Práctica 2: Limpieza y análisis de datos

UOC - Tipología y ciclo de vida de los datos

Miguel Santos Pérez y Alejandro Arzola García

17 de junio de 2020

Índice

```
 # Cargamos las librerías necesarias para el proyecto
suppressMessages(library(dplyr))
suppressMessages(library(plyr))
```

```
## Warning: package 'plyr' was built under R version 3.6.3
```

```
suppressMessages(library(data.table))
```

```
## Warning: package 'data.table' was built under R version 3.6.3
```

```
suppressMessages(library(ggcorrplot))
```

```
## Warning: package 'ggcorrplot' was built under R version 3.6.3
```

```
suppressMessages(library(pROC))
```

```
## Warning: package 'pROC' was built under R version 3.6.3
```

```
suppressMessages(library(factoextra))
```

```
## Warning: package 'factoextra' was built under R version 3.6.3
```

```
suppressMessages(library(reshape2))
```

```
## Warning: package 'reshape2' was built under R version 3.6.3
```

```
suppressMessages(library(nortest))
suppressMessages(library(ggplot2))
suppressMessages(library(grid))
suppressMessages(library(caret))
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

1 Introducción y descripción de los datasets

En la Práctica 1 de la asginatura de *Tipología y ciclo de vida de los datos* se creó un proceso *web scraping* en el que se obtuvieron los datos relacionados a todos los partidos y las estadísticas individuales de todos los jugadores de la actual temporada (2019-2020) de la Euroliga, la máxima competición de clubes de baloncesto de Europa. El objetivo final era contribuir al aumento de explotación de los datos para los equipos europeos y tratar de igualar lo que se realiza en América con las poderosas franquicias de NBA.

Mediante el proceso de obtención de datos realizado en *Python* que comentamos anteriormente, conseguimos crear dos datasets: el primero contiene los datos generales de un partido (nombre de los equipos, puntuación de cada equipo, fecha y hora del partido) y el segundo contiene más de quince estadísticas individuales para cada jugador por cada partido (puntos, minutos, rebotes, asistencias, ...). Este trabajo está disponible en un repositorio de *GitHub* al que se puede acceder haciendo click aquí.

El primer dataset (`euroleague_scoreboards.csv`) contiene las siguientes variables:

- **Id:** identificador único de partido.
- **Date:** fecha del partido.
- **HomeTeam:** nombre del equipo local.
- **HomeScore:** puntos anotados por el equipo local.
- **VisitingTeam:** nombre del equipo visitante.
- **VisitingScore:** puntos anotados por el equipo visitante.
- **Link:** enlace a la página web con las estadísticas individuales del partido.

El segundo dataset (`euroleague_stats_per_game.csv`) contiene las siguientes variables:

- **MatchId:** identificador único de partido.
- **Team:** equipo al que pertenece el jugador.
- **PlayerNumber:** número del jugador.
- **PlayerName:** nombre del jugador.
- **Min:** minutos jugados.
- **Pts:** puntos anotados.
- **2FG:** tiros de dos (metidos-anotados).
- **3FG:** tiros de tres (metidos-anotados).
- **FT:** tiros libres (metidos-anotados).
- **O:** rebotes ofensivos.
- **D:** rebotes defensivos.
- **T:** rebotes totales.
- **As:** asistencias.
- **St:** recuperaciones.
- **To:** pérdidas.
- **Fv:** tapones a favor.
- **Ag:** tapones en contra.
- **Cm:** faltas cometidas.
- **Rv:** faltas recibidas.
- **PIR:** valoración del jugador.

En esta segunda práctica queremos seguir progresando para lograr el objetivo de igualarnos con la NBA y por ello vamos a utilizar los datasets que hemos mencionado para realizar un análisis estadístico. Este análisis se va a enfocar en analizar las estadísticas globales por partido de cada equipo para tratar de estimar y predecir las victorias de un equipo o definir qué equipos defienden mejor, cuales tienen un mejor ataque y aquellos que tienen una forma de jugar más equilibrada.

Los datasets y el código utilizado para generar el análisis estadístico que se desarrolla a continuación está disponible en el siguiente repositorio de *GitHub*:

- https://github.com/aarzola-uoc/practica2-tycvd

2 Integración y selección de los datos de interés a analizar

Los datos relevantes del primer dataset son exclusivamente los nombres de los equipos y el marcador. Sin embargo, estos datos son posibles calcularlos a partir del segundo dataset, incluso aumentar los datos de equipo ya que son la suma de las estadísticas de todos los jugadores para cada partido.

En base a esto, hemos decidido que se creará un nuevo dataset utilizando código R, a partir del dataset de estadísticas de los jugadores, que tendrá las estadísticas de cada equipo por partido (suma de cada jugador es el total del equipo). El primer dataset se utilizará únicamente como referencia para obtener el equipo oponente de cada partido.

Además vamos a calcular una serie de estadísticos muy interesantes y que permiten un análisis mucho más técnico y útli para los equipos. Estos estadísticos son el ritmo de juego, eficiencia ofensiva/defensiva, porcentaje efectivo de tiros de campo, tasa de rebote y algunos más. La explicación de cada uno de estos estadísticos y su correspondiente fórmula se explicará más adelante.

---

3 Limpieza de los datos

El primer paso que tenemos que realizar es cargar los ficheros csv `euroleague_stats_per_game.csv` y `euroleague_scoreboards.csv`. Para ello vamos a utilizar la función `read.csv2` ya que el fichero a cargar está en formato csv español (los separadores son el caracter `;`). Como resultado obtendremos dos objetos `data.frame` que contendrán todos los datos de nuestros dos datasets:

```
data_scoreboards <- read.csv2('../csv/euroleague_scoreboards.csv', header = TRUE)
data_euroleague <- read.csv2('../csv/euroleague_stats_per_game.csv', header = TRUE)
```

Mostramos los primeros registros de ambos datasets para verificar que los datos se han cargado correctamente:

```
# Primer dataset (euroleague_scoreboards.csv)
head(data_scoreboards)
```

```
##   Id                Date                  HomeTeam HomeScore
## 1  1 October 3 19:00 CET      Khimki Moscow Region        89
## 2  2 October 3 20:00 CET Panathinaikos OPAP Athens        87
## 3  3 October 3 20:30 CET          FC Bayern Munich        78
## 4  4 October 3 21:00 CET               Real Madrid        81
## 5  5 October 4 19:00 CET           Zalgiris Kaunas        58
## 6  6 October 4 19:30 CET     Anadolu Efes Istanbul        64
##                        VisitingTeam VisitingScore
## 1              Maccabi FOX Tel Aviv            83
## 2        Crvena Zvezda mts Belgrade            82
## 3          AX Armani Exchange Milan            64
## 4          Fenerbahce Beko Istanbul            77
## 5 KIROLBET Baskonia Vitoria-Gasteiz            70
## 6                      FC Barcelona            74
##                                                                                 Link
## 1 https://www.euroleague.net/main/results/showgame?gamecode=1&seasoncode=E2019
## 2 https://www.euroleague.net/main/results/showgame?gamecode=8&seasoncode=E2019
## 3 https://www.euroleague.net/main/results/showgame?gamecode=2&seasoncode=E2019
## 4 https://www.euroleague.net/main/results/showgame?gamecode=4&seasoncode=E2019
## 5 https://www.euroleague.net/main/results/showgame?gamecode=5&seasoncode=E2019
## 6 https://www.euroleague.net/main/results/showgame?gamecode=6&seasoncode=E2019
```

```r
# Segundo dataset (euroleague_stats_per_game.csv)
head(data_euroleague)
```

```
##   MatchId                 Team PlayerNumber          PlayerName   Min Pts X2FG
## 1       1 Khimki Moscow Region            1       SHVED, ALEXEY 30:19  22  4/7
## 2       1 Khimki Moscow Region            5       BOOKER, DEVIN 19:21   4  1/4
## 3       1 Khimki Moscow Region            6        TIMMA, JANIS 33:53  17  1/3
## 4       1 Khimki Moscow Region            8 ZAYTSEV, VYACHESLAV 10:23   0    0
## 5       1 Khimki Moscow Region            9      VIALTSEV, EGOR  4:58   0    0
## 6       1 Khimki Moscow Region           10 DESIATNIKOV, ANDREI   DNP   -    -
##   X3FG   FT O D T As St To Fv Ag Cm Rv PIR
## 1 2/10 8/10 0 2 2  6  0  3  1  0  1  5  19
## 2    0  2/2 3 3 6  0  2  1  0  0  2  1   7
## 3 5/12    0 1 3 4  1  4  1  0  0  5  2  13
## 4    0    0 1 1 2  2  1  1  0  0  2  1   3
## 5    0    0 0 0 0  1  0  0  0  0  2  0  -1
## 6    -    - - - -  -  -  -  -  -  -  -   -
```

La variable MatchId es una variable que necesitamos para cruzar ambos datasets. Sin embargo, en el primer dataset, el nombre de esta variable es Id por lo que vamos a homogenizar su nombre:

```r
names(data_scoreboards)[names(data_scoreboards) == "Id"] <- "MatchId"
```

En total disponemos de estadísticas de 252 partidos y de 306 jugadores diferentes.

3.1 Elementos vacíos

Los datos vacíos o no definidos pueden presentarse en distintos formatos, normalmente en forma de cadena de caracteres vacía (`""`) o `NA` (*Not Available en inglés*), pero en algunos contextos pueden incluso tomar valores numéricos como 0 o 999. Es fundamental identificar para cada variable los valores que indiquen que existe una pérdida de datos para poder aplicar una solución y que no afecte en la calidad de los análisis estadísticos que se realicen posteriormente.

Procedemos a buscar valores vacíos en nuestro primer dataset:

```r
colSums(is.na(data_scoreboards))
```

```
##       MatchId          Date      HomeTeam     HomeScore  VisitingTeam
##             0             0             0             0             0
## VisitingScore          Link
##             0             0
```

```r
colSums(data_scoreboards=="")
```

```
##       MatchId          Date      HomeTeam     HomeScore  VisitingTeam
##             0             0             0             0             0
## VisitingScore          Link
##             0             0
```

Como se puede observar no se ha detectado ningún elemento vacío del tipo `NA` o cadena de caracteres vacía. En el caso de que se haya utilizado valores numéricos fuera del dominio del atributo lo podremos detectar en el análisis de valores extremos.

Realizamos el mismo análisis para el segundo dataset. En este caso, como hemos creado nosotros el dataset, sabemos que se ha utilizado el caracter `"-"` para indicar las estadísticas de los jugadores que no han jugado el partido y `"DNP"` (*Did Not Play*) en el atributo de tiempo:

```
colSums(is.na(data_euroleague))
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0          504            0            0            0
##         X2FG         X3FG           FT            0            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0           NA            0            0            0
##         X2FG         X3FG           FT            0            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="-")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0           NA            0            0          492
##         X2FG         X3FG           FT            0            D            T
##          492          492          492          492          492          492
##           As           St           To           Fv           Ag           Cm
##          492          492          492          492          492          492
##           Rv          PIR
##          492          492
```

```
colSums(data_euroleague=="DNP")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0           NA            0          492            0
##         X2FG         X3FG           FT            0            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

Para solucionar este problema y poder realizar un análisis estadístico en el que se pueda asegurar un nivel alto de calidad de datos, hemos optado por informar todos estos campos con el valor 0:

```
data_euroleague[is.na(data_euroleague)] <- 0
data_euroleague[data_euroleague=="-"] <- 0
data_euroleague[data_euroleague=="DNP"] <- 0

colSums(is.na(data_euroleague))
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0            0            0            0            0
##         X2FG         X3FG           FT            O            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0            0            0            0            0
##         X2FG         X3FG           FT            O            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="-")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0            0            0            0            0
##         X2FG         X3FG           FT            O            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="DNP")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0            0            0            0            0
##         X2FG         X3FG           FT            O            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

Hemos vuelto a buscar elementos vacíos para comprobar que se han solucionado todos los problemas que habíamos detectado y como se puede ver en los fragmentos anteriores se han resuelto todos correctamente.

3.2 Tratamiento de variables

El siguiente paso consisten en analizar el tipo de dato que ha asignado R para cada una de nuestras variables al cargar los datasets:

```
str(data_scoreboards)
```

```
## 'data.frame':    252 obs. of  7 variables:
##  $ MatchId      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Date         : Factor w/ 218 levels "December 12 18:00 CET",..: 203 204 205 206 214 215 216 217 2
##  $ HomeTeam     : Factor w/ 18 levels "ALBA Berlin",..: 9 14 7 15 17 2 1 11 16 5 ...
##  $ HomeScore    : int  89 87 78 81 58 64 85 82 71 79 ...
##  $ VisitingTeam : Factor w/ 18 levels "ALBA Berlin",..: 12 4 3 8 10 6 18 13 5 7 ...
##  $ VisitingScore: int  83 82 64 77 70 74 65 63 96 68 ...
##  $ Link         : Factor w/ 252 levels "https://www.euroleague.net/main/results/showgame?gamecode=1&s
```

Para el dataset `data_scoreboards` se han interpretado correctamenta las variables `MatchId`, `HomeScore` y `VisitingTeam` con el tipo de dato `int` ya que son variables cuantitativas discretas, mientras que el resto de variables (`Date`, `HomeTeam` y `VisitingTeam`) que son variables cualitativas nominales, se han interpretado como tipo factor.

Verificamos ahora el segundo dataset:

