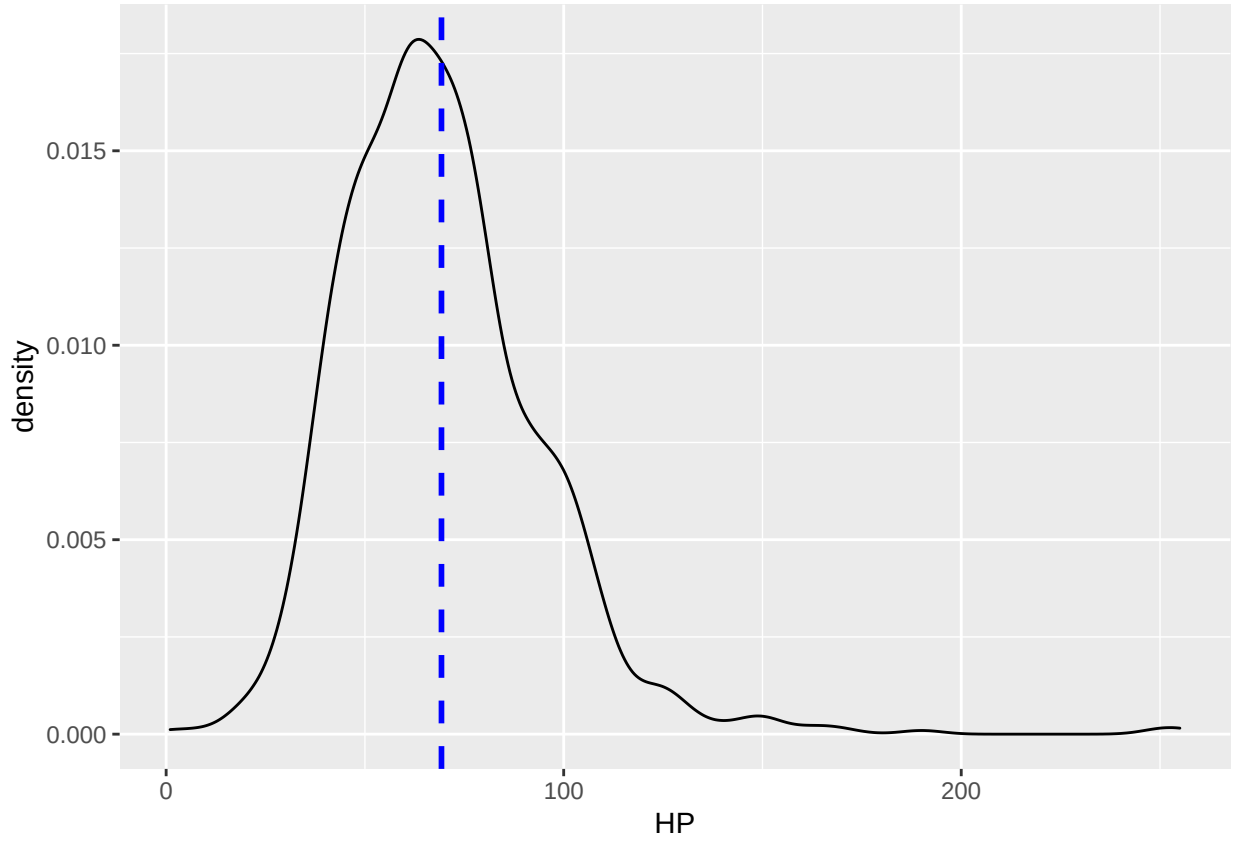
¿Los pokemon de fuego tienen mejor ataque que los de agua?

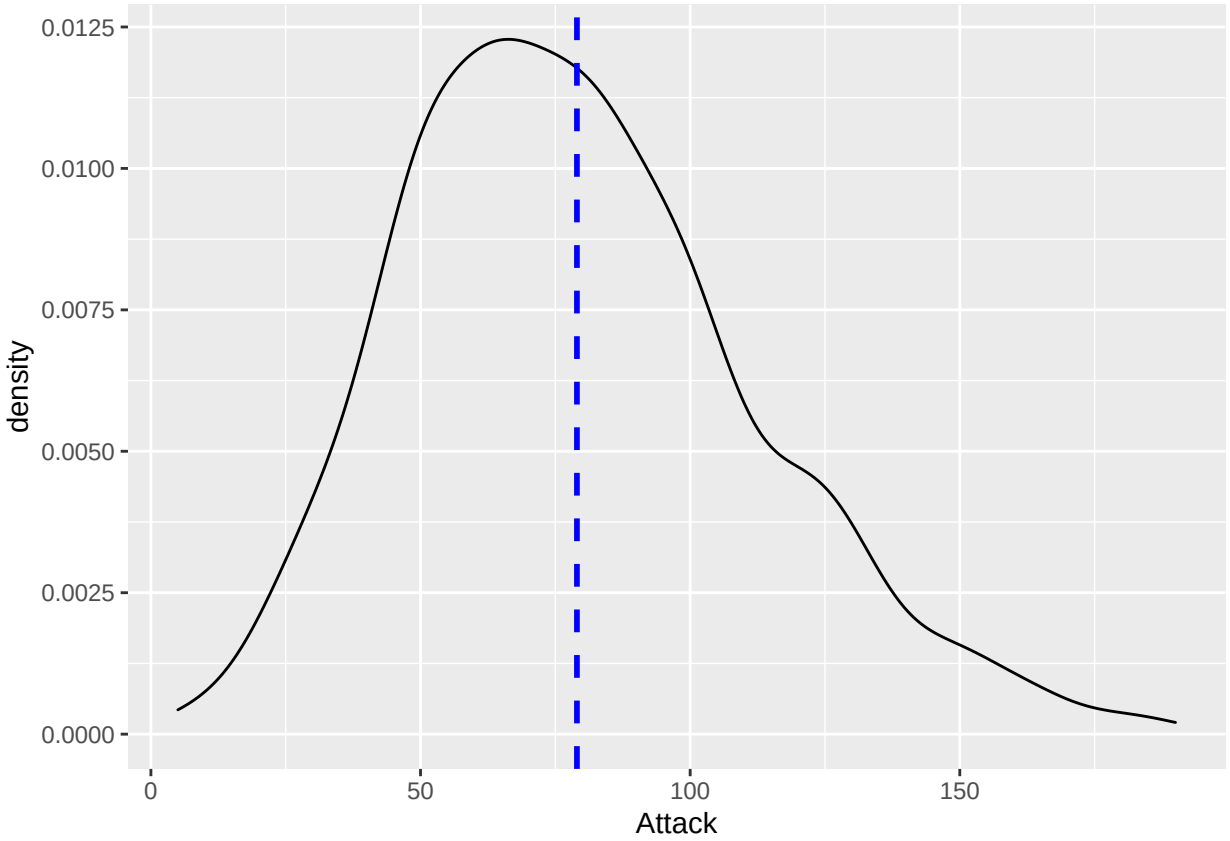
4.1 Comprobación de la normalidad y homogeneidad de la varianza.

