UNIVERSITAT POLITÈCNICA DE CATALUNYA

FACULTAT D'INFORMÀTICA DE BARCELONA

Proyecto Final de Carrera

ESTRATEGIA EXPLOTADORA EN JUEGOS DE PÓQUER

Autor: Marc Velmer
Director: Lluís Belanche

Enero 2016

Índice

Motivación ..... - 3 -

Introducción ..... - 6 -

El póquer: ¿Azar o estrategia? ..... - 6 -

La importancia de la banca ..... - 8 -

El delicado ecosistema ..... - 10 -

El juego del póquer ..... - 12 -

El juego del póquer ..... - 12 -

Reglas básicas ..... - 12 -

Acciones ..... - 12 -

Rondas de apuestas ..... - 13 -

La modalidad del Texas Hold'em ..... - 14 -

Posiciones ..... - 14 -

Rondas de apuestas y desarrollo de una mano ..... - 16 -

Ranking de jugadas ..... - 19 -

Alcance del proyecto ..... - 21 -

Parte 1: La información ..... - 22 -

El "edge" en el póquer ..... - 22 -

La digitalización de la información ..... - 24 -

Los "trackers" ..... - 25 -

Estructura de la base de datos ..... - 27 -

Extraer la información (I) ..... - 29 -

Extraer la información (II) ..... - 35 -

Estructura de la base de datos del proyecto ..... - 39 -

Parte 2: La estrategia ..... - 43 -

La importancia del cálculo ..... - 43 -

Situaciones del juego para estudiar ..... - 46 -

RFI (Raise First In) ..... - 46 -

Facing 3-bet (Recibiendo una subida) ..... - 52 -

Facing RFI ..... - 56 -

Polarización de los rangos ..... - 62 -
Parte 3: La simulación ..... - 66 -
La simulación ..... - 66 -
Crear una simulación ..... - 68 -
Funcionamiento interno de la simulación ..... - 70 -
Funciones específicas de la estrategia explotadora ..... - 74 -
Evaluación de la estrategia ..... - 78 -
Almacenaje de las simulaciones ..... - 82 -
Conclusiones ..... - 83 -
Actualidad y futuro ..... - 87 -
Costes ..... - 88 -
Análisis económico ..... - 88 -
Diagrama de Gannt ..... - 89 -
Referencias ..... - 91 -
Información general ..... - 91 -
Información de software ..... - 91 -
Literatura ..... - 91 -

Motivación

El objetivo de este proyecto es estudiar el comportamiento de jugadores de póquer online a través de la información que obtenemos de su historial de juego y obtener una forma de explotar sus errores. Analizaremos los datos de cada jugador para generar un tipo de estadísticas que serán útiles para la toma de decisiones futuras, creando así una estrategia basada en el cálculo estadístico que permita tomar decisiones basadas en el valor esperado de cada acción. Finalmente pondremos a prueba los resultados obtenidos y analizaremos el resultado de las simulaciones.

El resultado final que pretendemos obtener es un software capaz de gestionar toda esta información y realizar los cálculos correctamente.

El peso importante de este proyecto reside en captar, gestionar, almacenar y mostrar toda la información que necesitaremos para después nutrir de datos una estrategia de juego que vamos a crear.

Para el diseño de la estrategia es inevitable conocer las bases más elementales del juego, por esto dedicaremos la introducción de este documento a explicar, en varias partes, el auge del póquer online y de por qué puede ser un juego perfectamente rentable. Como pretendemos mostrar una imagen fidedigna de lo que es el panorama actual del juego, hablaremos sobre el ecosistema sobre el cual se sustenta el juego hoy en día, cuales son los principales actores y que implicaciones tiene para nuestros objetivos. Aquí veremos justificada la necesidad de tener información del rival para tomar decisiones que nos den una ventaja respecto a los rivales.

Después de esta introducción, explicaremos el propio juego del póquer. Esta parte es una pequeña guía para entender el funcionamiento de una partida, las normas, las rondas de apuestas, los valores de las manos y las jugadas, etc. y no es de necesaria lectura en caso de conocer las normas del juego.

Seguidamente empezaremos con la descripción del software realizado para este proyecto. A partir de aquí, el documento está dividido en tres partes.

En la primera parte nos centraremos en lo más importante del proyecto, que consiste en recopilar la información de los jugadores y el tratamiento posterior.

Todo el proyecto se basa en datos reales, en jugadores en activo actualmente en las salas más populares online. Para ello, hemos comprado en Internet los historiales de partidas necesarios para tener suficiente información y realizar un análisis lo menos sesgado

posible, ya que hay ciertos tipos de datos que necesitan miles, decenas de miles e incluso cientos de miles de muestras para converger a un valor que podamos considerar correcto para hacer cualquier cálculo.

Es vital entender que trataremos con grandes cantidades de información y que los procesos que seleccionemos tienen que ser óptimos para no generar ningún error con datos duplicados o erróneos, por ejemplo. Para ello usaremos un software especial diseñado para jugadores de póquer online y nos ayudaremos de su estructura interna para obtener estos datos de una manera más eficaz y ahorrarnos mucho trabajo. Esta decisión añade una capa de abstracción muy necesaria para completar nuestro software final y es importante porque requiere usar los recursos y datos de un proyecto que no es nuestro, que no está documentado y no es de acceso libre para el usuario, con la problemática que esto conlleva.

Los datos que vayamos a recopilar los organizaremos en una base de datos propia. Explicaremos cómo funciona el proceso que se encargará de recolectar los datos desde la base de datos del software, tratarlos y organizarlos. Esta información es primordial para el proyecto, ya que luego servirá tanto para hacer los cálculos de la estrategia como para mostrarlos en el software final.

La visualización de los datos se realiza a través una aplicación web, donde se presentan los datos de cada rival de una forma totalmente adaptada para el jugador de póquer profesional. Veremos como teniendo la información necesaria, podemos visualizar una especie de radiografía sobre los comportamiento de juego de los rivales. Para ello usaremos un servidor web, donde tendremos la base de datos propia y la aplicación completa del proyecto.

En la segunda parte del proyecto entraremos de lleno en la creación de una estrategia y empezaremos introduciendo las ventajas que obtenemos al realizar correctamente los cálculos en el juego. Veremos cómo afecta a nuestra estrategia el hecho de tener una cantidad enorme de datos y cómo podemos usarlos para actuar en consecuencia.

Para ello, vamos a desglosar tres situaciones de estudio que son las más representativas del juego. Analizaremos la relación entre los datos y el estilo de juego de los rivales y explicaremos los métodos de cálculo necesarios para definir la estrategia desde un punto de vista teórico.

Usaremos los campos de la base de datos que hemos recolectado en la primera parte para justificar el cálculo del valor esperado de cada acción en cada situación de estudio.

Pero será en la tercera parte donde realizaremos una simulación de la estrategia del proyecto con los rivales que tenemos. Esta simulación se realizará en la misma aplicación

web, desde donde podremos introducir todos los datos iniciales y visualizar la gráfica de cada resultado, así como obtener el historial de simulaciones realizadas.

En el documento hablaremos de cómo hemos programado la simulación y traduciremos, paso a paso, de la parte teórica a la parte práctica, explicando una toma de decisión en pseudocódigo, para ilustrar el funcionamiento interno de la estrategia.

Usaremos diferentes datos de configuración para la simulación y analizaremos unas cuantas ejecuciones realizadas con el software y veremos cómo se comporta la estrategia del proyecto jugando contra otros rivales.

Esto nos dará paso a las conclusiones finales, donde analizaremos las decisiones que hemos ido tomando a lo largo de todo el proyecto con tal de conseguir nuestros objetivos. También haremos una reflexión sobre el uso de esta estrategia en la actualidad y que retos plantea una futura estrategia más amplia que la programada aquí.

Finalmente tenemos una estimación de los costes, junto con un diagrama de Gantt y enlaces a las referencias usadas durante todo el proyecto.

Introducción

El póquer: ¿Azar o estrategia?

En el año 2009 se fundó la Federación Internacional de Póquer. Es una organización sin ánimo de lucro que cumple con dos objetivos. El primero es promover el desarrollo en todo el mundo del póquer y asegurar su reconocimiento como *un deporte mental basado en habilidades estratégicas*. En abril del 2010 lo consigue y la Federación pasa a ser miembro de la Asociación Internacional de Deportes Mentales (IMSA). El segundo objetivo es poder ayudar a los miembros de la Federación a obtener un marco de legislación justo en los países para que los jugadores puedan jugar de una manera segura y legal, tanto en casinos como en el juego online.

Muchos jugadores profesionales de póquer han sido grandes jugadores de ajedrez o de otro tipo de juegos similares. No es una mera casualidad, puesto que en la más pura esencia de ambos juegos, el concepto es el mismo: La toma de decisiones estratégicas. Tanto en un juego como en el otro, cada acción que realicemos nosotros como nuestro rival afecta al juego posterior. Solo cuando hemos finalizado, podemos trazar una línea hacia atrás y ver como cada decisión ha afectado al juego global.

Obviamente siguen habiendo grandes diferencias, pero quizás, la más importante no sea que en una se usa un tablero y que en la otra se usen cartas. La gran diferencia es, que mientras el ajedrez es un juego descubierto, el póquer no. Si observamos una partida de ajedrez, en cada momento sabremos que está sucediendo, mientras que quizás nos será imposible entender algunos movimientos por la psicología y la complejidad de la partida. Pero a diferencia del póquer, toda la información está disponible en cada momento, desde el inicio hasta el final. En el póquer esto no es así. Hay ciertas cartas que permanecen ocultas hasta el final de la partida. Esta ocultación de la información hace que nuestra decisión no pueda basarse en una información clara, sino en un rango de posibles informaciones. Es decir, al no saber que cartas son las ocultas, nuestra decisión será una ponderación sobre que decisiones serían las correctas dependiendo de qué posibles cartas fueran las ocultas. Esta diferencia hace que en el juego haya un componente de azar. Escoger la decisión que más nos conviene no nos garantiza en el mismo momento del juego que esta sea la correcta. Simplemente tomamos la acción que creemos que tiene más posibilidades de ser la correcta.

Esto tiene muchas implicaciones en cómo se desarrolla el juego. En una partida de ajedrez donde nuestro rival es mejor que nosotros, nuestras posibilidades de ganar son mínimas,

mientras que en el póquer no es así, siempre existe la posibilidad de que un jugador inexperto acabe ganando al mejor jugador de todos. Obviamente es algo que a largo plazo no sucederá, porque a largo plazo minimizamos el efecto del azar en nuestra toma de decisiones. Esto hace que el juego del póquer tenga un atractivo más para la gente inexperta.

El hecho de que exista un componente de suerte en el póquer crea diferentes opiniones en la sociedad. Las más radicales opinan que el azar es el componente básico de este juego, mientras que los grandes jugadores, los más experimentados, están convencidos de que no. Para intentar explicar esto, veamos un ejemplo.

Propongo al lector el siguiente juego. Lanzamos un dado equilibrado. En caso de que salga el número 1 o 2 el lector debe pagarme 1 Euro. En caso de que salga un 3, un 4 o un 5, yo le pagaré al lector 1 Euro. En caso de salir el 6, se repite la tirada.

Es fácil ver como este juego es un regalo. El 100% de los lectores deberían querer jugar a este juego sin duda alguna, puesto que a largo plazo, la ganancia está asegurada. Eso no implica que no haya un factor de azar. Podría ser que en las primeras 10 tiradas, los números fuesen 1, 2, 1, 6, 4, 5, 1, 2, 6, 3 y el lector sale perdiendo. A pesar de esto, deberíamos querer jugar siempre y el máximo número de veces, puesto que el juego es sencillo y sabemos ver claramente la ventaja: Nosotros ganamos en 3 números, nuestro adversario en 2 y el otro número simplemente deja una situación neutra. Cualquier persona es capaz de ver esta *ventaja*, simplemente porque el juego es *simple*. Si complicamos el juego, esta simplicidad desaparece, pero no olvidemos que, en última instancia, todo esto son cálculos.

Este ejemplo es vital para entender el proyecto que se expone. Primero, no hablaremos de azar. El "azar" es muy ambiguo. Es un término confuso y no expresa nada concreto. Hablaremos de **varianza** y de **valor esperado**. En el anterior ejemplo, hemos visto que nuestro *valor esperado* es positivo (a largo plazo, en el infinito, ganaremos). Pero la *varianza* puede hacernos estar en negativo en diferentes situaciones. El azar genera varianza, pero la varianza es calculable y transitoria. Segundo, hablaremos de "ventaja" en el sentido más completo de la palabra anglosajona "**edge**", que es definida como "competitive advantage". En el anterior juego, el lector tenía una ventaja competitiva frente a su rival: Había un número más que beneficiaba sus intereses.

La importancia de la banca

Algunas veces me he encontrado en la situación de tener que explicar a alguien porque no debería intentar ganar dinero en según qué juegos. Absolutamente todos se pueden estudiar y explicar matemáticamente, aunque en algunos casos sea un trabajo tedioso, ya sea por la dinámica del juego o por la gran magnitud de posibilidades o ramificaciones que el propio juego plantea.

Una manera muy fácil de explicarlo es usar la pura lógica, sin usar cálculo alguno. Tan fácil como plantearse la pregunta: ¿Existe una banca? En el caso de que la respuesta sea afirmativa, el juego, para nosotros, no puede ser rentable a largo plazo. Esta afirmación es muy rotunda, pero tiene un gran sentido. ¿Qué interés tiene la banca en dejarnos jugar a un juego donde puede perder dinero? ¿Por qué la banca ofrecerá un espacio en forma de casino con trabajadores cobrando un sueldo y pagando sus impuestos? ¿Por qué nos invitará a bebidas si ve que estamos jugando grandes cantidades de dinero? La respuesta cae por su propio peso: La banca siempre tiende a ganar dinero. Y dado que ese dinero que gana tiene que servir para lucrarse, así como para pagar sus gastos de explotación, debe haber alguien que pierde ese dinero. El perdedor es el jugador que juega contra la banca. Si esto no fuese así, los casinos no existirían. No podrían porque sería insostenible. Este es su negocio.

Pero para algunos, creerse esta afirmación puede resultar un acto de fe, así que vamos a analizar uno de los juegos más sencillos que encontramos en la mayoría de casinos del mundo: La ruleta francesa.

Se supone que el gran encanto de la ruleta es que su tapete ofrece a los jugadores muchas modalidades de apuestas aparentemente diferentes, con premios de variada cantidad, y con la posibilidad de que con una sola ficha se pueda apostar agrupando la eventual ocurrencia de varias casillas numeradas del tapete, lo cual ciertamente en cada caso implica un aumento de las probabilidades de acertar porque con una sola ficha se apuesta a la aparición de un mayor número de resultados de los 37 que tiene disponibles.

Como el objetivo de este proyecto no es explicar los diferentes tipos de apuesta de la ruleta, nos centraremos básicamente en los dos más conocidos: *El pleno* y el *rojo o negro*.

Las normas son estas: cuando apostamos al pleno, digamos que con 1 euro, escogemos un solo número. Pueden darse dos escenarios. Si sale nuestro número, la banca nos paga la cantidad apostada multiplicada por 35. En caso contrario, perdemos nuestra apuesta.

Como la ruleta tiene 37 números (36 más el 0), 36 veces de 37 perderemos 1 euro. Cuando sale nuestro pleno, ganamos 35 veces. El cálculo es sumamente sencillo.

$$Valor\ esperado = \left(-1 * \frac{36}{37}\right) + \left(35 * \frac{1}{37}\right) = -\frac{36}{37} + \frac{35}{37} = -\frac{1}{37} = -0.0270$$

En el caso del rojo o negro, apostamos a uno de los dos colores. De los 37 números, el 0 es verde y no se puede apostar (al menos jugando por colores) y de los 36 restantes, hay 18 negros y 18 rojos. Cuando salga nuestro color, la banca nos paga nuestra apuesta. En caso contrario perdemos nuestra apuesta. Hagamos los cálculos.

$$Valor\ esperado = \left(-1 * \frac{19}{37}\right) + \left(1 * \frac{18}{37}\right) = -\frac{19}{37} + \frac{18}{37} = -\frac{1}{37} = -0.0270$$

Lo curioso del estudio de todos los tipos de apuesta posibles en una ruleta, ya sea el pleno, rojo o negro, impar o par, pasa o falta, columna, caballo, docena, etc… es que **el valor esperado es siempre el mismo**. Da igual en que modalidad de apuesta nos encontremos, siempre que juguemos por 1 euro, nuestro retorno es de 1-0.0270 = 0.973 euros.

Dicho de otra forma, en cada jugada **estamos destinados a perder el 2.7% de nuestro capital invertido**. Cualquier persona preocupada por mantener su patrimonio sabrá que perder un 2.7% del capital en tan poco tiempo solo nos puede llevar a la ruina inminente.

Uno de los atractivos del póquer y de porque se puede llegar a ganar dinero jugando a largo plazo es porque no existe una banca. El dinero que nosotros podamos ganar en el juego proviene siempre de otro jugador. El casino (ya sea online o en vivo) cobra una comisión del bote final generado, con independencia de quien sea el ganador.

Esto nos abre una posibilidad enorme cuando hablamos del “edge”, ya que aquí, nuestra ventaja competitiva se basa **en la calidad de las decisiones estratégicas que tomamos respeto al resto de los rivales**. Esto es de vital importancia, pues la diferencia en la calidad de las decisiones no está marcada por ninguna banca, sino por nosotros mismos.

Esto explica porque hay jugadores que son capaces de mantener unas ganancias estables en el largo plazo. Su juego está sometido a la varianza al igual que cualquier otro jugador, pero en el largo plazo premia la toma de decisiones correctas. En una mesa con N jugadores, con estrategias de juego exactamente iguales, el único beneficiado es la sala de juego, puesto que toma un porcentaje de cada partida. La única forma de que haya ganadores, es que su estrategia sea mejor que la del resto de jugadores y que además, lo sea lo suficiente como para batir ese parte que se lleva el proveedor del juego. Hay una frase muy popular en el mundo del póquer que dice que da igual que seamos el séptimo mejor jugador del mundo, que si nos sentamos a jugar contra los seis mejores jugadores vamos a tener serios problemas. Esto es extrapolable a cualquier situación del juego: Es imposible tener éxito a largo plazo si la calidad de nuestras decisiones son peores que las de nuestros rivales.

El delicado ecosistema

En la actualidad, el póquer es un ecosistema muy delicado que requiere ser alimentado constantemente. Esto ha sido históricamente así, pero con la llegada del juego online, hemos visto como este efecto se ha magnificado y por primera vez, ha podido ser cuantificado en cifras. En este ecosistema encontramos tres grandes participantes.

El primero de todos es la propia sala que ofrece la plataforma para jugar. Como hemos comentado anteriormente, los ingresos de los casinos en el póquer provienen de las comisiones que se lleva de los botes que se forman en cada partida.

El segundo grupo engloba a los jugadores recreacionales. Estos jugadores son los más abundantes y suelen jugar como forma de ocio o por las mismas razones que alguien compra lotería o juega a la quiniela.

El tercer grupo engloba los jugadores profesionales, semiprofesionales o “regular players”. Estos jugadores son los que juegan una cantidad significativa de manos y dan “liquidez” a la sala. Esta “liquidez” se materializa en forma de actividad en la sala. Es decir, al jugar una gran cantidad de partidas, estos jugadores dan vida a la sala y gracias a ellos hay mucha oferta de juego en la sala.

Este ecosistema es muy delicado porque cada grupo de participantes cumple su función y nadie puede fallar. Para ver exactamente cómo funciona, debemos mirar los objetivos de cada grupo por separado.

La sala quiere mantener su negocio y obtener un beneficio de ello. Este objetivo es el mismo que en cualquier empresa.

Los jugadores recreacionales quieren poder disfrutar de una gran oferta de juego que se adapte a sus necesidades.

Los jugadores regulares quieren ganar dinero de los jugadores recreacionales.

De esta forma, es fácil ver como la sala necesita a los jugadores regulares para que su negocio sea atractivo, ya que los necesitan para crear oferta de juego. De la misma forma, los jugadores regulares necesitan que la sala atraiga a jugadores recreacionales para que ellos puedan obtener beneficio al jugar contra jugadores menos experimentados. Esto es un tipo de contrato ficticio entre ambas partes. Si las condiciones se cumplen adecuadamente, es una situación de “win-win” para ambos.

Así, la sala invierte dinero en publicitar su producto y atrae a jugadores recreacionales. Estos entran a jugar y obtienen una amplia oferta de juego generada a partir de los

jugadores regulares. Los jugadores regulares obtienen un tráfico de gente menos experimentada y obtienen un beneficio a largo plazo, ya que su objetivo es enfrentarse a rivales de menor nivel. Al existir esta actividad constante de juego, la sala obtiene dinero con las comisiones de cada partida y puede reinvertirlo en publicitar su juego aún más para atraer a más público. Cuanta más gente sea atraída, más juego se genera entre los dos grupos de jugadores y más beneficio obtiene tanto la sala como el jugador regular.

Este ciclo es el que ha hecho funcionar a todas las salas de póquer desde los inicios del juego online. Por eso no es de extrañar que veamos constantemente anuncios en televisión y otros medios atrayendo a nuevos públicos.

La relación entre la sala y el jugador regular es muy delicada, ya que estadísticamente, un jugador regular realiza, de media, muchísimas más partidas que un jugador recreacional. Como la sala obtiene beneficio por cada partida jugada, es obvio que su principal fuente de ingresos es el jugador regular. Por lo tanto, el objetivo de publicitar y atraer a jugadores recreacionales es solo para satisfacer al jugador regular y mantenerlo jugando en su plataforma de juego para así mantener su negocio.

Esta relación es muy parecida a la que mantiene una empresa tradicional con sus accionistas. Si se les cuida, estos no venden su participación en la empresa y se puede seguir explotando el negocio, resultando en una situación de “win-win” para ambos a costa de los consumidores o clientes de la empresa.

Aunque no existen estadísticas oficiales de que porcentajes representa cada grupo de jugadores, ya que esto es una información privada que las empresas no les conviene sacar a la luz, se habla de que aproximadamente entre un 90% y un 95% de los jugadores son recreacionales y el resto son jugadores regulares. Estos últimos, que son minoría, representan el 80% de los beneficios de la sala. Por lo tanto, podemos ver como los números están muy polarizados y que el ecosistema es realmente débil y hay que mantenerlo vivo.

El juego del póquer

El juego del póquer

Vamos a empezar explicando en que consiste el juego del póquer en su esencia. Aunque hay diversas modalidades de póquer, todas ellas comparten unas reglas básicas y son las siguientes.

Reglas básicas

El póquer es un juego donde tenemos una cantidad de fichas que sirven para realizar apuestas. A través de estas apuestas se van formando botes en cada mano. El objetivo del juego es ganar esos botes y aumentar así el número de fichas que disponemos.

Un bote puede únicamente ganarse de dos formas:

1. En una ronda de apuestas, nuestros oponentes abandonan ante alguna de nuestras apuestas.
2. Al finalizar todas las rondas de apuestas, gana el jugador que tenga la mejor mano, según el ranking de jugadas.

En todas las modalidades de póquer, una mano empieza con unas apuestas iniciales obligatorias. En la mayor parte de modalidades estas apuestas se conocen como la ciega pequeña y la ciega grande.

Acciones

En cada ronda de apuestas, tendremos las siguientes opciones cuando sea nuestro turno:

1. Apostar, o subir la apuesta actual.
2. Igualar una apuesta o pasar.
3. Abandonar ante una apuesta.

En algunas situaciones, cuando otro jugador no haya apostado antes que nosotros, en lugar de subir una apuesta podremos apostar, y en lugar de abandonar ante una apuesta podremos pasar. Veamos las distintas acciones una por una.

- **Apostar**: Podremos apostar cuando sea nuestro turno y algún rival no haya apostado antes que nosotros.

- **Subir**: Subiremos una apuesta cuando algún jugador ya haya realizado una apuesta y nosotros queramos apostar una cantidad mayor.
- **Igualar**: Igualaremos una apuesta cuando algún jugador ya haya apostado y nosotros no queramos subir. Deberemos igualar la cantidad que el jugador haya apostado. También podemos igualar la ciega grande si no queremos apostar.
- **Abandonar**: Consiste en retirarse de una mano cuando alguien apuesta.
- **Pasar**: Pasar es una acción que solo tendremos cuando ningún jugador haya apostado previamente y queramos esperar a ver qué hacen el resto de rivales. Si todos los jugadores pasan, llegaremos a la siguiente ronda de apuestas. Si después de que pasemos alguien apuesta, volverá a llegar nuestro turno (por orden) para que decidamos qué hacer, pues debemos tomar una nueva decisión en la misma ronda de apuestas.

Rondas de apuestas

Las apuestas continúan hasta que todos los jugadores han igualado, subido o abandonado ante la apuesta actual. Si ningún jugador apuesta, la ronda se completa cuando todos los jugadores han pasado. Cuando finaliza la ronda de apuestas, continúa la siguiente ronda de reparto y apuestas o se completa la mano y se llega al destape. Si varios jugadores llegan al destape, los jugadores van enseñando sus manos una a una y el jugador que tenga la mejor jugada será el ganador.

No siempre se juegan todas las rondas de apuestas. Si en algún punto de la mano sólo queda un jugador, se le considerará vencedor del bote sin necesidad de mostrar sus cartas, y finalizará la mano. Esto significa que en muchas ocasiones algún jugador realizará una apuesta que nadie querrá igualar y no se llegarán a jugar futuras rondas de apuestas en esa mano.

Si un jugador pone todas tus fichas en juego (situación llamada “all-in”) en alguna ronda de apuestas y se enfrenta a un solo rival o a más de uno que también se encuentre all-in, se sucederán las rondas de reparto hasta llegar al destape sin que haya más apuestas, donde se determinará el vencedor de la mano. Si hay más de un jugador con más fichas una vez otro ha ido all-in, los restantes siguen jugando para ganar un bote paralelo en caso de que realicen apuestas. Si más de un jugador va all-in durante una mano, puede haber más de un bote paralelo. De esta forma se garantiza que ningún jugador puede ganar nunca más fichas de las que tenía en el momento de apostarlo todo, mientras que el resto pueden seguir jugando la mano en un bote distinto.

