Matemáticas y el juego del guiñote

Sergio Torralba Perdices

Trabajo de fin de grado en Matemáticas
Universidad de Zaragoza

Director del trabajo: Francisco Javier López Lorente
27 de junio de 2022

Prólogo

El juego del guiñote es un juego de cartas, típico de Aragón y sus alrededores, en el que participan 4 jugadores divididos en dos parejas cuyo objetivo es llegar a una puntuación de 101 puntos antes que la pareja rival. Existen referencias al guiñote en textos de Gustavo Adolfo Béquer y de Pío Baroja en los siglos XIX y XX, lo que nos hace pensar que es un juego tradicional cuyo conocimiento se ha transmitido de generación en generación hasta nuestros días.

Hasta la fecha, no se ha realizado ningún estudio como el de este documento sobre el juego del guiñote. Una de las razones principales es que no hay registros de las partidas que se juegan normalmente y menos aún de las cartas que reciben los jugadores. Pero gracias a la Asociación Aragonesa de Guiñote (Guiñarte) disponemos de un fichero que contiene un registro de las partidas que se jugaron los dos primeros meses de 2021 en su app.

En este Trabajo Fin de Grado utilizaremos las matemáticas para hacer un análisis del juego utilizando los datos cedidos. El archivo consta de más de 8 millones de registros. Cada registro corresponde a un jugador de una partida jugada en los meses de enero y febrero de 2021 en la app de Guiñarte; para formar el registro de una partida necesitamos los 4 registros de los jugadores. Partiendo de esta base de datos y utilizando Python y R, formaremos una nueva base depurada y propia con la que haremos el estudio. En esta base de datos no estará reflejada toda la información presente en los registros, sino que filtraremos toda la información que sea irrelevante para el estudio.

El trabajo se dividirá en tres partes. La primera parte es la propia creación de la base de datos explicada de forma constructiva y cronológica.

Una vez se tenga esta base de datos, usaremos por un lado el cálculo de probabilidades para ver la probabilidad de algunos de los sucesos más importantes dentro de este juego y compararemos esos resultados con los obtenidos en la muestra, tratando de explicar las posibles diferencias que se puedan dar.

Por otro lado, usaremos técnicas de regresión logística para ser capaces de predecir el resultado de la partida en función de las cartas de los jugadores y la calidad de estos. Con ello intentaremos responder a la pregunta de si el guiñote es un juego de azar o en cambio es más importante la destreza de los jugadores. Para medir esa destreza se realizarán a lo largo del trabajo dos sistemas de puntuación distintos. Además, se hará un análisis más detallado del modelo más completo que tengamos.

Summary

This document is an end-of-degree project of the bachelor degree in mathematics in the University of Zaragoza. This is the first mathematical study of the Guiñote and the general purpose of the project is to understand how the game works in terms of probability and statistics. It is the first project of this type done about this game because there is no data regarding the games that are played. It has been possible for us to do it because Guiñarte (Aragonese Asociation of Guiñote) has given us a file with the information of all the games that were played in January and February of 2021 in its app. So now we have an opportunity to do a mathematical study of the game.

Guiñote is a traditional game of the Spanish region Aragón and its surroundings. It is played by four people in teams of two with the goal of getting 101 points faster than the other pair. Since it is a Spanish game, the cards used are the Spanish cards. It is unknown when it was created but there are evidences that show us it was already played in the 19^{th} century.

The first part of the project is the creation of a data-base which the sofware R is able to work with. Starting with the data file that Guiñarte gave us we have created a new data-base using Python and R. This data-base is the item that we will be using for the rest of the project. Notice that we have chosen only the relevant information that was reflected in the original data so not only have we converted the original file into a .csv but we have filtered all the irrelevant data for the mathematical study.

It is also needed to develop a punctuation system because in the statistical part of the project we are going to need it. So we have done two ranking systems. The first one is based on the percentage of won games while the second one is based on the elo system used in games like chess. Our elo system is a version of the original because Guiñote is a 2v2 game instead of 1v1, so we have adapted the original version to our game by computing the elo using the average score of the pairs.

The second part is the probability part. Here we have done the computation of the probabilities, expectations and variances of the main events that can take place during a game. In this section we show and explain all the computations that we have done and we compare the results with those we can obtain from the data-base created in the first part of the project. It is also stated why there are some differences between what the data-base shows and what we have computed.

The third and most important part of the project is the statistical part. We have a brief introduction to the logistic regression models. These models are capable of explaining the behaviour of a binary variable (in our case it is the result of the game) using some explanatory variables.

So in this section our aim is, by using logistic regression techniques, to develop models that are capable of predicting the outcome of a game based on the events that have occured among it.

We consider two kinds of events:

- Players' score: each player of a game has its own score (elo and percentage of wins). It will measure tha quality of the players, the higher the score the better the player is.

- Events: we consider the events that have taken place in each particular game. These events are based on the cards that the players have recieved during the game, so it is a random fact.

So we have searched for a model that is able to predict the outcome of a match accurately based on those factors.

We have explained the whole process that we have made in order to create the final models and there is also an analysis of the complete model where we show the ROC curves and the residuals of the predictions. These residuals present a problem that we are not able to explain but we show how to solve it.

The last part is the conclusion of the project where we try to see what is the most important thing in order to win a match (luck or ability). The luck is represented in the models by the events that depend on the cards each player has been given and the ability is represented by the score each player has.

So, using the complete model, we try to see the efect that each of the events has in the outcome of the game and we try to see how much difference must be in the punctuation in order to make the same effect as the random event.

It is also possible to see all the Python programms that were made in the attachment A (Anexo A) and the R code used to get all the results from the data-base in the attachment B (Anexo B).

If you want to learn or understand the rules of the game, they are explained (in Spainsh) on [7].

Índice general

Capítulo 1

Introducción y objetivos

1.1. El guiñote

El guiñote es un juego de cartas propio de Aragón y algunas zonas limítrofes, como partes de Navarra, Soria, La Rioja, Cuenca, Guadalajara, Castellón o Tarragona. El juego tiene dos variantes, la de dos jugadores o la de cuatro. La primera se suele utilizar para enseñar a jugar porque es la más sencilla y sirve para aprender el valor de las cartas, los objetivos del juego y a contar las puntuaciones al final. La segunda versión es la principal; 4 jugadores distribuidos en dos parejas juegan entre ellos con el objetivo de conseguir 101 puntos antes que la pareja rival. Esta versión es mucho más difícil de dominar porque el jugador no solo tiene en cuenta las cartas de su mano, sino también las posibles cartas de su compañero y de sus rivales. Esta versión es mucho más practicada que la versión de dos jugadores porque una vez que vas aprendiendo a jugar la versión inicial se vuelve muy trivial. Para este juego de cartas se utiliza la baraja española tradicional compuesta por 40 cartas (sin ochos ni nueves).

1.1.1. Historia

El guiñote es un juego de cartas muy parecido a la brisca o la briscola italiana y aunque las primeras referencias a él en textos aparecen más tarde, se cree que su origen data de la expansión de la Corona de Aragón, allá por el siglo XIV. Hay referencias a este juego ya en el siglo XIX, donde Gutavo Adolfo Béquer lo cita en alguna de sus obras o donde aparece en sentencias del Tribunal Supremo que datan de 1871 y 1874 en las que el guiñote es el motivo de la riña. En el siglo XX, Pío Baroja también nombra el guiñote en su obra.

El juego ha sido siempre muy popular en la región y ha llegado hasta nuestros días gracias al interés de las familias por introducir a los jóvenes a un juego tan tradicional y arraigado.

1.1.2. El guiñote en la actualidad

El guiñote es un juego muy extendido por Aragón y es bastante usual que en los bares se esté jugando una partida de guiñote en alguna mesa, en las piscinas en verano o en las reuniones familiares. Además, en las fiestas populares es típico que se organice un torneo de guiñote por parejas.

Desde el 2015 existe una asociación, la Asociación Aragonesa de Guiñote o Guiñarte que se encarga de organizar torneos promocionales y actividades, dispone de una aplicación para móviles y tiene una página web dónde se puede aprender a jugar desde cero.

Además también dispone de un archivo con documentos relacionados con el guiñote, desde explicaciones y referencias en la literatura hasta pequeños cálculos probabilísticos que acercan las matemáticas a este juego.

Hasta la fecha no hay ningún estudio matemático del guiñote, más allá de algunos cálculos simples. Esto es debido a que no hay registros de las partidas que se juegan normalmente. Aunque ahora somos capaces de realizar el primer estudio del guiñote de este tipo gracias a que disponemos de un registro de las partidas que se han jugado en la app de Guiñarte.

1.1.3. Nociones básicas:

En el juego del guiñote cada jugador empieza con una mano de 6 cartas repartidas de tres en tres. Cuando todos los jugadores disponen de sus 6 cartas, la siguiente carta del mazo se descubre, se pone boca arriba y se pone el mazo encima de ella de forma que se pueda ver la carta descubierta. Cada tirada se compone de dos fases. Primero los jugadores (en un orden) deben tirar una de las cartas que tienen en la mano, después quien haya ganado esa tirada se queda las cartas en un mazo aparte. Después el jugador que ha ganado roba una carta del mazo central seguido por el de su derecha y así hasta que hayan robado los 4 jugadores. Este proceso se repite 4 veces hasta que el mazo desaparece, en ese momento se hacen 6 tiradas más sin robar cartas hasta que nadie tenga ninguna en la mano. Después se cuentan los puntos conseguidos en las tiradas ganadas.
Algunas expresiones clave:

- **Idas y vueltas** son las dos partes de la partida, si una pareja llega a 101 un puntos en la primera parte se termina, en cambio si aún nadie ha llegado se juega una partida de vueltas hasta que una de las dos parejas consigue llegar.
- **Buenas y malas:** de los 101 un puntos, se suelen llamar malas a los 50 primeros y buenas a los 50 últimos, por ejemplo, decir que hemos conseguido 12 buenas es que has tenido en total 62 puntos.
- **Triunfo** es el palo de cartas que va a mandar sobre el resto a lo largo de la partida y viene dado por la carta descubierta que se encuentra debajo del mazo central.
- **Cantar un veinte** es mostrar que tienes rey y sota del mismo palo sin ser ese palo triunfo. Se pueden cantar tras hacer baza y la pareja que lo haga se lleva 20 puntos extra.
- **Cantar las cuarenta** es decir que tienes rey y sota de triunfo y la pareja suma 40 puntos extra.
- **Las 10 últimas** son los 10 puntos extra que se lleva la pareja que ha ganado la última baza.
- **Puntuaciones:** El as vale 11 puntos, seguido por el tres que vale 10, el rey con 4, la sota con 3 y el caballo con 2.

El total de sucesos y las normas del juego se pueden consultar en [7].

1.2. Objetivos

Al ver por un lado que los jugadores son capaces de tener un estilo de juego o una forma de jugar ya que la decisión de qué cartas tirar es suya y en cambio por otro lado tener que robar cartas sin saber cuáles son nos lleva a una de las cuestiones principales del trabajo. Queremos saber qué es más importante a la hora de decidir el resultado de una partida, las cartas que te tocan (azar) o las cartas que decides jugar y los momentos en los que decides hacerlo (calidad).

Para la realización de este trabajo, Guiñarte nos cedió una base de datos. Esta base de datos corresponde a las partidas que se jugaron entre enero y febrero de 2021 en su app. Este archivo esta en formato "json", consta de más de 8 millones de registros, siendo cada registro un string en la que se reflejan los datos referidos a un jugador en una partida. Por ejemplo están reflejados los identificadores de partida y jugador, si ha tenido el as o el tres de triunfo, cuántos veintes ha tenido, si ha tenido las cuarenta, si ha ganado la partida o la cantidad de monedas virtuales que ha ganado o perdido. Se pueden consultar

varios ejemplos de registros en el anexo.

El sofware estadístico R no es capaz de trabajar con un archivo en este formato, por lo tanto otro de los objetivos es ser capaz de crear una base de datos con la que R pueda trabajar.

Una vez que se tenga la base de datos utilizaremos R para obtener nuestros resultados y conclusiones.

- En una parte probabilística, haremos los cálculos de los sucesos más importantes que se pueden dar a lo largo de una partida y compararemos estos resultados con aquellos obtenidos con nuestra base de datos.
- En una parte estadística, hablaremos de la regresión logística y haremos una serie de modelos que sean capaces de predecir el resultado de una partida. Estos modelos tendrán en cuenta, aparte de los distintos sucesos que han ocurrido a lo largo de las partidas, la puntuación de los jugadores que participan en la partida; estas puntuaciones serán creadas en función del tiempo, del resultado de la partida y de las partidas que ha jugado el participante hasta la fecha.

1.3. Estructura del documento

Capítulo 2: Desarrollo de la base de datos
En esta primera parte del trabajo se explicará el desarrollo de la base de datos que ha sido utilizada en el resto del trabajo. A lo largo del capítulo se explicará de forma ordenada y cronológica la manera de desarrollar la base de datos. Además, utilizando ya la base de datos obtenida, se realizará un análisis descriptivo de los datos recogidos que se comparará más adelante con los resultados probabilísticos obtenidos.

Capítulo 3: Probabilidades y primeros resultados.
En esta sección se explicarán brevemente los conceptos combinatorios que se han utilizado para los cálculos de las probabilidades, se expondrán los cálculos probabilísticos de algunos de los sucesos más importantes en una partida y se compararán con lo obtenido en el análisis descriptivo.

Capítulo 4: Modelos de regresión
Este capítulo constará de una introducción a la regresión logística en la que se explicarán los conceptos básicos para comprender cómo se va a desarrollar en nuestro caso y cuáles son los resultados que buscamos. Usando los datos obtenidos en el capítulo 2, se realizarán y compararán una serie de modelos de regresión logística que sean capaces de predecir quién va a ganar.

Capítulo 5: Conclusión
En esta parte final del trabajo se responderá a la pregunta de qué es más importante si el azar o la calidad de los jugadores. El azar queda reflejado en las cartas que reciben los jugadores a lo largo de la partida, mientras que la calidad se refleja en las puntuaciones que tienen.

Capítulo 2

Base de datos

En este capítulo estructuraremos y explicaremos la creación de la base de datos utilizada a lo largo del trabajo. Los programas de Python utilizados en esta capítulo se pueden consultar en el Anexo A.

2.1. Depuración

La base de datos cedida por Guiñarte es un fichero "json"que consta de 8.406.399 de registros. Cada registro se refiere a un jugador en una partida y en él se reflejan todos los datos correspondientes a la partida del jugador. Por lo tanto, cada partida está reflejada en 4 registros, uno para cada participante. Es importante destacar que no todos los registros tienen el mismo número de campos, lo que dificulta el estudio. De los datos reflejados no nos interesan todos, por eso el primer paso es quitar todo lo que no nos sirve para el estudio matemático y tener los registros en forma de matriz. Se pueden consultar cómo son los registros en el Anexo A donde hay unos cuantos registros originales.

Lo primero es ver si el jugador ha sufrido alguna desconexión a lo largo de la partida. Del total de registros, tenemos un total de 326.517 en los que el jugador se ha desconectado. Estos registros se han de desechar porque, tras mantener una reunión con el presidente y el informático de Guiñarte, se nos dijo que cuando un jugador se desconecta una máquina releva al jugador y por lo tanto no nos sirve esta partida porque el comportamiento no es humano.

Una vez eliminados estos registros, procedemos a crear un fichero en el cada fila corresponda a un registro original y las columnas sean los datos que nos interesan para el estudio estadístico.

- La primera columna es el identificador de la partida, que será el dato que utilizaremos para conectar los 4 registros de una partida.
- La segunda columna será el identificador del jugador, en este momento se desechan 8 registros por no tener el identificador un formato apto para nuestra matriz.
- En la tercera columna está el resultado de la partida, es decir, si el jugador ha ganado (1) o no (0). Aquí nos encontramos con 47.918 registros desechados por no tener resultado.
- En la cuarta columna tenemos reflejado el número de veintes que ha tenido el jugador en las idas de la partida (de 0 a 3). Desechamos 164 registros por tener 4 veintes, este error se nos dijo que era porque el jugador se había desconectado a lo largo de la partida y luego había vuelto, lo que hacía que se contabilizaran los veintes varias veces.
- En la quinta columna se refleja si el jugador ha tenido (1) o no (0) las 40 de idas en la partida. Se desechan 5.108 registros por tener más de un cuarenta.
- En la sexta columna, si ha tenido (1) o no (0) el tres de triunfo de idas. Se desechan 17.020 registros por tener más de un tres de triunfo en la mano.

- En la séptima columna tenemos si el jugador ha tenido (1) o no (0) el as de triunfo en las idas. 16.092 registros son desechadas por tener varios ases de triunfo.
- Las columnas octava, novena, décima y undécima son análogas a las columnas cuarta, quinta, sexta y séptima respectivamente, pero en vez de referirse a las idas se refiere a las vueltas de la partida. El número de registros desechados son 32, 755, 3.756 y 4.894 respectivamente y por los mismo motivos.
- La duodécima columna de la matriz es la fecha en la que se jugó la partida, como todas las partidas son de 2021, tenemos la primera cifra que es el mes (1 o 2 dependiendo de si es de enero o de febrero, la hora, los minutos, segundos...). Nos hemos preocupado especialmente de que tengan las mismas cifras para así ser capaces en un futuro de ordenar las partidas por fecha simplemente ordenando de menor a mayor.
- La última columna corresponde al número de registro que se refleja en la fila, esta información se ha utilizado para poder localizar los registros que a lo largo de los siguientes apartados nos hayan podido dar algún tipo de problema.

Pese a tener datos que reflejan sucesos ocurridos en las vueltas de la partida, no disponemos de información suficiente para ser capaces de trabajar con ellas. Por ejemplo, no sabemos cuándo hay o no vueltas en una partida, ni tampoco las puntuaciones con las que se acaba la parte de idas. Esta falta de información hace que a la hora de hacer el estudio no tengamos en cuenta los sucesos ocurridos en las vueltas.

2.2. Ordenación

A continuación, hemos abierto la base de datos creada en R para ordenar las partidas por identificador de partida. El objetivo de este paso es tener las partidas con el mismo identificador juntas, de este modo será más fácil y rápido juntar los cuatro registros que correspondan a la misma partida en uno solo.

Además de esto, utilizando las funciones unique y match de R renombramos a los jugadores para que los números sean más pequeños y manejables ya que los identificadores de jugador obtenidos en el paso anterior eran números de demasiadas cifras.

2.3. De registros a partidas

Volviendo a Python, ahora nuestro objetivo es tener una matriz en la que cada fila se corresponda a una partida, de este modo obtendremos la base de datos principal del trabajo. Cabe decir que la app escoge a los jugadores de forma aleatoria, es decir, no hay parejas fijas.

Para ello, lo primero es hacer un programa que al principio descarte aquellas partidas de las que no disponemos de todos los registros. Puede suceder que haya menos de cuatro registros ya que alguno de ellos haya sido desechado en la depuración del primer conjunto de datos. Para la nueva base de datos tendremos las siguientes columnas reflejando la información de una partida completa:

- La primera columna es el identificador de partida que comparten los 4 registros que vamos a juntar.
- Las siguientes 4 columnas corresponden a los identificadores de los jugadores que participan en la partida. El programa esta hecho de modo que el jugador del primer registro que coge se coloca en la segunda columna, su compañero de pareja en la tercera y los otros dos en las dos siguientes. De este modo tendremos jugador 1, 2, 3 y 4.
- Las columnas sexta y séptima muestran cuál ha sido el jugador que ha tenido el tres y el as de triunfo de idas respectivamente.

- Las siguientes 4 columnas van referidas al número de veintes de idas que tienen los jugadores 1, 2, 3 y 4 respectivamente.
- La duodécima columna refleja el jugador que ha tenido las cuarenta. En caso de que no las haya tenido ningún jugador hay un 0.
- Las columnas decimotercera y decimocuarta son análogas a sexta y la séptima respectivamente pero reflejan el tres y as de triunfo de vueltas, por lo tanto, en estas columnas es posible que haya ceros porque pueden no haber salido cuando la partida termina.
- Las 4 columnas que reflejan el número de veintes que ha tenido cada jugador de vueltas son análogas a las 4 de las idas, al igual que la columna correspondiente a las 40 de vueltas.
- La columna 20 tiene la fecha a la que se jugó la partida. Como hay 4 fechas distintas, siendo cada una la fecha concreta en la que el jugador entró a la partida con minutos segundos, décimas, centésimas, etc; se escoge como fecha de la partida la del primer registro de los 4 que se están manejando a la vez.
- Las siguientes 4 columnas son los números de los registros de los que se ha obtenido la fila. De este modo, se podrían localizar entre los más de 8 millones de registros iniciales los 4 que corresponden a la partida.
- Por último, la columna del final tiene un 1 si ha ganado la pareja 1 (la de los dos primeros jugadores) o un 2 si ha ganado la segunda pareja.

