PRA2-Pokémon species

Carlos Martin Mari

2 de enero de 2020

## Descripción del dataset.¿Por qué es importante y qué pregunta pretende responder?



Pokémons Legendarios

Se trata de un listado de todos los pokémons desde la primera generación hasta la sexta. Se encuentra disponible en kaggle en la siguiente ubicación: <https://www.kaggle.com/abcsds/pokemon>.

### Descripción de los campos

* *#* : Identificador del pokémon dentro del juego (por simplicidad, lo renombramos a *ID Pokémon*)
* *Name*: Nombre del pokémon
* *Type 1*: Tipo del pokémon (agua, fuego, planta, dragón…)
* *Type 2*: Segundo tipo del pokémon (puede ser nulo en pokémons que no tengan más de un tipo)
* *Total*: Suma de los stats numéricos (HP, Attack, Defense, Sp. Attack, Sp. Def, Speed) del pokémon
* *HP*: Puntos de vida
* *Attack*: Puntuación de ataque
* *Defense*: Puntuación de defensa
* *Sp. Atk*: Puntuación de ataque especial
* *Sp. Def*: Puntuacion de defensa especial
* *Speed*: Velocidad
* *Generación*: La generación indica el juego en que apareció. Primera generación sería la original (el primer videojuego para *game boy* azul, rojo y adicionalmente amarillo). Hasta la 6.
* *Legendary*: Flag que nos indica que ese pokémon es o no de tipo legendario

### ¿Qué pregunta pretende resolver?

Se trata de entender mejor cómo son los pokémons, tener datos para ver qué diferencias hay entre generaciones, entre tipologías, e incluso ser capaces de tener datos que nos ayuden a crear el mejor escuadrón de pokémons entre jugadores serios.

En nuestro caso, vamos a tratar de resolver una pregunta sencilla: ¿Podríamos predecir los pokémons legendarios por sus stats? En la práctica, crearemos un clasificador que nos ayude a distinguir entre los pokémons legendarios y no legendarios.

## Integración y selección de los datos a analizar

Veamos primero un resumen de estos campos

summary(input\_data)

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

Lo cual nos indica ya alguna cosa interesante, como por ejemplo que de las 800 líneas, hay *65 correspondientes a pokémons legendarios*… *¿Qué hace que un pokémon sea o no legendario?*

Otra cosa que llama la atención es en el primer campo, el identificador, vemos que nos puede dar problemas a la hora de invocarlo porque la almohadilla tiene otros usos. Así que vamos a renombrarlo. Vemos además que tenemos un total de *721 pokémons diferentes*. Las repeticiones vemos que se tratan de, por ejemplo, las “mega evoluciones”, que no están identificadas en el set de datos y vamos a tratar de discernir también.

Vamos a ver además si en estos campos tenemos valores nulos que deban ser tratados.

na\_count <- sapply(input\_data, function(y) sum(length(which(is.na(y)))))  
  
data.table::data.table("Nombre del campo" = names(na\_count),"#NAs" = na\_count)

## Nombre del campo #NAs  
## 1: ID Pokémon 0  
## 2: Name 0  
## 3: Type 1 0  
## 4: Type 2 386  
## 5: Total 0  
## 6: HP 0  
## 7: Attack 0  
## 8: Defense 0  
## 9: Sp. Atk 0  
## 10: Sp. Def 0  
## 11: Speed 0  
## 12: Generation 0  
## 13: Legendary 0

Vemos que el único que tiene valores nulos es el de la tipología del pokémon. Pero en este caso son valores reales, porque no todos los pokémons tienen dos tipologías. Por simplicidad sólo tendremos en cuenta el primer tipo, porque es el que está documentado como el tipo principal.

Para evitar duplicados en pokémons con características diferentes (por ejemplo, megaevoluciones, pokémons de diferentes tamaños con stats diferenciadas…), vamos a coger sólo la primera línea de cada uno de los pokémons con ID repetida.