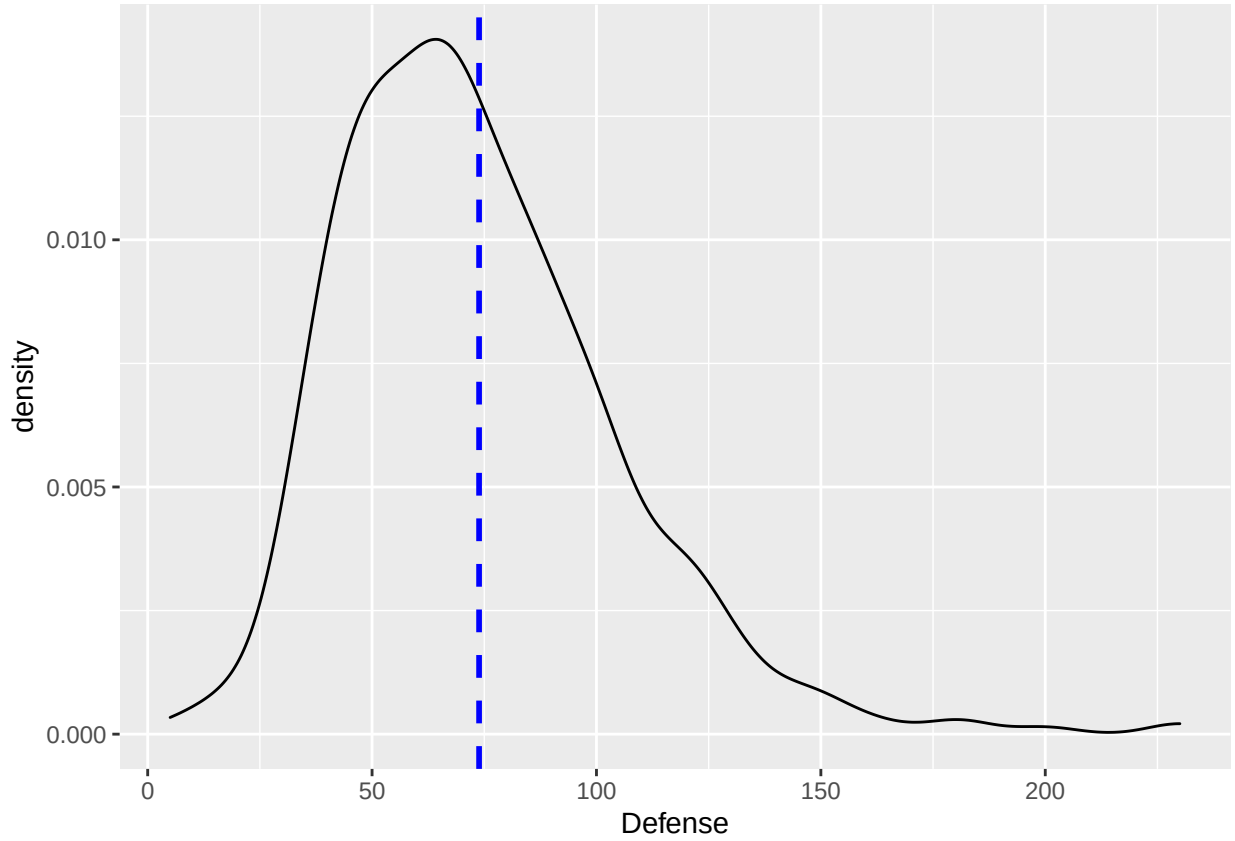
4.1.1 Estudio visual

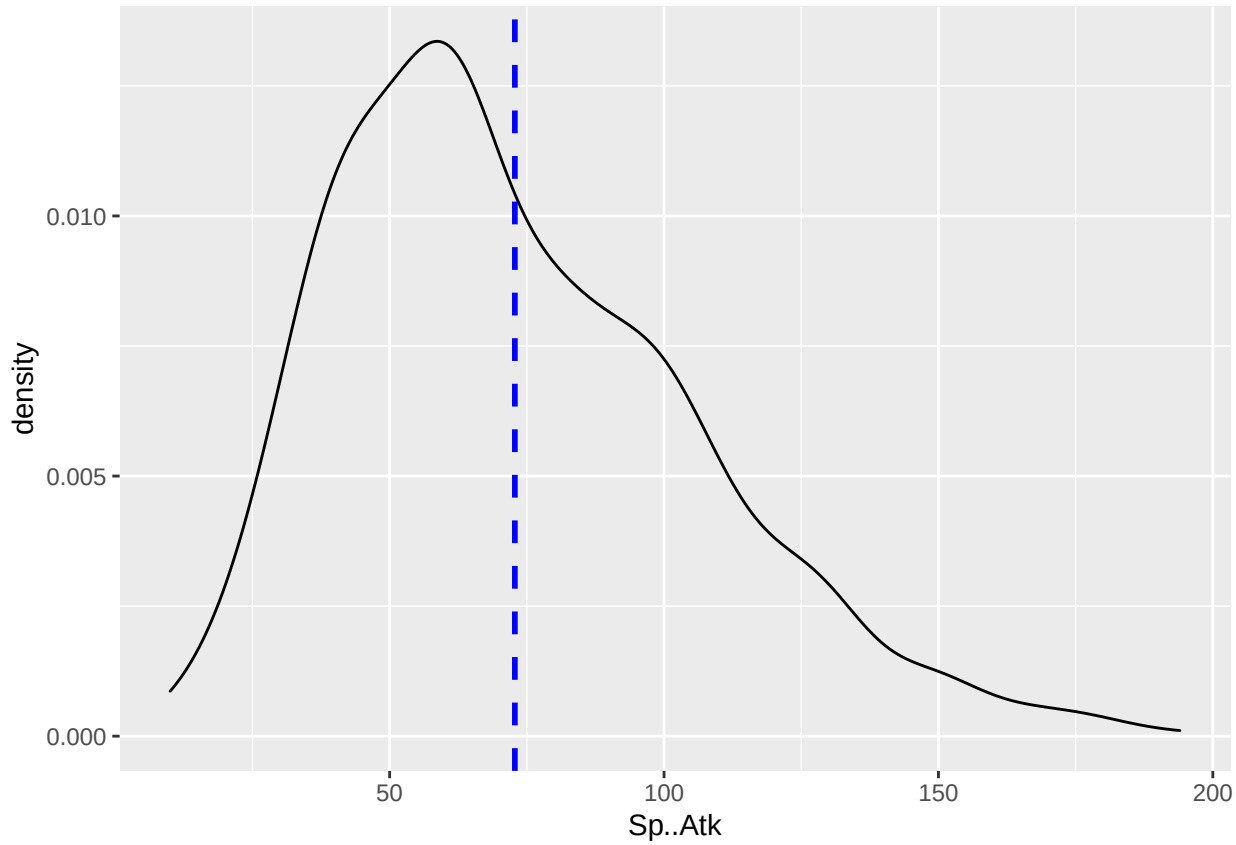
A continuación hacemos un análisis visual de la normalidad de los tributos cuantitativos primero con un gráfico de densidad y luego con un gráfico cuantil cuantil.

