Tipología: PRA2 - Limpieza y análisis de un juego de datos

Autor: Carlos Mas Estevez

Junio 2022

Contents

Descripción del origen del conjunto de datos

Centraremos nuestro estudio en el análisis de un juego de datos con información relativa a más de 20000 partidas de ajedrez. La base de datos es de uso libre (CC0 1.0 Universal Public Domain Dedication) y se llama "Chess Game Dataset (Lichess)".

Se puede obtener a través de Google Dataset Search:

https://datasetsearch.research.google.com/search?src=0&query=chess&docid=L2cvMTFqOWJ0MnJuNw%3D%3D

o bien desde Kaggle:

https://www.kaggle.com/datasets/datasnaek/chess

La motivacón de nuestra investigación radica en la obtención de información acerca de las condiciones de victoria de partidas de ajedrez según determinados factores, tales como la puntuación ELO de cada jugador, la duración de la partida o el tipo de apertura (combinación inicial de movimientos). La elección de esta base de datos se debe a que ofrece una gran cantidad de datos a los que se les podrán aplicar técnicas supervisadas (y no supervisadas) a fin de extraer conclusiones en base al estudio de distintos atributos que

podrían condicionar la victoria de algún jugador. Al ser el jugador blanco el primero en realizar una jugada, resultará interesante saber cómo de relevante será este hecho a la hora de condicionar su victoria.

Para cada uno de estos juegos separados de Lichess, se recopiló estos datos utilizando la API de Lichess, que permite la recopilación del historial de juegos de cualquier usuario dado. Mucha información está contenida dentro de un solo juego de ajedrez, y más aún en un conjunto de datos completo con múltiples partidas. La ciencia de datos se trata de detectar patrones en los datos, y al tratarse el ajedrez de un juego principalmente de patrones, ha sido uno de los más usados en áreas de IA en el pasado. Este conjunto de datos recopila toda la información disponible de 20058 juegos y la presenta en un formato que es fácil de procesar para el análisis de, por ejemplo, lo que permite a un jugador ganar como blanco o negro, cuánto afectan los factores meta (fuera del juego) a un juego, la relación entre las aperturas y la victoria para blanco y negro, etc.

Varias preguntas que nos podríamos plantear a la hora de extraer conclusiones acerca de obtener victorias en partidas de ajedrez son: ¿Hasta qué punto es relevante la diferencia en puntuación ELO para poder predecir el ganador? ¿Existe una clara ventaja del equipo blanco en las partidas rápidas (poco tiempo de juego)? ¿Existe alguna apertura que influya en la condición de victoria en alguno de los jugadores? ¿Las partidas clasificadas (rated) suelen ser más conservadoras en las aperturas?

Mediante inspecciones visuales, tales como los histogramas o los boxplot, nos haremos una idea inicial de la distribucción de las partidas ganadas según determinados factores a fin de detectar tendencias. A la vez, realizaremos pruebas de covarianza y filtrado de datos para determinar más claramente las dependencias entre diferentes variables, así como un breve estudio de PCA.

```
# Cargamos los datos del documento "games.csv" que contiene más de 20000 registros
# de partidas de ajedrez

chess = read.csv("games.csv")
```

A continuación, mostramos un breve resumen de la tipología de los datos y su naturaleza a fin de verificar su estructura.

```
# Visualizamos la estructura de las columnas de nuestro dataset

structure = str(chess)
```

```
## 'data.frame':    20058 obs. of  16 variables:
##  $ id            : chr  "TZJHLljE" "l1NXvwaE" "mIICvQHh" "kWKvrqYL" ...
##  $ rated         : chr  "FALSE" "TRUE" "TRUE" "TRUE" ...
##  $ created_at    : num  1.5e+12 1.5e+12 1.5e+12 1.5e+12 1.5e+12 ...
##  $ last_move_at  : num  1.5e+12 1.5e+12 1.5e+12 1.5e+12 1.5e+12 ...
##  $ turns         : int  13 16 61 61 95 5 33 9 66 119 ...
##  $ victory_status: chr  "outoftime" "resign" "mate" "mate" ...
##  $ winner        : chr  "white" "black" "white" "white" ...
##  $ increment_code: chr  "15+2" "5+10" "5+10" "20+0" ...
##  $ white_id      : chr  "bourgris" "a-00" "ischia" "daniamurashov" ...
##  $ white_rating  : int  1500 1322 1496 1439 1523 1250 1520 1413 1439 1381 ...
##  $ black_id      : chr  "a-00" "skinnerua" "a-00" "adivanov2009" ...
##  $ black_rating  : int  1191 1261 1500 1454 1469 1002 1423 2108 1392 1209 ...
##  $ moves         : chr  "d4 d5 c4 c6 cxd5 e6 dxe6 fxe6 Nf3 Bb4+ Nc3 Ba5 Bf4" "d4 Nc6 e4 e5 f4 f6 dxe
##  $ opening_eco   : chr  "D10" "B00" "C20" "D02" ...
##  $ opening_name  : chr  "Slav Defense: Exchange Variation" "Nimzowitsch Defense: Kennedy Variation"
##  $ opening_ply   : int  5 4 3 3 5 4 10 5 6 4 ...
```

Tenemos un total de 20058 registros y 16 variables, las cuales explicamos a continuación:

- **id**: identificador partida
- **rated**: partida competitiva o amistosa (TRUE = competitiva / FALSE = amistosa)
- **created_at**: Tiempo de inicio de la partida
- **last_move_at**: Tiempo de final de la partida
- **turns**: número de turnos
- **victory_status**: condición de victoria (abandono, tiempo agotado, victoria común, tablas, etc.)
- **winner**: indica el jugador ganador (white/black), o tablas
- **increment_code**: cantidad de tiempo de la partida e incremento por turno.
- **white_id**: identificador de jugador blanco.
- **white_rating**: puntuación ELO jugador blanco.
- **black_id**: identificador de jugador negro.
- **black_rating**: puntuación ELO jugador negro.
- **moves**: movimientos de la partida en Standard Chess Notation.
- **opening_eco**: Código estandarizado de la apertura (https://www.365chess.com/eco.php)
- **opening_name**: Nombre de la apertura.
- **opening_ply**: Número de movimientos de la fase de apertura.

Preprocesamiento y gestión de características

Eliminación de valores nulos y outliers

Procedemos a hacer limpieza de datos, por lo que veremos en primer lugar si existen valores NA o en blanco en alguno de los atributos:

```
# Comprobamos si hay valores NA

colSums(is.na(chess))
```

```
##             id          rated     created_at   last_move_at          turns
##              0              0              0              0              0
## victory_status         winner increment_code       white_id   white_rating
##              0              0              0              0              0
##       black_id   black_rating          moves    opening_eco   opening_name
##              0              0              0              0              0
##    opening_ply
##              0
```

```
# Comprobamos si hay valores NA

colSums(chess=="")
```

```
##             id          rated     created_at   last_move_at          turns
##              0              0              0              0              0
## victory_status         winner increment_code       white_id   white_rating
##              0              0              0              0              0
##       black_id   black_rating          moves    opening_eco   opening_name
##              0              0              0              0              0
##    opening_ply
##              0
```

Observamos que no existen valores NA ni blancos en ninguno de los atributos, por lo que de momento no borraremos ningún registro.

Selección de variables

En primer lugar, nos dispondremos a limpiar la variable *increment_code*, la cual tiene el siguiente formato:

```
# Muestra de valores de la variable increment_code

head(chess$increment_code)
```

```
## [1] "15+2" "5+10" "5+10" "20+0" "30+3" "10+0"
```

```
tail(chess$increment_code)
```

```
## [1] "10+10" "10+10" "10+0"  "10+0"  "10+0"  "10+0"
```

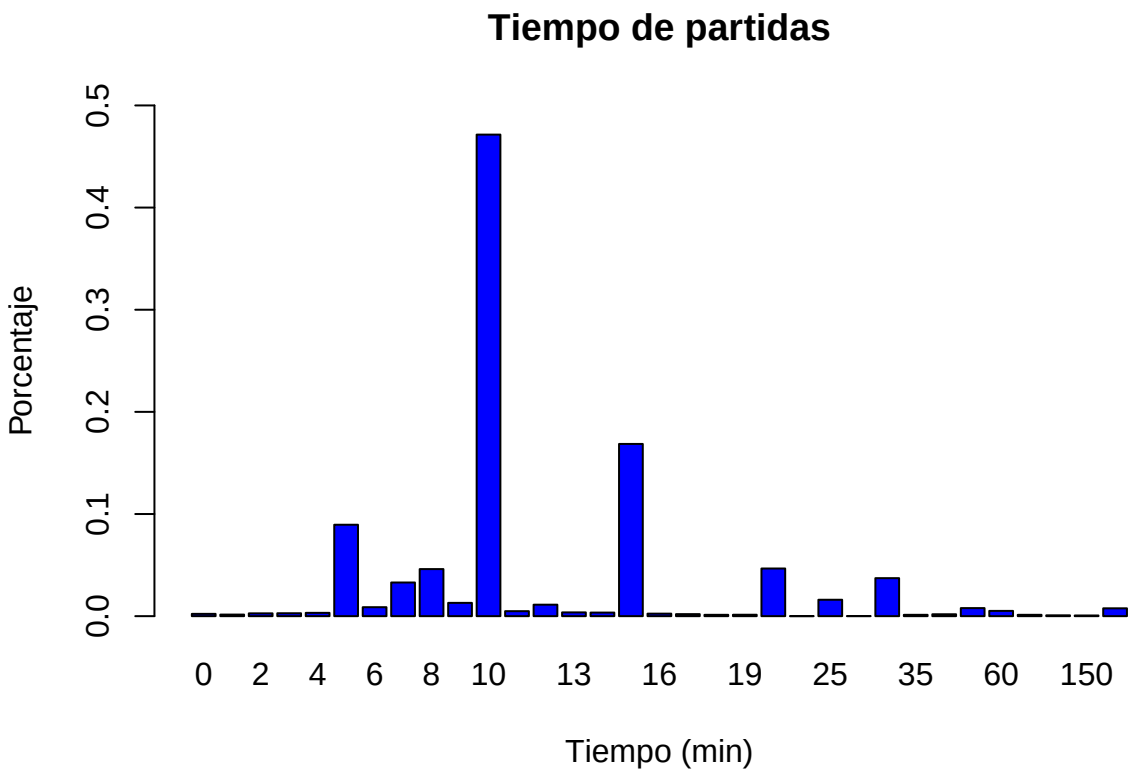
Este formato de la variable *increment_code* indica el tiempo inicial (en minutos) designado para la partida + un incremento de tiempo en segundos por cada jugada realizada. Por ejemplo, si una partida se juega con 5+10 significa que inicialmente cada jugador tiene 5 minutos de reloj para jugar la partida, más un incremento de 10 segundos por cada jugada realizada. Es decir, si un jugador invierte 1 minuto en una jugada, le quedarán 4 minutos + 10 segundos de incremento para la siguiente jugada que realice.

