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

```r
 # Cargamos las librerías necesarias para el proyecto
suppressMessages(library(dplyr))
suppressMessages(library(plyr))
suppressMessages(library(data.table))
suppressMessages(library(ggcorrplot))
suppressMessages(library(pROC))
suppressMessages(library(factoextra))
```

```
## Warning: package 'factoextra' was built under R version 3.6.2
```

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

```
## Warning: package 'reshape2' was built under R version 3.6.2
```

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
```

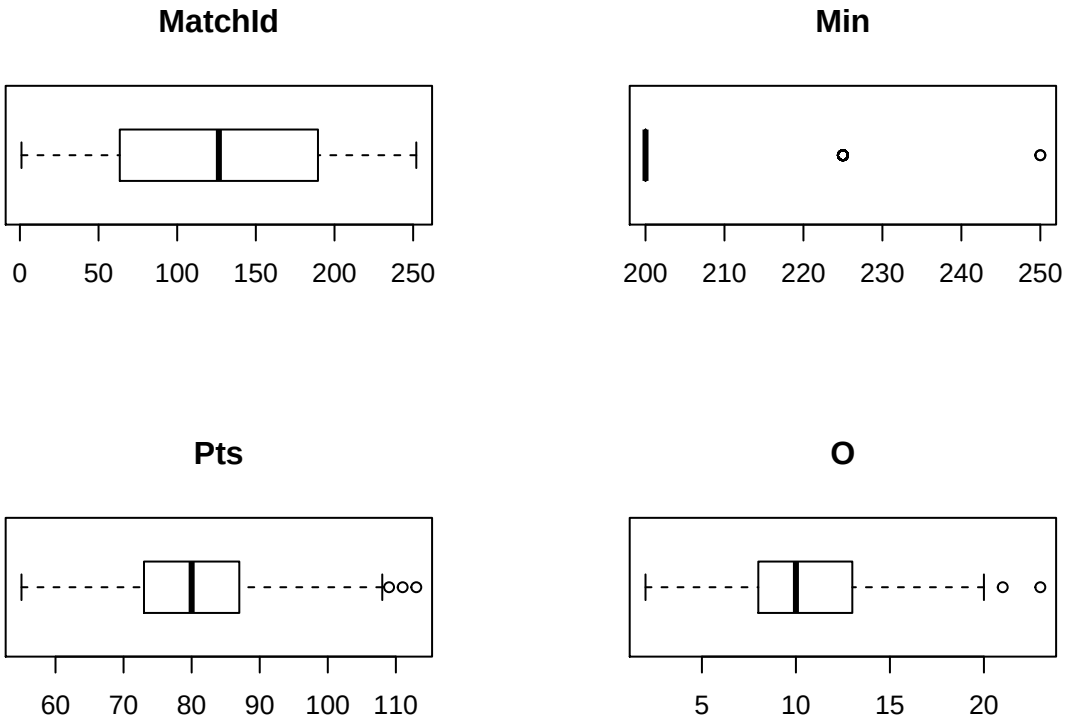
Para los partidos con prórroga se deben sumar 5 minutos por cada parte de tiempo extra y multiplicar por los 5 jugadores. Por eso vemos partidos con 225 minutos.

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. Generalmente se considera que cuando un valor se encuentra alejado 3 desviaciones estándar con respecto a la media del conjunto es un *outlier*. Para encontrar estos valores extremos se utiliza los diagramas de cajas o *boxplot*, que permiten identificar de manera muy sencilla si existen valores extremos para una determinada variable.

Para detectar los valores extremos de nuestro dataset de estadísticas de equipo, vamos a crear un diagrama de caja por cada variable cuantitativa:

```
par(mfrow = c(2,2))
for (i in c(seq(2,5))) {
  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
## [20] 225 225 225 250 225 225 225 225 225 225 225 225 225 225 225
```

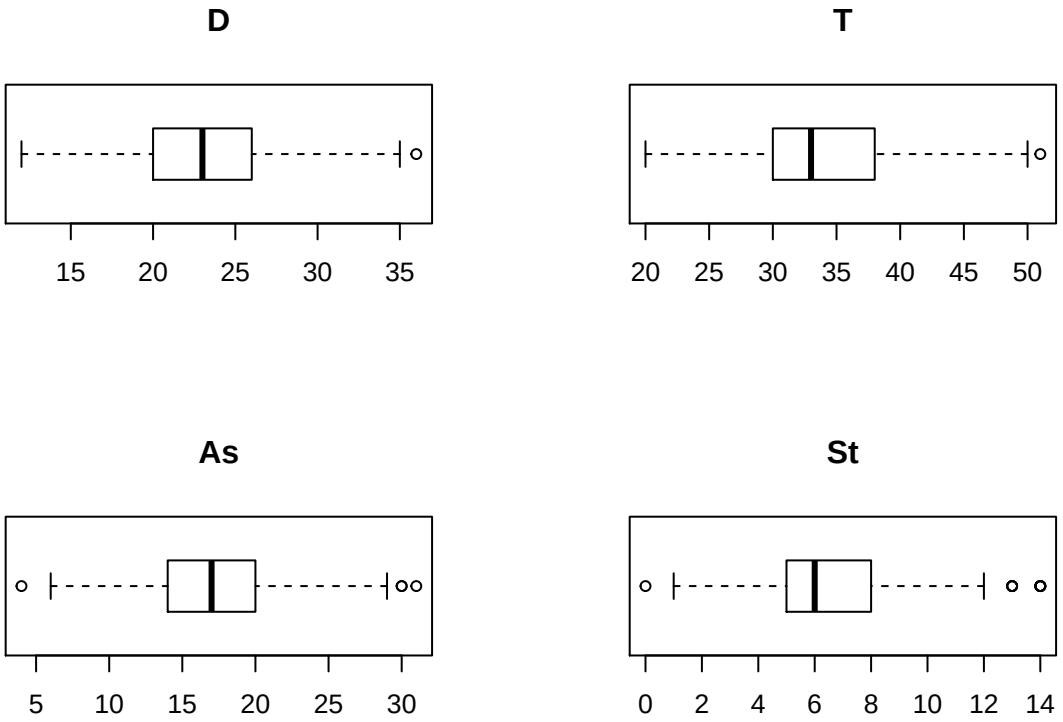
```
# Valores extremos de la variable Pts
outlier.4$out
```

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

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

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

```
par(mfrow = c(2,2))
for (i in c(seq(6,9))) {
  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 D
outlier.6$out
```

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

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

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

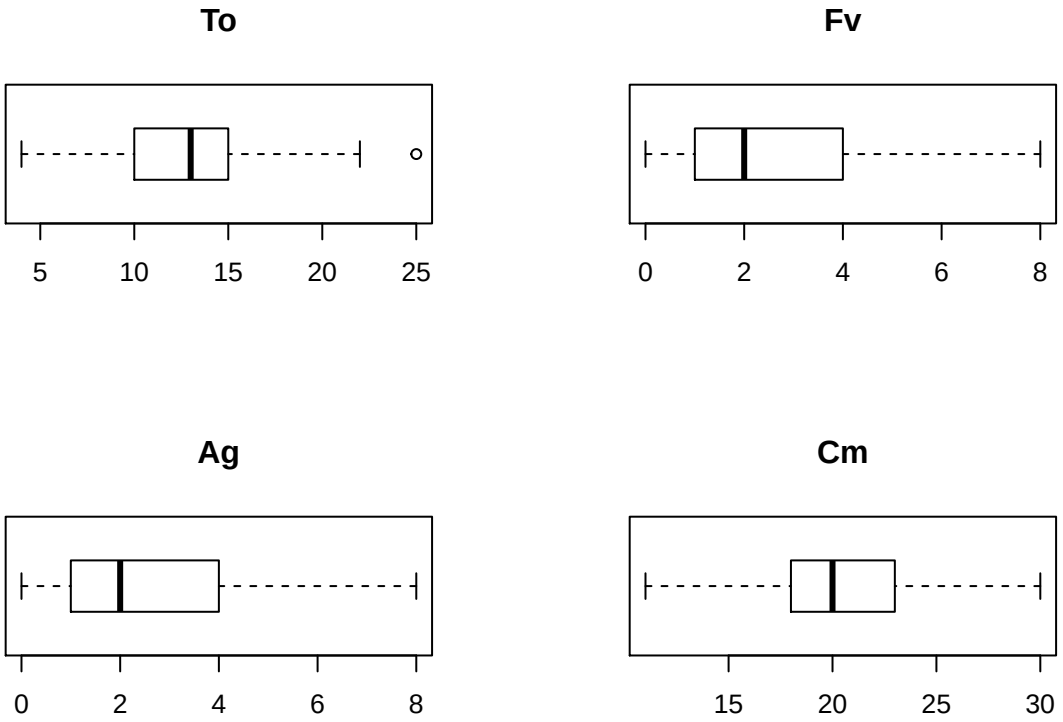
```
# Valores extremos de la variable As
outlier.8$out
```

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

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

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

