Proyecto NBA

Alvaro Francisco Ruiz Cornejo, Carlos García Vázquez, Carlos García-Mauriño Villanueva

13/11/2021

1. PREPROCESADO

En primer lugar, se cargan las librerías necesarias para el desarrollo del trabajo.

```
library(tidyverse)
library(readxl)
```

A continuación, se cargan los datasets que se van a utilizar y comprobamos las cinco primeras líneas de cada uno de ellos para conocer su forma y datos principales.

```
NBAStats <- read_excel("data/NBAStats.xlsx")
NBAStats2 <- read_csv("data/Stats2.csv")
```

```
head(NBAStats, 5)
```

```
## # A tibble: 5 x 30
##      Rk Player Pos     Age Tm        G    GS MP    FG    FGA   `FG%` `3P`  `3PA`
##   <dbl> <chr>  <chr> <dbl> <chr> <dbl> <dbl> <chr> <chr> <chr> <chr> <chr> <chr>
## 1     1 Preci~ PF       21 MIA      61     4 12.1  2.0   3.7   .544  0.0   0.0
## 2     2 Jayle~ PG       24 MIL       7     0 2.6   0.1   1.1   .125  0.0   0.3
## 3     3 Steve~ C        27 NOP      58    58 27.7  3.3   5.3   .614  0.0   0.1
## 4     4 Bam A~ C        23 MIA      64    64 33.5  7.1   12.5  .570  0.0   0.1
## 5     5 LaMar~ C        35 TOT      26    23 25.9  5.4   11.4  .473  1.2   3.1
## # ... with 17 more variables: 3P% <chr>, 2P <chr>, 2PA <chr>, 2P% <chr>,
## #   eFG% <chr>, FT <chr>, FTA <chr>, FT% <chr>, ORB <chr>, DRB <chr>,
## #   TRB <chr>, AST <chr>, STL <chr>, BLK <chr>, TOV <chr>, PF <chr>, PTS <chr>
```

```
head(NBAStats2, 5)
```

```
## # A tibble: 5 x 29
##      Rk Player     Pos     Age Tm        G    MP   PER `TS%` `3PAr`   FTr `ORB%`
##   <dbl> <chr>      <chr> <dbl> <chr> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>
## 1     1 Precious ~ PF       21 MIA      61   737  14.2 0.55   0.004 0.482   11.5
## 2     2 Jaylen Ad~ PG       24 MIL       7    18  -6.5 0.125  0.25  0        0
## 3     3 Steven Ad~ C        27 NOP      58  1605  15.1 0.596  0.01  0.438   14.4
## 4     4 Bam Adeba~ C        23 MIA      64  2143  22.7 0.626  0.01  0.443    7.7
## 5     5 LaMarcus ~ C        35 TOT      26   674  15.7 0.556  0.27  0.159    3
## # ... with 17 more variables: DRB% <dbl>, TRB% <dbl>, AST% <dbl>, STL% <dbl>,
## #   BLK% <dbl>, TOV% <dbl>, USG% <dbl>, ...20 <lgl>, OWS <dbl>, DWS <dbl>,
## #   WS <dbl>, WS/48 <dbl>, ...25 <lgl>, OBPM <dbl>, DBPM <dbl>, BPM <dbl>,
## #   VORP <dbl>
```

El siguiente paso sería *limpiar los datos de la tabla.* Esto es, eliminar columnas con valores nulos que han sido mal importadas. La primera tabla no contiene columnas vacias de datos, la segunda sí. Por tanto, se procede a realizar un limpiado del segundo dataset:

```
NBAStats2 <- NBAStats2 %>%
  select(-...20,-...25)
```

Para poder trabajar con ambas tablas, se ha realizado un *merge*, renonmbrando algunas columnas para una mejor compresión y accesibilidad, así como eliminando algunas variables que no aportan especial interés de estudio.

```
TotalNBA <- merge(x = NBAStats, y = NBAStats2, by = c("Rk","Tm"))

NBADef <-  TotalNBA %>%
  select(-GS, -PF, -'eFG%', -Player.y, -Pos.y, -Age.y, -G.y, -MP.y, -FTr, -'WS/48',
         -OBPM, -DBPM, -VORP)

NBADef <- rename(NBADef, Player = Player.x, Pos = Pos.x, Age = Age.x, G = G.x, MP = MP.x)
```

En algunos casos, nos encontramos con jugadores que han jugado en dos posiciones (por ejemplo, SF-PF), por lo que únicamente se va a dejar la primera que supuestamente, es la principal (SF).

```
NBADef <- NBADef %>%
  separate(col = Pos, sep = "-", into = c("Pos", NULL))
```

Además, se comprueba como en el dataset hay otra variable muy importante como lo es el nombre del jugador, que se encuentra separada por el caracter "–" de una abreviatura extraña (por ejemplo, Gary Clark–clarkga01) que realmente ni aporta nada ni resulta estético para futuros estudios que devuelvan como salida el nombre del jugador. Por tanto, eliminamos la abreviatura en todos los campos de la variable, dejando así únicamente el nombre del jugador.

```
NBADef <- NBADef %>%
  separate(col = Player, sep = '--', into = c("Player", NULL))
```

También es necesario eliminar aquellas observaciones cuyo equipo es *TOT* (significa que ha jugado para dos o más equipos diferentes ese año). Por tanto, esa observación contiene la suma de los valores obtenido para los equipos para los que ha jugado (suma de puntos, suma de rebotes, suma de...). Esto no resulta útil para el estudio ya que desvirtuaría algunas estadísticas por tener observaciones con sumas de otras.

```
NBADef <- NBADef %>%
  filter(Tm != "TOT")
```

Por último, los valores nulos se cambian a 0 a la vez que algunas variables de tipo caracter se convierten en factor para tratarlas en el análisis.

```r
# Valores nulos a 0
NBADef <-  mutate_all(NBADef, ~ replace(., is.na(.), 0))

# Cambio de las variables de tipo caracter a factor
catvars = c(2,4)
NBADef[c(catvars)] = lapply(NBADef[c(catvars)] , factor)

# El resto de variables (algunos datos numéricos vienen como caracter) a numéricas
numericvars <- c(5:42)
NBADef[c(numericvars)] <- lapply(NBADef[c(numericvars)], as.numeric)
```

Antes de empezar con los cálculos, vamos a comprobar alguna información básica acerca del dataset definitivo que se va a utilizar para el resto del trabajo.

```r
# Dimensiones del dataset, número de filas y columnas respectivamente
dim(NBADef)
```

```
## [1] 626  42
```

```r
# ¿Hay algún dato ausente?
any(is.na(NBADef))
```

```
## [1] FALSE
```

```r
# ¿Cuáles son las variables y de qué tipo?
head(summary.default(NBADef), 10)
```

```
##        Length Class  Mode
## Rk     626    -none- numeric
## Tm     626    factor numeric
## Player 626    -none- character
## Pos    626    factor numeric
## Age    626    -none- numeric
## G      626    -none- numeric
## MP     626    -none- numeric
## FG     626    -none- numeric
## FGA    626    -none- numeric
## FG%    626    -none- numeric
```

```r
# str(NBADef) Da información más completa
# summary(NBADef) Da información más específica
```

Con todas estas modificaciones, ya se tiene un dataset limpio, con variables útiles y accesibles e información completa para comenzar la siguiente parte del trabajo.

2. RESULTADOS

En primer lugar, es importante mencionar que las estadísticas están ponderadas por partido. Esto es, son las estadísticas totales de la temporada dividida entre el número de partidos que un jugador ha disputado para un determinado equipo.

2.1. Uso de *dplyr* y *ggplot*

Como primer acercamiento, se van a hacer algunas operaciones básicas de *dplyr* (select, arrange, filter, summarise...) sobre la base de datos con el objetivo de extraer alguna información que pueda resultar como estadísticas interesantes.

- **Media de edad por equipos**

Agrupando por equipos, se puede hacer la media de edad de los jugadores que lo componen, descubriendo así los 5 equipos con más jugadores veteranos (media más alta).

```
NBADef %>%
  group_by(Tm) %>%
  summarise(mediaEdad = round(mean(Age),2)) %>%
  arrange(desc(mediaEdad)) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   Tm    mediaEdad
##   <fct>     <dbl>
## 1 MIA        28.1
## 2 LAL        28
## 3 LAC        27.6
## 4 MIL        27.1
## 5 BRK        27
```

- **Jugadores que han jugado en más equipos diferentes durante la temporada**

Agrupando por jugador, es posible contar los equipos diferentes en los que éstos han jugado en la temporada de estudio. Ordenándolos de mayor a menor, queda:

```
NBADef %>%
  group_by(Player) %>%
  summarise(equiposDiferentes = n_distinct(Tm)) %>%
  arrange(desc(equiposDiferentes)) %>%
  head(8)
```