Al seguir trabajando con esta base de datos nos dimos cuenta de que había unas pocas partidas en las que se cantaban 4 o 5 veintes, cuando como mucho debería haber 3. Esto se debe al mismo error generado por la desconexión y conexión de alguno de los jugadores. Por ejemplo se detectaron partidas en las que tres jugadores tenían veintes pero uno de ellos tenía dos y por eso había pasado el filtro que se hizo en la parte de depuración, pues el jugador no superaba el número máximo de veintes pero en el global de la partida si que sucede esto, así que solucionamos ese error modificando el código de Python.

Una vez eliminadas estas partidas llegamos a la base de datos final formada por 1.742.897 partidas que será utilizada para hacer un análisis descriptivo y para comparar los cálculos realizados en el siguiente capítulo. Pero de cara a los modelos de regresión logística necesitaremos más cosas.

2.4. Rankings

Como queremos saber hasta qué punto la habilidad de un jugador influye en el resultado de un partida, tenemos que tener una puntación para cada jugador en cada partida que juega. El siguiente paso, y de cara a hacer los modelos de regresión logística, es programar un sistema de puntuación para poder evaluar la calidad de los jugadores ya que esta información no estaba en la base de datos que se nos cedió. En nuestro caso hemos programado dos rankings distintos. Para programarlos se ha ordenado primero las partidas por fecha mediante la función arrange de R. El primero es un ranking con porcentajes de partidas ganadas a lo largo del tiempo, aunque en Python no lo hemos programado, hemos añadido a lo largo del programa las columnas con el número de partidas jugadas y ganadas de cada jugador. Así que desde R será sencillo hallar los porcentajes. El segundo es un ranking elo, como los del ajedrez, adaptado a un juego de parejas.

2.4.1. Ranking Elo

El Elo es el método que se utiliza en muchos juegos, siendo el ajedrez uno de los más importantes, para dar una puntuación a sus jugadores en función de las victorias obtenidas y de la puntuación de los jugadores a los que se han enfrentado. En él se tiene en cuenta la puntuación de los jugadores que se enfrentan en una partida y se halla la nueva puntuación de cada uno en función del resultado de la partida

y del resultado esperado de la partida. El resultado esperado de la partida es una predicción donde la diferencia de puntuaciones entre los jugadores el predictor. Fue elaborado por Arpad Elo, un ajedrecista y profesor de física que vivió en el siglo XX.

Para nuestro juego, tenemos que usar una versión adaptada del Elo porque no tenemos jugadores sino parejas pero cada jugador tiene su Elo individual. Lo haremos de forma cronológica, es decir, actualizando la puntuación de los participantes en una partida al final de la misma, basándonos en [6]:

- La puntuación inicial de todos los jugadores será de 1200 puntos.
- Antes de cada partida, se hace el promedio de las puntuaciones de los jugadores de ambas parejas, P_1 y P_2.
- Calcularemos el resultado esperado para cada pareja E_1 y E_2. Dado por las siguientes fórmulas:

$$E_1 = \frac{1}{1+10^{\frac{P_2-P_1}{400}}} \quad E_2 = \frac{1}{1+10^{\frac{P_1-P_2}{400}}}$$

- R_1 y R_2 será el resultado obtenido. Un 1 para la pareja que gane y un 0 para la que pierda.
- Las nuevas puntuaciones de cada pareja serán:

$$P_1' = P_1 + 32 \cdot (R_1 - E_1) \quad P_2' = P_2 + 32 \cdot (R_2 - E_2)$$

- Por último, a la puntuación individual de los jugadores que componen la pareja 1 se les sumará $\frac{P_1'-P_1}{2}$ y a los de la pareja 2 se les sumará $\frac{P_2'-P_2}{2}$ dando lugar a las puntuaciones que tendrán los jugadores al terminar la partida.

2.4.2. Implementación en la matriz de datos

Añadiremos las nuevas columnas a la matriz utilizando Python. Creamos una nueva matriz donde las primeras columnas sean las mismas que ya teníamos y las siguientes sean el número de partidas que ha jugado, ha ganado y la puntuación Elo del jugador 1; análogamente con los otros tres jugadores.

2.4.3. Preparativos para los modelos

Como los rankings presentan una gran variabilidad cuando los jugadores no han jugado las suficientes partidas, hemos decidido no usar para estimar los modelos las que haya jugadores que han jugado poco. Entendemos que un jugador no ha jugado lo suficiente para pasar el corte si ha jugado menos de 200 partidas, porque con 200 partidas hemos comprobado que el porcentaje de partidas ganadas es estable.

En la Tabla 2.1 se muestra un resumen del número de datos que hemos ido manejando a lo largo del proceso de creación de la base.

Registros iniciales	8.406.399
Registros filtrados	7.984.135
Partidas obtenidas	1.743.222
Partidas finales	1.742.897
Partidas para modelos	739.801

Tabla 2.1: Número de registros y partidas

2.5. Análisis descriptivo de los datos

En esta sección expondremos los resultados que hemos podido extraer de forma descriptiva de la base de datos. Notar que en todo momento se hablará de los sucesos en la parte de idas donde se juegan las 40 cartas de la baraja.

Las cuarenta:

Las cuarenta se cantan en un total de 394.717 partidas, lo que supone un 22.65 % de las partidas. Además en las partidas que se cantan las 40, la pareja que las tiene gana un 87.55 % de las veces. Lo que muestra lo beneficioso que es tenerlas para ganar una partida.

Los veintes:

Puede haber entre 0 y 3 veintes a lo largo de una partida de idas y de media se cantan 0.55 veintes por partida, es decir, algo más de un veinte cada dos partidas. Además el número de veintes tiene una desviación típica de aproximadamente 0.68. En la tabla se puede observar en cuántas partidas hay 1, 2 y 3 veintes.

Tres y as de triunfo:

Un 61.39 % de las partidas las gana la pareja que tiene el tres, mientras que la pareja que tiene el as gana el 65.96 % de las partidas. El hecho de que casi el doble de veces gane la pareja que tiene el as nos hace pensar en su gran importancia de cara al resultado de la partida aunque a priori no parezca tan importante.

Además, en un 48.85 % de las partidas el tres y el as les toca a una misma pareja. En estas partidas, la pareja que tiene tres y as gana el 78.03 % de las veces. Si diferenciamos entre que el as y el tres los tenga el mismo jugador o que un jugador de la pareja tenga el as y otro el tres observamos:

- En un 23.26 % aproximadamente del total de las partidas el as y el tres de triunfo los tiene el mismo jugador, en este caso un 79.27 % de las partidas se decantan por la pareja con el jugador con as y tres.
- En un 25.59 % aproximadamente de las partidas el tres y el as los tienen jugadores distintos de la misma pareja. En estos casos la victoria se la lleva la pareja con el tres y el as un 76.9 % de las veces.

Los porcentajes de victorias son muy parecidos aunque la diferencia nos hace pensar que es algo más fácil ganar si es el mismo jugador quien tiene as y tres.

Suceso	Número de partidas	Porcentajes aproximados
Las cuarenta	394.717	22.65
1 veinte	628.607	36.06
2 veintes	145.398	8.34
3 veintes	12.661	0.72
As y tres pareja	851.376	48.85
As y tres jugador	405.344	23.26
As y tres pareja dist. jug.	446.032	25.59

Tabla 2.2: Número de sucesos y porcentajes

Capítulo 3

Probabilidades y primeros resultados

Nuestra intención en este capítulo es calcular las probabilidades, esperanzas y varianzas de los sucesos y variables aleatorias que pueden decantar una partida, como pueden ser cantar veintes, las cuarenta o tener el as y el tres.

A lo largo del capítulo entenderemos como una mano las 10 cartas que un jugador tiene a lo largo de una partida y diremos que tendrá un 20 o las 40 a lo largo de la partida si entre esas 10 cartas le coinciden sota y rey del mismo palo. Después compararemos los resultados obtenidos con el análisis descriptivo del capítulo anterior.

Conceptos clave:

- Cálculo de las probabilidades: Estamos tratando con un juego de cartas, por lo tanto todas las manos son equiprobables. Si queremos calcular la probabilidad de un suceso S se puede hacer de la siguiente manera:
$$P(S) = \frac{\text{casos favorables}}{\text{casos posibles}}$$
- El número de subconjuntos de r elementos que se pueden hacer en un conjunto de n elementos viene dado por el número combinatorio $\binom{n}{r}$. En nuestro caso, las manos son subconjuntos de 10 elementos elegidos entre 40 cartas. Es decir, el número de manos diferentes que puede recibir un jugador es:
$$\binom{40}{10} = \frac{40!}{10! \cdot (40-10)!} = \text{ 847.660.528 manos posibles}$$

Mediante estos dos conceptos vamos a realizar los cálculos de las probabilidades de los sucesos más importantes

3.1. Cálculos con los veintes

Para los veintes calcularemos la esperanza del número de veintes por partida, la varianza y las probabilidades de que haya 1, 2 y 3 veintes.

Sin pérdida de generalidad, establecemos que el triunfo es bastos. Sea

$$X_{ij} = \begin{cases} 1 \text{ si el jugador i tiene el 20 j} \\ 0 \text{ en los demás casos} \end{cases} \tag{3.1}$$

Con $i = 1,2,3$ y 4 y $j = 1,2$ y 3 que se corresponden con oros, copas y espadas respectivamente. Empezamos por la esperanza del número de veintes, notar que:

$$\text{número de veintes} = \sum_{i=1}^{4} \sum_{j=1}^{3} X_{ij}$$

$$S = \text{ se coge el triunfo de muestra del final}$$

Luego:

$$E(\text{número de veintes}) = E(\sum_{i}^{4}\sum_{j}^{3} X_{ij}) = \sum_{i}^{4}\sum_{j}^{3} E(X_{ij}) = 12 \cdot E(X_{ij}) =$$
$$12 \cdot P(X_{ij}=1) = 12 \cdot (P(X_{ij}=1|S) \cdot P(S) + P(X_{ij}=1|S^c) \cdot P(S^c)) =$$
$$12 \cdot (\frac{\binom{37}{7}}{\binom{39}{9}}\frac{1}{4} + \frac{\binom{37}{8}}{\binom{39}{10}}\frac{3}{4}) = 12 \cdot \frac{3}{52} = \frac{9}{13}$$

Por lo tanto se esperan alrededor de 0.69 veintes por partida. Si lo comparamos con los 0.55 veintes que se cantan por partida vemos una diferencia bastante significativa.

Esto se debe a que solo se tienen 6 cartas en la mano a la vez y es posible que se tire la sota o el rey antes que de que te venga la otra. Por lo tanto es de esperar que el número real de veintes que se cantan sea menor que el que se podrían tener.

Notar que para hacer el cálculo de la esperanza, se condiciona a que se coja o no el triunfo de muestra que está al fondo del mazo. En caso de que se coja tendremos 39 cartas disponibles y 9 restantes en nuestra mano y en caso de que no se coja serán 39 cartas disponibles y 10 en nuestra mano.

Ahora es el turno de la varianza:

$$Var(\text{número de veintes}) = Var(\sum_{i}^{4}\sum_{j}^{3} X_{ij}) = \sum_{i}^{4}\sum_{j}^{3} Var(X_{ij}) + \text{Término de las covarianzas}$$

Por un lado, calculamos la varianza:

$$Var(X_{ij}) = E(X_{ij}^2) - E(X_{ij})^2 = \frac{3}{52} - (\frac{3}{52})^2 = \frac{147}{2704}$$

Para calcular el término de las covarianzas hay que calcular las covarianzas en función de i y j:

$$Cov(X_{ij}, X_{i'j'}) = E(X_{ij}, X_{i'j'}) - E(X_{ij})E(X_{i'j'}) = P(X_{ij}=1, X_{i'j'}=1) - (\frac{3}{52})^2 =$$
$$\frac{1}{4}\frac{\binom{35}{7}}{\binom{39}{9}}\frac{\binom{28}{8}}{\binom{30}{10}} + \frac{3}{4}(\frac{\binom{35}{8}}{\binom{39}{10}} \cdot (\frac{1}{3}\frac{\binom{27}{7}}{\binom{29}{9}} + \frac{2}{3}\frac{\binom{27}{8}}{\binom{29}{9}})) - (\frac{3}{52})^2 = \frac{135}{36556} - (\frac{3}{52})^2 =$$
$$= 0{,}00036456185$$

$$Cov(X_{ij}, X_{ij'}) = E(X_{ij}, X_{ij'}) - E(X_{ij})E(X_{ij'}) = P(X_{ij}=1, X_{ij'}=1) - (\frac{3}{52})^2 =$$
$$\frac{1}{4}\frac{\binom{35}{5}}{\binom{39}{9}} + \frac{3}{4}\frac{\binom{35}{6}}{\binom{39}{10}} - (\frac{3}{52})^2 = \frac{21}{9139} - (\frac{3}{52})^2 = -0{,}00103055796$$

$$Cov(X_{ij}, X_{i'j}) = E(X_{ij}, X_{i'j}) - E(X_{ij})E(X_{i'j}) = P(X_{ij}=1, X_{i'j}=1) - (\frac{3}{52})^2 =$$
$$0 - (\frac{3}{52})^2 = \frac{-9}{2704}$$

Con las covarianzas ya calculadas, hay que tener en cuenta que para cada uno de los 12 X_{ij} hay 6 $X_{i'j'}$, 3 $X_{i'j}$ y 2 $X_{ij'}$. Por lo tanto el término de las covarianzas es:

$$\text{Término de las covarianzas} = 12 \cdot (6Cov(X_{ij}, X_{i'j'}) + 2Cov(X_{ij}, X_{ij'}) + 3Cov(X_{ij}, X_{i'j})) = -0{,}118307423$$

Luego volviendo a la varianza tenemos:

$$Var(\text{número de veintes}) = Var(\sum_{i}^{4}\sum_{j}^{3} X_{ij}) = 12 \cdot Var(X_{ij}) + \text{Términos de las covarianzas} = 12 \cdot \frac{147}{2704} - 0{,}118307423 = 0{,}5340594409$$

Para hallar la probabilidad de que haya 1, 2, 3 o ningún veinte a lo largo de la partida (P_1, P_2, P_3 y P_0 respectivamente) vamos a hacerlo planteando el siguiente sistema:

$$\begin{aligned} &P_0 + P_1 + P_2 + P_3 = 1 \\ &P_1 + 2 \cdot P_2 + 3 \cdot P_3 = \tfrac{9}{13} \\ &P_1 + 4 \cdot P_2 + 9 \cdot P_3 = 0{,}5340594409 + (\tfrac{9}{13})^2 \end{aligned} \tag{3.2}$$

Tenemos un sistema de 3 ecuaciones con 4 incógnitas, pero P_3 no es difícil de calcular, para ello vamos a diferenciar tres casos distintos: que un jugador tenga 3 veintes P_{33}, que un jugador tenga 2 y otro 1 P_{32} y que haya tres jugadores con un 20 P_{31} y razonaremos de la misma forma, condicionando a que se haya cogido o no la última carta del mazo.

$$P_{33} = 4P(\text{el jugador i tenga 3 veintes}) = 4(\frac{1}{4}\frac{\binom{33}{3}}{\binom{39}{9}} + \frac{3}{4}\frac{\binom{33}{4}}{\binom{39}{10}}) = \frac{2}{9139}$$

$$P_{32} = 4 \cdot 3P(\text{2veintes jugador i, 1 veinte jugador j}) = 12(\frac{1}{4}\frac{\binom{33}{5}}{\binom{39}{9}}\frac{\binom{28}{8}}{\binom{30}{10}} + \frac{3}{4}(\frac{\binom{33}{6}}{\binom{39}{10}} \cdot (\frac{1}{3}\frac{\binom{27}{7}}{\binom{29}{9}} + \frac{2}{3}\frac{\binom{27}{8}}{\binom{29}{10}})) = \frac{18}{9139}$$

$$P_{31} = 4P(\text{1 veinte para jugadores i,j k}) = 4(\frac{1}{4}\frac{\binom{33}{7}}{\binom{39}{9}}\frac{\binom{26}{8}}{\binom{30}{10}}\frac{\binom{18}{8}}{\binom{20}{10}} + \frac{3}{4}(\frac{\binom{33}{8}}{\binom{39}{10}} \cdot (\frac{1}{3}\frac{\binom{25}{7}}{\binom{29}{9}}\frac{\binom{18}{8}}{\binom{20}{10}} + \frac{2}{3}\frac{\binom{25}{8}}{\binom{29}{10}} \cdot (\frac{1}{2}\frac{\binom{17}{7}}{\binom{19}{9}} + \frac{1}{2}\frac{\binom{17}{8}}{\binom{19}{10}}))) = = 0{,}001055$$

Luego P_3 = $\frac{2}{9139}$+$\frac{18}{9139}$ + 0.001055 = 0.00324.

Ahora resolviendo el sistema (3.2) obtenemos que $P_0 = \frac{302119}{650000}$, $P_1 = \frac{91967}{162500}$ y $P_2 = \frac{97907}{650000}$.

Resumimos estas probabilidades en la siguiente tabla:

Número de veintes	**Porcentaje calculado aprox.**	**Porcentaje observado**
1 veinte	56.60	36.06
2 veintes	15.06	8.34
3 veintes	0.324	0.72

Tabla 3.1: Comparativa de veintes

Como era de esperar vemos que se cantan menos veintes de los que deberían porque la manera normal de jugar hace que se prefiera descartar algunos de los palos de la baraja en vez de mantener el rey o la sota. Pero vemos una clara discrepancia, hay más del doble de partidas de las que debería con 3 veintes cantados. Esto se debe a que dentro de estas partidas hay algunas en las que se ha contado más veces de las que se debería alguno de los veintes. Esto se nos dijo que era por la posible desconexión y conexión de el jugador con el 20. Es el mismo error que nos aparecía en la depuración de datos, pudimos eliminar los registros con más de 3 veintes y las partidas que en conjunto tenían más de 3 veintes. Pero en las partidas con menos de tres veintes no es posible diferenciar las que están bien y las que han contabilizado algún veinte más del que deberían. Pero sabemos que hay partidas en las que se sigue produciendo el error.

3.2. Cálculos con las 40

A la hora de hacer los cálculos tenemos en cuenta que las 40 involucran dos cartas, rey y sota de triunfo. Si una de estas dos cartas es el triunfo de muestra que está al final del mazo, las cuarenta las cantará el jugador que tenga el rey/sota (dependiendo cuál de las dos sea el triunfo de muestra) y el 7. Por lo tanto razonaremos sin condicionar a que se coja o no el triunfo de muestra porque de una mano de 10 cartas tendremos 2 que serán las 40, independientemente de que sea rey y sota o que el 7 esté involucrado.

Queremos calcular la probabilidad de que haya un jugador que tenga las cuarenta en su mano de toda la partida (P_c).

$$X_i = \begin{cases} 1 \text{ si el jugador i tiene las cuarenta} \\ 0 \text{ si el jugador i no las tiene} \end{cases} \tag{3.3}$$

Luego tenemos:

$$P_c = E(\sum_i^4 X_i) = \sum_i^4 P(X_i = 1) = 4 \cdot P(X_i = 1) =$$
$$4 \cdot \frac{2}{40} \cdot \frac{1}{39} \cdot \binom{10}{8} = \frac{3}{13} = 0{,}2308$$

Además llamando $X = \sum_{i=1}^4 X_i$ tenemos que X es el indicador de que algún jugador ha tenido las cuarenta en la mano, es decir, una variable Bernoulli con probabilidad $\frac{3}{13}$, por lo tanto tiene una esperanza de $\frac{3}{13}$ y una varianza de $\frac{30}{169}$.

Entonces podemos afirmar que las cuarenta estarán en la mano de algún jugador un 23,08 % de las partidas. Comparando esto con los datos de análisis descriptivo, vemos que las cuarenta se cantan en el 22,65 % de las partidas. Esta ligera diferencia se debe a que no siempre que se tienen las cuarenta se pueden cantar, porque pueden arrastrar y de ese modo hacer que tires el rey o la sota sin haberlas cantado.