Así pues, vamos a separar la variable *increment_code* (carácter) en dos variables numéricas para poder manipular mejor los datos: el tiempo inicial de partida en minutos (*time*) y el incremento en segundos (*increment*):

```
# Creación de las variables *time* y *increment* a partir de *increment_code*

chess$time <- as.integer( str_match(chess$increment_code, '\\w+') )

chess$increment <- as.integer( chartr("+", " ", str_match(chess$increment_code, '\\+\\w+')) )
```

Hacemos un barplot de la variable *time* para detectar posibles outliers y hacernos una idea de cuánto suelen durar las partidas.

```
# Barplot del tiempo inicial de las partidas

counts <- table(chess$time)
barplot(prop.table(counts),col="blue", main="Tiempo de partidas",xlab ="Tiempo (min)", ylab = "Porcenta
```



Observamos que casi el 50% de las partidas duran 10 minutos por jugador. Además, vemos que hay algunas que tienen asignadas una duración inicial de 0 minutos; esto podría indicar que son partidas relámpago que únicamente cuentan con el tiempo de incremento después de cada jugada. Por ello, en caso de observar partidas de 0 minutos iniciales con 0 segundos de incremento, serán consideradas como outliers y se descartarán.

```
# Buscamos partidas co 0 minutos iniciales y 0 segundos de incremento

which(chess$time == 0 & chess$increment == 0)
```

```
## integer(0)
```

Ninguna partida tiene asignada 0 minutos iniciales con 0 segundos de incremento, por lo que podremos suponer que todas las partidas tienen un tiempo razonable para ser jugadas en función de la modalidad. Así pues, crearemos una nueva variable *modality* en la que categorizaremos las partidas según el tiempo inicial asignado a cada jugador (e ignorando los incrementos por jugada):

- **Partida relámpago**: tiempo inicial $<$ 10 minutos por jugador (*modality* = Light)
- **Partida rápida**: tiempo inicial entre 10-60 minutos por jugador (*modality* = Fast)
- **Partida clásica**: tiempo inicial mayor a 60 minutos por jugador (*modality* = Classic)

Esta clasificación la hemos escogido según la web de las leyes oficiales del ajedrez:

https://web.archive.org/web/20021204101135/http://handbook.fide.com/handbook.cgi?level=E&level=E1&level=01&

```r
# Categorizamos las partidas según la modalidad (partida relámpago, rápida o clásica)

chess$modality <- NA
chess$modality[chess$time < 10] <- "Light"
chess$modality[chess$time >= 10 & chess$time <= 30] <- "Fast"
chess$modality[chess$time > 30] <- "Classic"
```

En segundo lugar, nos dispondremos a crear una nueva variable *duration* que nos indique el tiempo de duración total de la partida, ya que tenemos dos columnas que nos hablan acerca del instante de inicio y final de la misma en formato Unix. No hemos de confundir esta variable con la creada anteriormente *time*, ya que *duration* nos hablará de los segundos transcurridos entre el inicio y final de la partida, la cual puede aplazarse incluso por varios días.

```r
# Creación de la variable time para saber la duración de la partida mediante difftime

time_begin <- as.POSIXct.numeric(chess$created_at, origin = "1970-01-01")
time_end <- as.POSIXct.numeric(chess$last_move_at, origin = "1970-01-01")
chess$duration <- difftime(time_end, time_begin)
table(chess$duration == 0)
```

```
##
## FALSE  TRUE
## 11510  8548
```

Observamos que hay 8548 registros en los que la duración de la partida no se registró, o es nula. Podemos considerar esto como un valor centinela o como una carencia en la recolección de datos de esas determinadas partidas. A su vez, inspeccionaremos los valores más altos de la variable *duration* a fin de determinar qué duraciones de partidas podemos considerar exageradamente altas y por tanto enmascarar dichos registros.

```r
table(chess$duration[chess$duration >= 10000000])
```

```
##
##     1e+07  10187296  10351034  11190773  12020863  12285839  19863947     2e+07
##       734         1         1         1         1         1         1         3
##  21301600  72349390     9e+07 605844701
##         1         1         2         1
```

Vemos que existen 734 registros con valor de 1e+07 segundos (~116 días). Esto se podría considerar algo habitual en ciertas partidas en determinados torneos o eventos de ajedrez en los que pueden posponerse o pausarse durante días; en las variables temporales que disponemos únicamente se tiene en cuenta el instante inicial y final de la partida, pero no el tiempo de juego. No obstante, la existencia de tantos registros con este valor concreto (1e+07) nos hace pensar que es un valor centinela o atípico pese existir otros 14 registros con mayores valores de duración de partida.

Por lo tanto, puesto que nos interesará obtener información relativa a las condiciones de victoria según la duración de la partida, enmascaremos todos los datos con registro de tiempo nulo o mayor a 10000000 segundos para evitar outliers. Así mismo, si la partida dura más de 8 horas se asumirá automáticamente que fue pospuesta en diferentes tandas.

```r
# Nos quedamos únicamente con aquellas duraciones de partida nulas o mayores a 1e+07 secs
chess <- subset(chess, chess$duration > 0 & chess$duration < 10000000)
```

```
chess$duration <- as.integer(chess$duration)
```

A continuación, inspeccionaremos las variables cualitativas *rated*, *victory_status*, *winner* y *increment_code* a fin de categorizar sus valores y detectar posibles fallos que sanear.

```
# Comprobamos qué valores toman las diferentes variables cualitativas rated

table(chess["rated"])
```

```
##
## False  True
##  2045  8717
```

Vemos que los valores que toma *rated* son 2. Así pues, reescribiremos los valores *True* y *False* en mayúsculas.

```
# Cambiamos los valores True y False a mayúsculas

chess$rated[chess$rated == "True"] <- "TRUE"
chess$rated[chess$rated == "False"] <- "FALSE"
table(chess["rated"])
```

```
##
## FALSE  TRUE
##  2045  8717
```

Ahora tenemos dos únicos valores de *rated*: 8717 partidas son competitivas y 2045 son amistosas.

```
# Comprobamos qué valores toman las diferentes variables cualitativas victory_status y winner

table(chess["victory_status"])
```

```
##
##      draw      mate outoftime    resign
##       511      3209       941      6101
```

```
table(chess["winner"])
```

```
##
## black  draw white
##  4832   532  5398
```

Observamos que *victory_status* solo toma 4 valores bien definidos según el tipo de victoria que haya habido (tablas, jaque mate, agotamiento de tiempo y abandono), siendo la más frecuente el abandono-rendición (resign). Así mismo, la variable *winner* toma solo 3 valores en función de quién haya ganado: black, white y draw (empate), existiendo una superioridad de 566 partidas ganadas por el jugador blanco.

Apreciamos que las variables *victory_status* y *winner* parecen contradecirse cuando toman el valor draw, pues toma valores de 511 y 532, respectivamente, cuando deberían dar lo mismo. A continuación, mostraremos los registros que difieren el valor draw en estas dos variables para determinar cuál puede ser la causa de esta aparente contradicción.

```
# Mostramos los registros que difieren de los valores draw en las variables victory_status y winner

chess.draw <- subset(chess, chess$victory_status != "draw" & chess$winner == "draw")
table(chess.draw[c("victory_status", "winner")])
```

```
##                winner
## victory_status draw
##      outoftime   21
```

Observamos que en todos los casos las variables *winner* = *draw* mientras que *victory_status* = *outoftime*. Además, hay un total de 21 registros con esta condición, que es justamente lo que marca la diferencia entre las apariciones de los valores *draw* en ambas variables (511 apariciones de *draw* en *victory_status* y 532 apariciones de *draw* en *winner*). Es decir, hay 21 registros en los que, pese a que el final de partida haya ocurrido por agotamiento de tiempo, el resultado ha quedado en tablas (empate). Así pues, a fin de facilitar el análisis, cambiaremos los valores *outoftime* de estos 21 registros por *draw*.

```
# En los registros en los que las variables victory_status = outoftime y winner = draw,
# cambiaremos el valor outoftime por draw.

chess$victory_status[chess$victory_status == "outoftime" & chess$winner == "draw"] <- "draw"
table(chess["victory_status"])
```

```
##
##      draw      mate outoftime    resign
##       532      3209       920      6101
```

```
table(chess["winner"])
```

```
##
## black  draw white
##  4832   532  5398
```

Vemos que ahora sí coinciden la cantidad de apariciones *draw* en las variables *victory_status* y *winner*.

Las demás variables categóricas (*id*, *white_id*, *black_id*, *opening_eco*, *opening_name*) asumiremos que no es necesaria una limpieza debido a que no existen valores perdidos.