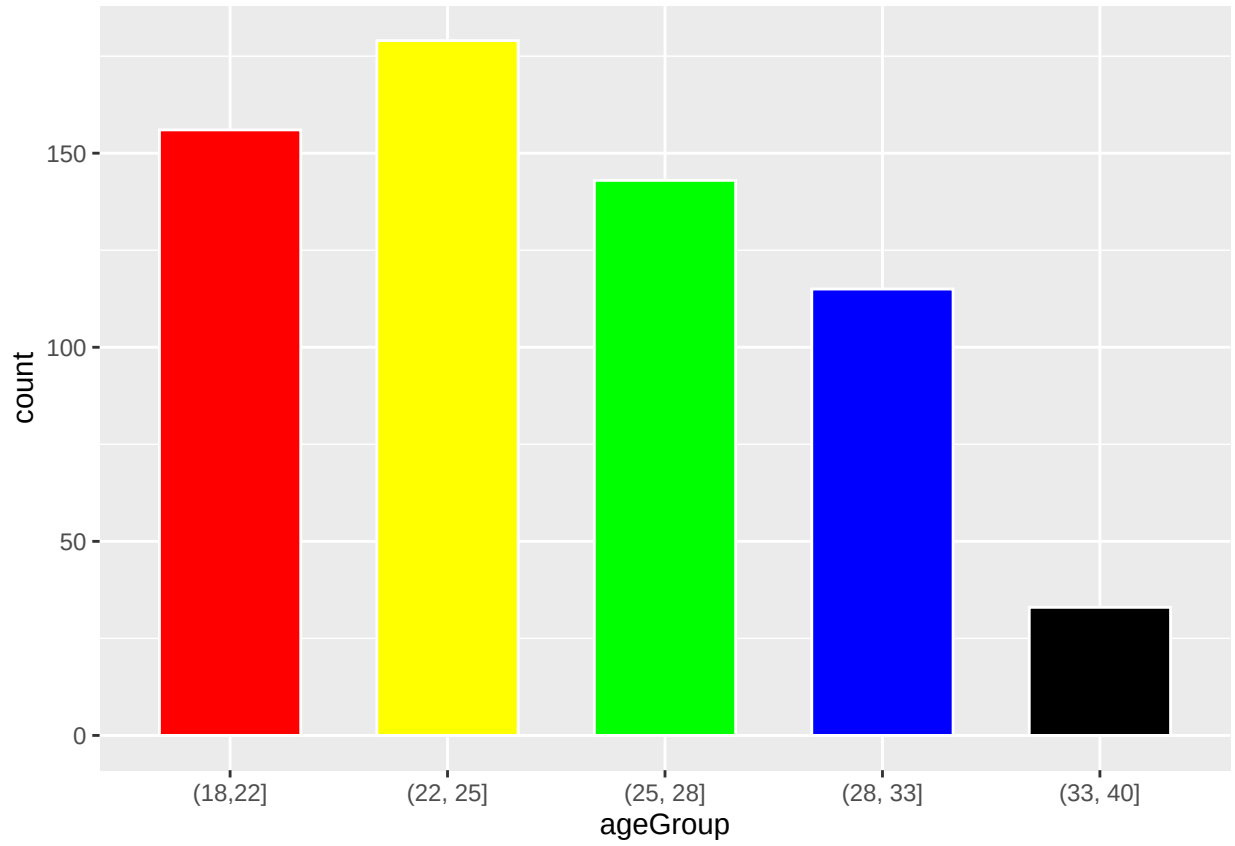
```
## # A tibble: 8 x 2
##   Player           equiposDiferentes
##   <chr>                        <int>
```

```
## 1 Damian Jones                  3
## 2 Gary Clark                    3
## 3 Ignas Brazdeikis              3
## 4 Moritz Wagner                 3
## 5 Norvel Pelle                  3
## 6 Rodions Kurucs                3
## 7 Victor Oladipo                3
## 8 Aaron Gordon                  2
```

Asimismo, para el estudio de variables no numéricas, podemos establecer grupos de edades para realizar algunos gráficos y comprobar su distribución en la temporada 2020.

```
NBADef <- NBADef %>%
  mutate(ageGroup = cut(Age, breaks = c(18, 22, 25, 28, 33, 40),
                        labels = c("(18,22]", "(22, 25]", "(25, 28]",
                                   "(28, 33]", "(33, 40]"),
                        include.lowest = FALSE, right = TRUE))
```

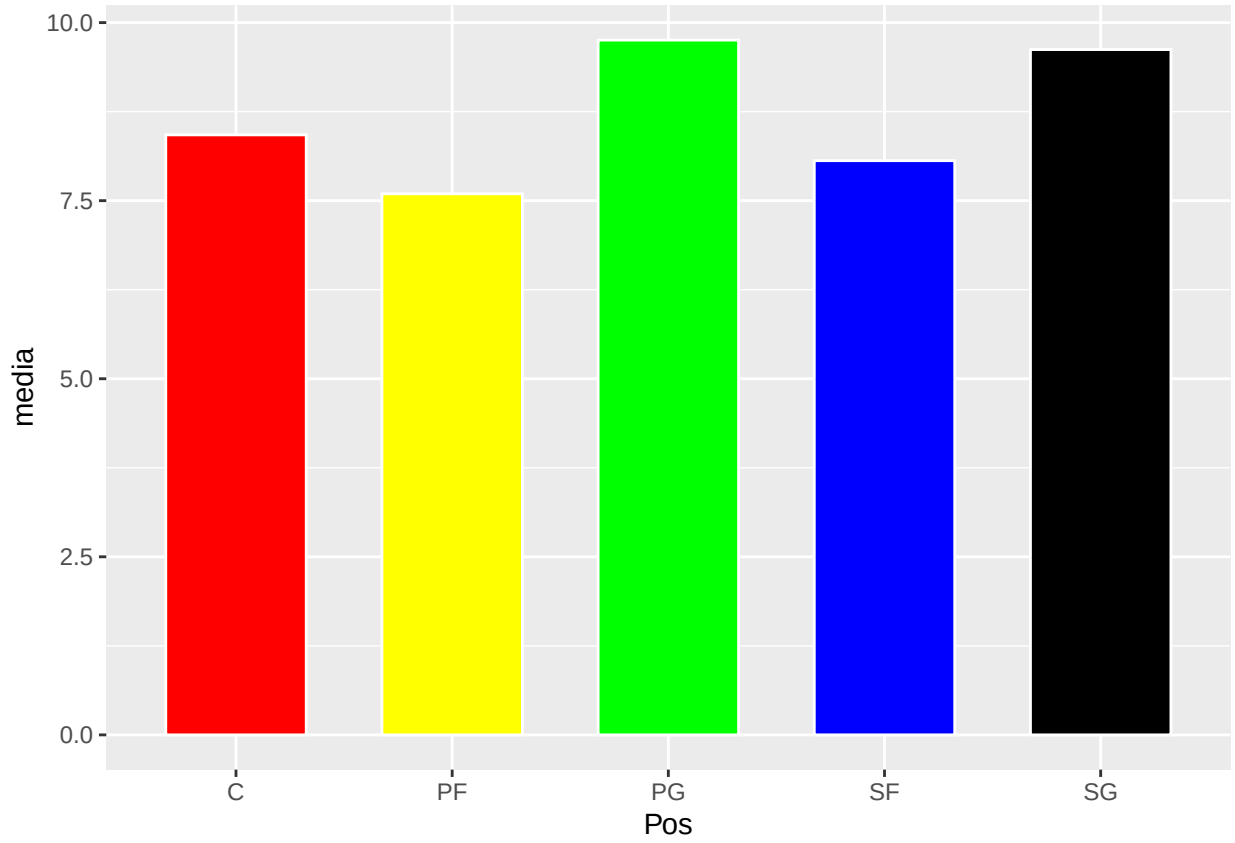
```
ggplot(data = NBADef) +
  geom_bar(mapping = aes(x = ageGroup), colour = "white",
           fill = c("red", "yellow", "green", "blue", "black"),
           width = 0.65)
```