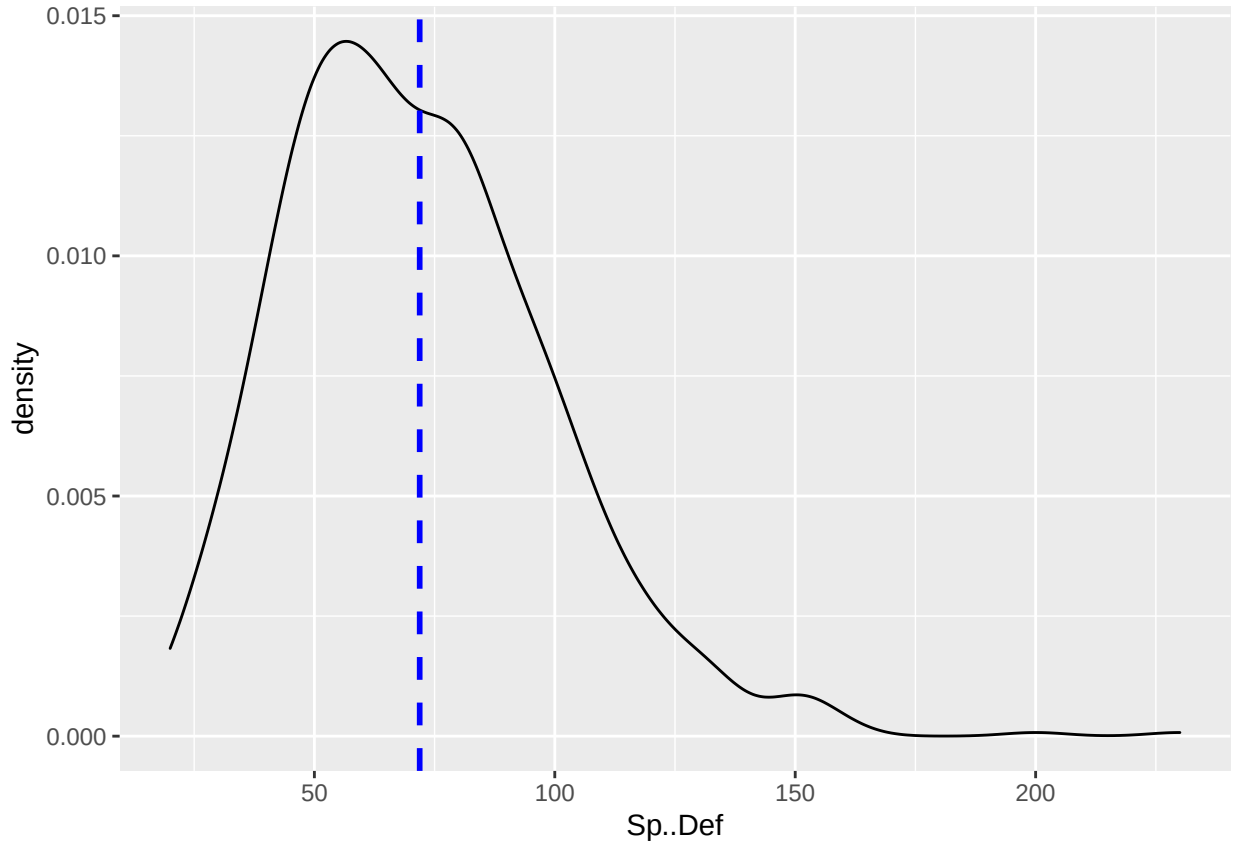
```
par(mfrow=c(2,3))
for(i in 1:ncol(temp)) {
  print(ggplot(mapping= aes(x=temp[,i]))+ geom_density() + geom_vline(aes(xintercept=mean(temp[,i])),
            color="blue", linetype="dashed", size=1)+  xlab(colnames(temp)[i]))
}
```

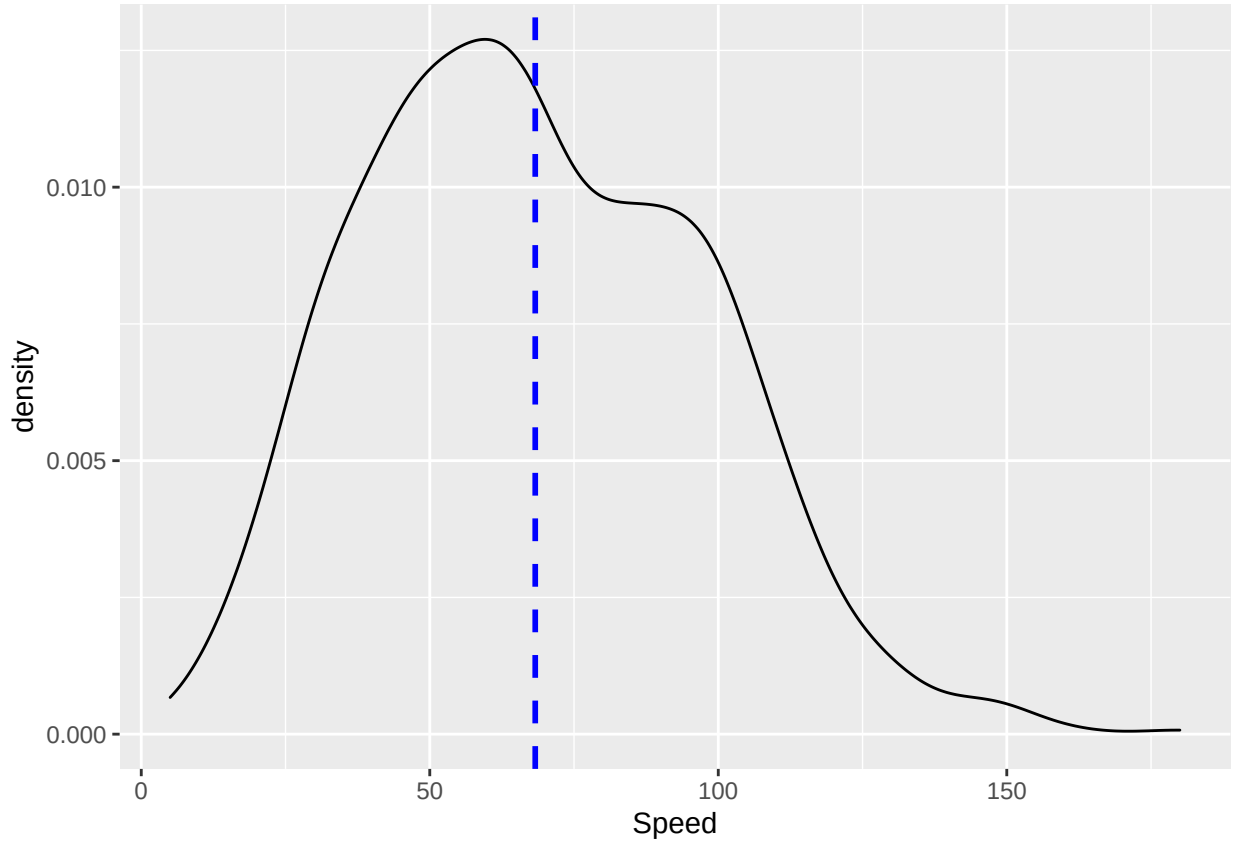

0.015
0.010
0.005
0.000
density
0
100
200
HP


0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
density
0
50
100
150
Attack