La modalidad del Texas Hold'em

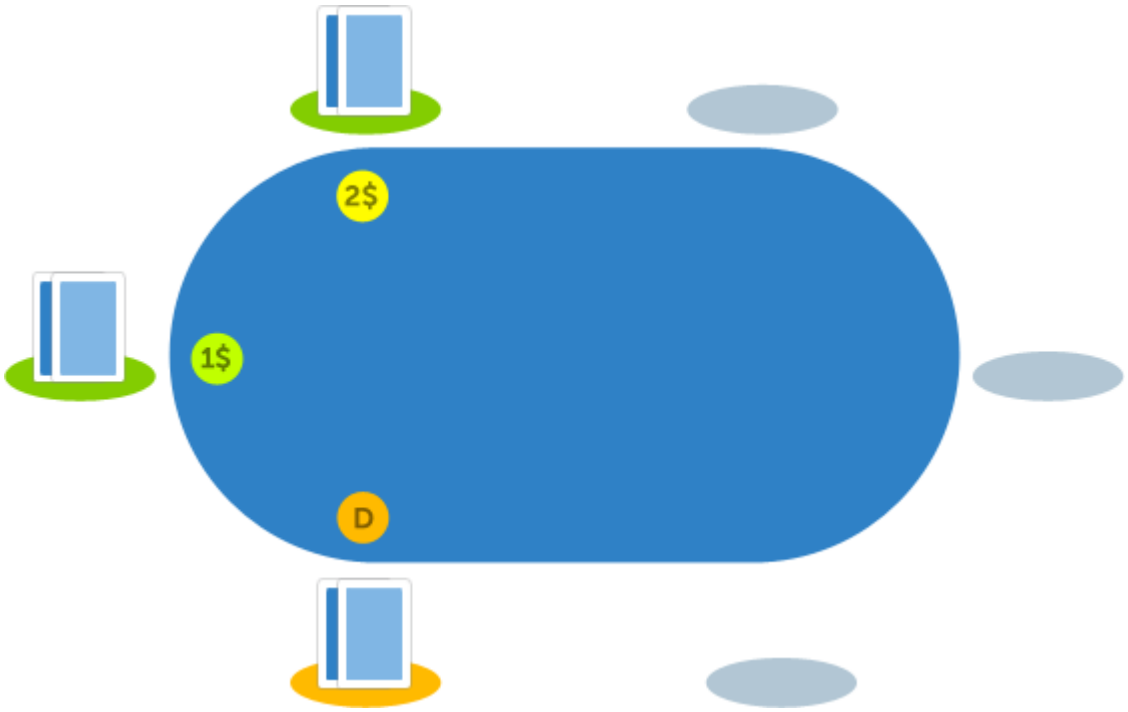
En este proyecto estudiaremos una parte de la modalidad más extendida del póquer: El Texas Hold'em.

El funcionamiento del juego es simple:

- Dos jugadores ponen una apuesta inicial obligatoria, apuestas conocidas como ciegas.
- Se reparten dos cartas tapadas (privadas) a cada jugador.
- Se reparten en la mesa hasta cinco cartas destapadas (llamadas cartas comunitarias) durante las rondas de apuestas.
- Dado que una mano de póquer se compone de cinco cartas, se usará una combinación de las dos cartas tapadas y las cinco que salen en la mesa para formar la jugada.
- En Hold'em, un jugador puede formar su jugada con dos, una o ninguna de sus cartas privadas.

Posiciones

Debemos saber que hay dos tipos de mesa: La mesa corta y la mesa larga. La mesa corta se caracteriza por tener un máximo de 6 jugadores y la mesa larga por tener un máximo de 10 jugadores. En este proyecto nos centraremos en la mesa corta para simplificar las posiciones en la mesa, aunque los cálculos son exactamente los mismos para la mesa larga.



Como podemos apreciar en la imagen, hay 3 posiciones características: La última posición (marcada con una D, en naranja), la ciega pequeña (marcada con 1$, en verde) y la ciega grande (marcada con 2$, en amarillo).

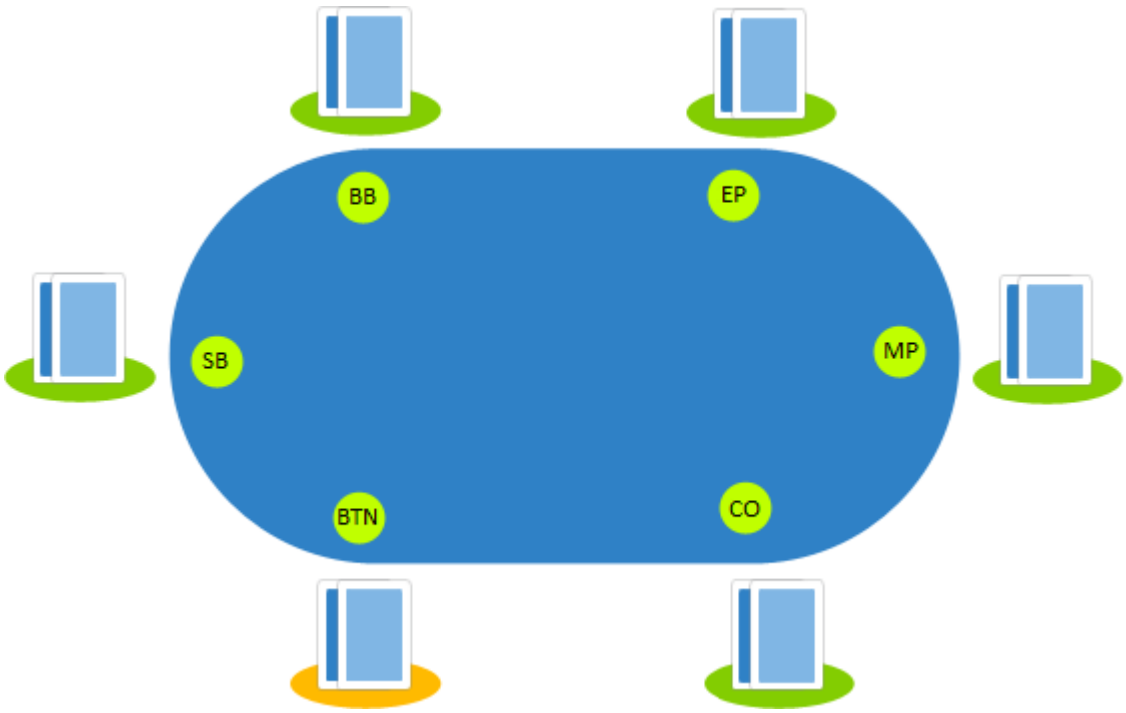
La última posición es la más importante de la mesa ya que marca el orden del resto de posiciones. Justo a su izquierda se encuentra la ciega pequeña y justo después se encuentra la ciega grande. La última posición se desplaza a la izquierda un lugar al finalizar cada mano jugada. Por lo tanto, la ciega pequeña será última posición en la siguiente mano; el jugador que estaba en última posición estará ahora en penúltima posición, y así sucesivamente.

La ciega pequeña es la posición inmediatamente a la izquierda de la última posición. El jugador que esté sentado en la ciega pequeña está obligado a poner una "apuesta a ciegas". De ahí su nombre.

La ciega grande es la posición inmediatamente a la izquierda de la ciega pequeña y por lo tanto, dos sitios a la izquierda de la última posición. Al igual que la ciega pequeña, la ciega grande es una posición donde el jugador tiene que realizar una "apuesta a ciegas". El valor de esta apuesta dependerá del tipo de mesa que se esté jugando, y siempre es el doble que el de la ciega pequeña.

El resto de posiciones son variables, pues depende de la cantidad de jugadores sentados en la mesa.

Para este proyecto, supondremos que tenemos una mesa corta llena, con 6 jugadores, y nombraremos las posiciones como en la siguiente imagen.



La primera posición (**EP**) es "Early Position" y es el primer jugador en tomar una decisión en la primera ronda de apuestas. Seguidamente tenemos la segunda posición (**MP**) o "Middle Position", la tercera posición (**CO**) o "Cut-off", el repartidor (**BTN**) es el "Dealer" o "Button" y finalmente tenemos la ciega pequeña (**SB**, de "Small Blind") y la ciega grande (**BB**, de "Big Blind"). Los nombres de **CO**, **BTN**, **SB** y **BB** permanecen intactos sea cual sea la cantidad de jugadores. Las posiciones que van aumentando cuando hay más de 6 jugadores en la mesa son **EP** y **MP** y se van renombrando con números (**MP1**, **MP2**, etc…)

Rondas de apuestas y desarrollo de una mano

Ahora vamos a ver cómo se desarrolla una mano de póquer en Texas Hold'em y las fases de las que se compone.

En primer lugar, una vez los jugadores están situados en sus posiciones en la mesa, se reparten las cartas. El reparto empieza por la ciega pequeña, que recibe una carta. Después, el jugador en la ciega grande recibe su primera carta, y así sucesivamente para el resto, en sentido horario hasta que todos reciben dos cartas. En el Texas Hold'em hay dos fases de juego con un total de cuatro rondas de apuestas. Sus nombres son de origen anglosajón, pero es importante aprenderlos ya que no existe una traducción válida para ellos. Las dos fases y sus rondas de apuestas correspondientes son:

- **Fase Preflop**: tiene una única ronda de apuesta, por lo que simplemente se denomina Preflop.
- **Fase Postflop**: se compone de tres rondas de apuestas, que son Flop, Turn, y River.

Vamos a cada fase y ronda un poco más a fondo:

Preflop

El Preflop es la primera ronda de apuestas y es la única que estudiaremos en este proyecto. **Es la ronda en la que sólo contamos con nuestras dos cartas privadas.** Su nombre viene de *"Pre-flop"* o *"Antes del Flop"*, dado que el Flop es la siguiente ronda de apuestas.

Una vez se han repartido las cartas y se han puesto las apuestas obligatorias (ciega pequeña y ciega grande), empieza la partida. **El primer jugador en realizar su acción en el Preflop será el que está situado a la izquierda de la posición de ciega grande**. Este jugador tendrá 3 opciones:

- **Subir**: el jugador puede subir la apuesta de la ciega grande, ya que es la apuesta más alta en la mesa actualmente. Debe subir como mínimo al doble de la ciega grande, y como máximo a todas sus fichas.
- **Igualar**: el jugador puede igualar la ciega grande.
- **Abandonar**: el jugador puede tirar sus cartas y retirarse si su mano no le parece suficientemente buena. Al abandonar, dejará de jugar esa mano.

Una vez escoja su acción el primer jugador, será el turno del jugador sentado a su izquierda, y así sucesivamente. Todos los jugadores tienen las mismas opciones, excepto el jugador en la ciega grande que tiene la opción de "pasar" si nadie ha subido su apuesta.

La primera ronda de apuestas o Preflop finaliza cuando todos los jugadores han elegido sus acciones y se han igualado todas las apuestas de la mesa.

Postflop

Como hemos dicho, se compone de tres rondas de apuestas, que vemos con detalle a continuación:

Flop

A partir de este punto, el primer jugador en hablar ya no será el jugador a la izquierda de la ciega grande, sino el jugador situado en la ciega pequeña, o si ya no está en la mano, el siguiente a su izquierda, y así sucesivamente.

En esta ronda de apuestas se muestran 3 cartas en el centro de la mesa, que serán comunitarias para todos los jugadores. Es decir, con nuestras dos cartas privadas y esas 3 cartas comunes componemos nuestra jugada en esta ronda de apuestas.

A partir de esta ronda de apuestas, la apuesta mínima siempre será igual al tamaño de la ciega grande. Al no estar las ciegas como sucedía preflop, los jugadores tienen la opción de pasar si otro jugador no ha apostado antes en esta ronda. Del mismo modo que en la ronda anterior, se procede a una nueva ronda de apuestas. La ronda de apuestas finaliza cuando todos los jugadores han hablado y todas las apuestas se han igualado. Si todos los jugadores deciden pasar, también se acaba la ronda de apuestas y se pasa a la siguiente.

Turn

Llegados al Turn, **se muestra una cuarta carta comunitaria** que también formará parte de nuestra jugada. Ahora disponemos de cuatro cartas comunes y dos privadas, lo que hacen un total de seis cartas disponibles. Con el mismo procedimiento que en el Flop, comienza una nueva ronda de apuestas con los jugadores restantes.

River

La última ronda de apuestas es el River. En esta ronda **se muestra una quinta y última carta comunitaria** que podremos utilizar para nuestra jugada. Del mismo modo que en el Turn, en el River nuestra jugada será la más alta de todas las posibles con las siete cartas de las que disponemos.

Al igual que se hizo en el Flop y en el Turn, comienza una ronda de apuestas con el mismo procedimiento. Una vez termine, se enseñan las cartas de todos los jugadores que hayan llegado hasta el final y se comprueba quién tiene la jugada más alta según el ranking de jugadas.

Ranking de jugadas

Una vez se destapen las cartas, gana el jugador que tenga la jugada más alta. Para saber cuál es esa jugada, mostramos la clasificación de jugadas posibles ordenadas de mayor a menor valor:

Escalera real

Cinco cartas consecutivas del mismo palo hasta el As. 

Escalera de color

Cinco cartas consecutivas del mismo palo. 

Póquer

Cuatro cartas del mismo rango. 

Full

Tres cartas del mismo rango y dos cartas de otro rango. 

Color

Cinco cartas del mismo palo no consecutivas. 

Escalera

Cinco cartas consecutivas pero no del mismo palo. 

Trío

Tres cartas del mismo rango. 

Dobles parejas

Dos cartas de un rango y dos cartas de otro rango. 

Pareja

Dos cartas del mismo rango. 

Carta alta

Carta de mayor rango si no tenemos una jugada mejor. 

Cuando dos jugadores tienen la misma jugada, gana el que tiene la jugada de mayor rango. Por ejemplo, la jugada  , dobles parejas de reyes y dieces, es superior a  , dobles parejas de damas y jotas. Lo mismo sucede para el resto de jugadas: En el caso del color, gana el que tenga la carta más alta del color. En la escalera, por ejemplo, la jugada  , escalera al 9, gana a  , escalera al 8.

Cuando dos jugadores tienen la misma jugada del mismo rango, gana el que tiene las cartas de desempate de mayor rango. Esto solo sucede con las jugadas que se forman con menos de 5 cartas, como el póquer (4), el trío (3), dobles parejas (4), pareja (2) y carta alta (1). Por ejemplo, la jugada  , dobles parejas de reyes y jotas, es superior a  , ya que el  es de rango superior al  .

Alcance del proyecto

En este proyecto nos centraremos únicamente en el análisis de la primera ronda de apuestas (Preflop) en donde los jugadores únicamente tienen sus dos cartas privadas, es decir, aquellas que se mantienen ocultas hasta el final de la mano. Esto nos facilita el análisis por dos razones.

Primero, el cálculo de las posibles combinaciones de manos se reduce notablemente, ya que no tenemos que combinarlas con las cartas comunitarias en rondas posteriores del juego. Esto nos lleva a la segunda razón: Las posibles acciones de los jugadores se basan únicamente en sus cartas comunitarias y las decisiones del resto de jugadores en la misma fase del juego. Si quisiéramos ir más lejos y analizar las siguientes rondas de apuestas, tendríamos que tener en cuenta todas las acciones tomadas anteriormente, lo cual complica los cálculos exponencialmente.

El estudio de una mano completa requeriría un tiempo exageradamente grande para la finalidad de este proyecto, además del uso de estadísticas muy avanzadas que necesitan miles de manos para converger en valores que puedan ser considerados estadísticamente correctos.

El objetivo final de este proyecto es encontrar un juego óptimo en la primera ronda de apuestas y ser un punto de partida para una posible estrategia en rondas posteriores.

Para simplificar aún más supondremos algunas variables del juego estáticas para no complicar los cálculos. Estas simplificaciones se irán introduciendo más adelante.

Parte 1: La información

El “edge” en el póquer

Anteriormente hemos realizado hincapié en la importancia de tener una ventaja competitiva frente a nuestros rivales para desarrollar una estrategia ganadora a largo plazo. Esta ventaja sólo se puede conseguir tomando mejores decisiones que nuestros jugadores.

Una decisión puede tomarse en base a muchos factores. Por ejemplo, y como hemos visto antes, hay aspectos estadísticos que nos permiten acotar nuestra decisión al milímetro, de acuerdo a cuanto nos cuesta seguir pagando una apuesta en relación al premio que esperamos obtener. Al fin y al cabo, estos cálculos son un simple ejercicio de riesgo/recompensa.

Pero en el póquer no toda la información es visible. En todas las fases del juego nos enfrentamos a rivales que tienen dos cartas privadas que desconocemos. Esto hace que el cálculo de las decisiones sea mucho más complejo, ya que el abanico de posibles combinaciones de cartas que desconocemos es amplio y, además, hay que ponderarlo, ya que no siempre es igual de probable tener una combinación que otra.

Por tanto, la información del tipo de juego que realiza nuestro rival es brutalmente importante, ya que nos dará una información vital para poder estrechar el rango de cartas privadas que posee.

La esencia del juego se basa precisamente en este último factor. Conocer todas las decisiones que toma un jugador en base a sus cartas nos permite tomar decisiones mucho más ricas en información que hacerlo sin ella.

Para ver la importancia de conocer al jugador, pongamos un ejemplo.

Imaginemos que nos enfrentamos a un jugador que, aunque nosotros no lo sepamos, decide jugarse todas sus fichas en la primera ronda de apuestas, sea cual sea su combinación de cartas privadas.

En la primera mano nuestro rival lo apuesta todo. Nosotros no lo conocemos y, por lo tanto, no podemos afirmar nada de esta acción. Puede ser que tenga una mano muy buena y por eso decide apostar todo su dinero. Nosotros tomaremos una decisión precavida, ya que su movimiento nos desconcierta.

En la segunda mano nuestro rival lo vuelve a apostar todo. Esta vez nosotros tenemos una información importante. Su decisión ha sido exactamente la misma que la anterior. Podría ser que nuestro rival volviese a tener una muy buena mano, aunque es estadísticamente más difícil que esto suceda. Por si acaso, volvemos a ser precavidos.

En la tercera mano nuestro rival, nuevamente, vuelve a apostar todo.

Podríamos seguir así las veces que quisiéramos, pero lo importante es ver como sus decisiones afectan a las nuestras.

En la primera mano, hemos decidido ser precavidos, en la segunda también pero con ciertas sospechas, y a medida que vemos como su decisión es siempre apostarlo todo, la nuestra se ve condicionada por la suya.

Es decir, cada vez que nosotros nos nutrimos de una información nueva, nuestra decisión de ser precavidos se va desvaneciendo. Cada vez estamos ampliando el rango de manos con el que estamos dispuesto a pagar su apuesta ya que vemos que él lo está haciendo con la totalidad de sus manos. Dicho de otra forma, una mano mediocre con la que no hubiésemos pagado nunca su apuesta en la primera mano, puede convertirse en una mano totalmente válida para pagar en la cuarta o quinta apuesta suya, porque supera con creces la posibilidad de ganar a un rango tan amplio. Pero saber la amplitud del rango de nuestro rival (el 100% de las cartas) nos ha costado varias manos de recopilar información.

Obviamente esto es un ejemplo poco creíble, pero nos sirve para ver de una forma bastante sencilla y lógica, como la información del tipo de juego que realiza un rival es altamente importante para adaptar nuestras decisiones a su juego.

El proyecto consiste en hacer un estudio exhaustivo de la información que obtenemos de los jugadores en la primera ronda de apuestas y a partir de ello, tomar mejores decisiones futuras en nuestro juego basándonos en estrechar los rangos de manos de nuestros rivales.

La digitalización de la información

Como hemos visto antes, la información que obtenemos de las decisiones que toma un jugador en relación a sus cartas privadas son una pieza clave para definir una estrategia de juego.

Con la aparición del póquer online aparece la oportunidad de digitalizar toda esta información que hasta la fecha era imposible realizar. De esta forma, consiguiendo los registros de las acciones de las manos que se juega se puede programar una estructura de datos donde almacenar la información que nos parezca interesante.

Esto es una ventaja muy grande, pues ahora podemos disponer de información exacta de cada jugador y ajustar nuestras decisiones al juego del rival de una forma muy precisa.

En la práctica, esto supone delegar el esfuerzo de memorizar el estilo de juego de un rival y poder centrarse en tomar las decisiones correctas. Con la llegada de esta nueva forma de almacenar la información, han aparecido en el mercado programas, llamados “trackers”, dedicados exclusivamente a esta tarea.

Los “trackers”

Un software “tracker” es un programa, de código privado normalmente, que se vende para registrar la información de las manos de póquer jugadas en diferentes salas.

La totalidad de las salas que ofrecen juegos de póquer online guardan un registro de las manos que jugamos. Estas se guardan en local en el ordenador del jugador y los “trackers” analizan estos archivos y guardan la información en una base de datos, para luego ofrecer diferentes tipos de estadísticas y gráficos al usuario acerca de su juego.

Estos archivos contienen toda la información necesaria acerca del estado inicial de la partida, el estado final, así como todas las acciones y apuestas que han tenido lugar en el transcurso de la mano. La información permite reconstruir, paso a paso, todos los hechos que determinan una partida completa.

A continuación podemos ver un ejemplo de historial. En verde tenemos el estado inicial, en morado las acciones de los jugadores y en naranja el resultado y estado final de la mano.

```
PokerStars Game #27738502010: Tournament #160417133, $0.25+$0.00 Hold'em No Limit -
Level XV (250/500) - 2009/05/02 13:32:38 ET
Table '160417133 3' 9-max Seat #8 is the button
Seat 1: LLC 4Eva (9182 in chips)
Seat 2: 618shooter (25711 in chips) is sitting out
Seat 3: suposd2bRich (21475 in chips)
Seat 4: ElT007 (60940 in chips)
Seat 5: Orlando I (18044 in chips)
Seat 6: ih82bcool2 (8338 in chips)
Seat 7: kovilen007 (8353 in chips)
Seat 8: GerKingTiger (4404 in chips)
Seat 9: Phontaz (23553 in chips)
LLC 4Eva: posts the ante 60
618shooter: posts the ante 60
suposd2bRich: posts the ante 60
ElT007: posts the ante 60
Orlando I: posts the ante 60
ih82bcool2: posts the ante 60
kovilen007: posts the ante 60
GerKingTiger: posts the ante 60
Phontaz: posts the ante 60
Phontaz: posts small blind 250
LLC 4Eva: posts big blind 500
*** HOLE CARDS ***
Dealt to ElT007 [Qd Qc]
618shooter: folds
suposd2bRich: folds
ElT007: raises 2000 to 2500
Orlando I: raises 15484 to 17984 and is all-in
ih82bcool2: folds
kovilen007: calls 8293 and is all-in
GerKingTiger: folds
Phontaz: calls 17734
LLC 4Eva: folds
ElT007: raises 15484 to 33468
Phontaz: calls 5509 and is all-in
Uncalled bet (9975) returned to ElT007
*** FLOP *** [2d 2c 3c]
*** TURN *** [2d 2c 3c] [8h]
*** RIVER *** [2d 2c 3c 8h] [4d]
*** SHOW DOWN ***
```

```
Phontaz: shows [9s 9h] (two pair, Nines and Deuces)
ElT007: shows [Qd Qc] (two pair, Queens and Deuces)
618shooter has returned
ElT007 collected 11018 from side pot-2
Orlando I: shows [5d 5h] (two pair, Fives and Deuces)
ElT007 collected 29073 from side pot-1
kovilen007: shows [Kh As] (a pair of Deuces)
ElT007 collected 34212 from main pot
*** SUMMARY ***
Total pot 74303 Main pot 34212. Side pot-1 29073. Side pot-2 11018. | Rake 0
Board [2d 2c 3c 8h 4d]
Seat 1: LLC 4Eva (big blind) folded before Flop
Seat 2: 618shooter folded before Flop (didn't bet)
Seat 3: suposd2bRich folded before Flop (didn't bet)
Seat 4: ElT007 showed [Qd Qc] and won (74303) with two pair, Queens and Deuces
Seat 5: Orlando I showed [5d 5h] and lost with two pair, Fives and Deuces
Seat 6: ih82bcool2 folded before Flop (didn't bet)
Seat 7: kovilen007 showed [Kh As] and lost with a pair of Deuces
Seat 8: GerKingTiger (button) folded before Flop (didn't bet)
Seat 9: Phontaz (small blind) showed [9s 9h] and lost with two pair, Nines and Deuces
```

Actualmente existe un oligopolio en el mercado de “trackers” privados para jugar a póquer. Existen dos grandes softwares, pero en este proyecto nos vamos a centrar en el más usado, el Hold’em Manager.

La necesidad de usar la información que nos ofrece este software para este proyecto es grande.

La primera razón es la complejidad. El trabajo de analizar y descomponer la información que hay en un historial de una mano es altamente complejo. Para empezar, existen multitud de posibilidades dentro una mano, desde diferentes números de jugadores, rondas de apuestas de longitud desconocida o situaciones externas al juego (desconexiones y time-outs), además de cálculos derivados de la diferencia entre las apuestas de los jugadores. Obviamente todo es posible, pero hacer un “parser” de este tipo equivale a hacer un pequeño compilador de un lenguaje de programación sencillo y nos puede llevar muchísimas horas de trabajo.

La segunda razón y la más importante es que este software nos permite tener una capa de abstracción enorme. Recordemos que hay muchas salas de juego y cada una guarda los historiales de la manera que quiere. Este software unifica toda esta información, sea de la sala que sea, en una misma base de datos. Además, estamos exentos de tener en cuenta las posibles modificaciones que haya en la forma de guardar los datos: Este software tiene miles de actualizaciones a sus espaldas y mantienen al día los “parsers” que analizan los archivos de datos.

De esta forma, nosotros tenemos acceso a una base de datos donde tenemos los historiales de manos guardados de una forma coherente y lo más importante, la base de datos es estable y no contiene ni información errónea ni redundante.