Podemos calcular los intervalos de confianza para la probabilidad de las cuarenta con la siguiente fórmula.

Sea $\hat{P}$ la probabilidad de un suceso obtenida a partir de la muestra y n el tamaño de la muestra:

$$IC = \left[\hat{P} - z_{\frac{\alpha}{2}} \sqrt{\frac{\hat{P}(1-\hat{P})}{n}}, \hat{P} + z_{\frac{\alpha}{2}} \sqrt{\frac{\hat{P}(1-\hat{P})}{n}} \right] \tag{3.4}$$

Para un intervalo de confianza al 95 % tenemos que $z_{\frac{\alpha}{2}} = 1{,}96$, además $\hat{P} = 0{,}2265$ y $n = 1742897$ quedando el intervalo de confianza $[0{,}2259, 0{,}2271]$. Observamos que $\frac{3}{13}$ está bastante por encima del intervalo, lo que es normal ya que puedes tener las cuarenta en la mano y no cantarlas.

3.3. Cálculos con as y tres de triunfo

Podemos intercambiar el papel del rey y la sota por el del as y el tres y tenemos el mismo razonamiento que en el apartado anterior, por lo tanto la probabilidad de que algún jugador tenga el as y el tres en la mano es $\frac{3}{13}$. La esperanza es $\frac{3}{13}$ y la varianza $\frac{30}{169}$.

Comparando esto con lo obtenido en análisis descriptivo tenemos que que un jugador tenga as y tres en una misma partida sucede un 23,25 % de las partidas.

Podemos hacer los mismos cálculos para ver la probabilidad de que una pareja tenga as y tres en la misma partida.

$$Y_i = \begin{cases} 1 \text{ si la pareja i tiene el as y el tres de triunfo} \\ 0 \text{ si la pareja i no los tiene} \end{cases} \tag{3.5}$$

Por lo tanto:

$$P(\text{alguna pareja tenga as y tres de triunfo}) = E(\sum_{i=1}^{2} Y_i) = \sum_{i=1}^{2} P(Y_i = 1) = 2 \cdot P(Y_i = 1) =$$
$$2 \cdot \frac{2}{40} \cdot \frac{1}{39} \cdot \binom{20}{18} = \frac{19}{39} = 0{,}4872$$

Igual que en el apartado anterior $Y = \sum_{i=1}^{2} Y_i$ es una Bernoulli con $p = \frac{19}{39}$, por lo tanto, con esperanza $\frac{19}{39}$ y varianza $\frac{380}{1521}$.

Además en la base de datos tenemos que en un 48,85 % de las partidas el as y el tres los tiene la misma pareja.

Si calculamos los intervalos de confianza de la misma manera que en el apartado anterior tenemos:

- Para as y tres en el mismo jugador tenemos que $\hat{P} = 0{,}2326$, por lo tanto el intervalo de confianza al 95 % tenemos que es: $[0{,}23257, 0{,}2332]$. Nuestro P calculado, que es $\frac{3}{13} = 0{,}23077$, se queda fuera de él.
- Para as y tres en la misma pareja tenemos que $\hat{P} = 0{,}4885$ y el intervalo de confianza nos queda $[0{,}4877, 0{,}4892]$. Siendo $P = \frac{9}{39} = 0{,}4872$ vemos que se queda un poco por debajo del intervalo.

Que ambas probabilidades se queden fuera del intervalo de confianza se puede deber a la gran cantidad de datos con la que trabajamos y a que haya algún tipo de sesgo a la hora de de generar aleatoriedad o en la forma de afrontar las desconexiones de los jugadores. En todo caso, debe notarse que los verdaderos valores de la probabilidad se quedan fuera del intervalo por muy poco y, de hecho, si se hicieran los intervalos al 99 % de confianza quedarían dentro.

Capítulo 4

Modelos de regresión

4.1. Notas sobre regresión logística

La regresión logística es un tipo de análisis de regresión cuya finalidad es predecir el resultado de una variable dicotómica en función de varias variables explicativas. Estas variables explicativas pueden ser cualitativas o cuantitativas. ([1] [4])

La variable que se pretende predecir será de tipo Bernoulli:

$$Y = \begin{cases} 1 \text{ si el suceso se produce} \\ 0 \text{ si el suceso no se produce} \end{cases}$$

El objetivo de la regresión logística es estimar P(Y=1), es decir, la probabilidad de que el suceso ocurra. Introduzcamos dos conceptos previos a la regresión logística:

El **Odds** de un suceso S viene dado por la razón de la probabilidad de que ocurra el suceso entre la probabilidad de que no ocurra.

$$O(S) = \frac{P(S)}{1 - P(S)}$$

El Odds indica cuánto más probable es que ocurra el suceso S a que no ocurra.

Además también podemos expresar la probabilidad de que ocurra un suceso en función del Odds:

$$P(S) = \frac{O(S)}{1 + O(S)}$$

Sean S un suceso y A y B los dos valores que puede tomar una variable, el **Odds ratio** del suceso S es la razón de los Odds de S condicionado a A y de S condicionado a B.

$$OR = \frac{O(S|A)}{O(S|B)} = \frac{\frac{P(S|A)}{1-P(S|A)}}{\frac{P(S|B)}{1-P(S|B)}}$$

El Odds ratio se interpreta de la siguiente forma:

- Si es 1, quiere decir que no hay relación entre el suceso S y la variable que puede tomar los valores A y B
- Si es mayor que 1 indica que la probabilidad de que el suceso S ocurra es mayor bajo la condición A que bajo la B
- Si es menor que 1 indica que la probabilidad de que el suceso S ocurra es mayor bajo la condición B que bajo la A

La variable Y puede tomar los valores 0 y 1, por lo tanto no tiene sentido aplicar regresión lineal porque no podemos aproximar por una recta ya que el valor que estamos buscando es una probabilidad, es decir, está en el intervalo (0,1) y el resultado de la regresión lineal no tiene por qué estar acotado.

Para solucionar este problema se hace la transformación logística, que consiste en aplicar el logaritmo al odds de la probabilidad que estamos buscando y usaremos una combinación lineal de las variables explicativas para obtener una aproximación de este valor.

Sea $P = P(Y = 1|\mathbf{X})$ y $\mathbf{X} = X_1, X_2, ..., X_n$ las variables explicativas, entonces el modelo de regresión lineal tendrá la siguiente fórmula

$$\ln \frac{P}{1-P} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n \tag{4.1}$$

β_i son los parámetros desconocidos del modelo. Entonces podemos escribir la ecuación del modelo de regresión logística de la siguiente manera:

$$P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n)}} \tag{4.2}$$

4.2. Modelos

En nuestro caso particular tendremos:

- El suceso Y será que gane la pareja 1, es decir:

$$Y = \begin{cases} 1 \text{ si la pareja 1 gana} \\ 0 \text{ si la pareja 1 pierde} \end{cases}$$

- Las variables que corresponden al as (*aspareja1*) y al tres (*trespareja1*) serán unas variables binarias que valdrán 1 si la pareja 1 ha tenido el as/tres o 0 si no lo ha tenido.

- La variable que representa el efecto de las cuarenta (*cuaren*) será :

$$cuaren = \begin{cases} 1 \text{ si la pareja 1 canta las cuarenta} \\ 0 \text{ si no aparecen las cuarenta} \\ -1 \text{ si la pareja 2 canta las cuarenta} \end{cases}$$

- Para tener en cuenta el efecto de los veintes, tenemos una variable que será la diferencia de veintes cantados.

- Para tener en cuenta las puntuaciones de los jugadores tenemos la diferencia entre el promedio de Elo de las parejas, el mínimo y el máximo de cada pareja y análogo con el sistema de puntación basado en los porcentajes.

Diremos que un modelo da como ganador a la pareja 1 en una partida si al tener en cuenta los sucesos que han ocurrido en la partida y trasladarlos al modelo, nos da una probabilidad $P > 0{,}5$. Con esta información veremos qué modelos aciertan más y cuáles son los mejores a priori. En el Anexo C se pueden consultar todas las salidas de los modelos abajo expuestos

4.2.1. Modelos con sucesos

Nuestro modelo inicial y a partir del que iremos construyendo los demás utiliza como variable explicativa la variable que refleja quién tiene el as de triunfo, la carta más fuerte y valiosa de la partida. .

Modelo inicial (as)

```
rankale2 = read.csv(file="rankale2f.csv", head=TRUE,sep=",")
GLM.1 <- glm(resbin ~ aspareja1, family=binomial(logit), data=rankale2)
```

$$\ln \frac{P}{1-P} = -0{,}66 + 1{,}32 \cdot \text{aspareja1}$$

En este caso, el modelo acierta un 66 % de las veces solamente teniendo en cuenta si la pareja1 tiene el as o no.

Modelo con as y tres

De forma constructiva añadimos la variable del tres al modelo y nos queda el segundo modelo:

```
GLM.2<- glm(resbin ~ aspareja1+trespareja1, family=binomial(logit), data=rankale2)
```

$$\ln \frac{P}{1-P} = -1{,}27 + 1{,}45 \cdot \text{aspareja1} + 1{,}09 \cdot \text{trespareja1}$$

Este segundo modelo también acierta un 66 % de las veces.

Modelo con as, tres y veintes

```
GLM.20<- glm(resbin ~ aspareja1+trespareja1+difveintes,
family=binomial(logit), data=rankale2)
```

$$\ln \frac{P}{1-P} = -1{,}30 + 1{,}48 \cdot \text{aspareja1} + 1{,}12 \cdot \text{trespareja1} + 0{,}35 \cdot \text{difveintes}$$

Con este modelo acertamos un 66,8 % de las veces.

Modelo con as, tres, veintes y cuarenta

```
GLM.3<- glm(resbin ~ aspareja1+trespareja1+difveintes+cuaren,
family=binomial(logit), data=rankale2)
```

$$\ln \frac{P}{1-P} = -1{,}64 + 1{,}85 \cdot \text{aspareja1} + 1{,}42 \cdot \text{trespareja1} + 0{,}5 \cdot \text{difveintes} + 2{,}52 \cdot \text{cuaren}$$

Al añadir las cuarenta a los modelos, el salto en aciertos es notable ya que acertamos un 73,75 %.

4.2.2. Modelos con sucesos y puntaciones

Lo primero que se hizo fue comparar los dos sistemas de puntuación para ver qué impacto tenían en los modelos. Los modelos que se compararon fueron hechos con el as y uno de los dos sistemas de puntuación:

- El sistema basado en porcentajes de partidas ganadas daba lugar a modelos que acertaban algo menos de un 66 % de las veces. Lo que nos hace compararlos con el modelo inicial con tan solo el as ya que tenía el mismo porcentaje de acierto. Para corroborarlo, calculamos los estadísticos de Nagelkerke y Cox-Snell para el modelo inicial y para este modelo con el as y los porcentajes de partidas ganadas de todos los jugadores y observamos que son muy parecidos y que el añadir los porcentajes de partidas ganadas no mejoraba el modelo de forma notable, así que los descartamos.

- Introducir las puntuaciones Elo sí que aportaba información a los modelos y éstos mejoraban, así que éste es el sistema de puntuación que hemos utilizado a la hora de hacer los modelos.

  Tuvimos dos acercamientos distintos:

  - Con las medias: esta forma tiene en cuenta la puntuación promedio de los jugadores de cada pareja. Esto se puede expresar como la diferencia de las puntaciones medias de las parejas ya que el coeficiente de la puntuación media de la pareja 1 es el mismo que el de la pareja 2 pero de signo contrario.
  - Con los máximos y los mínimos de las puntuaciones: el objetivo de este acercamiento era ver si los coeficientes de los máximos o de los mínimos eran más grandes o más pequeños, es decir, se quería comprobar si penalizaba tener un integrante de la pareja malo o si beneficiaba mucho tener a alguien muy bueno dentro de la pareja. Tras ver que los coeficientes de los máximos y los mínimos eran prácticamente iguales, se llegó a la conclusión de que no merecía la pena diferenciar entre el máximo y el mínimo de las puntuaciones, porque tenían el mismo efecto dentro del modelo.

Por lo tanto, la variable que represente el efecto de las puntuaciones de las parejas en los modelos será una variable cuantitativa que refleje la diferencia de las puntuaciones medias de las parejas. En caso de querer ver los modelos que se hicieron con otras variables o con otros sistemas de puntuación se puede consultar el Anexo B. Se hicieron también modelos que tenían en cuenta el número de partidas jugadas y otras variables que resultaron ser no significativas, así que se descartaron.

Para la construcción de estos modelos empezaremos con un modelo que tiene en cuenta la diferencia de puntuaciones e iremos añadiendo variables de la misma forma que hemos hecho en 4.2.1.

Modelo con diferencia de Elo

```
GLM.elo<-glm(resbin ~ difelopar, family=binomial(logit), data=rankale2)
```

$$\ln \frac{P}{1-P} = 0{,}000722 + 0{,}0114 \cdot \text{difelopar}$$

Tenemos que acierta el 62,31 %.

Modelo con as y diferencia de Elo

```
GLM.4<- glm(resbin ~ aspareja1+difelopar, family=binomial(logit), data=rankale2)
```

$$\ln \frac{P}{1-P} = -0{,}66 + 1{,}33 \cdot \text{aspareja1} + 0{,}0114 \cdot \text{difelopar}$$

Con este modelo, conseguimos acertar un 68,16 % de las partidas. Esto es más que el modelo que teníamos con as, tres y veintes.

Modelo con as, tres y diferencia de Elo

```
GLM.6<- glm(resbin ~ aspareja1+ trespareja1 + difelopar,

family=binomial(logit), data=rankale2)
```

$$\ln \frac{P}{1-P} = -1{,}27 + 1{,}45 \cdot \text{aspareja1} + 1{,}10 \cdot \text{trespareja1} + 0{,}0115 \cdot \text{difelopar}$$

70,35 % de las partidas son predichas de manera correcta por este modelo.

Modelo con as, tres, veintes y diferencia de Elo

```
GLM.7<- glm(resbin ~ aspareja1+ trespareja1 +difveintes + difelopar,
family=binomial(logit), data=rankale2)
```

$$\ln \frac{P}{1-P} = -1{,}31 + 1{,}49 \cdot \text{aspareja1} + 1{,}13 \cdot \text{trespareja1} + 0{,}35 \cdot \text{difveintes} + 0{,}0115 \cdot \text{difelopar}$$

Este modelo mejora al anterior, pues acierta en un 70,78 % de las partidas.

Modelo completo: as, tres, veintes, cuarenta y diferencia de Elo

```
GLM.8<- glm(resbin ~ aspareja1+ trespareja1 +difveintes + cuaren + difelopar,
family=binomial(logit), data=rankale2)
```

$$\ln \frac{P}{1-P} = -1{,}65 + 1{,}86 \cdot \text{aspareja1} + 1{,}43 \cdot \text{trespareja1} + 0{,}51 \cdot \text{difveintes} + 2{,}55 \cdot \text{cuaren} + 0{,}0118 \cdot \text{difelopar}$$

Este es el modelo más completo y también el que más porcentaje de aciertos tiene, con éxito a la hora de predecir el resultado de un 76,58 %. Al observar los coeficientes vemos que tener las cuarenta es lo más importante para ganar, seguido del as y el tres. El coeficiente de la diferencia de puntuaciones no se puede comparar directamente porque la variable tiene unidades distintas. Esta comparación se realiza en el capítulo 5.

4.3. Análisis del modelo completo

Como hemos visto en el apartado anterior, el modelo completo tiene el cuenta los cuatro sucesos más importantes de la partida (qué pareja tiene el as y el tres, cuántos veintes ha cantado cada pareja y si alguien ha cantado las 40) y la diferencia de las puntuaciones medias de la pareja, es decir, la calidad de las parejas.

En esta sección analizaremos la calidad del modelo usando herramientas estadísticas, veremos la curva ROC generada por él y el área debajo de esta curva, veremos también si los errores a la hora de predecir tienen o no una forma o una distribución concreta.

Recordemos que del apartado anterior hemos obtenido que el modelo presenta la siguiente expresión:

$$\ln \frac{P}{1-P} = -1{,}65 + 1{,}86 \cdot \text{aspareja1} + 1{,}43 \cdot \text{trespareja1} + 0{,}51 \cdot \text{difveintes} + 2{,}55 \cdot \text{cuaren} + 0{,}0118 \cdot \text{difelopar} \tag{4.3}$$

A partir de esta expresión podemos despejar la probabilidad P que nos da el modelo en función de las variables explicativas:

$$P = \frac{1}{1 + e^{1{,}65 - 1{,}86 \cdot \text{aspareja1} - 1{,}43 \cdot \text{trespareja1} - 0{,}51 \cdot \text{difveintes} - 2{,}55 \cdot \text{cuaren} - 0{,}0118 \cdot \text{difelopar}}} \tag{4.4}$$

Y como hemos dicho con anterioridad, prediremos una victoria para la pareja 1 cuando esta P resulte ser mayor que 0.5.

4.3.1. Curva ROC

La curva ROC es una herramienta matemática que nos permite valorar la capacidad de explicar la variable dependiente que tiene un modelo. Un modelo se considera mejor y con más capacidad de expresar la variable dependiente binaria cuanto mayor sea el área bajo la curva ROC. Vamos a ver una tabla con el área bajo la curva ROC de los modelos del apartado anterior. En algunos modelos hay el mismo porcentaje de aciertos pese a haber más variables, pero el área bajo la curva ROC es diferente.

Modelos	Área bajo la curva ROC
GLM.1	0.6597
GLM.2	0.7186
GLM.20	0.7291
GLM.3	0.8594
GLM.elo	0.6705
GLM.4	0.7379
GLM.6	0.7713
GLM.7	0.7783
GLM.8	0.8621

Tabla 4.1: Comparativa de áreas bajo la curva

4.3.2. Curva ROC del modelo completo

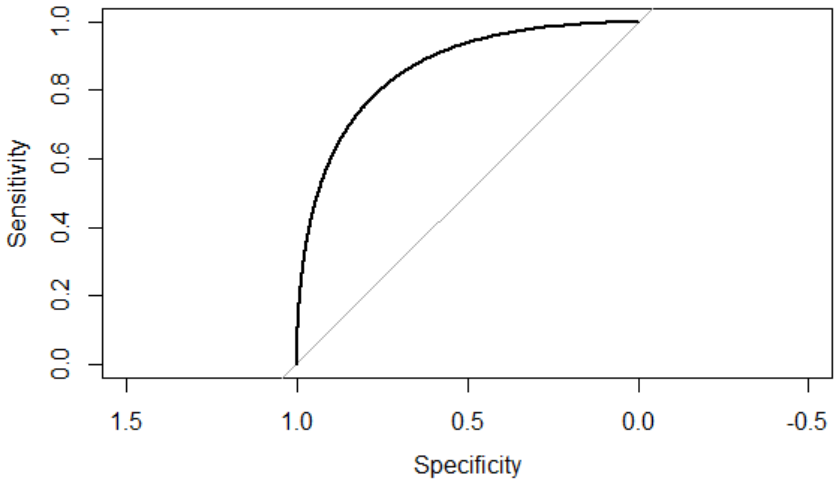

Figura 4.1: Curva ROC generada por el modelo completo

Además obtenemos que el área bajo la curva es de 0.8621. Teniendo en cuenta que un modelo teóricamente perfecto tendría área 1 bajo la curva, este modelo es bastante bueno a priori.

4.3.3. Análisis de los residuos

Ahora vamos a ver qué ocurre con los residuos generados por el modelo. Por un lado, tenemos los valores esperados de P que nos da el modelo de regresión logística y por otro lado tenemos los residuos generados, es decir, la diferencia entre el valor observado y el valor obtenido. En un buen modelo de regresión logística, estos residuos no deben tener ninguna distribución ni deben seguir un patrón. Para

observar cómo se comportan los residuos, dividimos la muestra en grupos y en cada uno de ellos vemos la media de los residuos.