```
str(data_euroleague)
```

```
## 'data.frame':    6505 obs. of  20 variables:
##  $ MatchId     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Team        : Factor w/ 18 levels "ALBA Berlin",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ PlayerNumber: num  1 5 6 8 9 10 11 12 13 24 ...
##  $ PlayerName  : Factor w/ 306 levels "ABALDE, ALBERTO",..: 250 28 276 303 288 67 127 184 97 132 ...
##  $ Min         : Factor w/ 1910 levels "0","0:01","0:02",..: 1345 626 1502 99 1645 1 976 617 924 486
##  $ Pts         : Factor w/ 41 levels "-","0","1","10",..: 17 33 11 2 2 2 8 38 12 2 ...
##  $ X2FG        : Factor w/ 92 levels "-","0","0/1",..: 49 14 13 2 2 2 32 2 31 2 ...
##  $ X3FG        : Factor w/ 74 levels "-","0","0/1",..: 22 2 58 2 2 2 28 25 38 4 ...
##  $ FT          : Factor w/ 66 levels "-","0","0/1",..: 58 22 2 2 2 2 38 2 42 2 ...
##  $ O           : Factor w/ 10 levels "-","0","1","2",..: 2 5 3 3 2 2 3 2 5 3 ...
##  $ D           : Factor w/ 14 levels "-","0","1","10",..: 7 8 8 3 2 2 11 7 8 2 ...
##  $ T           : Factor w/ 19 levels "-","0","1","10",..: 12 16 14 12 2 2 17 12 16 3 ...
##  $ As          : Factor w/ 19 levels "-","0","1","10",..: 16 2 3 12 3 2 2 2 14 14 ...
##  $ St          : Factor w/ 8 levels "-","0","1","2",..: 2 4 6 3 2 2 4 3 2 2 ...
##  $ To          : Factor w/ 11 levels "-","0","1","2",..: 5 3 3 3 2 2 3 2 2 3 ...
##  $ Fv          : Factor w/ 7 levels "-","0","1","2",..: 3 2 2 2 2 2 2 2 2 2 ...
##  $ Ag          : Factor w/ 5 levels "-","0","1","2",..: 2 2 2 2 2 2 4 2 3 2 ...
##  $ Cm          : Factor w/ 7 levels "-","0","1","2",..: 3 4 7 4 4 2 4 3 7 3 ...
##  $ Rv          : Factor w/ 13 levels "-","0","1","10",..: 9 3 6 3 2 2 7 2 9 6 ...
##  $ PIR         : Factor w/ 57 levels "-","-1","-10",..: 24 55 18 36 2 13 16 56 28 36 ...
```

En el segundo dataset vemos que la mayoría de variables se han interpretado como tipo `factor` debido a que muchas contenían elementos vacíos. Como ya hemos solucionado este problema, vamos a proceder a asignar el tipo de dato correspondiente a cada variable:

```
# Variables cuantitativas discretas
data_euroleague$PlayerNumber <- as.integer(as.character(data_euroleague$PlayerNumber))
data_euroleague$Pts <- as.integer(as.character(data_euroleague$Pts))
data_euroleague$O <- as.integer(as.character(data_euroleague$O))
data_euroleague$D <- as.integer(as.character(data_euroleague$D))
data_euroleague$T <- as.integer(as.character(data_euroleague$T))
data_euroleague$As <- as.integer(as.character(data_euroleague$As))
data_euroleague$St <- as.integer(as.character(data_euroleague$St))
data_euroleague$To <- as.integer(as.character(data_euroleague$To))
data_euroleague$Fv <- as.integer(as.character(data_euroleague$Fv))
data_euroleague$Ag <- as.integer(as.character(data_euroleague$Ag))
data_euroleague$Cm <- as.integer(as.character(data_euroleague$Cm))
data_euroleague$Rv <- as.integer(as.character(data_euroleague$Rv))
data_euroleague$PIR <- as.integer(as.character(data_euroleague$PIR))
```

Verificamos que se han realizado los cambios de tipos de variable:

```
str(data_euroleague)
```

```
## 'data.frame':    6505 obs. of  20 variables:
##  $ MatchId     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Team        : Factor w/ 18 levels "ALBA Berlin",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ PlayerNumber: int  1 5 6 8 9 10 11 12 13 24 ...
##  $ PlayerName  : Factor w/ 306 levels "ABALDE, ALBERTO",..: 250 28 276 303 288 67 127 184 97 132 ...
##  $ Min         : Factor w/ 1910 levels "0","0:01","0:02",..: 1345 626 1502 99 1645 1 976 617 924 486
##  $ Pts         : int  22 4 17 0 0 0 14 6 18 0 ...
##  $ X2FG        : Factor w/ 92 levels "-","0","0/1",..: 49 14 13 2 2 2 32 2 31 2 ...
##  $ X3FG        : Factor w/ 74 levels "-","0","0/1",..: 22 2 58 2 2 2 28 25 38 4 ...
##  $ FT          : Factor w/ 66 levels "-","0","0/1",..: 58 22 2 2 2 2 38 2 42 2 ...
##  $ O           : int  0 3 1 1 0 0 1 0 3 1 ...
##  $ D           : int  2 3 3 1 0 0 6 2 3 0 ...
##  $ T           : int  2 6 4 2 0 0 7 2 6 1 ...
##  $ As          : int  6 0 1 2 1 0 0 0 4 4 ...
##  $ St          : int  0 2 4 1 0 0 2 1 0 0 ...
##  $ To          : int  3 1 1 1 0 0 1 0 0 1 ...
##  $ Fv          : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ Ag          : int  0 0 0 0 0 0 2 0 1 0 ...
##  $ Cm          : int  1 2 5 2 2 0 2 1 5 1 ...
##  $ Rv          : int  5 1 2 1 0 0 3 0 5 2 ...
##  $ PIR         : int  19 7 13 3 -1 0 11 8 22 3 ...
```

Hemos conseguido limpiar bastante los datos pero aún debemos realizar algunas modificaciones en algunas variables:

- **Min:** la variable minutos la pasaremos a formato decimal.
- **Tiros:** las variables tiros de dos (`X2FG`), tiros de tres (`X3FG`) y tiros libres (`FT`), se encuentran en formato "M/A" donde M es convertidos (*made*) y A intentados (*attempts*). Por cada una de ellas, crearemos por lo tanto dos variables, separando los aciertos de los intentos.

```
# Modificamos la variable tiempo
levels(data_euroleague$Min) <- c(levels(data_euroleague$Min), "0:00")
data_euroleague$Min[data_euroleague$Min=="0"] <- "0:00"

data_euroleague$Min <- sapply(strsplit(as.character(data_euroleague$Min),":"),
  function(x) {
    x <- as.numeric(x)
    x[1]+x[2]/60
  })
```

```
# Modificamos la variables tiros de 2
levels(data_euroleague$X2FG) <- c(levels(data_euroleague$X2FG), "0/0")
data_euroleague$X2FG[data_euroleague$X2FG=="0"] <- "0/0"

data_euroleague$x2M <- sapply(strsplit(as.character(data_euroleague$X2FG),"/"),
  function(x) {
    x <- as.integer(x)
    x[1]
  })

data_euroleague$x2A <- sapply(strsplit(as.character(data_euroleague$X2FG),"/"),
  function(x) {
    x <- as.integer(x)
    x[2]
  })

data_euroleague$X2FG <- NULL
```

```
# Modificamos la variables tiros de 3
levels(data_euroleague$X3FG) <- c(levels(data_euroleague$X3FG), "0/0")
data_euroleague$X3FG[data_euroleague$X3FG=="0"] <- "0/0"

data_euroleague$x3M <- sapply(strsplit(as.character(data_euroleague$X3FG),"/"),
  function(x) {
    x <- as.integer(x)
    x[1]
  })

data_euroleague$x3A <- sapply(strsplit(as.character(data_euroleague$X3FG),"/"),
  function(x) {
    x <- as.integer(x)
    x[2]
  })

data_euroleague$X3FG <- NULL
```

```r
# Modificamos la variables tiros libres
levels(data_euroleague$FT) <- c(levels(data_euroleague$FT), "0/0")
data_euroleague$FT[data_euroleague$FT=="0"] <- "0/0"

data_euroleague$FTM <- sapply(strsplit(as.character(data_euroleague$FT),"/"),
  function(x) {
    x <- as.integer(x)
    x[1]
  })

data_euroleague$FTA <- sapply(strsplit(as.character(data_euroleague$FT),"/"),
  function(x) {
    x <- as.integer(x)
    x[2]
  })

data_euroleague$FT <- NULL
```

Realizamos la última comprobación de los tipos de datos de las variables del dataset `data_euroleague`:

```r
sapply(data_euroleague, function(x) class(x))
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##    "integer"     "factor"    "integer"     "factor"    "numeric"    "integer"
##            0            D            T           As           St           To
##    "integer"    "integer"    "integer"    "integer"    "integer"    "integer"
##           Fv           Ag           Cm           Rv          PIR          x2M
##    "integer"    "integer"    "integer"    "integer"    "integer"    "integer"
##          x2A          x3M          x3A          FTM          FTA
##    "integer"    "integer"    "integer"    "integer"    "integer"
```

Por tanto, el resumen es el siguiente:

- Variables cualitativas nominales: `Team`, `PlayerName` (tipo `factor`).
- Variables cuantitativas discretas: `MatchId`, `PlayerNumber`, `Pts`, `O`, `D`, `T`, `As`, `St`, `To`, `Fv`, `Ag`, `Cm`, `Rv`, `PIR`, `x2M`, `x2A`, `x3M`, `x3A`, `FTM`, `FTA` (tipo `integer`).
- Variables cuantitativas continuas: `Min` (tipo `numeric`).

3.3 Agregación de datos

Como previamente comentamos en el apartado de integración y selección de datos de interés, la idea principal es crear un nuevo dataset que agrupe las estadísticas para cada equipo y partido. Por tanto, el resultado será la suma de las estadísticas de todos los jugadores del equipo para cada partido. Este nuevo dataset se llamará `data_team`:

```
data_team <- as.data.frame(data_euroleague %>% group_by(Team, MatchId) %>%
  select_if(is.numeric) %>%
  summarise_each(list(sum)))

data_team[c("PlayerNumber")] <- NULL
```

Mostramos los primeros registros de este nuevo dataset:

```
head(data_team)
```

```
##          Team MatchId Min Pts  O  D  T As St To Fv Ag Cm Rv PIR x2M x2A x3M x3A
## 1 ALBA Berlin       7 200  85 11 31 42 23  5 10  4  4 19 17 106  19  40  13  29
## 2 ALBA Berlin      16 225 105 14 27 41 28  4  7  2  3 21 19 127  25  48  14  30
## 3 ALBA Berlin      26 200  84  6 18 24 21  9 17  2  4 27 24  75   8  24  14  35
## 4 ALBA Berlin      34 200  66 14 19 33 20 10 15  0  1 21 22  64  16  40   7  24
## 5 ALBA Berlin      39 200  78 13 21 34 15 10 13  2  2 20 21  81   9  28  14  37
## 6 ALBA Berlin      54 200  71 11 23 34 14  8 20  2  4 25 17  56  20  38   8  26
##   FTM FTA
## 1   8   8
## 2  13  15
## 3  26  30
## 4  13  22
## 5  18  20
## 6   7  12
```

A primera vista este dataset se ha creado correctamente. Como prueba de la calidad de los datos y del proceso de transformación podemos comprobar que los minutos de los partidos son correctos:

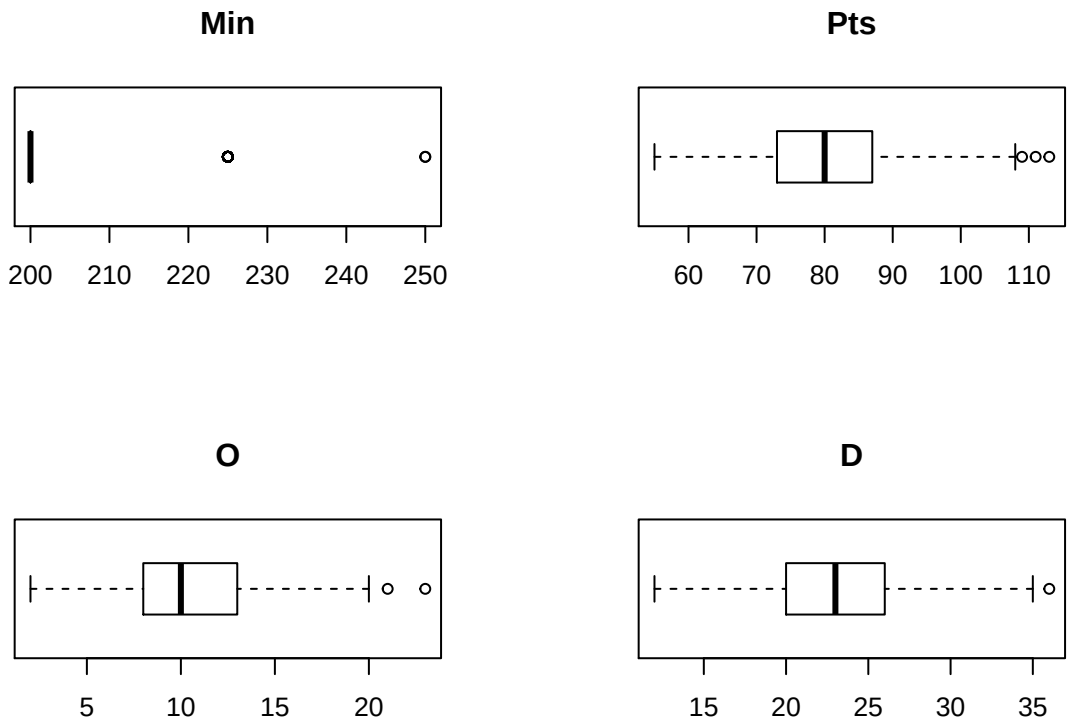
```
5 jugadores * 10 mintuos * 4 cuartos = 200 minutos
```

Para los partidos con prórroga se deben sumar 5 minutos por cada parte de tiempo extra y multiplicar por los 5 jugadores. Por eso vemos partidos con 225 minutos.

3.4 Identificación y tratamiento de valores extremos

Los valores extremos (*extreme scores* o *outliers*) son aquellos datos que se encuentran muy alejados de la distribución normal de una variable o población, normalmente 3 desviaciones estándar con respecto a la media. Para encontrar estos valores extremos se utiliza los diagramas de cajas o *boxplot*, que permiten identificar si existen valores extremos para una determinada variable. Crearemos uno de estos diagramas para cada variable cuantitativa de nuestro dataset de estadísticas de equipo:

```
par(mfrow = c(2,2))
for (i in c(seq(3,6))) {
  outlier <- boxplot(data_team[,i], main = colnames(data_team)[i], horizontal = TRUE)
  assign(paste("outlier", i, sep = "."), outlier)
}
```




```
par(mfrow = c(1,1))
```

```
# Valores extremos de la variable Min
outlier.3$out
```

```
##  [1] 225 250 225 225 225 225 225 225 225 225 225 225 225 225 225 225 225 225 225 225
## [20] 225 225 225 250 225 225 225 225 225 225 225 225 225 225 225 225
```