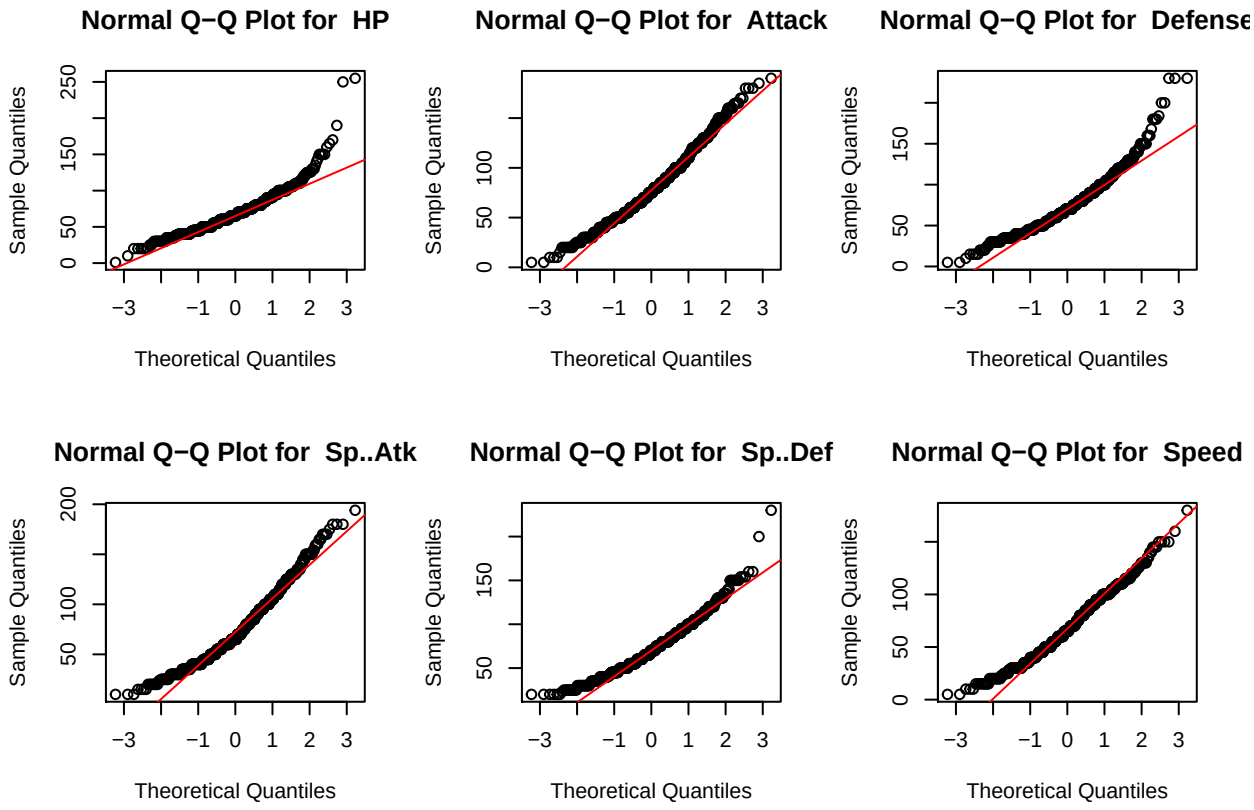

density
0.010
0.005
0.000
0
50
100
150
200
Defense


density
0.010
0.005
0.000
50
100
150
200
Sp..Atk


0.015
0.010
0.005
0.000
density
50
100
150
200
Sp..Def




```
par(mfrow=c(2,3))
for(i in 1:ncol(temp)) {
  qqnorm(temp[,i],main = paste("Normal Q-Q Plot for ",colnames(temp)[i]))
  qqline(temp[,i],col="red")
}
```



A la vista de los datos no podemos afirmar que sigan una distribución normal por lo que a continuación realizaremos un test de normalidad para asegurar el resultado.

###Test de normalidad

El test de Shapiro-Wilks plantea la hipótesis nula que una muestra proviene de una distribución normal. Elegimos un nivel de significancia, por ejemplo 0,05, y tenemos una hipótesis alternativa que sostiene que la distribución no es normal.

Tenemos:

H_0 : La distribución es normal

H_1 : La distribución no es normal

El test Shapiro-Wilks intenta rechazar la hipótesis nula a nuestro nivel de significancia. Para realizar el test usamos la función `shapiro.test` en R:

```
#Test shapiro-wilks para todas las variables cuantitativas
for(i in 1:ncol(temp)) {
  norm_test <- shapiro.test(temp[,i])
  print(paste("p-valor para", colnames(temp)[i], norm_test$p.value))
}
```

```
## [1] "p-valor para HP 1.15236449336525e-20"
## [1] "p-valor para Attack 2.47215419020232e-09"
## [1] "p-valor para Defense 9.92317296967491e-18"
## [1] "p-valor para Sp..Atk 4.66514063674639e-14"
## [1] "p-valor para Sp..Def 8.25179172178909e-14"
## [1] "p-valor para Speed 1.30954199196094e-07"
```

Vemos que en todos los casos el valor de probabilidad (p) es muy inferior al nivel de significancia (0,05), por lo que rechazamos la hipótesis nula, y por tanto, concluimos que las variables no siguen una distribución normal y por lo tanto tampoco es necesario hacer el test de varianzas.

4.2 Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

4.2.1 Correlación entre variables

Vamos a estudiar la correlación de todas las variables cuantitativas.

```
#Calculamos la matriz de correlación
cor(temp) %>% kable( caption="Matriz de correlaciónd de varibles cuantitativas")
```

Table 11: Matriz de correlaciónd de varibles cuantitativas

	HP	Attack	Defense	Sp..Atk	Sp..Def	Speed
HP	1.0000000	0.4223860	0.2396223	0.3623799	0.3787181	0.1759521
Attack	0.4223860	1.0000000	0.4386871	0.3963618	0.2639896	0.3812397
Defense	0.2396223	0.4386871	1.0000000	0.2235486	0.5107466	0.0152266
Sp..Atk	0.3623799	0.3963618	0.2235486	1.0000000	0.5061214	0.4730179
Sp..Def	0.3787181	0.2639896	0.5107466	0.5061214	1.0000000	0.2591331
Speed	0.1759521	0.3812397	0.0152266	0.4730179	0.2591331	1.0000000

Atendiendo a los resultados de la matriz no podemos afirmar que haya alguna correlación fuerte entre alguna de las variables.

4.2.2 Contraste de Hipótesis

Pregunta de investigación: ¿Los Pokémon de fuego tienen mejor ataque que los de agua?

Hipótesis nula: Los Pokémon de fuego tienen un ataque peor o igual que los de agua.

Hipótesis alternativa: Los Pokémon de fuego tienen mejor ataque que los de agua.