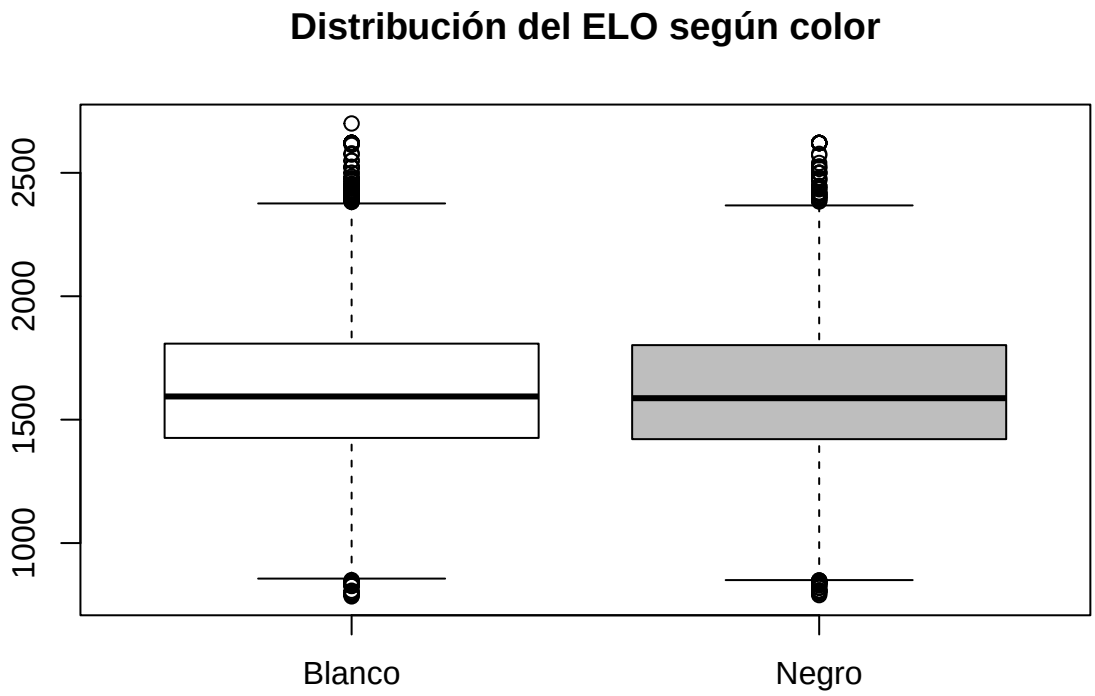
A continuación, mostramos los parámetros estadísticos básicos de las variables cuantitativas (media, mediana, rango y cuantiles) para hacernos una idea de la distribución de los datos, así como asegurarnos que no haya ningún dato fuera de rango:

```
# Inspección variables cuantitativas

summary(chess[c("created_at", "last_move_at", "turns", "white_rating", "black_rating", "moves", "opening
```

```
##    created_at         last_move_at          turns         white_rating
##  Min.   :1.377e+12   Min.   :1.377e+12   Min.   :  1.00   Min.   : 784
##  1st Qu.:1.481e+12   1st Qu.:1.481e+12   1st Qu.: 38.00   1st Qu.:1426
##  Median :1.500e+12   Median :1.500e+12   Median : 56.00   Median :1594
##  Mean   :1.483e+12   Mean   :1.483e+12   Mean   : 61.25   Mean   :1618
##  3rd Qu.:1.504e+12   3rd Qu.:1.504e+12   3rd Qu.: 80.00   3rd Qu.:1808
##  Max.   :1.504e+12   Max.   :1.504e+12   Max.   :349.00   Max.   :2700
```

```
##   black_rating     moves              opening_ply
##  Min.   : 789   Length:10762       Min.   : 1.000
##  1st Qu.:1421   Class :character   1st Qu.: 3.000
##  Median :1587   Mode  :character   Median : 4.000
##  Mean   :1610                      Mean   : 4.926
##  3rd Qu.:1802                      3rd Qu.: 6.000
##  Max.   :2621                      Max.   :28.000
```

```r
# Mostramos la distribución de las variables white_rating y black_rating con un boxplot

boxplot(chess$white_rating,
        chess$black_rating,
        col=c('white', 'grey'),
        main='Distribución del ELO según color',
        ylab='',
        names=c("Blanco", "Negro"))
```



Del gráfico boxplot anterior podemos observar que tanto la mediana como los rangos intercuartílicos de *white_rating* y *black_rating* toman valores muy semejantes (medianas de 1594 y 1587, respectivamente; rangos intercuartílicos de 1426-1808 y 1421-1802, respectivamente), por lo que vamos a suponer que ambas variables siguen distribuciones iguales y son aptas para compararlas entre ellas. Además, se observa que no existen valores centinela o atípicos, pues el rango de las variables abarcan valores con significado real (*white_rating* tiene 784 de mínimo y 2700 de máximo, mientras que *black_rating* tiene 789 de mínimo y 2621 de máximo), por lo que no enmascaremos ningún valor según estas consideraciones.

Construcción del conjunto de datos final

Así pues, para que nuestro set de datos sea más fácil de manipular y en aras de mejorar la objetividad del análisis, nos centraremos únicamente en aquellas partidas (una partida = un registro) cuyos participantes tengan una diferencia pequeña de puntuación ELO a fin de suponer que ambos tienen el mismo nivel y juegan con igualdad de condiciones. Además, siguiendo este criterio también podremos despreciar otros factores personales, tales como efectos psicológicos que puedan alterar el resultado de la partida (como por ejemplo, la intimidación por jugar contra alguien con mucha más experiencia, o la falta de seriedad por parte de alguien muy superior y que basa toda su partida en experimentar jugadas y/o técnicas que perturbarían el análisis).

Por ello, vamos a crear una nueva variable *diff* que nos indicará la diferencia de puntuación ELO entre 2 jugadores. En decir, calcularemos la diferencia entre *white_rating* y *black_rating* tal que un resultado positivo de +200 indicaría que el jugador blanco tiene 200 puntos más de ELO que el jugador negro.

```
# Creación de la variable diff

chess$diff <- chess$white_rating - chess$black_rating
```

Cremos una nueva variable dicotómica *diff_num* que tomará 3 posibles valores en función de la diferencia de ELO de los jugadores, tal que:

- *diff_num* = 1 si *diff* > 0
- *diff_num* = 0 si *diff* = 0
- *diff_num* = -1 si *diff* < 0

```
# Creación de la variable diff_num

chess$diff_num <- NA
chess$diff_num[chess$diff > 0] <- 1
chess$diff_num[chess$diff == 0] <- 0
chess$diff_num[chess$diff < 0] <- -1
```

Creamos una nueva variable *winner_num* a la que le asignaremos valores numéricos según el valor que tome *winner*, tal que:

- *winner_num* = 1 si *winner* = white
- *winner_num* = 0 si *winner* = draw
- *winner_num* = -1 si *winner* = black

```
# Creación de la variable winner_num

chess$winner_num <- NA
chess$winner_num[chess$winner == "white"] <- 1
chess$winner_num[chess$winner == "draw"] <- 0
chess$winner_num[chess$winner == "black"] <- -1
```

También crearemos una variable categórica *rated_num* que nos indique si una partida se juega de modo competitivo (*rated_num* = 1) o amistoso (*rated_num* = 0) que nos será de utilidad en los posteriores análisis.

```
# Creación de la variable rated_num

chess$rated_num <- NA
chess$rated_num[chess$rated == "TRUE"] <- 1
chess$rated_num[chess$rated == "FALSE"] <- 0
```

A continuación, crearemos una función que hará una selección de un dataset que discrimina todos aquellos registros cuyos jugadores tengan una diferencia de ELO mayor a la indicada. Además, nos dará información de la covarianza entre las variables dicotómicas *winner_num* y *diff_num*.

```
# Función que genera un subset con los registros en que los jugadores tengan una diferencia de ELO
# menor o igual a la indicada en dif_rat

min_rating <- function(df, dif_rat){

  y <- df[df$diff <= dif_rat & df$diff >= -dif_rat,]

  print(paste("La covarianza es de",
              round(cov(y$winner_num, y$diff_num)*100, 2),
              "% con diferencia de ELO <= ",
              dif_rat,
              ". El dataset tiene los siguientes registros:",
              length(y$winner_num)

              ) )


  return(y)
}
```

Veamos ahora con qué dataset nos quedamos; nos interesará aquel que sea suficientemente grande y tenga una covarianza pequeña entre *winner_num* y *diff_num*, ya que implicará que podremos suponer que las victorias no se deben tanto por la diferencia de ELO entre jugadores, sino debido a otros factores que estudiaremos posteriormente.

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 2000
chess.reduced.1 <- min_rating(chess, 2000)
```

```
## [1] "La covarianza es de 29.18 % con diferencia de ELO <=  2000 . El dataset tiene los siguientes reg
```

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 1000
chess.reduced.2 <- min_rating(chess, 1000)
```

```
## [1] "La covarianza es de 29.01 % con diferencia de ELO <=  1000 . El dataset tiene los siguientes reg
```

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 500
chess.reduced.3 <- min_rating(chess, 500)
```

```
## [1] "La covarianza es de 26.45 % con diferencia de ELO <=  500 . El dataset tiene los siguientes regi
```

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 300
chess.reduced.4 <- min_rating(chess, 300)
```

```
## [1] "La covarianza es de 21.39 % con diferencia de ELO <=  300 . El dataset tiene los siguientes regi
```

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 200
chess.reduced.5 <- min_rating(chess, 200)
```

```
## [1] "La covarianza es de 16.34 % con diferencia de ELO <=  200 . El dataset tiene los siguientes regi
```

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 150
chess.reduced.6 <- min_rating(chess, 150)
```

```
## [1] "La covarianza es de 13.5 % con diferencia de ELO <=  150 . El dataset tiene los siguientes regis
```

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 100
chess.reduced.7 <- min_rating(chess, 100)
```

```
## [1] "La covarianza es de 8.48 % con diferencia de ELO <=  100 . El dataset tiene los siguientes regis
```

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 50
chess.reduced.8 <- min_rating(chess, 50)
```

```
## [1] "La covarianza es de 4.14 % con diferencia de ELO <=  50 . El dataset tiene los siguientes regist
```

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 25
chess.reduced.9 <- min_rating(chess, 25)
```

```
## [1] "La covarianza es de 4.68 % con diferencia de ELO <=  25 . El dataset tiene los siguientes regist
```

```
# Covarianza y número de registros de sub_dataset con diferencias de ELO menores a 10
chess.reduced.10 <- min_rating(chess, 10)
```

```
## [1] "La covarianza es de 2.09 % con diferencia de ELO <=  10 . El dataset tiene los siguientes regist
```

En vista de los resultados obtenidos de las covarianzas de las variables *winner_num* y *diff_num*, podemos determinar la dependencia de condición de victoria según la diferencia de puntuación ELO de los jugadores. Así pues, observamos que existe una covarianza de 29.2% cuando la diferencia de ELO es de 2000, hasta una covarianza de 2.1% cuando la diferencia de ELO es de 10. Sin embargo, decidimos quedarnos con el sub_dataset *chess.reduced.7* y usarlo para el resto del estudio, pues nos ofrece una covarianza bastante baja (**8.5%**) con un alto número de registros (**4942**).