Para poder observar los residuos existe la función `binnedplot` en R que al implementarla a nuestro modelo obtenemos lo siguiente:

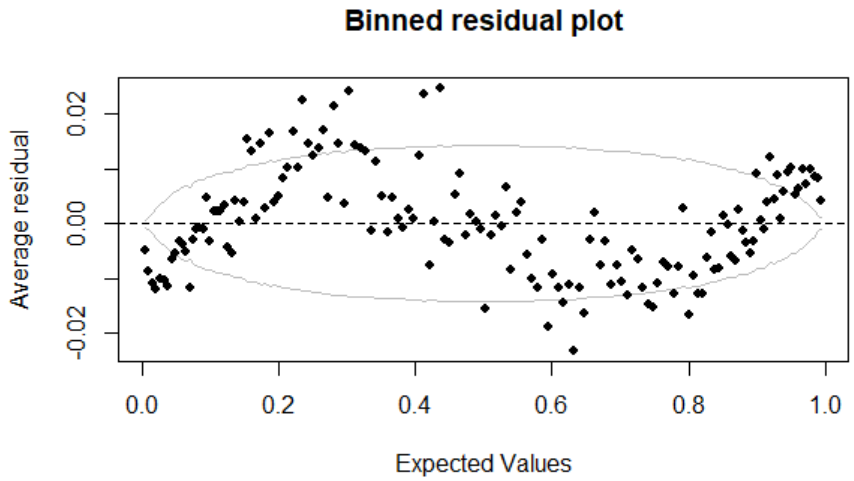

Figura 4.2: Medias de los residuos (150 bins)

Observamos una forma definida en lugar de tener una distribución aleatoria. Por lo tanto, este modelo no es del todo bueno. Para solucionar este problema, vemos si le ocurre lo mismo al modelo que no tiene en cuenta las cuarenta:

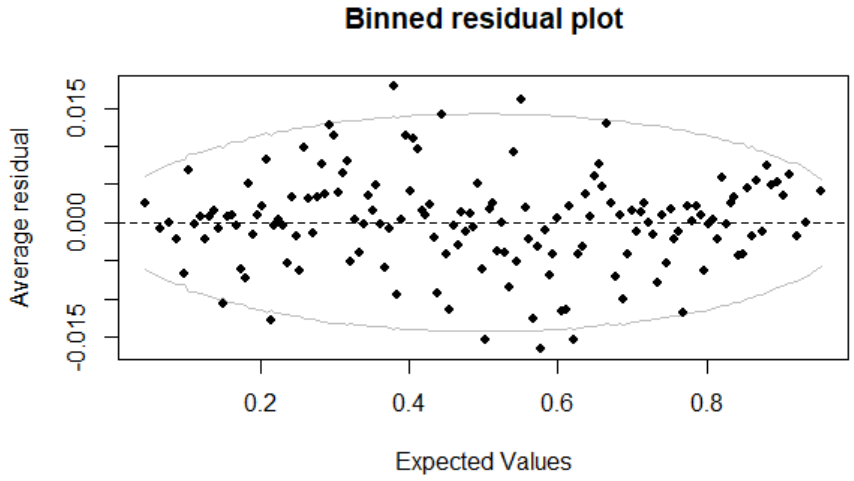

Figura 4.3: Medias de los residuos modelo sin 40

Observamos que para este modelo los residuos no tienen una distribución concreta, por lo tanto el problema lo generan las cuarenta, aunque no hemos averiguado la razón por la que sucede esto. Para solucionar este problema, vamos a hacer a partir del modelo completo dos modelos distintos, uno para partidas sin las cuarenta y otro para partidas con cuarenta.

Modelo con as, tres, veintes y diferencia de Elo, para partidas sin 40

```
rankalesin40 = read.csv(file="rankalesin40.csv", head=TRUE,sep=",")
GLM.401<- glm(resbin ~ aspareja1+ trespareja1 +difveintes + difelopar,
family=binomial(logit), data=rankalesin40)
```

$$\ln \frac{P}{1-P} = -1{,}58 + 1{,}79 \cdot \text{aspareja1} + 1{,}30 \cdot \text{trespareja1} + 0{,}49 \cdot \text{difveintes} + 0{,}0117 \cdot \text{difelopar}$$

Dentro de las partidas en las que no se cantan las cuarenta, el modelo acierta un 73,08 % de ellas. Además, bajo la curva ROC hay un área de 0.8067.

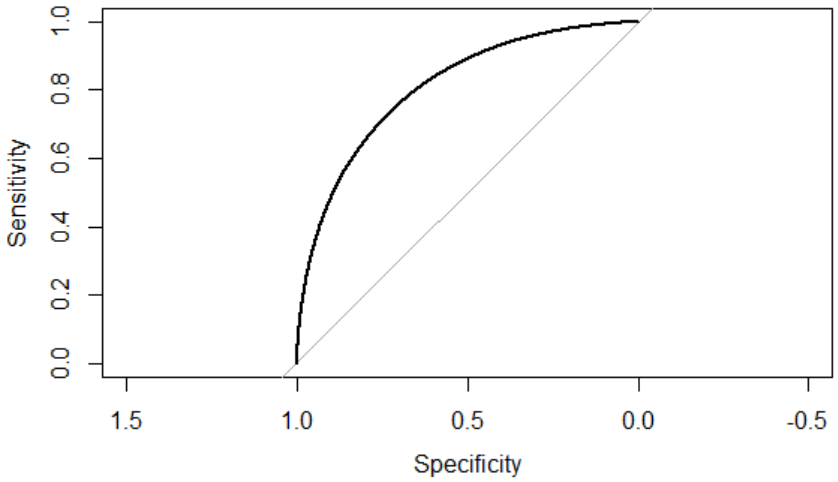

Figura 4.4: Curva ROC del modelo para las partidas sin 40

Modelo con as, tres, veintes, cuarenta y diferencia de Elo, para partidas con las 40

```
rankale40 = read.csv(file="rankale40.csv", head=TRUE,sep=",")
GLM.402<- glm(resbin ~ aspareja1 + trespareja1 + difveintes + cuaren + difelopar,
family=binomial(logit), data=rankale40)
```

$$\ln \frac{P}{1-P} = -2{,}24 + 2{,}53 \cdot \text{aspareja1} + 1{,}97 \cdot \text{trespareja1} + 0{,}61 \cdot \text{difveintes} + 2{,}91 \cdot \text{cuaren} + 0{,}012 \cdot \text{difelopar}$$

En este caso, todas las variables son significativas y el modelo es exitoso prediciendo en el 88,70 % de los casos.

Ahora veamos la curva ROC del modelo, el área que guarda bajo la curva y si este modelo sigue presentando algún tipo de distribución en la media de los residuos.

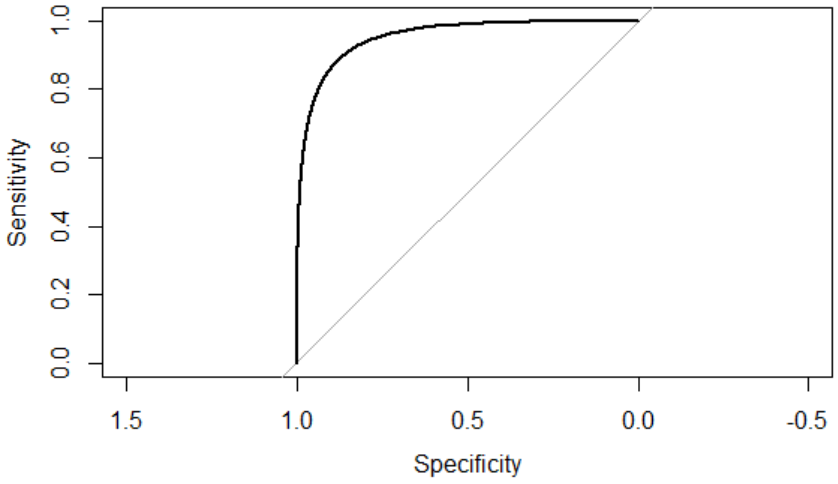

Figura 4.5: Curva ROC del modelo completo en solo las partidas con las 40

Presenta una área bajo la curva de 0.9535, siendo 1 el área de un modelo perfecto, estamos ante un modelo teóricamente muy bueno.

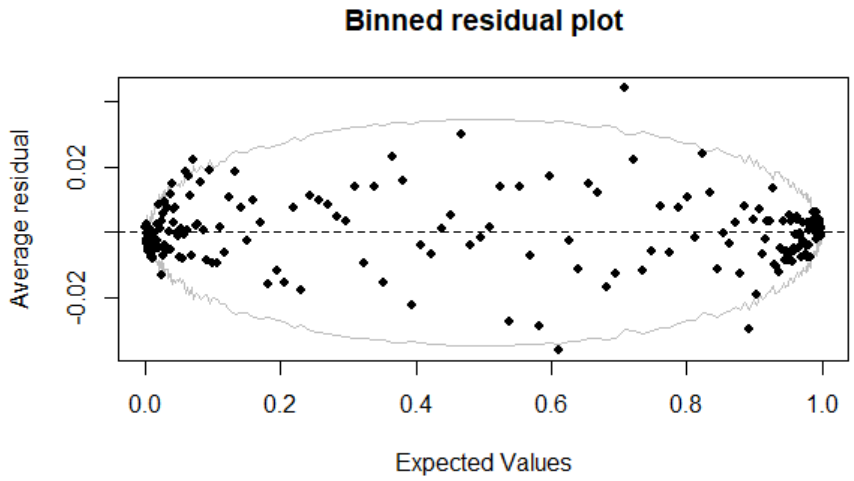

Figura 4.6: Medias de los residuos del modelo completo en solo partidas con 40

Como podemos observar, no hay ninguna forma definida en este caso, por lo tanto el problema que nos presentaba este modelo en el conjunto de partidas inicial se soluciona cuando el conjunto de partidas se reduce a las partidas en las que se han cantado las cuarenta. Además, podemos observar que en los dos extremos hay muchos puntos, esto se debe a que el modelo predice una P cercana a 1 cuando la pareja 1 tiene las cuarenta y cercana a 0 cuando no las tiene, por lo tanto es esperable que presente muchos puntos en los extremos.

Cabe destacar que a diferencia de calidad entre los jugadores afecta de la misma manera en ambos modelos (coeficientes casi iguales).

Capítulo 5

Conclusión

Ahora es el momento de abordar la cuestión más importante del trabajo. Tenemos modelos capaces de predecir el resultado de una partida de una forma bastante razonable, pero aún no hemos dicho nada de qué es más importante a la hora de decantar el resultado de una partida si el azar o la calidad de los participantes.

- **El azar** se refleja en las cartas que un jugador recibe, no es algo que el jugador pueda controlar y no hay ninguna herramienta para controlar las cartas que le tocan a cada jugador quitando el 7 de triunfo, que se puede cambiar por el triunfo de muestra en algunos casos.

- **La calidad de los jugadores** se mide por la puntuación que tienen y depende de las victorias que los jugadores hayan obtenido a lo largo del tiempo y de los contrincantes a los que se han enfrentado.

Ya hemos visto, en el análisis descriptivo de los datos, que la pareja que tiene el as gana un 65,96 % de las veces, es decir, casi el doble de victorias se consiguen teniendo el as que sin él. Lo que nos hace pensar en la importancia de las cartas, pero debemos indagar más en el tema.

Consideramos la base de datos con la que se han hecho los modelos, en la que no hay jugadores que estén aprendiendo a jugar, ni jugadores principiantes que pese a saber jugar no lo hacen de manera consistente y cometen muchos errores. De esta forma nos podemos fijar en los modelos para intentar resolver esta pregunta.

Nos fijamos en el modelo completo que pese a tener el problema de los residuos lo consideramos suficientemente bueno. Notar que no se puede comparar el efecto del as o del tres con el efecto de la diferencia de las puntuaciones de la pareja directamente comparando los coeficientes que tienen en el modelo porque las variables no son comparables. La variable del tres o del as es un indicador, mientras que que la variable de la diferencia de Elo es una variable cuantitativa.

Por lo tanto, para comparar y analizar qué es más importante, vamos a ver cuanto aporta al log(Odds) en la ecuación (4.3) que la pareja 1 tenga por separado: el as, tres, un veinte de ventaja y las cuarenta; y calcularemos cómo debe ser la diferencia de elo entre las parejas para igualar ese efecto.

- Que la pareja 1 tenga el as aporta al log(Odds) 1.86. Para igualar esto tendría que haber una diferencia de elo de 157.63 puntos. Esta diferencia de elo corresponde al percentil 99.65, es decir, solo en un 0,35 % de las partidas se puede igualar el efecto del as.

- El efecto que produce que el tres lo tenga la pareja 1 es de 1,43, que se iguala con una diferencia de elo de 121.19, que es el percentil 98.22. Luego el efecto del tres se puede igualar en un 1,78 % de las partidas.

- Un veinte de ventaja tiene un efecto de 0.51 en el modelo, que se puede igualar teniendo una diferencia de puntuación de 43.22 puntos, que se corresponde con el percentil 77.53, luego se puede igualar el efecto en más de 1 de cada 5 partidas.
- El suceso que en la práctica es más determinante es las cuarenta, en los modelos queda reflejado aportando 2.55 al log(Odds). La diferencia necesaria es de 216.1 puntos, que corresponde al percentil 99.976, lo que nos lleva a decir que solo se puede igualar esta diferencia en el 0,024 % de las partidas.

Suceso	Aportación al log(Odds)	Diferencia necesaria	Porcentaje de partidas donde se da
as	1.86	157.63	0.35
tres	1.43	121.19	1.78
un veinte	0.51	43.22	22.47
cuarenta	2.55	216.1	0.024

Tabla 5.1: Análisis de aportaciones y diferencias

Por lo tanto, vemos que en la mayoría de partidas entre jugadores que no son principiantes, las cartas generan un efecto que no se puede igualar por calidad. Por lo tanto podemos afirmar que es mucho más importante tener buenas cartas que ser muy buen jugador o ser una pareja muy fuerte porque el efecto que esa diferencia de calidad puede generar a lo largo de la partida es fácilmente compensado con las cartas que se pueden tener.

Utilizando el modelo que solo tiene en cuenta el Elo de las parejas, vamos a ver cómo afecta esa diferencia de puntuaciones al resultado de una partida y al resultado de un coto (un coto son 5 partidas al mejor de 3).

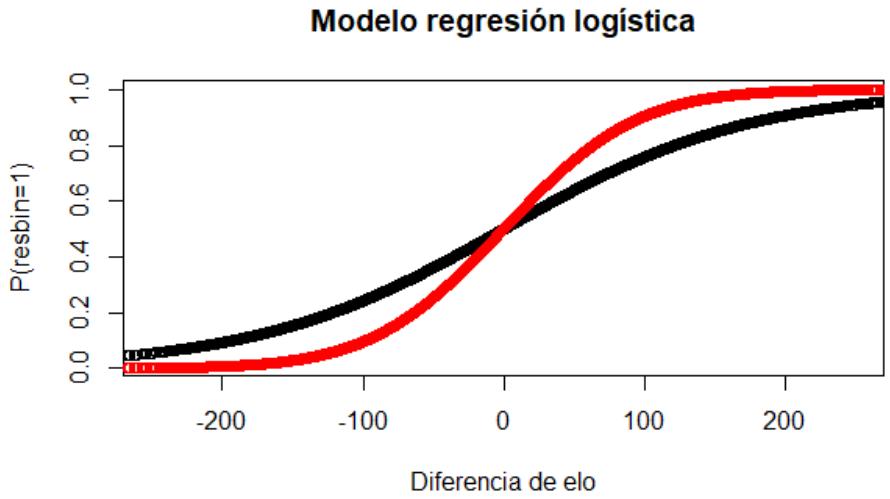

Figura 5.1: Probabilidades de ganar partida y coto en función de la diferencia de Elo.

Si bien en una partida la diferencia de Elo no se influye mucho en el resultado, al pasar al coto vemos que la curva es mucho más pronunciada, lo que indica que el efecto es mayor.

Bibliografía

[1] SIMON J. SHEATHER *A modern approach to regression with R*, 2009

[2] N. H. BINGHAM, JOHN M. FRY, *Regression: Linear Models in Statistics*, 2010

[3] PAUL ROBACK AND JULIE LEGLER, *Beyond Multiple Linear Regression Applied Generalized Linear Models and Multilevel Models in R*, 26/1/2021.

[4] TANIA IGLESIAS CABO, *Mariano Peralta Horte*, *Métodos de Bondad de Ajuste en Regresión Logística*, Trabajo Fin de Master, Universidad de Granada, 2013. https://masteres.ugr.es/moea/pages/tfm-1213/tfm_iglesiascabo_tania/!

[5] RUBÉN FERNÁNDEZ-CASAL, JAVIER ROCA-PARDIÑAS, JULIÁN COSTA Y MANUEL OVIEDO, *Introducción al Análisis de Datos con R*, 2022-05-19 https://rubenfcasal.github.io/intror/.

[6] NEBITRAMS, *Individual Ranking for Doubles Game*, 9 de junio 2017 https://elosportschallenge.wordpress.com/2017/06/09/individual-ranking-for-doubles-game/.

[7] ASOCIACIÓN ARAGONESA DE GUIÑOTE, GUIÑARTE https://www.xn--guiarte-6za.es/

[8] MARCOS NAVARRO *El guiñote, un juego de naipes con arraigo pero poco documentado*, Diario de Teruel, 1/02/2021. https://www.diariodeteruel.es/cultura/el-guinote-un-juego-de-naipes\-con-gran-arraigo-pero-poco-documentado

ANEXO A

June 22, 2022

Registros y programas de python

0.1 Ejemplos de algunos registros:

{“_id”:{“oid” : ”5fefa8f411009544edf4c54c”,”numero_partida” : 0,”provider_full” : ”15210216525688839389 – Facebook”,”sala_id” : ”RO – 73938.63425813447”,”IP” : ” :: ffff : 127.0.0.1”,”v”:0,”amigo”:false,”asi”:1,”desconectado”:false,”fecha”:”date”:“2021-01-01T23:00:32.373Z”},“_fecha_trunc”:{“$date”:“2021-01-01T23:00:00Z”},“_id_partida”:“19379.701294196773”,“_gbits_diario”:0,“_gbits_resultado”:0,“_resultado”:2}

{“_id”:{“oid” : ”5fefa8bf11009544edf4b3e0”,”numero_partida” : 4,”provider_full” : ”10311301047429747420789 2 – Google”,”sala_id” : ”RO – 81679.46254689217”,”IP” : ” :: ffff : 127.0.0.1”,”v”:0,”amigo”:false,”desconectado”:false,”fecha”:”date”:“2021-01-01T23:00:31.733Z”},“_fecha_trunc”:{“$date”:“2021-01-01T23:00:00Z”},“_id_partida”:“80898.76997973508”,“_veinte_i”:1,“_as_i”:1,“_tres_v”:1,“_gbits_diario”:0,“_gbits_resultado”:50,“_resultado”:1}

{“_id”:{“oid” : ”5fefa8f711009544edf4c646”,”numero_partida” : 1,”provider_full” : ”1511611853052296777 – Facebook”,”sala_id” : ”RO – 93847.02847321196”,”IP” : ” :: ffff : 127.0.0.1”,”v”:0,”amigo”:false,”desconectado”:false,”fecha”:”date”:“2021-01-01T23:00:16.388Z”},“_fecha_trunc”:{“$date”:“2021-01-01T23:00:00Z”},“_id_partida”:“34127.801012597”,“_veinte_i”:1,“_gbits_diario”:0,“_gbits_resultado”:0,“_resultado”:2}

{“_id”:{“oid” : ”5fefa8f911009544edf4c6e8”,”numero_partida” : 4,”provider_full” : ”1561966957716959082 – Facebook”,”sala_id” : ”RO – 43826.335528410906”,”IP” : ” :: ffff : 127.0.0.1”,”v”:0,”amigo”:false,”desconectado”:false,”fecha”:”date”:“2021-01-01T23:00:18.745Z”},“_fecha_trunc”:{“$date”:“2021-01-01T23:00:00Z”},“_id_partida”:“38040.81437215021”,“_veinte_i”:1,“_as_i”:1,“_gbits_diario”:0,“_gbits_resultado”:50,“_resultado”:1}

{“_id”:{“oid” : ”5fefa8fc11009544edf4c7f6”,”numero_partida” : 0,”provider_full” : ”19210225948770369351006 8 – Google”,”sala_id” : ”RO – 38405.104559136904”,”IP” : ” :: ffff : 127.0.0.1”,”v”:0,”amigo”:false,”asi”:1,”desconectado”:false,”fecha”:”date”:“2021-01-01T23:00:10.189Z”},“_fecha_trunc”:{“$date”:“2021-01-01T23:00:00Z”},“_id_partida”:“18096.764101365025’’,“_gbits_diario”:0,“_gbits_resultado”:0,“_resultado”:1}