### Como los datos ya vienen ordenados convenientemente, vamos a hacer un loop sobre cada fila y comprobaremos si la ID es la misma que el anterior, en cuyo caso lo marcaremos para borrar  
  
transformed\_data <- input\_data  
  
transformed\_data$`Type 2` <- NULL ## quitamos la columna  
  
transformed\_data$RemoveRow <- FALSE  
  
for(i\_row in seq(nrow(transformed\_data))){  
 if(i\_row < 2) next  
  
 if(transformed\_data[i\_row,]$`ID Pokémon` == transformed\_data[i\_row-1,]$`ID Pokémon`)   
 transformed\_data[i\_row,]$RemoveRow <- TRUE  
}  
  
transformed\_data <- subset(transformed\_data, !transformed\_data$RemoveRow)  
  
transformed\_data$RemoveRow <- NULL  
  
paste0("Numero de filas: ", nrow(transformed\_data),", Numero de Pokémons: ", length(unique(transformed\_data$`ID Pokémon`)))

## [1] "Numero de filas: 721, Numero de Pokémons: 721"

Vemos que tras esta limpieza, tenemos efectivamente una tabla con tantas filas como pokémons listados.

## Limpieza de los datos

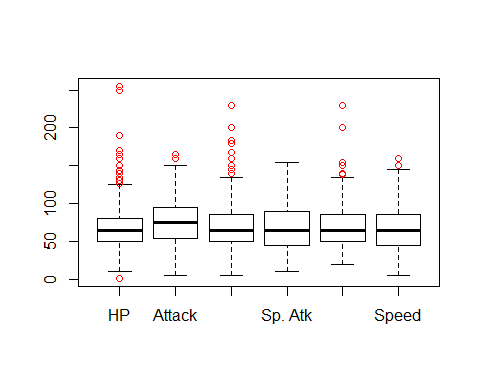
### Valores nulos

Como discutíamos en la sección anterior, el único campo con valores nulos es el de Type 2, ya que se trata de la tipología secundaria del pokémon, no lo tendremos en cuenta.

### Outliers

Vamos a crear un boxplot para buscar los outliers entre los campos cuantitativos del set de datos.

quanti\_fields <- c("HP", "Attack","Defense","Sp. Atk","Sp. Def", "Speed")  
   
boxplot(transformed\_data$HP, transformed\_data$Attack, transformed\_data$Defense, transformed\_data$`Sp. Atk`,transformed\_data$`Sp. Def`, transformed\_data$Speed,  
 names = quanti\_fields,  
 pars=list(outcol="red"))



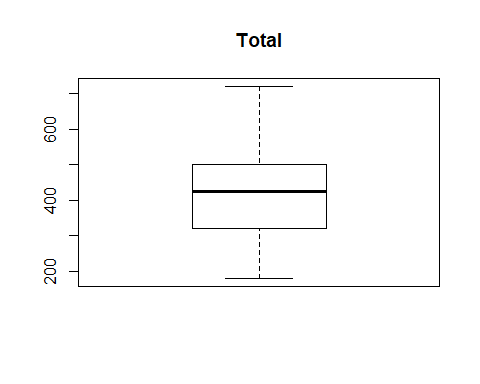
Vemos que tras hacer un boxplot, unos cuantos puntos que quedan fuera. Vamos a echar un vistazo a qué puntos podrían ser:

head(transformed\_data[order(transformed\_data$HP,decreasing = T),])

## # A tibble: 6 x 12  
## `ID Pokémon` Name `Type 1` Total HP Attack Defense `Sp. Atk` `Sp. Def`  
## <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 242 Blis… Normal 540 255 10 10 75 135  
## 2 113 Chan… Normal 450 250 5 5 35 105  
## 3 202 Wobb… Psychic 405 190 33 58 33 58  
## 4 321 Wail… Water 500 170 90 45 90 45  
## 5 594 Alom… Water 470 165 75 80 40 45  
## 6 143 Snor… Normal 540 160 110 65 65 110  
## # … with 3 more variables: Speed <dbl>, Generation <dbl>, Legendary <lgl>

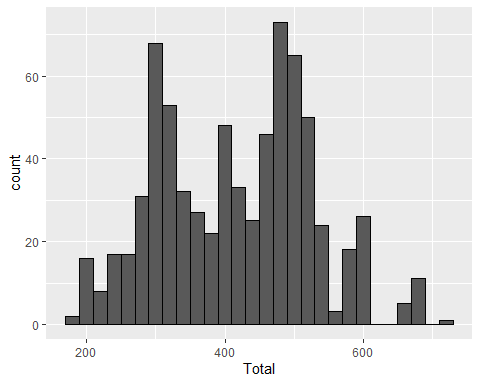
Vemos que estos pokémons con muchísima defensa, a la vez tienen el resto de stats bastante bajas… ¿Es posible que sobre el total haya menos outliers?