$\mathbf{H_0}$: $_{\text{fuego}} <= _{\text{agua}}$

$\mathbf{H_1}$: $_{\text{fuego}} > _{\text{agua}}$

Para comenzar el análisis extraemos las muestras del estudio, una para los Pokémon de fuego y otra para los Pokémon de agua.

```
#División de muestras
poke_fire <- data[ data$Type.1=="Fire",]
poke_water <- data[ data$Type.1=="Water",]
```

Aplicamos un test paramétrico usando la distribución T-student (T-student se asemeja a la distribución normal para muestras grandes n>30) sobre la diferencia de medias de las dos muestras independientes mediante la función t.test de R con un nivel de confianza del 95%.

```r
#Aplicacion del t-test
t.test(poke_fire$Attack, poke_water$Attack, alternative="greater", conf.level = 0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  poke_fire$Attack and poke_water$Attack
## t = 2.2088, df = 98.269, p-value = 0.01476
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  2.635492      Inf
## sample estimates:
## mean of x mean of y
##  84.76923  74.15179
```

El p-valor es menor que el nivel de significancia ($0.01 < 0.05$), lo cual significa que podemos rechazar la hipótesis nula a favor de la hipótesis alternativa. Podemos concluir con un 95% de nivel de confianza que el ataque de los pokémon de fuego es mejor que el ataque de los pokémon de agua.

4.2.3 Regresión logística

A continuación, vamos a crear un modelo de regresion logistica para predecir la probabilidad de que un pokémon sea legendario. Aplicaremos este método ya que la variable Legendario es una variable dicotómica dependiente. Filtramos las columnas necesarias.

```r
#Seleccionamos las variables para la regresión
data_mlog <- data[c(2:3,6:11,13)]
```

En la regresión logística se modela la probabilidad de que la variable respuesta Y (en este caso ser Legendario) pertenezca al nivel de referencia en función del valor que adquieran los predictores, mediante el uso de LOG of ODDs.Permite calcular la probabilidad de que la variable dependiente pertenezca a cada una de las categorías en función del valor que adquieran las variables independientes. Calculamos el modelo de regresión logística:

```r
#Creamos el modelo de regresión logística
model_regresion_log <- glm(Legendary ~ HP + Attack + Defense + Sp..Atk + Sp..Def + Speed, data=data_mlog
summary(model_regresion_log)
```

```
##
## Call:
## glm(formula = Legendary ~ HP + Attack + Defense + Sp..Atk + Sp..Def +
##     Speed, family = binomial(link = logit), data = data_mlog)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.26688  -0.14919  -0.03675  -0.00546   2.11519
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -21.877464   2.442548  -8.957  < 2e-16 ***
## HP            0.034396   0.008776   3.919 8.89e-05 ***
## Attack        0.017868   0.006629   2.695  0.00703 **
```

```
## Defense       0.033009   0.008294   3.980 6.89e-05 ***
## Sp..Atk       0.035763   0.007141   5.008 5.50e-07 ***
## Sp..Def       0.043027   0.009013   4.774 1.81e-06 ***
## Speed         0.050447   0.009592   5.259 1.44e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 450.90  on 799  degrees of freedom
## Residual deviance: 176.89  on 793  degrees of freedom
## AIC: 190.89
##
## Number of Fisher Scoring iterations: 8
```

Los coeficientes nos dan el cambio en el logaritmo de Legendary (logit) como resultado de un aumento de una unidad en cada una de las variables predictoras (HP, Attack, ...). Si miramos el p-valor de los coeficientes, vemos que todos son estadísticamente significativos en el análisis.

Calculamos la razón de ventajas (OR) que permite cuantificar el efecto de las variables explicativas en la respuesta (Incremento proporcional en la ventaja o probabilidad de éxito, al aumentar una unidad la variable manteniendo las demás fijas). Para la interpretación de los coeficientes los exponenciamos y obtenemos así los Odds ratio/razón de ventajas/OR. También listamos los intervalos de confianza de los coeficientes del modelo para mayor información.

```r
#Razon de ventajas e intervalos de confianza
exp(cbind(OR = coef(model_regresion_log), confint(model_regresion_log)))
```

```
##                         OR        2.5 %       97.5 %
## (Intercept) 3.153102e-10 1.567851e-12 2.410934e-08
## HP          1.034994e+00 1.016853e+00 1.053050e+00
## Attack      1.018028e+00 1.005077e+00 1.031671e+00
## Defense     1.033560e+00 1.016875e+00 1.050771e+00
## Sp..Atk     1.036410e+00 1.022655e+00 1.051841e+00
## Sp..Def     1.043966e+00 1.026256e+00 1.063484e+00
## Speed       1.051741e+00 1.033301e+00 1.073200e+00
```

La interpretación del odds-ratio es que valores mayores que 1 indican que si el predictor aumenta los odds de la variable dependiente crecen. Inversamente, un valor menor que 1 indica que tal como el predictor aumente el odds del resultado decrece.

- Un OR = 1 implica que no existe asociación entre la variable respuesta y la covariable.
- Un OR inferior a la unidad se interpreta como un factor de protección, es decir, el suceso es menos probable en presencia de dicha covariable.
- Un OR mayor a la unidad se interpreta como un factor de riesgo, es decir, el suceso es más probable en presencia de dicha covariable.

En nuestro caso todos los OR son mayores a la unidad por lo que concluimos que todos los atributos son factores de riesgo en este análisis.

Curva ROC

Vamos a analizar el rendimiento del modelo creado con la curva ROC. El análisis ROC proporciona un modo de seleccionar modelos posiblemente óptimos y subóptimos basado en la calidad de la clasificación a

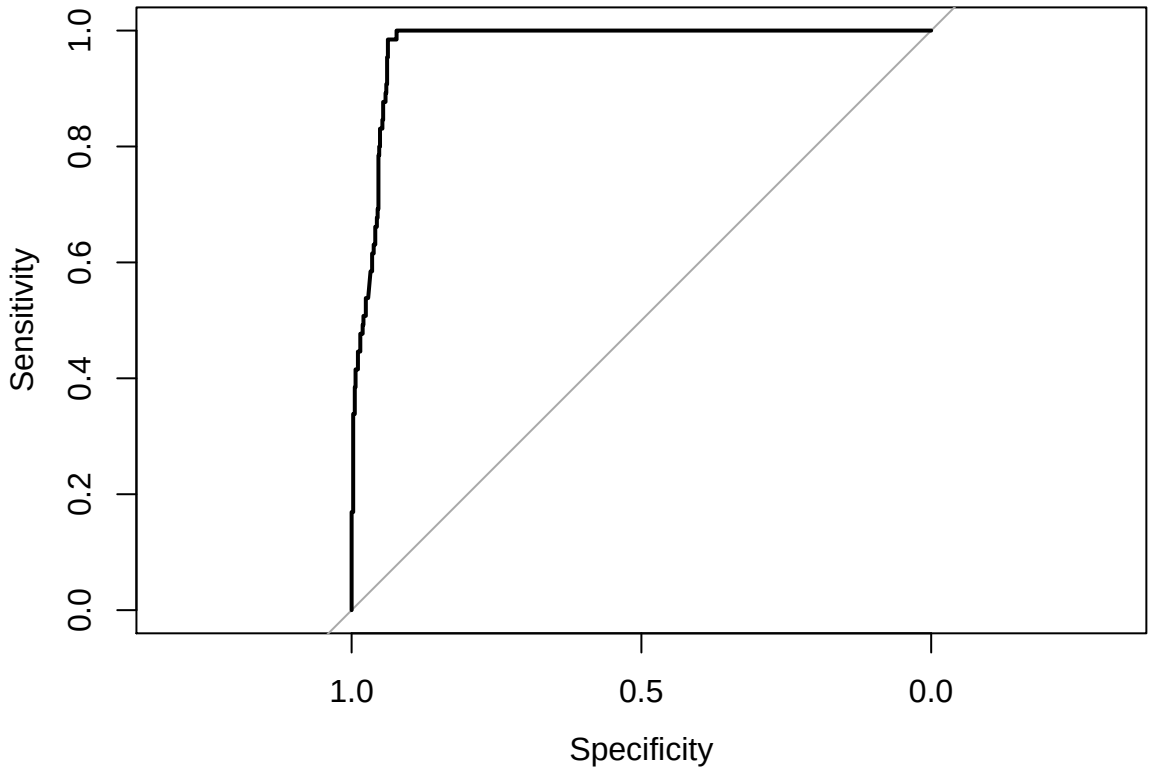
diferentes niveles o umbrales. Para tener una regla objetiva de comparación de las curvas ROC, se calcula el área bajo la curva, simplemente llamada AUROC (area under the ROC).