```
# Renombramos el dataset *chess.reduced.8* para usarlo más cómodamente a lo largo de la práctica
chess <- chess.reduced.7
```

```
# Exportamos el nuevo set de datos
write.csv(chess, "chess_clean.csv", row.names = FALSE)
```

Ahora tenemos un nuevo set de datos *chess* con el que poder trabajar más cómodamente debido a la reducción de registros y en el que podemos suponer mayor igualdad de condiciones entre los diferentes jugadores (todos ellos con ELO semejante de 100 o menos puntos de diferencia).

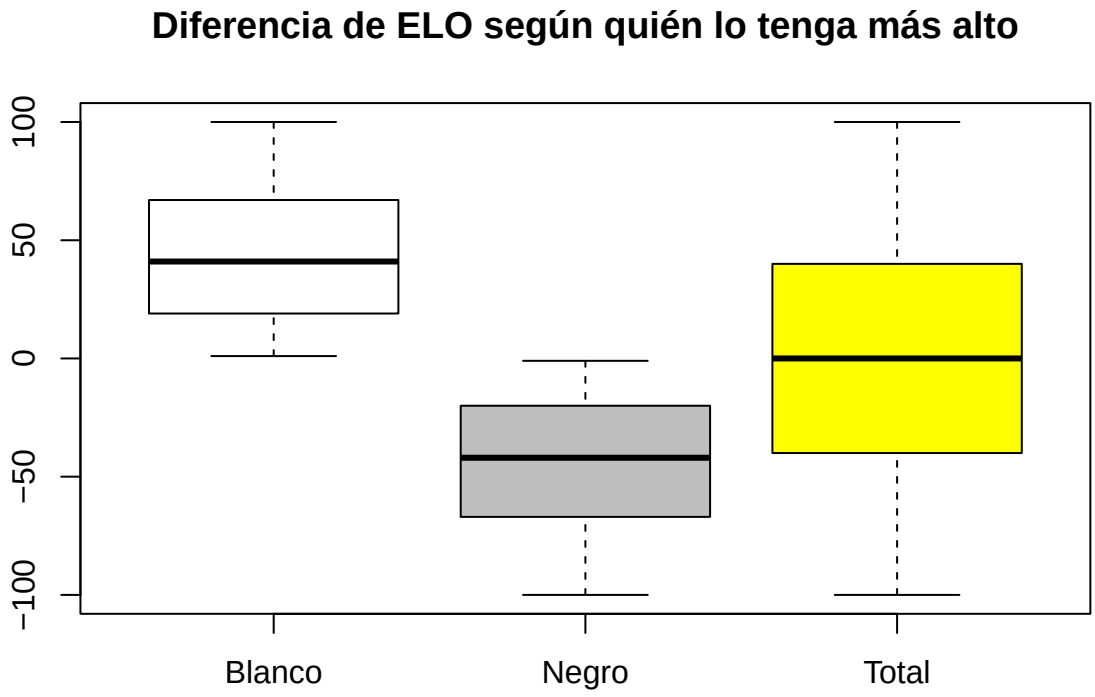
Análisis de los datos

Análisis exploratorio

Con la limpieza de datos realizada, ya podremos inspeccionar algunas características de las diferentes variables a fin de comprobar las tendencias que condicionan la victoria de cada jugador.

Mostramos a continuación un boxplot del comportamiento de la variable *diff* a fin de comprobar visualmente que la diferencia de puntuación ELO tiene un comportamiento similar según el jugador que lo tenga más alto. Nos interesará que las distribuciones sean semejantes para poder comparar más correctamente la condición de victoria a pequeñas diferencias de ELO, pues necesitaremos dos distribuciones similares de jugadores con diferencia de ELO superior a la del contrincante (tanto para el blanco como para el negro).

```
# Mostramos la distribución de la variable diff según su valor

boxplot(chess$diff[chess$diff>0],
        chess$diff[chess$diff<0],
        chess$diff,
        col=c('white', 'grey', 'yellow'),
        main='Diferencia de ELO según quién lo tenga más alto',
        ylab='',
        names=c("Blanco", "Negro", "Total"))
```



Observamos que tanto la diferencia de ELO del jugador blanco como del negro siguen una distribución similar: son simétricas, situándose sus medianas a un valor de +50 y -50 respectivamente. La distribución total (amarilla) con mediana cercana a 0 nos confirma que las diferencias de puntuaciones ELO son aproximadamente simétricas, implicando que estamos trabajando con un conjunto de datos que no está sesgado con

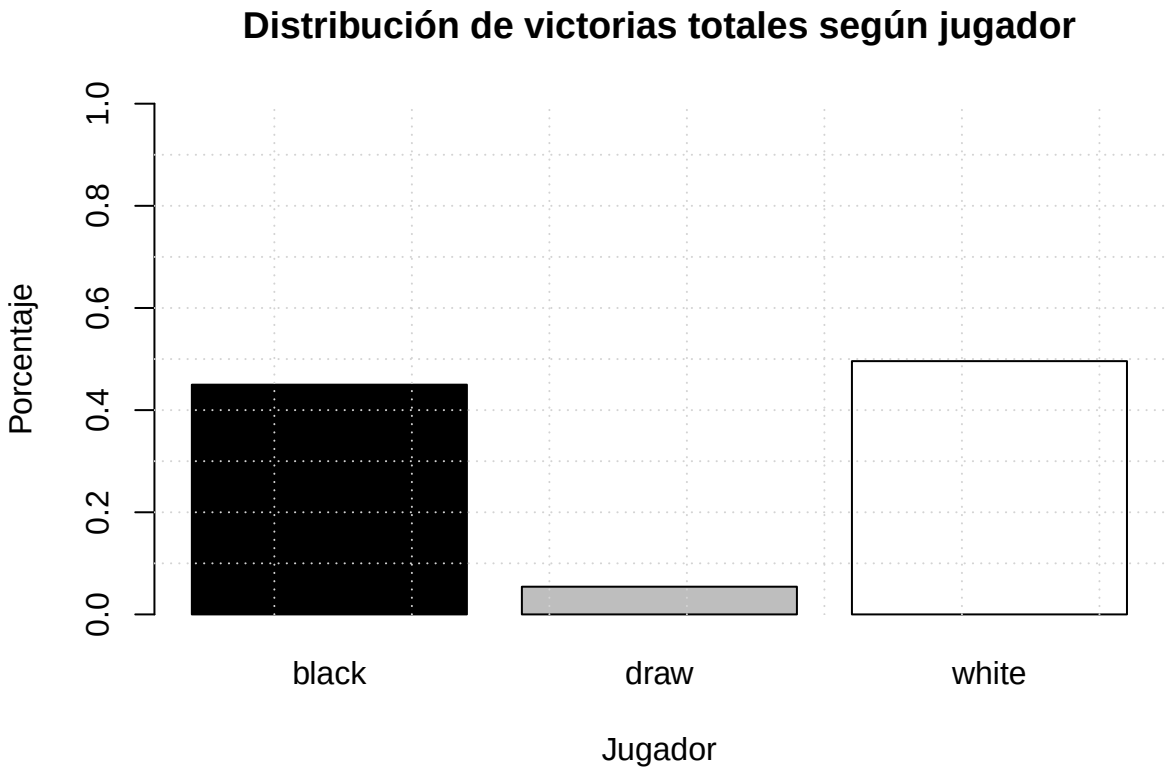
una dominancia de puntuación ELO por parte de ninguno de los jugadores. Así pues, supondremos que la distribución de la diferencia de ELO es normal.

Veremos ahora la proporción de partidas ganadas de cada jugador.

```
# Distribución de victorias totales
table(chess["winner"])
```

```
##
## black  draw white
##  2223   268  2451
```

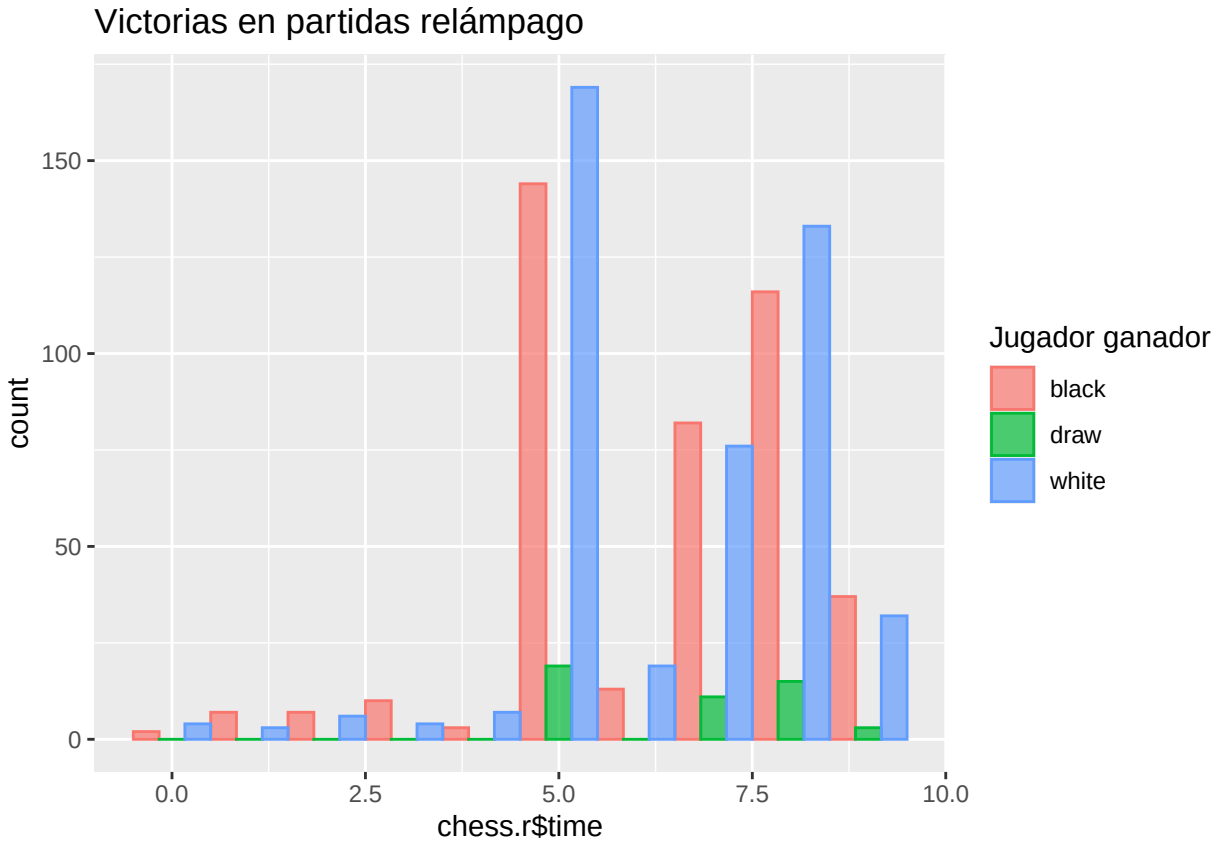
```
counts <- table(chess$winner)
barplot(prop.table(counts),
        col=c("black", "grey", "white"),
        ylim=c(0,1),
        main="Distribución de victorias totales según jugador",
        xlab="Jugador",
        ylab="Porcentaje" )