```r
par(mfrow = c(2,2))
for (i in c(seq(10,13))) {
  outlier <- boxplot(data_team[,i], main = colnames(data_team)[i], horizontal = TRUE)
  assign(paste("outlier", i, sep = "."), outlier)
}
```

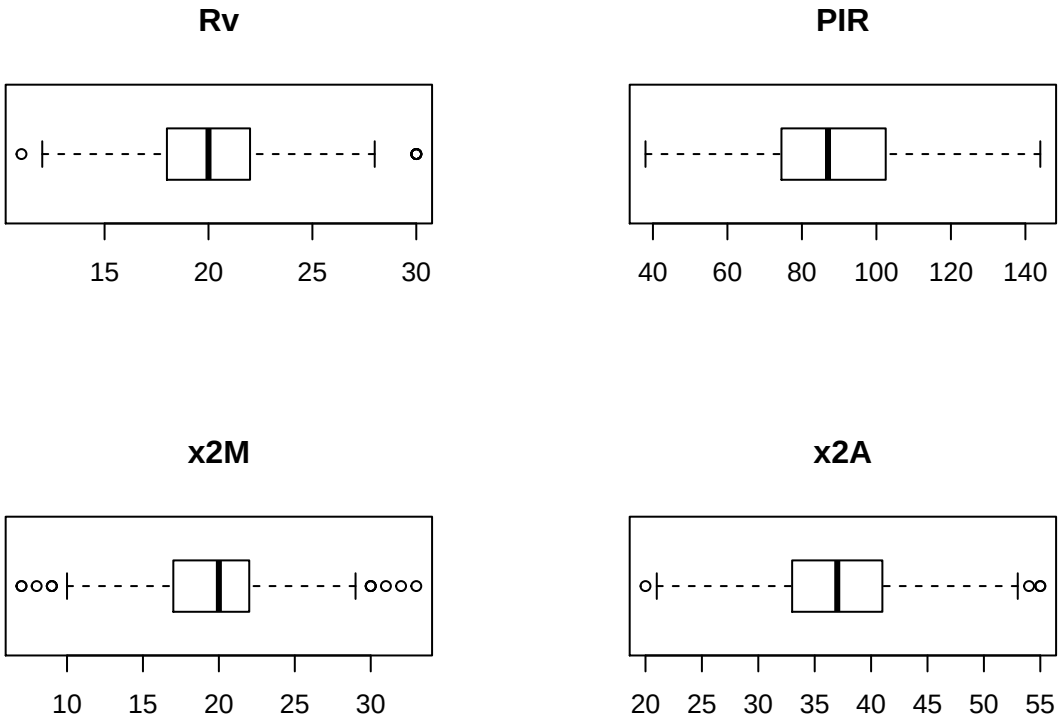



```r
par(mfrow = c(1,1))
```

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

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

```r
par(mfrow = c(2,2))
for (i in c(seq(14,17))) {
  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
```

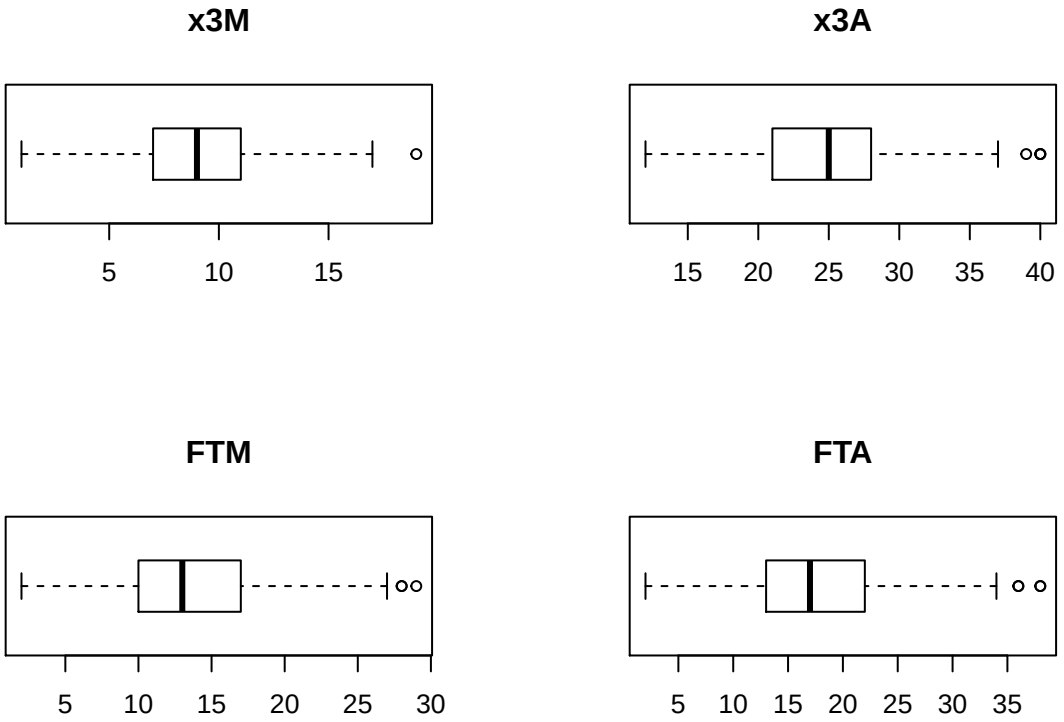
```r
# Valores extremos de la variable x2M
outlier.16$out
```

```
##  [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
par(mfrow = c(2,2))
for (i in c(seq(18,21))) {
  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 x3M
outlier.18$out
```

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