Estructura de la base de datos

Como hemos visto, guardar toda la información de las manos que jugamos no es un trabajo fácil. El primer paso del proyecto consiste en hacer un análisis de cómo queda la base de datos de un software privativo después de introducir miles de manos.

Hold'em Manager usa PostgreSQL para funcionar. Con la instalación del software, se instala también todo el sistema gestor de la base de datos, incluido el programa "pgAdmin III" que nos permitirá acceder de una forma gráfica a la estructura interna de la base de datos de usa el programa.

Como el software es privativo, no tenemos ninguna API pública ni ninguna forma de analizar el código fuente del programa para ver cómo se organiza la información en las diferentes tablas de la base de datos. Un primer vistazo nos muestra como hay 55 tablas, lo cual nos pone las cosas realmente muy cuesta arriba, porque hay cientos de relaciones entre ellas y estudiarlas sin tener ningún tipo de diagrama UML es una tarea complicada.

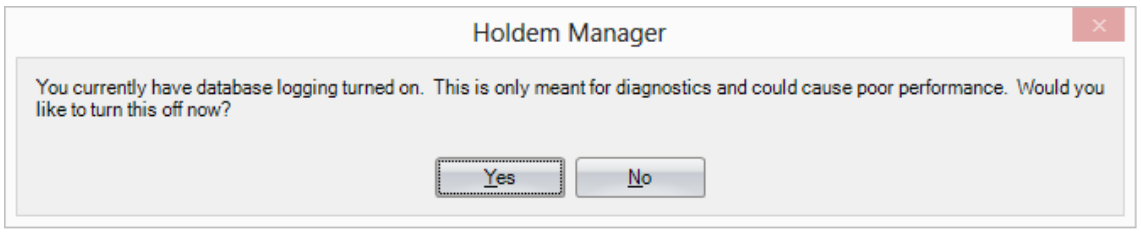
Al analizar los archivos de configuración del programa, podemos encontrar en el archivo **HoldemManager.config** un archivo en formato XML, donde hay la siguiente información:

```
<setting name="LogDatabaseCommands">False</setting>
```

Si modificamos esta línea por

```
<setting name="LogDatabaseCommands">True</setting>
```

Al ejecutar el programa nos avisa de que entramos en un modo debug y que puede afectar al rendimiento.



Decidimos no desactivar el modo "logging" de la base de datos y vemos como en la carpeta "Logs" del programa se crea un archivo llamado **dbtrace.log.**

Al abrir este archivo encontramos todas las sentencias SQL que ejecuta el programa a la base de datos PostgreSQL.

```
05/24/14 10:08:02 HM COMMAND: Select * from Settings limit 1
```

```
05/24/14 10:08:02 HM COMMAND FINISHED: Select * from Settings
limit 1
```

El programa siempre escribe la sentencia SQL dos veces. La primera antes de ejecutarla y la segunda una vez ejecutada.

Esto nos permite ver las consultas que el programa realiza a la base de datos, lo cual es muy útil para ver como el software recoge toda la información para mostrarla.

Tras hacer un filtrado de diferentes situaciones en el programa y analizar las consultas lanzadas por el programa, podemos apreciar como la información esta guardada en una forma que llamaremos “raw”, que no es lo que buscamos.

Llamaremos información “raw” a la información que no está sintetizada en un valor que nos resulte beneficioso para nuestros cálculos. Para que esto quede claro, pongamos un ejemplo. Si queremos saber cuántas veces el jugador X ha recibido una mano Y o ha tomado una acción Z, en la base de datos hemos de ejecutar la consulta filtrando esta situación y obtendremos los registros de cada mano donde ha sucedido la acción que queremos buscar. No obtendremos el valor porcentual respecto a la totalidad de las manos, esta tarea se realiza en el software una vez se han realizado las consultas necesarias.

Por lo tanto, la mejor forma que tenemos para extraer la información de la base de datos es filtrar en el software las situaciones que queremos analizar, interceptar las consultas realizadas y emularlas desde un software propio para comprimir esta información y hacerla fácilmente accesible para nuestro propósito.

Más adelante hablaremos sobre que situaciones vamos a estudiar, pero antes profundizaremos en la primera parte del proyecto que consiste en extraer, analizar y sintetizar la información de la base de datos.

Extraer la información (I)

Al analizar las consultas realizadas por el software a la base de datos nos encontramos con estructuras muy complejas para entender a primera vista. Para ello, vamos a empezar desglosando un poco la forma de guardar la información más básica para así poder entender más adelante que es exactamente lo que estamos preguntando a la base de datos.

Para empezar, buscamos donde tenemos el listado completo de todos los jugadores que están registrados.

Lo encontramos en la tabla *players* y para realizar un primer vistazo, ejecutamos una consulta para ver el contenido completo.

```
SELECT * FROM players;
```

El resultado es el siguiente

	player_id integer	site_id integer	playername text	lastplayeddate timestamp without time zone	cashha intege	tourneyhands integer	playertype_id smallint
1	11308	2	rememb7	2013-09-27 06:25:51	213	0	0
2	160	2	LaDameNoir	2013-09-30 16:24:38	181	0	0
3	4726	2	Gravity8T1	2013-09-21 15:34:44	112	0	0
4	66	2	Nogginthenog	2013-09-30 17:11:56	1413	0	0
5	7810	2	carlos_piane	2013-09-28 02:44:55	31	0	0
6	7807	2	wywrotX	2013-09-28 11:38:19	1114	0	0
7	384	2	RecordMe	2013-09-30 20:00:35	56	0	0
8	154	2	ck007bond	2013-09-30 16:20:45	75	0	0
9	148	2	slugger47	2013-09-30 16:20:25	57	0	0
10	1545	2	evertdelange	2013-10-01 14:59:38	42	0	0
11	3770	2	tragos13	2013-10-04 14:34:11	422	0	0
12	2736	2	lopergolo	2013-10-02 23:04:04	18	0	0
13	903	2	strolek_1972	2013-10-03 18:31:47	20124	0	0

Esta información es importante porque aquí tenemos los identificadores de todos los jugadores, sus nombres y el número de muestras de manos en el campo *cashhands*. De esta forma, podremos analizar a todos los jugadores por individual y filtrar aquellos que no nos resulten interesantes. Para este proyecto ignoraremos a todos los jugadores de los cuales no tengamos una muestra igual o superior a 10000 manos. Así nos aseguraremos que la información que sintetizamos de cada jugador, al menos para la primera etapa del juego, no este sesgada por la varianza.

Si miramos más tablas, encontramos diversa información sobre los tipos de juegos y acciones de los jugadores.

Por ejemplo, en la tabla *gametypes*, seleccionando toda la información nuevamente:

```
SELECT * FROM gametypes;
```

El resultado es el siguiente

	gametype_id integer	gametypedescription character varying(20)	bigblind integer	smallblind integer	ante integer	istourney boolean	pokergametype smallint	currency_id smallint	pokergame smallint
1	1	$1/2 ZOOM NL	200	100	0	f	8	1	1
2	2	$2.5/5 ZOOM NL	500	250	0	f	8	1	1

Toda la información acerca de los niveles o “stakes” de los cuales tenemos información. En nuestra base de datos tenemos información de los niveles donde jugamos con ciegas de 1$ y 2$ y ciegas de 2.5$ y 5$.

Seguimos con la tabla *actiontypes*

```
SELECT * FROM actiontypes;
```

El resultado es el siguiente

	actiontype_id integer	actionstring character(6)
1	-1	
2	0	F
3	1	XF
4	2	XCF
5	3	XCCF
6	4	XCCC
7	5	XCCR
8	6	XCC
9	7	XCRF
10	8	XCRC
11	9	XCCR
12	10	XCR
13	11	XC

En esta tabla tenemos todas las acciones que los jugadores pueden llegar a tomar en cualquier momento de la partida. Las siglas usadas son **F** para “Fold” o Retirarse, **X** para “Check” o Pasar, **C** para “Call” o pagar la apuesta y **R** para “Raise” o resubir la apuesta. No existe ningún otro tipo de acción posible. En esta tabla se guardan todas las combinaciones que se han visto en todas las partidas guardadas, por ejemplo, “XCF” hace referencia a la acción de pasar sin realizar ninguna apuesta (X), seguidamente pagar la apuesta realizada por un jugador (C) y finalmente retirarse (F) ante una apuesta.

Hay muchas más tablas que podemos analizar una a una, pero estas son las más importantes para poder realizar las consultas que necesitamos.

Al seleccionar un jugador, interceptamos la consulta que carga las primeras manos que muestra del jugador.

```
SELECT
ph.pokerhand_id,
ph.preflopplayeractiontype_id,
ph.streetwentallin,
ph.handtimestamp,
gt.gametypedescription,
gt.pokergame,
ph.holecard1int,
ph.holecard2int,
ph.holecard3int,
ph.holecard4int,
pkh.flopcard1int,
pkh.flopcard2int,
pkh.flopcard3int,
pkh.turncardint,
pkh.rivercardint,
ph.flopplayeractiontype_id,
ph.turnplayeractiontype_id,
ph.riverplayeractiontype_id,
ph.netamountwon,
gt.bigblind,
ph.positiontype_id,
ph.preflopaction_id,
ph.didpfr,
ph.didvpip,
ev.sklanskybucks,
ev.equitypct,
getwinningplayerjoincashver2(ph.pokerhand_id) AS
winningplayerjoin
FROM playerhandscashkeycolumns_hero ph
JOIN pokerhands_hero pkh ON pkh.pokerhand_id =
ph.pokerhand_id
JOIN gametypes gt ON gt.gametype_id = ph.gametype_id
LEFT JOIN allinsituations_hero ev
ON ph.playerhand_id = ev.playerhand_id
WHERE (ph.player_id = 3)
AND (ph.gametype_id = 1 OR ph.gametype_id = 2 OR
ph.gametype_id = 3 OR ph.gametype_id = 4 OR ph.gametype_id =
44 OR ph.gametype_id = 70 OR ph.gametype_id = 113)
AND gt.pokergame = 1
AND ph.handTimeStamp BETWEEN to_timestamp('06/05/2012
20:25:23','MM/DD/YYYY HH24:MI:SS') AND
```

```
to_timestamp('09/28/2012 16:23:26','MM/DD/YYYY HH24:MI:SS')
ORDER BY ph.handtimestamp DESC LIMIT 100
```

Como podemos observar, en la parte de selección de los datos, encontramos diferentes campos que usa el software para mostrar los datos de cada registro.

Entre ellos encontramos algunos interesantes que necesitaremos usar:

`ph.preflopplayeractiontype_id:` Define el tipo de acción que ha tomado el jugador en la primera ronda de apuestas (preflop). Este identificador está relacionado con la tabla actiontypes que hemos analizado anteriormente.

`ph.holecard1int, ph.holecard2int:` Define la primera y la segunda carta privada del jugador en cada mano.

`ph.positiontype_id:` Define la posición del jugador en la mesa.

En la parte de las condiciones encontramos también algunos campos interesantes.

`ph.player_id:` Define el identificador del jugador que analizamos. Estos identificadores los obtendremos directamente de la tabla *players*, que ya hemos comentado anteriormente.

`ph.gametype_id:` Define los identificadores de los tipos de juego que queremos filtrar en la consulta. Como en la base de datos del proyecto solo hemos añadido manos compradas de un tipo de juego concreto, podremos obviar esta condición ya que todas las manos son válidas como resultado de la consulta.

`gt.pokergame:` Este dato también podremos ignorarlo porque en la base de datos solo tendremos partidas del tipo “Holdem Texas No Limit” y ninguna de otro juego.

El resto de condiciones limitan el resultado a un lapso concreto de tiempo, a un máximo de resultados o modifican el orden de los resultados, pero para este proyecto no son interesantes.

De esta forma podemos simplificar la consulta que lanza el software a la base de datos y adaptarla a nuestras necesidades.

```
SELECT
ph.preflopplayeractiontype_id,
ph.holecard1int,
ph.holecard2int,
ph.positiontype_id
FROM playerhandscashkeycolumns ph
WHERE (ph.player_id = 3)
```

Con esta consulta simplificada ya tenemos todo lo necesario para analizar al jugador. Por una parte tenemos la acción que realiza el jugador, tenemos las cartas con las que lo hace (si hubiese) y desde que posición realiza el movimiento.

Empezamos realizando una consulta en la base de datos para saber que jugadores tienen más de 10000 muestras. Así nos aseguramos que la mayoría de datos que vamos a extraer han convergido a su valor real.

```
SELECT player_id, playername
FROM players
WHERE cashhands > 10000
```

Aunque no nos interesa realmente saber el identificador del jugador, sino su nombre, lo necesitamos para realizar las futuras consultas. A modo práctico y para evitar confusiones, identificaremos en nuestra base de datos al jugador por el mismo número.

Ahora guardamos este listado en nuestra base de datos para poder iterar sobre cada jugador en concreto y obtener la información que necesitamos.

Más adelante se explicará en profundidad que tipos de situaciones queremos estudiar, pero de momento vamos a requerir la información de las siguientes situaciones que explicamos anteriormente en la tabla *actiontypes:*

- F: El jugador abandona la mano en su turno de juego.
- C: El jugador paga la apuesta inicial (el valor de la ciega grande).
- R: El jugador sube la apuesta inicial (realiza un 2-bet).
- RF: El jugador sube la apuesta y se retira ante otra subida (se retira frente a un 3-bet).
- RC: El jugador sube la apuesta y paga otra subida (paga un 3-bet).
- RR: El jugador sube la apuesta y resube frente a otra subida (realiza un 4-bet cuando recibe un 3-bet).
- RRF: El jugador sube la apuesta, resube frente a otra subida (4-bet) y se retira ante otra resubida (5-bet).
- RRC: El jugador sube la apuesta, resube frente a otra subida (4-bet) y paga ante otra resubida (5-bet). En la práctica esto suele significar que paga el all-in del rival.
- RRR: El jugador sube la apuesta, resube frente a otra subida (4-bet) y resube ante otra resubida (5-bet). En la práctica esto suele significar que paga el all-in del rival.

Como el resultado de las consultas SQL que vamos a ejecutar devuelve un registro por cada mano, tendremos que hacer los cálculos pertinentes en base a la cantidad de resultados.

Por ejemplo, si sumamos todos los posibles casos de F, C y R de un jugador en todas las posiciones, el resultado será el número de manos (muestras) que tenemos del jugador. Esto es lógico, ya que como hemos explicado en las reglas básicas del juego, no se puede realizar ninguna otra acción en el juego.

El resto de situaciones están anidadas dentro de las manos que realizan una subida en la primera ronda de apuestas, las etiquetadas con una R.

La suma de RF, RC y RR de todas las posiciones equivale a R en todas las posiciones. De la misma forma sucede cuando elegimos solo una posición.

Para los casos de RRF, RRC y RRR no sucede exactamente lo mismo. La suma no equivale a los casos de RR y la razón es porque hay casos de RR que son del tipo RRRC, pero en la práctica, tanto RRC como RRR o RRRC son casos altamente improbables y equivalen a una situación donde los rivales apuestan todas sus fichas. Dicho de otra forma, aunque tengan movimientos diferentes, el concepto de apuesta es el mismo: Apostarlo todo. Por lo tanto, en este proyecto las trataremos como casos de RRR, puesto que no hay ninguna diferencia práctica.

Extraer la información (II)

Tal como hemos comentado anteriormente, interceptar las consultas SQL que lanza el programa es el primer objetivo, pero la información que obtendremos de simular estas consultas es básicamente las acciones que realiza el jugador en cada una de las manos que tenemos en la base de datos.

Esta información no nos sirve como tal, porque un listado de acciones no nos sirve para acotar nuestras decisiones en la estrategia que usemos. Así que nuestro segundo objetivo es interpretar estos datos. Este paso es crucial porque el resultado que generemos aquí será almacenado en nuestra base de datos propia y el resto de cálculos posteriores en el proyecto dependerán de la integridad de estos.

Estos cálculos vamos a realizarlos en 3 pasos ejecutando diferentes consultas SQL. Vamos a detallar exactamente qué información queremos obtener al final de cada paso.

1) En el primer paso, nuestro objetivo es obtener el porcentaje de manos con las que el rival realiza un RFI (Raise First In) desde cualquier posición de la mesa. Como buscamos la situación en la que el jugador es el primero en apostar, el estado de la partida debe ser “Unopened” (“UO”, en adelante). Y esto tiene que añadirse a la consulta.

   Una vez tengamos el resultado de la consulta, leeremos todos los registros y clasificaremos los resultados según la acción que realiza el jugador. Para saber cuándo el jugador realiza un RFI, basta con tener en cuenta cuando el jugador realiza un R, RC, RR y RF. Si sumamos estos casos y lo dividimos por el total de registros, obtendremos el buscado porcentaje con el que el jugador realiza un RFI desde la posición desde donde lanzamos la consulta.

   En este paso vamos a generar 10 resultados, dos valores por cada posición de la mesa. El primer valor será el total de manos que tenemos del jugador en esa posición y el segundo será las veces que el jugador realiza un RFI. Llamaremos a estos valores couldRFI_XX y didRFI_XX, respectivamente, siendo XX el acrónimo de la posición (EP, MP, CO, BTN, SB).

   Añadiremos 5 campos adicionales para tener una representación del rango de manos del jugador. En cada registro obtenemos también los valores de cada una de las dos cartas privadas. En la mayoría de casos estos valores estarán vacíos, ya que para conocer la mano del rival necesitamos llegar hasta la última ronda de

apuestas y que haya destape. Esto ocurre en un relativamente pequeño porcentaje de los casos, por lo que trataremos estos rangos con cierta cautela, ya que necesitamos muestras gigantescas para considerar que han convergido correctamente.

Estos campos los llamaremos RangesRFI_XX, siendo XX, otra vez, el acrónimo de la posición en la mesa (EP, MP, CO, BTN, SB). La información la guardaremos en una estructura de datos bastante simple para poder recuperar la información rápidamente.

2) El siguiente paso es calcular los porcentajes de cada acción cuando el jugador se enfrenta a una resubida. Esta situación la conocemos como "Facing 3bet". Como hemos visto anteriormente, cuando nos enfrentamos a un 3bet, solo existen 3 posibles acciones: Retirarse, pagar la apuesta o volver a subir la apuesta. Estas acciones están definidas en los actiontype de la base de datos: Retirarse frente a una subida se identifica como RF (Raise-Fold), pagar la apuesta como RC (Raise-Call) y volver a subir la apuesta como RR (Raise-Raise). Para realizar este paso, podemos usar los resultados de la primera consulta, posición por posición.

   El valor porcentual de, por ejemplo, retirarse frente a una resubida (RF), lo queremos en base a las veces que el jugador realiza un RFI y existe acción por parte de otro jugador. Como hemos visto, el RFI es la suma de R+RC+RR+RF, pero el caso de R representa el escenario en que el jugador realiza un RFI y el resto de jugadores abandona. Como no podemos saber que acción hubiese tomado el jugador en caso de recibir una resubida, ignoraremos para el cálculo las manos que representan R. Por lo tanto, usaremos de denominador común la suma de RF, RC y RR para calcular cada porcentaje.

   En este paso vamos a generar 15 resultados, tres valores por cada posición de la mesa. El primer valor será el total de manos con las que el jugador se retira frente a una resubida (RF), el segundo el total de manos con las que el jugador paga una resubida (RC) y el tercer valor será el total de manos con las que el jugador vuelve a resubir, es decir, realizar un 4bet (RR). Estos valores tomaran el nombre de F3B_XX, C3B_XX y 4B_XX, siendo XX, otra vez, el acrónimo de la posición en la mesa (EP, MP, CO, BTN, SB).

   Adicionalmente añadiremos 10 campos para representar los rangos de manos para cada acción. Como es lógico, no podremos obtener información alguna de la una mano de un jugador cuando este abandona, así que solo podremos aspirar a construir un rango con las acciones RC y RR. Llamaremos a estos campos

RangesC3B_XX y Ranges4B_XX respectivamente, siendo XX el nombre de la posición.

3) En el último paso estudiaremos la situación que hemos llamado “Facing RFI” en las últimas posiciones de la mesa, donde encontramos la “guerra de ciegas” que hemos explicado anteriormente. Hay un interés especial en esta situación, ya que es precisamente aquí donde los rangos de los jugadores son más amplios y, por lo tanto, la información que obtendremos es especialmente valiosa, porque describirá mejor las tendencias del jugador.
   Para ello, vamos a generar una cantidad importante de información, ya que queremos detallar cada acción en cada posición. De las posiciones EP, MP, CO, BTN, SB y BB, nos centraremos en las cuatro últimas: CO, BTN, SB y BB.
   Nuestra finalidad en este paso es ver cómo reacciona el jugador **desde cualquier posición** cuando recibe un RFI **desde cualquier otra posición**.

   Por lo tanto, cuando un jugador realiza un RFI desde, por ejemplo, la primera posición para analizar, CO, los tres jugadores que quedan en BTN, SB y BB pueden realizar 3 acciones cada uno (F, C y R). Como queremos ver el comportamiento detallado, vamos a registrar cada posible situación, así que en este caso, multiplicamos el número de posiciones restantes por el número de acciones (3x3=9) y tendremos toda la información desde la posición CO, donde se ha realizado el RFI.
   Lo mismo sucede para BTN, quedan dos posiciones por tres posibles acciones (2x3=6) y SB, donde queda solo una posición y tres acciones (1x3=3).

   Para nombrar cada dato, usaremos la posición del RFI, la posición del jugador y la acción que realiza, que es la información que identifica cada acción. Esto se ve claramente en la siguiente tabla.

RFI	Posición	Acción F	Acción C	Acción R
CO	BTN	*vsCO_BTN_Fold*	*vsCO_BTN_Call*	*vsCO_BTN_Raise*
	SB	*vsCO_SB_Fold*	*vsCO_SB_Call*	*vsCO_SB_Raise*
	BB	*vsCO_BB_Fold*	*vsCO_BB_Call*	*vsCO_BB_Raise*
BTN	SB	*vsBTN_SB_Fold*	*vsBTN_SB_Call*	*vsBTN_SB_Raise*
	BB	*vsBTN_BB_Fold*	*vsBTN_BB_Call*	*vsBTN_BB_Raise*
SB	BB	*vsSB_BB_Fold*	*vsSB_BB_Call*	*vsSB_BB_Raise*

   En cada campo guardaremos las veces que el jugador realiza la acción y así lo tendremos separado por diferentes campos.

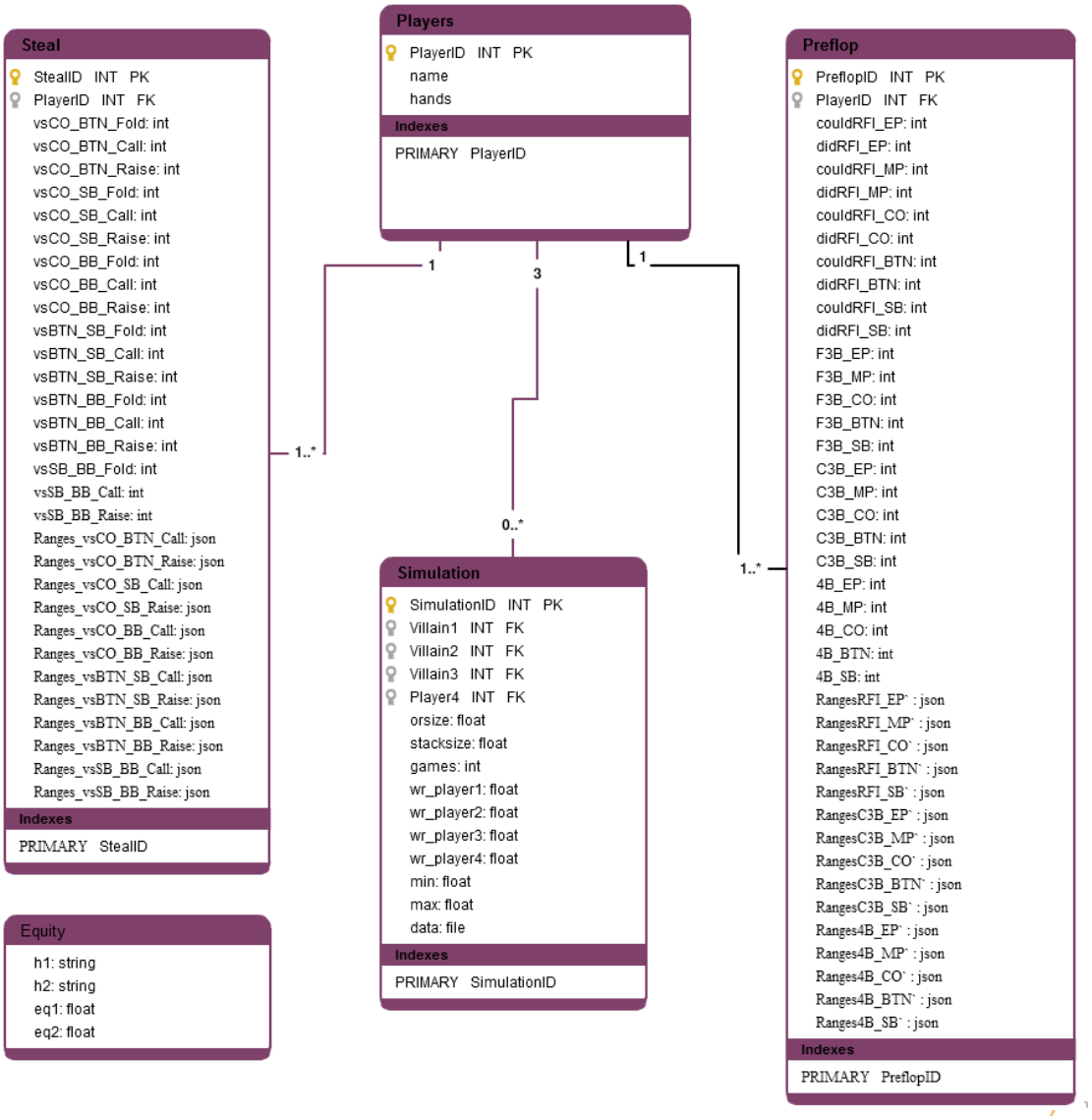
Igual que en los otros dos casos anteriores, duplicaremos estos campos para guardar información acerca de los rangos que representan cada acción.