boxplot(transformed\_data$Total, main = "Total")



Podemos verlo más en detalle en un histograma:

ggplot2::ggplot(transformed\_data, aes(x=Total)) +  
 geom\_histogram(binwidth=20, colour="black")



Aunque sea sólo por curiosidad, veamos cuáles son esos valores tan altos de puntuación total:

head(transformed\_data[order(transformed\_data$Total,decreasing = T),])

## # A tibble: 6 x 12  
## `ID Pokémon` Name `Type 1` Total HP Attack Defense `Sp. Atk` `Sp. Def`  
## <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 493 Arce… Normal 720 120 120 120 120 120  
## 2 150 Mewt… Psychic 680 106 110 90 154 90  
## 3 249 Lugia Psychic 680 106 90 130 90 154  
## 4 250 Ho-oh Fire 680 106 130 90 110 154  
## 5 384 Rayq… Dragon 680 105 150 90 150 90  
## 6 483 Dial… Steel 680 100 120 120 150 100  
## # … with 3 more variables: Speed <dbl>, Generation <dbl>, Legendary <lgl>

Efectivamente, ¡parece que los *pokémons legendarios* son los valores más limite entre los pokémons! Al menos desde lo alto.

¿Y qué pasa con los valores más bajos?

head(transformed\_data[order(transformed\_data$Total,decreasing = F),])

## # A tibble: 6 x 12  
## `ID Pokémon` Name `Type 1` Total HP Attack Defense `Sp. Atk` `Sp. Def`  
## <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 191 Sunk… Grass 180 30 30 30 30 30  
## 2 298 Azur… Normal 190 50 20 40 20 40  
## 3 401 Kric… Bug 194 37 25 41 25 41  
## 4 10 Cate… Bug 195 45 30 35 20 20  
## 5 13 Weed… Bug 195 40 35 30 20 20  
## 6 265 Wurm… Bug 195 45 45 35 20 30  
## # … with 3 more variables: Speed <dbl>, Generation <dbl>, Legendary <lgl>

Parece que los pokémon de tipo “hierba” y “bicho” son de los que tienen peores stats. No parece tampoco que pueda ser incorrecto, al final son los primeros pokémons que nos encontramos en el juego, pero rara vez se utilizan en muy grandes niveles por algo.

Por tanot, en resumen, no identificamos valores extremos que deban ser tratados porque los pokémons se han diseñado para que tengan stats diferenciados uno a uno.

## Análisis de los datos

Vamos a tener en cuenta para este apartado que el objetivo es esutdiar las diferencias entre la población de pokémons legendarios y no legendarios.

### Test de normalidad

Vamos a aplicar el test de Shappiro Wilk sobre las variables cuantitativas.

Probémoslo primero para el total:

cat("Legendary Pokémons:")

## Legendary Pokémons:

legendary\_data <- subset(transformed\_data, transformed\_data$Legendary == T)  
shapiro.test(legendary\_data$Total)

##   
## Shapiro-Wilk normality test  
##   
## data: legendary\_data$Total  
## W = 0.75685, p-value = 2.424e-07

cat("Non Legendary Pokémons:")

## Non Legendary Pokémons:

non\_legendary\_data <- subset(transformed\_data, transformed\_data$Legendary == F)  
shapiro.test(non\_legendary\_data$Total)

##   
## Shapiro-Wilk normality test  
##   
## data: non\_legendary\_data$Total  
## W = 0.96314, p-value = 5.483e-12

Debería tener un p-value > 0.05, que no es el caso.

Miremos ahora para el resto de variables cuantitativas:

pvalue\_shapirotest <- function(df){  
 shapiro.test(df)$p.value  
}  
cat("Legendary Pokémons:\n")

## Legendary Pokémons:

apply(legendary\_data[,quanti\_fields], 2, pvalue\_shapirotest)

## HP Attack Defense Sp. Atk Sp. Def Speed   
## 0.078407400 0.322302083 0.006845207 0.091960782 0.018099079 0.012586851

cat("\nNon Legendary Pokémons:\n")

##   
## Non Legendary Pokémons:

apply(non\_legendary\_data[,quanti\_fields], 2, pvalue\_shapirotest)

## HP Attack Defense Sp. Atk Sp. Def Speed   
## 1.345819e-21 5.942148e-05 1.244077e-16 4.669263e-09 6.968816e-14 3.746209e-06

Vemos que variable por variable, para los pokémons legendarios quizá sí que podríamos asumir que se distribuye normalmente. No así para los pokémons no legendarios.