```
# Valores extremos de la variable Pts
outlier.4$out
```

```
## [1] 113 109 111
```

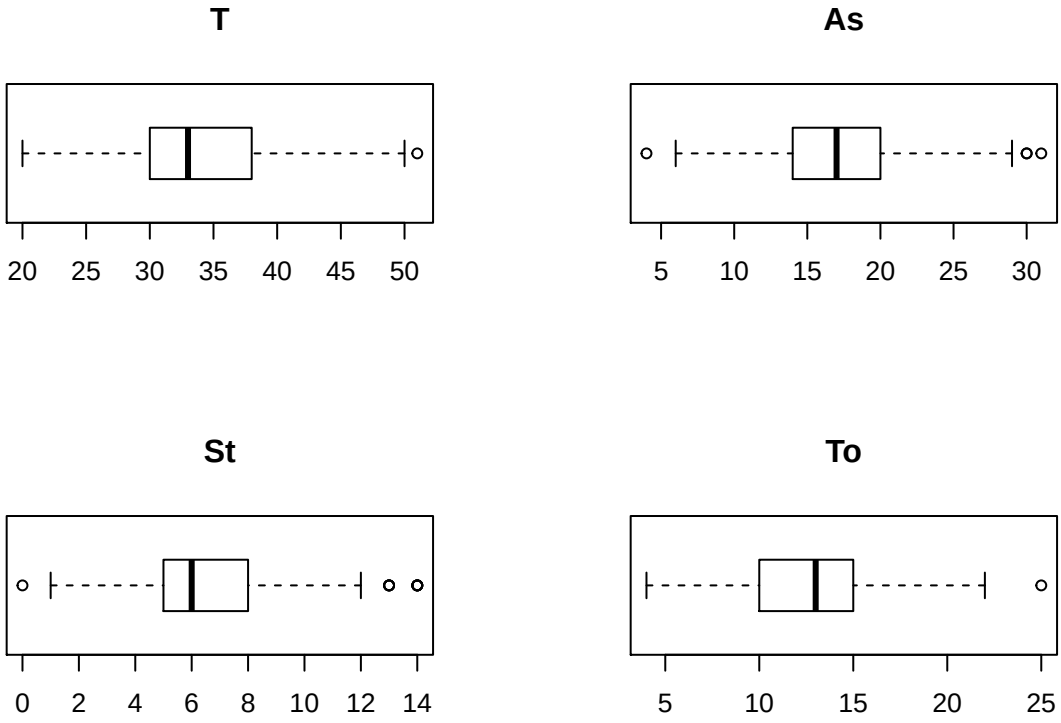
```
# Valores extremos de la variable O
outlier.5$out
```

```
## [1] 21 23
```

```
# Valores extremos de la variable D
outlier.6$out
```

```
## [1] 36
```

```r
par(mfrow = c(2,2))
for (i in c(seq(7,10))) {
  outlier <- boxplot(data_team[,i], main = colnames(data_team)[i], horizontal = TRUE)
  assign(paste("outlier", i, sep = "."), outlier)
}
```




```r
par(mfrow = c(1,1))
```

```r
# Valores extremos de la variable T
outlier.7$out
```

```
## [1] 51
```

```r
# Valores extremos de la variable As
outlier.8$out
```

```
## [1]  4 30 31 30
```

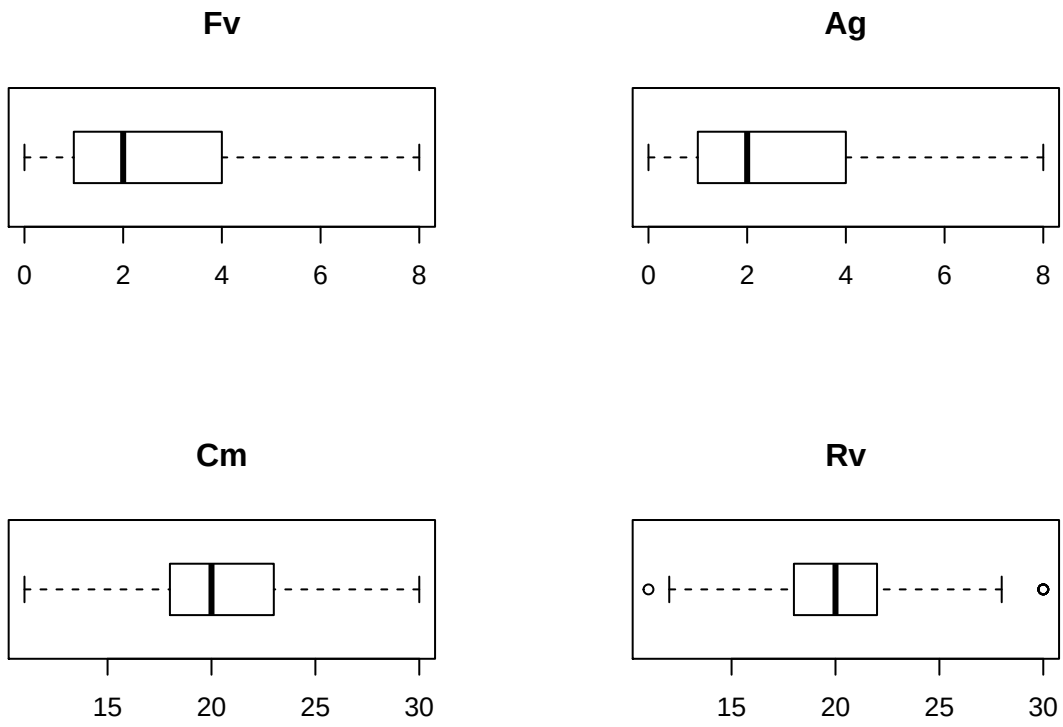
```r
# Valores extremos de la variable St
outlier.9$out
```

```
##  [1]  0 14 14 13 13 14 14 13 13 14
```

```r
# Valores extremos de la variable To
outlier.10$out
```

```
## [1] 25
```

```r
par(mfrow = c(2,2))
for (i in c(seq(11,14))) {
  outlier <- boxplot(data_team[,i], main = colnames(data_team)[i], horizontal = TRUE)
  assign(paste("outlier", i, sep = "."), outlier)
}
```

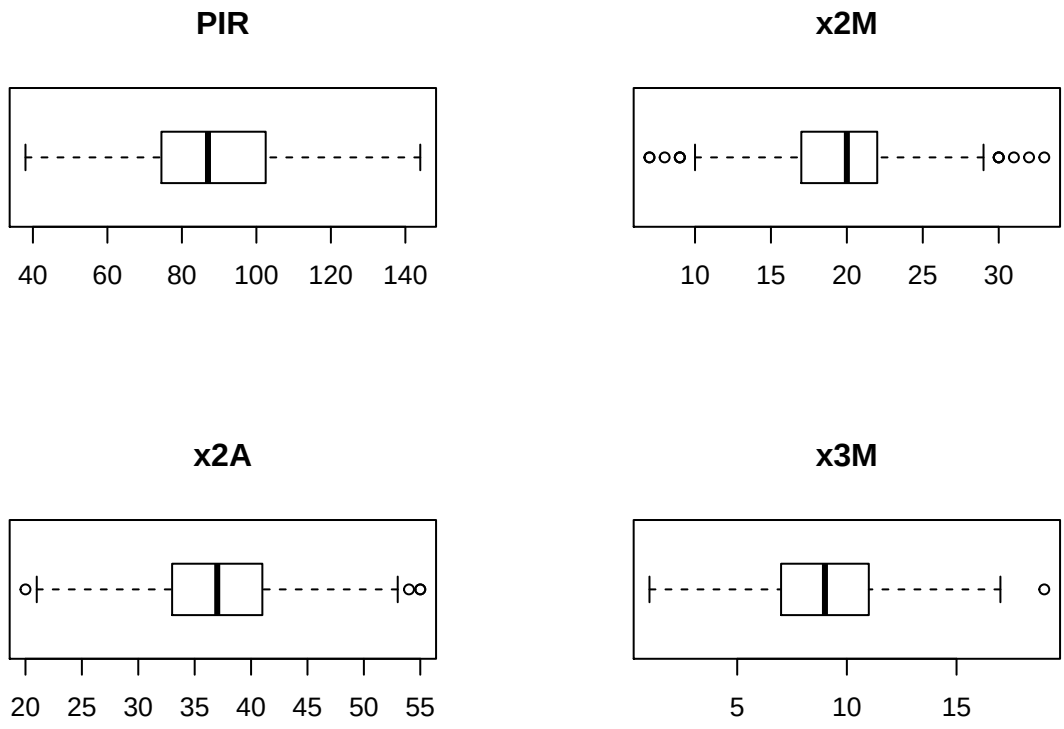



```r
par(mfrow = c(1,1))
```

```r
# Valores extremos de la variable Rv
outlier.14$out
```

```
## [1] 30 30 11 30 30 30 30
```

```r
par(mfrow = c(2,2))
for (i in c(seq(15,18))) {
  outlier <- boxplot(data_team[,i], main = colnames(data_team)[i], horizontal = TRUE)
  assign(paste("outlier", i, sep = "."), outlier)
}
```




```r
par(mfrow = c(1,1))
```

```r
# Valores extremos de la variable x2M
outlier.16$out
```

```
##  [1]  8  9 33 31 32  9  7 30 30 30  7  9
```

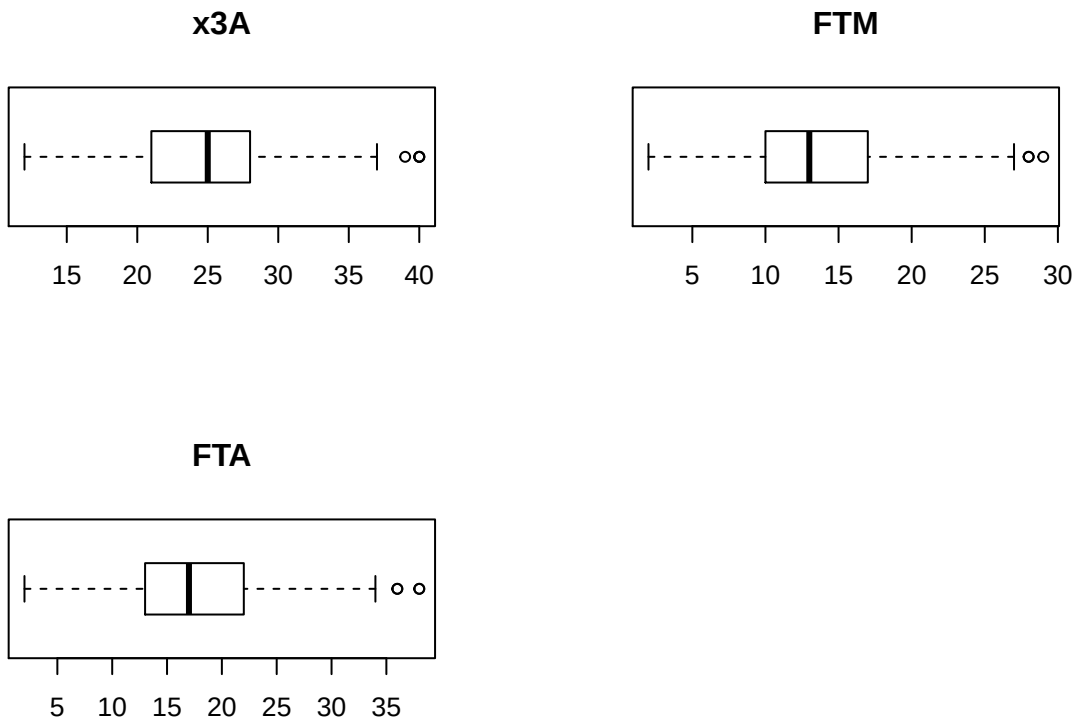
```r
# Valores extremos de la variable x2A
outlier.17$out
```

```
## [1] 55 20 54 55
```

```r
# Valores extremos de la variable x3M
outlier.18$out
```

```
## [1] 19
```

```r
par(mfrow = c(2,2))
for (i in c(seq(19,21))) {
  outlier <- boxplot(data_team[,i], main = colnames(data_team)[i], horizontal = TRUE)
  assign(paste("outlier", i, sep = "."), outlier)
}
par(mfrow = c(1,1))
```




```r
# Valores extremos de la variable x3A
outlier.19$out
```

```
## [1] 39 40 40 40
```

```r
# Valores extremos de la variable FTM
outlier.20$out
```

```
## [1] 28 29 28
```

```r
# Valores extremos de la variable FTA
outlier.21$out
```

```
## [1] 36 38 38 36
```

Mediante los gráficos de caja anteriores hemos obtenido la siguiente información:

- **Variables sin *outliers*:** `Fv`, `Ag`, `Cm` y `PIR`.
- **Variables con *outliers*:** `Min`, `Pts`, `O`, `D`, `T`, `As`, `St`, `To`, `Rv`, `x2M`, `x2A`, `x3M`, `x3A`, `FTM` y `FTA`.

Tras revisar todos individualmente todos los valores extremos hemos determinado que todos ellos son valores posibles y que están dentro del rango del atributo, por lo que se considerarán como valores legítimos y se contemplarán en los análisis posteriores.

Un ejemplo claro es el de la variable que representa los minutos, donde la mayoría de partidos duran 200 minutos con excepción de los que tienen prórroga, como ya explicamos anteriormente cuando creamos el dataset con las estadísticas de los equipos.

Otro ejemplo de valores extremos legítimos serían los de la variable que representa los puntos obtenidos en un partido por un equipo: 113, 109, 111. Por lo que vemos, en la Euroliga podemos considerar como fuera de lo habitual que un equipo meta más de 110 puntos. Si realizaramos el mismo análisis sobre equipos de la NBA veríamos que este valor cambiaría bastante ya que en la liga americana es más habitual que los equipos consigan marcadores más abultados. Si hubiésemos obtenido un *outlier* de 200 o 300 para la variable puntos si hubiésemos tenido que eliminar ese registro o revisar si se produjo un error en el proceso de *web scraping* o en la carga de datos, ya que sería un dato que se aleja bastante de la normalidad.

---

4 Enriquecimiento de los datos

4.1 Datos del oponente.

Durante un encuentro de baloncesto, es tan importante medir lo que tú equipo ha hecho como lo que tú equipo ha recibido. Por lo tanto, buscamos enriquecer la información por partido de cada equipo añadiendo la información del oponente. Para ello, utilizaremos el dataset `data_scoreboards` para completar así los registros:

```
names(data_scoreboards)[names(data_scoreboards) == "Id"] <- "MatchId"
names(data_scoreboards)[names(data_scoreboards) == "HomeTeam"] <- "Team"
names(data_scoreboards)[names(data_scoreboards) == "VisitingTeam"] <- "Opponent"

df_1 <- merge(data_team, data_scoreboards, by=c("MatchId","Team"))
df_1["Local"] <- 1

names(data_scoreboards)[names(data_scoreboards) == "Team"] <- "HomeTeam"
names(data_scoreboards)[names(data_scoreboards) == "Opponent"] <- "Team"
names(data_scoreboards)[names(data_scoreboards) == "HomeTeam"] <- "Opponent"
df_2 <- merge(data_team, data_scoreboards, by=c("MatchId","Team"))
df_2["Local"] <- 0

data_team <- bind_rows(df_1,df_2)

data_team[c("Date")] <- NULL
data_team[c("HomeScore")] <- NULL
data_team[c("VisitingScore")] <- NULL
data_team[c("Link")] <- NULL

data_team2 <- data_team [c("MatchId","Team","Min","Pts","O","D","T","As","St","To",
                           "Fv","Ag","Cm","Rv","PIR","x2M","x2A","x3M","x3A","FTM",
                           "FTA")]

names(data_team2)[names(data_team2) == "Pts"] <- "Pts_Opp"
names(data_team2)[names(data_team2) == "O"] <- "O_Opp"
names(data_team2)[names(data_team2) == "D"] <- "D_Opp"
names(data_team2)[names(data_team2) == "T"] <- "T_Opp"
names(data_team2)[names(data_team2) == "As"] <- "As_Opp"
names(data_team2)[names(data_team2) == "St"] <- "St_Opp"
names(data_team2)[names(data_team2) == "To"] <- "To_Opp"
names(data_team2)[names(data_team2) == "Fv"] <- "Fv_Opp"
names(data_team2)[names(data_team2) == "Ag"] <- "Ag_Opp"
names(data_team2)[names(data_team2) == "Cm"] <- "Cm_Opp"
names(data_team2)[names(data_team2) == "Rv"] <- "Rv_Opp"
names(data_team2)[names(data_team2) == "PIR"] <- "PIR_Opp"
names(data_team2)[names(data_team2) == "x2M"] <- "x2M_Opp"
names(data_team2)[names(data_team2) == "x2A"] <- "x2A_Opp"
names(data_team2)[names(data_team2) == "x3M"] <- "x3M_Opp"
names(data_team2)[names(data_team2) == "x3A"] <- "x3A_Opp"
names(data_team2)[names(data_team2) == "FTM"] <- "FTM_Opp"
names(data_team2)[names(data_team2) == "FTA"] <- "FTA_Opp"

data_teamopp <- merge(data_team, data_team2, by.x=c("MatchId","Opponent"),
                      by.y=c("MatchId","Team"))
data_teamopp["y.min"]<-NULL
```

4.2 Creación de nuevas métricas de equipo

Durante los últimos años, se ha dado una vuelta al análisis de la estadísticas en baloncesto. Como los diferentes equipos, de acuerdo a su estilo, juegan a distintos ritmos, no podemos usar los promedios por partido para compararlos. Para poder lograr las comparaciones hay definir el concepto de las posesiones, que es la base de estos cálculos. El basket es un juego en el que ambos equipos se alternan la posesión de la pelota. El equipo que aproveche mejor sus posesiones será el equipo ganador. Se entiende que una posesión termina con un tiro al aro, una pérdida de balón o un tiro libre. Allí el balón pasa al rival y la posesión se termina. ¿Qué ocurre si el equipo falla el tiro de campo pero captura el rebote ofensivo? Hoy por hoy, la mayoría de los estadistas de baloncesto consideran que no se le debe anotar una nueva posesión al equipo, sino considerar que la misma posesión continúa.

Para tener en cuenta lo anteriormente mencionado, a continuación definiremos nuevas métricas.

4.2.1 Ritmo (Pace)

Nos da una idea del ritmo de juego del equipo, expresado en cantidad de posesiones por juego que utiliza. Hay equipos que corren más y equipos que prefieren el juego estático. Por eso las estadísticas por juego no sirven para comparar equipos. Las posesiones se calculan con la siguiente fórmula:

$$Pos = FGA - OR + TO + (FTA * 0.4)$$

Donde:

- Pos: posesiones
- FGA (field goal attempts): lanzamientos de campo (tanto de 2 y de 3)
- OR (ofensive rebounds): rebotes ofensivos
- TO (turnovers): pelotas perdidas
- FTA (free throw attempts): tiros libres lanzados

```
data_teamopp["Poss"] <- data_teamopp["x2A"] + data_teamopp["x3A"] -
  data_teamopp["O"] + data_teamopp["To"] + (0.4*data_teamopp["FTA"])
```

4.2.2 Eficiencia ofensiva (Rating ofensivo)

Habitualmente se evalúa la ofensiva en puntos convertidos por juego, lo cual es una manera un tanto absurda. Si pensamos el juego como una serie de posesiones, el equipo que más puntos convierta en sus posesiones, será el más efectivo. Se multiplica por 100 para expresar los puntos cada 100 posesiones, y no manejar números con decimales. Así:

$$OffensiveRating = (puntos/posesiones) * 100$$

```
data_teamopp ["Off_Rat"] <- +data_teamopp["Pts"]/data_teamopp["Poss"]
```

4.2.3 Eficiencia defensiva (Rating defensivo)

Así como medimos la eficiencia ofensiva en base a puntos convertidos cada 100 posesiones, podemos medir la defensa en base a puntos recibidos (o puntos del oponente) cada 100 posesiones del equipo contrario.

$$DefensiveRating = (puntos_{oponente}/posesiones) * 100$$

```
data_teamopp ["Def_Rat"] <- +data_teamopp["Pts_Opp"]/data_teamopp["Poss"]
```

4.2.4 Porcentaje efectivo de tiros de campo (eFG%)

Esta estadística ajusta los tiros de campo dandole el valor extra (un punto más) a los triples. Esto corrige el FG% común que subestima a los triples. Por ejemplo, si un jugador ha hecho 2/5 en T2 y 1/4 en T3, habrá convertido 3/9 en tiros de campo (33%), que es similar a si hubiera metido 3/5 de T2 y 0/4 en T3. Sin embargo, en el primer supuesto ha conseguido más puntos para el equipo. Así, el eFG% del primero será 44.4% que se ajusta más a la realidad. La fórmula, por tanto es:

$$eFG\% = \frac{(FGM+0.5*3PM)}{FGA}$$

Donde:

- FGM (field goal made): tiros de campo convertidos.
- 3PM (3 points made): triples convertidos.
- FGA (field goal attempts): tiros de campo intentados.

```
data_teamopp ["eFG"] <- (data_teamopp["x2M"] + 1.5*data_teamopp["x3M"]) /
  (data_teamopp["x2A"] + data_teamopp["x3A"])

data_teamopp ["eFG_Opp"] <- (data_teamopp["x2M_Opp"] + 1.5*data_teamopp["x3M_Opp"]) /
  (data_teamopp["x2A_Opp"]+data_teamopp["x3A_Opp"])
```

4.2.5 Lanzamientos reales (True Shooting)

Esta métrica tiene en cuenta los dobles, triples y tiros libres, para dar una idea de cómo tira el jugador globalmente. Ejemplo: en la 2009-2010, Martin Leiva quedó #8 en FG% con 58,72. Pero si le agregamos los libres, cae al puesto #91 con 54.8%. La fórmula es:

$$TS = \frac{puntos}{2*(FGA+0.44*FTA)}$$

Donde:

- FGA (field goal attempts): lanzamientos de cancha intentados
- FTA (free throw attempts): tiros libres intentados

```
data_teamopp ["TS"] <- data_teamopp["Pts"] /
  (data_teamopp["x2A"] + data_teamopp["x3A"] + 0.44*data_teamopp["FTA"])

data_teamopp ["TS_Opp"] <- data_teamopp["Pts_Opp"] /
  (data_teamopp["x2A_Opp"] + data_teamopp["x3A_Opp"] + 0.44*data_teamopp["FTA_Opp"])
```

4.2.6 Rebotes

Los rebotes totales de un equipo son de poco valor. Capturar un rebote ofensivo requiere diferentes habilidades que capturar uno defensivo, por lo que deben analizarse por separado.

Tener en cuenta el número absoluto de rebotes conseguidos, o el promedio de rebotes por partido, nos puede llevar a errores, ya que los rebotes disponibles dependen de la efectividad: si un equipo falla poco, hay pocos rebotes por tomar.

Ejemplo: el equipo A obtuvo en un partido 20 rebotes defensivos. Si el equipo B falló 30 lanzamientos (o sea que hubo 30 rebotes en el aro defensivo de A) entonces A capturó 66% de los rebotes en su aro (20 de 30). Pero si B erró 25 tiros, A tomó 80% de los rebotes (20 de 25).

Esto hace que, para evaluarlo correctamente, definamos: DR% como porcentaje de rebotes defensivos y OR% porcentaje de rebotes ofensivos.

4.2.6.1 Tasa de rebotes ofensivos

$$\%derebotesofensivos = [OR/(OR + opDR)] * 100$$

Donde:

- OR (ofensive rebounds): rebotes ofensivos.
- Op DR (oponent defensive rebounds): rebotes defensivos del rival.

4.2.6.2 Tasa de rebotes defensivos

$$\%derebotesdefensivos = [DR/(DR + opOR)] * 100$$

Donde:

- DR (defensive rebounds): rebotes defensivos.
- Op OR (oponent ofensive rebounds): rebotes ofensivos del rival.

```
data_teamopp ["Off_Reb"] <- data_teamopp["O"] /  (data_teamopp["O"]+data_teamopp["D_Opp"])

data_teamopp ["Def_Reb"] <- data_teamopp["D"] /  (data_teamopp["D"]+data_teamopp["O_Opp"])
```

4.2.7 Porcentaje de asistencias y pérdidas

Al igual que con los rebotes y otras estadísitcas, las asistencias por juego no son un buen parámetro, ya que dependen del ritmo de juego. Es más preciso calcular las asistencias expresadas en posesiones terminan con una pérdida de balón. Habitualmente expresado en porcentaje.

4.2.8 Porcentaje de asistencias

Se calculan mediante la siguiente fórmula:

$$\%deasistencias = (asistencias/posesiones) * 100$$

```
data_teamopp ["Pct_ass"] <- data_teamopp["As"] /  (data_teamopp["Poss"])

data_teamopp ["Pct_ass_opp"] <- data_teamopp["As_Opp"] /  (data_teamopp["Poss"])
```

4.2.8.1 Porcentaje de pérdidas

Lo mismo ocurre con las pérdidas. No es lo mismo perder 10 pelotas en un partido en que hubo 100 posesiones, que en uno que hubo 80. Por eso es mejor calcular las pérdidas cada 100 posesiones. El valor ideal depende del ritmo de juego, pero podríamos decir que el objetivo sería tener menos de 15% de TO y provocar en el oponente más de 15%. La fórmula es:

$$\%depérdidas = (pelotaspérdidas/posesiones) * 100$$

```
data_teamopp ["Pct_To"] <- data_teamopp["To"] /  (data_teamopp["Poss"])

data_teamopp ["Pct_To_opp"] <- data_teamopp["To_Opp"] /  (data_teamopp["Poss"])
```

4.2.8.2 Tiros libres, respecto a tiros de campo (FTM/FGA)

Es simplemente una manera de expresar el número de veces que un equipo va a la línea y cuántas veces envía al oponente a la línea. Esta considerado (junto con el *effective field goal percentage*, la tasa de rebotes ofensivos y la tasa de pérdidas) uno de los cuatro factores con los que se miden los partidos. La fórmula es:

$$Libresporlanzamientosdecancha = (libresconvertidos/tirosdecanchaintentados) * 100$$

```
data_teamopp["FTR"] <- data_teamopp["FTM"] /  (data_teamopp["x2A"]+ data_teamopp["x3A"])

data_teamopp["FTR_opp"] <- data_teamopp["FTM_Opp"] /
  (data_teamopp["x2A_Opp"]+ data_teamopp["x3A_Opp"])
```

4.2.9 Victoria

Otro aspecto fundamental en un partido de baloncesto y que necesitaremos para realizar los análisis es saber quién ganó el partido. Esto se puede obtener simplemente comparando el resultado final y determinando quién metió más puntos:

```
data_teamopp$Win <- ifelse(data_teamopp$Pts>=data_teamopp$Pts_Opp, 1, 0)
```

4.2.10 Triunfos esperados (Expected wins)

Para ganar, obviamente hay que meter más puntos que el rival. Existe un cálculo (comprobado en la NBA, baloncesto FIBA y universitario) que de acuerdo a los puntos convertidos y recibidos, se puede estimar la cantidad de partidos que habría que haber ganado. Suele correlacionar muy bien con la realidad.

$$Triunfosesperados = eficienciaofensiva^{14}/(eficienciaofensiva^{14} + eficienciadefensiva^{14})$$

```
data_teamopp ["Expected_Wins"] <- data_teamopp["Off_Rat"]^14 /
  (data_teamopp["Off_Rat"]^14+data_teamopp["Def_Rat"]^14)
```

4.3 Creación de nuevas métricas de jugadores

Una vez definidas los nuevos estadísticos para el dataset de equipos, realizaremos el mismo trabajo para el dataset de jugadores. De esta forma contaremos con nuevas variables que permitan determinar con mayor precisión la calidad y participación de los jugadores en un partido.

Para obtener estos estadísticos nos hemos basado en un artículo de la página web **Bleacher Report** donde se explican métricas y estadísticas avanzadas para la NBA. Se puede acceder al artículo haciendo click aquí.

Como necesitaremos datos del equipo, de nuevo el primer paso es enriquecer la tabla cruzando los datos de players y de equipos por partidos.

```
data_players <- merge(data_euroleague, data_teamopp, by=c("MatchId","Team"))
```

4.3.1 Assist Percentage (AST%)

$$A/(((MP/(TMP/5))*TFG)-FG)$$

Donde:

- A: Asistencias.
- MP: Minutos jugados por el jugador.
- TMP: Minutos jugados por el equipo.
- TFG: Tiros de campo del equipo.
- FG: Tiros de campo del jugador.

Explicación

Si bien las asistencias en sí mismas son bastante útiles para observar, el porcentaje de asistencias es una estadística mucho mejor para describir la habilidad del jugador. Las estadísticas de asistencia se pueden completar de dos maneras: la cantidad de tiempo que un jugador está en la cancha y el ritmo al que juega su equipo.

Un jugador en un equipo de ritmo rápido como Miami Heat o Washington Wizards tendrá más oportunidades ofensivas para acumular asistencias, lo que generará un mayor número de asistencias por partido y por minuto.

Del mismo modo, un jugador que juega 35 minutos por juego tendrá más oportunidades de generar asistencias que un jugador en la cancha durante 20 minutos por partido. Puede parecer de sentido común que promediar 10 asistencias por juego en 20 minutos de acción por competencia es más impresionante que promediar 10 asistencias por juego en 35 minutos de acción por competencia, pero esa distinción se pierde cuando solo se citan asistencias por partido.

La métrica esencialmente estima (estima, no calcula) qué porcentaje de tiros realizados por los compañeros de equipo fueron asistidos por un jugador mientras estaba en la cancha.

Limitaciones

Sin tener el detalle de las jugadas por partido, lo mejor que puede hacer esta estadística es proporcionar una estimación del porcentaje antes mencionado. No hay forma de medir y dar crédito a los buenos pases que resultaron en tiros liberados pero fallados o pérdidas por parte del receptor.

Al igual que con cualquier estadística de asistencia, un jugador está a merced de la habilidad de sus compañeros de equipo.