En general:

- Si AUROC 0,5, el modelo no ayuda a discriminar.
- Si 0,6 AUROC $<$ 0,8, el modelo discrimina de manera adecuada.
- Si 0,8 AUROC $<$ 0,9, el modelo discrimina de forma excelente.
- Si AUROC 0,9, el modelo discrimina de modo excepcional.

Dibujamos la curva ROC:

```
#Gráfico curva ROC
prob=predict(model_regresion_log, data, type="response")
r=roc(data_mlog$Legendary, prob, data=data_mlog)
plot (r)
```



Y a continuación calculamos el área debajo de la curva:

```
#Área debajo de la curva
auc(r)
```

```
## Area under the curve: 0.9745
```

Como el valor obtenido es 0.9745, podemos decir que el modelo de regresión creado discrimina de forma excepcional.

Por último vamos a realizar algunas predicciones con el modelo generado en base a los valores de los atributos de unos pokémon de ejemplo. El modelo predice la probabilidad de que el pokémon introducido sí sea legendario (valor de referencia).

```
#Predicción de nuevos Pokémon
new_pokemon1 <- data.frame(HP=105, Attack=109, Defense=121, Sp..Atk=113, Sp..Def=105, Speed=111)
pr <- predict(model_regresion_log, newdata = new_pokemon1, type = 'response')
pr
```

```
##         1
## 0.8622823
```

En este caso, con los valores introducidos de HP=105, Attack=109, Defense=121, Sp..Atk=113, Sp..Def=105, Speed=111, el pokémon tiene una probabilidad del 86% de ser legendario.

```
#Predicción de nuevos Pokémon
new_pokemon2 <- data.frame(HP=87, Attack=78, Defense=59, Sp..Atk=67, Sp..Def=65, Speed=81)
pr <- predict(model_regresion_log, newdata = new_pokemon2, type = 'response')
pr
```

```
##           1
## 0.001898559
```

Con estos otros valores, HP=87, Attack=78, Defense=59, Sp..Atk=67, Sp..Def=65, Speed=81, el pokémon tiene una probabilidad del 0.2% de ser legendario.

---

5 Representación de los resultados a partir de tablas y gráficas.

A continuación vamos a presentar una serie de gráficas que completan el estudio que ya se ha realizado. Durante el análisis se han presentado tablas y gráficas que apoyaban lo explicado y daban detalle de los procesos, por lo que la información que ahora se presenta es un resumen que completa de manera visual lo ya analizado.

```
ggplot(data, aes(x=data$Type.1))+
  geom_bar(stat="count", width=0.7, fill="steelblue")+
  labs(title="Recuento de pokémon por tipo primario",y="Recuento",x="Tipo pokémon") +
  theme(text = element_text(size=12),
  axis.text.x = element_text(angle=90, hjust=1, vjust = 0.25))
```