### Test de homocedasticidad

Aunque quizá no tenga mucho sentido realizar el test sin que cumpla la normalidad, lo realizaremos por propósitos didácticos. Realizaremos el test de Bartlet.

Probémoslo primero para el total:

bartlett.test(transformed\_data$Total, g = paste0(transformed\_data$Legendary))

##   
## Bartlett test of homogeneity of variances  
##   
## data: transformed\_data$Total and paste0(transformed\_data$Legendary)  
## Bartlett's K-squared = 33.842, df = 1, p-value = 5.979e-09

Como era de esperar, los datos no pasan el test de homocedasticidad.

### Análisis de los datos

Como los datos no muestran ser homogéneos en las varianzas ni se distribuyen normalmente, vamos a tener que hacer análisis más robustos. HAn sido barajadas pruebas estadísticas no paramétricas equivalentes a, por ejemplo, ANOVA, como el test de KRuskall-Wallis, pero estos tests siguen esperando que nuestros datos estén distribuidos con una función de densidad parecida, que sólo con ver las diferencias en el test de normalidad ya vemos que no.

Por ello vamos a evitar utilizar tests de hipótesis y regresiones, que parece que no sean adecuadas para nuestros datos.

#### Método 1: Reducción dimensional

Vamos a ver si reduciendo las dimensiones, podemos gráficamente distinguir a los pokémons legendarios del resto. Vamos a probar con un análisis de componentes princiupales

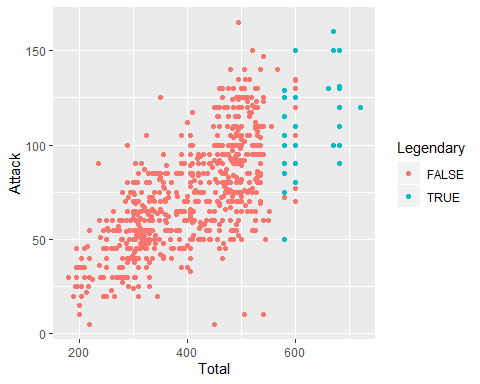
quanti\_data <- transformed\_data[,c(quanti\_fields)]  
  
poke\_pca <- prcomp(quanti\_data)  
  
summary(poke\_pca)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6  
## Standard deviation 45.1232 29.7676 25.8174 22.6353 18.83015 14.58333  
## Proportion of Variance 0.4361 0.1898 0.1428 0.1098 0.07595 0.04556  
## Cumulative Proportion 0.4361 0.6260 0.7687 0.8785 0.95444 1.00000

Vemos que con las dos primeras componentes ya podemos explicar un 62% de la varianza de los datos.

Echemos un vistazo a ver qué tal podemos distinguir entre pokémons legendarios y no legendarios:

# ggplot(df\_pca,aes(x=PC1,y=PC2,color=Legendary )) + geom\_point()  
ggplot(transformed\_data,aes(x=Total,y=Attack,color=Legendary )) + geom\_point()



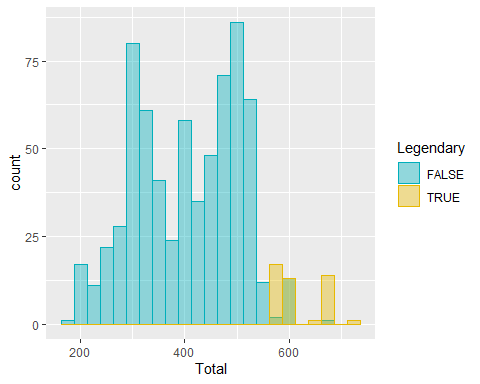
Vemos que con la primera componente tenemos una muy buena capacidad de predicción. ¿Cuál sería la matriz de confusión?

transformed\_data$LegendaryMeth1a <- FALSE  
  
transformed\_data[poke\_pca$x[,1] < -60,]$LegendaryMeth1a <- TRUE  
  
table(transformed\_data$Legendary, transformed\_data$LegendaryMeth1a)

##   
## FALSE TRUE  
## FALSE 659 16  
## TRUE 0 46

Antes hemos visto que entre los pokémons con un valor de stats total mayor, parecía que podía haber más legendarios. ¿Por qué no lo comparamos?