```
data_players$Assist_pct <- data_players$As.x/(((data_players$Min/(data_players$Min.x/5))*
                                                 (data_players$x2A.y+data_players$x3A.y))-
                                                (data_players$x2A.x+data_players$x3A.x))
```

4.3.2 Turnover Percentage (TOV%)

$$100 * TO/(FGA + 0.44 * FTA + TO)$$

Donde:

- TO: Pérdidas
- FGA: Tiros de Campo intentados.
- FTA: Tiros Libres intentados.

Explicación*

Sin embargo, antes de explicar exactamente qué hace esta estadística, nos gustaría centrarnos en el denominador entre paréntesis del cálculo anterior: (FGA + 0.44 * FTA + TO). Esta expresión matemática es la mejor manera de cuantificar el número de resultados de juego en los que estuvo involucrado un jugador sin realmente contar.

Hay tres formas en que un jugador puede participar en el resultado final de una posesión. Pueden intentar un tiro de campo, pueden terminar en la línea de personal o pueden dar perder el balón. Sin embargo, simplemente sumar esos tres resultados no proporciona el número de posesiones porque los tiradores pueden intentar uno, dos o tres tiros libres en cualquier posesión dada. En base a los últimos años, se ha estimado que el coeficiente de tiros libres que representan el último tiro es el 44% de los tirados.

Entonces, ahora que lo hemos eliminado, todo lo que esta estadística realmente hace es calcular el número de pérdidas de balón que un jugador hará en 100 jugadas individuales.

Las pérdidas de balón y las pérdidas de balón por partido dependen una vez más del ritmo de juego y de la cantidad de tiempo que un jugador pasa en la cancha. Esta estadística de tasas elimina esos efectos y se enfoca únicamente en el porcentaje de veces que un jugador pierde la pelota en comparación con la cantidad de veces que está involucrado en el juego de manera directa.

Limitaciones

El porcentaje de pérdidas aún no puede tener en cuenta los pases que un jugador hace que no generan pérdidas de balón (es decir, asiste o pasa a otros jugadores que fallan un tiro o no tiran).

Por lo tanto, sigue siendo una estadística bastante limitada.

```
data_players$Turnover_pct<-ifelse((data_players$x2A.x+data_players$x3A.x+
                                     0.44*data_players$FTA.x+data_players$To.x)==0,0,
                                  data_players$To.x/(data_players$x2A.x+
                                                     data_players$x3A.x+
                                                     0.44*data_players$FTA.x+
                                                     data_players$To.x))
```

4.3.3 Offensive Rebound Percentage (ORB%)

$$100 * (ORB * (TMP/5)/(MP * (TORB + ODRB))$$

Donde:

- ORB: Rebotes ofensivos.
- TMP: Minutos jugados por el equipo.
- MP: Minutos del jugador.
- TORB: Rebotes ofensivos del equipo.
- ODRB: Rebotes ofensivos del equipo rival.

Explanation

Cada vez que un tiro sale del aro, hay cuatro resultados posibles: el balón podría salirse de los límites del campo y contarse como un rebote del equipo defensivo; la pelota podría rebotar de un jugador defensivo y

salirse de los límites para ser contada como un rebote ofensivo del equipo; la pelota podría ser caputra por un jugador defensivo y contarse como un rebote defensivo; o un jugador ofensivo podría capturar la pelota y contarse como un rebote ofensivo.

Si se suman los cuatro resultados, cuentas todos los posibles rebotes en un juego.

El porcentaje de rebote ofensivo calcula el porcentaje de rebotes ofensivos disponibles que un jugador toma mientras está en la cancha.

Al igual que las dos anteriores, no tiene en cuenta el ritmo o el volumen.

Limitaciones

De nuevo, sin tener el detalle de jugada a jugada, es simplemente una estimación, aunque precisa.

De manera análoga, definimos el porcentaje de rebotes defensivos.

```
data_players$Offensive_Reb_pct <- data_players$O.x*(data_players$Min.x/5)/
  (data_players$Min*(data_players$O.y+data_players$D_Opp))

data_players$Defensive_Reb_pct <- data_players$D.x*(data_players$Min.x/5)/
  (data_players$Min*(data_players$D.y+data_players$O_Opp))
```

4.3.4 Effective Field-Goal Percentage (eFG%)

$$(FG + 0.5 * 3P)/FGA$$

Donde:

- FG: Tiros de Campo convertidos.
- 3P: Triples convertidos.
- FGA: Intentos de tiro de campo.

Explicación

El porcentaje de tiro de campo (% FG) fue una de las mejores estadísticas para estimar la capacidad de tiro hasta la popularización de esta métrica a principio de los años dos mil.

La única diferencia entre las dos estadísticas es que los triples tienen mayor peso en % de eFG. En vista de que los tres puntos valen tres puntos y los dos puntos valen dos puntos, esto tiene sentido.

Limitaciones

Si bien eFG% es una buena métrica, no tiene en cuenta los tiros libres.

```
data_players$eFG_player<-ifelse((data_players$x3A.x+data_players$x2A.x)==0,0,
                                (data_players$x2M.x+1.5*data_players$x3M.x)/
                                  (data_players$x3A.x+data_players$x2A.x))
```

4.3.5 True Shooting Percentage (TS%)

$$PT/(2 * (FGA + 0.44 * FTA))$$

Donde:

- PT: Puntos
- FGA: Tiros de campo intentados.
- FTA: Tiros libres intentaos.

Explicación

Hay tres formas en que un jugador puede anotar: triples, dos puntos y tiros libres. Simplemente tiene sentido que la mejor medida del porcentaje de tiro tomase en cuenta las tres posibles manera de anotación.

Como puede ver en los "Intentos de tiros de campo" e "Intentos de tiro libre" en el cálculo, el% de TS claramente al menos tiene en cuenta dos puntos y tiros libres. En cuanto al multiplicador 0.44, se aplica el mismo razonamiento que para el porcentaje de pérdidas.

Los triples son un poco más difíciles de encontrar en la fórmula, pero todavía están allí, escondidos dentro de la palabra "Puntos". El máximo% de TS es en realidad 150 por ciento y solo se puede lograr si un jugador realiza todos y cada uno de sus tiros y todos son del triple. Por ejemplo, si un jugador tira y anota una sola vez en un partido y es desde el triple, la fórmula se leerá y simplificará de la siguiente manera: 3 / (2 * 1 + 0.44 * 0)) = 3/2 = 1.5.

Debido a que esta estadística tiene en cuenta todo, es fácilmente la mejor medida de capacidad de tiro que tenemos.

Limitaciones

Además de tener en cuenta la capacidad de tiro y, por lo tanto, no calificar como una evaluación general de la capacidad ofensiva de un jugador, la única limitación verdadera del TS% es que es posible (aunque extremadamente improbable) tener un TS% de más del 100 por ciento.

Además, al igual que con el porcentaje efectivo de tiros de campo, todos los intentos fallidos cuentan igual.

```
data_players$TS_player <- ifelse((2*(data_players$x3A.x+data_players$x2A.x+
                                  0.44*data_players$FTA.x))==0,0,data_players$Pts.x/
                                   (2*(data_players$x3A.x+data_players$x2A.x
                                       +0.44*data_players$FTA.x)))
```

4.3.6 Usage Rate (USG%)

$$100 * ((FGA + 0.44 * FTA + TO) * (TMP/5))/(MP * (TFGA + 0.44 * TFTA + TTO))$$

donde * FGA=Intentos de tiro de campo * FTA=Intentos de tiro libre. * TO=Pérdidas. * TMP=Minutos jugados por equipo. * MP=Minutos jugados. * TFGA=Tiros de campo intentados del equipo. * TFTA=Tiros libres intentados del equipo. * TTO=Pérdidas del equipo.

Explicación

Este puede ser el cálculo de métricas más complicado hasta ahora, pero el concepto detrás de esto es realmente bastante simple. La tasa de uso calcula en qué porcentaje de las jugadas de equipo estuvo involucrado un jugador mientras estaba en la cancha, siempre que la jugada termine en uno de los tres resultados siguientes: tiro de campo, tiro libre o pérdida.

En promedio, por lo tanto, un jugador tendrá una tasa de uso del 20 por ciento.

Limitaciones

Aquí solo se miden los resultados reales, por lo que queda bastante fuera. Por ejemplo, un jugador como Ricky Rubio, que prefiere pasar más que tirar, tendrá un porcentaje de USG mucho menor que un jugador que tira mucho como Kobe Bryant.

```
data_players$Usage_rate <-
(data_players$x3A.x+data_players$x2A.x+0.44*data_players$FTA.x+data_players$To.x)*
  (data_players$Min.x/5)/(data_players$Min*(data_players$x3A.y+data_players$x2A.y+
                                            0.44*data_players$FTA.y+data_players$To.y))
```

4.3.7 Points Per Possession (PPP)

$$PT/(FGA + 0.44 * FTA + TO)$$

Donde:

- PT: Puntos

- FGA: Tiros de campo intentados.
- FTA: Tiros libres intentados.
- TO: Pérdidas.

Explicación

El numerador está claramente representado por la palabra "Puntos". Realmente no hay explicación necesaria allí.

Ya hemos visto cómo se pueden estimar mejor las posesiones por lo que está escrito en el denominador anterior. La explicación completa se proporciona en la diapositiva de porcentaje de pérdidas.

Esta estadística en su forma más simple explica cuán eficientemente un jugador usa su tiempo con la pelota para anotar. Con la ayuda de empresas como Synergy, el PPP puede desglosarse aún más en ciertas situaciones como situaciones de aclarado, situaciones de pick-and-roll, etc.

Limitaciones

Esta estadística solo se refiere a la eficiencia de puntuación y muchos aspectos importantes del baloncesto no se consideran.

Aparte de eso y el hecho de que las posesiones son estimaciones, no hay demasiadas limitaciones para PPP.

```
data_players$Points_per_possesion <-ifelse((data_players$x3A.x+data_players$x2A.x+
                                              0.44*data_players$FTA.x+data_players$To.x)==0,
                                            0,data_players$Pts.x/
                                              (data_players$x3A.x+data_players$x2A.x+
                                                 0.44*data_players$FTA.x+data_players$To.x))
```

Completamos los estadísticos con los encontrados en: https://www.fromtherumbleseat.com/pages/advanced-basketball-statistics-formula-sheet

4.3.8 Free Throw Rate

```
data_players$FTR_player <- ifelse((data_players$x2A.x+data_players$x3A.x)==0,
                                  0,data_players$FTM.x/
                                    (data_players$x2A.x+data_players$x3A.x))
```

4.3.9 Hollinger Assist Ratio (hAST%)

$$hAST\% = AST/(FGA + .44 * FTA + AST + TOV)$$

Miden a los jugadores idividualistas y relacionan lo que generan en relación a la vez que acaban la jugada. Esto divide la cantidad de asistencias que tiene un jugador por la cantidad de posesiones que terminan en las manos de ese jugador.

```
data_players$Hollinger_Assist_ratio <- ifelse((data_players$x2A.x+data_players$x3A.x+0.44*
                                                 data_players$FTA.x+data_players$As.x+
                                                 data_players$To.x)==0,0,
                                               data_players$As.x/
                                                 (data_players$x2A.x+data_players$x3A.x+
                                                    0.44*data_players$FTA.x+
                                                    data_players$As.x+
                                                    data_players$To.x))
```

4.3.10 Pomeroy Assist Ratio (pAST%)

$$pAST\% = AST/(((MP/(TmMP/5)) * TmFG) - FG)$$

Estimación del porcentaje de canastas de compañeros de equipo en las que asistió un jugador mientras estaba en la cancha.

```
data_players$Pomeroy_Assist_ratio <- data_players$As.x/
  (((data_players$Min / (data_players$Min.x / 5)) *
      (data_players$x3A.y+data_players$x2A.y) ) -
     (data_players$x3A.x+data_players$x2A.x))
```

4.3.11 Steal Percentage (STL%)

$$STL\% = STL/((MP/(TmMP/5)) * OppPoss)$$

Porcentaje de posesiones del oponente en las que un jugador obtiene un robo.

```
data_players$Steal_pct<-data_players$St.x/
  (data_players$Min/(data_players$Min.x/5)*(data_players$Poss))
```

4.3.12 Block Percentage (BLK%)

$$BLK\% = BLK/((MP/(TmMP/5)) * (OppFGA - Opp3PA))$$

265/5000 Porcentaje de tiros "taponables" de los oponentes que el jugador bloquea (elimina los tiros de 3 puntos, ya que estos generalmente no son "taponables", algo arbitrario ya que los ves bloqueados y es poco probable que un tiro de salto de medio alcance sea bloqueado, pero es incluido en la muestra)

```
data_players$Block_pct<-data_players$Fv.x/
  (data_players$Min/(data_players$Min.x/5)*(data_players$x2A_Opp))

data_players$ThreePointers_ratio <- ifelse((data_players$x3A.x+data_players$x2A.x)==0
                                           ,0,data_players$x3A.x/
                                             (data_players$x3A.x+
                                                data_players$x2A.x))
```

---

5 Análisis de los datos

5.1 Comprobación de la normalidad y homogeneidad de la varianza

Para la comprobación de que los valores que toman nuestras variables cuantitativas provienen de una población distribuida normalmente, utilizaremos la prueba de normalidad de *Anderson-Darling*.

Así, se comprueba que para que cada prueba se obtiene un p-valor superior al nivel de significación prefijado $\alpha = 0.05$. Si esto se cumple, entonces se considera que variable en cuestión sigue una distribución normal.