grid(ny=10)
```



Aproximadamente ambos jugadores tienen la misma proporción de victorias, aunque el blanco presenta una ligera superioridad no despreciable (~10%). A continuación, veremos cómo se comporta esta tendencia según la duración inicial de la partida.

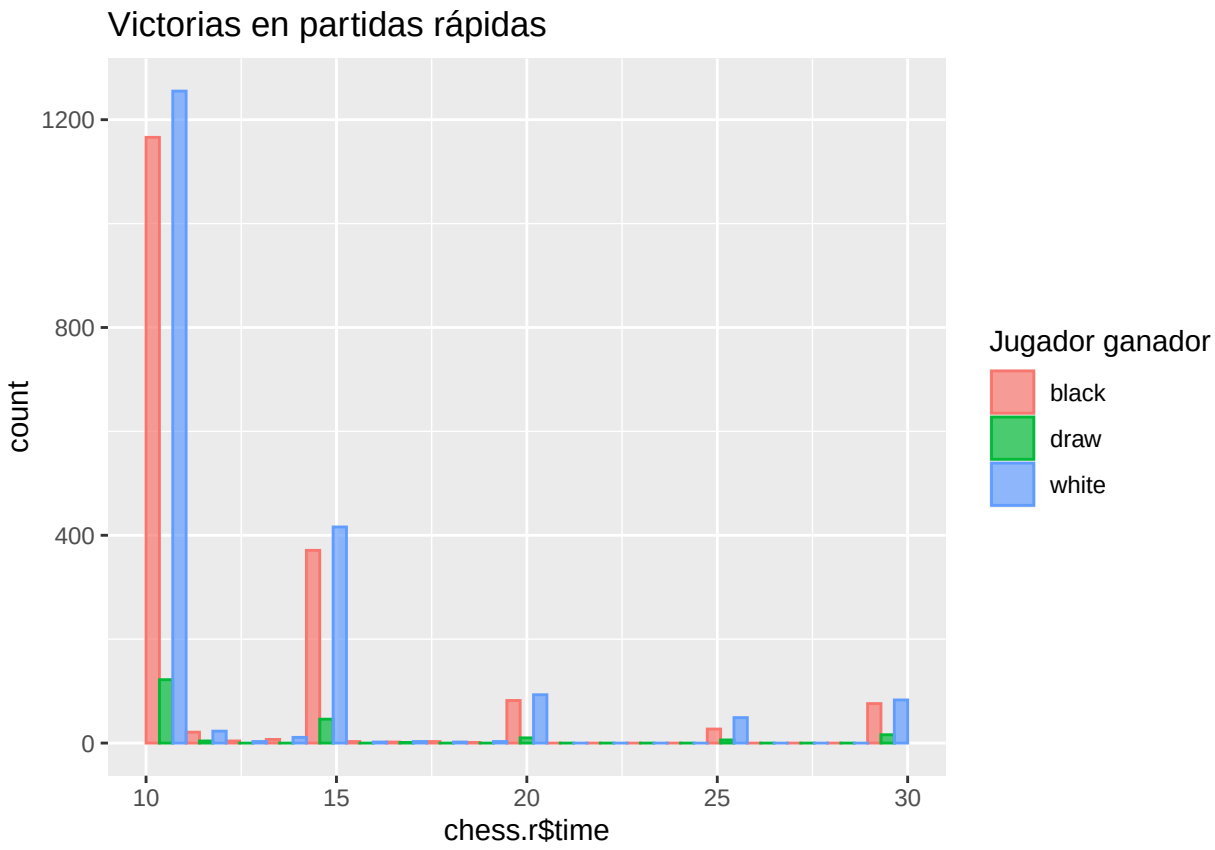
```r
# Cantidad de victorias de cada jugador en partidas relámpago

chess.r <- chess[chess$modality == "Light",]

ggplot(chess.r, aes(x = chess.r$time, fill = chess.r$winner, colour = chess.r$winner)) +
  geom_histogram(alpha = 0.7, position = "dodge", bins = 10) +
  ggtitle("Victorias en partidas relámpago") +
  guides(fill = guide_legend(title = "Jugador ganador"),
         colour = guide_legend(title = "Jugador ganador"))
```

```
## Warning: Use of ‘chess.r$time‘ is discouraged. Use ‘time‘ instead.

## Warning: Use of ‘chess.r$winner‘ is discouraged. Use ‘winner‘ instead.

## Warning: Use of ‘chess.r$winner‘ is discouraged. Use ‘winner‘ instead.
```



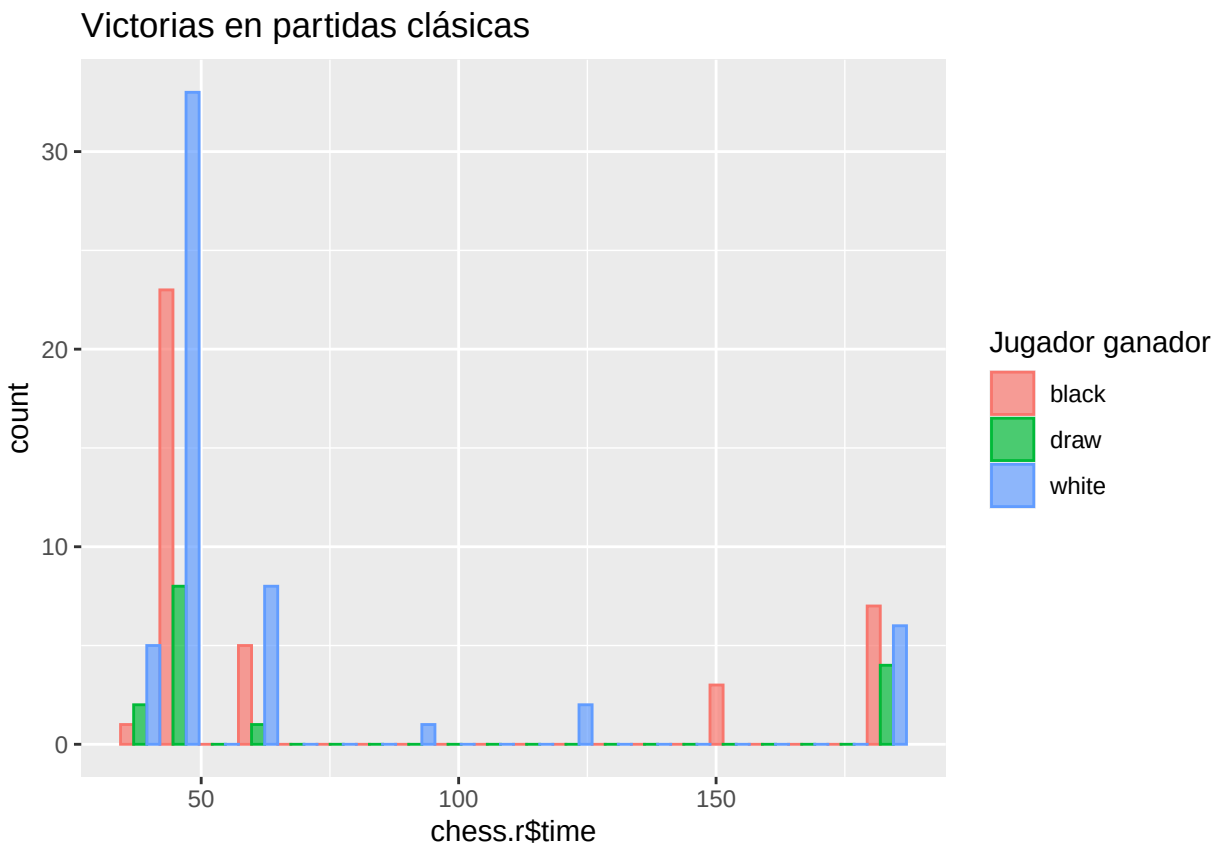

```r
# Cantidad de victorias de cada jugador en partidas rápidas

chess.r <- chess[chess$modality == "Fast",]

ggplot(chess.r, aes(x = chess.r$time, fill = chess.r$winner, colour = chess.r$winner)) +
  geom_histogram(alpha = 0.7, position = "dodge", bins = 20) +
  ggtitle("Victorias en partidas rápidas") +
  guides(fill = guide_legend(title = "Jugador ganador"),
         colour = guide_legend(title = "Jugador ganador"))
```

```
## Warning: Use of ‘chess.r$time‘ is discouraged. Use ‘time‘ instead.

## Warning: Use of ‘chess.r$winner‘ is discouraged. Use ‘winner‘ instead.

## Warning: Use of ‘chess.r$winner‘ is discouraged. Use ‘winner‘ instead.
```




```
# Cantidad de victorias de cada jugador en partidas clásicas

chess.r <- chess[chess$modality == "Classic",]

ggplot(chess.r, aes(x = chess.r$time, fill = chess.r$winner, colour = chess.r$winner)) +
  geom_histogram(alpha = 0.7, position = "dodge", bins = 20) +
  ggtitle("Victorias en partidas clásicas") +
  guides(fill = guide_legend(title = "Jugador ganador"),
         colour = guide_legend(title = "Jugador ganador"))
```

```
## Warning: Use of ‘chess.r$time‘ is discouraged. Use ‘time‘ instead.

## Warning: Use of ‘chess.r$winner‘ is discouraged. Use ‘winner‘ instead.

## Warning: Use of ‘chess.r$winner‘ is discouraged. Use ‘winner‘ instead.
```




```
summary(chess$modality=="Classic")
```

```
##    Mode   FALSE    TRUE
## logical    4833     109
```

Vemos que la tendencia de victoria es más o menos homogénea en las tres modalidades. En las rápidas y relámpago observamos que hay superioridad por parte del jugador blanco, aunque en las clásicas hay poca estadística para poder sacar conclusiones firmes pues solo disponemos de 109 datos.

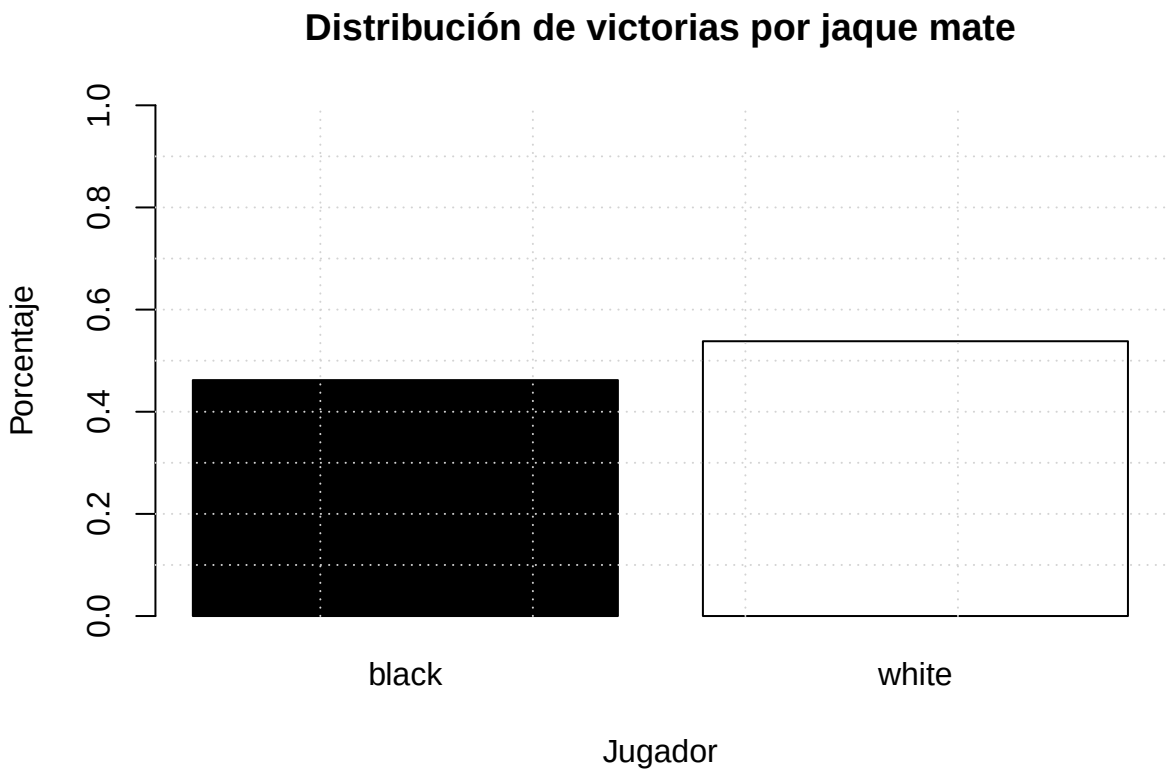
Ahora veremos cuál es la proporción de victorias según cómo suelan acabar las partidas (*victory_status*).