- **Media de puntos por posición**

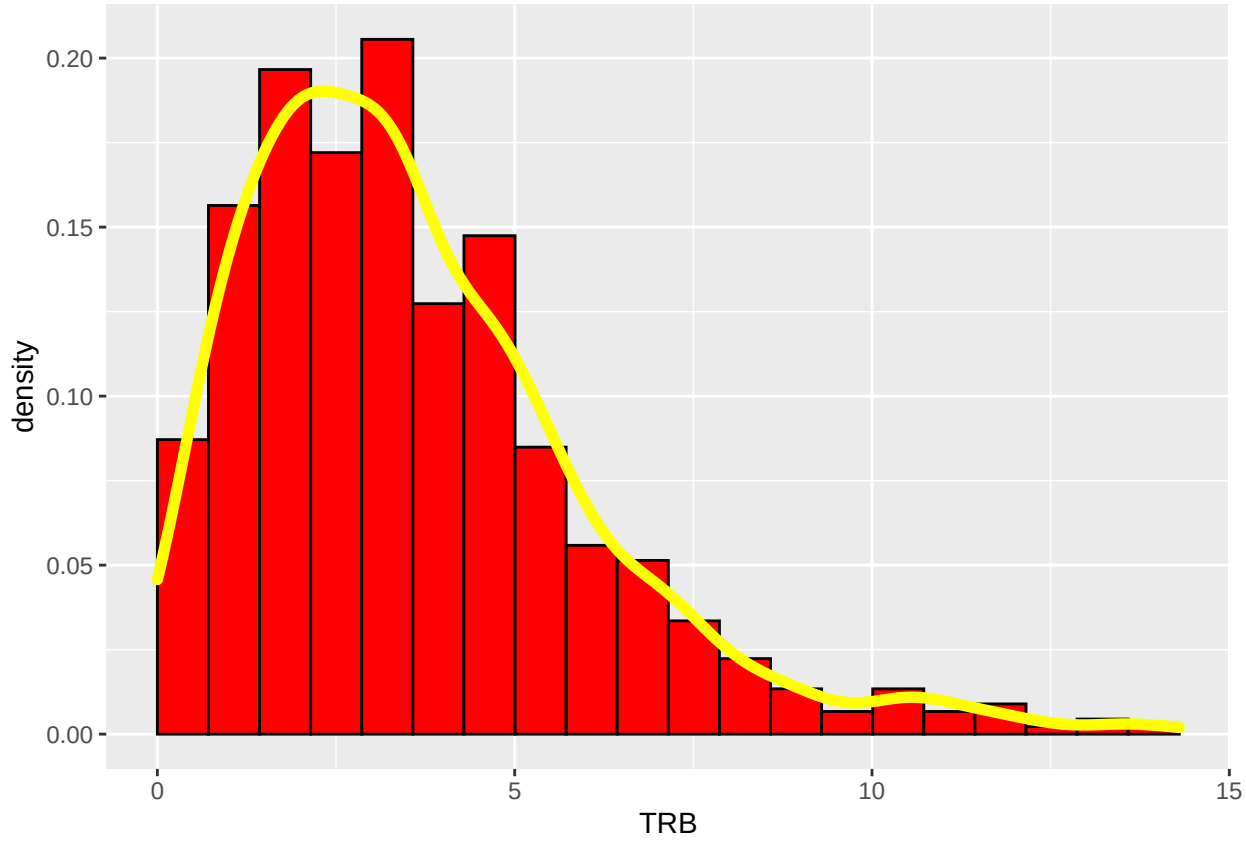
```
ptosPos = NBADef %>%
  group_by(Pos) %>%
  summarise(media = mean(PTS))
```

```
ggplot(data = ptosPos) +
geom_col(mapping = aes(x = Pos, y = media), colour = "white",
         fill = c("red", "yellow", "green", "blue", "black"),
         width = 0.65)
```

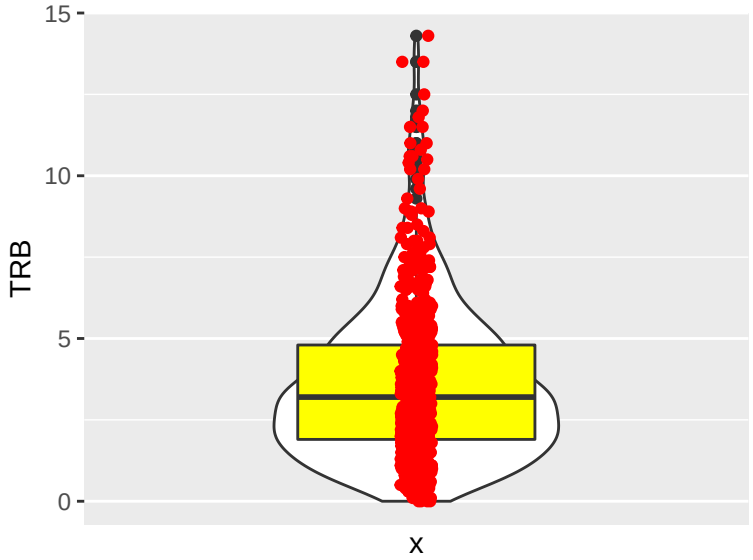


Aparte de diagramas de barras y columnas, *ggplot* nos permite la representación de otros tipos de gráficos, tales como histogramas y funciones de densidad o como en este caso, la combinación de ambas. Se ha realizado para la variable *TRB* (rebotes totales por partido). También se ha realizado un violinplot de la misma variable.

```
cortes = seq(min(NBADef$TRB), max(NBADef$TRB), length.out = 21)

ggplot(data = NBADef, mapping = aes(x = TRB)) +
  geom_histogram(aes(y = stat(density)), breaks = cortes, color = "black", fill = "red") +
  geom_density(color = "yellow", size = 2)
```

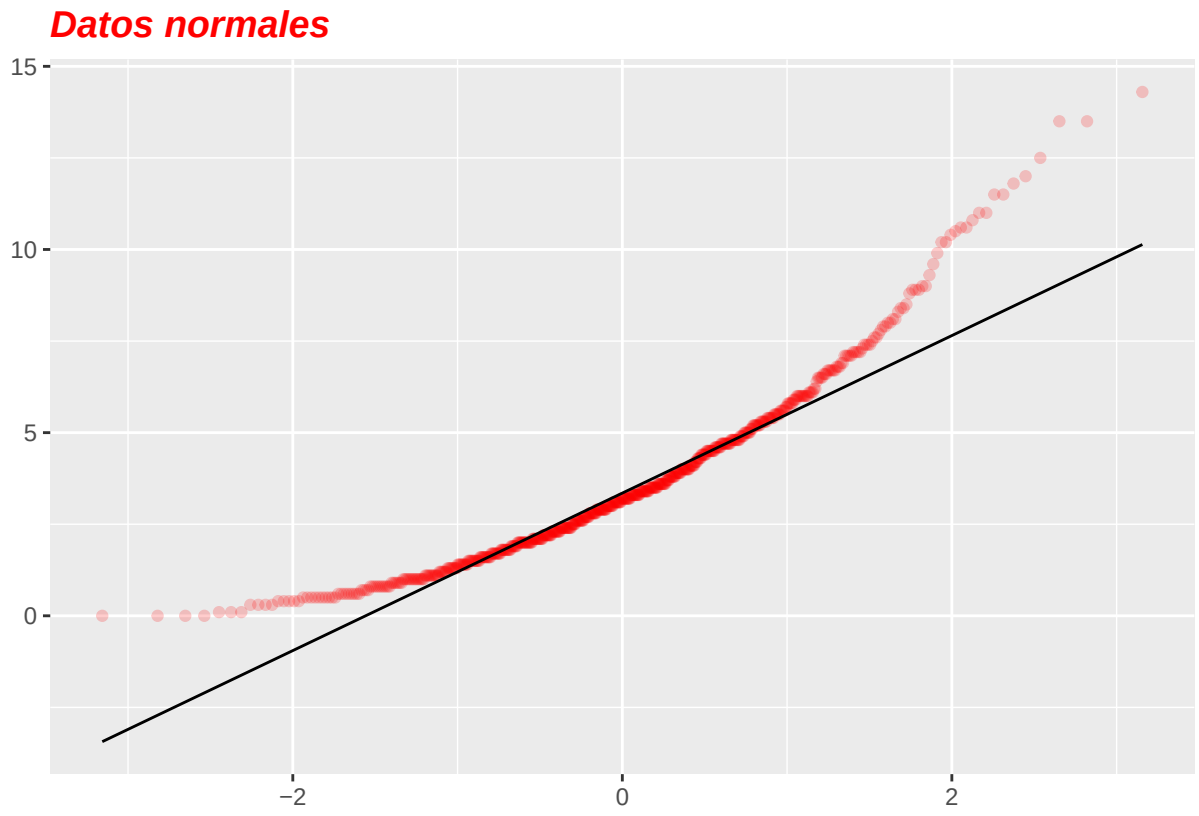



```
ggplot(data = NBADef) +
  geom_violin(mapping = aes(x = 0, y = TRB)) + scale_x_discrete(breaks = c()) +
  geom_boxplot(mapping = aes(y = TRB), fill = "yellow") +
  geom_jitter(aes(x = 0, y = TRB), position = position_jitter(w = 0.05, h = 0),
              col = "red")
```



También se puede estudiar la normalidad de la variable *TRB* mediante un *qqplot*. Tal y como era de esperar a simple vista con los gráficos obtenidos anteriormente, dicha variable no sigue una distribución normal. Esto, se puede comprobar de manera más precisa en el siguiente gráfico. De haber sido normal, la distribución de puntos rojos sería prácticamente coincidente con la línea negra. Puesto que ello no ocurre en los extremos, se puede concluir con una no-normalidad de la variable de estudio.