```
alpha = 0.05
col.names = colnames(data_team)

for (i in 1:ncol(data_team)) {
  if (i == 1) cat("Variables que no siguen una distribución normal:\n")
  if (is.integer(data_team[,i])| is.numeric(data_team[,i])) {
    p_val = ad.test(data_team[,i])$p.value
    if (p_val < alpha) {
      cat(col.names[i])

      # Format output
      if (i < ncol(data_team) - 1) cat(", ")
      if (i %% 3 == 0) cat("\n")
    }
  }
}
```

```
## Variables que no siguen una distribución normal:
## MatchId, Min,
## Pts, O, D,
## T, As, St,
## To, Fv, Ag,
## Cm, Rv, x2M, x2A, x3M,
## x3A, FTM, FTA,
## Local
```

A continuación, estudiamos la homogenidad de varianzas mediante la aplicaicón del test de *Fligner-Killeen*. En este caso, estudiaremos esta homogeneidad en cuanto a los equipos con victoria frente a los equipos con derrota. En el siguiente test, la hipótesis nula consiste en que ambas varianzas son iguales:

```
fligner.test(Pts ~ Win, data = data_teamopp)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Pts by Win
## Fligner-Killeen:med chi-squared = 0.24674, df = 1, p-value = 0.6194
```

Puesto que obtenemos un p-valor superior a 0,05, aceptamos la hipótesis de que las varianzas de ambas muestras son homogéneas.

5.2 Regresión lineal para estimar los ratings

Una de las primeras conclusiones a las que hemos llegado es que podemos darnos una idea muy buena de cuál será el rendimiento de un equipo en función del Rating Ofensivo y del Rating Defensivo.

Pero, ¿Cómo podemos estimar dichos Ratings en base a las variables recogidas en la estadística?

Uno de los primeros estadísticos dedicados al anáisis de los datos para el deporte fue el estadounidense Dean Olliver. Él decía que el rendimiento de un equipo depende de cuatro factores fundamentales anteriormente definidios: - eFG. - Offensive Rebounding Percentage. - Turnover Percentage. - Free Throw Rate

Proponemos, por tanto, estimar el rating ofensivo en base a los cuatro factores de ataque y el defensivo en función de los cuatro factores de defensa.

Para ello nos valdremos de una regresión lineal.

Rating ofensivo en función de los Four Factors de Dean Olliver

Comencemos pues por el rating ofensivo. El primer paso que daremos para resolver el problema, será el de calcular la correlación entre las variables explicativas.

```
# Rating de ataque
data_teamopp["Off_Reb_Opp"]<-1-data_teamopp["Def_Reb"]

data_ataque <- data_teamopp[c("Off_Rat","eFG","Off_Reb","Pct_To","FTR")]
corr_ataque <- round(cor(data_ataque),2)
corr_ataque
```

```
##         Off_Rat   eFG Off_Reb Pct_To   FTR
## Off_Rat    1.00  0.79    0.29  -0.45  0.18
## eFG        0.79  1.00   -0.05  -0.04 -0.05
## Off_Reb    0.29 -0.05    1.00   0.09  0.04
## Pct_To    -0.45 -0.04    0.09   1.00  0.00
## FTR        0.18 -0.05    0.04   0.00  1.00
```

Llegamos a una primer conclusión importante. Las variables, guardan relación lineal con la variable objetivo (especialmente eFG y porcentaje de pérdidas) pero son independientes entre sí.

Modelicemos pues, el rating ofensivo en función de los cuatro factores propuestos:

```
model_offesive_rating<- lm(Off_Rat~., data=data_ataque)
summary(model_offesive_rating)
```

```
##
## Call:
## lm(formula = Off_Rat ~ ., data = data_ataque)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.080251 -0.012458  0.001318  0.013292  0.057465
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.315316   0.008022   39.31   <2e-16 ***
## eFG          1.465926   0.010968  133.65   <2e-16 ***
## Off_Reb      0.645129   0.010561   61.09   <2e-16 ***
## Pct_To      -1.398715   0.018344  -76.25   <2e-16 ***
## FTR          0.333343   0.009688   34.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.01976 on 499 degrees of freedom
## Multiple R-squared:  0.9822, Adjusted R-squared:  0.982
## F-statistic:  6880 on 4 and 499 DF,  p-value: < 2.2e-16
```

La fórmula obtenida es por tanto: $Rating\_Ofensivo = 0.315316 + 1.465926eFG + 0.645129Off\_Reb - 1.398715Pct\_To + 0.333343FTR$

Analizando la tabla de coeficientes, vemos que todas las variables aportan valor.

Por último, su R^2 ajustado es de 98%. Es decir, nos encontramos ante un ajuste realmente bueno.

Repitamos el análisis para el Rating Defensivo.

Rating defensivo en función de los Four Factors de Dean Olliver

```
# Rating de defensa
data_defensa <- data_teamopp[c("Def_Rat","eFG_Opp","Off_Reb_Opp","Pct_To_opp","FTR_opp")]
corr_defensa <- round(cor(data_defensa),2)
corr_defensa
```

```
##             Def_Rat eFG_Opp Off_Reb_Opp Pct_To_opp FTR_opp
## Def_Rat        1.00    0.78        0.29      -0.44    0.13
## eFG_Opp        0.78    1.00       -0.05      -0.04   -0.05
## Off_Reb_Opp    0.29   -0.05        1.00       0.09    0.04
## Pct_To_opp    -0.44   -0.04        0.09       1.00   -0.02
## FTR_opp        0.13   -0.05        0.04      -0.02    1.00
```

Como era de suponer, las conclusiones obtenidas en el análisis de correlación son análogas a las del apartado anterior en cuando a pesos e independencia lineal de la variables.

Modelizamos ahora el rating defensivo:

```
model_defensive_rating<- lm(Def_Rat~., data=data_defensa)
summary(model_defensive_rating)
```

```
##
## Call:
## lm(formula = Def_Rat ~ ., data = data_defensa)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.112900 -0.024161  0.001754  0.026002  0.164258
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.32545    0.01555   20.93   <2e-16 ***
## eFG_Opp      1.46506    0.02122   69.04   <2e-16 ***
## Off_Reb_Opp  0.64297    0.02043   31.48   <2e-16 ***
## Pct_To_opp  -1.33526    0.03520  -37.94   <2e-16 ***
## FTR_opp      0.24327    0.01875   12.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03824 on 499 degrees of freedom
## Multiple R-squared:  0.9346, Adjusted R-squared:  0.9341
## F-statistic:  1783 on 4 and 499 DF,  p-value: < 2.2e-16
```

La fórmula obtenida es por tanto: $Rating\_Defensivo = 0.32545 + 1.46506eFG\_Opp + 0.64297Off\_Reb\_Opp -$

$1.33526 Pct\_To\_Opp + 0.24327 FTR\_Opp$

Analizando la tabla de coeficientes, vemos que todas las variables aportan valor.

Por último, su R^2 ajustado es de 93.4%. Es decir, algo peor que para el caso del rating ofensivo pero de nuevo un ajuste muy bueno.

En conclusión, podemos estimar el rating ofensivo y defensivo de un equipo en base a solo cuatro variables construidas desde su boxscore. Así, podremos estimar qué rating ofensivo podríamos esperar si logramos subir dos puntos el porcentaje de rebote por ejemplo o reducir el porcentaje de pérdidas.

5.3 Regresión logística para estimar victoria

Vamos a ser aún más ambiciosos. Hemos sido capaces de estimar cómo de bueno o malo será nuestro ataque pero, a la hora de ganar un partido, ¿Qué signfica eso?

El objetivo de este apartado es estimar la probabilidad de victoria en base a los ocho factores (los cuatro de ataque y los cuatro de defensa) a los que añadiremos la variable de si el equipo ha jugado en casa o no.

El primer paso por tanto es definir nuestro dataset:

```
datos_logit<-data_teamopp[c("eFG","Off_Reb","Pct_To","FTR","eFG_Opp","Off_Reb_Opp",
                            "Pct_To_opp","FTR_opp","Win","Local")]
```

Vamos a dividir nuestro dataframe en dos subconjuntos, el 70% para realizar el modelo y el 30% para validarlo.

```
set.seed(430)
default_idx = createDataPartition(datos_logit$Win, p = 0.7, list = FALSE)
training = datos_logit[default_idx, ]
testing = datos_logit[-default_idx, ]
```

Ahora estamos en condiciones de modelizar. Si anteriormente nos encontrábamos ante un problema de regresión, actualmente es un problema de clasificación cuyo clasificador es binario.

Podríamos aplicar diferentes técnicas de modelización. En este caso, optaremos por una regresión logística:

```
model_victoria <- glm(formula=Win ~ . , data=datos_logit, family=binomial(link="logit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_victoria)
```

```
##
## Call:
## glm(formula = Win ~ ., family = binomial(link = "logit"), data = datos_logit)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.15554  -0.03581   0.00000   0.03559   2.03533
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.5232     3.4395  -0.152  0.87909
## eFG          81.5654    11.5204   7.080 1.44e-12 ***
## Off_Reb      31.8897     5.5721   5.723 1.05e-08 ***
## Pct_To      -66.6958    10.9892  -6.069 1.29e-09 ***
## FTR          15.2153     3.5642   4.269 1.96e-05 ***
## eFG_Opp     -81.5839    11.4920  -7.099 1.25e-12 ***
## Off_Reb_Opp -31.9103     5.5841  -5.714 1.10e-08 ***
## Pct_To_opp   64.9885    10.7277   6.058 1.38e-09 ***
```

```
## FTR_opp     -15.3415     3.5466  -4.326 1.52e-05 ***
## Local         1.7459     0.6492   2.689  0.00716 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 698.69  on 503  degrees of freedom
## Residual deviance: 102.19  on 494  degrees of freedom
## AIC: 122.19
##
## Number of Fisher Scoring iterations: 9
```

De nuvo, analizando la tabla de coeficientes vemos que todas las variables son explicativas. Además, de que cada variable ofensiva con respecto a su homóloga defensiva, como es lógico, tiene peso similar y el signo cambiado.

Nos surge ahora la pregunta, ¿Cómo de bueno es nuestro modelo? Vamos a analizarlo:

```
prediccion_test <- format(round(predict(model_victoria,testing,type="response")))
confusionMatrix(as.factor(prediccion_test),as.factor(testing$Win))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 72  7
##          1  3 68
##
##                Accuracy : 0.9333
##                  95% CI : (0.8808, 0.9676)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8667
##
##  Mcnemar's Test P-Value : 0.3428
##
##             Sensitivity : 0.9600
##             Specificity : 0.9067
##          Pos Pred Value : 0.9114
##          Neg Pred Value : 0.9577
##              Prevalence : 0.5000
##          Detection Rate : 0.4800
##    Detection Prevalence : 0.5267
##       Balanced Accuracy : 0.9333
##
##        'Positive' Class : 0
##
```

El accuracy es del 93.3%. Es decir, de las 150 observaciones de test, el modelo es capaz de acertar 140, lo que son unos resultados muy buenos.

Otro estadístico importante cuando se analizan regresiones logísticas es el ROC, que nos da una idea de cómo de bien ordena un modelo. Es decir, cuantas veces nos equivocamos por cada acierto.

Calculemos su área bajo la curva .

```
rocobj <- roc( datos_logit$Win, predict(model_victoria, datos_logit),
               auc = TRUE, ci = TRUE )
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
print(rocobj)
```

```
##
## Call:
## roc.default(response = datos_logit$Win, predictor = predict(model_victoria,     datos_logit), auc = 
##
## Data: predict(model_victoria, datos_logit) in 252 controls (datos_logit$Win 0) < 252 cases (datos_log
## Area under the curve: 0.9937
## 95% CI: 0.9901-0.9973 (DeLong)
```

En este caso, el ROC es de 99.4%, es decir, de nuevo tenemos un modelo casi perfecto.

5.4 Clustering de jugadores

El último paso de este proyecto es ambicioso. Nos ponemos en la piel del director deportivo de uno de los equipos con un proyecto con recursos limitados.

Quiere hacer la mejor plantilla posible, con el mínimo gasto. Para ello, parte del mejor quinteto de la Euroliga y por cada jugador, quiere alternativas con nombres de jugadores que puedan tener un rendimiento similar pero menor precio (obviamente el mejor quinteto será por lo general el más caro).

Para ello, nos piden soporte. Planteamos para ello un proyecto que se dividirá en las siguientes fases:

- Depuración de datos.
- Definición de estadísticos para jugadores individuales.
- Búsqueda de jugadores de perfil similar.
- Presentación del informe.

Depuración de datos.

Comencemos depurando los datos. Para ello, eliminaremos del dataframe aquellos registros en los que el jugador en cuestión haya jugado menos minutos de los que consideramos relevantes, que lo delimitamos en un cuarto (salvo que el motivo sea por expulsión).

```
data_players_relevant <- data_players[data_players$Min>10 | data_players$Cm.x==5,]
```

Hemos reducido la muestra 1955 registros.

Buscaremos ahora eliminar aquellos jugadorse que no hayan jugado un número mínimo de partidos, ya que queremos un análisis estable y no que esté influenciado por un mal o buen rendimiento de un solo partido.

Seleccionamos aquellos jugadores que al menos hayan jugado un 20% de los partidos. Es decir, 6 partidos.