0.2 Programas de python

Código para pasar de los registros iniciales a la matriz de registros

```
In [ ]: import numpy as np
        import json
        from scipy import stats
        filas=8406399 #necesito saber el numero de datos exacto
        i=0
        sinres=0
        res4=0
        veintei4=0
        veintev4=0
        cuarentaimal=0
        cuarentavmal=0
        tresimal=0
        tresvmal=0
        asimal=0
        asvmal=0
        contdesctrue=0
        contdes=0
        contnr=0
        contju=0
        contid=0
        matriz=np.zeros([filas,13])
        with open("partidas.json","r") as f:
                j=0
                l=1
                for linea in f:
                    #
                    subcadenai = '"_desconectado":'
                    if linea.find(subcadenai)==-1:
                        print(linea)
                        l=l+1
                        continue
                    posicion = linea.index(subcadenai)
                    ESTADO = linea[posicion+16:posicion+20]
                    if 'tru' in ESTADO:
                        contdesctrue=contdesctrue+1
                        l=l+1
                        continue
                    ###########
                    subcadena1 = '"_id_partida":"'
                    filtro1= 'conocido'
                    if linea.find(subcadena1)==-1:
                            contid=contid+1

                            l=l+1
                            continue
```

```
posicion = linea.index(subcadena1)
ID = linea[posicion+15:posicion+33]
if filtro1 in ID:
    contdes=contdes+1
    #print(l)
    l=l+1
    continue
characters = '  " . , _  q w e r t y u i o p a f d g h j k l ñ z x c v b n m
for x in range(len(characters)):
    ID = ID.replace(characters[x],"")
if filtro1 in ID:
    contdes=contdes+1
    #print(l)
    l=l+1
    continue
ID= int(float(ID))
matriz[i,j]=ID #primera columna, codigo de partida

subcadena2 = '"_provider_full":"'
posicion = linea.index(subcadena2)
jugador = linea[posicion+18:posicion+71]

characters = '  " . , _ s - Twitteralid Facebook Google'
for x in range(len(characters)):
    jugador = jugador.replace(characters[x],"")
filtro2= ':'
if filtro2 in jugador:
    jugador= linea[posicion+18:posicion+50]
characters = ' : " . , _ s - Twitteralid Facebook Google A PAPPLE f'
for x in range(len(characters)):
    jugador = jugador.replace(characters[x],"")
if jugador.isdigit()== False:
    contju=contju+1
    #print(l)
    l=l+1
    continue

matriz[i,j+1]=jugador #segunda columna, codigo del jugador

resultado = '"_resultado":'
if linea.find(resultado)== -1:
    sinres=sinres+1
    l=l+1
    continue
else:
    posicion = linea.index(resultado)
    res = linea[posicion+13:posicion+14]
    res= int(float(res))
```

```
    if res==3:
        res=2
    if res==4:
        res4=res4+1
        l=l+1
        continue
    matriz[i,j+2]=res #tercera columna, si gana o pierde el jugador

subcadena3= '_veinte_i'
if subcadena3 in linea:
    posicion = linea.index(subcadena3)
    veintes = linea[posicion+11:posicion+12]
    veintes=int(veintes)
    if veintes == 4:
        veintei4=veintei4+1
        l=l+1
        continue
    #########
    matriz[i,j+3]=veintes #cuarta, si canta veinte o no de idas

subcadena4= '_cuarenta_i'
if subcadena4 in linea:
    posicion = linea.index(subcadena4)
    cuarenta = linea[posicion+13:posicion+14]
    cuarenta=int(cuarenta)
    if cuarenta != 1:
        cuarentaimal=cuarentaimal+1
        l=l+1
        continue
    matriz[i,j+4]=1 #quinta, cuarenta idas

subcadena5= '_tres_i'
if subcadena5 in linea:
    posicion= linea.index(subcadena5)
    tres= linea[posicion+9:posicion+10]
    tres=int(tres)
    if (tres!=1):
        tresimal=tresimal+1
        l=l+1
        continue
    matriz[i,j+5]=1 #sexta, tres de triunfo idas

subcadena6= '_as_i'
if subcadena6 in linea:
    posicion = linea.index(subcadena6)
    ass= linea[posicion+7:posicion+8]
    ass=int(ass)
    if ass!=1:
```

```
            asimal=asimal+1
            l=l+1
            continue
        matriz[i,j+6]=1 #sept, as idas

    subcadena7= '_veinte_v'
    if subcadena7 in linea:
        posicion = linea.index(subcadena7)
        veintes = linea[posicion+11:posicion+12]
        veintes=int(veintes)
        if veintes == 4:
            veintev4=veintev4+1
            l=l+1
            continue
        ######
        matriz[i,j+7]=veintes #oct, veinte vueltas

    subcadena8= '_cuarenta_v'
    if subcadena8 in linea:
        posicion = linea.index(subcadena8)
        cuarenta = linea[posicion+13:posicion+14]
        cuarenta=int(cuarenta)
        if cuarenta != 1:
            cuarentavmal=cuarentavmal+1
            l=l+1
            continue
        matriz[i,j+8]=1 #novena, cuarenta vueltas

    subcadena9= '_tres_v'
    if subcadena9 in linea:
        posicion = linea.index(subcadena9)
        tres = linea[posicion+9:posicion+10]
        tres= int(tres)
        if tres!=1:
            tresvmal=tresvmal+1
            l=l+1
            continue
        matriz[i,j+9]=1 #decima,tres vueltas

    subcadena10= '_as_v'
    if subcadena10 in linea:
        posicion= linea.index(subcadena10)
        ass= linea[posicion+7:posicion+8]
        ass=int(ass)
        if ass!=1:
            asvmal=asvmal+1
            l=l+1
            continue
```

```
                matriz[i,j+10]=1 #undecima, as vueltas

            subcadena11 = '_fecha":{"$date":"'
            posicion = linea.index(subcadena11)
            fecha = linea[posicion+23:posicion+41]
            characters = '- T Z " : , . } '
            for x in range(len(characters)):
                fecha = fecha.replace(characters[x],"")
                if len(fecha) == 7:
                    fecha=fecha+'00000'
                if len(fecha) == 8:
                     fecha=fecha+'0000'
                if len(fecha) == 9:
                     fecha=fecha+'000'
                if len(fecha) == 10:
                     fecha=fecha+'00'
                if len(fecha) == 11:
                    fecha=fecha+'0'

            matriz[i,j+11]=fecha #duodecima, refleja una cifra que es la fecha.
            #
            matriz[i,j+12]=l
            i=i+1
            #print(l)
            l=l+1
        #filas=filas-contdes-contnr
        matriz=matriz[0:filas-contdes-contnr-contju-contid-res4-contdesctrue
        -veintei4-veintev4-cuarentaimal-cuarentavmal-tresimal-tresvmal-asimal
        -asvmal-sinres]
        print(contdes)
        print(res4)
        print(contid)
        print(contju)
        print(res4)
        print(sinres,contdesctrue,veintei4,veintev4,cuarentaimal)
        print,cuarentavmal,tresimal,tresvmal,asimal,asvmal)
```

print(matriz) matfinal=matriz[0:-1] la ultima fila de la matriz son todos ceros, la eliminamos print(matfinal) print(matfinal.shape)

De registros a partidas

```
In [ ]: import numpy as np
        matpar=np.zeros([2500000,25])
        ceros=0
        j=0
        filas=0
        for i in range(7984131):
```

```
ceros=0

if data[i,0]==data[i+1,0]==data[i+2,0]==data[i+3,0]:
        filas=filas+1
        matpar[j,0]=data[i,0]
        matpar[j,1]=matpar[j,2]=matpar[j,3]=matpar[j,4]=0
        if (data[i,2]!=0)and(data[i+1,2]!=0)and(data[i+2,2]!=0)and(data[i+3,2]!=0):
            matpar[j,1]=data[i,1]

            if data[i+1,2]==data[i,2]:
                matpar[j,2]=data[i+1,1]
                matpar[j,3]=data[i+2,1]
                matpar[j,4]=data[i+3,1]


            if data[i+1,2]!=data[i,2]:
                matpar[j,3]=data[i+1,1]
                if data[i+2,2]==data[i,2]:
                    matpar[j,2]=data[i+2,1]
                    matpar[j,4]=data[i+3,1]
                if data[i+2,2]!=data[i,2]:
                    matpar[j,4]=data[i+2,1]
                    matpar[j,2]=data[i+3,1]



        if (data[i,2]==0)or(data[i+1,2]==0)or(data[i+2,2]==0)or(data[i+3,2]==0):
            #matpar[j,0]=0
            #filas=filas-1
            #continue
            if data[i,2]==0:
                ceros=ceros+1
            if data[i+1,2]==0:
                ceros=ceros+1
            if data[i+2,2]==0:
                ceros=ceros+1
            if data[i+3,2]==0:
                ceros=ceros+1



            if ceros!=1:
                matpar[j,0]=0
                filas=filas-1
                continue
            if ceros==1:
                if data[i,2]!=0:
                    matpar[j,1]=data[i,1]
```

```
                if data[i,2]==data[i+1,2]:
                    matpar[j,2]=data[i+1,1]
                    matpar[j,3]=data[i+2,1]
                    matpar[j,4]=data[i+3,1]
                if data[i,2]==data[i+2,2]:
                    matpar[j,2]=data[i+2,1]
                    matpar[j,3]=data[i+1,1]
                    matpar[j,4]=data[i+3,1]
                if data[i,2]==data[i+3,2]:
                    matpar[j,2]=data[i+3,1]
                    matpar[j,3]=data[i+1,1]
                    matpar[j,4]=data[i+2,1]
                if data[i+1,2]==data[i+2,2]:
                    matpar[j,2]=data[i+3,1]
                    matpar[j,3]=data[i+1,1]
                    matpar[j,4]=data[i+2,1]
                if data[i+1,2]==data[i+3,2]:
                    matpar[j,2]=data[i+2,1]
                    matpar[j,3]=data[i+1,1]
                    matpar[j,4]=data[i+3,1]
                if data[i+2,2]==data[i+3,2]:
                    matpar[j,2]=data[i+1,1]
                    matpar[j,3]=data[i+2,1]
                    matpar[j,4]=data[i+3,1]
            if data[i,2]==0:
                matpar[j,1]=data[i,1]
                if data[i+1,2]==data[i+2,2]:
                    matpar[j,2]=data[i+3,1]
                    matpar[j,3]=data[i+1,1]
                    matpar[j,4]=data[i+2,1]
                if data[i+1,2]==data[i+3,2]:
                    matpar[j,2]=data[i+2,1]
                    matpar[j,3]=data[i+1,1]
                    matpar[j,4]=data[i+3,1]
                if data[i+2,2]==data[i+3,2]:
                    matpar[j,2]=data[i+1,1]
                    matpar[j,3]=data[i+2,1]
                    matpar[j,4]=data[i+3,1]



 #ahora el tresi
if data[i,5]!=0:
    if matpar[j,1]==data[i,1]:
        matpar[j,5]=1
    elif matpar[j,2]==data[i,1]:
        matpar[j,5]=2
```

```
        elif matpar[j,3]==data[i,1]:
            matpar[j,5]=3
        else:
            matpar[j,5]=4
    elif data[i+1,5]!=0:
        if matpar[j,1]==data[i+1,1]:
            matpar[j,5]=1
        elif matpar[j,2]==data[i+1,1]:
            matpar[j,5]=2
        elif matpar[j,3]==data[i+1,1]:
            matpar[j,5]=3
        else:
            matpar[j,5]=4
    elif data[i+2,5]!=0:
        if matpar[j,1]==data[i+2,1]:
            matpar[j,5]=1
        elif matpar[j,2]==data[i+2,1]:
            matpar[j,5]=2
        elif matpar[j,3]==data[i+2,1]:
            matpar[j,5]=3
        else:
            matpar[j,5]=4
    else:
        if matpar[j,1]==data[i+3,1]:
            matpar[j,5]=1
        elif matpar[j,2]==data[i+3,1]:
            matpar[j,5]=2
        elif matpar[j,3]==data[i+3,1]:
            matpar[j,5]=3
        else:
            matpar[j,5]=4



    if data[i,6]!=0:
        if matpar[j,1]==data[i,1]:
            matpar[j,6]=1
        elif matpar[j,2]==data[i,1]:
            matpar[j,6]=2
        elif matpar[j,3]==data[i,1]:
            matpar[j,6]=3
        else:
            matpar[j,6]=4


    elif data[i+1,6]!=0:
        if matpar[j,1]==data[i+1,1]:
            matpar[j,6]=1
```

```
    elif matpar[j,2]==data[i+1,1]:
        matpar[j,6]=2
    elif matpar[j,3]==data[i+1,1]:
        matpar[j,6]=3
    else:
        matpar[j,6]=4
elif data[i+2,6]!=0:
    if matpar[j,1]==data[i+2,1]:
        matpar[j,6]=1
    elif matpar[j,2]==data[i+2,1]:
        matpar[j,6]=2
    elif matpar[j,3]==data[i+2,1]:
        matpar[j,6]=3
    else:
        matpar[j,6]=4
else:
    if matpar[j,1]==data[i+3,1]:
        matpar[j,6]=1
    elif matpar[j,2]==data[i+3,1]:
        matpar[j,6]=2
    elif matpar[j,3]==data[i+3,1]:
        matpar[j,6]=3
    else:
        matpar[j,6]=4
#veintes
matpar[j,7]=matpar[j,8]=matpar[j,9]=matpar[j,10]=0

if data[i,3]!=0:
    if data[i,1]==matpar[j,1]:
        matpar[j,7]=data[i,3]
    if data[i,1]==matpar[j,2]:
        matpar[j,8]=data[i,3]
    if data[i,1]==matpar[j,3]:
        matpar[j,9]=data[i,3]
    if data[i,1]==matpar[j,4]:
        matpar[j,10]=data[i,3]
if data[i+1,3]!=0:
    if data[i+1,1]==matpar[j,1]:
        matpar[j,7]=data[i+1,3]
    if data[i+1,1]==matpar[j,2]:
        matpar[j,8]=data[i+1,3]
    if data[i+1,1]==matpar[j,3]:
        matpar[j,9]=data[i+1,3]
    if data[i+1,1]==matpar[j,4]:
        matpar[j,10]=data[i+1,3]
if data[i+2,3]!=0:
    if data[i+2,1]==matpar[j,1]:
        matpar[j,7]=data[i+2,3]
```

```
    if data[i+2,1]==matpar[j,2]:
        matpar[j,8]=data[i+2,3]
    if data[i+2,1]==matpar[j,3]:
        matpar[j,9]=data[i+2,3]
    if data[i+2,1]==matpar[j,4]:
        matpar[j,10]=data[i+2,3]
if data[i+3,3]!=0:
    if data[i+3,1]==matpar[j,1]:
        matpar[j,7]=data[i+3,3]
    if data[i+3,1]==matpar[j,2]:
        matpar[j,8]=data[i+3,3]
    if data[i+3,1]==matpar[j,3]:
        matpar[j,9]=data[i+3,3]
    if data[i+3,1]==matpar[j,4]:
        matpar[j,10]=data[i+3,3]
#cuarenta
if data[i,4]!=0:
    if data[i,1]==matpar[j,1]:
        matpar[j,11]=1
    if data[i,1]==matpar[j,2]:
        matpar[j,11]=2
    if data[i,1]==matpar[j,3]:
        matpar[j,11]=3
    if data[i,1]==matpar[j,4]:
        matpar[j,11]=4
elif data[i+1,4]!=0:
    if data[i+1,1]==matpar[j,1]:
        matpar[j,11]=1
    if data[i+1,1]==matpar[j,2]:
        matpar[j,11]=2
    if data[i+1,1]==matpar[j,3]:
        matpar[j,11]=3
    if data[i+1,1]==matpar[j,4]:
        matpar[j,11]=4
elif data[i+2,4]!=0:
    if data[i+2,1]==matpar[j,1]:
        matpar[j,11]=1
    if data[i+2,1]==matpar[j,2]:
        matpar[j,11]=2
    if data[i+2,1]==matpar[j,3]:
        matpar[j,11]=3
    if data[i+2,1]==matpar[j,4]:
        matpar[j,11]=4
elif data[i+3,4]!=0:
    if data[i+3,1]==matpar[j,1]:
        matpar[j,11]=1
    if data[i+3,1]==matpar[j,2]:
        matpar[j,11]=2
```

```
    if data[i+3,1]==matpar[j,3]:
        matpar[j,11]=3
    if data[i+3,1]==matpar[j,4]:
        matpar[j,11]=4
else:
    matpar[j,11]=0



 #ahora el tresv
matpar[j,12]=0
if data[i,9]!=0:
    if matpar[j,1]==data[i,1]:
        matpar[j,12]=1
    elif matpar[j,2]==data[i,1]:
        matpar[j,12]=2
    elif matpar[j,3]==data[i,1]:
        matpar[j,12]=3
    else:
        matpar[j,12]=4
elif data[i+1,9]!=0:
    if matpar[j,1]==data[i+1,1]:
        matpar[j,12]=1
    elif matpar[j,2]==data[i+1,1]:
        matpar[j,12]=2
    elif matpar[j,3]==data[i+1,1]:
        matpar[j,12]=3
    else:
        matpar[j,12]=4
elif data[i+2,9]!=0:
    if matpar[j,1]==data[i+2,1]:
        matpar[j,12]=1
    elif matpar[j,2]==data[i+2,1]:
        matpar[j,12]=2
    elif matpar[j,3]==data[i+2,1]:
        matpar[j,12]=3
    else:
        matpar[j,12]=4
elif data[i+3,9]!=0:
    if matpar[j,1]==data[i+3,1]:
        matpar[j,12]=1
    elif matpar[j,2]==data[i+3,1]:
        matpar[j,12]=2
    elif matpar[j,3]==data[i+3,1]:
        matpar[j,12]=3
    else:
        matpar[j,12]=4
else:
```

```
    matpar[j,12]=0



if data[i,10]!=0:
    if matpar[j,1]==data[i,1]:
        matpar[j,13]=1
    elif matpar[j,2]==data[i,1]:
        matpar[j,6]=2
    elif matpar[j,3]==data[i,1]:
        matpar[j,13]=3
    else:
        matpar[j,13]=4


elif data[i+1,10]!=0:
    if matpar[j,1]==data[i+1,1]:
        matpar[j,13]=1
    elif matpar[j,2]==data[i+1,1]:
        matpar[j,13]=2
    elif matpar[j,3]==data[i+1,1]:
        matpar[j,13]=3
    else:
        matpar[j,13]=4
elif data[i+2,10]!=0:
    if matpar[j,1]==data[i+2,1]:
        matpar[j,13]=1
    elif matpar[j,2]==data[i+2,1]:
        matpar[j,13]=2
    elif matpar[j,3]==data[i+2,1]:
        matpar[j,13]=3
    else:
        matpar[j,13]=4
elif data[i+3,10]!=0:
    if matpar[j,1]==data[i+3,1]:
        matpar[j,13]=1
    elif matpar[j,2]==data[i+3,1]:
        matpar[j,13]=2
    elif matpar[j,3]==data[i+3,1]:
        matpar[j,13]=3
    else:
        matpar[j,13]=4
else:
    matpar[j,13]=0
#veintes
matpar[j,14]=matpar[j,15]=matpar[j,16]=matpar[j,17]=0

if data[i,7]!=0:
```

```
    if data[i,1]==matpar[j,1]:
        matpar[j,14]=data[i,7]
    if data[i,1]==matpar[j,2]:
        matpar[j,15]=data[i,7]
    if data[i,1]==matpar[j,3]:
        matpar[j,16]=data[i,7]
    if data[i,1]==matpar[j,4]:
        matpar[j,17]=data[i,7]
if data[i+1,7]!=0:
    if data[i+1,1]==matpar[j,1]:
        matpar[j,14]=data[i+1,7]
    if data[i+1,1]==matpar[j,2]:
        matpar[j,15]=data[i+1,7]
    if data[i+1,1]==matpar[j,3]:
        matpar[j,16]=data[i+1,7]
    if data[i+1,1]==matpar[j,4]:
        matpar[j,17]=data[i+1,7]
if data[i+2,7]!=0:
    if data[i+2,1]==matpar[j,1]:
        matpar[j,14]=data[i+2,7]
    if data[i+2,1]==matpar[j,2]:
        matpar[j,15]=data[i+2,7]
    if data[i+2,1]==matpar[j,3]:
        matpar[j,16]=data[i+2,7]
    if data[i+2,1]==matpar[j,4]:
        matpar[j,17]=data[i+2,7]
if data[i+3,7]!=0:
    if data[i+3,1]==matpar[j,1]:
        matpar[j,14]=data[i+3,7]
    if data[i+3,1]==matpar[j,2]:
        matpar[j,15]=data[i+3,7]
    if data[i+3,1]==matpar[j,3]:
        matpar[j,16]=data[i+3,7]
    if data[i+3,1]==matpar[j,4]:
        matpar[j,17]=data[i+3,7]
#cuarenta
if data[i,8]!=0:
    if data[i,1]==matpar[j,1]:
        matpar[j,18]=1
    if data[i,1]==matpar[j,2]:
        matpar[j,18]=2
    if data[i,1]==matpar[j,3]:
        matpar[j,18]=3
    if data[i,1]==matpar[j,4]:
        matpar[j,18]=4
elif data[i+1,8]!=0:
    if data[i+1,1]==matpar[j,1]:
        matpar[j,18]=1
```

```
    if data[i+1,1]==matpar[j,2]:
        matpar[j,18]=2
    if data[i+1,1]==matpar[j,3]:
        matpar[j,18]=3
    if data[i+1,1]==matpar[j,4]:
        matpar[j,18]=4
elif data[i+2,8]!=0:
    if data[i+2,1]==matpar[j,1]:
        matpar[j,18]=1
    if data[i+2,1]==matpar[j,2]:
        matpar[j,18]=2
    if data[i+2,1]==matpar[j,3]:
        matpar[j,18]=3
    if data[i+2,1]==matpar[j,4]:
        matpar[j,18]=4
elif data[i+3,8]!=0:
    if data[i+3,1]==matpar[j,1]:
        matpar[j,18]=1
    if data[i+3,1]==matpar[j,2]:
        matpar[j,18]=2
    if data[i+3,1]==matpar[j,3]:
        matpar[j,18]=3
    if data[i+3,1]==matpar[j,4]:
        matpar[j,18]=4
else:
    matpar[j,18]=0

matpar[j,19]=data[i,11]

matpar[j,20]=data[i,12]
matpar[j,21]=data[i+1,12]
matpar[j,22]=data[i+2,12]
matpar[j,23]=data[i+3,12]

if (data[i,2]==1):
    matpar[j,24]=1
if (data[i,2]==2):
    matpar[j,24]=2
if (data[i,2]==0):
    if data[i+1,2]==data[i+2,2]:
        if data[i+1,2]==1:
            matpar[j,24]=2
        if data[i+1,2]==2:
            matpar[j,24]=1
    if data[i+1,2]==data[i+3,2]:
        if data[i+1,2]==1:
            matpar[j,24]=2
        if data[i+1,2]==2:
```

```
                        matpar[j,24]=1
                    if data[i+2,2]==data[i+3,2]:
                        if data[i+1,2]==1:
                            matpar[j,24]=1
                        if data[i+1,2]==2:
                            matpar[j,24]=2

                j=j+1

        matpar=matpar[0:filas]
        print(matpar)
        print(matpar[0,])
```