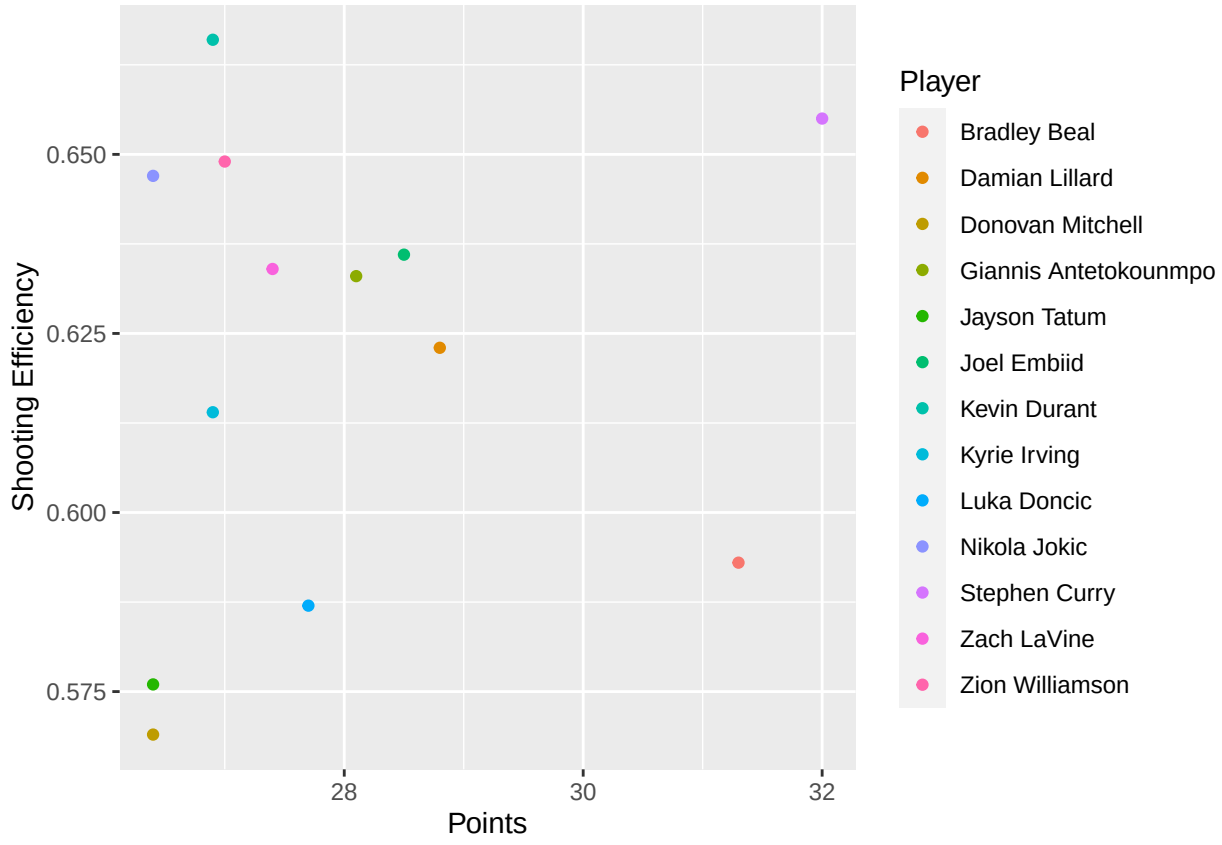
```
ggplot(tibble(x = NBADef$TRB), aes(sample = x))+
      geom_qq(alpha = 0.2, color = "red") +
      geom_qq_line() +
      ggtitle("Datos normales") +
      xlab("") +  ylab("") +
      theme(plot.title =
            element_text(color="red", size = 14, face = "bold.italic"))
```



Asimismo, también resulta interesante mostrar la relación entre los puntos y la eficiencia de tiro de los máximos anotadores promedio de la temporada.

```
PPPEff = NBADef %>%
  select("Tm","Player","Pos","PTS","TS%") %>%
  filter(PTS > 26)
```

```
ggplot(PPPEff) +
  geom_point(aes(x = PTS, y = PPPEff$`TS%`, color = Player)) +
  xlab("Points") +
  ylab("Shooting Efficiency")
```



2.2. Cálculo de probabilidades

- **Si elegimos al azar un jugador de los Boston Celtics (BOS), ¿cual es la probabilidad de que tenga más de 25 años?**

```
# Probabilidad de que sea de Boston
pBOS = as.numeric(NBADef %>%
  count(Tm) %>%
  mutate(frecRel = n/sum(n)) %>%
  .[2,3])

# Probabilidad de la interseccion (Boston y > 25 años)
P25yBOS = as.numeric(NBADef %>%
  filter(Tm == "BOS" & Age > 25) %>%
  count()/nrow(NBADef))

# Probabilidad condicionada
(p25siBOS = P25yBOS/pBOS)
```

```
## [1] 0.3809524
```

- **Si elegimos al azar 20 filas al azar y con reemplazamiento, ¿cual es la probablidad de que exactamente 7 de ellos hayan jugado 50 o más partidos (columna G)? ¿Y de que al menos 7 de ellos hayan jugado 50 o más?**

```
# Probabilidad de haber jugado 50 o más partidos
p50_mas = as.numeric((NBADef) %>%
filter(G >= 50) %>%
count()/nrow(NBADef))
```

```
#B(20, p50_mas) P[X = 7]
(dbinom(4, size = 20, prob = p50_mas))
```

```
## [1] 0.06643792
```

```
#B(20, p50_mas) P[X >= 7] = 1 - P[X < 7]
(1 - sum((dbinom(0:6, size = 20, prob = p50_mas))))
```

```
## [1] 0.6118991
```

- **Se supone que los triples de un equipo depende en su mayoría del escolta y el base. Se va a comprobar mediante una tabla de contingencia la relación entre escolta (SG) o base (PG) y algún triple anotado por partido.**

```
escoltaBase = ifelse(NBADef$Pos == "SG" | NBADef$Pos == "PG", "Escolta_Base", "Otro")
tripleAnotado = ifelse(NBADef$`3P` >= 1, "Triple", "NoTriple")
```

```
(tablaContingencia = table(escoltaBase, tripleAnotado))
```

```
##               tripleAnotado
## escoltaBase    NoTriple Triple
##   Escolta_Base      118    146
##   Otro              245    117
```

```
(tablaContingenciaAmpliada = addmargins(tablaContingencia))
```

```
##               tripleAnotado
## escoltaBase    NoTriple Triple Sum
##   Escolta_Base      118    146 264
##   Otro              245    117 362
##   Sum               363    263 626
```

```
(tablaContingenciaRel = tablaContingenciaAmpliada/sum(tablaContingencia))
```

```
##               tripleAnotado
## escoltaBase     NoTriple    Triple       Sum
##   Escolta_Base 0.1884984 0.2332268 0.4217252
##   Otro         0.3913738 0.1869010 0.5782748
##   Sum          0.5798722 0.4201278 1.0000000
```

```
# Probabilidad de que un escolta o base haya anotado un triple por partido
(pEscoltaBaseTriple = tablaContingenciaRel[1,2])
```

```
## [1] 0.2332268
```

```
# Probabilidad de que un escolta o base no haya anotado un triple por partido
tablaContingenciaRel[1,1]
```

```
## [1] 0.1884984
```

```
# Probabilidad de que algún no escolta o base no haya anotado algún triple por partido
tablaContingenciaRel[2,1]
```

```
## [1] 0.3913738
```

- **Si las variables posición Escolta_Base y condición de meter algún triple por partido fueran independientes, se cumpliría: P(Escolta_Base_Triple) = P(Escolta_Base) · P(Triple).**

```
pEscoltaBase = tablaContingenciaRel[1,3]
pTriple = tablaContingenciaRel[3,2]
(pEscoltaBaseTripleIndep = pEscoltaBase * pTriple)
```

```
## [1] 0.1771785
```

Comparamos con el valor obtenido anteriormente de la tabla de contingencia:

```
pEscoltaBaseTriple - pEscoltaBaseTripleIndep
```

```
## [1] 0.05604834
```

Puesto que la diferencia es de aproximadamente un 5,6%, podemos garantizar que ambas variables no son independientes.

- **Si elegimos de forma independiente (con remplazamiento) 15 jugadores, ¿cuál es la probabilidad de que 10 de ellos no sean ni escolta ni base?¿Cuántos jugadores habría que escoger entre esos 15 para garantizar un 50% de probabilidad de que hayan metido algún triple?**

```
pOtro = 1 - pEscoltaBase
(pOtro = dbinom(10, size = 15, prob = pOtro))
```

```
## [1] 0.1675119
```

```
(numJugadores = qbinom(p = 0.5, size = 15, prob = pTriple))
```

```
## [1] 6
```

2.3. Inferencia estadística y contraste de hipótesis

- **Asumiendo la normalidad de los datos, calcular un intervalo de confianza al 95% del total de rebotes de los pivots (Pos = C)**

```
NBAPivot <- NBADef %>%
  filter(Pos == "C")
```

```
n = length(NBAPivot$TRB)
barX = mean(NBAPivot$TRB)
s = sd(NBAPivot$TRB)
nc = 0.95
alfa = 1 - nc
zc = qnorm(alfa/2, lower.tail = FALSE)
(intervalo = signif((barX + c(-1,1) * zc * s /sqrt(n)), 4))
```

```
## [1] 4.995 6.111
```

```
# Otra manera
t.test(NBAPivot$TRB, conf.level = 0.95)
```

```
##
##  One Sample t-test
##
## data:  NBAPivot$TRB
## t = 19.502, df = 119, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  4.989495 6.117172
## sample estimates:
## mean of x
##  5.553333
```

- **Cogemos una muestra aleatoria de 10 jugadores de Houston (HOU) sin reemplazamiento. Contrastar la siguiente hipótesis nula: El número medio de minutos jugados es 30 (95% nivel de significancia)**

```
set.seed(2019)
nBAHouston <- NBADef %>%
  filter(Tm == "HOU")
muestra = nBAHouston[sample(1:nrow(nBAHouston), 10, replace = FALSE),]
```

```
# H0: Numero medio de mins jugados == 30
barX = mean(muestra$MP)
mu0 = 30
s = sd(muestra$MP)
n = nrow(muestra)
estadistico = abs((barX-mu0))/(s/sqrt(n))
signif(2*pt(estadistico, df = n - 1, lower.tail = FALSE), 4)
```

```
## [1] 0.0669
```

Puesto que el p-valor $>$ 1 - nc = 0.05, aceptamos la hipótesis nula.