```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*:** `MatchId`, `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

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)
```

---

5 Análisis de los datos

5.1 Comprobación de la normalidad y homogeneidad de la varianza

FALTA POR HACER

5.2 Regresión lineal para estimar los ratings

```
# 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)

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
```

```r
# 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)

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
```

5.3 Regresión logística para estimar victoria

Buscamos estimar la probabibilidad de victoria en función de los estadísticos creados.

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

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

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

```
summary(model_victoria)
```

```
##
## Call:
## glm(formula = Win ~ ., family = binomial, 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
```

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 performance 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..

Comencemos depurando los datos. Para ello, eliminaremos del registro aquellos registros en los que el jugador en cuestión haya jugado menos minutos de los que consideramos relevantes.

Para comenzar, eliminamos los registros de aquellos jugadores en un partido en el que han jugado menos de un cuarto (salvo que el motivo sea por expulsión).

```
n_registros <- nrow(data_euroleague)
summary(data_euroleague[c("Min")])
```

```
##       Min
##  Min.   : 0.000
##  1st Qu.: 7.117
##  Median :16.800
##  Mean   :15.634
##  3rd Qu.:23.550
##  Max.   :48.883
```

```
data_players <- data_euroleague[data_euroleague$Min>10 | data_euroleague$Cm==5,]
n_registros2 <- nrow(data_players)
n_registros - n_registros2
```

```
## [1] 1955
```

Hemos reducido la muestra 1955 registros.

El siguiente paso será calcular el número total de partidos jugados por jugador.

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

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

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

Ahora tenemos el dataframe con los candidatos.

El siguiente paso sería realizar un research para definir métricas avanzadas que nos ayuden a comparar jugadores.

Nos basaremos en la siguiente web para definir los estadísticos:

https://bleacherreport.com/articles/1039116-understanding-the-nba-explaining-advanced-offensive-stats-and-metrics

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_players2<-merge(data_players, data_teamopp, by=c("MatchId","Team"))
```

1. Assist Percentage (AST%) A/(((MP/(TMP/5))*TFG)-FG)

Where A=Assists, MP=Minutes Played, TMP=Team Minutes Played, TFG=Team Field Goals, FG=Field Goals

Explanation

Leading off this guide to advanced offensive metrics in the NBA is perhaps the most basic of the stats: assist percentage. I say perhaps "basic" because it's one of many tempo-free stats, or stats adjusted for pace and volume.

While assists in themselves are kind of useful to look at, assist percentage is a much better statistic to cite when trying to make a case for a player's skill in the passing department. Assist stats can be padded by two things: the amount of time that a player is on the court and the pace at which his team plays.

A player on a fast-paced team like the Miami Heat or Washington Wizards is going to have more offensive opportunities to rack up counting stats, leading to elevated numbers of assists per game and assists per minute (which are also a better way of looking at assists than just assists by themselves).

Similarly, a player who plays 35 minutes per game is going to have more opportunities to generate assists than a player on the court for 20 minutes per game. It may seem like common sense that averaging 10 assists per game in 20 minutes of action per contest is more impressive than averaging 10 assists per game in 35 minutes of action per contest, but that distinction is lost when only assists per game is cited.

Because assist percentage is free from the effects of pace and volume, it's a better indication of how effective a player is at racking up the dimes during each and every one of the team's possessions.

The stat essentially estimates (note: estimates, not calculates) what percentage of made shots by teammates were assisted by a player while he was on the floor.

Limitations

Without looking at play-by-play data, the best this stat can do is provide an estimate of the aforementioned percentage it is meant to calculate. Additionally, there is no way to measure and give credit for good passes that resulted in missed open looks.

Just as with any assist statistic, a player is at the mercy of the skill of his teammates.

```
data_players2$Assist_pct <- data_players2$As.x/(((data_players2$Min/(data_players2$Min.x/5))*(data_play
```

Turnover Percentage (TOV%)

Calculation

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

Where TO=Turnovers, FGA=Field Goals Attempted, FTA=Free Throws Attempted

Explanation

The second of the offensive tempo-free stats, turnover percentage, is free from the effects of tempo because it isolates the possessions in which the player in question made a box score impact.

Before explaining exactly what this stat does though, I'd like to focus on the parenthetical denominator of the above calculation: (FGA+0.44*FTA+TO). This mathematical expression is the best way of quantifying the number of play results a player was involved in without simply going back through every box score and actually counting.

There are three ways that a player can be involved in the end result of a possession. They can attempt a field goal (regardless of whether it's a two-pointer or a three-pointer), they can end up on the foul line or

they can turn the ball over. However, simply summing those three results does not provide the number of possessions because shooters can attempt either one, two or three free throws on any given possession.

Box scores don't explain how many shots a player was fouled on, so we have no idea of knowing which fouls resulted in and-ones (for example) without looking through historical play-by-play data.