Filtro de los veintes

```
In [ ]: import numpy as np
        mat=np.zeros([2500000,25])
        veintetoterror=0
        for i in range(1743221):
            if (data[i,7]+data[i,8]+data[i,9]+data[i,10]<4)and(data[i,14]
            +data[i,15]+data[i,16]+data[i,17]<4):
                mat[i,]=data[i,]

            if (data[i,7]+data[i,8]+data[i,9]+data[i,10]>3)or(data[i,14]+data[i,15]

            +data[i,16]+data[i,17]>3):
                veintetoterror=veintetoterror+1
        print(mat[0:1743222])
        print(veintetoterror)

        mat=mat[0:1743221]
```

Código para crear las puntuaciones de los jugadores

```
In [ ]: import numpy as np
        mat=np.zeros([1742897,40])
        print(mat.shape)
        import numpy as np
        numpar=np.zeros([15583,2])
        import numpy as np
        pargan=np.zeros([15583,2])
        import numpy as np
        elo=np.zeros([15583,2])

        for i in range(15583):
            numpar[i,0]=i+1
            pargan[i,0]=i+1
            elo[i,0]=i+1
```

```
    elo[i,1]=1200

for i in range(1742897):
    for j in range(25):
        mat[i,j]=data[i,j]

for i in range(1742897):
    a=int(mat[i,1])-1
    b=int(mat[i,2])-1
    c=int(mat[i,3])-1
    d=int(mat[i,4])-1

    numpar[a,1]=numpar[a,1]+1
    numpar[b,1]=numpar[b,1]+1
    numpar[c,1]=numpar[c,1]+1
    numpar[d,1]=numpar[d,1]+1


    elo1=(int(elo[a,1])+int(elo[b,1]))/2
    elo2=(int(elo[c,1])+int(elo[d,1]))/2


    if mat[i,24]==1:
        pargan[a,1]=pargan[a,1]+1
        pargan[b,1]=pargan[b,1]+1

        elo1n=elo1+32*(1-(1/(1+10**((elo2-elo1)/400))))
        dif1=elo1n-elo1
        elo[a,1]=elo[a,1]+dif1/2
        elo[b,1]=elo[b,1]+dif1/2
        elo2n=elo2+32*(-(1/(1+10**((elo1-elo2)/400))))
        dif2=elo2n-elo2
        elo[c,1]=elo[c,1]+dif2/2
        elo[d,1]=elo[d,1]+dif2/2

    if mat[i,24]==2:
        pargan[c,1]=pargan[c,1]+1
        pargan[d,1]=pargan[d,1]+1

        elo1n=elo1+32*(-(1/(1+10**((elo2-elo1)/400))))
        dif1=elo1n-elo1
        elo[a,1]=elo[a,1]+dif1/2
        elo[b,1]=elo[b,1]+dif1/2
        elo2n=elo2+32*(1-(1/(1+10**((elo1-elo2)/400))))
        dif2=elo2n-elo2
        elo[c,1]=elo[c,1]+dif2/2
        elo[d,1]=elo[d,1]+dif2/2
```

```
            mat[i,27]=numpar[a,1]
            mat[i,28]=pargan[a,1]
            mat[i,29]=elo[a,1]

            mat[i,30]=numpar[b,1]
            mat[i,31]=pargan[b,1]
            mat[i,32]=elo[b,1]

            mat[i,33]=numpar[c,1]
            mat[i,34]=pargan[c,1]
            mat[i,35]=elo[c,1]

            mat[i,36]=numpar[d,1]
            mat[i,37]=pargan[d,1]
            mat[i,38]=elo[d,1]

            if (mat[i,27]>=200)and(mat[i,30]>=200)and(mat[i,33]>=200)and(mat[i,36]>=200):
                 mat[i,39]=1
```

Código para aleatorizar parejas

```
In [ ]: import numpy as np
        mat=np.zeros([739801,39])
        print(mat.shape)

        for i in range(739801):
            if (data[i,1]+data[i,2])%2==0:
                mat[i,]=data[i,]
            if (data[i,1]+data[i,2])%2!=0:
                #los que son iguales
                mat[i,0]=data[i,0]
                mat[i,19]=data[i,19]
                mat[i,20]=data[i,20]
                mat[i,21]=data[i,21]
                mat[i,22]=data[i,22]
                mat[i,23]=data[i,23]

                #lo que cambia

                mat[i,1]=data[i,3]
                mat[i,2]=data[i,4]
                mat[i,3]=data[i,1]
                mat[i,4]=data[i,2]

                if data[i,5]<3:
```

```
    mat[i,5]=data[i,5]+2
if data[i,5]>2:
    mat[i,5]=data[i,5]-2

if data[i,6]<3:
    mat[i,6]=data[i,6]+2
if data[i,6]>2:
    mat[i,6]=data[i,6]-2

mat[i,7]=data[i,9]
mat[i,8]=data[i,10]
mat[i,9]=data[i,7]
mat[i,10]=data[i,8]

if data[i,11]==0:
    mat[i,11]=0
if data[i,11]!=0:
    if data[i,11]<3:
        mat[i,11]=data[i,11]+2
    if data[i,11]>2:
        mat[i,11]=data[i,11]-2

if data[i,12]==0:
    mat[i,12]=0
if data[i,12]!=0:
    if data[i,12]<3:
        mat[i,12]=data[i,12]+2
    if data[i,12]>2:
        mat[i,12]=data[i,12]-2

if data[i,13]==0:
    mat[i,13]=0
if data[i,13]!=0:
    if data[i,13]<3:
        mat[i,13]=data[i,13]+2
    if data[i,13]>2:
        mat[i,13]=data[i,13]-2

mat[i,16]=data[i,14]
mat[i,17]=data[i,15]
mat[i,14]=data[i,16]
mat[i,15]=data[i,17]

if data[i,18]==0:
    mat[i,18]=0
if data[i,18]!=0:
    if data[i,18]<3:
        mat[i,18]=data[i,18]+2
```

```
    if data[i,18]>2:
        mat[i,18]=data[i,18]-2

if data[i,24]==1:
    mat[i,24]=2
if data[i,24]==2:
    mat[i,24]=1

mat[i,31]=data[i,25]
mat[i,32]=data[i,26]
mat[i,33]=data[i,27]
mat[i,34]=data[i,28]
mat[i,35]=data[i,29]
mat[i,36]=data[i,30]
mat[i,25]=data[i,31]
mat[i,26]=data[i,32]
mat[i,27]=data[i,33]
mat[i,28]=data[i,34]
mat[i,29]=data[i,35]
mat[i,30]=data[i,36]

mat[i,37]=data[i,37]

if mat[i,24]==1:
    mat[i,38]=1
if mat[i,24]==2:
    mat[i,38]=0
```

ANEXO B: Notas de R

En este anexo vamos a mostrar y explicar el código usado en R para trabajar la base de datos.

Preparativos previos:

Antes de pasar de registros a partidas:

```
colnames(filfechacorta2)<-c("id", "jugador", "resultado", "20i", "40i", "ti", "ci",
                        "20v", "40v", "tv", "cv","fecha","registro")
nombres=unique(filfechacorta2$jugador)
filfechacorta2$jugador=match(filfechacorta2$jugador,nombres)

arrange(filfechacorta2, id)
show(arrange(filfechacorta2, id))
write.csv(arrange(filfechacorta2, id),row.names = FALSE,"renum1.csv")
```

Ponemos nombres a las columnas de la matriz de registros creada en Python, renombramos a los jugadores para tener números más cómodos para manejar y los ordenamos por id de partida para llevarlos a Python y crear a partir de 4 registros el registro de una partida.

Depuración

```
colnames(datf)<-c("id", "jugador1", "jugador2","jugador3","jugador4", "tres_i", "as_i",
"veinteg1","veinteg2","veintep1","veintep2", "cuarenta_i","tres_v", "as_v"
,"veinteg1v","veinteg2v","veintep1v","veintep2v" ,"cuarenta_v","fecha","registro1"
,"registro2","registro3","registro4","resultado")
partidaslimpias<- datf[datf$id!=0,]
write.csv(datf, rownames=FALSE,"")
write.csv(partidaslimpias,row.names = FALSE,"partidas.csv")
```

En el programa de Python sustituimos las filas con errores por ceros, entonces aquí las eliminamos.

Código para el análisis descriptivo

```
table(partidas$cuarenta_i)#/length(partidas$id)

part40<-partidas[partidas$cuarenta_i!=0,] #aislar las partidas en las que se hayan cantado 40
prop.table(table(part40$resultado,part40$cuarenta_i))#porcentaje de partidas ganadas con las 40
prop.table(table((partidas$cuarenta_i))) #proporcion de partidas en las que se cantan las cuarenta



veinte1<- partidas$veinteg1
veinte2<- partidas$veinteg2
veinte3<- partidas$veintep1
veinte4<- partidas$veintep2
```

```
matvein <- cbind(veinte1, veinte2, veinte3, veinte4)

numveint<- apply(X=matvein, MARGIN=1, FUN=sum ) #es el numero de veintes que se cantan en cada partida
table(numveint)/length(partidas$id)
mean(numveint) # 0.5493073
sd(numveint) #0.676758
max(numveint) #en este punto se detectó el problema de más de 3 veintes en algunas partidas
min(numveint)

table(partidas$as_i,partidas$resultado)/length(partidas$id)*100
```

```
tresyas<- function(as,tres){
  if (as == 1){
    if(tres < 3)
      {return(1)}
    if (tres > 2)
      {return(0)}}
  if (as == 2){
    if(tres < 3)
     { return(1)}
    if(tres > 2)
      {return(0)}}
  if (as == 3){
    if(tres > 2)
      {return(2)}
    if(tres < 3)
      {return(0)}}
  if (as == 4){
    if(tres > 2)
      {return(2)}
    if (tres < 3)
      {return(0)}}
}
partidas$nueva_columna=mapply(tresyas,partidas$as_i,partidas$tres_i)
table(partidas$nueva_columna)#/length(partidas$id)
partasytres<-partidas[partidas$nueva_columna!=0,]#as y tres en la misma pareja
table(partasytres$resultado,partasytres$nueva_columna)/length(partasytres$id)*100
#78.0345

table(partidas$nueva_columna)/length(partidas$id)*100
tresyasjugador<- function(as,tres){
 if(as==tres)
 {return(as)}
  else
  {return(0)}

}
partidas$nueva_columna=mapply(tresyasjugador,partidas$as_i,partidas$tres_i)
table(partidas$nueva_columna)#/length(partidas$id)*100

partasytresjugador<-partidas[partidas$nueva_columna!=0,]
ganadorcontresyas<-function(resultado,ncol){
  if(ncol==1){
```

```
    if (resultado==1)
    {return(1)}
    if (resultado==2)
    {return(0)}
  }
  if(ncol==2){
    if (resultado==1)
    {return(1)}
    if (resultado==2)
    {return(0)}
  }
  if(ncol==3){
    if (resultado==1)
    {return(0)}
    if (resultado==2)
    {return(1)}
  }
  if(ncol==4){
    if (resultado==1)
    {return(0)}
    if (resultado==2)
    {return(1)}
  }
}


partasytresjugador$nueva_columna2=mapply(ganadorcontresyas,partasytresjugador
$resultado,partasytresjugador$nueva_columna)

table(partasytresjugador$nueva_columna2)/length(partidas$id)*100
```

```
tresyasdistju<- function(as,tres){
  if (as == 1){
    if(tres ==2)
    {return(12)}
    if (tres != 2)
    {return(0)}}
  if (as == 2){
    if(tres == 1)
    { return(21)}
    if(tres !=1)
    {return(0)}}
  if (as == 3){
    if(tres == 4)
    {return(34)}
    if(tres !=4)
    {return(0)}}
  if (as == 4){
    if(tres ==3)
    {return(43)}
    if (tres != 3)
    {return(0)}}
}
partidas$nueva_columna4=mapply(tresyasdistju,partidas$as_i,partidas$tres_i)
```

```
table(partidas$nueva_columna4)#/length(partidas$id)*100

partasytresdistju<-partidas[partidas$nueva_columna4!=0,]

victdistju<-function(ncol,resul){
  if(ncol<25){
    if(resul==1){
      return(1)}
    if(resul==2){
      return(0)
    }}
  if(ncol>25){
    if(resul==1){
      return(0)}
    if(resul==2){
      return(1)
    }
  }

}
partasytresdistju$nueva_columna=mapply
(victdistju,partasytresdistju$nueva_columna4,partasytresdistju$resultado)

table(partasytresdistju$nueva_columna)/length(partasytresdistju$id)*100
```

En algunas partes está /length(partidas$id)*100 como comentario ya que si se pone te da el porcentaje de partidas y si no da el número de partidas.

Puntuaciones

Preparativos

```
RESULTADOBINARIO<-function(result){
  if(result==1)
  {return(1)}
  if(result==2)
  {return(0)}
}
partidas$resbin=mapply(RESULTADOBINARIO,partidas$resultado)
write.csv(arrange(partidas, fecha), row.names = FALSE, "porfecha.csv")
```

Se pasan los resultados a binario y se ordenan las partidas por fecha para poder asignar en Python las puntuaciones de los jugadores.

Base de datos para los modelos

```
colnames(ranking2_0)<-c("id", "jugador1", "jugador2","jugador3","jugador4", "tres_i", "as_i",
"veinteg1","veinteg2","veintep1","veintep2", "cuarenta_i","tres_v", "as_v"
,"veinteg1v","veinteg2v","veintep1v","veintep2v" ,"cuarenta_v","fecha","registro1"
,"registro2","registro3","registro4","resultado","numpar1","pargan1","elo1","numpar2","pargan2"
,"elo2","numpar3","pargan3","elo3","numpar4","pargan4","elo4","valida")
```

```
rankbueno<-ranking2[ranking2$valida!=0,]
write.csv(rankbueno, row.names = FALSE, "rankbueno.csv")
rankbueno$resbin=mapply(RESULTADOBINARIO,rankbueno$resultado)
write.csv(rankbueno, row.names = FALSE, "ranknoale2.csv")
```

En este momento, tras obtener en Python la base de datos con las puntuaciones, eliminamos las partidas no válidas para los modelos. Las partidas no válidas son aquellas en las que participa algún jugador con menos de 200 partidas jugadas.

```
plot(seq(1,500),seq(1,500),type="n",xlim=c(1,500),ylim=c(960,1500),ylab="Prop
ganadas")

numerito=400
esta=rankale[which((rankale$jugador1==numerito)
|(rankale$jugador2==numerito)|(rankale$jugador3==numerito)|(rankale$jugador4==numerito)),]
esta$ratio=(esta$pargan1/esta$numpar1)*(esta$jugador1==numerito)+(esta$pargan2/esta$numpar2)
*(esta$jugador2==numerito)+(esta$pargan3/esta$numpar3)*(esta$jugador3==numerito)
+(esta$pargan4/esta$numpar4)*(esta$jugador4==numerito)
lines(seq(1,length(esta$ratio)),estar$ratio)
```

Con este código se comprobó que los porcentajes de partidas ganadas se estabilizaban en 200 partidas jugadas, por lo tanto se decidió que el filtro debía de ser de 200 partidas.

Este conjunto de datos se lleva a Python y se aleatorizan las parejas.

Nuevos datos para los modelos

```
elopareja<- function(elo1,elo2){
  elotot=(elo1+elo2)/2
  return(elotot)

}
rankale2$elopar1=mapply(elopareja,rankale2$elo1,rankale2$elo2)
rankale2$elopar2=mapply(elopareja, rankale2$elo3,rankale2$elo4)
difelo<-function(elo1,elo2){
  diferencia=elo1-elo2
  return(diferencia)
}
rankale2$difelopar=mapply(difelo, rankale2$elopar1, rankale2$elopar2)
##con los minimos y maximos
rankale2$minelo1=mapply(min, rankale2$elo1,rankale2$elo2)
rankale2$minelo2=mapply(min, rankale2$elo3,rankale2$elo4)
rankale2$maxelo1=mapply(max, rankale2$elo1,rankale2$elo2)
rankale2$maxelo2=mapply(max, rankale2$elo3,rankale2$elo4)


porcentajeganadas<-function(vict,nume){
  totpor= (vict/nume)*100
  return(totpor)
}
rankale2$porc1=mapply(porcentajeganadas, rankale2$pargan1, rankale2$numpar1)
rankale2$porc2=mapply(porcentajeganadas, rankale2$pargan2, rankale2$numpar2)
rankale2$porc3=mapply(porcentajeganadas, rankale2$pargan3, rankale2$numpar3)
rankale2$porc4=mapply(porcentajeganadas, rankale2$pargan4, rankale2$numpar4)
```

```
tres1<-function(tresi){
  if((tresi==1)|(tresi==2))
  {return(1)}
  if((tresi==3)|(tresi==4))
  {return(0)}
}
as1<-function(asi){
  if((asi==1)|(asi==2))
  {return(1)}
  if((asi==3)|(asi==4))
  {return(0)}
}

rankale2$trespareja1=mapply(tres1,rankale2$tres_i)
rankale2$aspareja1=mapply(as1,rankale2$as_i)

cuarentapar<-function(cuaren){
  if((cuaren==1)|(cuaren==2))
  {return(1)}
  if((cuaren==3)|(cuaren==4))
  {return(-1)}
  if(cuaren==0)
  {return(0)}
}
rankale2$cuaren=mapply(cuarentapar, rankale2$cuarenta_i)

difveintes<-function(v1,v2,v3,v4){
  difeveinte=v1+v2-v3-v4
  return(difeveinte)
}
rankale2$difeveintes= mapply(difveintes, rankale2$veintej1,rankale2$veintej2,rankale2
$veintej3,rankale2$veintej4)

rankale2$minpor1=mapply(min, rankale2$porc1,rankale2$porc2)
rankale2$minpor2=mapply(min, rankale2$porc3,rankale2$porc4)
rankale2$maxpor1=mapply(max, rankale2$porc1, rankale2$porc2)
rankale2$maxpor2=mapply(max, rankale2$porc3, rankale2$porc4)
```

Con la base de datos ya aleatorizada, introducimos todos los datos y mediciones que consideramos importantes a la hora de hacer los modelos.