- **Sospechamos que la cantidad media de puntos de los jugadores menores de 25 años es mayor que 7. ¿Avalan los datos esta sospecha, con un nivel de significación del 95% y asumiendo la normalidad de los datos?**

```
NBA25 <- NBADef %>%
  filter(Age < 25)
```

```
# Hipótesis alternativa: Puntos anotados mayor que 7
barX = mean(NBA25$PTS)
mu0 = 7
s = sd(NBA25$PTS)
n = length(NBA25$PTS)
estadistico = (barX-mu0)/(s/sqrt(n))
signif(pnorm(estadistico, lower.tail = FALSE))
```

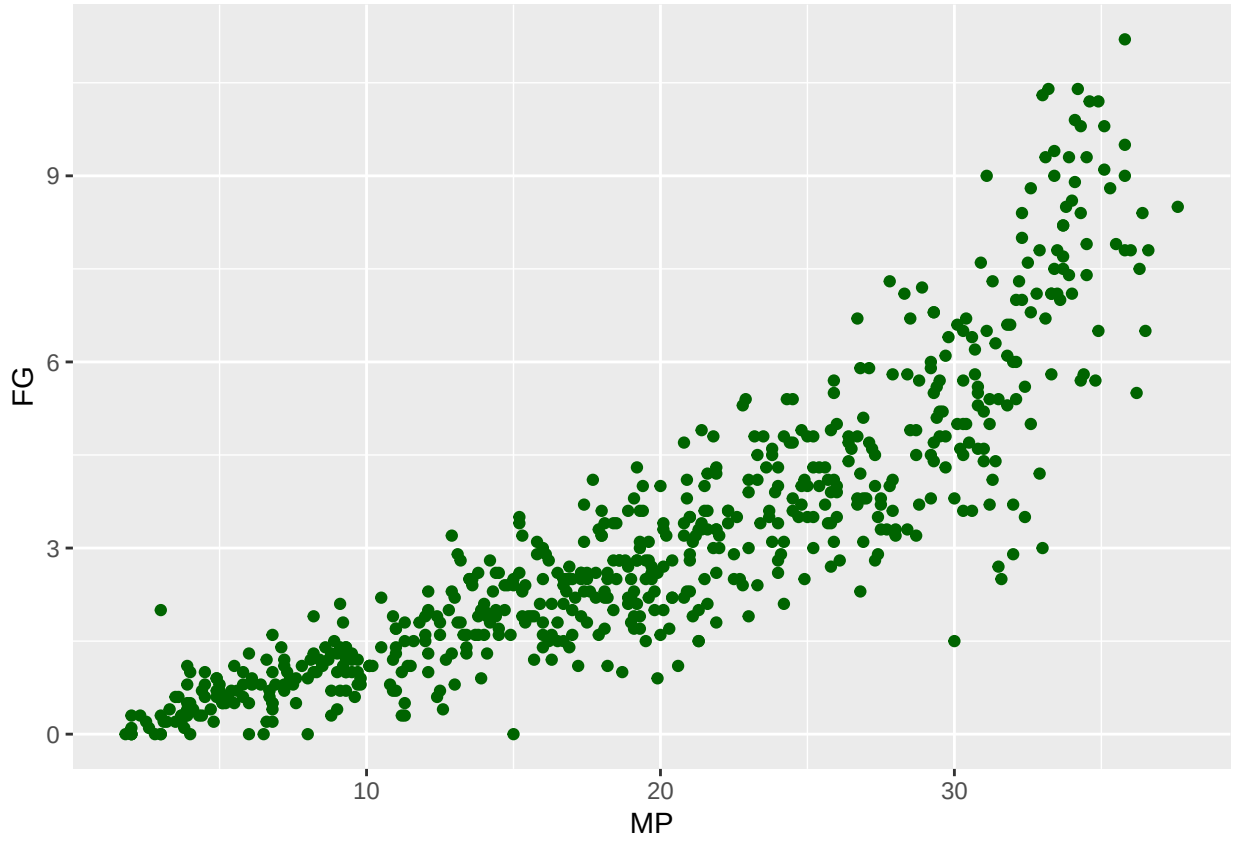
```
## [1] 0.039757
```

Como pvalor $<$ 1 - 0.95 rechazamos la hipótesis nula y aceptamos la alternativa.

2.4. Regresión lineal

- **Se va a estudiar la relación entre los minutos jugados y las canastas anotadas en campo (sin contar tiros libres). Esta variable se refleja en la columna *FG*.**

```
(plt = ggplot(NBADef) +
  geom_point(aes(MP, FG), col ="darkgreen"))
```



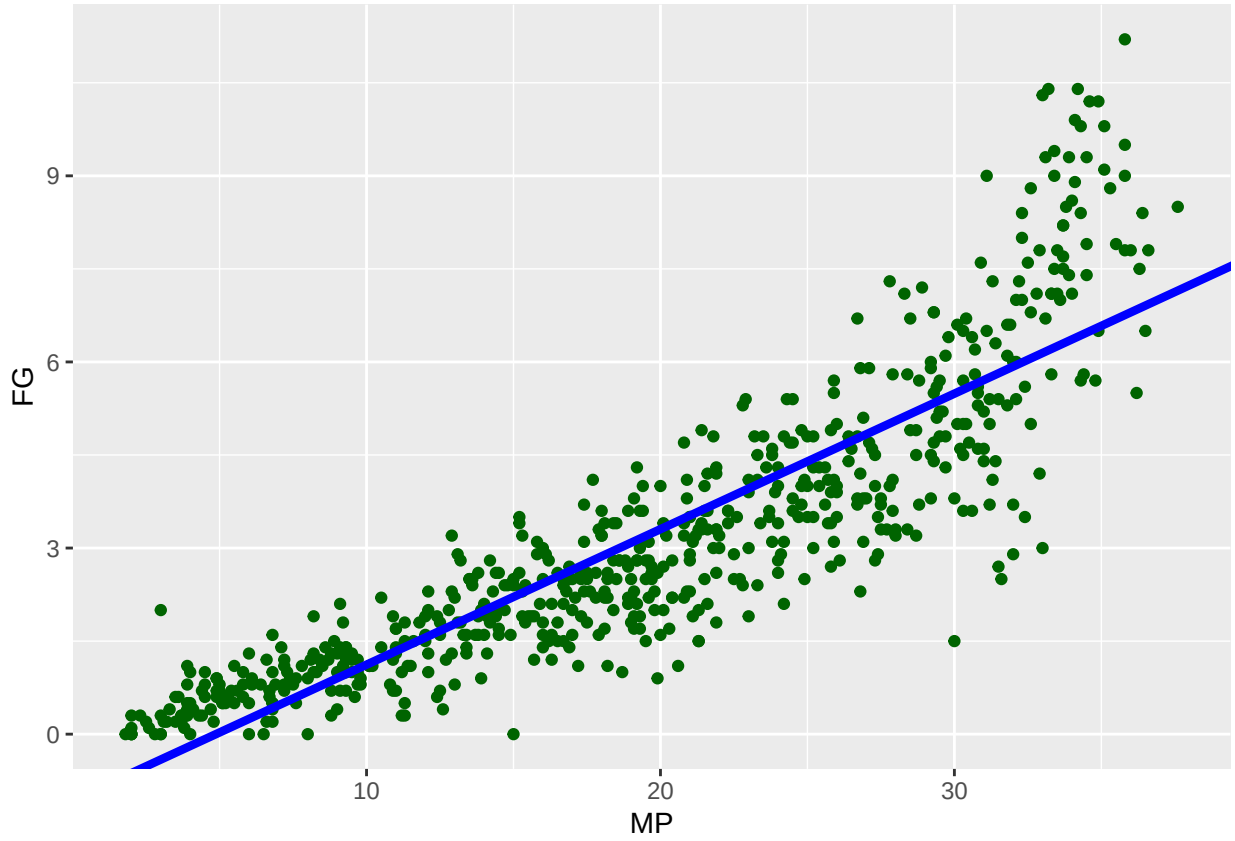

```
modelo = lm(FG ~ MP, data = NBADef)
```

```
modelo$coefficients
```

```
## (Intercept)          MP
##  -1.0646917   0.2184588
```

```
b0 = modelo$coefficients[1]
b1 = modelo$coefficients[2]
```

```
(plt <- plt + geom_abline(intercept = b0, slope = b1,
                          color="blue", size = 1.5))
```



- **¿Que porcentaje de la variabilidad de FG se explica con el modelo?**

```
# Con el modelo se explica un 77,2% de la variabilidad de FG
(R2 = cor(NBADef$FG, NBADef$MP)^2)
```

```
## [1] 0.7727709
```

- **¿Cúanto aumenta el FG por cada minuto jugado?**

```
# Aumenta el valor de la pendiente de la recta de regresión
unname(b1)
```

```
## [1] 0.2184588
```