ggplot2::ggplot(transformed\_data, aes(x=Total, color = Legendary)) +  
 geom\_histogram(aes(color = Legendary, fill = Legendary),   
 position = "identity", binwidth = 25, alpha = 0.4) +  
 scale\_color\_manual(values = c("#00AFBB", "#E7B800")) +  
 scale\_fill\_manual(values = c("#00AFBB", "#E7B800"))



Por el método utilizando el total, habríamos tenido:

transformed\_data$LegendaryMeth1b <- FALSE  
  
transformed\_data[transformed\_data$Total > 575,]$LegendaryMeth1b <- TRUE  
  
table(transformed\_data$Legendary, transformed\_data$LegendaryMeth1b)

##   
## FALSE TRUE  
## FALSE 660 15  
## TRUE 0 46

Es decir, estaríamos cogiendo 15 falsos positivos vs 16.

Aquí además estamos viendo un objetivo secundario cumplido: ¿Podríamos identificar estos 15 pokémons que parecen legendarios pero no lo son? Seguro que son más fáciles de encontrar en el juego, pero siguen siendo muy buenos.

subset(transformed\_data, transformed\_data$LegendaryMeth1b == TRUE & transformed\_data$Legendary == FALSE)

## # A tibble: 15 x 14  
## `ID Pokémon` Name `Type 1` Total HP Attack Defense `Sp. Atk` `Sp. Def`  
## <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 149 Drag… Dragon 600 91 134 95 100 100  
## 2 151 Mew Psychic 600 100 100 100 100 100  
## 3 248 Tyra… Rock 600 100 134 110 95 100  
## 4 251 Cele… Psychic 600 100 100 100 100 100  
## 5 289 Slak… Normal 670 150 160 100 95 65  
## 6 373 Sala… Dragon 600 95 135 80 110 80  
## 7 376 Meta… Steel 600 80 135 130 95 90  
## 8 445 Garc… Dragon 600 108 130 95 80 85  
## 9 488 Cres… Psychic 600 120 70 120 75 130  
## 10 490 Mana… Water 600 100 100 100 100 100  
## 11 635 Hydr… Dark 600 92 105 90 125 90  
## 12 647 Keld… Water 580 91 72 90 129 90  
## 13 648 Melo… Normal 600 100 77 77 128 128  
## 14 649 Gene… Bug 600 71 120 95 120 95  
## 15 706 Good… Dragon 600 90 100 70 110 150  
## # … with 5 more variables: Speed <dbl>, Generation <dbl>, Legendary <lgl>,  
## # LegendaryMeth1a <lgl>, LegendaryMeth1b <lgl>

¿No era Mew un pokémon legendario? Aparentemente, aunque Mewtwo (su clon) sí que es legendario, no es así Mew que se trata de un pokémon míticov <https://bulbapedia.bulbagarden.net/wiki/Mythical_Pok%C3%A9mon>, al igual que otros pokémons de esta lista: Celebi y Manaphy. Son descritos de forma semejante a los legendarios, como pokémons rarísimos y muy poderosos.

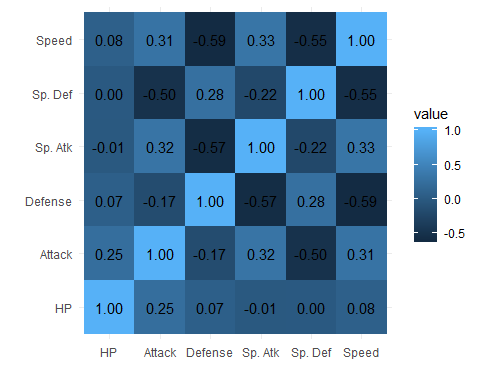
Mirando los datos del listado de míticos, aparentemente hay otros pokémons que son a la vez míticos y legendarios como Volcanion, pero está claro que es un factor de confusión. Pero no lo sería para la mayoría de estos pokémons.

#### Método 2: Correlaciones entre variables

Vamos a mirar si hay alguna diferencia entre las correlaciones entre variables de los pokémons legendarios o no legendarios.