```
table(chess["victory_status"])
```

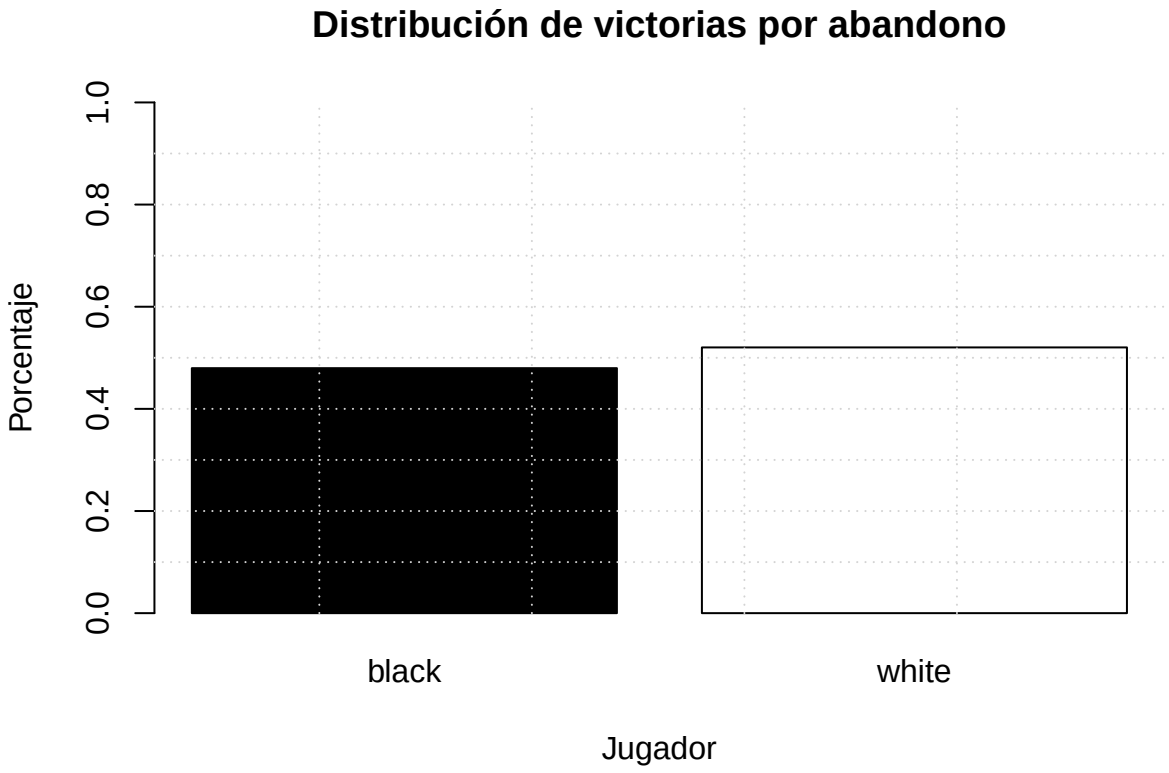
```
##
##      draw      mate outoftime    resign
##       268      1444       447      2783
```

```
# Victorias por jaque mate

counts <- table(chess$winner[chess$victory_status == "mate"])
barplot(prop.table(counts),
        col=c("black", "white"),
        ylim=c(0,1),
        main="Distribución de victorias por jaque mate",
```

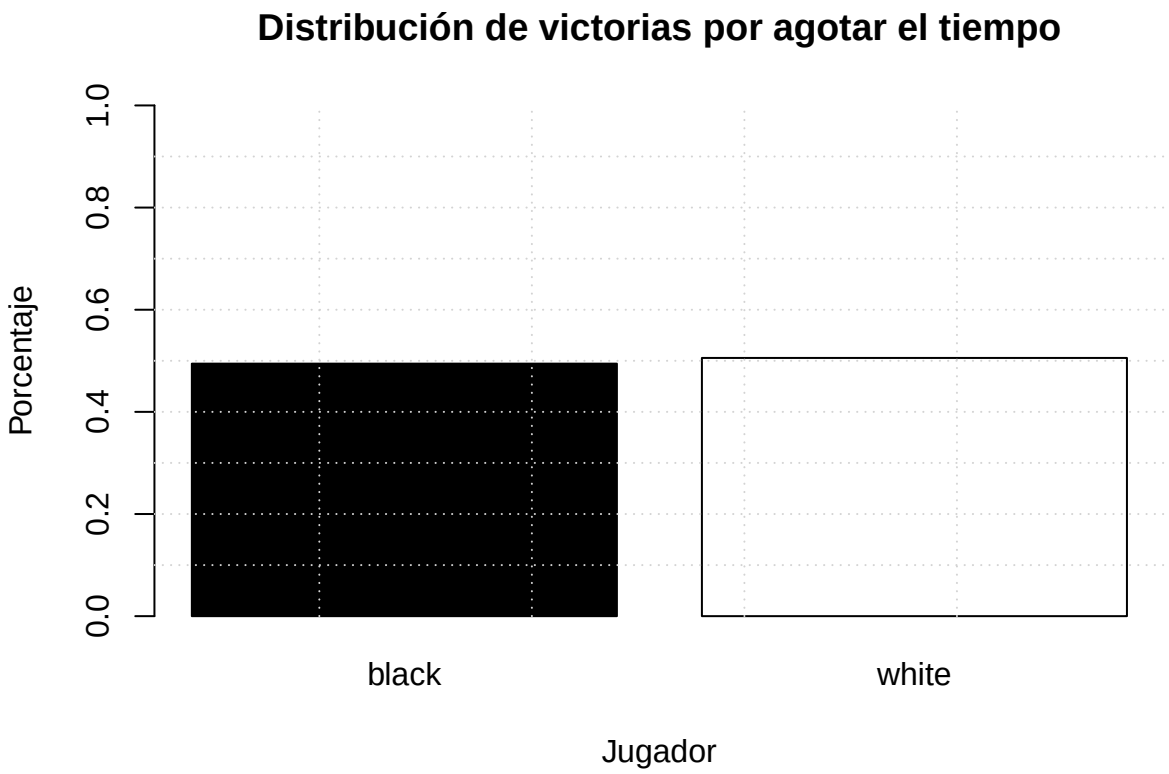
```
        xlab="Jugador",
        ylab="Porcentaje" )

grid(ny=10)
```




```
# Victorias por abandono

counts <- table(chess$winner[chess$victory_status == "resign"])
barplot(prop.table(counts),
        col=c("black", "white"),
        ylim=c(0,1),
        main="Distribución de victorias por abandono",
        xlab="Jugador",
        ylab="Porcentaje" )

grid(ny=10)
```



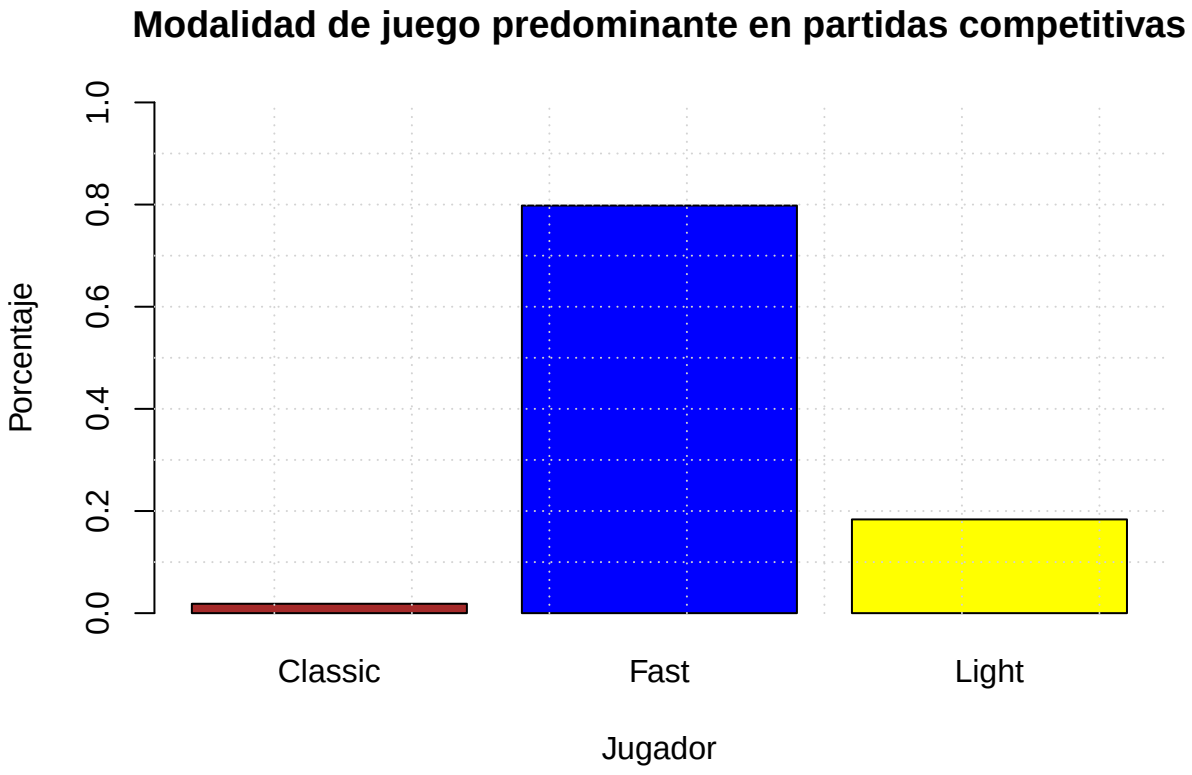

```r
# Victorias por agotar el tiempo

counts <- table(chess$winner[chess$victory_status == "outoftime"])
barplot(prop.table(counts),
        col=c("black", "white"),
        ylim=c(0,1),
        main="Distribución de victorias por agotar el tiempo",
        xlab="Jugador",
        ylab="Porcentaje" )