- **¿Qué FG esperamos para alguien que juegue 20 minutos?**

```
unname(predict(modelo, newdata = data.frame(MP = 20)))
```

```
## [1] 3.304485
```

2.5. Regresión logística

Para la regresión logística se va a crear una nueva variable de tipo *dummy* que tome valor 1 cuando un jugador haya anotado 5 o más canastas promedio por partido y 0 cuando no.

```
NBADef <- NBADef %>%
  mutate(FG_5 = ifelse(FG >= 5, 1, 0))
```

Con esta nueva columna se modela una regresión logística respecto a la variable de minutos jugados.

```
glmPr = glm(FG_5 ~ MP, family = binomial, data = NBADef)
```

```
coefficients(glmPr)
```

```
## (Intercept)          MP
## -16.6937900   0.5697556
```

Con un modelo logístico y la función `predict` podemos hacer predicciones como hacíamos con un modelo lineal. Por ejemplo, ¿Cuál es la probabilidad de haber anotado más de cinco canastas (probabilidad de FG_5 = 1) que predice el modelo para un jugador con minutos jugados MP = 30 mins?

```
FGPredecir = data.frame(MP = 30)
(prob = unname(predict(glmPr, newdata = FGPredecir, type = 'response')))
```

```
## [1] 0.5984178
```

Es decir, un 60% de probabilidad de que un jugador que haya disputado 30 minutos anote más de 5 canastas promedio por partido.

3. MODELADO DE UNA PREDICCIÓN

En este apartado se ha estimado un modelo de regresión logística a partir de los datos históricos de *ALL-STARS* durante las tres temporadas previas a la actual. Con dicho modelo se podría predecir los *ALL-STARS* del año actual de estudio.

3.1. Preprocesado de los datasets iniciales

```
library(MLTools)
library(caret)
```

En primer lugar, importamos los datasets que se emplearán como entrenamiento del modelo (*training set*).

```
#2017-2018
NBAStats1_2017 <- read_csv("data/NBAStats1_2017.csv")
NBAStats2_2017 <- read_csv("data/NBAStats2_2017.csv")

#2018-2019
NBAStats1_2018 <- read_csv("data/NBAStats1_2018.csv")
NBAStats2_2018 <- read_csv("data/NBAStats2_2018.csv")

#2019-2020
NBAStats1_2019 <- read_csv("data/NBAStats1_2019.csv")
NBAStats2_2019 <- read_csv("data/NBAStats2_2019.csv")
```

De nuevo, eliminamos las columnas vacías de datos, unimos los datos en una única tabla, separamos el nombre del jugador, etc... (el preprocesado de estos nuevos conjuntos de datos coincide con el realizado al inicio del proyecto).

```
NBAStats2_2017  <- NBAStats2_2017  %>%
  select(-...20,-...25)
NBAStats2_2018  <- NBAStats2_2018  %>%
  select(-...20,-...25)
NBAStats2_2019  <- NBAStats2_2019  %>%
  select(-...20,-...25)
```

```
TotalNBA2017 <- merge(x = NBAStats1_2017, y = NBAStats2_2017, by = c("Rk","Tm"))

NBADef2017 <-  TotalNBA2017 %>%
  select(-GS, -PF, -'eFG%', -Player.y, -Pos.y, -Age.y, -G.y, -MP.y, -FTr, -'WS/48',
         -OBPM, -DBPM, -VORP)

NBADef2017 <- rename(NBADef2017, Player = Player.x, Pos = Pos.x,
                     Age = Age.x, G = G.x, MP = MP.x)
```

```
TotalNBA2018 <- merge(x = NBAStats1_2018, y = NBAStats2_2018, by = c("Rk","Tm"))

NBADef2018 <-  TotalNBA2018 %>%
  select(-GS, -PF, -'eFG%', -Player.y, -Pos.y, -Age.y, -G.y, -MP.y, -FTr, -'WS/48',
         -OBPM, -DBPM, -VORP)

NBADef2018 <- rename(NBADef2018, Player = Player.x, Pos = Pos.x,
                     Age = Age.x, G = G.x, MP = MP.x)
```

```
TotalNBA2019 <- merge(x = NBAStats1_2019, y = NBAStats2_2019, by = c("Rk","Tm"))

NBADef2019 <-  TotalNBA2019 %>%
  select(-GS, -PF, -'eFG%', -Player.y, -Pos.y, -Age.y, -G.y, -MP.y, -FTr, -'WS/48',
         -OBPM, -DBPM, -VORP)

NBADef2019 <- rename(NBADef2019, Player = Player.x, Pos = Pos.x,
                     Age = Age.x, G = G.x, MP = MP.x)
```

```
NBADef2017 <- NBADef2017 %>%
  separate(col = Pos, sep = "-", into = c("Pos", NULL))

NBADef2018 <- NBADef2018 %>%
  separate(col = Pos, sep = "-", into = c("Pos", NULL))

NBADef2019 <- NBADef2019 %>%
  separate(col = Pos, sep = "-", into = c("Pos", NULL))
```

```
NBADef2017 <- NBADef2017 %>%
  separate(col = Player, sep = '--', into = c("Player", NULL))

NBADef2018 <- NBADef2018 %>%
  separate(col = Player, sep = '--', into = c("Player", NULL))

NBADef2019 <- NBADef2019 %>%
  separate(col = Player, sep = '--', into = c("Player", NULL))
```

3.2. Filtrado de los datos de entrenamiento

Para cada temporada se realiza un filtrado con el objetivo de seleccionar las estadísticas de los jugadores de forma única.

```
Aux2017 = NBADef2017 %>%
  group_by(Player) %>%
  count()

Filtrado2017_Tot <- merge(x = NBADef2017, y = Aux2017, by = c("Player"))

Prev2017 = Filtrado2017_Tot %>%
  filter((n == 1) | (n > 1 & Tm=="TOT"))
```

```
Aux2018 = NBADef2018 %>%
  group_by(Player) %>%
  count()

Filtrado2018_Tot <- merge(x = NBADef2018, y = Aux2018, by = c("Player"))

Prev2018 = Filtrado2018_Tot %>%
  filter((n == 1) | (n > 1 & Tm == "TOT"))
```

```
Aux2019 = NBADef2019 %>%
  group_by(Player) %>%
  count()

Filtrado2019_Tot <- merge(x = NBADef2019, y = Aux2019, by = c("Player"))

Prev2019 = Filtrado2019_Tot %>%
  filter((n == 1) | (n > 1 & Tm == "TOT"))
```

A su vez, para cada uno de los tres años anteriores se elabora un listado de los jugadores que han sido *ALL-STARS* y se crea una variable output (tipo *dummy*) en la tabla de manera que si el jugador está en la lista toma el valor de 1 y si no 0.

```
AllStar2017 = c("Kyrie Irving","DeMar Derozan","Lebron James","Joel Embiid",
                "Giannis Antetokounmpo","Bradley Beal","Goran Dragi?",
                "Al Horford","Kevin Love","Kyle Lowry","Victor Oladipo",
                "Kristaps Porzi??is", "John Wall","Andre Drummond","Stephen Curry",
                "James Harden","Kevin Durant","DeMarcus Cousins",
                "Anthony Davis","Russell Westbrook","Damian Lillard",
                "Draymond Green","Karl-Anthony Towns","LaMarcus Aldridge",
                "Klay Thompson","Jimmy Butler")

Prev2017$Allstar = ifelse((Prev2017$Player %in% AllStar2017), 1, 0)
```

```
AllStar2018 = c("Kyrie Irving","Kemba Walker","Kawhi Leonard","Joel Embiid",
                "Giannis Antetokounmpo","Kyle Lowry","Victor Oladipo",
                "Khris Middleton","Bradley Beal","Ben Simmons",
                "Blake Griffin","Nikola Vu?evi?","D'Angelo Russell",
                "Stephen Curry","James Harden","Kevin Durant",
                "Paul George","LeBron James","Anthony Davis",
                "Russell Westbrook","Damian Lillard","Klay Thompson",
                "Karl-Anthony Towns","LaMarcus Aldridge","Nikola Joki?")

Prev2018$Allstar = ifelse((Prev2018$Player %in% AllStar2018), 1, 0)
```

```
AllStar2019 = c("Trae Young","Kemba Walker","Pascal Siakam","Joel Embiid",
                "Giannis Antetokounmpo","Jimmy Butler","Bam Adebayo",
                "Ben Simmons","Khris Middleton","Kyle Lowry","Domantas Sabonis",
                "Jayson Tatum","Luka Don?i?","James Harden","LeBron James",
                "Kawhi Leonard","Anthony Davis","Nikola Joki?","Damian Lillard",
                "Rudy Gobert","Brandon Ingram","Chris Paul","Donovan Mitchell",
                "Russell Westbrook","Devin Booker")

Prev2019$Allstar = ifelse((Prev2019$Player %in% AllStar2019), 1, 0)
```

Por último, hacemos un join para definir el set de entrenamiento:

```
df_Training = union_all(union_all(Prev2017,Prev2018),Prev2019)
df_Training <-  mutate_all(df_Training, ~ replace(., is.na(.), 0))
```