RFI	Posición	Acción F	Acción C	Acción R
CO	BTN	*Ranges_vsCO_BTN_Fold*	*Ranges_vsCO_BTN_Call*	*Ranges_vsCO_BTN_Raise*
	SB	*Ranges_vsCO_SB_Fold*	*Ranges_vsCO_SB_Call*	*Ranges_vsCO_SB_Raise*
	BB	*Ranges_vsCO_BB_Fold*	*Ranges_vsCO_BB_Call*	*Ranges_vsCO_BB_Raise*
BTN	SB	*Ranges_vsBTN_SB_Fold*	*Ranges_vsBTN_SB_Call*	*Ranges_vsBTN_SB_Raise*
	BB	*Ranges_vsBTN_BB_Fold*	*Ranges_vsBTN_BB_Call*	*Ranges_vsBTN_BB_Raise*
SB	BB	*Ranges_vsSB_BB_Fold*	*Ranges_vsSB_BB_Call*	*Ranges_vsSB_BB_Raise*

Estructura de la base de datos del proyecto

Como hemos visto anteriormente, hemos recolectado la información que necesitamos para estudiar y poder alimentar de datos una estrategia explotadora para cada rival.

De las tres situaciones que analizamos, guardaremos las dos primeras en una tabla llamada *Preflop* que contendrá todos los datos de las dos primeras situaciones. En la tabla *Steal* guardaremos toda la información referente a la tercera parte.



Además, añadiremos dos tablas adicionales para la última parte del proyecto, donde ejecutaremos varias simulaciones.

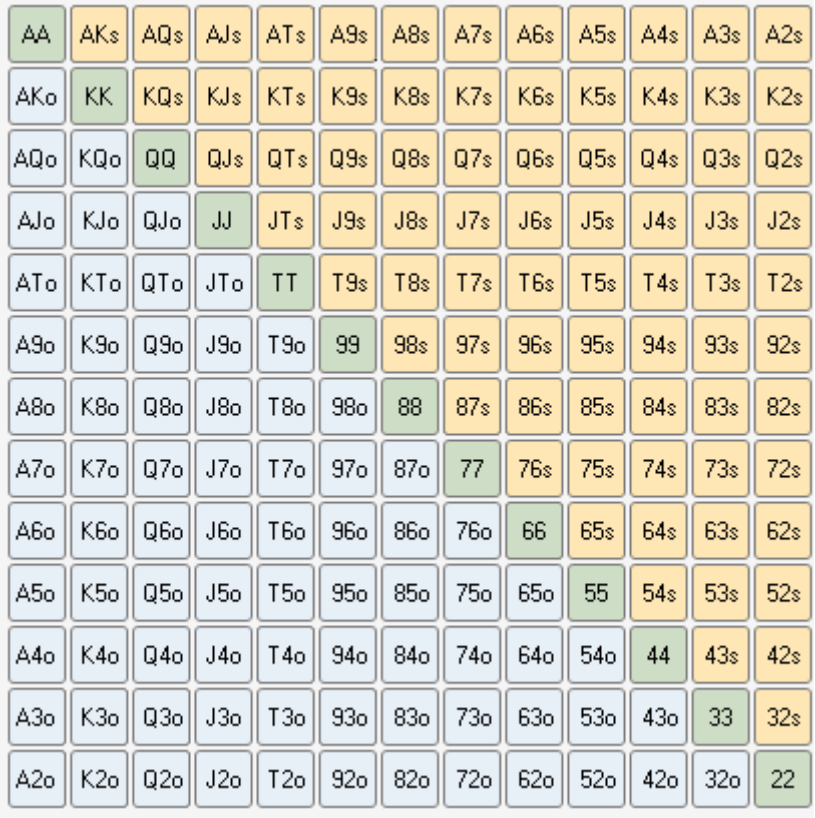
La primera tabla contendrá información sobre cada simulación que ejecutemos. Además de los identificadores de los jugadores, encontraremos valores constantes para cada ejecución, como el número de manos por simulación. Todos estos valores los explicaremos más adelante.

La segunda tabla que añadimos no guarda ningún tipo de relación con el resto de la base de datos, pero es crucial para reducir el tiempo de ejecución de las simulaciones. En ella guardaremos el equity (o porcentaje de ganancia del bote a largo plazo) de cada mano cuando se enfrenta contra cualquier otra mano. De esta forma, cuando estemos ejecutando la simulación, no tendremos que calcular cada vez el porcentaje de ganancia de cada mano, sino que simplemente consultaremos el valor en la base de datos.

No hay otra forma de realizar esto, ya que no existe ninguna fórmula para calcular el equity entre dos manos de una forma rápida y los programas que nos proporcionan esta información realizan simulaciones enumerando todas las posibles combinaciones de cartas. Como las situaciones que queremos analizar en términos de equity son de un par de cartas contra otro par de cartas, enumerar todo nos va a servir para este proyecto porque no entraremos en situaciones más complejas de n-pares contra n-pares donde requeriríamos métodos más complejos como Monte Carlo.

Esta tabla tendrá cuatro campos y carece de claves, tanto primarias como foráneas. Tendremos los campos h1 y h2, que representan una mano y los campos eq1 y eq2 que representan el valor de la equity de cada mano cuando están enfrentadas.

En la siguiente imagen podemos ver todas las posibles combinaciones de manos que hay:



AA	AKs	AQs	AJs	ATs	A9s	A8s	A7s	A6s	A5s	A4s	A3s	A2s
AKo	KK	KQs	KJs	KTs	K9s	K8s	K7s	K6s	K5s	K4s	K3s	K2s
AQo	KQo	QQ	QJs	QTs	Q9s	Q8s	Q7s	Q6s	Q5s	Q4s	Q3s	Q2s
AJo	KJo	QJo	JJ	JTs	J9s	J8s	J7s	J6s	J5s	J4s	J3s	J2s
ATo	KTo	QTo	JTo	TT	T9s	T8s	T7s	T6s	T5s	T4s	T3s	T2s
A9o	K9o	Q9o	J9o	T9o	99	98s	97s	96s	95s	94s	93s	92s
A8o	K8o	Q8o	J8o	T8o	98o	88	87s	86s	85s	84s	83s	82s
A7o	K7o	Q7o	J7o	T7o	97o	87o	77	76s	75s	74s	73s	72s
A6o	K6o	Q6o	J6o	T6o	96o	86o	76o	66	65s	64s	63s	62s
A5o	K5o	Q5o	J5o	T5o	95o	85o	75o	65o	55	54s	53s	52s
A4o	K4o	Q4o	J4o	T4o	94o	84o	74o	64o	54o	44	43s	42s
A3o	K3o	Q3o	J3o	T3o	93o	83o	73o	63o	53o	43o	33	32s
A2o	K2o	Q2o	J2o	T2o	92o	82o	72o	62o	52o	42o	32o	22

Estas combinaciones no tienen en cuenta los diferentes palos, simplemente tiene en cuenta si el par de cartas son del mismo palo o no.

Por lo tanto, tenemos 13x13=169 manos diferentes. El orden no importa, ya que enfrentar h1 con h2 es lo mismo que hacerlo con h2 contra h1, así que no tendremos repeticiones en la base de datos donde h1=h2 y h2=h1. Como queremos enfrentar todas contra todas, y teniendo en cuenta esto, el número de registros será de 169+168+…+1, ya que en cada iteración hemos de restar un registro ya calculado.

Por lo tanto, como queremos calcular el valor de 1+2+3+…+n usaremos la fórmula

$$\sum_{i=1}^{i=n} i = \frac{n(n+1)}{2} = 14365\ (n = 169)$$

Tenemos, por lo tanto, 14365 registros en la tabla *Equity* que nos garantiza que podremos saber la equity de cada mano en todo enfrentamiento de cara a la simulación.

En este proyecto no entraremos en cómo realizar los cálculos necesarios para calcular la equity mano por mano, sino que usaremos un programa llamado pokenum (http://download.gna.org/pokersource/archives/poker-eval/) para evaluar todas las manos.

Esto lo realizaremos en el archivo filling_equities.php del proyecto. En este archivo está el algoritmo para enfrentar todas las manos, descartando las repeticiones antes de ejecutar el cálculo y guardando el resultado en la base de datos.

Ejecutamos el programa para calcular la equity entre dos manos de la siguiente forma:

```
>pokenum -h Ad 3d Ac 4c
Holdem Hi: 1712304 enumerated boards
cards     win    %win      lose   %lose       tie    %tie        EV

Ad 3d  433520  25.32    528104  30.84    750680  43.84     0.472
Ac 4c  528104  30.84    433520  25.32    750680  43.84     0.528
```

Los argumentos del programa son las cartas a enfrentar y en el propio archivo ya realizamos la limpieza del output del programa y nos quedamos con el EV, que es el dato que finalmente necesitamos y guardamos.

Parte 2: La estrategia

La importancia del cálculo

Anteriormente hemos hablado de la importancia de tener un “edge” en cualquier tipo de juego para garantizar la ganancia en el largo plazo. En el juego del póquer, este “edge” viene dado por la calidad de las decisiones que tomamos. Estas decisiones deben ser de mayor calidad que la de nuestros rivales.

Una de los pilares básicos del póquer es que es un juego matemático y completamente racional. Las decisiones que tomemos siempre tienen un cálculo asociado y la importancia de hacerlo correctamente es la base fundamental de nuestro beneficio a largo plazo.

Veamos un ejemplo que nos ayudará a ver la importancia de un cálculo correcto y nos introducirá algunos conceptos importantes del juego y de la necesidad del proyecto.

Imaginemos que estamos jugando con un amigo. Nuestras dos cartas privadas son 
. Las cartas comunitarias en el Turn (la segunda ronda de apuestas) son  .

Ahora imaginemos también que nuestro amigo comete un error con la mano y sin querer da la vuelta a sus cartas, las vuelve a colocar bien rápidamente, pero nos ha dado tiempo a ver que su mano es  . Es decir, tiene la misma mano que nosotros, con la gran diferencia de que a una carta por completar las cinco cartas comunitarias, su mano tiene un proyecto de color.

Es decir, si la carta que queda por salir es un diamante, su mano será mejor que la nuestra y nos ganará. En caso contrario, si no sale ningún diamante, será un empate: Tenemos la misma mano y no podría haber desempate alguno. Dicho de otra forma, nuestra posibilidad de ganar la mano es 0%. Solo podemos perder o empatar.

¿Cuál es la posibilidad de perder? Como solo podemos perder si sale un diamante, podemos realizar el cálculo con un simple conteo. Hay 52 cartas, repartidas en 4 palos, 13 cartas por cada palo. Entre las cartas comunitarias y las cartas privadas de ambos suman 2+2+4=8, por lo que quedan 52-8=44 cartas en la baraja. Las cartas comunitarias contienen 2 diamantes y nuestro rival tiene 2 más. Por lo tanto, en la baraja, de las 44 cartas quedan 9 que le benefician a él. El resto, 44-9=35, nos benefician a nosotros, porque empatamos, que es a lo máximo que aspiramos. Es decir, que la probabilidad de que la última carta sea un diamante, es de 9/44=0,2045.

Nuestro amigo, nervioso por haber cometido semejante error, decide que juega el resto de sus fichas. En este punto vamos a separar el ejemplo en dos partes.

En el primer caso, supondremos que el bote es de 100 Euros y que tanto a nosotros como a nuestro rival, nos quedan 20 Euros por apostar.

Como nuestro rival ha apostado sus 20 Euros restantes, nosotros tenemos dos opciones: o abandonamos la mano o igualamos su apuesta. Estudiemos el valor esperado de cada acción.

- **Abandonamos**: El valor esperado es 0. Al abandonar no podemos recuperar nada del bote.
- **Igualamos**: Al igualar pueden suceder dos cosas:
  1. **Igualamos y perdemos**: Perdemos los 20 Euros de la apuesta que hemos pagado.
  2. **Igualamos y empatamos**: Se reparte el bote final.

Como hemos calculado anteriormente, la probabilidad de perder que tenemos es la probabilidad de que en la siguiente carta aparezca un diamante. Esta probabilidad es del 20.45%. Con este dato, podemos ponderar el valor esperado de igualar la apuesta.

$$Valor\ esperado\ igualar = Beneficio - Coste$$

$$VE = \left(Prob_{perder} * Beneficio_{perder} + Prob_{empatar} * Beneficio_{empatar}\right) - Coste$$

$$Valor\ esperado\ igualar = (0.2045 * -20 + (1 - 0.2045) * 70) - 20 = 51.595 - 20 = 31.595$$

El valor esperado de igualar (31.595) es superior al de abandonar (0). Por lo tanto, igualar la apuesta es la opción correcta.

En el segundo caso, supondremos que el bote es de 20 Euros y que tanto a nosotros como a nuestro rival, nos quedan 60 Euros por apostar. Aunque las cantidades del bote como las de las apuestas son diferentes, la suma total sigue siendo la misma.

Como nuestro rival ha apostado sus 60 Euros restantes, nosotros tenemos dos opciones: o abandonamos la mano o igualamos su apuesta. Estudiemos el valor esperado de cada acción.

- **Abandonamos**: El valor esperado es 0. Al abandonar no podemos recuperar nada del bote.
- **Igualamos**: Al igualar pueden suceder dos cosas:
  3. **Igualamos y perdemos**: Perdemos los 60 Euros de la apuesta que hemos pagado.

4. **Igualamos y empatamos**: Se reparte el bote final.

Al igual que en el ejemplo anterior, la probabilidad de que salga un diamante y perder es del 20.45%. Con este dato, podemos ponderar el valor esperado de igualar la apuesta.

$$Valor\ esperado\ igualar = Beneficio - Coste$$

$$VE = \left(Prob_{perder} * Beneficio_{perder} + Prob_{empatar} * Beneficio_{empatar}\right) - Coste$$

$$Valor\ esperado\ igualar = (0.2045 * -60 + (1 - 0.2045) * 70) - 60 = 43.415 - 60 = -16.585$$

El valor esperado de igualar (-16.585) es inferior al de abandonar (0). Por lo tanto, abandonar la apuesta es la opción correcta.

Como se puede apreciar, simplemente estamos realizando un cálculo de probabilidad para determinar la opción correcta de juego. Esto, en cierta forma, es una sensación que todos tenemos dentro sin hacer ningún cálculo. En el primer caso, muchas personas aceptarían pagar la apuesta, porque por 20 Euros aspiramos a repartir el bote final de 140, es decir, ganar 70. En la segunda opción, aceptar ya no es tan fácil, puesto que tenemos que arriesgar 60 para ganar la mitad de los 140.

Si lo modelamos como una función y escogemos X como el coste de pagar la apuesta y asumimos que ambos jugadores hemos empezado con 70 Eur y el bote final es de 140, tenemos la siguiente función lineal.

$$f(x) = (-0.2045x + 55.685) - x$$

Si igualamos la función a 0

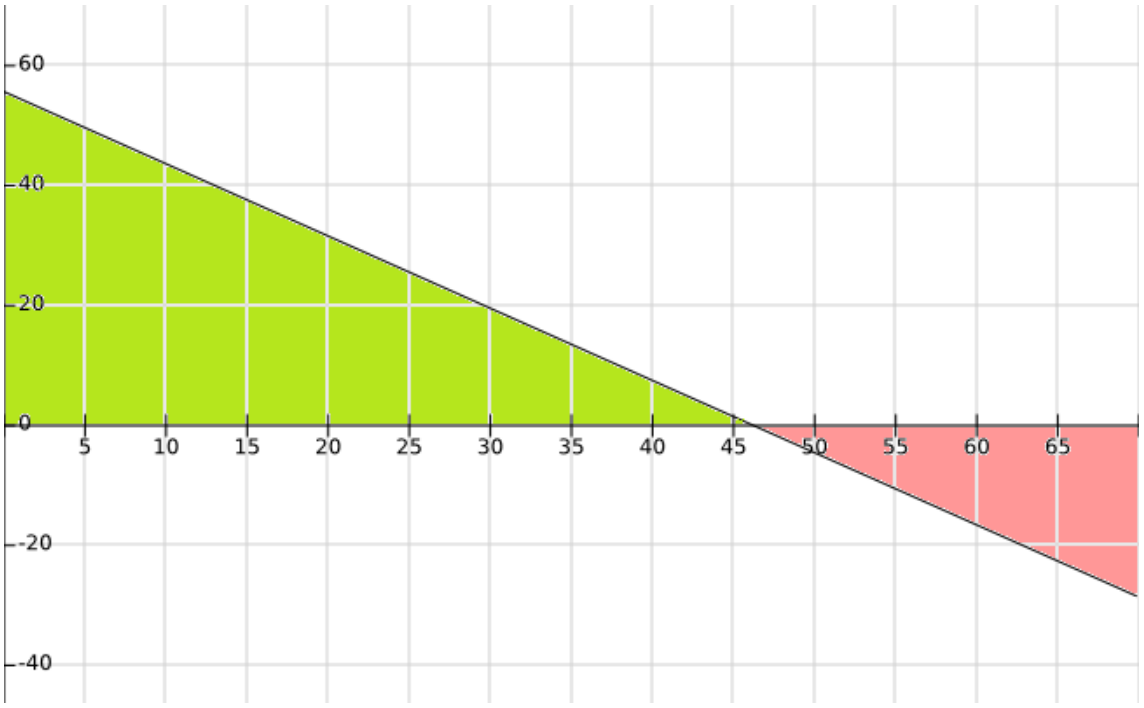
$$f(x) = (-0.2045x + 55.685) - x = 0$$

$$\frac{-55.685}{-1.2045} = x$$

$$x = 46.23$$

Si la apuesta del rival cuando decide jugarse todas sus fichas fuese de 46.23 Euros (y, por lo tanto, el bote fuese de (70-46.23)*2), tendríamos la situación donde la opción de igualar tiene el mismo valor esperado que la opción de abandonar.

Esta es la función de la situación analizada:



Situaciones del juego para estudiar

En este proyecto estudiaremos las tres situaciones más comunes en la primera ronda de apuestas en el Texas Hold'em, tal y como hemos comentado en el alcance del proyecto.

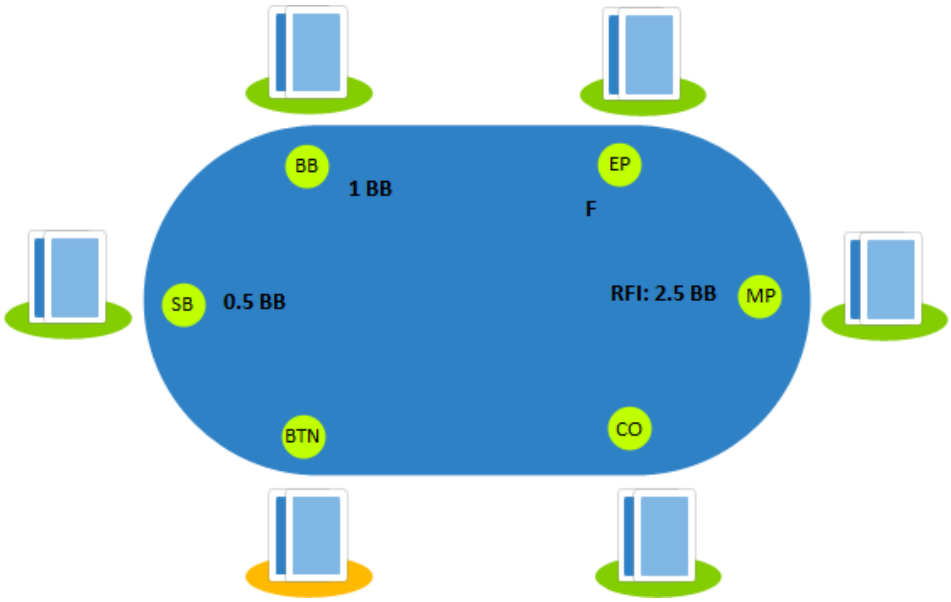
Estas tres situaciones no están elegidas al azar: Son las tres situaciones más típicas en una partida de póquer. De hecho, la mayoría de manos empiezan con alguna de las situaciones que estudiaremos, por lo que maximizamos el alcance de situaciones que podremos analizar con este proyecto.

RFI (Raise First In)

La primera y probablemente la más importante, es el "Raise First In" (RFI, en adelante) y como actúa el jugador frente a un movimiento de un rival. El RFI es la acción de subir la apuesta en la fase Pre-flop del juego siendo el primer jugador en hacerlo. Aproximadamente, entre el 93 y el 97% de las manos que se juegan a póquer empiezan con un RFI por parte de algún jugador.

No es una casualidad que el RFI sea la acción más común en la primera ronda de apuestas. De hecho, en una partida de jugadores profesionales, el 100% de las manos empezaran de esta forma. Esto se debe primordialmente a dos factores. El primero de todos es el hecho de que en el juego existen las ciegas ("blinds", en inglés). Esta apuesta obligatoria se realiza totalmente a ciegas, antes de recibir las cartas. Por lo tanto hay un dinero en juego que no obedece a ninguna acción de los jugadores. Ese dinero está allí porque así son las reglas del juego. Esto nos lleva irremediablemente al segundo factor. En algún momento, un jugador realizará una apuesta por el simple hecho de conseguir ese dinero, bien porque trata de rentabilizar su movimiento o porque considera que su mano es mejor a la mano que tienen sus rivales.

Para ilustrar mejor que es un RFI, vamos a usar la siguiente imagen:



La secuencia de juego que se representa es la siguiente:

1. SB (Small Blind) está obligado a apostar 0.5 BB (Big Blind, Ciegas grandes, Unidades de apuesta)
2. BB (Big blind) está obligado a apostar 1 BB (Big Blind, Ciegas grandes, Unidades de apuesta)
3. EP es el primero en realizar un movimiento. Decide retirarse (Fold)
4. MP es el segundo en realizar un movimiento. Decide apostar y subir la apuesta (RFI) a 2.5 BB

El hecho de realizar un RFI obliga al resto de rivales que quedan por hablar a meter más dinero si desean seguir jugando. Como es lógico, el RFI de un jugador depende directamente de que posición ocupa en la mesa. Esto es así, porque no es lo mismo realizar un RFI en la primera posición, donde tenemos 5 jugadores por hablar, que en la posición del botón, donde solo quedan 2 jugadores, los que han puesto las ciegas.

Dicho de otra forma, si nuestro objetivo al realizar un RFI es conseguir ganar las apuestas obligatorias de 0.5 BB y 1 BB, esta rentabilidad es la misma para todas las posiciones, lo que es diferente es la posibilidad de que otro jugador pague o resuba la apuesta. Cuanto más tardía sea la posición, mayor es la posibilidad de materializar la rentabilidad con un RFI.

Para demostrar esto vamos a ver cuál es el porcentaje de RFI de los jugadores según la posición.