grid(ny=10)
```



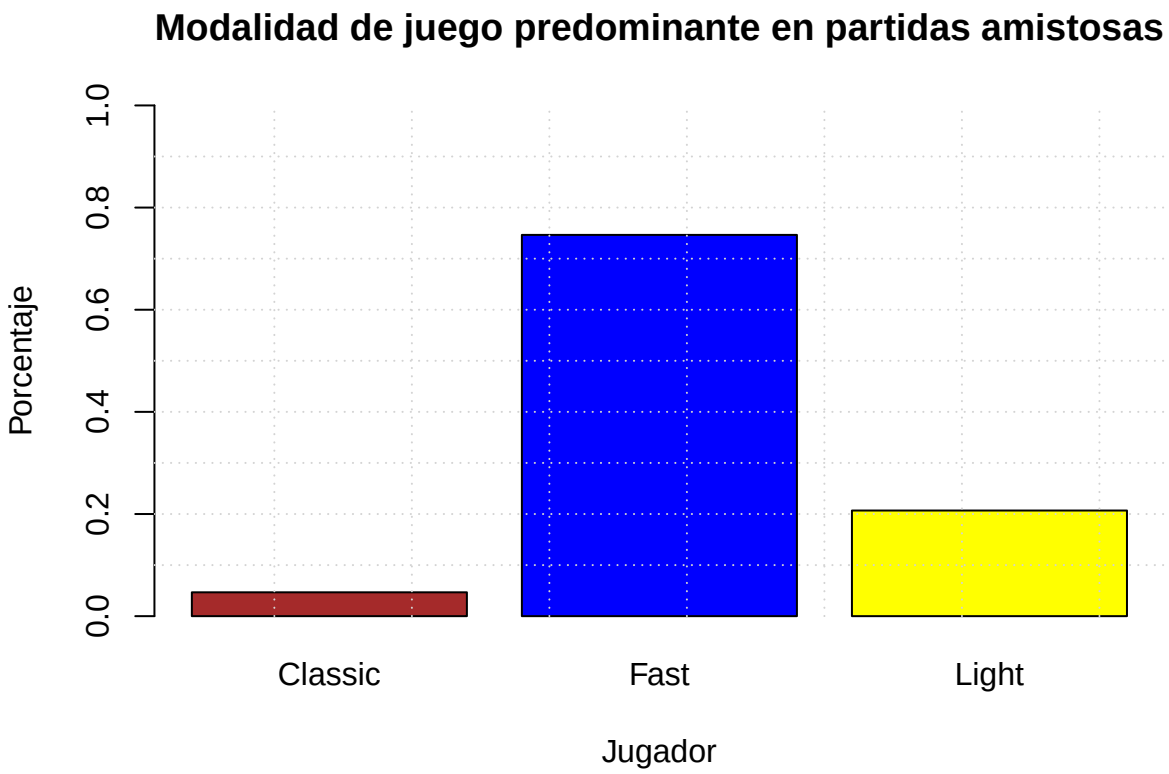
Aunque en todos los casos haya una ligera superioridad de victoria por parte del jugador blanco, esta es prácticamente despreciable en el caso de agotamiento de tiempo. Cuando la partida acaba por jaque mate o abandono, vemos más claramente una tendencia del blanco a ganar.

Nos interesará saber también qué modalidad se suele jugar con mayor frecuencia en las partidas competitivas y amistosas.

```
# Modalidad en partidas competitivas

counts <- table(chess$modality[chess$rated == "TRUE"])
barplot(prop.table(counts),
        col=c("brown", "blue" ,"yellow"),
        ylim=c(0,1),
        main="Modalidad de juego predominante en partidas competitivas",
        xlab="Jugador",
        ylab="Porcentaje" )

grid(ny=10)
```




```r
# Modalidad en partidas amistosas

counts <- table(chess$modality[chess$rated == "FALSE"])
barplot(prop.table(counts),
        col=c("brown", "blue" ,"yellow"),
        ylim=c(0,1),
        main="Modalidad de juego predominante en partidas amistosas",
        xlab="Jugador",
        ylab="Porcentaje" )

grid(ny=10)
```



Aunque el cambio de tendencia sea pequeño, observamos que hay menos partidas jugadas en modalidad rápida (fast) cuando se tratan de partidas amistosas, aumentando en estas los casos en partidas relámpago (light) y clásicas.

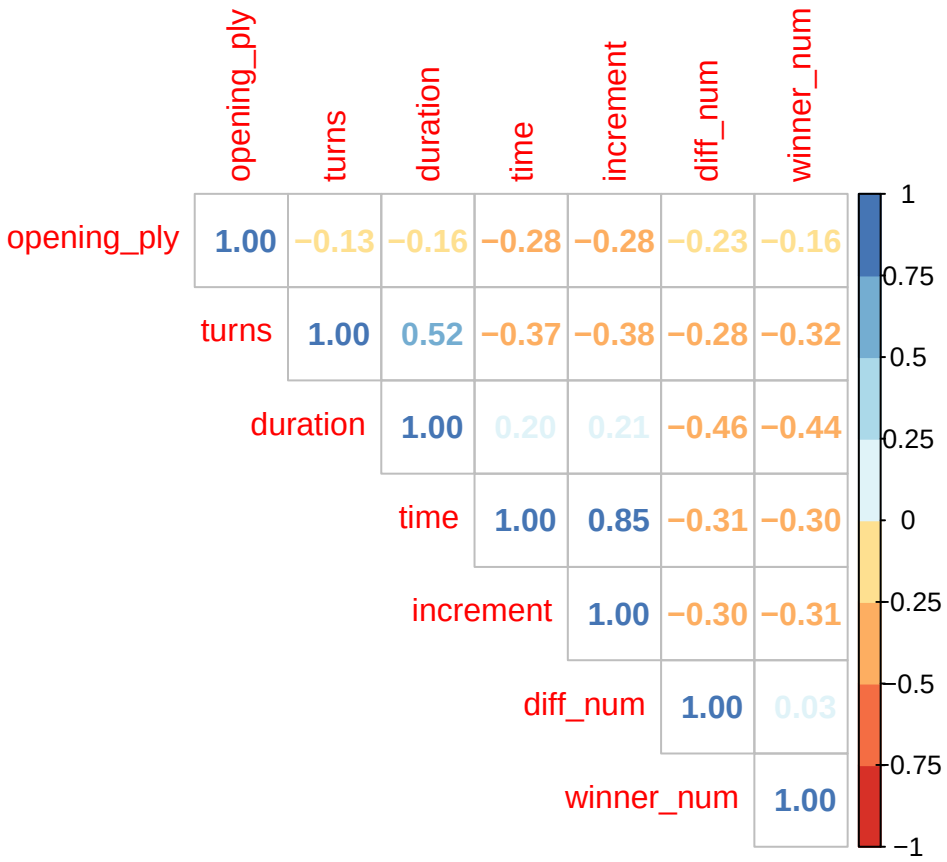
Análisis estadístico

Una vez hecho un breve análisis visual, podríamos llegar a pensar que en la mayoría de modalidades y situaciones el jugador blanco tiene más posibilidades de ganar. No obstante, requeriremos de la realización de tests estadísticos para verificar si esta diferencia es relevante.

Empezaremos por estudiar la correlación entre las distintas variables. En particular, nos interesará ver el comportamiento de *winner_num* en función de las demás variables. Recordemos que *winner_num*=1 significa que el blanco gana, *winner_num*=-1 gana el negro y *winner_num*=0 son tablas (empate).

```
# Tabla de correlación

chess.numeric <- chess[ , unlist(lapply(chess, is.numeric))]

chess.numeric <- chess.numeric[ , c("turns", "opening_ply", "time", "increment",
                                    "duration", "diff_num", "winner_num")]

# Cálculo correlaciones

cor <- cor(chess.numeric)

# Mapa de calor de correlaciones
```

```
corrplot(cor(cor), type="upper", order="hclust", method="number",
         col=brewer.pal(n=8, name="RdYlBu"))
```



Centrándonos en el comportamiento de *winner_num*, vemos que tiene una correlación de:

- *opening_ply* = -16%. Cuantas más jugadas dure la apertura, hay una baja tendencia a que la partida la gane el negro.
- *turns* = -32%. Cuantos más turnos dure la partida, hay una baja tendencia a que la partida la gane el negro, y una alta para el blanco en caso de que la partida dure pocas jugadas.
- *duration* = -44%. Cuanto menor sea la duración total de la partida, habrá mayor probabilidad de que el negro gane.
- *time* = -30%. Cuanto menor sea el tiempo establecido de partida, habrá más probabilidad de que el negro gane.
- *increment* = -31%. Parecido al comportamiento de *time*, cuanto menor incremento de tiempo haya, más probabilidad de que el negro gane.
- *diff_num* = 3%. Hay muy baja correlación entre la diferencia de puntuación de los jugadores y la condición de victoria. Esto es esperable, pues ya nos encargamos previamente de mitigar esta dependencia al haber escogido un dataset cuyos jugadores tuvieran poca diferencia de puntuación.

También estudiaremos la covarianza entre quién gana y si la partida es competitiva.

```r
# Covarianza entre la condición de victoria y si la partida es competitiva.

cov(chess$winner_num, chess$rated_num)
```

```
## [1] 0.002563229
```

Vemos que ambas variables tienen una relación inferior al 1%, por lo que asumiremos que la condición de que la partida sea competitiva o amistosa apenas afecta a la condición de victoria.

Modelo de regresión logística

Así pues, predeciremos la probablidad de que un determinado jugador gane la partida dependiendo de las condiciones de juego. Como la variable de interés *winner_num* tiene valores -1, 0, 1 (es categórica), la convertiremos en una variable dicotómica (negro = 0, blanco = 1) a fin de poder hacer un modelo de regresión logística. En este modelo tendremos que excluir todas aquellas partidas que hayan acabado en tablas, por lo que crearemos un nuevo dataset *chess.prediction* y lo prepararemos para la regresión logística.

```r
# Creción dataset chess.prediction

chess.prediction <- chess[chess$winner_num != 0,]

# Renombramos el valor de winner_num tal que el ganador blanco = 1 y el ganador negro = 0

chess.prediction$winner_num[chess.prediction$winner_num == -1] <- 0
```

Veamos la cantidad de jugadores blancos y negros que han ganado cada partida.