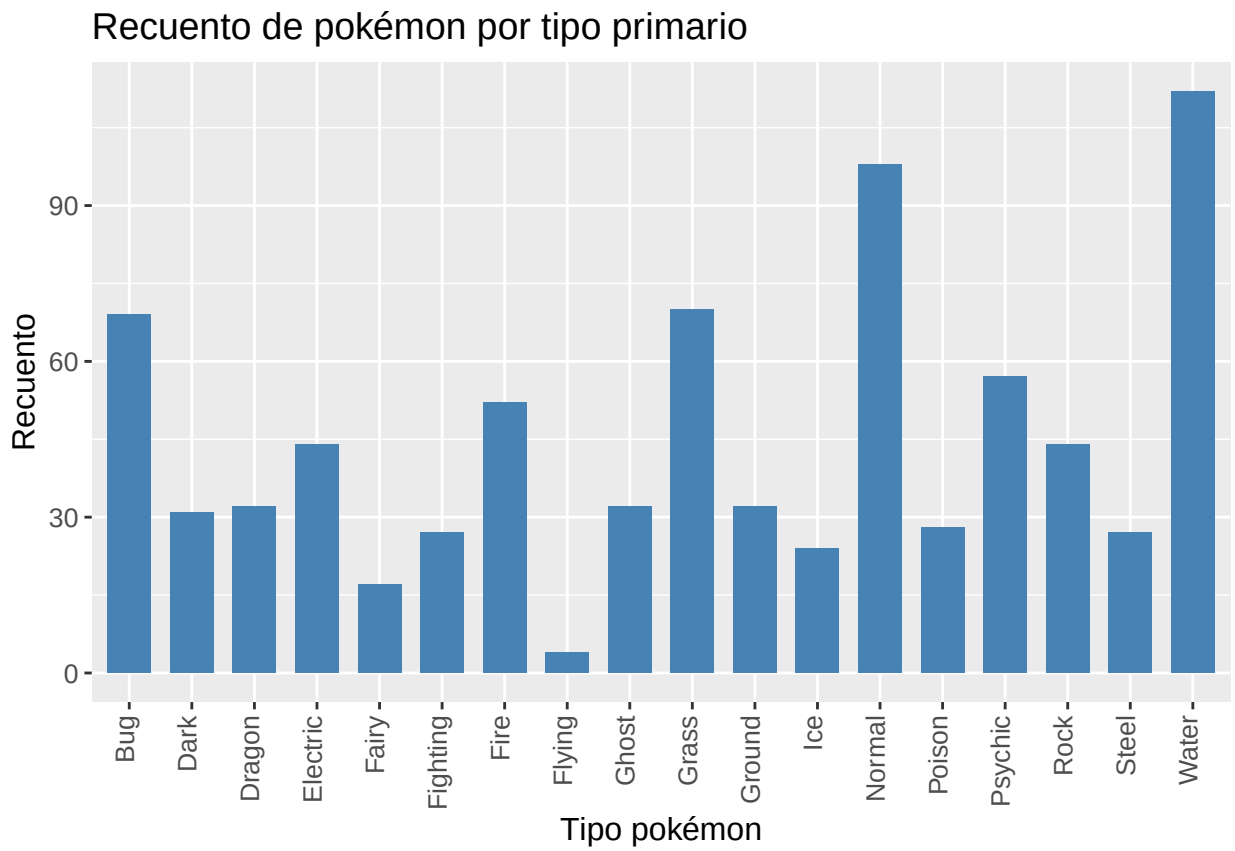
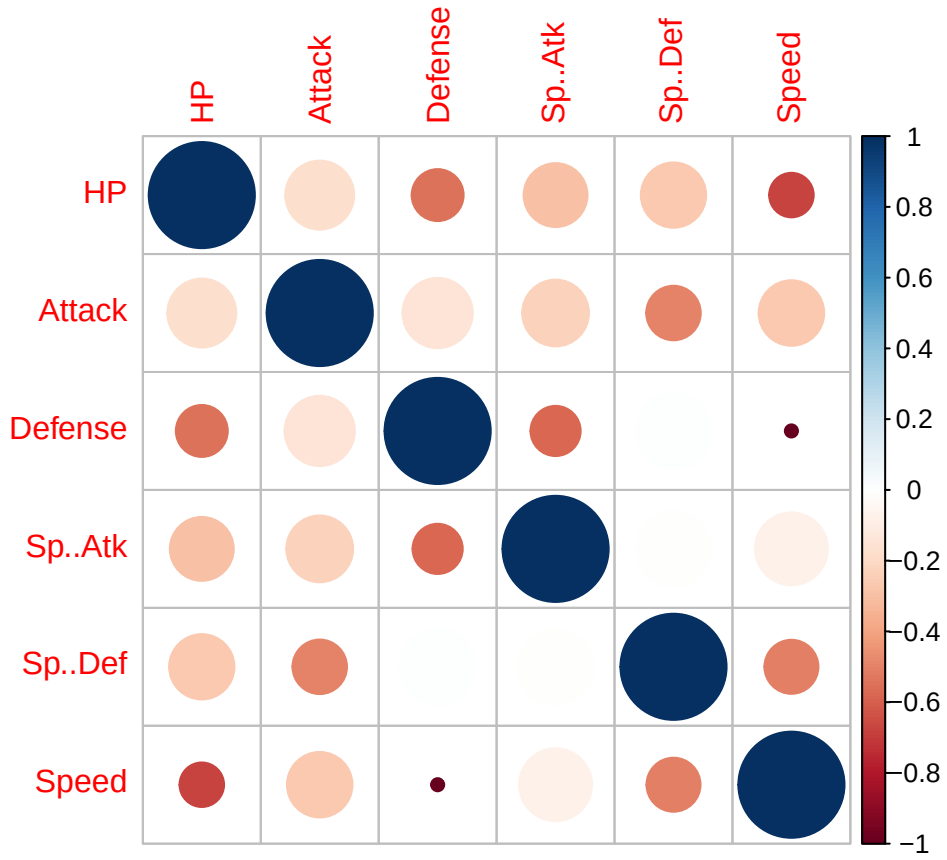


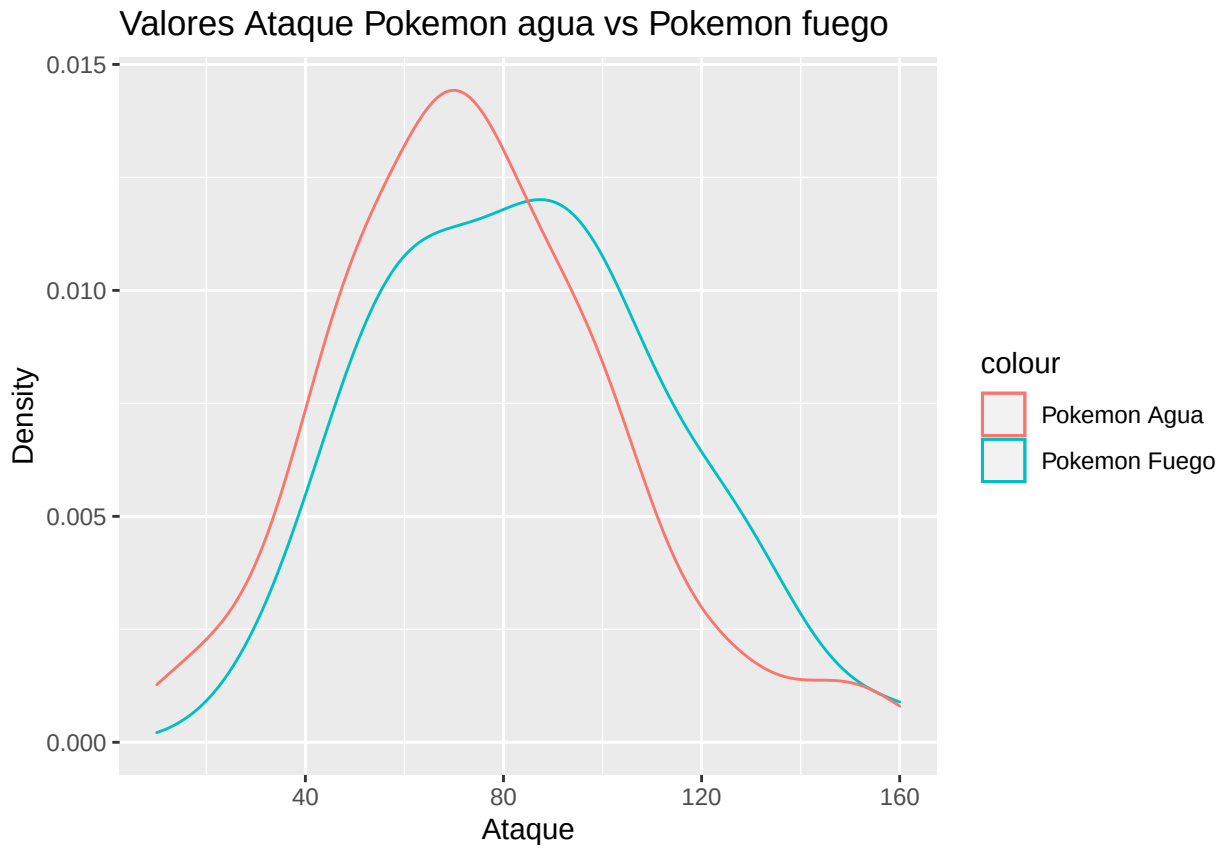
Gráfico resumen con los tipos pokémon presentes en el conjunto de datos. Los más numeros son los de tipo agua, seguidos por los del tipo normal, bug y grass.

```
#Gráfico de correlación de las variables cuantitativas
corrplot(cor(temp) ,method="circle")
```