Modelos

En este apartado tenemos todo el código que corresponde a los modelos, en él se pueden ver todos los modelos, incluso los descartados, además de el estadístico de Cox-Snell y el de Nagelkerke que son los que hemos usado para ver la capacidad explicativa del modelo y descartar algunos.

```
##modelo1
#unicamente con el as
GLM.1 <- glm(resbin ~ aspareja1, family=binomial(logit), data=rankale2)
summary(GLM.1)
exp(coef(GLM.1))

CHI1=GLM.1$null.deviance-GLM.1$deviance
```

```
Difgrlib1=GLM.1$df.null-GLM.1$df.residual
CHI1#76814.86
Difgrlib1#1


hl1=CHI1/GLM.1$null.deviance
hl1
CS1=1-exp((GLM.1$deviance-GLM.1$null.deviance)/length(rankale2$id))
n1=CS1/(1-(exp(-(GLM.1$null.deviance/length(rankale2$id)))))
#estadisticos de Cox-Snell y Nagelkerke
#entre un 13.14% y un 7.4%
pre1<-predict(object=GLM.1,newdata=rankale2, type='response')
pre1
rankale2$premod1<-ifelse(test= pre1>0.5,yes=1,no=0)
rankale2$acim1=mapply(poracierto, rankale2$resbin, rankale2$premod1)
table(rankale2$acim1)/length(rankale2$id)
#el modelo es capaz de acertar el 0.66% de las partidas


#as y tres
GLM.2<- glm(resbin ~ aspareja1+trespareja1, family=binomial(logit), data=rankale2)
summary(GLM.2)
exp(coef(GLM.2))

CHI2=GLM.2$null.deviance-GLM.2$deviance
CHI2 #123417.4
Difgrlib2=GLM.2$df.null-GLM.2$df.residual
Difgrlib2#2

hl2=CHI2/GLM.2$null.deviance
CS2=1-exp((GLM.2$deviance-GLM.2$null.deviance)/length(rankale2$id))
hl2
n2=CS2/(1-(exp(-(GLM.2$null.deviance/length(rankale2$id)))))
CS2
n2


pre2<-predict(object=GLM.2,newdata=rankale2, type='response')
pre2

rankale2$premod2<-ifelse(test= pre2>0.5,yes=1,no=0)


rankale2$acim2=mapply(poracierto, rankale2$resbin, rankale2$premod2)
table(rankale2$acim2)/length(rankale2$id)

#0.66 de aciertos igual que el anterior
#con el as y el tres, explicamos un 20.49% y un 15.36%
```

```
#as tres y veinte
GLM.20<- glm(resbin ~ aspareja1+trespareja1+difeveintes, family=binomial(logit), data=rankale2)
summary(GLM.20)
exp(coef(GLM.20))
Difgrlib20=GLM.20$df.null-GLM.20$df.residual
```

```
Difgrlib20#3
CHI20=GLM.20$null.deviance-GLM.20$deviance
CHI20#133173.6


CS20=1-exp((GLM.20$deviance-GLM.20$null.deviance)/length(rankale2$id))
n20=CS20/(1-(exp(-(GLM.20$null.deviance/length(rankale2$id)))))
CS20
n20

pre20<-predict(object=GLM.20,newdata=rankale2, type='response')
rankale2$premod20<-ifelse(test= pre20>0.5,yes=1,no=0)
rankale2$acim20=mapply(poracierto, rankale2$resbin, rankale2$premod20)
table(rankale2$acim20)/length(rankale2$id)
#acierta el 66,8 de las partidas


#as tres y cuarenta +20
GLM.3<- glm(resbin ~ aspareja1+trespareja1+cuaren+difeveintes, family=binomial(logit), data=rankale2)
summary(GLM.3)
exp(coef(GLM.3))

CHI3=GLM.3$null.deviance-GLM.3$deviance
CHI3#257423.9


hl3=CHI3/GLM.3$null.deviance
hl3
CS3=1-exp((GLM.3$deviance-GLM.3$null.deviance)/length(rankale2$id)
n3=CS3/(1-(exp(-(GLM.3$null.deviance/length(rankale2$id)))))
CS3
n3

pre3<-predict(object=GLM.3,newdata=rankale2, type='response')


rankale2$premod3<-ifelse(test= pre3>0.5,yes=1,no=0)


rankale2$acim3=mapply(poracierto, rankale2$resbin, rankale2$premod3)
table(rankale2$acim3)/length(rankale2$id)
##0.73 de aciertos
```

```
#elo pareja
GLM.elo<- glm(resbin ~difelopar, family=binomial(logit), data=rankale2)
summary(GLM.elo)
exp(coef(GLM.elo))

CHIelo=GLM.elo$null.deviance-GLM.elo$deviance
CHIelo
hlelo=CHIelo/GLM.elo$null.deviance #
CSelo=1-exp((GLM.elo$deviance-GLM.elo$null.deviance)/length(rankale2$id))
```

```
nelo=CSelo/(1-(exp(-(GLM.elo$null.deviance/length(rankale2$id)))))
CSelo
nelo


preelo<-predict(object=GLM.elo,newdata=rankale2, type='response')


rankale2$premodelo<-ifelse(test= preelo>0.5,yes=1,no=0)


rankale2$acimelo=mapply(poracierto, rankale2$resbin, rankale2$premodelo)
table(rankale2$acimelo)/length(rankale2$id)

# as y elos pareja

GLM.4<- glm(resbin ~ aspareja1+difelopar, family=binomial(logit), data=rankale2)
summary(GLM.4)
exp(coef(GLM.4))

CHI4=GLM.4$null.deviance-GLM.4$deviance
CHI4
hl4=CHI4/GLM.4$null.deviance
CS4=1-exp((GLM.4$deviance-GLM.4$null.deviance)/length(rankale2$id))
n4=CS4/(1-(exp(-(GLM.4$null.deviance/length(rankale2$id)))))
CS4
n4


pre4<-predict(object=GLM.4,newdata=rankale2, type='response')
rankale2$premod4<-ifelse(test= pre4>0.5,yes=1,no=0)
rankale2$acim4=mapply(poracierto, rankale2$resbin, rankale2$premod4)
table(rankale2$acim4)/length(rankale2$id)
##0.68 aciertos
```

```
# as y minimo y maximo
GLM.5<- glm(resbin ~ aspareja1+minelo1+maxelo1+minelo2+maxelo2, family=binomial(logit), data=rankale2)
summary(GLM.5)
exp(coef(GLM.5))
CHI5=GLM.5$null.deviance-GLM.5$deviance
CHI5
hl5=CHI5/GLM.5$null.deviance
CS5=1-exp((GLM.5$deviance-GLM.5$null.deviance)/length(rankale2$id))
n5=CS5/(1-(exp(-(GLM.5$null.deviance/length(rankale2$id)))))
CS5
n5
pre5<-predict(object=GLM.5,newdata=rankale2, type='response')
rankale2$premod5<-ifelse(test= pre5>0.5,yes=1,no=0)
rankale2$acim5=mapply(poracierto, rankale2$resbin, rankale2$premod5)
table(rankale2$acim5)/length(rankale2$id)
#0.68 de aciertos, igual que el anterior, no merece la pena separar el maximo y el minimo

#as tres y elo
GLM.6<- glm(resbin ~ aspareja1+ trespareja1 + difelopar, family=binomial(logit), data=rankale2)
```

```
summary(GLM.6)
exp(coef(GLM.6))
CHI6=GLM.6$null.deviance-GLM.6$deviance
CHI6
hl6=CHI6/GLM.6$null.deviance
CS6=1-exp((GLM.6$deviance-GLM.6$null.deviance)/length(rankale2$id))
n6=CS6/(1-(exp(-(GLM.6$null.deviance/length(rankale2$id)))))
CS6
n6
pre6<-predict(object=GLM.6,newdata=rankale2, type='response')
rankale2$premod6<-ifelse(test= pre6>0.5,yes=1,no=0)
rankale2$acim6=mapply(poracierto, rankale2$resbin, rankale2$premod6)
table(rankale2$acim6)/length(rankale2$id)
#0.7035 de acierto



#elo dif as tres y veinte
GLM.7<- glm(resbin ~ aspareja1+trespareja1+difeveintes+
difelopar, family=binomial(logit), data=rankale2)
summary(GLM.7)
exp(coef(GLM.7))
CHI7=GLM.7$null.deviance-GLM.7$deviance
CHI7

hl7=CHI7/GLM.7$null.deviance
hl7
CS7=1-exp((GLM.7$deviance-GLM.7$null.deviance)/length(rankale2$id))
n7=CS7/(1-(exp(-(GLM.7$null.deviance/length(rankale2$id)))))
CS7
n7

pre7<-predict(object=GLM.7,newdata=rankale2, type='response')
rankale2$premod7<-ifelse(test= pre7>0.5,yes=1,no=0)
rankale2$acim7=mapply(poracierto, rankale2$resbin, rankale2$premod7)
table(rankale2$acim7)/length(rankale2$id)
#0.7592136 de aciertos
```

```
#curva ROC del modelo
rankale2$probpred7 <- fitted(GLM.7)

plot(roc(rankale2$resbin, rankale2$probpred7,smooth = TRUE) )


#elo as tres y cuarenta y 20, modelo final
GLM.8<- glm(resbin ~ aspareja1+trespareja1+difeveintes
+cuaren+difelopar, family=binomial(logit), data=rankale2)
summary(GLM.8)
exp(coef(GLM.8))
CHI8=GLM.8$null.deviance-GLM.8$deviance
CHI8
hl8=CHI8/GLM.8$null.deviance
CS8=1-exp((GLM.8$deviance-GLM.8$null.deviance)/length(rankale2$id))
n8=CS8/(1-(exp(-(GLM.8$null.deviance/length(rankale2$id)))))
```

```
CS8
n8
pre8<-predict(object=GLM.8,newdata=rankale2, type='response')
rankale2$premod8<-ifelse(test= pre8>0.5,yes=1,no=0)
rankale2$acim8=mapply(poracierto, rankale2$resbin, rankale2$premod8)
table(rankale2$acim8)/length(rankale2$id)
#76.58

rankale2$probpred8 <- fitted(GLM.8)

#curva ROC y area bajo la curva
plot(roc(rankale2$resbin, rankale2$probpred8,smooth = TRUE, print.auc=TRUE) )
auc(roc(rankale2$resbin, rankale2$probpred8,smooth = TRUE))


##area bajo la curva de todos los modelos
rankale2$probpred1 <- fitted(GLM.1)
auc(roc(rankale2$resbin, rankale2$probpred1,smooth = FALSE))
rankale2$probpred2 <- fitted(GLM.2)
auc(roc(rankale2$resbin, rankale2$probpred2,smooth = FALSE))
rankale2$probpred3 <- fitted(GLM.3)
auc(roc(rankale2$resbin, rankale2$probpred3,smooth = TRUE))
rankale2$probpred20 <- fitted(GLM.20)
auc(roc(rankale2$resbin, rankale2$probpred20,smooth = TRUE))
rankale2$probpredelo <- fitted(GLM.elo)
auc(roc(rankale2$resbin, rankale2$probpredelo,smooth = TRUE))
rankale2$probpred4 <- fitted(GLM.4)
auc(roc(rankale2$resbin, rankale2$probpred4,smooth = TRUE))
rankale2$probpred6 <- fitted(GLM.6)
auc(roc(rankale2$resbin, rankale2$probpred6,smooth = TRUE))
rankale2$probpred7 <- fitted(GLM.7)
auc(roc(rankale2$resbin, rankale2$probpred7,smooth = TRUE))
rankale2$probpred8 <- fitted(GLM.8)
auc(roc(rankale2$resbin, rankale2$probpred8,smooth = TRUE))



##medias de los residuos de los modelos mas completos
library(arm)
binnedplot(fitted(GLM.8),
           residuals(GLM.8, type = "response"),
           nclass = 150,
           xlab = "Expected Values",
           ylab = "Average residual",
           main = "Binned residual plot",
           cex.pts = 0.8,
           col.pts = 1,
           col.int = "gray")



binnedplot(fitted(GLM.7),
           residuals(GLM.7, type = "response"),
```

```
            nclass = 150,
            xlab = "Expected Values",
            ylab = "Average residual",
            main = "Binned residual plot",
            cex.pts = 0.8,
            col.pts = 1,
            col.int = "gray")
```

```
##MODELOS DESCARTADOS

# as y minimo y maximo porcentajes
GLM.9<- glm(resbin ~ aspareja1+minpor1+minpor2+maxpor1+maxpor2,
family=binomial(logit), data=rankale2)
summary(GLM.9)
exp(coef(GLM.9))
CHI9=GLM.9$null.deviance-GLM.9$deviance
CHI9
hl9=CHI9/GLM.9$null.deviance
CS9=1-exp((GLM.9$deviance-GLM.9$null.deviance)/length(rankale2$id))
n9=CS9/(1-(exp(-(GLM.9$null.deviance/length(rankale2$id)))))
CS9
n9
pre9<-predict(object=GLM.9,newdata=rankale2, type='response')
rankale2$premod9<-ifelse(test= pre9>0.5,yes=1,no=0)
rankale2$acim9=mapply(poracierto, rankale2$resbin, rankale2$premod9)
table(rankale2$acim9)/length(rankale2$id)
#malisimo
#lo mismo que el que solo tiene el as, 0.66


#modelo con porcentaje de victorias  minimo y maximo + as
GLM.10<- glm(resbin ~  minpor1+maxpor1+aspareja1, family=binomial(logit), data=rankale2)
summary(GLM.10)
exp(coef(GLM.10))
CHI10=GLM.10$null.deviance-GLM.10$deviance
CHI10
hl10=CHI10/GLM.10$null.deviance
CS10=1-exp((GLM.10$deviance-GLM.10$null.deviance)/length(rankale2$id))

n10=CS10/(1-(exp(-(GLM.10$null.deviance/length(rankale2$id)))))
CS10
n10
pre10<-predict(object=GLM.10,newdata=rankale2, type='response')


rankale2$premod10<-ifelse(test= pre10>0.5,yes=1,no=0)


rankale2$acim10=mapply(poracierto, rankale2$resbin, rankale2$premod10)
table(rankale2$acim10)/length(rankale2$id)
#igual que el del as
```

```
##vamos a ver si quitando el tres no perdemos nada de prediccion
GLM.11<- glm(resbin ~ aspareja1+cuaren+minelo1+maxelo1+minelo2+maxelo2,
```

```
family=binomial(logit), data=rankale2)
summary(GLM.11)
exp(coef(GLM.11))
CHI11=GLM.11$null.deviance-GLM.11$deviance
CHI11


hl11=CHI11/GLM.11$null.deviance
hl11
CS11=1-exp((GLM.11$deviance-GLM.11$null.deviance)/length(rankale2$id))
n11=CS11/(1-(exp(-(GLM.11$null.deviance/length(rankale2$id)))))
CS11
n11

pre11<-predict(object=GLM.11,newdata=rankale2, type='response')


rankale2$premod11<-ifelse(test= pre11>0.5,yes=1,no=0)


rankale2$acim11=mapply(poracierto, rankale2$resbin, rankale2$premod11)
table(rankale2$acim11)/length(rankale2$id)
# quitando el tres en el modelo 6 se falla un 2% mas



reselo<-function(el,elo){
  dife=elo-el
  return(dife)
}
rankale2$difelo1=mapply(reselo, rankale2$minelo1, rankale2$maxelo1)
rankale2$difelo2=mapply(reselo, rankale2$minelo2, rankale2$maxelo2)

#diferencias de elo en la pareja
GLM.12<- glm(resbin ~ aspareja1+trespareja1+cuaren+difelo1+difelo2,
family=binomial(logit), data=rankale2)
summary(GLM.12)
exp(coef(GLM.12))
CHI12=GLM.12$null.deviance-GLM.12$deviance
CHI12


Difgrlib12=GLM.12$df.null-GLM.12$df.residual

Difgrlib12#2
hl12=CHI12/GLM.12$null.deviance
hl12
CS12=1-exp((GLM.12$deviance-GLM.12$null.deviance)/length(rankale2$id))

n12=CS12/(1-(exp(-(GLM.12$null.deviance/length(rankale2$id)))))
CS12
n12
CHI12
```

```
pre12<-predict(object=GLM.12,newdata=rankale2, type='response')


rankale2$premod12<-ifelse(test= pre12>0.5,yes=1,no=0)


rankale2$acim12=mapply(poracierto, rankale2$resbin, rankale2$premod12)
table(rankale2$acim12)/length(rankale2$id)
```

```
#añadimos los numero de partidas, salen no significativos
GLM.13<- glm(resbin ~ aspareja1+trespareja1+cuaren+elopar1+elopar2+numpar1+numpar2
+numpar3+numpar4, family=binomial(logit), data=rankale2)
summary(GLM.13)
exp(coef(GLM.13))
CHI13=GLM.13$null.deviance-GLM.13$deviance
CHI13

hl13=CHI13/GLM.13$null.deviance
hl13
CS13=1-exp((GLM.13$deviance-GLM.13$null.deviance)/length(rankale2$id))
CS13
pre13<-predict(object=GLM.13,newdata=rankale2, type='response')


rankale2$premod13<-ifelse(test= pre13>0.5,yes=1,no=0)


rankale2$acim13=mapply(poracierto, rankale2$resbin, rankale2$premod13)
table(rankale2$acim13)/length(rankale2$id)

#######MODELOS CON MUCHISIMAS MAS VARIABLES
GLM.14<- glm(resbin ~ aspareja1+trespareja1+cuaren+elopar1+elopar2+minpor1+minpor2
+maxpor1+maxpor2, family=binomial(logit), data=rankale2)
summary(GLM.14)
exp(coef(GLM.14))
CHI14=GLM.14$null.deviance-GLM.14$deviance
CHI14
hl14=CHI14/GLM.14$null.deviance
hl14
CS14=1-exp((GLM.14$deviance-GLM.14$null.deviance)/length(rankale2$id))
CS14

pre14<-predict(object=GLM.14,newdata=rankale2, type='response')

rankale2$premod14<-ifelse(test= pre14>0.5,yes=1,no=0)

rankale2$acim14=mapply(poracierto, rankale2$resbin, rankale2$premod14)
table(rankale2$acim14)/length(rankale2$id)
##76.18414%
```

```
GLM.15<- glm(resbin ~ aspareja1+trespareja1+cuaren+minelo1+minelo2+maxelo1+maxelo2
+minpor1+minpor2+maxpor1+maxpor2, family=binomial(logit), data=rankale2)
summary(GLM.15)
exp(coef(GLM.15))
```

```
pre15<-predict(object=GLM.15,newdata=rankale2, type='response')

rankale2$premod15<-ifelse(test= pre15>0.5,yes=1,no=0)

rankale2$acim15=mapply(poracierto, rankale2$resbin, rankale2$premod15)
table(rankale2$acim15)/length(rankale2$id)
##76.18697%

GLM.16<- glm(resbin ~ aspareja1+trespareja1+cuaren+minpor1+minpor2+maxpor1+maxpor2,
family=binomial(logit), data=rankale2)
summary(GLM.16)
exp(coef(GLM.16))

pre16<-predict(object=GLM.16,newdata=rankale2, type='response')

rankale2$premod16<-ifelse(test= pre16>0.5,yes=1,no=0)

rankale2$acim16=mapply(poracierto, rankale2$resbin, rankale2$premod16)
table(rankale2$acim16)/length(rankale2$id)


GLM.17<- glm(resbin ~aspareja1+trespareja1+cuaren+minelo1+minelo2+minpor1+minpor2
+maxpor1+maxpor2 , family=binomial(logit), data=rankale2)
summary(GLM.17)
exp(coef(GLM.17))

pre17<-predict(object=GLM.17,newdata=rankale2, type='response')


rankale2$premod17<-ifelse(test= pre17>0.5,yes=1,no=0)


rankale2$acim17=mapply(poracierto, rankale2$resbin, rankale2$premod17)
table(rankale2$acim17)/length(rankale2$id)
###75.24615
```

```
GLM.18<- glm(resbin ~aspareja1+trespareja1+cuaren+minelo1+minelo2+porc1+porc2+porc3+porc4 ,
family=binomial(logit), data=rankale2)
summary(GLM.18)
exp(coef(GLM.18))

pre18<-predict(object=GLM.18,newdata=rankale2, type='response')


rankale2$premod18<-ifelse(test= pre18>0.5,yes=1,no=0)


rankale2$acim18=mapply(poracierto, rankale2$resbin, rankale2$premod18)
table(rankale2$acim18)/length(rankale2$id)
#75.1813%
```