Primero vamos a ver cuál es el valor medio de cada posición mediante una consulta SQL

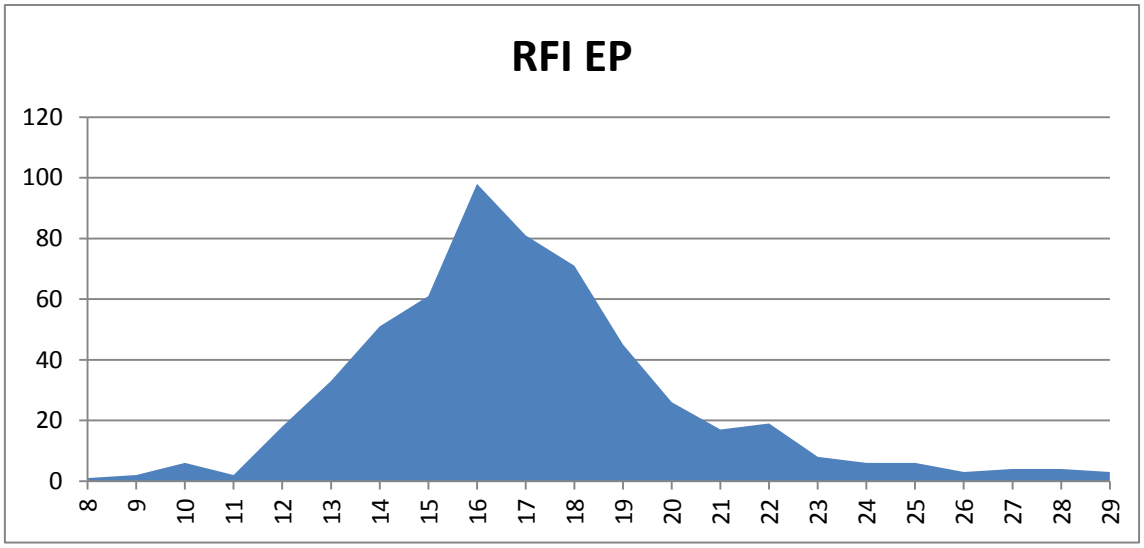
```
SELECT avg(didRFI_EP/couldRFI_EP*100) as avgEP,

avg(didRFI_MP/couldRFI_MP*100) as avgMP,

avg(didRFI_CO/couldRFI_CO*100) as avgCO,

avg(didRFI_BTN/couldRFI_BTN*100) as avgBTN,

avg(didRFI_SB/couldRFI_SB*100) as avgSB

FROM preflop
```

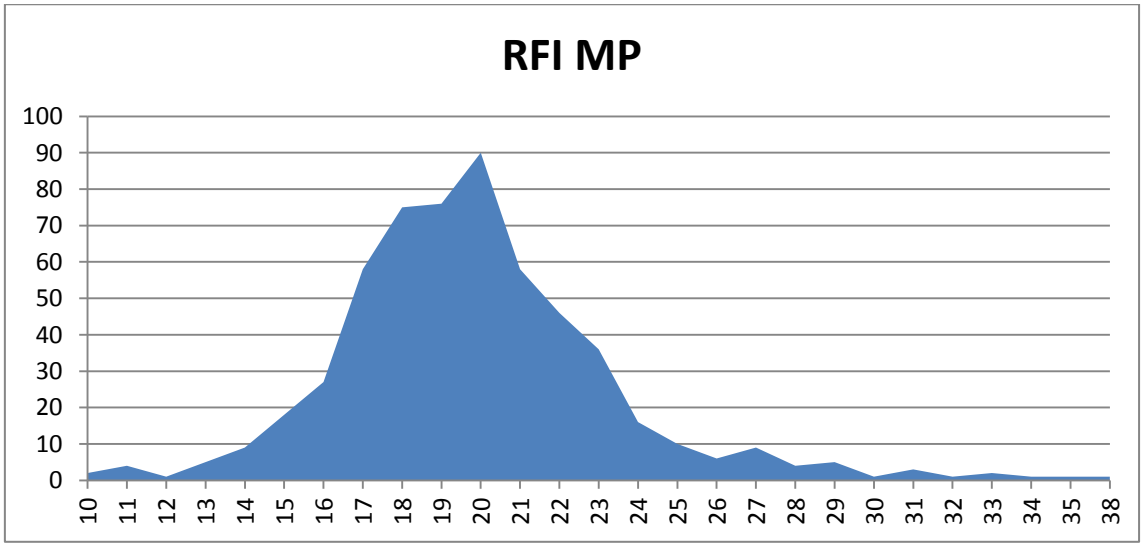
El resultado es el siguiente:

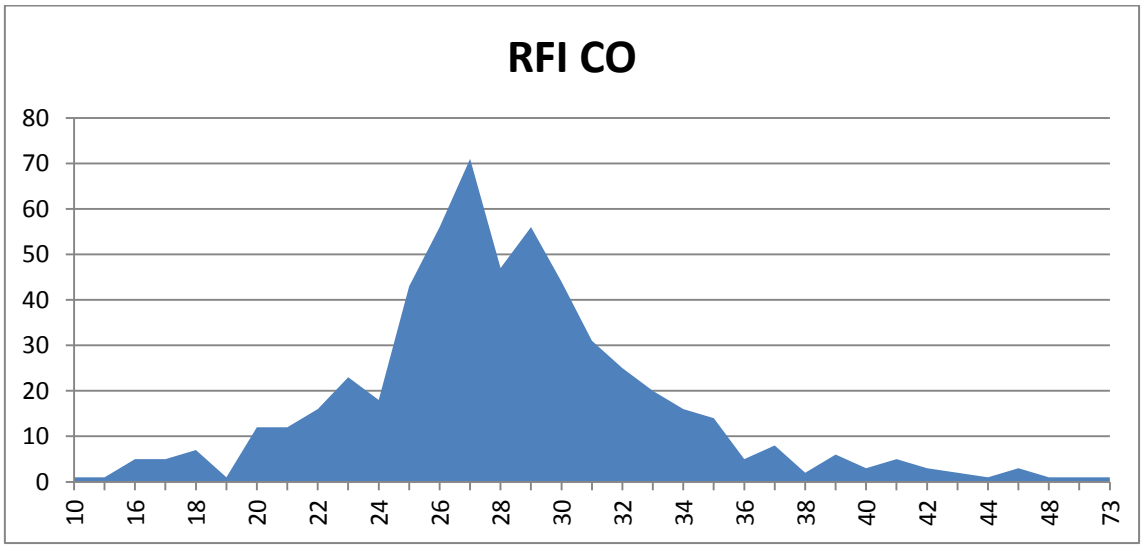
avgEP	avgMP	avgCO	avgBTN	avgSB
17.01585368	19.85284957	28.25499092	52.05347538	44.27820833

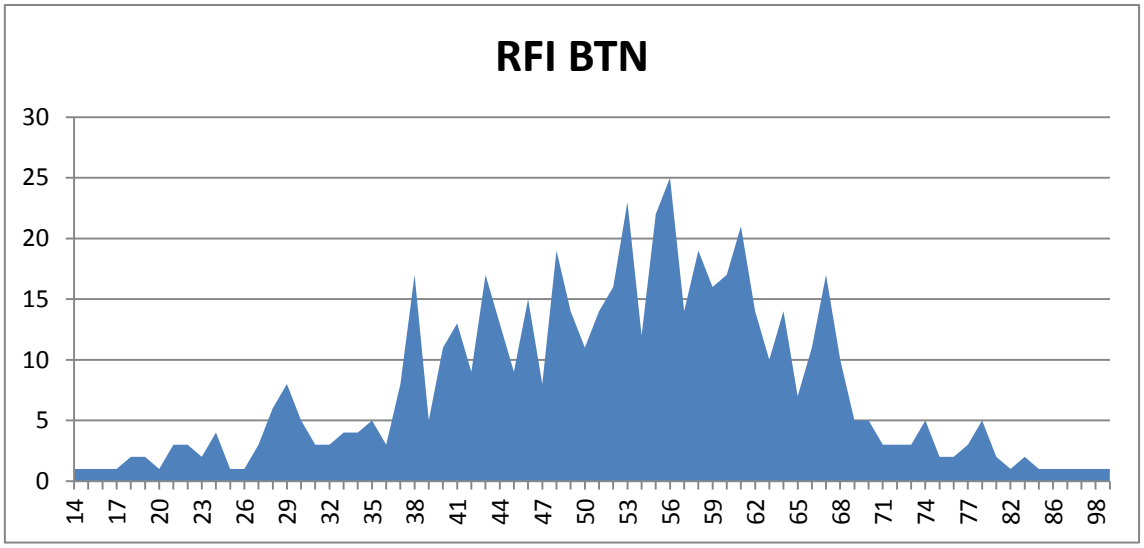
Como podemos observar el valor de avgBTN es mayor que avgSB, aun siendo SB una posición más tardía en el juego. Esto suele ser un error generalizado en la mayoría de jugadores y la razón es porque SB es la única posición donde el único rival que queda por hablar tiene una posición mejor en la mesa en rondas posteriores. Esto es visto como una desventaja y, por lo tanto, los jugadores suelen realizar menos subidas. De la misma forma, los jugadores en BB tienden a defender más manos por encontrarse en una mejor posición cuando reciben una subida de SB. En ese caso, si tiene sentido que SB realice menos subidas, pero siempre respetando el ratio de riesgo/recompensa que la situación conlleva. Por lo general, la mayoría de jugadores tiende a sobrevalorar la posición en el juego.

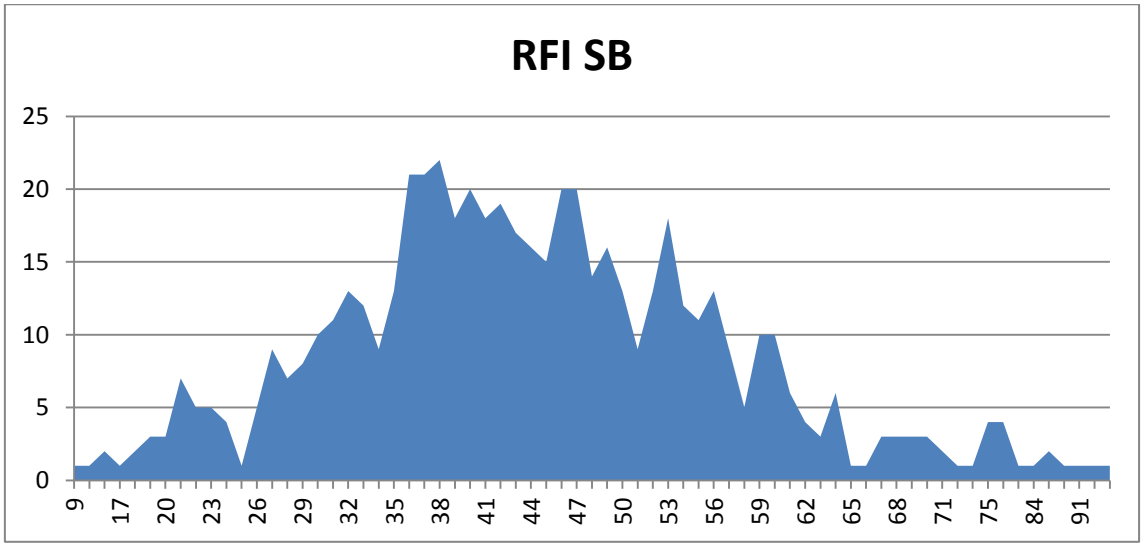
A continuación podemos ver como se distribuyen los diferentes valores de RFI por posiciones de los jugadores de la base de datos.


RFI EP
120
100
80
60
40
20
0
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29


RFI MP
100
90
80
70
60
50
40
30
20
10
0
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 38


RFI CO
80
70
60
50
40
30
20
10
0
10 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 48 73





El valor del RFI nos indica el porcentaje de manos que el jugador está dispuesto a jugar desde cada posición. Por ejemplo, para el valor medio de RFI en la primera posición EP, que es de un 17% aproximadamente, podemos representar un rango de manos.

Este rango de manos supondremos de ahora en adelante que nunca está polarizado. Es decir, cuando tenemos un porcentaje de manos, siempre lo tomaremos como una representación del rango más alto.

Esta suposición, en la práctica, es cierta para todos los casos de RFI, porque de no serlo, no tendría sentido alguno. Esto es así porque el movimiento RFI sucede cuando ningún jugador ha realizado alguna acción anteriormente, excepto la opción de retirarse, lo cual lo deja fuera del juego y lo convierte en irrelevante para la toma de decisiones futuras. Como el jugador realiza un RFI sin ninguna información de los movimientos del resto de jugadores, su rango esta predeterminado por su posición y como no tiene sentido que decida jugar una mano cuya fuerza sea menor a una que decide no jugar, podemos suponer con gran acierto que el rango no está polarizado.

Dicho más formalmente, si suponemos que R es un rango y H una mano dentro del rango de R, no existe ninguna mano que no pertenezca a R y cuya fuerza sea mayor la fuerza de H.

Por lo tanto, el rango del 17% no polarizado para la media de RFI en la posición EP sería este

En esta imagen, están seleccionadas las manos que representan ese rango. En la diagonal encontramos todas las parejas, en la parte superior encontramos las manos del mismo palo (“suited”, en inglés), caracterizadas por una “s” después de la combinación de dos cartas y finalmente, en la parte inferior, tenemos las combinaciones de diferente palo (“offsuited”, en inglés), caracterizadas por una “o” después de la combinación de dos cartas.

En la cuadricula salen representadas 1326 manos posibles. Cada pareja tiene 6 posibles combinaciones, cada combinación del mismo palo tiene 4 y las combinaciones de diferentes palos 12.

La combinación seleccionada representa el 17% de las estadísticamente mejores manos. No todos los jugadores tienen porque querer jugar estas combinaciones precisamente. Por ejemplo, podríamos eliminar del rango manos como QTo y KTo e incluir el resto de parejas no seleccionadas (55, 44, 33 y 22) y el rango sería exactamente el mismo, puesto que hemos eliminado 24 combinaciones (2x12) y hemos añadido 24 más (6x4).

Conocer el rango que está dispuesto a jugar nuestro rival es fundamental para cualquier tipo de estrategia. En el ejemplo anterior, con un rango del 17%, hemos podido descartar el restante 83% de manos que no estarán en su rango. Esta información es muy relevante para elevar nuestra ventaja competitiva, tanto para este proyecto como para cualquier tipo de estrategia que planteemos en el juego.

Facing 3-bet (Recibiendo una subida)

La segunda situación que nos interesa estudiar es como nuestro rival actúa frente a una subida una vez ha realizado un RFI. Como es lógico, no podemos estudiar una situación en la que los rivales pagan la apuesta del jugador que hace el RFI, porque entonces pasamos a otra fase del juego (Flop) ni tampoco la situación en la que los demás jugadores se retiran, puesto que acaba la mano.

Cuando un jugador realiza un RFI, realmente está realizando un 2-bet. El 1-bet son las apuestas obligatorias de las ciegas. Si un rival decide subir de nuevo la apuesta después del RFI del jugador, decimos que el rival está realizando un 3-bet. Por eso, la situación que vamos a analizar se llama “Facing 3-bet”, porque estamos interesados en analizar el juego del jugador cuando se enfrenta a esta situación.

Como en todos los casos, al recibir un 3-bet el jugador puede retirarse, pagar la apuesta o volver a subir. En este caso estamos interesados en analizar todas sus decisiones. Cuando el jugador se retira, llamaremos a esta situación “Fold to 3-bet”, cuando paga, la llamamos “Call 3bet” y cuando vuelve a subir “4-bet”.

Empezamos por las situaciones en las que el jugador que hace un RFI se retira ante una subida. Primero miraremos cual es el valor medio de “Fold to 3-bet” desde las distintas posiciones con una consulta SQL:

```
SELECT avg(F3B_EP/(F3B_EP+C3B_EP+4B_EP))*100 as avgFold_EP,

avg(F3B_MP/(F3B_MP+C3B_MP+4B_MP))*100 as avgFold_MP,

avg(F3B_CO/(F3B_CO+C3B_CO+4B_CO))*100 as avgFold_CO,

avg(F3B_BTN/(F3B_BTN+C3B_BTN+4B_BTN))*100 as avgFold_BTN,

avg(F3B_SB/(F3B_SB+C3B_SB+4B_SB))*100 as avgFold_SB
```

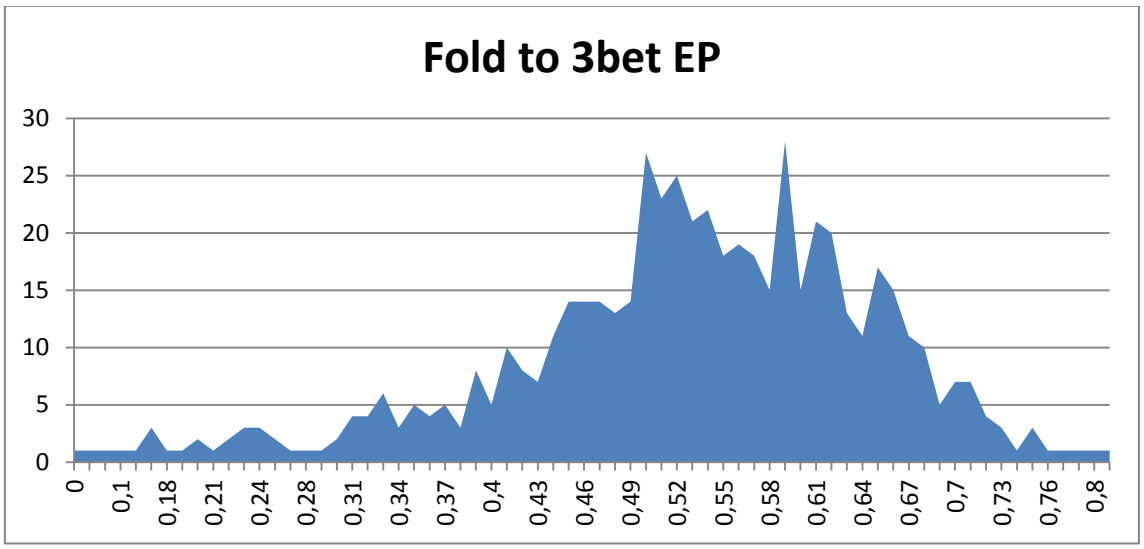
```
FROM `preflop`
```

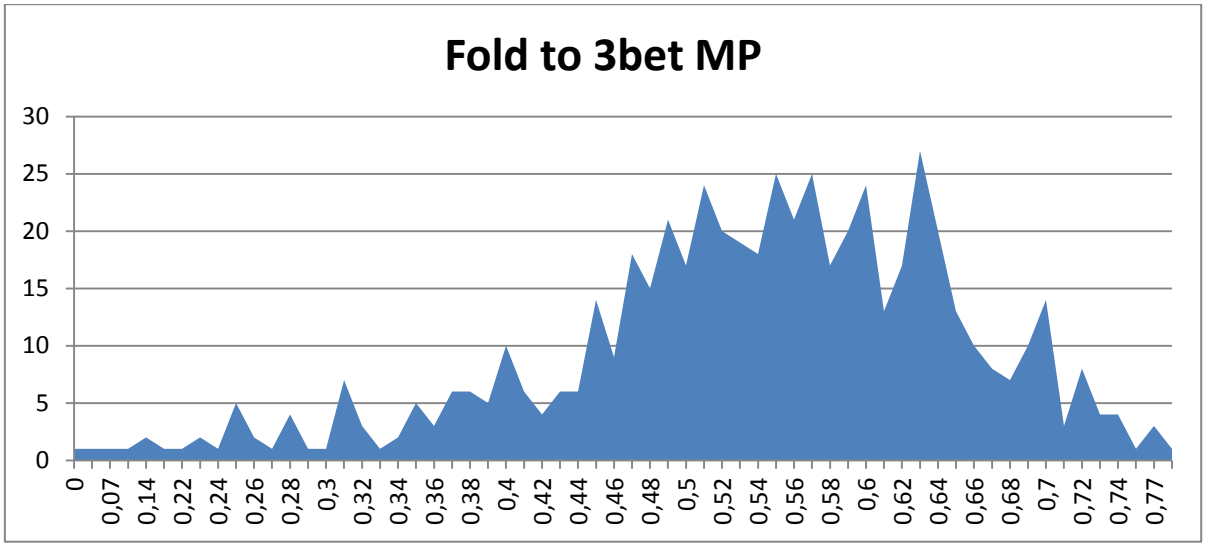
El resultado es el siguiente:

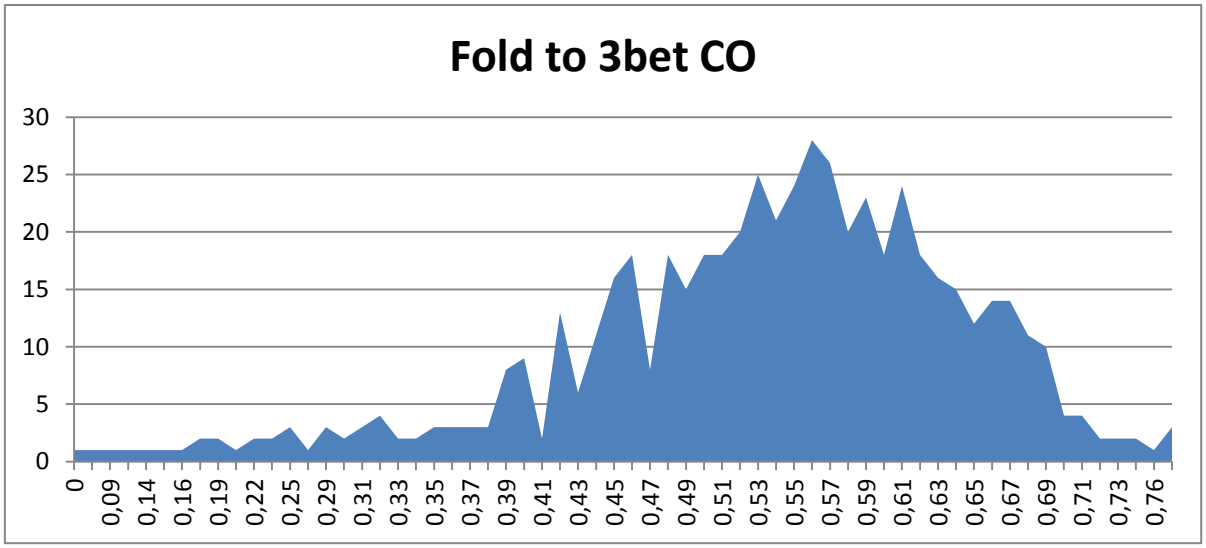
avgFold_EP	avgFold_MP	avgFold_CO	avgFold_BTN	avgFold_SB
52.72438330	53.63050261	53.25432088	58.05638355	56.56780560

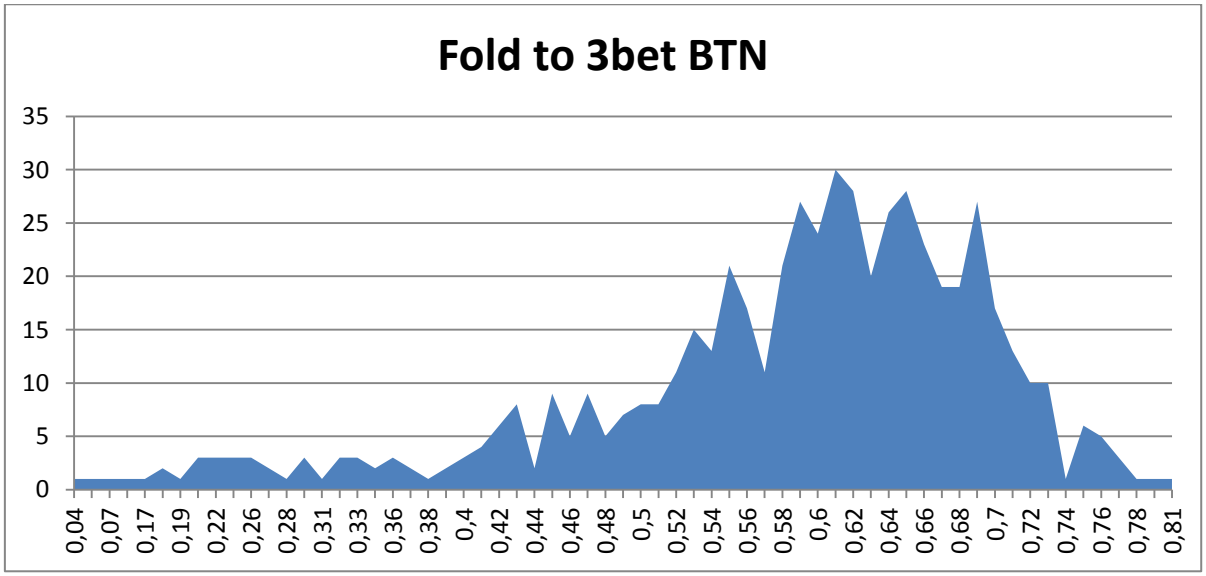
A diferencia del RFI, los resultados son muy parecidos entre sí. Hay que tener en cuenta que el valor de Fold to 3bet es relativo al RFI. Así, si el valor medio de RFI en la posición EP era del 17%, la media del valor para retirarse, es el 52% del 17%, es decir, se retiran del 9% de manos aproximadamente.

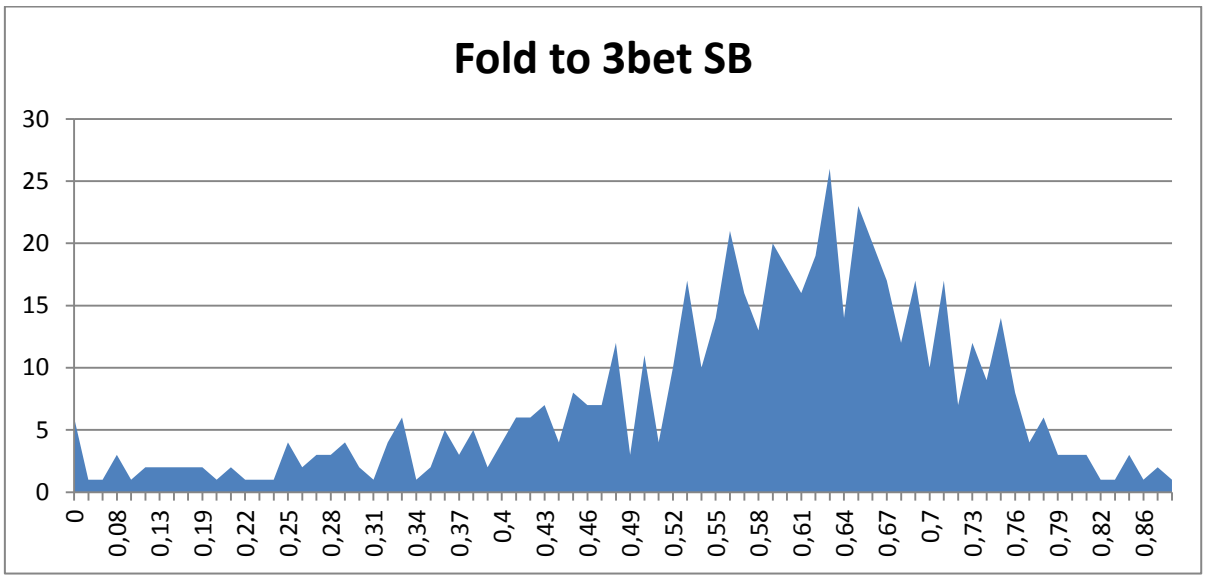
A continuación podemos ver los diferentes gráficos según la posición para los datos de "Fold to 3bet" de los jugadores de la base de datos.