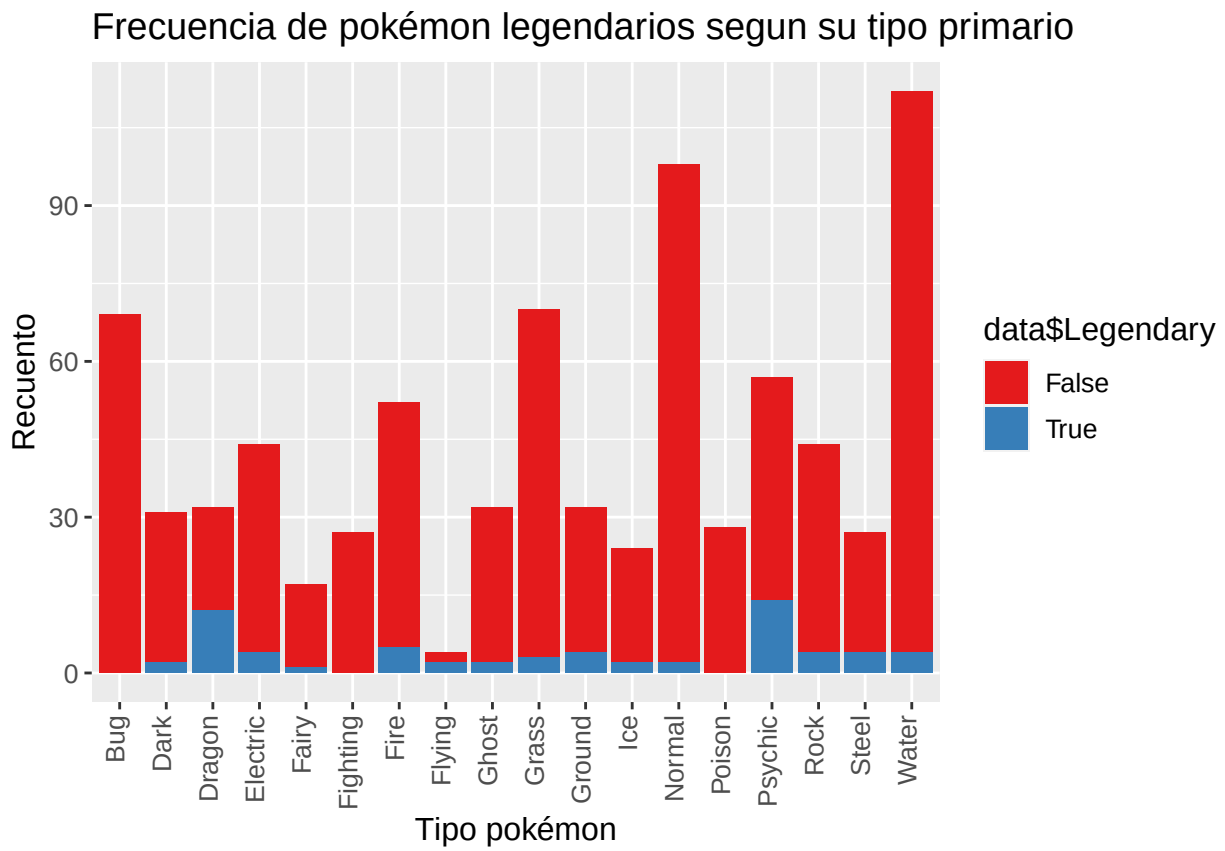
En el gráfico anterior tenemos la matriz de correlación entre las variables cuantitativas del estudio representada gráficamente donde se observa que ninguna de ellas está fuertemente correlacionada.

```
ggplot() +
  geom_density(data=poke_fire, aes(x = poke_fire$Attack, color= "Pokemon Fuego")) +
  geom_density(data=poke_water, aes(x = poke_water$Attack, color= "Pokemon Agua"))+labs(title="Valores
```



En esta representación vemos como la media del ataque de los pokémon de agua se sitúa claramente entre 60 y 80, mientras que la de los de fuego se sitúa más a la derecha, habiendo una mayor densidad de pokemons de fuego con ataque entre 90 y 160. Este gráfico viene a confirmar el contraste de hipótesis anteriormente realizado.

```
ggplot(data,aes(x=data$Type.1,fill=data$Legendary))+geom_histogram(stat="count")+labs(title="Frecuencia
  theme(text = element_text(size=12),
  axis.text.x = element_text(angle=90, hjust=1, vjust = 0.25))
```



En esta gráfica vemos que la familia de pokémon más potente (la que posee más legendarios) son los Psychic seguidos de los Dragon.

---

6 Exportación del archivo final

Una vez finalizado el proceso de limpieza y análisis de nuestro dataset, lo guardamos.

```
write.csv(data, file = "pokemon_clean.csv", row.names = FALSE)
```

---

7 Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

- Se ha realizado el análisis exploratorio, visual y la limpieza de un conjunto de datos sobre Pokémon que consta de 800 registros y 13 variables.
- El dataset es de acceso público y se puede descargar en: https://www.kaggle.com/abcsds/pokemon.
- El conjunto de datos contiene información acerca de las características de cada Pokémon, cuantificando su poder de ataque, defensa, velocidad y salud.
- Durante el proyecto se han realizado las siguientes pruebas: un estudio de correlaciones entre las variables cuantitativas, un contraste de hipótesis entre dos tipos de pokémon y un modelo de regresión logística para la predicción del atributo Legendario.
- Las preguntas de investigación que se planteaban al inicio del proyecto eran: ¿Siguen algún patrón las características de los Pokémon? ¿Están relacionados los atributos de los Pokémon? ¿Los Pokémon de fuego tienen mejor ataque que los de agua? ¿Es posible construir un modelo predictor para identificar Pokémon legendarios?
- Una vez concluido el análisis podemos dar respuesta a las cuestiones planteadas.
- Las características o atributos de los Pokémon no siguen una distribución normal, ni están relacionadas entre ellas.
- Se puede concluir con nivel de confianza del 95% que el ataque de los pokémon de fuego es mejor que el ataque de los pokémon de agua.
- Se ha construido un modelo de regresión logística en el que la variable dependiente es Legendario y las variables independientes HP, Attack, Defense, Sp..Atk, Sp..Def y Speed. Para ver el rendimiento del modelo se ha calculado el área debajo de la curva ROC, obteniendo un valor de 0.9745, lo cual indica que el modelo discrimina de forma excepcional.
- Se han hecho predicciones con el modelo creado para calcular la probabilidad de ser legendarios de dos pokémon diferentes.

---

8 Contribuciones

```
knitr::include_graphics("./tabla_contribuciones.jpeg")
```

Contribuciones	Firmas
Investigación previa	CLS, VNO
Redacción de las respuestas	CLS, VNO
Desarrollo código	CLS, VNO

---

9 Referencias bibliográficas

SUBIRATS MATÉ, Laia, CALVO GONZÁLEZ, Mireia y PÉREZ TRENARD, Diego Oswaldo. *Introducción a la limpieza y análisis de los datos.* [en línea]. Barcelona: UOC, (s/f). Disponible en: https://materials.campus.uoc.edu/daisy/Materials/PID_00265704/pdf/PID_00265704.pdf

Pokemon with stats. [en línea]. [fecha de consulta:17 de mayo 2021]. Disponible en: https://www.kaggle.com/abcsds/pokemon