3.3. Filtrado de los datos de testeo

Como set de testeo, utilizamos los datos de la temporada 2020. Por tanto, volvemos a repetir el proceso de preprocesado para este dataset.

```
NBAStats1_2020 <- read_csv("data/NBAStats1_2020.csv")
NBAStats2_2020 <- read_csv("data/NBAStats2_2020.csv")
```

```
NBAStats2_2020  <- NBAStats2_2020  %>%
  select(-...20,-...25)
```

```
TotalNBA2020 <- merge(x = NBAStats1_2020, y = NBAStats2_2020, by = c("Rk","Tm"))

NBADef2020 <-  TotalNBA2020 %>%
  select(-GS, -PF, -'eFG%', -Player.y, -Pos.y, -Age.y, -G.y, -MP.y, -FTr, -'WS/48',
         -OBPM, -DBPM, -VORP)

NBADef2020 <- rename(NBADef2020, Player = Player.x, Pos = Pos.x,
                     Age = Age.x, G = G.x, MP = MP.x)
```

```
NBADef2020 <- NBADef2020 %>%
  separate(col = Pos, sep = "-", into = c("Pos", NULL))
```

```
NBADef2020 <- NBADef2020 %>%
  separate(col = Player, sep = '--', into = c("Player", NULL))
```

```
Aux2020 = NBADef2020 %>%
  group_by(Player) %>%
  count()

Filtrado2020_Tot <- merge(x = NBADef2020, y = Aux2020, by = c("Player"))

Prev2020 = Filtrado2020_Tot %>%
  filter((n == 1) | (n > 1 & Tm == "TOT"))
```

```
AllStar2020 = c("Bradley Beal","Kyrie Irving","Kevin Durant","Joel Embiid",
              "Giannis Antetokounmpo","Jaylen Brown","James Harden",
              "Zach LaVine","Ben Simmons","Julius Randle","Jayson Tatum",
              "Nikola Vu?evi?","Domantas Sabonis","Stephen Curry","LeBron James",
              "Nikola Joki?","Kawhi Leonard","Luka Don?i?","Damian Lillard",
              "Rudy Gobert","Donovan Mitchell","Chris Paul","Anthony Davis",
              "Paul George","Devin Booker","Zion Williamson","Mike Conley")

Prev2020$Allstar = ifelse((Prev2020$Player %in% AllStar2020),1,0)

df_Test = Prev2020
df_Test <-  mutate_all(df_Test, ~ replace(., is.na(.), 0))
```

3.4. Selección de variables más interesantes para el estudio

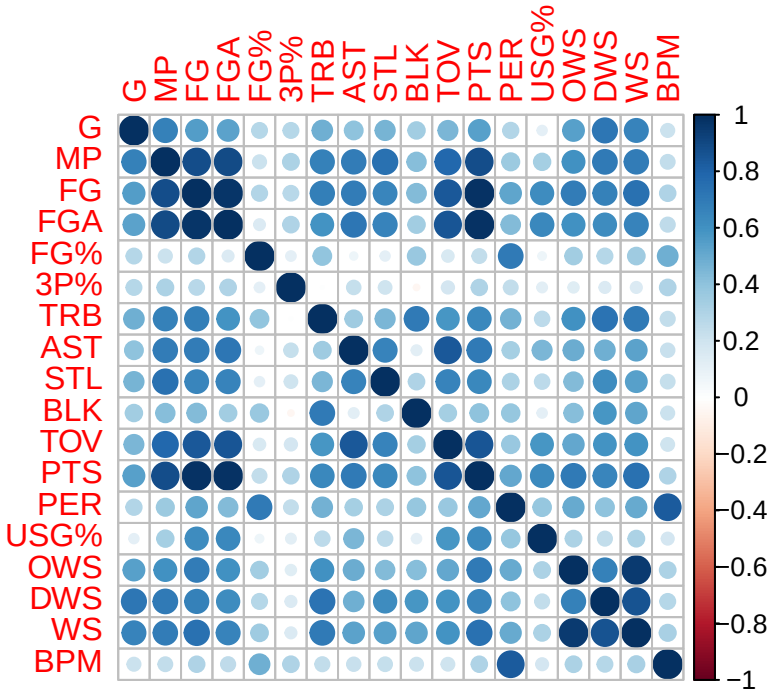
Eliminamos aquellas columnas innecesarias para el estudio y cambiamos las variables de tipo caracter a factor.

```
df_Training <- df_Training %>%
  select(-"Player",-"Rk",-"Age",-"3P",-"3PA",-"2P",-"2PA",-"2P%",-"FT",-"FTA",-"FT%",
         -"ORB",-"DRB",-"TS%",-"3PAr",-"ORB%",-"DRB%",-"TRB%",-"AST%",-"STL%",-"BLK%",
         -"TOV%",-"n")

df_Training = df_Training %>%
  mutate(Y = Allstar) %>%
  mutate(Allstar = NULL)

# Cambio de las variables de tipo caracter a factor
df_Training$Y = as.factor(df_Training$Y)
levels(df_Training$Y) = c("NO","YES")
df_Training$Tm = as.factor(df_Training$Tm)
df_Training$Pos = as.factor(df_Training$Pos)
# summary(df_Training)
```

Comprobamos la existencia de variables correladas con el siguiente gráfico.

```
library(GGally)
```

```
#Se dejan aunque se aprecia cierta correlacion lineal entre variables
numvars <- sapply(df_Training, class) %in% c("integer","numeric")
C <- cor(df_Training[,numvars])
corrplot::corrplot(C)
```



3.5. Ajuste del modelo

Ante la situación en la que nos encontramos, donde disponemos de muchas más muestras de una clase (No AllStar) que de otra, se ha optado por recurrir a una técnica conocida como upSampling, para que el resultado de las métricas obtenidas con nuestro modelo no esté sesgado.

```
df_Training = upSample(df_Training[,-ncol(df_Training)],
                       df_Training$Y)
df_Training = df_Training %>%
  mutate(Y = Class) %>%
  mutate(Class = NULL)
```

Se incluye en el modelo el método de validación cruzada.

```
ctrl <- trainControl(method = "cv",                          #k-fold cross-validation
                     number = 10,                            #Number of folds
                     summaryFunction = defaultSummary,       #Performance summary
                     classProbs = TRUE)
```

Se realiza el entrenamiento del modelo con el dataset unido de la información de los tres años anteriores. Se ha escogido el método de regresión logística para dicho entrenamiento.

```
set.seed(150)
LogReg.fit <- train(form = Y ~ ., #formula for specifying inputs and outputs.
                    data = df_Training,                #Training dataset
                    method = "glm",                    #Train logistic regression
                    preProcess = c("center","scale"), #Center an scale inputs
                    trControl = ctrl,                  #trainControl Object
                    metric = "Accuracy")               #metric used for hyperparameters
LogReg.fit     #información sobre el remuestreo empleado en el cross-validation
```

```
## Generalized Linear Model
##
## 3050 samples
##   20 predictor
##    2 classes: 'NO', 'YES'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2746, 2745, 2745, 2745, 2745, 2746, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9868959  0.9737923
```

```
# summary(LogReg.fit)
```

Obtenemos el accuracy y el kappa obtenido a partir del remuestreo, los p-valores y la significancia de las variables.

Quitamos la variable Tm, ya que se ve que no tiene demasiada significancia en la explicación de la variable respuesta (en el summary del modelo).

```
df_Training=df_Training %>%
  select(-Tm)
```

Se vuelve a entrenar el modelo con la modificación realizada.

```
set.seed(150)
LogReg.fit <- train(form = Y ~ ., #formula for specifying inputs and outputs.
                    data = df_Training,               #Training dataset
                    method = "glm",                   #Train logistic regression
                    preProcess = c("center","scale"), #Center an scale inputs
                    trControl = ctrl,                 #trainControl Object
                    metric = "Accuracy")              #metric used for hyperparameters


LogReg.fit     #información sobre el remuestreo empleado en el cross-validation
```

```
## Generalized Linear Model
##
## 3050 samples
##   19 predictor
##    2 classes: 'NO', 'YES'
##
## Pre-processing: centered (22), scaled (22)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2746, 2745, 2745, 2745, 2745, 2746, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9747668  0.9495349
```

```
#summary(LogReg.fit) Para volver a ver la información
```

Realizamos las mismas modificaciones en el dataset de testeo (2020) para incluir predicciones y poder realizar la evaluación

```
fTR_eval <- df_Training
df_Test <- df_Test %>%
  select(-"Player",-"Rk",-"Age",-"3P",-"3PA",-"2P",-"2PA",-"2P%",-"FT",-"FTA",-"FT%",
         -"ORB",-"DRB",-"TS%",-"3PAr",-"ORB%",-"DRB%",-"TRB%",-"AST%",-"STL%",-"BLK%",
         -"TOV%",-"n",-"Tm")

df_Test = df_Test %>%
  mutate(Y = Allstar) %>%
  mutate(Allstar = NULL)
```

```
# Cambio de las variables de tipo caracter a factor
df_Test$Y = as.factor(df_Test$Y)
levels(df_Test$Y) = c("NO","YES")
df_Test$Pos = as.factor(df_Test$Pos)
fTS_eval <- df_Test
```

3.6. Evaluación del modelo

```
### Evaluate model----Rellenamos dataset con predicciones
### TEST
fTS_eval$LRprob <- predict(LogReg.fit, type = "prob",
                           newdata = df_Test) # predict probabilities
fTS_eval$LRpred <- predict(LogReg.fit, type = "raw",
                           newdata = df_Test) # predict classes
```

Se comprueban las métricas obtenidas para el set de testeo.