Just trust on this one (I've read the studies and they're too complicated to explain in a short space) and accept the fact that the 0.44 multiplier is the best way of estimating the total number of possessions a player is involved in.

So now that we've got that out of the way, all this stat really does is calculate the number of turnovers a player will make in 100 individual plays.

Turnovers and turnovers per game are both dependent once more on pace of play and the amount of time a player spends on the court. This rate statistic eliminates those detriments and focuses solely on the percentage of times a player turns the ball over compared to the amount of times they're involved in the play.

Limitations

Turnover percentage still can't factor in the passes that a player makes that result in non-turnovers (i.e. assists or passes to other players who either miss a shot or don't shoot).

Therefore, it's still a fairly limited stat because it only focuses on the true outcomes of possessions when that player is involved.

```
data_players2$Turnover_pct<-ifelse((data_players2$x2A.x+data_players2$x3A.x+0.44*data_players2$FTA.x+da
```

Offensive Rebound Percentage (ORB%)

Calculation

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

Where ORB=Offensive Rebounds, TMP=Team Minutes Played, MP=Minutes Played, TORB=Team Offensive Rebounds, ODRB=Opponents Defensive Rebounds

Explanation

Whenever a shot clanks off the rim, there are four possible outcomes: The ball could go out of bounds and be counted as a defensive team rebound; the ball could bounce of a defensive player and go out of bounds to be counted as an offensive team rebound; the ball could be pulled down by a defensive player and be counted as a defensive rebound; or the ball could be pulled down by an offensive player and be counted as an offensive rebound.

If you add up all four of those results, you account for all of the potential rebounds in a game.

Offensive rebound percentage calculates the percentage of available offensive rebounds that a player grabs while he's on the court.

This is the last of the commonly-used offensive tempo-free rate stats and much like the previous two, it doesn't account for pace or volume.

Limitations

Once more, it's too much trouble to go back and retroactively look at all historical box scores. No one really wants to do that.

Therefore, this is merely an estimate, albeit an accurate one.

De manera análoga, definimos el porcentaje de rebotes defensivos.

```
data_players2$Offensive_Reb_pct <- data_players2$O.x*(data_players2$Min.x/5)/(data_players2$Min*(data_pl

data_players2$Defensive_Reb_pct <- data_players2$D.x*(data_players2$Min.x/5)/(data_players2$Min*(data_pl
```

Effective Field-Goal Percentage (eFG%)

Calculation

(FG+0.5*3P)/FGA

Where FG=Field Goals, 3P=Three-Pointers, FGA=Field-Goal Attempts

Explanation

Field-goal percentage (FG%) was once one of the better stats for estimating shooting ability up until the popularization of effective field-goal percentage in the 1990s.

The only difference between the two stats is that three-pointers are weighted more heavily in eFG%. Seeing as three-pointers are worth three points and two-pointers are worth two points, this makes sense.

That may be the most ridiculously obvious sentence I've ever written.

Don't make the mistake of thinking that the multiplier in front of three-pointers should be 1.5 though. Essentially, it is in the above formula because three-pointers are counted one time in field goals and another 0.5 times on the other side of the addition sign.

So, as for the merits of effective field-goal percentage, tell me which of the following players you'd rather have:

Player A attempts 10 shots from the field, all from within the three-point arc, and drills five of them.

Player B attempts 10 shots from the field, all from outside the three-point arc, and drills five of them.

Player A, who was responsible for 10 points, has a FG% of 50 percent, just like Player B, who was responsible for 15 points, despite shooting the same number of times. However, Player A's eFG% was still 50 percent while Player B's was a much better 75 percent.

Limitations

While I like eFG% as a stat, there is still a measure of shooting that is significantly better. If for nothing else, that's because it takes free-throw shooting into account as well, unlike eFG%.

You'll see what that stat is pretty soon.

Also, the penalty for missing a three-pointer is the same as the penalty for missing a two-pointer because all attempts are weighted the same.

```
data_players2$eFG_player<-ifelse((data_players2$x3A.x+data_players2$x2A.x)==0,0,(data_players2$x2M.x+1.5
```

True Shooting Percentage (TS%)

Calculation

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

Where PT=Points, FGA=Field-Goal Attempts, FTA=Free-Throw Attempts

Explanation

There are three ways that an NBA player can score: three-pointers, two-pointers and free throws. It just makes sense that the best measure of shooting percentage would take all three of those methods of scoring into account.

So, does true shooting percentage look at all three?

As you can see by the "Field-Goal Attempts" and "Free-Throw Attempts" in the calculation, TS% clearly at least takes two-pointers and free throws into account. As for the 0.44 multiplier, the same reasoning applies as did for turnover percentage. You can find the full explanation on that slide.

Three-pointers are a little harder to find in the formula, but they're still there, hidden within the word "Points." The maximum TS% is actually 150 percent and can only be achieved if a player hits each and every

one of his shots and they're all from downtown. For example, if a player goes 1-for-1 and his only shot is a three-pointer, the formula will read and simplify as follows: $3/(21+0.440)) = 3/2 = 1.5$.