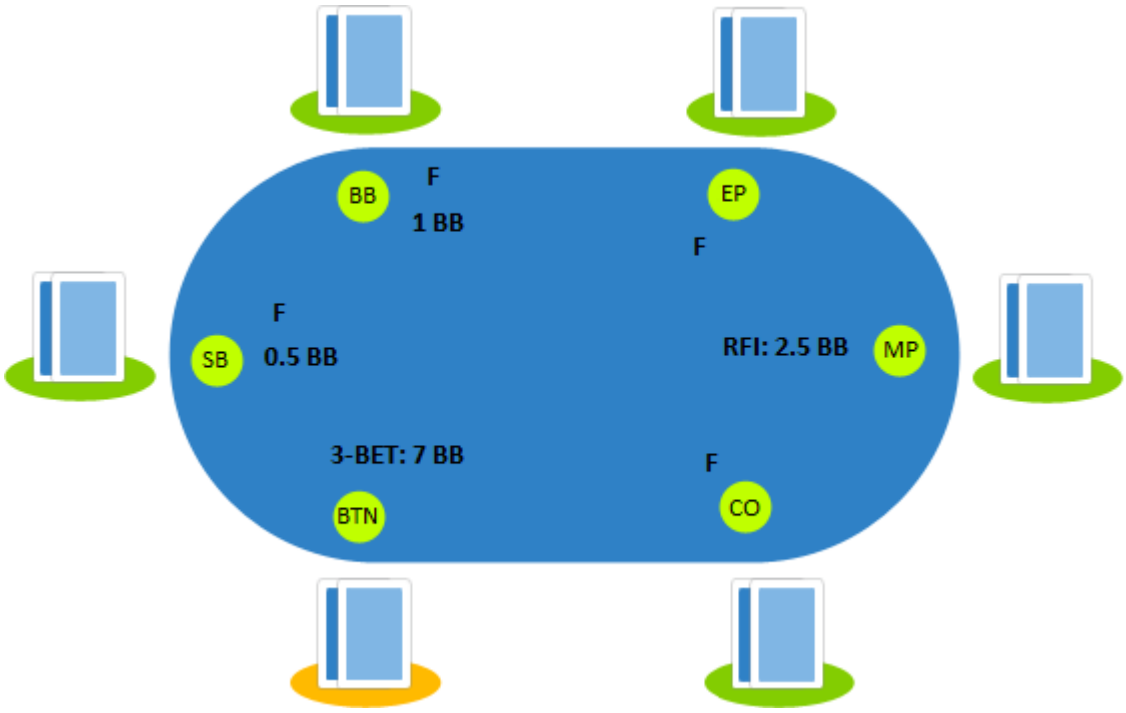

Fold to 3bet CO
30
25
20
15
10
5
0
0
0,09
0,14
0,16
0,19
0,22
0,25
0,29
0,31
0,33
0,35
0,37
0,39
0,41
0,43
0,45
0,47
0,49
0,51
0,53
0,55
0,57
0,59
0,61
0,63
0,65
0,67
0,69
0,71
0,73
0,76


Fold to 3bet BTN
35
30
25
20
15
10
5
0
0,04
0,07
0,17
0,19
0,22
0,26
0,28
0,31
0,33
0,36
0,38
0,4
0,42
0,44
0,46
0,48
0,5
0,52
0,54
0,56
0,58
0,6
0,62
0,64
0,66
0,68
0,7
0,72
0,74
0,76
0,78
0,81


Fold to 3bet SB
30
25
20
15
10
5
0
0
0,08
0,13
0,19
0,22
0,25
0,28
0,31
0,34
0,37
0,4
0,43
0,46
0,49
0,52
0,55
0,58
0,61
0,64
0,67
0,7
0,73
0,76
0,79
0,82
0,86

Es interesante, antes de continuar, realizar algunos cálculos sobre que significa en términos de rentabilidad hacer un 3-bet y las implicaciones que tiene para ambos jugadores.

Supongamos que Jugador1 en MP realiza un RFI a 2.5 BB. Jugador2 en BTN realiza un 3bet a 7 BB y el resto de jugadores se retira. Estos tamaños de apuesta son los más habituales en la mesas de juego online.



Para Jugador2, el coste de realizar el 3-bet son 7BB. Su beneficio, en caso de que Jugador1 se retire, será de 0.5BB (ciega pequeña) + 1BB (ciega grande) + 2.5BB (Jugador1). En total suman 4BB.

Para simplificar los cálculos supongamos que, si Jugador1 no se retira, nunca paga la apuesta, simplemente vuelve a subir (realiza un 4-bet) a 21BB y Jugador2 se retira, perdiendo las 7BB invertidas en el 3-bet.

El valor esperado de esta situación es el siguiente:

$$VE = Probabilidad_{retirada} * 4\,BB + Probabilidad_{resubida} * (-7BB)$$

Si ponemos la probabilidad de retirarse de Jugador1 en función de X

$$VE = x * 4\,BB + (1 - x) * (-7BB)$$

Como hemos simplificado la situación y hemos dicho que Jugador1 solo puede retirarse o volver a subir, estamos ignorando el caso en que Jugador1 pague simplemente la apuesta.

Si igualamos la función a 0 para saber dónde se encuentra el valor esperado neutro, obtenemos lo siguiente:

$$VE = x * 4\,BB + (1 - x) * (-7BB) = 0$$

$$4x - 7 + 7x = 0$$

$$x = \frac{7}{11} = 0.6363$$

Otra forma de realizar el mismo cálculo es:

$$VE = \frac{Coste}{Coste + Beneficio} = \frac{7}{7 + 4} = \frac{7}{11} = 0.6363$$

Es decir, si la probabilidad de retirada de Jugador1 cuando realizamos el 3-bet es superior al 63%, el **movimiento de Jugador2 resulta directamente rentable**, aunque no pueda ganar nunca en caso de igualar o volver a subir, lo cual es un escenario simple pero muy pesimista para Jugador2.

Si realizamos una rápida consulta a la base de datos para la posición EP, por ejemplo:

```
SELECT COUNT(*)

FROM `preflop`

WHERE `F3B_EP`/(`F3B_EP`+`C3B_EP`+`4B_EP`) > 0.63
```

Vemos como el resultado de la consulta devuelve 104 del total de 565 jugadores que hay en la base de datos. Estos 104 jugadores están cometiendo un gravísimo error, puesto que en el escenario planteado en el ejemplo, están perdiendo dinero a largo plazo. **A este perfil de jugadores, realizarles un 3-bet con el 100% de las manos es correcto**, porque aun suponiendo que **cuando no se retiran siempre perdemos e independientemente de la fuerza de nuestra mano, el valor esperado es positivo.**

Facing RFI

La tercera y última situación que estudiaremos en este proyecto es el comportamiento del rival cuando se enfrenta a un RFI por parte de un rival.

Como hemos visto anteriormente, el RFI va aumentando a medida que el jugador está en posiciones más tardías. Esto se debe a que la rentabilidad de realizar un RFI aumenta en la medida que quedan menos jugadores por hablar. Al aumentar los rangos en las últimas posiciones, los jugadores que quedan por tomar una decisión también aumentan sus

rangos de resubida y, por lo tanto, también lo hacen los rangos del jugador que realiza un RFI.

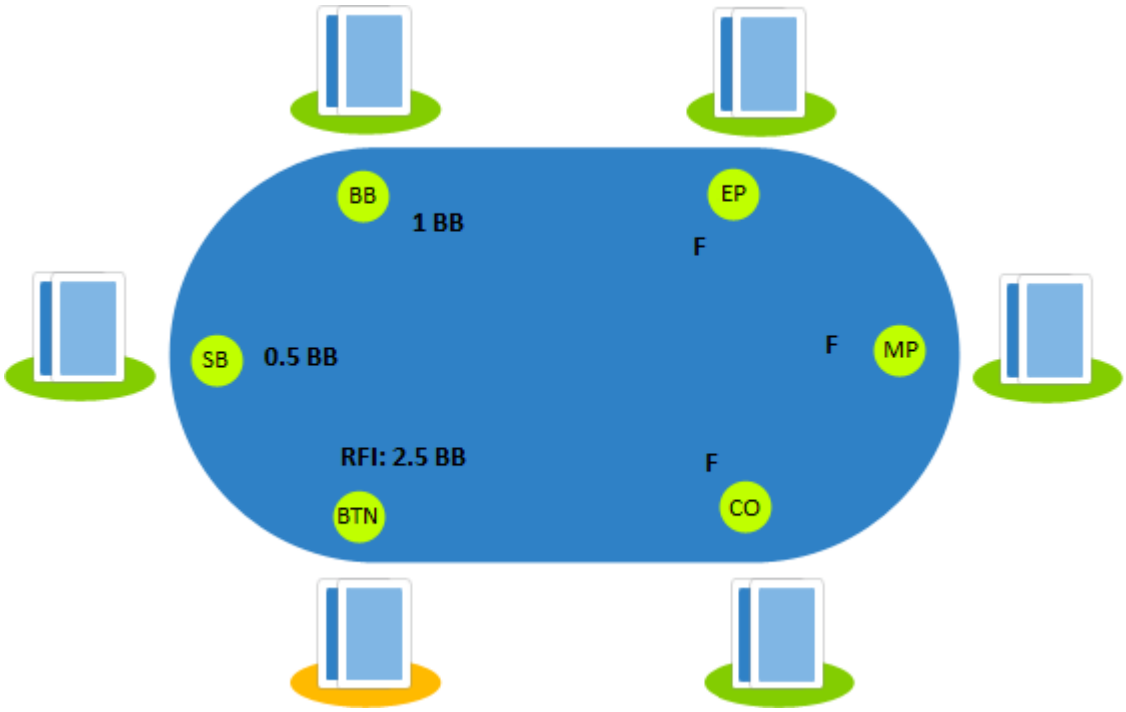
Esto tiene una consecuencia bastante grande en el juego y es que la mayoría de botes se generan en las últimas posiciones, donde están las ciegas y por eso, este tipo de situaciones toman el nombre de “Blind War” o guerra de ciegas.

Estudiar estas situaciones es básico para tener una ventaja respecto al resto de rivales. Simplemente aquí es donde radica la esencia del juego, ya que al ser los rangos de juego mucho más amplios, también aumentan la cantidad de errores que los jugadores cometen y de los cuales hemos de tratar de obtener beneficio.

Es por esto que en este proyecto vamos a tener en cuenta todas las posibles combinaciones de posiciones tanto para el agresor (el que realiza el RFI) como para el defensor (el que se enfrenta al 3-bet).

Empezaremos analizando las situaciones de guerra de ciegas separando los escenarios por la posición. Así, el agresor puede ocupar tres posiciones: estando en CO y atacando a BTN, SB y BB, estando en BTN y atacando a SB y BB o estando en SB y atacando a BB. Es por eso que separamos estos seis posibles escenarios y los analizaremos independientemente, porque es importante ver la tendencia de comportamiento entre unas situaciones y otras.

Aquí sucede exactamente lo mismo que en la situación anterior. Si observamos la situación desde el atacante, cuando realiza un RFI, desde por ejemplo la posición de BTN y, pongamos para este caso, que realiza una subida a 2.5 BB, tendremos la siguiente situación



Haremos los cálculos suponiendo el peor escenario para el atacante: Si alguna de las ciegas iguala o resube la apuesta, el atacante no puede ganar.

El valor esperado de esta situación es el beneficio que obtenemos menos el coste. El beneficio que obtiene el atacante solo se produce cuando los jugadores en SB y BB deciden retirarse ante la apuesta. En ese caso, el beneficio es la suma de la apuesta realizada (2.5 BB) más las ciegas (1.5BB), es decir, 4BB. Por lo tanto, para calcular el valor esperado, la fórmula queda así:

$$Valor\ esperado = Beneficio - Coste$$

$$Valor\ esperado = Prob_{abandonoSB} * Prob_{abandonoBB} * 4BB - 2.5BB$$

Si suponemos la probabilidad de abandonar de ambas ciegas como una única variable X e igualamos la función para encontrar el valor esperado neutro:

$$Valor\ esperado = X * 4BB - 2.5BB = 0$$

$$X = \frac{2.5BB}{4BB} = 0.625$$

Si la probabilidad de abandono combinada de SB y BB es del 62.5% o superior, **para el atacante, realizar un RFI con el 100% de las manos es rentable**.

Obviamente, si el tamaño del RFI varia, también lo hace la rentabilidad del movimiento.

$$(RFI = 2BB) => X = \frac{2BB}{3{,}5BB} = 0.57$$

$$(RFI = 3BB) => X = \frac{3BB}{4{,}5BB} = 0.66$$

$$(RFI = 4BB) => X = \frac{4BB}{5{,}5BB} = 0.73$$

Cuando aumentamos el tamaño de la apuesta, la probabilidad de abandono de SB y BB mínima para el valor esperado neutro incrementa, como es lógico, ya que aumentamos el coste del movimiento para obtener el mismo beneficio.

Volviendo al escenario que nos interesa, estudiaremos el comportamiento de los rivales cuando se enfrentan a un RFI.

Las acciones que se pueden tomar en esta situación son las mismas que siempre: retirarse, igualar o resubir la apuesta.

A diferencia del primer caso, donde estudiamos los rangos de RFI de un jugador y supusimos que los rangos no estaban polarizados, aquí no podremos hacer lo mismo. En el siguiente apartado profundizaremos en el tema de la polarización.

Por esta razón, hemos de tener en cuenta con que tipos de manos el jugador realizar según qué acción. Un método sencillo para poder distinguir el tipo de manos es el siguiente:

1) Crearemos el rango no polarizado según los datos del jugador en cada acción (pagar y resubir la apuesta) y posición.
2) Listaremos las manos guardadas en la base de datos correspondientes a cada rango.
3) Calcularemos que porcentaje de las manos están incluidas en su rango teórico y cuáles no.

Esta sencilla clasificación nos dará el porcentaje de manos con las que el rival realiza una subida y realmente cada mano está incluida dentro del rango que representa el jugador.

Veamos dos ejemplos:

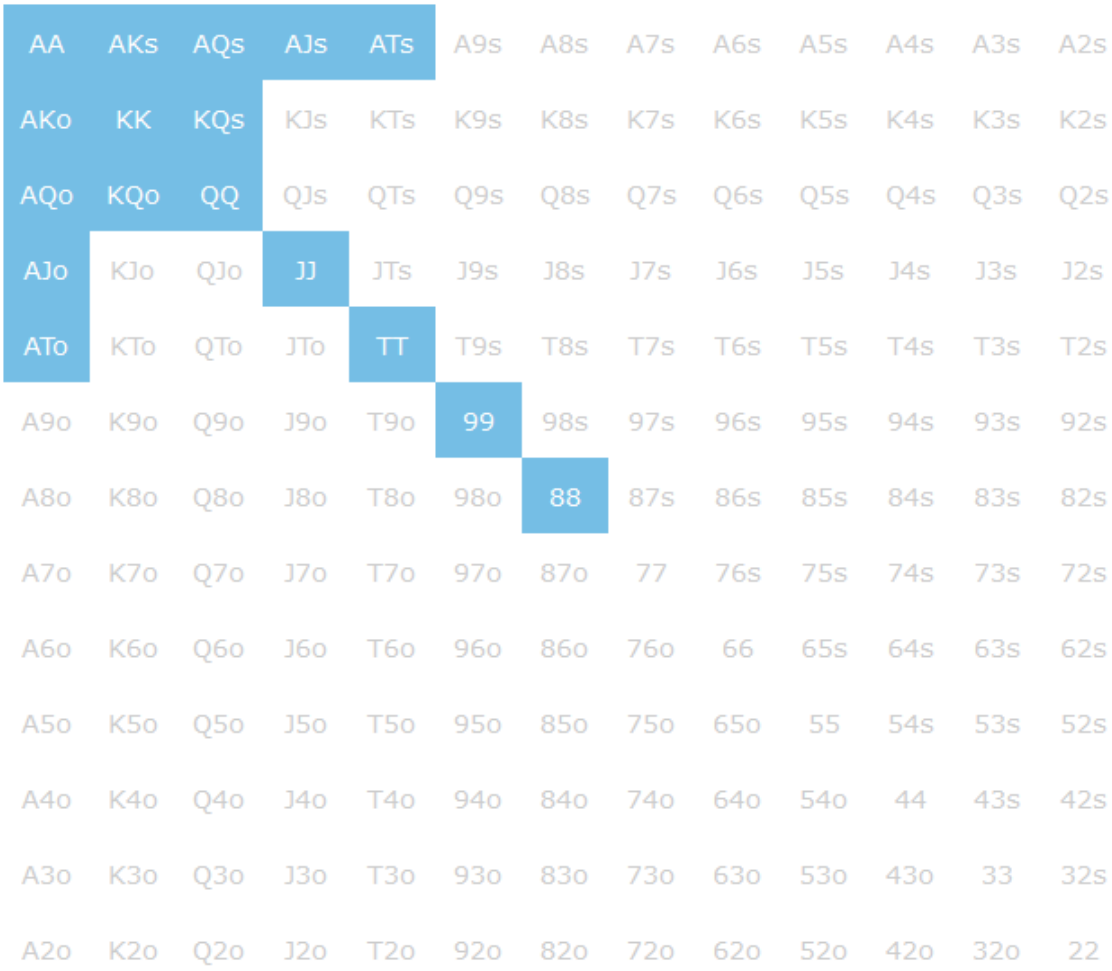
En el programa buscamos el jugador con el nick “TopLad14” y nos vamos a la sección “Blind war” del software. Este jugador no tiene por costumbre polarizar los rangos. Es decir, su rango de manos de resubida suele representar la parte más alta del rango.

Para verlo, podemos desplazarnos a la situación “vs BTN” en la posición “BB”.

vs BTN	in SB	80	6	14
	in BB	63	28	9

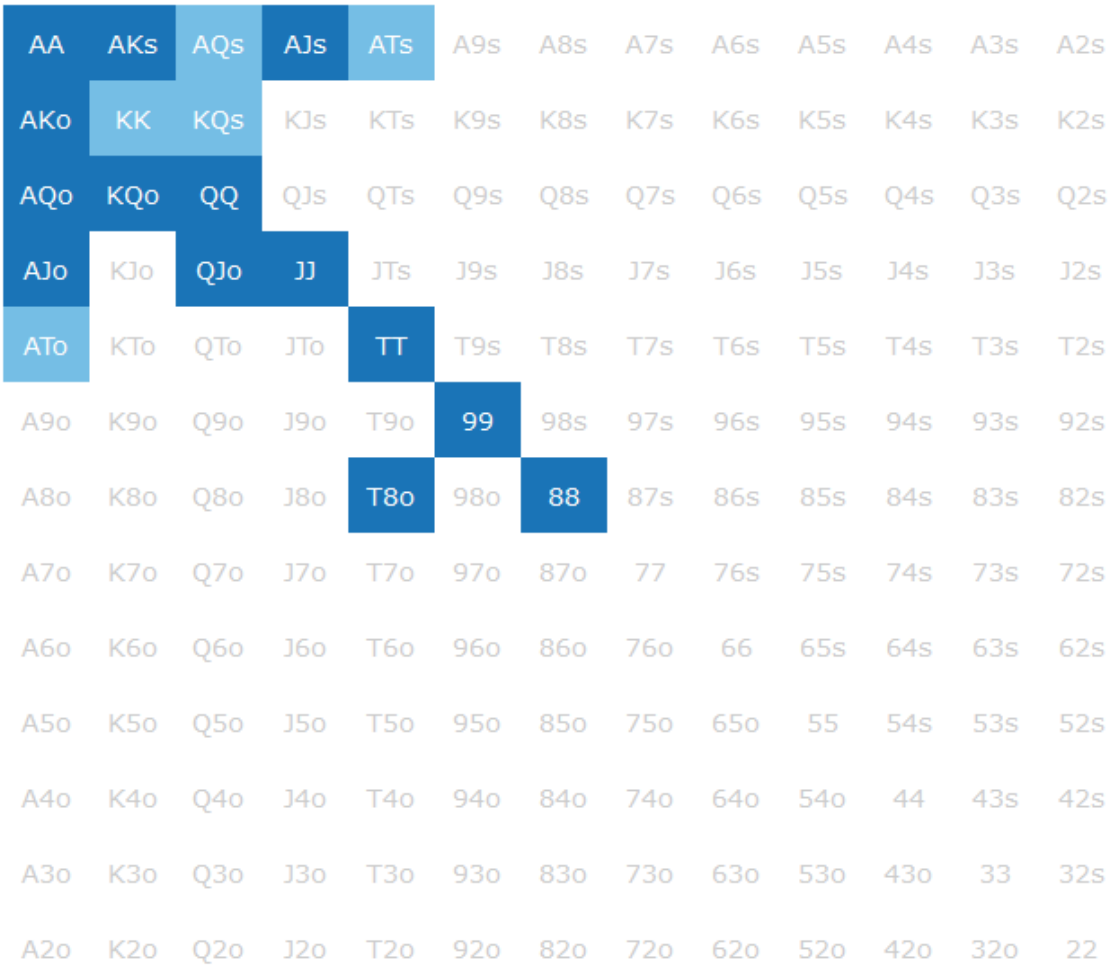
Como podemos observar, la probabilidad de retirada/pagar la apuesta/resubir en la posición BB para un RFI que proviene de BTN es 63/28/9, respectivamente.

Si pasamos el ratón por encima del valor de resubida o 3bet, podemos ver representado a la izquierda de la pantalla el rango teórico que tiene el valor 9%.

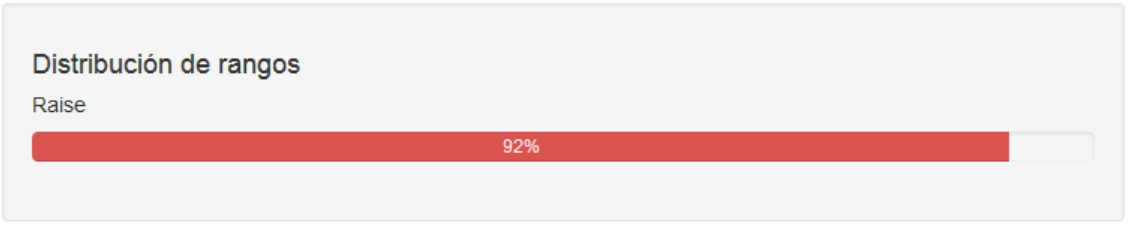


Las manos sombreadas de color azul pertenecen a la parte más alta del rango que representa un 9% de las posibles combinaciones.

Si hacemos click en el valor, aparecerán representadas las manos de las cuales tenemos constancia en nuestra base de datos. Es decir, si aparece sombreada de color más oscuro, al menos una vez el rival tenía esta mano cuando realizó una subida en la situación seleccionada.



Como podemos observar, hay dos combinaciones (QJo y T8o) que han sido vistas al menos una vez pero no pertenecen al rango teórico. El resto de manos están dentro de este rango. Así, si sumamos todas las manos que sí que pertenecen al rango y lo dividimos por el total de muestras, obtenemos un porcentaje que nos será útil para medir el grado de polarización de los rangos del rival en esta situación. Este porcentaje aparece en la parte derecha inferior de la pantalla cuando seleccionamos el rango de resubida del rival.

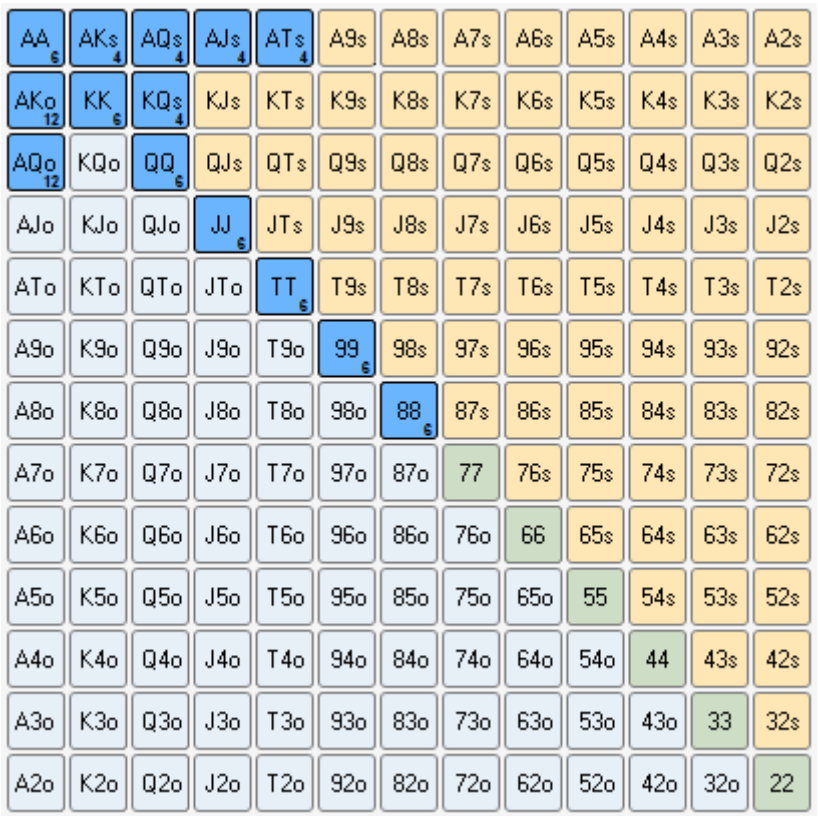


Por lo tanto, cuando hablamos de la distribución de rangos, estamos hablando de como los jugadores polarizan los rangos, que es algo de lo que hablaremos seguidamente. Este valor no lo usaremos a lo largo del proyecto, pero es realmente muy interesante para tener una referencia del juego de un rival en esta sección de visualización de los datos.

Polarización de los rangos

Polarizar los rangos es un hábito muy extendido en el póquer online, hasta el punto que es algo que podemos esperar de todos los jugadores mínimamente competentes.

Entendemos como polarización de rangos el hecho de que el rival no use el rango de manos que representan sus estadísticas. Por ejemplo, si decimos que un jugador resube el 6% de las manos, su rango no polarizado es el 6% de las mejores manos, que es el siguiente:



Aunque parezca que el rango de resubida deba pertenecer a la mejor parte de las manos (¿Qué sentido tendría apostar más dinero con peores manos?), esto no siempre es así. Este podría ser un ejemplo de rango polarizado, que también es el 6% de las manos.

Entonces, ¿qué sentido tiene polarizar los rangos?

Imaginemos la siguiente situación:

Nos encontramos en BB y recibimos un RFI del jugador en BTN. La situación es la misma que la que hemos estudiado anteriormente. Tenemos tres opciones: Nos retiramos, pagamos o resubimos la apuesta. Antes de tomar una decisión, nos centramos en estudiar el tipo de rival que realiza un RFI.

El rival realiza un RFI en esta situación con el 40% de las manos. Si decidimos resubir la apuesta, el rival va a retirarse un 70% de las veces y volverá a resubir el restante 30%. Sabemos que su rango no está polarizado, es decir, que el 30% de las manos con las que volverá a subir son la mejores dentro del 40%. Es decir, volverá a resubir con el 40%*0.3 = 12% de las manos.

Si nosotros tenemos una mano como K♥ Q♥, podemos ver que nuestra mano es mejor que el 40% de las manos que el rival está decidiendo jugar.

	Mesa:			
	Equity	**Ganar**	**Empate**	
MP2	56.34%	54.70%	1.64%	KQs
MP3	43.66%	42.03%	1.64%	44+, A2s+, K2s+, Q4s+, J7s+, T7s+, 97s+, 87s, A3o+, K7o+, Q8o+, J8o+, T8o+

Nuestra posibilidad de ganar a largo plazo contra su rango es del 56.34%.