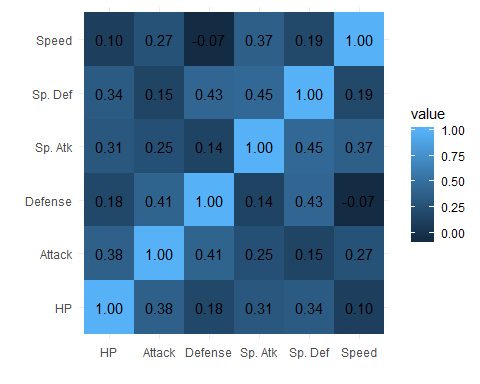
Primero miremos los pokémons legendarios:

matriz <- cor(subset(quanti\_data,transformed\_data$Legendary == T))  
m <- melt(matriz)  
m$value\_lab<-sprintf('%.2f',m$value)  
  
ggplot(m, aes(Var2, Var1, fill = value, label=value\_lab),color='blue') +   
 geom\_tile() +   
 geom\_text() +  
 xlab('')+  
 ylab('')+  
 theme\_minimal()



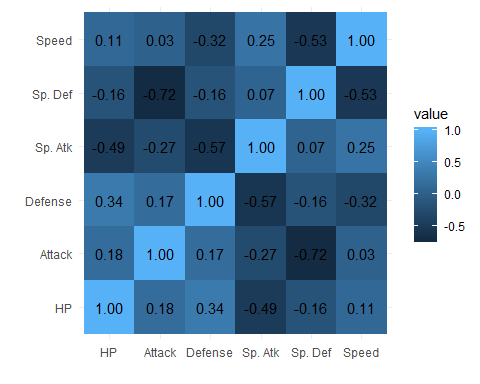
y comparemos con los no legendarios:

matriz <- cor(subset(quanti\_data,transformed\_data$Legendary == F))  
m <- melt(matriz)  
m$value\_lab<-sprintf('%.2f',m$value)  
  
ggplot(m, aes(Var2, Var1, fill = value, label=value\_lab),color='blue') +   
 geom\_tile() +   
 geom\_text() +  
 xlab('')+  
 ylab('')+  
 theme\_minimal()



¡Es muy interesante! Los pokémons legendarios parece que nos obligan mucho más a tomar concesiones: Si quieres un pokémon con buena salud, deberás comprometer tu defensa en un legendario, pero con los no legendarios no. Una razón para que esto ocurra, es porque al tener en general stats muy altos nos obligan a comprometer o un pokémon con muy buena defensa y ataque podría resultar demasiada ventaja competitiva. Comprobémoslo entre los pokémons con unos stats totales más altos (>550):

matriz <- cor(subset(quanti\_data,transformed\_data$Legendary == F & transformed\_data$Total > 550))  
m <- melt(matriz)  
m$value\_lab<-sprintf('%.2f',m$value)  
  
ggplot(m, aes(Var2, Var1, fill = value, label=value\_lab),color='blue') +   
 geom\_tile() +   
 geom\_text() +  
 xlab('')+  
 ylab('')+  
 theme\_minimal()



Resulta interesante, la suposición de que el valor de la suma total de stats podría ser un factor a tener en cuenta al mirar la matriz de correlación era acertada. Una vez filtrado por magnitudes semejantes, vemos que no sólo la primera idea de que los pokémons legendarios eran más exigentes a la hora de pedirnos concesiones era errónea… sino que resulta opuesta. Parece que los pokémons legendarios cumplen por ahora dos características:

1. Un volumen total de stats elevado
2. Los stats están repartidos más equitativamente. Cualitativamente vemos que nos exige menos entre los pares de ataque-ataque especial y defensa-defensa especial que es donde probablemente haya más diferencia entre pokémons de volumen de stats semejante pero legendario vs no-legendario.

#### Método 3: Creamos un clasificador

Con toda esta información, añadimos a las variables cualitativas tres nuevos campos: 1. La suma del total 2. Defensa/Defensa Especial 3. Ataque/Ataque Especial

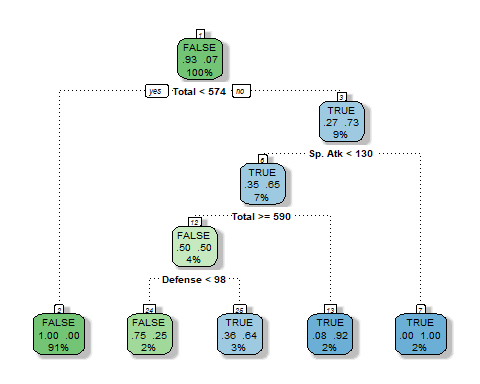
Parece que estas tres variables deberían darnos buen resultado para hacer un clasificador numérico.

quanti\_data$Total <- transformed\_data$Total  
quanti\_data$Legendary <- transformed\_data$Legendary  
quanti\_data$ATAvsSp <- quanti\_data$Attack / quanti\_data$`Sp. Atk`  
quanti\_data$DEFvsSp <- quanti\_data$Defense / quanti\_data$`Sp. Def`  
list\_of\_pokemons <- transformed\_data$Name

Ahora vamos a hacer un sencillo clasificador bayesiano.