Because this stat does take everything into account, it's easily the best measure of shooting ability we have. Just for the record, it can also be called adjusted shooting percentage, effective shooting percentage, effective percentage, points per shot attempted and scoring efficiency.

Limitations

Other than only accounting for shooting ability, and thereby not qualifying as an overall evaluation of a player's offensive ability, the only true limitation of TS% is that it's possible (although extremely unlikely) to have a TS% of over 100 percent.

Also, just like with effective field-goal percentage, all missed attempts count the same.

```
data_players2$TS_player <- ifelse((2*(data_players2$x3A.x+data_players2$x2A.x+0.44*data_players2$FTA.x)
```

Usage Rate (USG%)

Calculation

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

Where FGA=Field-Goal Attempts, FTA=Free-Throw Attempts, TO=Turnovers, TMP=Team Minutes Played, MP=Minutes Played, TFGA=Team Field-Goal Attempts, TFTA=Team Free-Throw Attempts, TTO=Team Turnovers

Explanation

This may be the most complicated looking calculation yet, but the concept behind it is really quite simple. Usage rate calculates what percentage of team plays a player was involved in while he was on the floor, provided that the play ends in one of the three true results: field-goal attempt, free-throw attempt or turnover.

On average, a player will have a usage rate of 20 percent. Think about it and it will make perfect sense.

Limitations

Only the true outcomes are measured here, so there is quite a bit left out. For example, a player like Ricky Rubio, who prefers to pass more than shoot will have a much lower USG% than a player who prefers to shoot.

Yet, a player who passes the ball is still involved in a possession in my mind.

```
data_players2$Usage_rate <-
(data_players2$x3A.x+data_players2$x2A.x+0.44*data_players2$FTA.x+data_players2$To.x)*(data_players2$Mi
```

Points Per Possession (PPP)

Calculation

PT/(FGA+0.44*FTA+TO)

Where PT=Points, FGA=Field-Goal Attempts, FTA=Free-Throw Attempts, TO=Turnovers

Explanation

The numerator is clearly represented by the word "Points." There is really no explanation necessary there.

Per refers to the division sign.

We've already seen how possessions can be best estimated by what's written in the above denominator. The full explanation is provided on the turnover percentage slide.

This stat in its simplest form explains how efficiently a player uses his time with the ball to score. With the help of companies like Synergy, PPP can be further broken down into certain situations like isolation plays, pick-and-roll situations, etc.

Limitations

This stat only refers to scoring efficiency and there's more than scoring to basketball, even on offense.

Other than that and the fact that the possessions are estimates, there aren't too many limitations to PPP.

```
data_players2$Points_per_possesion <-ifelse((data_players2$x3A.x+data_players2$x2A.x+0.44*data_players2
```

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

```
data_players2$FTR_player <- ifelse((data_players2$x2A.x+data_players2$x3A.x)==0,0,data_players2$FTM.x/(
```

```
Hollinger Assist Ratio (hAST%) = AST / (FGA + .44 * FTA + AST + TOV)
```

Think of Carmelo Anthony for this statistic, it is a "ball stopper" stat. This divides the number of assists a player has by the number of offensive possessions that end in that player's hands.

```
data_players2$Hollinger_Assist_ratio <- ifelse((data_players2$x2A.x+data_players2$x3A.x+0.44*data_playe
```

```
Pomeroy Assist Ratio (pAST%) = AST / (((MP / (TmMP / 5)) * TmFG ) - FG)
```

The percentage of teammate baskets a player assisted on while he was on the court - my preferred Assists statistic

```
data_players2$Pomeroy_Assist_ratio <- data_players2$As.x/(((data_players2$Min / (data_players2$Min.x / 
```

```
Steal Percentage (STL%) = STL / ((MP / (TmMP / 5)) * OppPoss)
```

Percent of opponent possessions in which a player gets a steal

```
data_players2$Steal_pct<-data_players2$St.x/(data_players2$Min/(data_players2$Min.x/5)*(data_players2$P
```

```
Block Percentage (BLK%) = BLK / ((MP / (TmMP / 5 )) * (OppFGA - Opp3PA))
```

Percent of opponents' "blockable" shots that the player blocks (removes 3 point shots, as these are generally not "blockable", somewhat arbitrary since you see these blocked and a mid range jump shot is just as unlikely to be blocked, yet is included in the sample)

```
data_players2$Block_pct<-data_players2$Fv.x/(data_players2$Min/(data_players2$Min.x/5)*(data_players2$x
```

```
data_players2$ThreePointers_ratio <- ifelse((data_players2$x3A.x+data_players2$x2A.x)==0,0,data_players
```

```
data_players_cluster <- data_players2[,c("PlayerName","Assist_pct","Turnover_pct","Offensive_Reb_pct","
```