Ahora bien, si resubimos la apuesta (3-bet), pueden suceder dos escenarios:

El rival abandona: Ganamos el tamaño de apuesta de su RFI.
El rival decide volver a subir (4-bet): Tendremos que volver a tomar una decisión.

El problema es que si nos enfrentamos a una nueva resubida, esta vez sabemos que el rango es el 12% que hemos calculado antes, y nuestra mano ya no es favorita contra ese rango.

	Mesa:			
	Equity	**Ganar**	**Empate**	
MP2	45.34%	41.80%	3.55%	KQs
MP3	54.66%	51.11%	3.55%	77+, A9s+, KTs+, QTs+, JTs, ATo+, KJo+

Nuestra posibilidad de ganar a largo plazo contra su nuevo rango es del 45.34%.

Es decir, resubiendo nuestra mano, hemos creado una situación adversa para nosotros: Lo que era una situación inicialmente favorable, ahora es una situación desfavorable y encima hemos tenido que asumir el coste de resubir la mano.

Aquí es cuando toma todo el sentido polarizar los rangos, sobre todo contra este tipo de rivales que no pagan las resubidas (call to 3-bet), sino que prefieren resubir nuevamente (4-bet).

Si decidimos simplemente pagar la apuesta del rival con manos como la del ejemplo, nos llevaremos las manos favoritas frente a su rango a las siguientes fases del juego.

Sin embargo, como queremos seguir resubiendo manos , porque **hacerlo es rentable por su porcentaje de abandonos**, podemos incrementar nuestro rango de 3-bet con manos que llamaremos “manos basura” como  ,  o similares. Estas manos tienen un valor esperado muy bajo pero como esperamos obtener una rentabilidad directa, la fuerza de nuestra mano es irrelevante para el caso y en rondas posteriores, tenemos decisiones mucho más sencillas, ya que difícilmente nuestras malas manos ligaran una mano buena y retirarnos será la mejor opción. En cierta forma, estamos polarizando nuestras decisiones, ya que mientras con manos buenas podemos tener situaciones complicadas contra el rango del rival, este tipo de manos suelen ser más fáciles de jugar en rondas posteriores (o llevamos la mejor mano con un porcentaje alto de probabilidad o nuestras opciones de ganar son mínimas).

En reglas generales, polarizar los rangos no suele ser una buena práctica por dos razones. La primera es porque, aunque para ejemplos como hemos visto el modelo es perfecto, si el rival decide simplemente pagar nuestras apuestas, nuestro valor esperado resubiendo cae drásticamente y afectará muy negativamente a nuestras ganancias. La segunda razón es que hacerlo implica estar muy atento a como pueda adaptarse el rival en un futuro: Si

nosotros nos fiamos ciegamente de sus estadísticas y el rival se da cuenta de nuestra forma de juego y se adapta nuevamente, nos puede perjudicar de una forma muy fácil y muy costosa para nosotros.

Esto solo tiene validez en la ronda de apuestas iniciales que es la que tratamos en este proyecto y en algunas situaciones en rondas posteriores. En la última fase del juego (el river), al no quedar más cartas por aparecer, los rangos se polarizan de forma natural, puesto que las probabilidades en el river son 100 y 0 (salvo en los pocos casos de empate). En este caso, apostar con las manos mediocres, que van a tirar a sus manos peores y van a ser pagadas por manos mejores, no tiene ningún sentido. Por tanto, polarizar los rangos en el river tiene mucho sentido.

Parte 3: La simulación

La simulación

La tercera parte del proyecto consiste en una simulación de juego creada específicamente para demostrar la eficacia del cálculo de rangos de una estrategia en base a la información del rival. Esta simulación es una simplificación del ambiente real de juego, ya que simular el juego en su totalidad no es práctico y no nos ofrece ninguna ventaja a la hora de demostrar la importancia de nutrir nuestras decisiones con información de los rivales. Lo que si hemos de tener en cuenta es que la forma de simplificar el juego no puede suponer una ventaja ni una desventaja para ningún jugador en concreto y debe permanecer invariable para el resultado final de cada estrategia.

Anteriormente ya hemos realizado algún cálculo para ver, por ejemplo, cual era el valor mínimo de abandono de un rival, para que realizar según que movimiento fuese matemáticamente rentable. En esta simulación, crearemos un escenario donde calcularemos para cada acción del rival una decisión razonada que tenga el mejor valor esperado a largo plazo y argumentaremos porque.

Antes de empezar, simplificaremos el escenario sobre el cual se desarrollará la partida para minimizar las posibles combinaciones del juego que consideremos innecesarias para nuestro propósito. Así, consideraremos las siguientes modificaciones.

1) La mesa tendrá cuatro posiciones. La razón por la cual vamos a reducir el número de posiciones es para simplificar los cálculos y realizar la simulación más rápidamente. Además, al reducir las posiciones, los jugadores se encuentran constantemente en una guerra por rentabilizar sus movimientos, y por consiguiente, los rangos que se juegan son mucho más amplios y la estrategia empleada por cada jugador es más explotable. Este cambio afecta a todos los jugadores de la misma forma, ya que siempre que eliminamos posiciones en una mesa lo hacemos empezando por las primeras que es donde los rangos son más estrechos, hay menor varianza y menor diferencia entre los diferentes perfiles y estrategias.
2) Limitaremos los tipos de movimientos que se pueden realizar y no tendremos en cuenta ningún tipo de apuesta más allá del 3bet. Es decir, una vez un jugador realice un 3bet, este se considerará como una apuesta all-in y el jugador que realiza la primera subida deberá decidir si paga la apuesta o no. A continuación vamos a enumerar los posibles movimientos que pueden suceder en la simulación

a. RFI
b. RFI + Call
c. RFI + Call + 3bet
d. RFI + 3bet

Nuevamente estos cambios afectan a todos los jugadores por igual, ya que las acciones que se pueden tomar en el juego solo son importantes a nivel del número de apuestas posibles. De esta manera permanece intacta la secuencia de acciones que estudiamos en este proyecto y simplemente estamos limitando el hecho de poder realizar resubidas a un 3-bet (4-bet) y, por lo tanto, estamos simplificando los movimientos disponibles para apuestas con tamaños grandes pero dejando intactos los ratios de riesgo/recompensa.

En la práctica, esto sucede para los jugadores que juegan con tamaños de stack de entre 10 y 15 ciegas grandes. Al realizar un 3bet, la relación entre la cantidad ya invertida para realizar la resubida y la que queda por pagar para realizar el all-in completo es tan pequeña respecto a la equity de la mano, que se considera que el jugador ya está comprometido con el bote y, por lo tanto, se supone igual realizar un 3bet como un all-in.

3) Los rangos de RFI de los rivales supondremos que no están polarizados. Anteriormente ya argumentamos porque, tanto desde el punto de vista lógico como en la práctica, esto no tiene ningún tipo de sentido.
4) Los rangos de 4bet y call3bet tampoco estarán polarizados en la simulación por la propia simplificación que realizamos. Vamos a entender que ambos rangos significan una voluntad de seguir en la mano y, al implicar apostar más dinero, los unificaremos como uno solo.
   Esta simplificación no es tan trivial como pueda parecer a simple vista. La razón es que al ser dos movimientos diferentes que dan rentabilidades diferentes podemos estar infravalorando o sobrevalorando la ventaja competitiva que un rival puede obtener o bien de realizar un movimiento o el otro. Esta ventaja solo se puede comprobar analizando el juego postflop, que es algo que no realizaremos en este proyecto, así que realizar la simulación con los datos de jugadores que tengan tamaños de apuestas suficientes como para llegar a fases tardías del juego puede implicar no tener en cuenta toda la rentabilidad real que un jugador puede obtener al realizar diferentes tipos de movimientos respeto al tipo de rival. Aún así, esta diferencia es mínima, puesto que en ambos casos, el rival está dispuesto a pagar la apuesta, y, en términos cuantitativos, la situación permanece inalterable.

Crear una simulación

A continuación vamos a describir el proceso para crear una simulación, modificar los parámetros que usaremos y poder observar los resultados.

En el menú de la izquierda podemos encontrar el acceso a la creación de una simulación. Si accedemos, veremos un listado de parámetros. Estos datos son los siguientes:

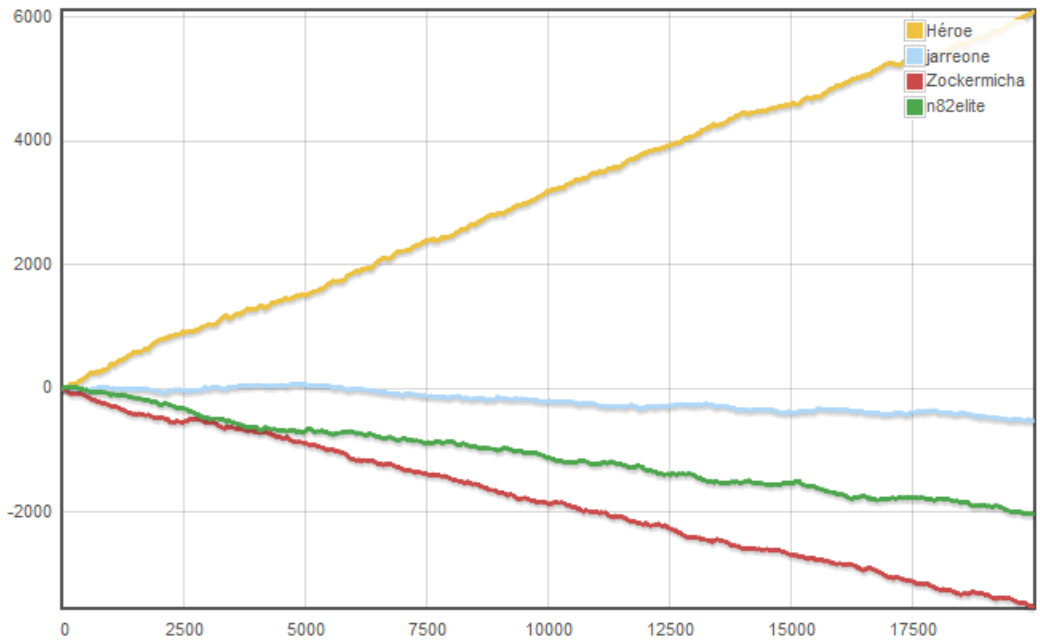
- Número de manos: Número de manos que tendrá la simulación (o número de rondas)
- Tamaño de stack: El tamaño de stack con el que jugarán todos los jugadores. Este dato se introduce en BB como unidades de apuesta.
- Tamaño de Raise First In: El tamaño del RFI por parte de cualquier jugador, en BB como unidades de apuesta.
- Los 3 jugadores que formarán parte de la simulación junto con la estrategia

Es importante resaltar que el tiempo de ejecución de la simulación es directamente proporcional al número de manos que pongamos como parámetro. El resto de variables no condicionan este tiempo.

Al ejecutar la simulación veremos un gráfico como el siguiente:

Simulación

Gráfico de simulación



Datos Tiempo de ejecución: 25.6 segundos

Winrate *Héroe*: **30.5150275** bb/100
Winrate *jarreone*: **-2.65476** bb/100
Winrate *n82elite*: **-10.1772075** bb/100
Winrate *Zockermicha*: **-17.68306** bb/100

Esto es el resultado final de la ejecución de la partida durante el número de manos indicadas en los parámetros de ejecución. El gráfico muestra los beneficios de cada jugador a lo largo de toda la partida, en cada una de las manos. En la parte inferior, tenemos el resultado de la partida indicado en ciegas grandes por 100 manos. Esta es una medida estándar para calcular el margen de beneficio que obtiene un jugador con su estrategia, independientemente del valor de la apuesta mínima, número de jugadores o otros factores.

Funcionamiento interno de la simulación

La simulación que ejecutamos está basada en los datos reales de los jugadores que previamente hemos analizado y estudiado en la primera sección del proyecto. Los datos que usaremos los escogemos de la base de datos que ya explicamos con anterioridad.

Para la simulación tenemos tres jugadores de la base de datos y la estrategia que hemos diseñado para este proyecto. Es importante matizar que, para el objetivo de este proyecto, cada jugador usará la misma estrategia en toda la partida mientras que la que hemos diseñado aquí será capaz de adaptarse al resto de estrategias en base de a los datos del resto de jugadores. De esta forma, obtendremos una ventaja competitiva frente a nuestros rivales y deberíamos poder tomar mejores decisiones que ellos. Esto nos comportará unos mayores beneficios a largo plazo que deberíamos observar en los resultados de la simulación. De ahora en adelante, hablaremos de jugadores cuando hagamos referencia a los que hayan sido seleccionados para la simulación y hablaremos de HERO (Héroe) para referirnos a la estrategia diseñada por nosotros.

A continuación vamos a explicar paso por paso el algoritmo interno que usamos para crear cada mano dentro de la simulación. Este paso se va a repetir tantas veces como hayamos definido en los parámetro.

1) Parámetros iniciales. Aparte de los parámetros que nos ofrece el usuario, hemos de definir unos cuantos más antes de empezar la simulación y el proceso de cada mano.
   El primero de todos será la posición que tomará cada jugador en la mesa. Los tres jugadores elegidos para la simulación estarán sentados en el mismo orden que han sido seleccionados en los parámetros iniciales y HERO inmediatamente después. Para cada jugador que no sea HERO, crearemos un perfil con todos los datos de su estilo de juego. Este perfil define exactamente la estrategia de juego del jugador y lo usaremos en cada mano. Así no tendremos que consultar en cada decisión los datos en la base de datos. También crearemos una estructura de datos vacía para almacenar el cómputo de perdidas y ganancias para cada jugador en cada mano. No guardaremos más información porque no es el objetivo de este proyecto saber que hacen los jugadores en cada mano en concreto, sino ver los resultados de la simulación a largo plazo y las tendencias.

2) El siguiente paso consiste en empezar con el reparto de cartas para la primera mano de la simulación. Las cartas son repartidas aleatoriamente, respetando las probabilidades propias de un reparto real. No habrá ningún tipo de distinción entre si un jugador recibe, por ejemplo, un 2 de corazones y un 3 de corazones que si recibe un 2 de picas y un 3 de picas. Diremos que recibe "32s" que es la forma más correcta que decir que recibe la combinación de la carta 3 y la carta 2 del mismo palo. Esta simplificación nos ahorrará tener en cuenta cartas muertas en el conteo de probabilidad y cálculo de la equity en una mano en concreto y así, de paso, minimizar los efectos de la varianza en el propio cálculo.

3) El siguiente paso consiste en ver si el primer jugador en tomar una decisión es uno de los jugadores seleccionados para la simulación o es HERO. Dependiendo del caso tomaremos una decisión u otra.
    a. Si no es HERO quien toma una decisión, miraremos la posición en la cual se encuentra el jugador y consultaremos los datos de su perfil para ver que dato de RFI tiene. Si sus dos cartas privadas entran dentro del rango definido por este dato, deduciremos que es una mano que el jugador esta dispuesto a jugar y realiza un RFI. En caso contrario, el jugador se retira de su mano.
    b. Si el jugador que tiene que tomar una decisión es HERO, llamaremos a la función shouldRFI_Hero() para decidir si se realiza un RFI o es mejor abandonar la mano. En ambos casos, pasamos al siguiente punto.

4) Una vez el jugador ha tomado una decisión, pasamos al siguiente jugador de la mesa. Este paso se repetirá hasta que suceda una de las tres siguientes opciones:
    a. Si anteriormente ningún jugador ha realizado un RFI, actuaremos igual que en el punto 3, teniendo en cuenta la nueva posición de la mesa, el jugador y los datos.
    b. Si nos encontramos en la penúltima posición de la mesa y el resultado de ejecutar el punto 3 no responde con un RFI y anteriormente no ha habido ninguno, la mano finaliza, ya que todos los jugadores han decidido retirarse hasta llegar a la posición de BB, quien gana la mano sin tomar ninguna decisión.
    c. Si anteriormente ha sucedido un RFI, pasamos al punto 5.

5) El jugador de la posición actual se enfrenta a un RFI por parte de otro jugador. En este punto vamos a distinguir entre tres tipos de opciones:
    a. Si el jugador es HERO, llamaremos a la función should3bet_Hero(). Si esta función devuelve un valor verdadero, HERO realizará un 3bet y pasaremos al siguiente punto. Si esta función devuelve un valor falso, llamaremos a la

función shouldCall_Hero(). En caso de que esta función devuelva un valor verdadero, HERO pagará la apuesta y pasaremos al siguiente jugador, en caso contrario HERO se retirará de la partida.

b. Si el jugador no es HERO, consultaremos los datos de su perfil para ver que decisión tomar. En el perfil encontraremos todos los datos referentes a los rangos con los cuales el jugador realizará una resubida dependiendo de la posición desde donde provenga el RFI. Si la mano del jugador entra dentro de este rango, realizará un 3bet. En caso contrario, deberemos ver el rango de manos con las que el jugador esta dispuesto a pagar la apuesta en vez de realizar una subida. De nuevo, si la mano del rival entra dentro de este rango, el jugador pagará la apuesta y en caso contrario, abandonará la mano.

c. Si llegados a la última posición no se ha realizado un 3bet o se ha pagado la apuesta, la mano termina, ya que ningún jugador ha decidido enfrentarse al RFI. En caso contrario, pasaremos al punto número 6.

6) Llegados a este punto, en la mano ha habido un RFI y algún jugador ha realizado un 3bet o ha pagado la apuesta. Vamos a diferenciar entre estas dos situaciones:

   a. Si el jugador ha pagado la apuesta, el juego preflop termina, ya que no ha habido ninguna subida más aparte del RFI del jugador inicial. Por lo tanto podemos pasar al punto número 7.

   b. Si el jugador ha realizado una resubida, la mano no termina, ya que el jugador que ha realizado el RFI tiene que tomar una decisión sobre la apuesta a la cual se enfrenta. Por lo tanto, miraremos en el perfil del jugador los datos acerca del porcentaje de retiradas y tomaremos una decisión en base a este dato. Si la mano que tiene el jugador entra dentro del rango de manos con las cuales esta dispuesto a pagar o resubir, el jugador pondrá la cantidad de la apuesta que se le exige y podremos pasar al punto número 7. Si esta mano no está dentro de este rango, significa que abandona y la mano termina aquí. Lo mismo sucede si es HERO quien ha realizado el RFI. En este caso, en vez de consultar el perfil del jugador, nos limitaremos a llamar a la función shouldCall3bet_Hero(). Esta función nos devolverá un valor verdadero en caso de que HERO pague la apuesta y un valor falso en caso contrario.

7) En este punto, los posibles movimientos que pueden realizar tanto los jugadores como HERO han terminado y es el momento de realizar el cálculo de la jugada. Vamos a ir por pasos.

a. Primero hay que descontar de las fichas de los jugadores de la mesa el coste de poner la ciega pequeña y la ciega grande y añadir esta cantidad al bote.
b. Seguidamente hay que descontar también el coste de realizar un RFI, un 3bet o pagar una apuesta a aquellos jugadores que lo hayan realizado y añadir esta cantidad al bote.
c. Ahora que el bote es consistente y contiene todas las fichas que se han usado en las apuestas, tenemos que diferenciar dos tipos de situaciones.
    i. Aquellas en las cuales termina la partida. Dentro de esta parte englobamos las jugadas cuyas apuestas no reciben contrapartida. Las posibles combinaciones son:
        1. Nadie realiza un RFI y BB gana la partida.
        2. El jugador realiza un RFI y nadie iguala o resube la apuesta.
        3. El jugador realiza un RFI, otro jugador realiza un 3bet y el primer jugador se retira.
    ii. Aquellas en las cuales no termina la partida porque los jugadores han igualado las apuestas, es decir, existe una contrapartida a la apuesta de algún jugador. Las posibles combinaciones son:
        1. El jugador realiza un RFI y otro jugador paga la apuesta.
        2. El jugador realiza un RFI, otro jugador realiza un 3bet y el primero iguala el 3bet.
d. Para aquellas manos en las cuales no termina la partida, tenemos que realizar el cálculo de cómo termina la partida. Como hemos explicado anteriormente, repartiremos la cantidad de fichas del bote entre la equity que tengan las dos manos. Así, por ejemplo, si en una mano donde ha habido contrapartida, la equity de la mano del primer jugador es 40% y la del segundo 60% y el bote es de 10 fichas, el primer jugador recibirá 4 fichas y el segundo 6.
e. Después de todos los cálculos, de la misma forma que hemos descontando el valor de las fichas de los jugadores por sus apuestas, ahora vamos a sumar la cantidad que han ganado una vez finalizada la mano. Esta información queda guardada en una estructura de datos que se va actualizando mano tras mano y donde va quedando reflejado el resultado de todas las manos. Guardaremos dos tipos de estructuras, una para mostrar la información acumulada hasta la última mano y otra para guardar el resultado de cada mano y luego poder mostrar una gráfica con los resultados. Esto nos permitirá ver la tendencia y no solo el resultado final.

Funciones específicas de la estrategia explotadora

Tal como hemos visto en el funcionamiento interno de la simulación, nos basamos en los datos que tenemos de los rivales para saber qué decisión tomar en función de las cartas repartidas en cada mano.

Cuando es el turno de HERO, usaremos las funciones específicamente programadas para realizar los cálculos sobre qué decisión tiene el mejor valor esperado.

Estas funciones tendrán unos parámetros de entrada y unos parámetros globales. Estos últimos son los que ha definido el usuario antes de ejecutar la simulación y son los siguientes:

- Tamaño de stack (**STACKSIZE**)
- Tamaño de RFI (**ORSIZE**)

En el caso de que HERO reciba la mano en el punto número 3 de la simulación, deberemos tomar una decisión sobre si realizar un RFI o no, así que tenemos que calcular si realizarlo es rentable o no. Esta función la hemos llamado shouldRFI_Hero() y vamos a describir su funcionamiento.

shouldRFI_Hero()

Los parámetros necesarios para conocer el estado de la partida son:

- La posición (**POSITION**).
- Los jugadores que quedan por hablar y su información (**PLAYERS**).
- Las cartas privadas de HERO (**HOLECARDS**).

Para acceder a cada dato de cada jugador lo haremos desde **PLAYERS**, indicando en cada momento que información estamos tomando.

Recordemos que el valor esperado de retirarnos (es decir, de no realizar un RFI) es cero. Por lo tanto, cuando realicemos un RFI tenemos que asegurarnos que nuestro valor esperado es mayor que cero. Este cálculo no es trivial, ya que cuando nosotros realizamos un RFI, en esta simulación, puede suceder que

1. Los rivales abandonen.
2. Un rival pague la apuesta.
3. Un rival resuba la apuesta (3bet).

Por lo tanto, hemos de calcular cual será nuestro coste de realizar el movimiento y ponderarlo con el beneficio potencial a largo plazo de cada situación.

Para calcular el beneficio cuando los rivales abandonan, hemos de ver la probabilidad de abandono y ponderarlo con el valor de las apuestas ya realizadas (las ciegas).

```
pct_folds = 0;
num_jugadores = 0;
ev = 0;

Para cada jugador de PLAYERS Hacer
  pct_folds += jugador[vsPOSITION_Fold];
  num_jugadores++;
Fin para

Si POSITION = "SB" Entonces
  beneficio = 1;
Si no Entonces
  beneficio = 1.5;
Fin si

ev = (pct_folds/num_jugadores)*beneficio;
```

Como podemos ver, el EV es siempre positivo porque solo estamos calculando las veces que el conjunto de jugadores se retira.

Vamos a calcular a continuación el EV de las veces que algún jugador nos pague la apuesta.

```
equity = 0;
pct_call = 0;
num_jugadores = 0;

Para cada jugador de PLAYERS Hacer
  equity += equity(HOLECARDS, jugador[vsPOSITION_Call]);
  pct_call += jugador[vsPOSITION_Call];
  num_jugadores++;
Fin para

equity_call = equity / num_jugadores;

Si POSITION = "SB" Entonces
  ev_call = ((ORSIZE*2) * equity_call) – (ORSIZE-0.5);
Si POSITION = "BTN" Entonces
  ev_call = (((ORSIZE*2)+0.75) * equity_call) – ORSIZE;
Si no Entonces
  ev_call = (((ORSIZE*2)+0.5) * equity_call) – ORSIZE;
Fin si

ev += (pct_call/num_jugadores) * ev_call;
```

La función equity(H1, H2) nos va a devolver la probabilidad de ganar de H1 vs H2. En este caso hemos abusado de la sintaxis con el pseudocódigo y enfrentamos las cartas de HERO contra el rango del rival. En la práctica, hemos de enfrentar todas las combinaciones del rango del rival con la combinación de HERO.

El valor que hemos calculado finalmente hay que sumarlo al EV de la anterior operación. Este resultado sí que puede ser negativo, ya que si el rango del rival que paga es mucho más fuerte que la mano de HERO, el valor de equity_call será bajo y reducirá el beneficio. Al restar el coste de realizar el RFI, el valor puede ser negativo.

Vamos a calcular ahora el EV de las veces que un jugador nos resuba la apuesta (3bet).

```
equity = 0; equity_needed = 0; pct_3bet = 0; num_jugadores = 0;

Para cada jugador de PLAYERS Hacer
  equity += equity(HOLECARDS, jugador[vsPOSITION_Raise]);
  pct_3bet += jugador[vsPOSITION_Raise];
  num_jugadores++;
Fin para

Si POSITION = "SB" Entonces
  equity_needed = (STACKSIZE-(ORSIZE-0.5)/(STACKSIZE*2);
Si POSITION = "BTN" Entonces
  equity_needed = (STACKSIZE-ORSIZE)/(STACKSIZE*2 + 0.75);
Si no Entonces
  equity_needed = (STACKSIZE-ORSIZE)/(STACKSIZE*2 + 0.5);
Fin si
equity_3bet = equity / num_jugadores;

Si equity_needed < equity_3bet Entonces
  Si POSITION = "SB" Entonces
    ev_raise = ((STACKSIZE*2) * equity_3bet) - (STACKSIZE-0.5);
  Si POSITION = "BTN" Entonces
    ev_raise = (((STACKSIZE*2)+0.75) * equity_3bet) - STACKSIZE;
  Si no Entonces
    ev_raise = (((STACKSIZE*2)+0.5) * equity_3bet) - STACKSIZE;
  Fin si
Si no Entonces
  Si POSITION = "SB" Entonces
    ev_raise = -(ORSIZE - 0.5);
  Si no Entonces
    ev_raise = -ORSIZE;
  Fin si
Fin si

ev += (pct_3bet/num_jugadores) * ev_raise;
```

A diferencia de los dos otros cálculos, esta vez tenemos que tener presente que para calcular el EV del RFI cuando un rival nos resube, debemos calcular de antemano si pagaremos una resubida o no, ya que el coste y beneficio no es el mismo al realizar un RFI y pagar la apuesta que al realizar un RFI y no pagar la apuesta.