Primero, vamos a partir el set de datos en uno de entrenamiento y uno de test

set.seed(123123) #para asegurarnos la reproducibilidad  
smp\_size <- floor(0.70 \* nrow(quanti\_data))  
train\_ind <- sample(seq\_len(nrow(quanti\_data)), size = smp\_size)  
  
train <- quanti\_data[train\_ind, ]  
test <- quanti\_data[-train\_ind, ]  
  
fit <- rpart(Legendary ~ ., data= train, method = "class")  
fancyRpartPlot(fit,sub = "")



Y comprobamos la matriz de confusión

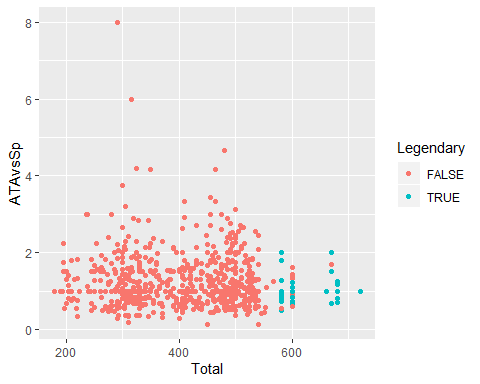
table(test$Legendary,predict(fit,newdata = test, type = "class"))

##   
## FALSE TRUE  
## FALSE 202 2  
## TRUE 1 12

Vemos que el clasificador a la hora de hacer el ajuste, no ha escogido las dos nuevas *features* que hemos creado. Por propósito didáctico, vamos a forzarle a que lo utilize a ver si el resultado es mucho peor.

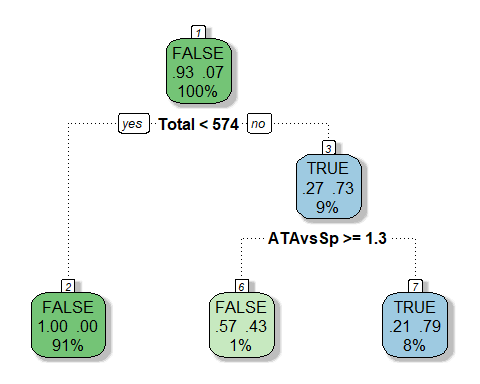
Primero veamos una gráfica de dispersión, a ver qué aspecto tiene con dos de esas variables:

ggplot(quanti\_data,aes(x=Total,y=ATAvsSp,color=Legendary )) + geom\_point()



Veamos ahora qué tal funciona el clasificador

set.seed(123123) #para asegurarnos la reproducibilidad  
smp\_size <- floor(0.70 \* nrow(quanti\_data))  
train\_ind <- sample(seq\_len(nrow(quanti\_data)), size = smp\_size)  
  
train <- quanti\_data[train\_ind, c("Total","ATAvsSp","DEFvsSp","Legendary") ]  
test <- quanti\_data[-train\_ind, c("Total","ATAvsSp","DEFvsSp","Legendary")]  
  
fit <- rpart(Legendary ~ ., data= train, method = "class")  
fancyRpartPlot(fit,sub = "")



table(test$Legendary,predict(fit,newdata = test, type = "class"))

##   
## FALSE TRUE  
## FALSE 201 3  
## TRUE 1 12

Vemos que el performance es prácticamente igual. Si complicáramos algo más el clasificador, es posible que uno de estos ratios entre ataque y ataque especial ya bastara además del total para hacer un buen clasificador, que además al tener menos pasos en el árbol podríamos considerarlo como mejor para escalar.

## Conclusiones

¿Hemos obtenido el resultado buscado? En mi opinión sí, aunque siempre se podría realizar un análisis más en profundidad, comprobar distintos clasificadores, por ejemplo. Hemos aprendido qué hace que un pokémon sea legendario… No solo tienen muy buenas stats en general, sino que te fuerzan menos a tomar compromisos. Es decir. ¿que hace que un pokémon sea legendario? Su *potencia* y su *versatilidad*.