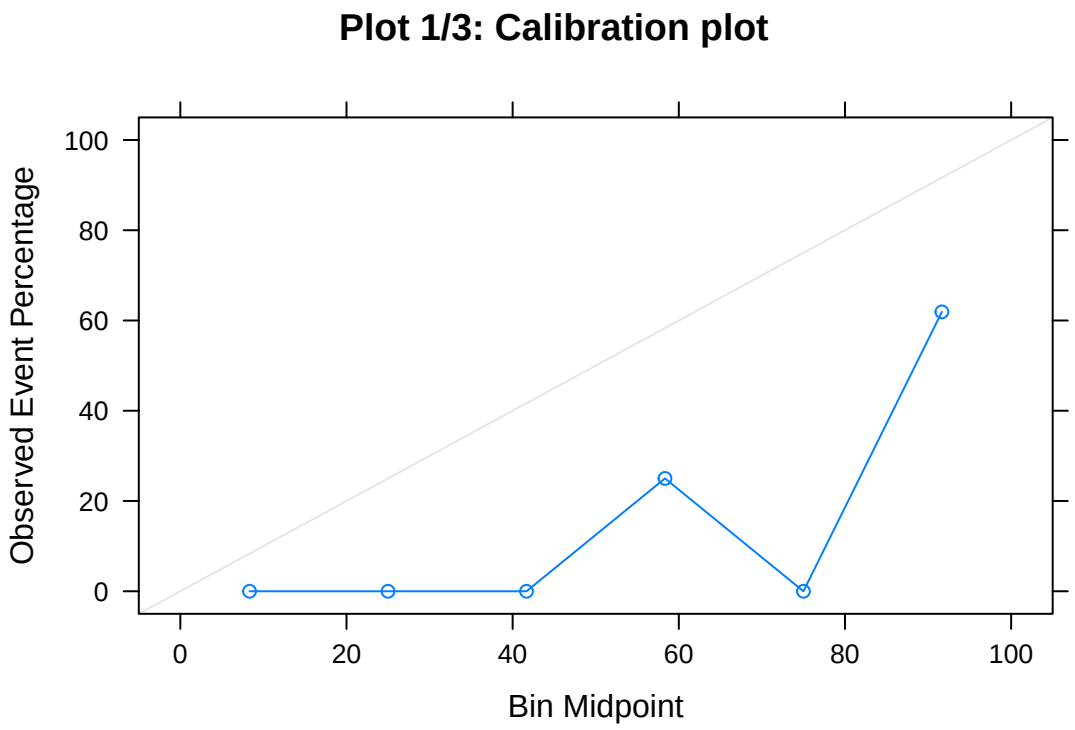
```
### Evaluación del modelo -------------
### Confusion matrices
### TEST
confusionMatrix(fTS_eval$LRpred,
                fTS_eval$Y,
                positive = "YES")
```

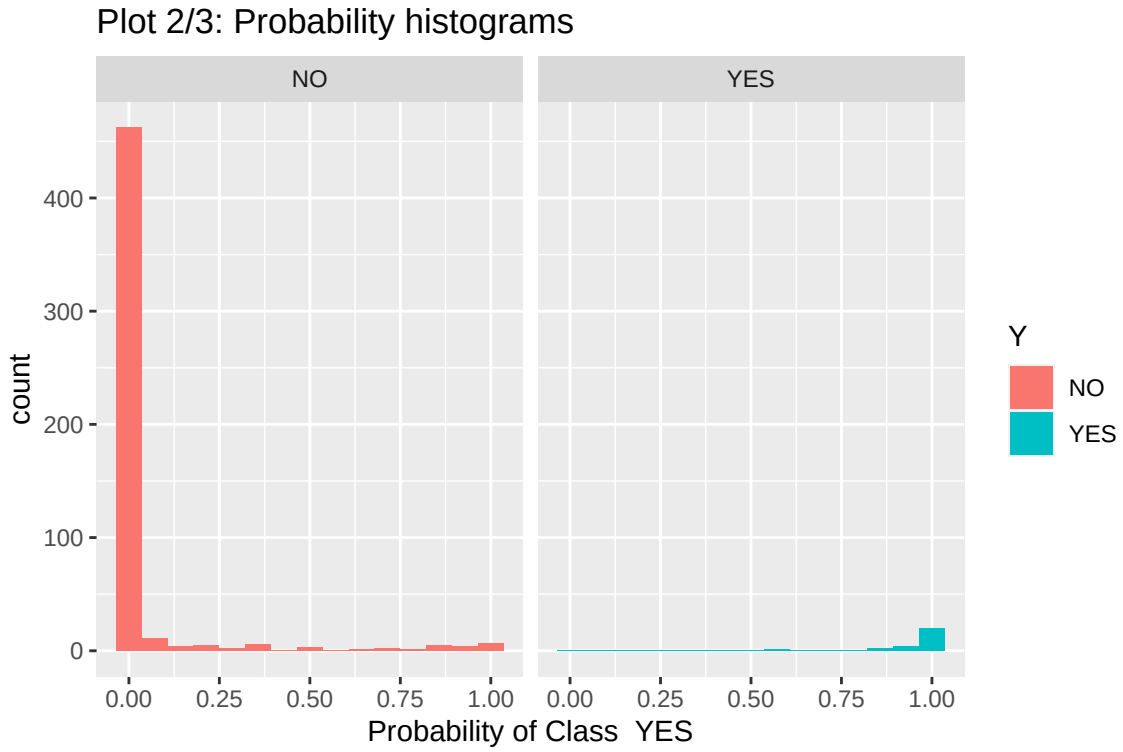
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  491   0
##        YES  22  27
##
##                Accuracy : 0.9593
##                  95% CI : (0.939, 0.9743)
##     No Information Rate : 0.95
##     P-Value [Acc > NIR] : 0.1886
##
##                   Kappa : 0.6906
##
##  Mcnemar's Test P-Value : 7.562e-06
##
##             Sensitivity : 1.00000
##             Specificity : 0.95712
##          Pos Pred Value : 0.55102
##          Neg Pred Value : 1.00000
##              Prevalence : 0.05000
##          Detection Rate : 0.05000
##    Detection Prevalence : 0.09074
```

```
##        Balanced Accuracy : 0.97856
##
##         'Positive' Class : YES
##
```

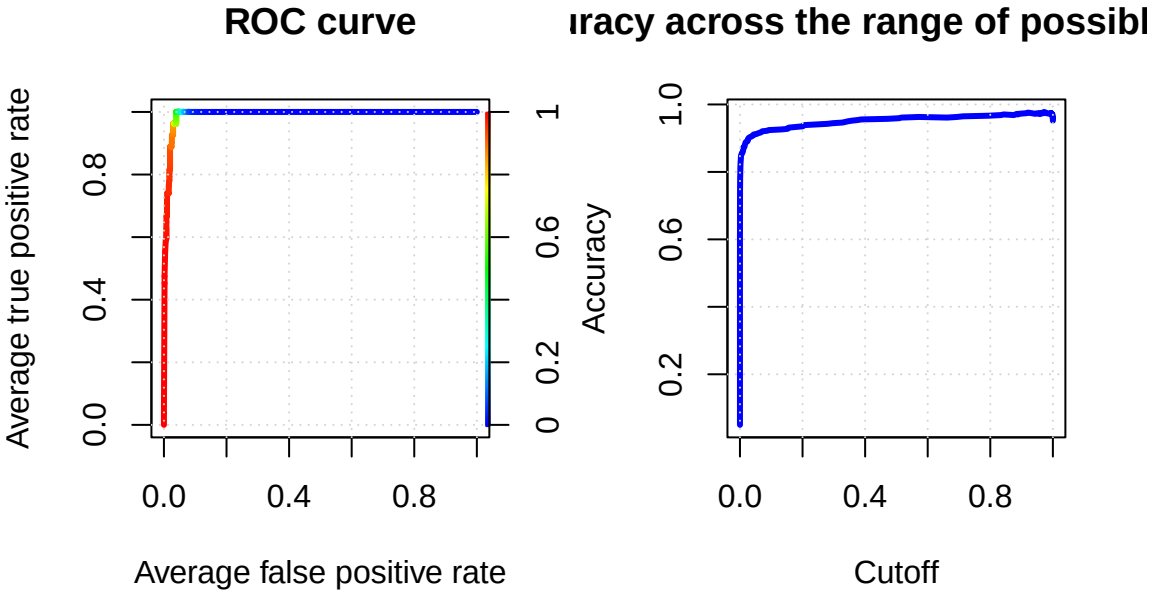
De igual manera, se muestra la curva ROC obtenida para el set de validación y la curva ROC bajo la misma (0.991119)

```
### Classification performance plots
### TEST
PlotClassPerformance(fTS_eval$Y,
                     fTS_eval$LRprob,
                     selClass = "YES")
```





Plot 3/3: ROC curves




```
## [1] "Area under the ROC curve (auc):  0.991625153418526"
```