Aplicamos ahora el algoritmo kNN.

```
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]
```

```r
set.seed(155)
km.res <- kmeans(acumulados_player[,-1], 18)
acumulados_player$cluster<-km.res$cluster

quinteto_ideal<-c("MICIC, VASILIJE","LARKIN, SHANE","DATOME, LUIGI","MIROTIC, NIKOLA","TAVARES, WALTER")

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"
```

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

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

```r
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"
```

```r
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
```

```r
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"
```

```r
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
```

```
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] PAYNE, ADREIAN    GIEDRAITIS, ROKAS JEREBKO, JONAS    FREDETTE, JIMMER
## [5] GILL, ANTHONY
## 306 Levels: ABALDE, ALBERTO ABRINES, ALEX ABROMAITIS, TIM ... ZUBKOV, ANDREY
```

```
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
```

---

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")
```

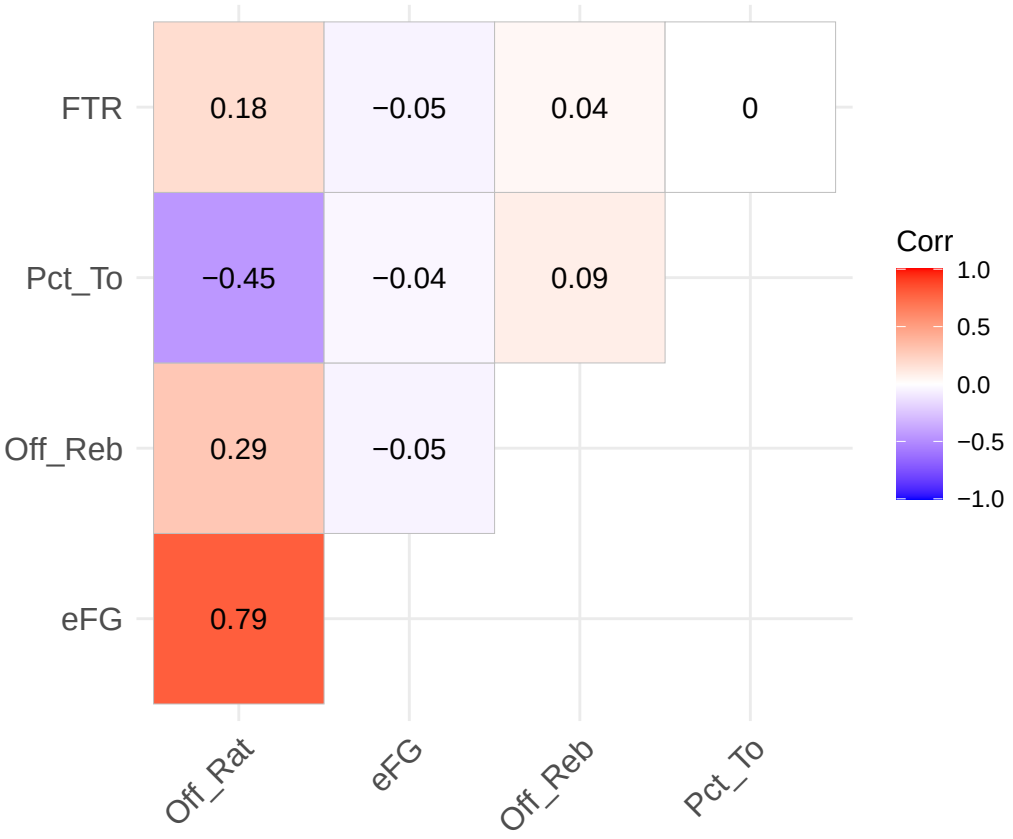
Estos nuevos datasets también estarán disponibles en el repositorio de GitHub mencionado en el primer apartado de este documento.

---

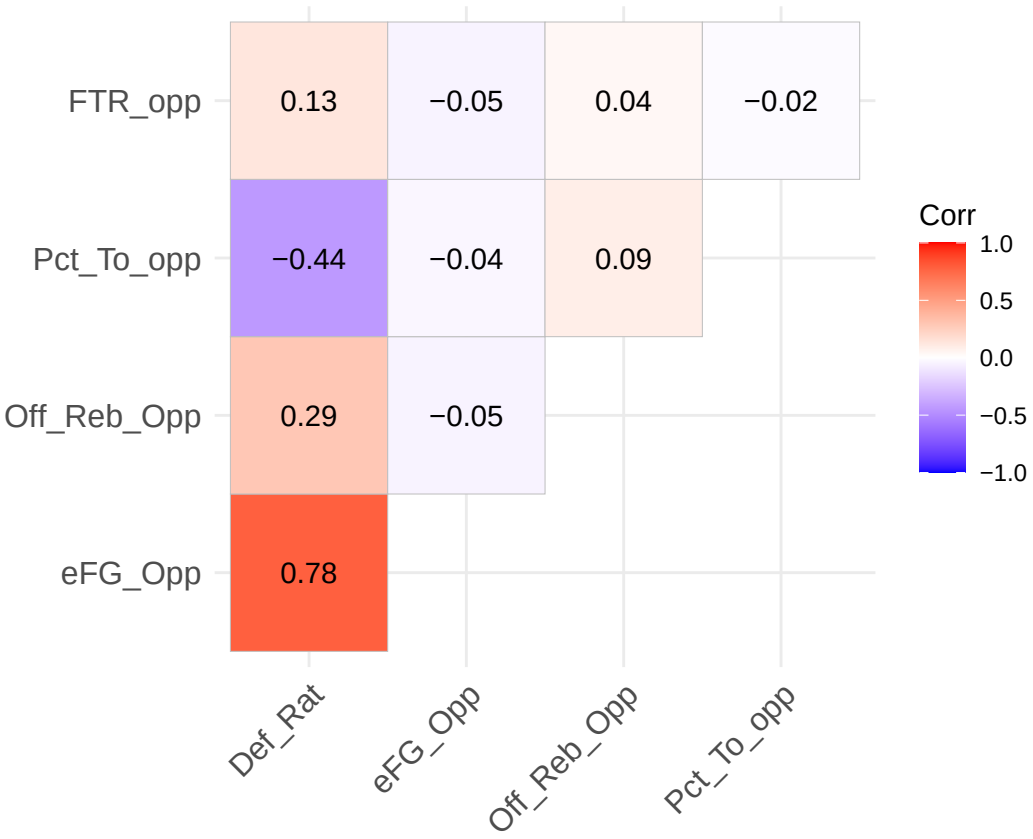
7 Representación de los resultados a partir de tablas y gráficas

FALTA POR HACER