```r
# Cantidad de jugadores blancos ganadores

winners.white <- length(which(chess.prediction$winner_num == 1))

print(paste("La catidad de jugadores blancos ganadores en chess.prediction es: ",
            winners.white))
```

```
## [1] "La catidad de jugadores blancos ganadores en chess.prediction es:  2451"
```

```r
# Cantidad de jugadores negros ganadores

winners.black <- length(which(chess.prediction$winner_num == 0))

print(paste("La catidad de jugadores negros ganadores en chess.prediction es: ",
            winners.black))
```

```
## [1] "La catidad de jugadores negros ganadores en chess.prediction es:  2223"
```

```r
chess.prediction$winner_num <- as.factor(chess.prediction$winner_num)
```

```r
# Diferencia entre el número de victorias de cada jugador

print(paste("Hay un",
            round( (winners.white - winners.black) / winners.black * 100, 2),
            "% más de partidas ganadas \n por los jugadores blancos que por los negros."))
```

```
## [1] "Hay un 10.26 % más de partidas ganadas \n por los jugadores blancos que por los negros."
```

Ante esta diferencia no despreciable de victorias, nos dispondremos a generar un modelo de regresión logística. Incluimos las variables *rated* (partida competitiva o amistosa) y *diff_num* (diferencia de puntuación ELO) pese a tener poca correlación con *winner_num*.

```
# Modelo de regresión logística

chess.logist <- glm(formula = winner_num ~ opening_ply + turns + time +
                      increment + diff_num + rated_num + duration,
                data = chess.prediction, family=binomial(link=logit))

summary(chess.logist)
```

```
##
## Call:
## glm(formula = winner_num ~ opening_ply + turns + time + increment +
##     diff_num + rated_num + duration, family = binomial(link = logit),
##     data = chess.prediction)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.445  -1.185   1.005   1.131   1.543
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.270e-01  1.161e-01   1.094  0.27398
## opening_ply  1.516e-02  1.037e-02   1.462  0.14362
## turns       -3.084e-03  9.956e-04  -3.098  0.00195 **
## time         2.395e-03  3.287e-03   0.728  0.46631
## increment   -2.842e-03  3.108e-03  -0.914  0.36047
## diff_num     1.825e-01  2.976e-02   6.132 8.66e-10 ***
## rated_num    6.291e-02  8.815e-02   0.714  0.47541
## duration     1.844e-08  4.564e-08   0.404  0.68613
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6468.4  on 4673  degrees of freedom
## Residual deviance: 6416.0  on 4666  degrees of freedom
## AIC: 6432
##
## Number of Fisher Scoring iterations: 4
```

Fijándonos en el *p-value*, observamos que las únicas variables relevantes en el modelo para predecir a un 95% de confianza son el número de turnos de la partida (*turns*) y la diferencia de ELO (*diff*). Pese a estar usando un dataset preparado con jugadores con poca diferencia de ELO, resulta curioso que la variable *diff* siga siendo tan determinante en la condición de victoria. La aparente poca contribución de las otras variables en este modelo podría explicarse debido a que todas ellas tienen cierta correlación con *turns* y *diff_num* y sus contribuciones podrían estar "incluidas" en ellas.

Excluiremos de nuestro modelo la contribución de *duration* por ser la que menos aporta:

```r
# Modelo de regresión logística

chess.logist <- glm(formula = winner_num ~ opening_ply + turns + time +
                      increment + diff_num + rated_num,
                data = chess.prediction, family=binomial(link=logit))

summary(chess.logist)
```

```
##
## Call:
## glm(formula = winner_num ~ opening_ply + turns + time + increment +
##     diff_num + rated_num, family = binomial(link = logit), data = chess.prediction)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.465  -1.186   1.004   1.131   1.520
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.1250985  0.1160150   1.078  0.28090
## opening_ply  0.0155098  0.0103278   1.502  0.13316
## turns       -0.0028988  0.0008832  -3.282  0.00103 **
## time         0.0026333  0.0032363   0.814  0.41582
## increment   -0.0026675  0.0030778  -0.867  0.38611
## diff_num     0.1824004  0.0297625   6.129 8.87e-10 ***
## rated_num    0.0633212  0.0881390   0.718  0.47250
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6468.4  on 4673  degrees of freedom
## Residual deviance: 6416.2  on 4667  degrees of freedom
## AIC: 6430.2
##
## Number of Fisher Scoring iterations: 4
```

A modo de prueba, veremos qué probabilidad tiene de ganar el jugador blanco en caso de que una partida competitiva dure 50 turnos (de los cuales 25 son de apertura) con un tiempo establecido inicial de 180 minutos con 0 segundos de incremento y una diferencia ELO de 0.

```r
# Predicción

x <- predict( chess.logist, data.frame(opening_ply=25, turns=50,
                                       time=180, increment=0,
                                       diff_num=0, rated_num=1),
              type = "response")


print(paste("Es esperable al",
            round( x * 100, 2),
            "% que gane el jugador blanco."))
```

```
## [1] "Es esperable al 71.2 % que gane el jugador blanco."
```

Veremos ahora qué probabilidad tiene el jugador blanco de ganar en una partida amistosa de 80 turnos (de los cuales 2 son de apertura) con un tiempo establecido inicial de 5 minutos con 120 segundos de incremento y una diferencia ELO de 0.

```
# Predicción

x <- predict( chess.logist, data.frame(opening_ply=2, turns=80,
                                       time=5, increment=120,
                                       diff_num=0, rated_num=0),
              type = "response")


print(paste("Es esperable al",
            round( x * 100, 2),
            "% que gane el jugador blanco."))
```

```
## [1] "Es esperable al 40.55 % que gane el jugador blanco."
```

```
# Predecimos los ganadores del dataframe chess.prediction

y <- predict( chess.logist, chess.prediction, type = "response")

# Mostramos en una tabla la cantidad de ganadores predecidos y reales

tab <- matrix(c (length(chess.prediction$winner_num[chess.prediction$winner_num==1]),
                 length(which(y>0.5)),
                 length(chess.prediction$winner_num[chess.prediction$winner_num==0]),
                 length(which(y<=0.5))),
              ncol = 2 , byrow = TRUE )
colnames(tab) <- c ('Real', 'Predecido')
rownames(tab) <- c ('Blanco', 'Negro')
tab <- as.table(tab)
tab
```

```
##        Real Predecido
## Blanco 2451      2816
## Negro  2223      1858
```

Vemos que hay una ligera sobreestimación del 15% en nuestro modelo predictivo en lo que respecta a categorizar los jugadores blancos como posibles ganadores. En ambos casos observamos que hay más casos de partidas ganadas por parte del blanco.

Conclusiones

Los datos estudiados contemplan 20056 partidas de ajedrez recolectados de Lichess.org en el que todos los jugadores tienen una determinada puntuación ELO y disputan partidas tanto amistosas (la victoria no altera su puntuación) como competitivas (quien gana, sube su puntuación ELO, y quien pierde la baja).

Como habría sido esperable, aquellas partidas con gran diferencia de puntuación ELO habrían podido provocar alteraciones en el análisis, tales como la tendencia a jugar de manera “experimental” por parte de un

jugador muy superior al contrincante, o la tendencia al abandono por parte de un jugador de nivel muy inferior. Por ello, hemos decidido reducir el tamaño de nuestro dataset a tan solo 4942 registros con tal que no exista una diferencia superior a 100 puntos ELO entre jugadores. Esto implica que solo exista un 8% de covarianza entre la condición de victoria según la puntuación ELO de los jugadores; en comparación al ~30% de covarianza ofrecida por el dataset original. Por ello, nuestro estudio se ha centrado en las condiciones de victoria entre jugadores con poca diferencia de puntuación ELO y las conclusiones que se saquen de aquí no serán válidas en caso de comparar varios jugadores con gran cantidad de diferencia de ELO, pues entrarían en juego otro tipo de factores que no hemos considerado.

Con el dataset reducido, nos hemos dispuesto a inspeccionar los distintos comportamientos de las variables según diferentes criterios. En primer lugar, hemos comprobado que la distribución de diferencia de ELO es simétrica, implicando que nuestro conjunto de datos no está sesgado por una cantidad mayor de jugadores blancos o negros con una mayor puntuación ELO a la de su contrincante negro o blanco, respectivamente.

De manera general, mediante inspección visual hemos comprobado que hay más porcentaje de partidas totales en las que el jugador blanco se alza con la victoria. Observamos claramente esta tendencia en las partidas de modalidad rápida y clásica (ya sean amistosas o competitivas); el jugador negro únicamente cuenta con más victorias en las partidas relámpago amistosas. Esto podría deberse a que en las partidas relámpago la apertura no es tan importante al tratarse de jugadas que responden a un pensamiento rápido e improvisado, más que por pura estrategia a largo plazo, por lo que la ventaja del jugador blanco al ser quien empieza la partida podría no ser tan relevante en esta modalidad. También hemos observado que la mayor cantidad de finales de partida se deben a la rendición por parte de uno de los contrincantes, siendo el jugador negro el que más se rinde. En cuanto a la condición de victoria por jaque mate, el blanco también tiene superioridad, mientras que cuando se debe a agotamiento de tiempo, ambos rivales tienen unas tendencias similares.

En cuanto al análisis estadístico realizado y suponiendo distribución normal en la diferencia de puntuación ELO, comprobamos las covarianzas de las diferentes variables con *winner_num* a fin de determinar qué factores son más relevantes para lograr la condición de victoria. Habiendo creado un modelo logístico en el que se excluye la duración total de la partida (pues no aportaba nada al modelo y nos generaba ruido), obtenemos que cuanto mayor sea el valor de *opening_ply*, *time*, *diff_num* y *rated_num*, mayor probabilidad de que el jugador negro gane, mientras que valores altos de *turns* y *increment* beneficiarán al jugador negro.

Concluimos con que se puede apreciar una clara tendencia a que el jugador blanco gane las partidas, pues la ventaja que dispone al empezarlas puede condicionar el resto del juego; cuanto más largas sean la apertura y el tiempo inicial de partida, los blancos tienen mayores probabilidades de ganar, mientras que le perjudicará aquellas partidas con gran número de turnos y en las que el rival negro tenga más puntuación ELO.

Tabla de contribuciones al trabajo

```
t <- matrix(c ("Investigación previa", "CME",
               "Redacción de las respuestas", "CME",
               "Desarrollo del código", "CME"),
              ncol = 2 , byrow = TRUE )
colnames(t) <- c ('CONTRIBUCIONES', 'FIRMA')
rownames(t) <- c ("", "", "")
t <- as.table(t)
t
```

```
##  CONTRIBUCIONES              FIRMA
##  Investigación previa        CME
##  Redacción de las respuestas CME
##  Desarrollo del código       CME
```