```
jugador_partido <- data.frame(table(data_players_relevant$PlayerName))
colnames(jugador_partido)[1]<-"PlayerName"
colnames(jugador_partido)[2]<-"Played_games"
jugador_partido <- jugador_partido[jugador_partido$Played_games>5,]

data_players_relevant<-merge(data_players_relevant, jugador_partido)
```

La definición de estadísticos se realizó en el apartado de enriquecimiento de datos, por lo que no es necesario volver a realizarlo. Ahora tenemos el dataframe con los candidatos:

```
data_players_cluster <- data_players_relevant[,c("PlayerName", "Assist_pct", "Turnover_pct",
                                                 "Offensive_Reb_pct", "eFG_player",
                                                 "TS_player", "Usage_rate",
                                                 "Points_per_possesion",
                                                 "ThreePointers_ratio", "Defensive_Reb_pct",
                                                 "FTR_player", "Hollinger_Assist_ratio",
                                                 "Pomeroy_Assist_ratio", "Steal_pct",
                                                 "Block_pct")]
```

Aplicamos ahora el algoritmo k-Means que nos dará, los jugadores que más se parezcan en función de los estadísticos seleccionados en el paso anterior y la distancia entre los mismos. Cada factor tendrá el mismo peso.

Preparamos pues los datos agrupándolos por juador para todos los partidos:

```
dt <- data.table(data_players_cluster)
acumulados_player<-dt[,list(Assist_pct=mean(Assist_pct),
Turnover_pct=mean(Turnover_pct),
Offensive_Reb_pct=mean(Offensive_Reb_pct),
eFG_player=mean(eFG_player),
TS_player=mean(TS_player),
Usage_rate=mean(Usage_rate),
Points_per_possesion=mean(Points_per_possesion),
ThreePointers_ratio=mean(ThreePointers_ratio),
Defensive_Reb_pct=mean(Defensive_Reb_pct),
FTR_player=mean(FTR_player),
Hollinger_Assist_ratio=mean(Hollinger_Assist_ratio),
Pomeroy_Assist_ratio=mean(Pomeroy_Assist_ratio),
Steal_pct=mean(Steal_pct),
Block_pct=mean(Block_pct)),by=PlayerName]
```

Aplicamos ahora el algoritmo k-means:

```
set.seed(121)
km.res <- kmeans(acumulados_player[,-1], 15)
acumulados_player$cluster<-km.res$cluster
```

Definimos el quinteto ideal y vamos a dar alternativas por puesto:

```
quinteto_ideal<-c("MICIC, VASILIJE","LARKIN, SHANE","DATOME, LUIGI",
                  "SHENGELIA, TORNIKE", "TAVARES, WALTER")
```

Calculamos los jugadores que comparten cluster con Micic. De ellos, escogeremos los que tengan los atributos en general más parecidos, o lo que es lo mismo, los que menos distancia entre su vector de atributos tengan.

```
i <-1
aux<-acumulados_player[acumulados_player$PlayerName==quinteto_ideal[i],]$cluster
distancias<-dist(acumulados_player[acumulados_player$cluster==aux,][,-1],upper = TRUE)
nombres_cluster<-acumulados_player[acumulados_player$cluster==aux,]$PlayerName
df <- melt(as.matrix(distancias), varnames = c("row", "col"))
df2<-df[df$row ==which(nombres_cluster==quinteto_ideal[i]),]
df2$nombre<-nombres_cluster
df2<- df2[order(df2$value),]
print("Recomendaciones Base")
```

```
## [1] "Recomendaciones Base"
```

```
print(head(df2$nombre[df2$row!=df2$col],5))
```

```
## [1] RODRIGUEZ, SERGIO DELANEY, MALCOLM  MALEDON, THEO     LO, MAODO
## [5] VAN ROSSOM, SAM
## 306 Levels: ABALDE, ALBERTO ABRINES, ALEX ABROMAITIS, TIM ... ZUBKOV, ANDREY
```

Como comprobamos, las recomendaciones son basante acertadas. Cinco bases, con un perfil relativamente anotador: - Sergio Rodríguez. - Malcolm Delaney. - Theo Maledon. - Maodo Lo - Sam Van Rossom.

Si tuviearmos un listado con posibles salarios, podríamos adaptar el problema a optimizar nuestro presupuesto.

Analicemos ahora las recomendacioens para el puesto de escolta:

```
i <-2
aux<-acumulados_player[acumulados_player$PlayerName==quinteto_ideal[i],]$cluster
distancias<-dist(acumulados_player[acumulados_player$cluster==aux,][,-1],upper = TRUE)
nombres_cluster<-acumulados_player[acumulados_player$cluster==aux,]$PlayerName
df <- melt(as.matrix(distancias), varnames = c("row", "col"))
df2<-df[df$row ==which(nombres_cluster==quinteto_ideal[i]),]
df2$nombre<-nombres_cluster
df2<- df2[order(df2$value),]
print("Recomendaciones Escolta")
```

```
## [1] "Recomendaciones Escolta"
```

```
print(head(df2$nombre[df2$row!=df2$col],5))
```

```
## [1] DIBARTOLOMEO, JOHN FREDETTE, JIMMER   NUNNALLY, JAMES    PAYNE, ADREIAN
## [5] MOTUM, BROCK
## 306 Levels: ABALDE, ALBERTO ABRINES, ALEX ABROMAITIS, TIM ... ZUBKOV, ANDREY
```

En este caso nos recomienda de nuevo a jugadores del perfil anotador de Larkin. John Dibartolomeo, immer Fredette, James Nunally y Rokas Giedriatis.. Sin embargo, nos llama la atención la aparición de Nikola Mirotic en esta lista, un jugador de perfil interior. Analizando en más detalle, vemos qeu sus datos en cuanto a importancia del lanzamiento exterior o creación de juego (estadísticos relacionados con asistencias) son más propios de un jugador exterior.

Queremos encontrar ahora candidatos al puesto de alero:

```
i <-3
aux<-acumulados_player[acumulados_player$PlayerName==quinteto_ideal[i],]$cluster
distancias<-dist(acumulados_player[acumulados_player$cluster==aux,][,-1],upper = TRUE)
nombres_cluster<-acumulados_player[acumulados_player$cluster==aux,]$PlayerName
df <- melt(as.matrix(distancias), varnames = c("row", "col"))
df2<-df[df$row ==which(nombres_cluster==quinteto_ideal[i]),]
df2$nombre<-nombres_cluster
df2<- df2[order(df2$value),]
print("Recomendaciones Alero")
```

```
## [1] "Recomendaciones Alero"
```

```
print(head(df2$nombre[df2$row!=df2$col],5))
```

```
## [1] VOIGTMANN, JOHANNES BAKER, RON          KOPONEN, PETTERI
## [4] CARROLL, JAYCEE     DOORNEKAMP, AARON
## 306 Levels: ABALDE, ALBERTO ABRINES, ALEX ABROMAITIS, TIM ... ZUBKOV, ANDREY
```

Como alternativas a Gigi Datome, nos presenta Ron Baker, Petteri Koponen, Jaycee Carroll, Aaron Doornekamp y Alex Abrines. Aleros con características de tirador.

Si analizamos ahora las alternativas al puesto de ala pívot, el algoritmo nos da:

```
i <-4
aux<-acumulados_player[acumulados_player$PlayerName==quinteto_ideal[i],]$cluster
distancias<-dist(acumulados_player[acumulados_player$cluster==aux,][,-1],upper = TRUE)
nombres_cluster<-acumulados_player[acumulados_player$cluster==aux,]$PlayerName
df <- melt(as.matrix(distancias), varnames = c("row", "col"))
df2<-df[df$row ==which(nombres_cluster==quinteto_ideal[i]),]
df2$nombre<-nombres_cluster
df2<- df2[order(df2$value),]
print("Recomendaciones Ala Pívot")
```

```
## [1] "Recomendaciones Ala Pívot"
```

```
print(head(df2$nombre[df2$row!=df2$col],5))
```

```
## [1] DUBLJEVIC, BOJAN    LEKAVICIUS, LUKAS   JEAN-CHARLES, LIVIO
## [4] HIGGINS, CORY       DIOP, ILIMANE
## 306 Levels: ABALDE, ALBERTO ABRINES, ALEX ABROMAITIS, TIM ... ZUBKOV, ANDREY
```

como opción a Livio Jean-Charles, una de las sopresas de la temporada de Asvel, Cory Higgins, Adrien Moerman, Konstantinos Mitoglou o Vladimir Micov.

Por último para alternativas a uno de los mejores cincos de la competición pero con un caché muy elevado (Walter Tavares), obtenemos:

```
i <-5
aux<-acumulados_player[acumulados_player$PlayerName==quinteto_ideal[i],]$cluster
distancias<-dist(acumulados_player[acumulados_player$cluster==aux,][,-1],upper = TRUE)
nombres_cluster<-acumulados_player[acumulados_player$cluster==aux,]$PlayerName
df <- melt(as.matrix(distancias), varnames = c("row", "col"))
df2<-df[df$row ==which(nombres_cluster==quinteto_ideal[i]),]
df2$nombre<-nombres_cluster
df2<- df2[order(df2$value),]
print("Recomendaciones Pívot")
```

```
## [1] "Recomendaciones Pívot"
```

```
print(head(df2$nombre[df2$row!=df2$col],5))
```

```
## [1] DUVERIOGLU, AHMET HINES, KYLE       STIMAC, VLADIMIR  VESELY, JAN
## [5] RUBIT, AUGUSTINE
## 306 Levels: ABALDE, ALBERTO ABRINES, ALEX ABROMAITIS, TIM ... ZUBKOV, ANDREY
```

a Ahmet Duveriouglu, Kyle Hines del CSKA, Colton Iverson de Zénit y Jan Vesely.

En conclusión, hemos obtenido unas cuantas alternativas que podríamos encontrar en el mercado a estos jugadores.

---

6 Exportación de datos finales

A continuación vamos a exportar nuestros dataframes finales a un archivo csv. Estos archivos se llamarán euroleague_scoreboards_clean.csv, euroleague_stats_per_game_clean.csv y euroleague_stats_per_team_clean.cs Utilizamos la función write.csv2() para exportar el fichero en formato csv español:

```
write.csv2(data_scoreboards, row.names = FALSE,
           file = "../csv/euroleague_scoreboards_clean.csv")

write.csv2(data_euroleague, row.names = FALSE,
           file = "../csv/euroleague_stats_per_game_clean.csv")

write.csv2(data_team, row.names = FALSE,
           file = "../csv/euroleague_stats_per_team_clean.csv")

write.csv2(data_teamopp, row.names = FALSE,
           file = "../csv/euroleague_team_and_opponent_clean.csv")
```

Estos nuevos datasets también estarán disponibles en el repositorio de GitHub mencionado en el primer apartado de este documento.

---

7 Representación de los resultados a partir de tablas y gráficas

7.1 Comparativa de equipos (Ataque vs. Defensa)

Para la primera representación gráfica vamos a realizar una comparativa entre equipos utilizando los estadísticos que se crearon anteriormente para definir la eficiencia en defensa y en ataque. Para ello, utilizaremos un diagrama del estilo del cuadrante Mágico de Gartner, en el que dividiremos a los equipos en cuatro cuadrantes.

El primer paso consistirá en crear una tabla en la que agrupemos a los equipos en base a la media de las variables `Off_Rat` y `Def_Rat` de todos los partidos:

```
data_team_ratings <- as.data.frame(data.table(data_teamopp)[, .(mean(Off_Rat*100),
                                                                mean(Def_Rat*100))
                                                            ,by = Team])
data_team_ratings
```

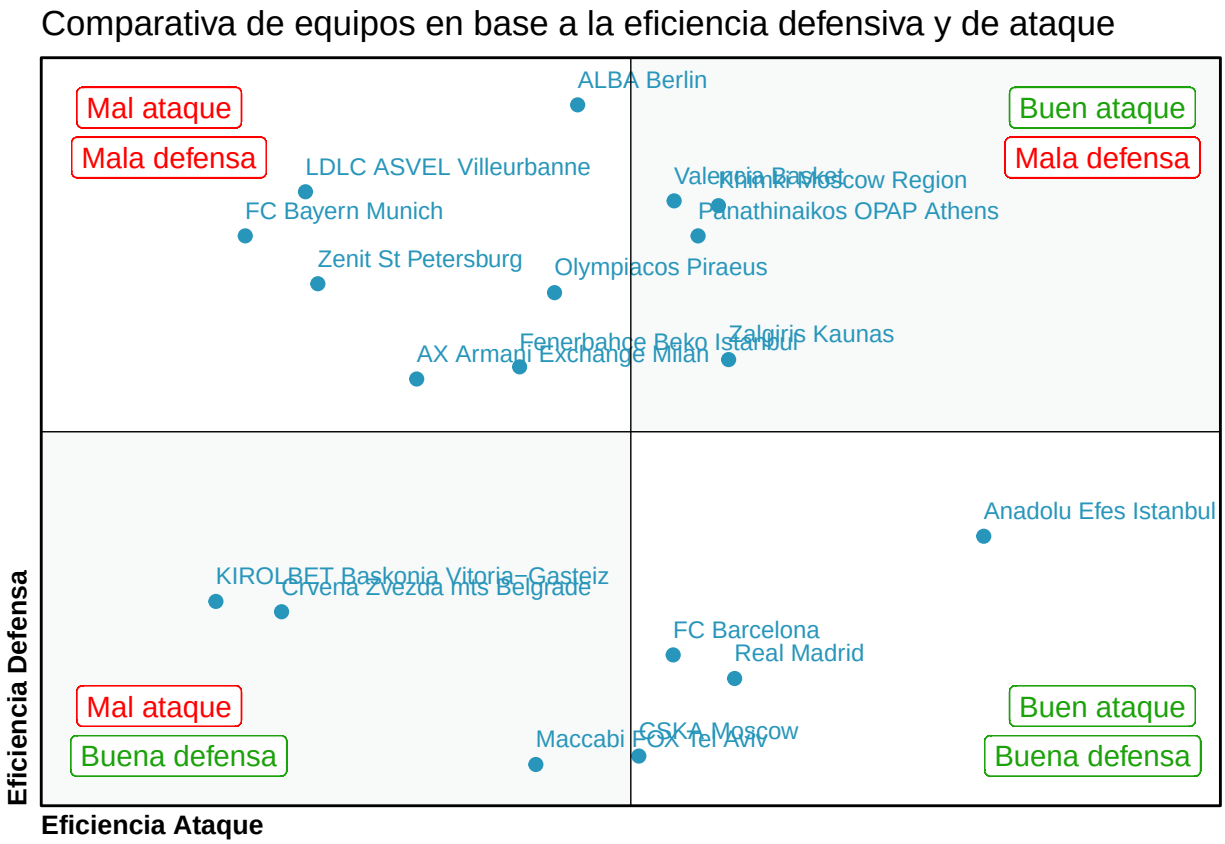
```
##                                 Team       V1       V2
## 1               Maccabi FOX Tel Aviv 112.5840 105.8275
## 2               Khimki Moscow Region 117.2305 117.0337
## 3                   FC Bayern Munich 105.1910 116.4281
## 4                       CSKA Moscow 115.2045 105.9969
## 5                    Zalgiris Kaunas 117.4884 113.9468
## 6         Crvena Zvezda mts Belgrade 106.1148 108.8904
## 7           AX Armani Exchange Milan 109.5526 113.5575
## 8                    Valencia Basket 116.1023 117.1300
## 9                       Real Madrid 117.6421 107.5525
## 10               Zenit St Petersburg 107.0377 115.4679
## 11           LDLC ASVEL Villeurbanne 106.7232 117.3123
## 12                      FC Barcelona 116.0781 108.0246
## 13          Fenerbahce Beko Istanbul 112.1715 113.8015
## 14                       ALBA Berlin 113.6479 119.0550
## 15         Panathinaikos OPAP Athens 116.7120 116.4275
## 16                Olympiacos Piraeus 113.0595 115.2889
## 17 KIROLBET Baskonia Vitoria-Gasteiz 104.4438 109.0975
## 18             Anadolu Efes Istanbul 123.9791 110.4048
```

Ahora que ya disponemos de los datos necesarios, procedemos a crear nuestro gráfico mediante la función `ggplot()`:

```
p <- ggplot(data_team_ratings, aes(V1, V2))

# Para facilitar la lectura de este informe, se han ocultado las distintas
# opciones de configuración del gráfico. No obstante, se pueden consultar en
# el fichero practica2.Rmd con el código fuente desde el que se ha generado
# este pdf.
```

```
gt = ggplot_gtable(ggplot_build(p))
gt$layout$clip[gt$layout$name=="panel"] = "off"
grid.draw(gt)
```



En la página web de la euroliga tenemos disponible la clasificación de los equipos hasta la jornada 28, que es la última jornada que se ha disputado hasta la fecha (31 de mayo de 2020). Se puede acceder a esta clasificación haciendo click aquí.

A modo de resumen, los primeros puestos son los siguientes:

Posición	Equipo	Victorias	Derrotas	Pts+	Pts-
1.	Anadolu Efes Istanbul	24	4	2432	2166
2.	Real Madrid	22	6	2371	2165
3.	FC Barcelona	22	6	2357	2193

Y los últimos puestos:

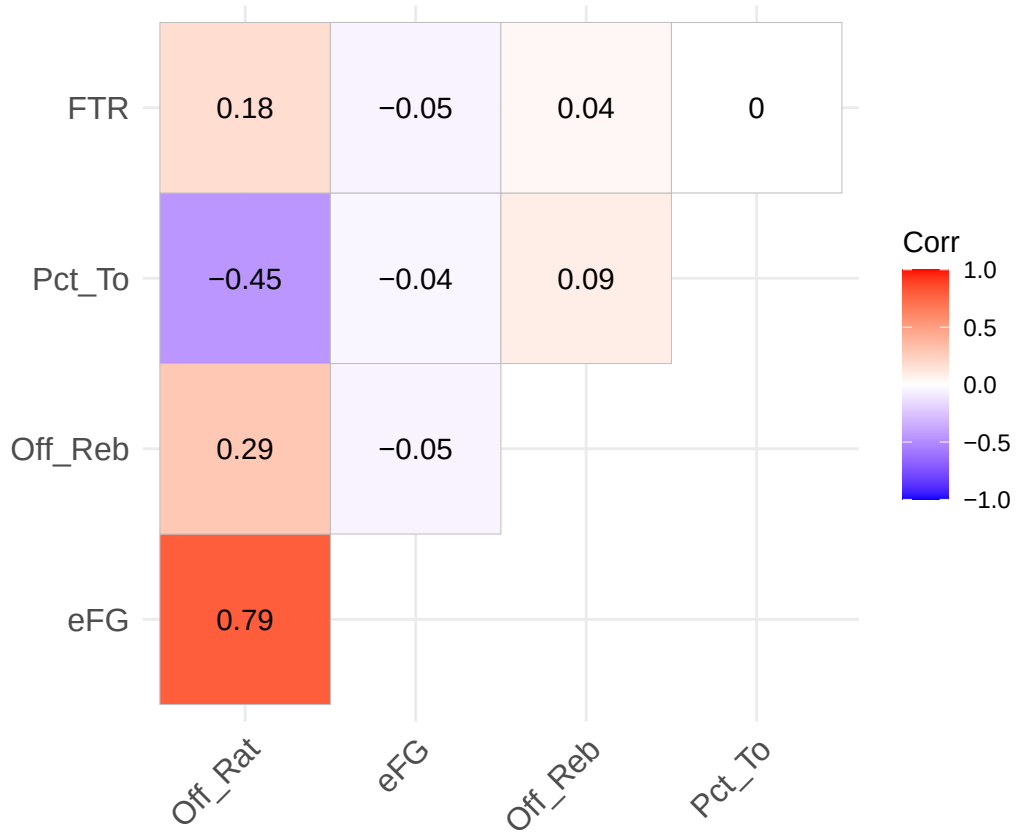
Posición	Equipo	Victorias	Derrotas	Pts+	Pts-
16.	ALBA Berlin	9	19	2304	2423
17.	FC Bayern Munich	8	20	2064	2281
18.	Zenit St Petersburg	8	20	2055	2240

Si comparamos la posición real en la clasificación con la posición de cada equipo en el gráfico vemos que queda completamente claro la relación entre las variables analizadas. Los equipos que están en los primeros puestos se caracterizan por un buen ataque y una buena defensa y los equipos que van últimos tienen mala defense y mal ataque.

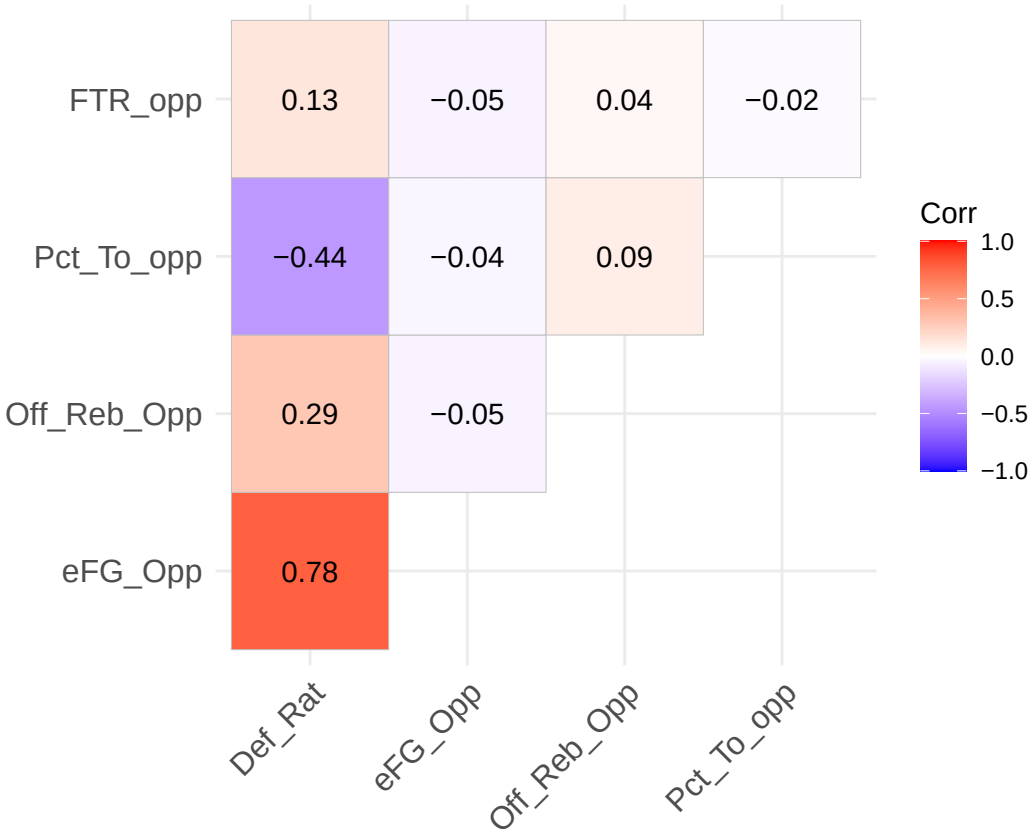
7.2 Correlaciones

Estos gráficos son complementarios a los presentados en la sección 5.2, de manera más visual. En probabilidad y estadística, la correlación indica la fuerza y la dirección de una relación lineal y proporcionalidad entre dos variables estadísticas. Se considera que dos variables cuantitativas están correlacionadas cuando los valores de una de ellas varían sistemáticamente con respecto a los valores homónimos de la otra: si tenemos dos variables (A y B) existe correlación entre ellas si al disminuir los valores de A lo hacen también los de B y viceversa. La correlación entre dos variables no implica, por sí misma, ninguna relación de causalidad.

```
ggcorrplot(corr_ataque, type = "upper", lab = TRUE)
```



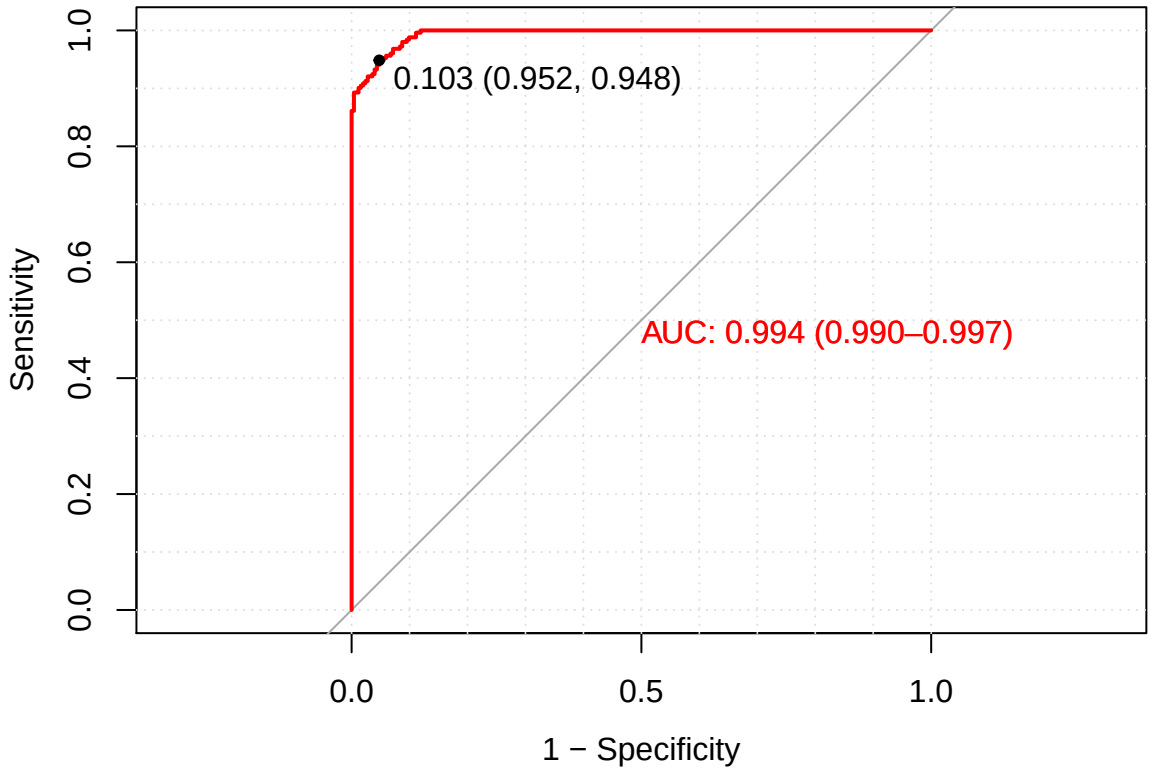

```
ggcorrplot(corr_defensa, type = "upper", lab = TRUE)
```


FTR_opp
Pct_To_opp
Off_Reb_Opp
eFG_Opp
0.13
−0.05
0.04
−0.02
−0.44
−0.04
0.09
0.29
−0.05
0.78
Def_Rat
eFG_Opp
Off_Reb_Opp
Pct_To_opp
Corr
1.0
0.5
0.0
−0.5
−1.0

7.3 Curva ROC

La curva ROC es la representación de la razón o ratio de verdaderos positivos (VPR = Razón de Verdaderos Positivos) frente a la razón o ratio de falsos positivos (FPR = Razón de Falsos Positivos) también según se varía el umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo). La idea de la curva ROC es que tenga la mayor área bajo la curva posible y, en general, nos interesa que la pendiente al principio sea muy elevada para disparar el área de captura de verdaderos positivos en relación a los errores cuanto antes.

```
plot.roc( rocobj, legacy.axes = TRUE, print.thres = "best",
          print.auc = TRUE, auc.polygon = FALSE, max.auc.polygon = FALSE,
          auc.polygon.col="gainsboro", col = 2, grid = TRUE )
```



****

8 Conclusión

Como conclusión al trabajo, queríamos destacar: - El Data Science tiene infinidad de campos a los que contribuir con su crecimiento y el deporte profesional es uno de ellos, donde además, su uso por lo general apenas está explotado. - La creación de nuevos estadísticos que depuren los datos y mejoren la percepción de los mismos, es uno de los puntos donde se está poniendo el foco. - Uno de los mayores problemas a la hora de analizar datos del mundo profesional es la mala calidad de las bases de datos. Somos consciente de que nuestra aproximación es una demostración de que la Ciencia de Datos puede servir como apoyo al baloncesto pero que no es realista desde el putno de vista de interpretación. Para ser más precisos es necesario bases de datos de calidad mayor, en los que se analice con más detalle los jugadores (veces que botan por derecha o izquierda), los quintetos con los que juegan (no es lo mismo jugar con un buen base a la hora de obtener tiros que con un base flojo), los rivales o el mometno en el que se toman las estad´siticas (es obvio qeu meter una canasta en los minutos finales o clutch es mucho más valorado que los jugadores cuyo mejor performance es al principio o en partidos resueltos). - Hay muchas más métricas de ataque que de defensa, por lo que es dificil darle el mismo valor a ambos a la hora de realizar análisis estadísticos. - Las aplicaciones de la Ciencia de datos al baloncesto son infinitas y de distintas aplicaciones, desde el entrenador que realiza el scouting y plantea el partido, el que analiza el resultado y los fallos que se han tenido o incluso la hora de que el director técnico confeccione el equipo. - Entendemos la ciencia sobre los datos como un complemento necesario a la hora de tomar decisiones en el día a día de deporte, pero no como un sustitutivo de las personas. Al final, siempre que se trabaja con personas hay muchos datos que son imposibles de medir y que podrían contribuir a la hora, por ejemplo, de crear tu plantilla en pretemporada. - La dinámica obviamente está cambiando y el ejemplo de la NBA es el más inmediato, donde todas las franquicias tienen un equipo de más de treinta personas trabjaando en Big Data & Advanced Analytics dando soporte a las decisiones.

---

9 Contribuciones al trabajo

Contribuciones	**Firma**
Selección del dataset	MSP, AAG
Creación del repositorio GitHub	MSP, AAG
Desarrollo código en R	MSP, AAG
Redacción de las respuestas	MSP, AAG