```
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

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

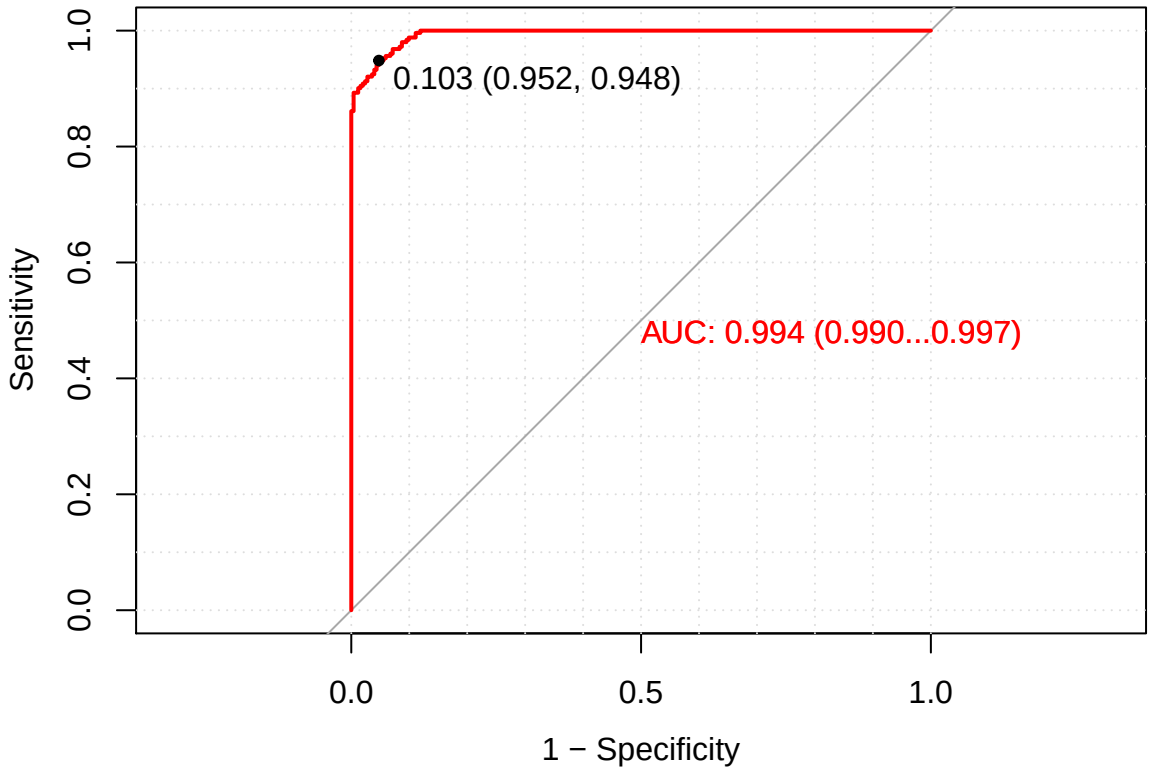
```
## Setting levels: control = 0, case = 1
```

```
## 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)
```

```
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

FALTA POR HACER

---

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