Modelos para solucionar el problema del modelo completo

Presentamos el código que se ha utilizado para dar solución al problema del modelo completo con las cuarenta.

```
##modelos con solo las partidas con 40
GLM.401<- glm(resbin ~ aspareja1+trespareja1+difveintes+difelopar,
family=binomial(logit), data=rankale40)
summary(GLM.401)
pre401<-predict(object=GLM.401,newdata=rankale40, type='response')


rankale40$premod401<-ifelse(test= pre401>0.5,yes=1,no=0)


rankale40$acim401=mapply(poracierto, rankale40$resbin, rankale40$premod401)
table(rankale40$acim401)/length(rankale40$resbin)
#64.56

GLM.402<- glm(resbin ~ aspareja1+trespareja1+difveintes+cuaren+difelopar,
family=binomial(logit), data=rankale40)
summary(GLM.402)
pre402<-predict(object=GLM.402,newdata=rankale40, type='response')


rankale40$premod402<-ifelse(test= pre402>0.5,yes=1,no=0)


rankale40$acim402=mapply(poracierto, rankale40$resbin, rankale40$premod402)
table(rankale40$acim402)/length(rankale40$resbin)
#88.7
```

```
rankale40$probpred402 <- fitted(GLM.402)
plot(roc(rankale40$resbin, rankale40$probpred402,smooth = TRUE, print.auc=TRUE) )
auc(roc(rankale40$resbin, rankale40$probpred402,smooth = TRUE))
binnedplot(fitted(GLM.402),
           residuals(GLM.402, type = "response"),
           nclass = 200,
           xlab = "Expected Values",
           ylab = "Average residual",
           main = "Binned residual plot",
           cex.pts = 0.8,
           col.pts = 1,
           col.int = "gray")
```

Código para la conclusión

```
##157.63 para IGUALAR el efecto del as
##121.19 para el del tres
##216.1 cuarenta
diferenciadepareja<-function(elop1,elop2){
  if(elop1>elop2+216.1){
    return(1)}
  if(elop1<elop2+216.1){
    return(0)
```

```
  }
}
rankale2$dif<-mapply(diferenciadepareja, rankale2$elopar1, rankale2$elopar2)

table(rankale2$dif)/length(rankale2$id)

difmedias<-function(elop1,elop2){
  dife=elop1-elop2
  return(dife)
}
rankale2$difelo<-mapply(difmedias, rankale2$elopar1, rankale2$elopar2)
mean(rankale2$difelo)
sd(rankale2$difelo)

quantile(x=rankale2$difelo, probs=c(0.9,0.95, 0.99, 0.999, 1))
hist(x =rankale2$difelopar )
quantile(rankale2$difelo, probs=c(0.99976))
```

Para encontrar las diferencias necesarias simplemente se divide el coeficiente del suceso entre el de la diferencia de elo.

```
GLM.elo<- glm(resbin ~difelopar, family=binomial(logit), data=rankale2)

probmodelo<- function (difel){
  p=exp(0.000722+ 0.0114*difel)/(1+exp(0.000722+ 0.0114*difel))

  return(p)
}

rankale2$p_partida=mapply(probmodelo,rankale2$difelopar)
probbinomneg<-function(p){
  prob=p^3+3*p^3*(1-p)+6*p^3*(1-p)^2

  return(prob)
}

rankale2$p_coto=mapply(probbinomneg, rankale2$p_partida)
plot(rankale2$difelopar,rankale2$p_partida,main = "Modelo regresión logística",
ylab = "P(resbin=1)",
     xlab = "Diferencia de elo")
points(rankale2$difelopar,rankale2$p_coto, col = "red")
```

Código para elaborar la última gráfica.

Anexo C: Salidas más importantes de R

En este anexo tenemos las salidas más importantes que nos ha dado R a lo largo del trabajo. En algunos momentos se puede ver que los jugadores 1 y 3 tienen contabilizados mayor cantidad de sucesos que los otros dos. Lo que se debe a que el registro entra en la base de datos cuando algún suceso le ocurre al jugador, por lo tanto los que más sucesos tienen son los primeros jugadores en aparecer, es decir, los primeros de cada pareja.

También afecta a las parejas, la pareja 1 suele cantar más cuarentas, tener más ases y treses que la 2, pero esto no es un problema porque porque a la hora de hacer los modelos las parejas estan puestas de forma aleatoria.

```
partidas <- read.csv('partidas.csv')
```

Análisis descriptivo

```
table(partidas$cuarenta_i)
```

```
##
##       0       1       2       3       4
## 1348180  279847   39480   67136    8254
```

```
table(partidas$cuarenta_i)/length(partidas$id)*100
```

```
##
##          0          1          2          3          4
## 77.3528212 16.0564279  2.2651941  3.8519775  0.4735793
```

279847+39480+67136+8254=394717, que es el número de partidas en las que se han cantado las 40, un 22.65% aproximadamente.

```
veinte1<- partidas$veinteg1
veinte2<- partidas$veinteg2
veinte3<- partidas$veintep1
veinte4<- partidas$veintep2

matvein <- cbind(veinte1, veinte2, veinte3, veinte4)

numveint<- apply(X=matvein, MARGIN=1, FUN=sum ) #es el numero de veintes que se cantan en cada partida
table(numveint)
```

```
## numveint
##      0      1      2      3
## 956231 628607 145398  12661
```

```
mean(numveint)
```

```
## [1] 0.5493073
```

```
sd(numveint)
```

```
## [1] 0.676758
```

Datos reflejados en la tabla 2.2

```
tresyas<- function(as,tres){
  if (as == 1){
    if(tres < 3)
      {return(1)}
    if (tres > 2)
      {return(0)}}
  if (as == 2){
    if(tres < 3)
     { return(1)}
    if(tres > 2)
      {return(0)}}
  if (as == 3){
    if(tres > 2)
      {return(2)}
    if(tres < 3)
      {return(0)}}
  if (as == 4){
    if(tres > 2)
      {return(2)}
    if (tres < 3)
      {return(0)}}
}
partidas$nueva_columna2=mapply(tresyas,partidas$as_i,partidas$tres_i)
table(partidas$nueva_columna2)
```

```
##
##      0      1      2
## 891521 596147 255229
```

851376 (596147+255229) partidas en las que una misma pareja tiene as y tres.

```
#tresyasjugador<- function(as,tres){
# if(as==tres)
# {return(as)}
#  else
#  {return(0)}
#}
#partidas$nueva_columna=mapply(tresyasjugador,partidas$as_i,partidas$tres_i)
table(partidas$nueva_columna)
```

```
##
##       0       1       2       3       4
## 1337553  221472   61999  109694   12179
```

Ejecutando la función y aplicandola como se muestra, obtenemos que 405344 (221472+61999+109694+12179) en las que el mismo jugador ha tenido as y tres.

```
partasytresjugador<-partidas[partidas$nueva_columna!=0,]
ganadorcontresyas<-function(resultado,ncol){
  if(ncol==1){
    if (resultado==1)
    {return(1)}
    if (resultado==2)
    {return(0)}
  }
```

```
  if(ncol==2){
    if (resultado==1)
    {return(1)}
    if (resultado==2)
    {return(0)}
  }
  if(ncol==3){
    if (resultado==1)
    {return(0)}
    if (resultado==2)
    {return(1)}
  }
  if(ncol==4){
    if (resultado==1)
    {return(0)}
    if (resultado==2)
    {return(1)}
  }
}


partasytresjugador$nueva_columna2=mapply(ganadorcontresyas,partasytresjugador$resultado,partasytresjuga

table(partasytresjugador$nueva_columna2)
```

```
##
##      0      1
##  84020 321324
```

```
table(partasytresjugador$nueva_columna2)
```

```
##
##      0      1
##  84020 321324
```

Dentro de las partidas en las que un jugador tiene as y tres, vemos que gana su pareja 321324 veces.

```
tresyasdistju<- function(as,tres){
  if (as == 1){
    if(tres ==2)
    {return(12)}
    if (tres != 2)
    {return(0)}}
  if (as == 2){
    if(tres == 1)
    { return(21)}
    if(tres !=1)
    {return(0)}}
  if (as == 3){
    if(tres == 4)
    {return(34)}
    if(tres !=4)
    {return(0)}}
  if (as == 4){
    if(tres ==3)
```

```
    {return(43)}
    if (tres != 3)
    {return(0)}}
}
partidas$nueva_columna4=mapply(tresyasdistju,partidas$as_i,partidas$tres_i)


table(partidas$nueva_columna4)
```

```
##
##       0      12      21      34      43
## 1296865  152983  159693   64951   68405
```

En 152983 partidas el jugador 1 tiene el as y el 2 el tres. En 159693 el jugador 2 el as y el 1 el tres. En 64951 el 3 el as y el 4 el tres. En 68405 el 4 el as y el 3 el tres.

Sumando todas tenemos que hay 446032 partidas que el as y el tres está en la misma pareja pero distinto jugador.

```
partasytresdistju<-partidas[partidas$nueva_columna4!=0,]
#victdistju<-function(ncol,resul){
#  if(ncol<25){
#    if(resul==1){
#      return(1)}
#    if(resul==2){
#      return(0)
#    }}
#  if(ncol>25){
#    if(resul==1){
#      return(0)}
#    if(resul==2){
#      return(1)
#    }
#  }

#}
#partasytresdistju$nueva_columna=mapply(victdistju,partasytresdistju$nueva_columna4,partasytresdistju$r

table(partasytresdistju$nueva_columna)
```

```
##
##      0
## 446032
```

343043 victorias para las parejas con as y tres en distintos jugadores frente a 102989 derrotas.

En caso de querer saber los porcentajes en vez de el número de partidas, añadimos /length(partidas$id)*100 detras de la orden table, para que nos de el porcentaje sobre el total de partidas.

Resumenes de los modelos

En orden, todos los modelos presentes en la meoria del trabajo.

```
rankale2 <- read.csv('rankale2.csv')
```

```
GLM.1 <- glm(resbin ~ aspareja1, family=binomial(logit), data=rankale2)
summary(GLM.1)
```

```
##
## Call:
## glm(formula = resbin ~ aspareja1, family = binomial(logit), data = rankale2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4679  -0.9118  -0.9118   0.9124   1.4686
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.662757   0.003471  -190.9   <2e-16 ***
## aspareja1    1.323955   0.004908   269.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025582  on 739800  degrees of freedom
## Residual deviance:  948767  on 739799  degrees of freedom
## AIC: 948771
##
## Number of Fisher Scoring iterations: 4
```

```
exp(coefficients(GLM.1))
```

```
## (Intercept)   aspareja1
##   0.5154282   3.7582577
```

```
GLM.2<- glm(resbin ~ aspareja1+trespareja1, family=binomial(logit), data=rankale2)
summary(GLM.2)
```

```
##
## Call:
## glm(formula = resbin ~ aspareja1 + trespareja1, family = binomial(logit),
##     data = rankale2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7405  -1.1024  -0.7044   1.1029   1.7409
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.267368   0.004748  -266.9   <2e-16 ***
## aspareja1    1.445189   0.005176   279.2   <2e-16 ***
## trespareja1  1.088479   0.005176   210.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025582  on 739800  degrees of freedom
## Residual deviance:  902164  on 739798  degrees of freedom
## AIC: 902170
##
```

```
## Number of Fisher Scoring iterations: 4
```

```
exp(coef(GLM.2))
```

```
## (Intercept)   aspareja1 trespareja1
##   0.2815718   4.2426559   2.9697544
```

```
GLM.20<- glm(resbin ~ aspareja1+trespareja1+difeveintes, family=binomial(logit), data=rankale2)
summary(GLM.20)
```

```
##
## Call:
## glm(formula = resbin ~ aspareja1 + trespareja1 + difeveintes,
##     family = binomial(logit), data = rankale2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2069  -1.1014  -0.5043   1.1019   2.2074
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.304606   0.004808 -271.32   <2e-16 ***
## aspareja1    1.484782   0.005244  283.14   <2e-16 ***
## trespareja1  1.123068   0.005233  214.62   <2e-16 ***
## difeveintes  0.346737   0.003551   97.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025582  on 739800  degrees of freedom
## Residual deviance:  892408  on 739797  degrees of freedom
## AIC: 892416
##
## Number of Fisher Scoring iterations: 4
```

```
exp(coef(GLM.20))
```

```
## (Intercept)   aspareja1 trespareja1 difeveintes
##   0.2712793   4.4140026   3.0742719   1.4144450
```

```
GLM.3<- glm(resbin ~ aspareja1+trespareja1+cuaren+difeveintes, family=binomial(logit), data=rankale2)
summary(GLM.3)
```

```
##
## Call:
## glm(formula = resbin ~ aspareja1 + trespareja1 + cuaren + difeveintes,
##     family = binomial(logit), data = rankale2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0524  -0.8316  -0.1075   0.8324   3.2114
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.635240   0.005576  -293.2   <2e-16 ***
## aspareja1    1.849078   0.006059   305.2   <2e-16 ***
```

```
## trespareja1  1.419014   0.005979   237.3   <2e-16 ***
## cuaren       2.517071   0.008375   300.5   <2e-16 ***
## difeveintes  0.499164   0.003977   125.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025582  on 739800  degrees of freedom
## Residual deviance:  751742  on 739796  degrees of freedom
## AIC: 751752
##
## Number of Fisher Scoring iterations: 5
```

```
exp(coef(GLM.3))
```

```
## (Intercept)   aspareja1 trespareja1      cuaren difeveintes
##   0.1949055   6.3539585   4.1330419  12.3922474   1.6473432
```

```
GLM.elo<- glm(resbin ~difelopar, family=binomial(logit), data=rankale2)
summary(GLM.elo)
```

```
##
## Call:
## glm(formula = resbin ~ difelopar, family = binomial(logit), data = rankale2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5637  -1.0961  -0.3773   1.0961   2.4337
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 7.215e-04  2.436e-03   0.296    0.767
## difelopar   1.137e-02  4.638e-05 245.149   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025582  on 739800  degrees of freedom
## Residual deviance:  956730  on 739799  degrees of freedom
## AIC: 956734
##
## Number of Fisher Scoring iterations: 4
```

```
exp(coef(GLM.elo))
```

```
## (Intercept)   difelopar
##    1.000722    1.011436
```

```
GLM.4<- glm(resbin ~ aspareja1+difelopar, family=binomial(logit), data=rankale2)
summary(GLM.4)
```

```
##
## Call:
## glm(formula = resbin ~ aspareja1 + difelopar, family = binomial(logit),
##     data = rankale2)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7700  -0.9788  -0.2932   0.9788   2.6811
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.648e-01  3.633e-03  -183.0   <2e-16 ***
## aspareja1    1.330e+00  5.147e-03   258.5   <2e-16 ***
## difelopar    1.143e-02  4.863e-05   235.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025582  on 739800  degrees of freedom
## Residual deviance:  886058  on 739798  degrees of freedom
## AIC: 886064
##
## Number of Fisher Scoring iterations: 4
```

```
exp(coef(GLM.4))
```

```
## (Intercept)   aspareja1   difelopar
##   0.5143597   3.7820860   1.0114932
```

```
GLM.6<- glm(resbin ~ aspareja1+ trespareja1 + difelopar, family=binomial(logit), data=rankale2)
summary(GLM.6)
```

```
##
## Call:
## glm(formula = resbin ~ aspareja1 + trespareja1 + difelopar, family = binomial(logit),
##     data = rankale2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7589  -0.9445  -0.2264   0.9445   2.7916
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.273e+00  4.948e-03  -257.3   <2e-16 ***
## aspareja1    1.453e+00  5.411e-03   268.4   <2e-16 ***
## trespareja1  1.095e+00  5.400e-03   202.8   <2e-16 ***
## difelopar    1.148e-02  5.011e-05   229.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025582  on 739800  degrees of freedom
## Residual deviance:  842766  on 739797  degrees of freedom
## AIC: 842774
##
## Number of Fisher Scoring iterations: 4
```

```
exp(coef(GLM.6))
```

```
## (Intercept)   aspareja1 trespareja1   difelopar
##   0.2799183   4.2740447   2.9892837   1.0115443
```

```
GLM.7<- glm(resbin ~ aspareja1+trespareja1+difeveintes+difelopar,
family=binomial(logit), data=rankale2)
summary(GLM.7)
```

```
##
## Call:
## glm(formula = resbin ~ aspareja1 + trespareja1 + difeveintes +
##     difelopar, family = binomial(logit), data = rankale2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7925  -0.9320  -0.2091   0.9313   2.8070
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.311e+00  5.012e-03 -261.68   <2e-16 ***
## aspareja1    1.493e+00  5.482e-03  272.38   <2e-16 ***
## trespareja1  1.130e+00  5.460e-03  207.05   <2e-16 ***
## difeveintes  3.515e-01  3.703e-03   94.94   <2e-16 ***
## difelopar    1.150e-02  5.045e-05  228.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025582  on 739800  degrees of freedom
## Residual deviance:  833549  on 739796  degrees of freedom
## AIC: 833559
##
## Number of Fisher Scoring iterations: 4
```

```
exp(coef(GLM.7))
```

```
## (Intercept)   aspareja1 trespareja1 difeveintes   difelopar
##   0.2694324   4.4514824   3.0969316   1.4212516   1.0115711
```

```
GLM.8<- glm(resbin ~ aspareja1+trespareja1+difeveintes+cuaren+difelopar,
family=binomial(logit), data=rankale2)
summary(GLM.8)
```

```
##
## Call:
## glm(formula = resbin ~ aspareja1 + trespareja1 + difeveintes +
##     cuaren + difelopar, family = binomial(logit), data = rankale2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3989  -0.7624  -0.0605   0.7623   3.4404
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -1.647e+00  5.798e-03  -284.1   <2e-16 ***
## aspareja1    1.863e+00  6.322e-03   294.7   <2e-16 ***
## trespareja1  1.431e+00  6.224e-03   229.9   <2e-16 ***
## difeveintes  5.056e-01  4.142e-03   122.1   <2e-16 ***
## cuaren       2.547e+00  8.701e-03   292.7   <2e-16 ***
## difelopar    1.177e-02  5.571e-05   211.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1025582  on 739800  degrees of freedom
## Residual deviance:  701472  on 739795  degrees of freedom
## AIC: 701484
##
## Number of Fisher Scoring iterations: 5
```

```
exp(coef(GLM.8))
```

```
## (Intercept)   aspareja1 trespareja1 difeveintes      cuaren   difelopar
##   0.1925753   6.4457748   4.1825001   1.6580131  12.7660201   1.0118355
```

Además, para obtener el porcetaje de aciertos en cada modelo, lo hacemos con este código (para el GLM.8, si se quiere el de otro modelo simplemente se sustituye GLM.8 por el nombre del modelo deseado):

```
poracierto<-function(res,predic){
  if(res==predic)
  {return(1)}
  if(res!=predic)
  {return(0)}
}
pre8<-predict(object=GLM.8,newdata=rankale2, type='response')


rankale2$premod8<-ifelse(test= pre8>0.5,yes=1,no=0)


rankale2$acim8=mapply(poracierto, rankale2$resbin, rankale2$premod8)
table(rankale2$acim8)/length(rankale2$id)
```

```
##
##         0         1
## 0.2341616 0.7658384
```