Por lo tanto, para saber si debemos pagar la resubida o no, calculamos primero el equity que necesitamos para pagar la apuesta una vez realizado el RFI. Esto es importante, porque una vez realizada la primera apuesta, hemos asumido un coste. También calcularemos el equity_3bet de nuestra mano contra el rango del rival. De esta forma, si el equity_needed (el equity que necesitamos dada nuestra inversión) es menor al equity_3bet (el equity de nuestra mano versus el rango del rival), nuestro RFI será con la intención de pagar el 3bet, y, por lo tanto, hemos de calcular el EV de esta situación. Si el equity que necesitamos es mayor, nos retiraremos de la mano y asumiremos en el EV el coste de realizar el RFI.

Una vez tenemos calculada el EV de cada situación y lo hemos ponderado con la probabilidad de cada situación, simplemente tenemos que decidir si realizamos el RFI o no.

```
Si ev > 0 Entonces
  devuelve CIERTO
Si no Entonces
  devuelve FALSO
Fin si
```

De esta forma, hemos realizado el cálculo para saber si Hero debe realizar un RFI o no.

Hemos elegido esta función porque es la más completa y la más representativa para mostrar como las funciones evalúan el valor esperado de cada acción. El procedimiento es el mismo para el resto de funciones de la estrategia.

Evaluación de la estrategia

Para poder evaluar la estrategia ejecutaremos varias simulaciones con diferentes rivales y compararemos los resultados.

Para empezar, vamos a escoger una muestra de 20000 manos, stacks de 30BB y tamaño de RFI de 3BB y realizaremos varias ejecuciones con los mismos rivales, para ver cuáles son los resultados. Hemos elegido a los 3 jugadores con más muestras en la base de datos:

	SIM1	SIM2	SIM3	SIM4	SIM5	SIM6	SIM7	SIM8	SIM9
HERO	10,7	12,4	8,7	10,9	13,7	12,2	11,2	12,7	7
jarreone	1,9	-1,9	1,4	-1,4	-0,4	-1,1	-0,7	-0,9	1,2
feedmesvp	-12,8	-9,5	-10,5	-10	-11,3	-11,1	-10,2	-13	-7
learnfriend	0,2	-1,1	0,4	0,4	-2	-0,1	-0,3	1,2	-1,2

Podemos ver como el winrate de HERO es en todas las simulaciones bastante mayor al del resto de rivales y que los winrates de las simulaciones son parecidos entre sí, lo que nos muestra claramente como la estrategia implementada puede explotar a unos jugadores más que a otros y lo hace a un ritmo parecido en cada simulación.

Si buscamos en el programa la información de estos tres jugadores, veremos por qué la estrategia está explotando a unos más que a otros.

Jarreone

	EP	MP	CO	BTN	SB
raiseFirstIn	19	23	30	40	31
fold3bet	49	50	58	62	60
4bet	9	10	11	9	8
call3bet	42	40	31	29	32

Feedmesvp

	EP	MP	CO	BTN	SB
raiseFirstIn	19	20	31	68	49
fold3bet	60	57	62	72	73
4bet	12	16	17	9	16
call3bet	28	26	21	19	11

Learnfriend

	EP	MP	CO	BTN	SB
raiseFirstIn	12	14	24	46	54
fold3bet	70	67	68	62	71
4bet	22	24	22	21	19
call3bet	7	8	10	17	9

Si nos fijamos en los datos de las posiciones CO, BTN y SB (que son las que están involucradas en la simulación), vemos que en la primera fila (RFI), los valores del segundo jugador son más altos que los de los otros jugadores. Además, en la segunda fila (Fold al 3bet), vemos como los valores del segundo jugador también son ligeramente más altos.

Por lo tanto, el segundo jugador está realizando un RFI con un rango de manos mucho más amplio del resto de jugadores y, por lo tanto, abandona gran parte de él cuando recibe una resubida. Por lo tanto, la estrategia está explotando esta debilidad del jugador, y por eso es el jugador que peor winrate obtiene en la muestra.

Como la mayor parte del beneficio proviene de explotar a los rivales por sus altos niveles de abandono al 3bet, vamos a ejecutar nuevamente el mismo número de simulaciones, con los mismos parámetros y los mismos rivales, pero esta vez pondremos el tamaño de stack a la mitad: 15BB.

El resultado es el siguiente:

	SIM1	SIM2	SIM3	SIM4	SIM5	SIM6	SIM7	SIM8	SIM9
HERO	50	48	54	52	56	55	52	55	55
jarreone	6	4	-6	-4	-7	-5	-4	-6	-4
feedmesvp	-28	-30	-29	-32	-28	-28	-30	-29	-27
learnfriend	-16	-10	-18	-19	-21	-22	-16	-20	-20

Como podemos ver, los resultados se polarizan muchísimo, ya que la estrategia explotadora consigue en este escenario quintuplicar el ritmo de beneficios respecto a la anterior simulación. Esto sucede porque al realizar las resubidas, el beneficio que obtenemos por abandono de los rivales es el mismo que en la otra simulación, pero el riesgo que asumimos, en unidades de apuesta, es exactamente la mitad. Por eso, al realizar los cálculos de EV en el momento de realizar una resubida, podemos asumir mucho más riesgo y explotar más la debilidad de cada jugador.

Ahora vamos a modificar el tamaño de stack y vamos a aumentarlo. Por la misma lógica, como ahora explotar los fallos de resubida va a ser más costoso, en principio no deberíamos obtener tanta rentabilidad como en el segundo caso.

Vamos a realizar otra ronda de simulaciones de 20000 manos, stacks de 100BB y tamaño de RFI de 3BB y realizaremos varias ejecuciones con los mismos rivales que en los otros ejemplos.

El resultado es el siguiente:

	SIM1	SIM2	SIM3	SIM4	SIM5	SIM6	SIM7	SIM8	SIM9
HERO	17	12.5	16.5	13	15	12	16	16	18
jarreone	-5	-2	-6	3.5	1	-1	-7.5	-0.5	2
feedmesvp	7	4	-3.5	4	7	-6	-3.5	-1	-1
learnfriend	-18	-14	-13	-21	-23	-17	-12.5	-17.5	-19

Este resultado es peor que el segundo pero mejor que el primero y este resultado se repite (proporcionalmente) si modificamos el tamaño de stack hacia valores como 200, 500 o 1000. Estos valores son absurdamente altos para la simulación, ya que partimos de escenarios muy limitados y estos valores tan altos carecen de sentido.

Pero aun así hay que analizar cada situación con detalle. Vemos como el jugador "feedmesvp" que estaba obteniendo peores rendimientos en los dos primeros ejemplos es ahora el que mejores resultados obtiene después de HERO. Esto resulta extraño, pero al ver los datos de las otras simulaciones y los del rival, vemos como anteriormente hemos explotado el fallo del fold3bet alto, realizando muchas resubidas. Por lo tanto, como ahora la resubida tiene un coste alto para nosotros, no podemos usar este recurso tan a menudo para explotar este fallo y el rival recupera parte del winrate que perdía en las otras simulaciones. Veamos el caso magnificado realizando otra ronda de simulaciones de 20,000 manos, stacks de 1000BB esta vez y tamaño de RFI de 3BB y realizaremos varias ejecuciones con los mismos rivales.

	SIM1	SIM2	SIM3	SIM4	SIM5	SIM6	SIM7	SIM8	SIM9
HERO	81	75	77	69	68	71	85	81	63
jarreone	-21	-20	-56	-32	-45	-39	-45	-42	-13
feedmesvp	46	57	59	49	67	66	38	44	47
learnfriend	-105	-100	-79	-82	-89	-90	-78	-85	-96

Como podemos apreciar, el resultado es parecido al anterior, simplemente que hemos aumentado los márgenes de beneficio. En este caso, y como hemos argumentado antes, como en estas condiciones realizar una resubida es costosa, tanto HERO como "feedmesvp" salen beneficiados.

Tras realizar muchas simulaciones con diferentes tamaños de stack y de RFI y valores coherentes al estilo de juego en las mesas online, vemos como la estrategia mejora muchísimo para valores de stack bajos y RFI bajos. Esto demuestra que la estrategia está intentado explotar al máximo los errores de los rivales porque consigue unos márgenes de beneficio muy elevados cuando el coste por realizar un RFI es bajo (ORSIZE) y el coste de realizar un 3bet también lo es (STACKSIZE). De esta forma, el beneficio proviene de ir

aprovechando la cantidad de retiradas de los rivales frente a una subida o resubida por parte de HERO.

En las conclusiones del proyecto haremos una valoración más extensa de porque no podemos obtener márgenes más elevados con tamaños de stack grandes y las limitaciones lógicas de tratar simplemente la primera parte del juego.

Almacenaje de las simulaciones

Anteriormente hablamos de la estructura de la base de datos del proyecto y explicamos las diferentes tablas que contenía para poder almacenar toda la información que necesitamos para hacer los cálculos de la estrategia explotadora.

Una de estas tablas contiene toda la información de cada simulación que ejecutemos. En ella guardaremos datos referentes a los parámetros de la simulación y los resultados de esta.

Los resultados de la simulación que nos importan son dos: el ratio de ganancias/pérdidas de cada jugador, que llamaremos winrate, y el estado final de cada mano simulada.

El winrate es un valor que define cuantas unidades de apuesta (BB) se gana cada 100 manos. Representa, también, la pendiente de la recta de nuestros beneficios en una gráfica. Este ratio está normalizado para que no importe cual es el valor de la unidad de apuesta y por lo tanto sirve para medir el rendimiento de una estrategia.

Realmente no necesitamos nada más para poder evaluar los resultados de una simulación, ya que si ejecutamos una muestra suficiente de manos como para que el resultado no se pueda ver afectado por la varianza, el winrate obtenido ya es un indicador válido para tomar una decisión cualitativa sobre una estrategia.

No obstante, guardaremos también el estado final de cada mano simulada y guardaremos esta información en una estructura de datos muy sencilla en un archivo. En la base de datos, para cada simulación, guardaremos el nombre de este archivo para poder recuperarlo en cualquier momento. Este archivo lo guardaremos fuera de la base de datos porque las simulaciones con miles de manos ocupan bastante espacio, lo suficiente como para ralentizar el funcionamiento de la base de datos. Como solamente queremos esta información para poder redibujar el gráfico de la simulación, guardaremos la cantidad de unidades de apuesta que llevan de beneficio/pérdida todos los jugadores de la simulación.

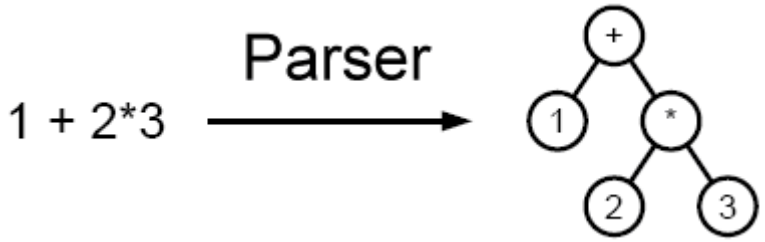
En realidad, podríamos guardar una infinidad de datos más, como, por ejemplo, cuáles han sido las manos repartidas, que acción ha realizado cada jugador, etc. Esto podría ser interesante para estudiar qué tipo de comportamientos toma la estrategia en base a cualquier jugador, pero en este proyecto nos limitaremos a estudiar cada simulación con las tendencias que quedan dibujadas en el gráfico resultante de cada simulación y evaluaremos su potencial comparando los winrates, que es la forma más sencilla y directa.

Conclusiones

Para este proyecto hemos tomado muchas decisiones para poder llegar a nuestro objetivo final. Estas decisiones no han sido fáciles ni arbitrarias y han supuesto algunos problemas añadidos.

Primeramente, hemos tomado la decisión de usar un software de pago y hemos decidido lanzar consultas a la base de datos que usa. Aunque a largo plazo hubiese sido una opción hacer nosotros mismos el "parser" para montar nuestra propia estructura, en la práctica es mucho más complicado de lo que parece.

Primero, cada proveedor de servicios de juego online guarda los datos de la forma que le resulta más conveniente. Esto nos obligaría a realizar un "parser" por cada sala de juego. Además, la información de cada mano no es fija ni sigue un patrón perfecto, ya que cada mano puede contener muchas o pocas acciones, pueden participar dos o más jugadores, etc… Por lo tanto, realizar un buen "parser" es muy parecido a realizar un analizador sintáctico de un lenguaje de programación, para conseguir representar los datos en una estructura con la que podamos trabajar.



Según Wikipedia, un analizador sintáctico "es un componente importante de un compilador. Analiza el código fuente de un lenguaje de programación para crear algún tipo de representación interna".

En nuestro caso, el lenguaje sería el resumen textual de la mano y la representación interna sería la estructura que usamos para guardar los datos antes de introducirlos en nuestra base de datos. Como delegamos todo este trabajo a un software de terceros, nos aseguramos que nuestra fuente de datos, que es la misma base de datos del programa, se mantiene siempre coherente y no contiene resultados duplicados y filtra los posibles errores en el proceso. Además permite usar este proyecto en cualquier tipo de proveedor de juego que esté soportada por este software y podemos olvidarnos de las futuras modificaciones que cada sala realice a lo largo del tiempo en sus historiales de juego.

Otra decisión importante en el proyecto ha sido limitar el número de situaciones que hemos estudiado y resuelto en una estrategia óptima. No obstante, hemos recopilado datos del juego preflop de los diferentes jugadores para tomar decisiones en tres escenarios muy habituales en la primera fase del juego.

De la misma manera que si analizamos el juego del ajedrez, la profundidad del estudio de las decisiones es la que marca la dificultad del problema, tanto en términos de computación como de complejidad a la hora de tomar una decisión, en el juego del póquer sucede algo muy parecido.

Es bastante obvio enfocar el proyecto a resolver el problema de la toma de decisiones en **la primera fase del juego** y los motivos son evidentes. En la fase preflop es donde se empiecen a generar movimientos que afectaran a todo el transcurso de una mano. Es decir, independientemente de la fase del juego que queramos estudiar, hay que tener en cuenta siempre que ha sucedido anteriormente. Hacerlo de otra forma es totalmente inútil, pues estaríamos perdiendo una información totalmente vital para establecer un posible rango de manos al rival. Al fin y al cabo, como ya hemos mencionado anteriormente, la esencia del juego se basa en calcular que movimiento es más rentable en base a la información que obtenemos de los movimientos del rival y su estilo de juego. Cuanta más información tengamos y mejor tratamiento le demos, más podremos estrechar el rango de nuestro rival para tomar una decisión de mejor calidad.

Además, la primera fase es, obviamente, la primera. Por lo tanto, en la totalidad de las manos que estudiamos existe esta fase, mientras que el resto no tienen por qué estar presentes. Esto hace que este proyecto sea útil para cualquier mano que queramos estudiar, ya que podemos encontrar una solución o una definición de rango para cualquier mano, ya que siempre existe la fase preflop. Este hecho también facilita la cantidad de información que hemos tenido que manejar, ya que cada mano nos aporta información sobre todos los jugadores presentes, mientras que si hiciésemos un estudio de las fases posteriores tendríamos que descartar manos en las cuales no se ha llegado a dicha fase.

Más compleja es la decisión de limitar la estrategia explotadora para solo tres escenarios, pero la dificultad del proyecto aumenta exponencialmente a medida que queremos añadir más variables en la ecuación. En el proyecto están programadas las funciones para recopilar los datos de todo el proyecto “versus Hero”, pero no las hemos añadido ni en el visualizador de datos ni en la simulación, y tampoco las hemos comentado anteriormente, ya que añaden un nivel de complejidad que no justifica el coste en tiempo del proyecto. La estrategia que hemos obtenido ya es explotadora en base a los datos de los rivales. No obstante hay que decir que añadir estos datos “versus Hero” al cálculo de la función explotadora enriquece la toma de decisiones, ya que se adapta a dos niveles al rival: a su estilo de juego y a su estilo de juego cuando se enfrenta a nosotros.

Como hemos podido ver en las simulaciones, la estrategia que hemos obtenido tiene una ventaja muy superior a los rivales. Aun así, como ya hemos comentado en la evaluación de la estrategia, hay que ser muy precavido con las conclusiones.

Primero de todo, hemos modificado el escenario de la simulación para adaptarlo a los tres escenarios que hemos elegido para el proyecto. Como ya hemos explicado, estas adaptaciones afectan a todos por igual y en ningún momento ningún jugador sale beneficiado o perjudicado por estas.

Pero nuestra estrategia explota al máximo los fallos a largo plazo del jugador y en ningún momento dejamos margen a los rivales a adaptarse frente a nosotros. Obviamente, es imposible realizar esto, ya que no sabemos cómo van a adaptar su juego frente a un jugador que tuviese una estrategia como la propuesta en este proyecto. Lo que está claro, es que todos los jugadores van a realizar cambios a largo plazo a medida que vayan jugando manos contra nosotros. Estos cambios modificaran el winrate de nuestra estrategia y muy probablemente sea en contra nuestra y esto hay que tenerlo en cuenta.

Una conclusión muy clara al evaluar la estrategia y los resultados que hemos obtenido es que cuanto mayor es el tamaño del stack en relación a las apuestas (RFI y 3bet), menos margen de beneficio obtiene la estrategia. Esto es totalmente lógico, ya que nuestros cálculos se basan en tres escenarios iniciales de una mano de póquer y no en situaciones más avanzadas del juego. Esto está directamente relacionado con la profundidad de las decisiones y el stack efectivo de los jugadores respeto las apuestas.

Por ejemplo, para un stack efectivo de 10BB, las posibles decisiones son muy pocas, ya que cada acción nos compromete mucho con el bote. Por lo tanto, la profundidad del juego es baja y lo que prima es el cálculo más aproximado posible del valor esperado de cada acción.

Por el contrario, si el stack efectivo es de 100BB, las posibles decisiones son muchísimas, incluso en la primera fase del juego. Es por esto que la profundidad en este caso es enorme y la importancia de las decisiones que tomamos al inicio son mucho menores, ya que aún quedan muchas apuestas por hacer y podemos obtener una ventaja competitiva mucho mayor en fases más tardías del juego.

Por esto este proyecto toma importancia en el juego con stacks pequeños, ya que la profundidad es muy baja y por lo tanto se adapta a los escenarios en los que trabaja la estrategia explotadora. Además, los cálculos son muy precisos y mientras haya información anterior suficiente, se puede conocer el rango de cualquier movimiento y la probabilidad de este con una precisión muy superior a la de cualquier humano. De esta forma estamos maximizando la ventaja competitiva que nos ofrece el hecho de poder calcular mejor y más rápidamente el valor esperado de cada acción.

Como conclusión final de este proyecto podemos afirmar que hemos conseguido los objetivos propuestos, ya que hemos realizado una estrategia que es capaz de explotar a los rivales usando la información que tenemos de ellos. Además las simulaciones han demostrado que la estrategia es muy eficiente para partidas donde las situaciones estudiadas representen la totalidad de acciones disponibles.

Actualidad y futuro

Seguramente una de las preguntas que nos podemos hacer una vez finalizado este proyecto, es si se puede poner en la práctica y usarlo para tomar decisiones en tiempo real durante el juego.

Aunque sobra decir que esta práctica está prohibida por todas las salas y su uso está penalizado con el cierre total y definitivo de la cuenta de juego, existen hoy en día muchos juegos en los cuáles se cumplen las características que hacen que el proyecto sea interesante.

De hecho, todos aquellos juegos en los cuales el stack sea menor a 15 unidades de apuesta mínima son los escenarios ideales donde poder ejecutar esta estrategia, ya que las decisiones son muy limitadas y los cálculos requieren de mayor precisión. En rangos de stack de entre 15 – 25 unidades de apuesta la estrategia aún es buena sin ser la óptima, mientras que es una estrategia mediocre para rangos entre 25 y 40 y carece de sentido de 40 hacia arriba.

Aumentar la relación entre el dinero para apostar en una jugada y la apuesta mínima es clave para saber la profundidad de nuestras decisiones. Al aumentarla, la profundidad crece exponencialmente. Este proyecto es completo para situaciones de juego tempranas con stacks pequeños. Más allá, habría que modificar la estrategia para poder adaptarla a las siguientes fases del juego.

Estas fases posteriores contienen elementos nuevos que no hemos visto aquí, como las cartas comunitarias, que son vitales para el cálculo de las probabilidades de cada mano, tanto la nuestra como la de nuestro rival. Es por esto que las acciones que existen más allá de la primera fase del juego que hemos analizado se ven condicionadas por el tipo de mesa que salga en las fases posteriores, es decir, ya no solo nos basta con evaluar una mano contra otra, sino en evaluar las acciones en base a las cartas comunitarias y las posibilidades de ganar con las siguientes cartas. Para ello necesitaríamos extraer muchos más datos de los que tenemos y muchísima más información inicial para alimentar la base de datos.

No obstante, **toda la información de este proyecto sigue siendo válida, ya que es independiente de los cálculos de las apuestas**. Los rangos que hemos encontrado y definido siguen siendo el punto de partida inicial y obligatorio para cualquier proyecto parecido. Incluso en juegos donde la ramificación de las decisiones sean más complejas y las acciones iniciales tengan poco peso, la construcción de un rango para el rival es el único camino para poder obtener la ventaja competitiva que hemos buscado constantemente y así poder tomar decisiones de más calidad.

Costes

En este capítulo mostraremos el diagrama de Gantt para ver la planificación del proyecto y la estimación económica de los costes del proyecto.

Análisis económico

Seguidamente tenemos una tabla que muestra los costes de cada parte del proyecto. Hemos separado los distintos roles, aunque en la práctica todas las tareas han sido realizadas por la misma persona.

Hemos separado el coste del proyecto en secciones grandes, aunque en el diagrama de Gantt aparecen desglosadas las tareas. Además, hemos añadido los costes adicionales asociados al proyecto, como la licencia del software de pago y las compras por Internet de los historiales de manos.

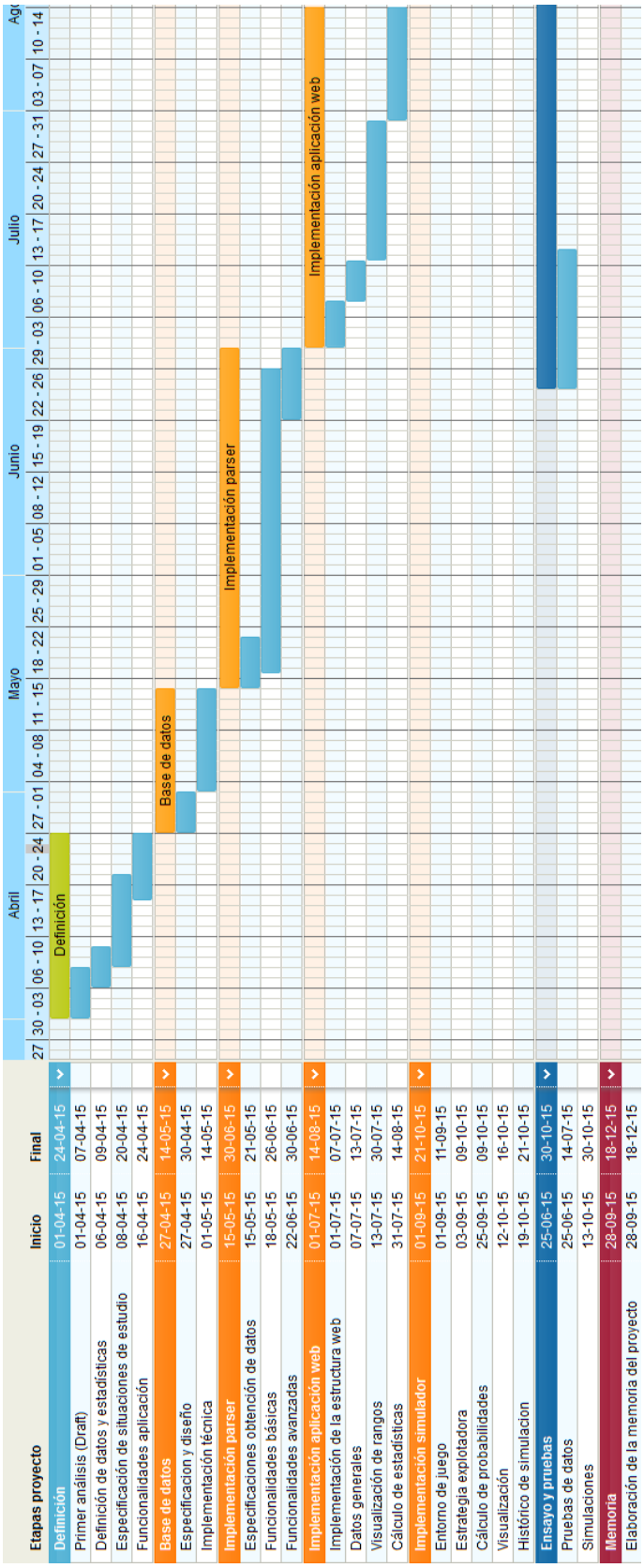
Tarea	Rol	Tiempo (h)	Coste/h	Coste (EUR)
Especificación	Analista	100	30	3000
Implementación Base de datos	Analista BD	15	35	525
Implementación Parser	Programador	240	25	6000
Implementación Aplicación web	Programador	210	25	5250
Implementación simulador	Programador	200	25	5000
Ensayos y pruebas	Tester	50	15	750
Total		**815**		**20525**

Producto	Coste (USD)
Licencia Hold'em Manager	100
Historiales de manos	32
Total	**132**

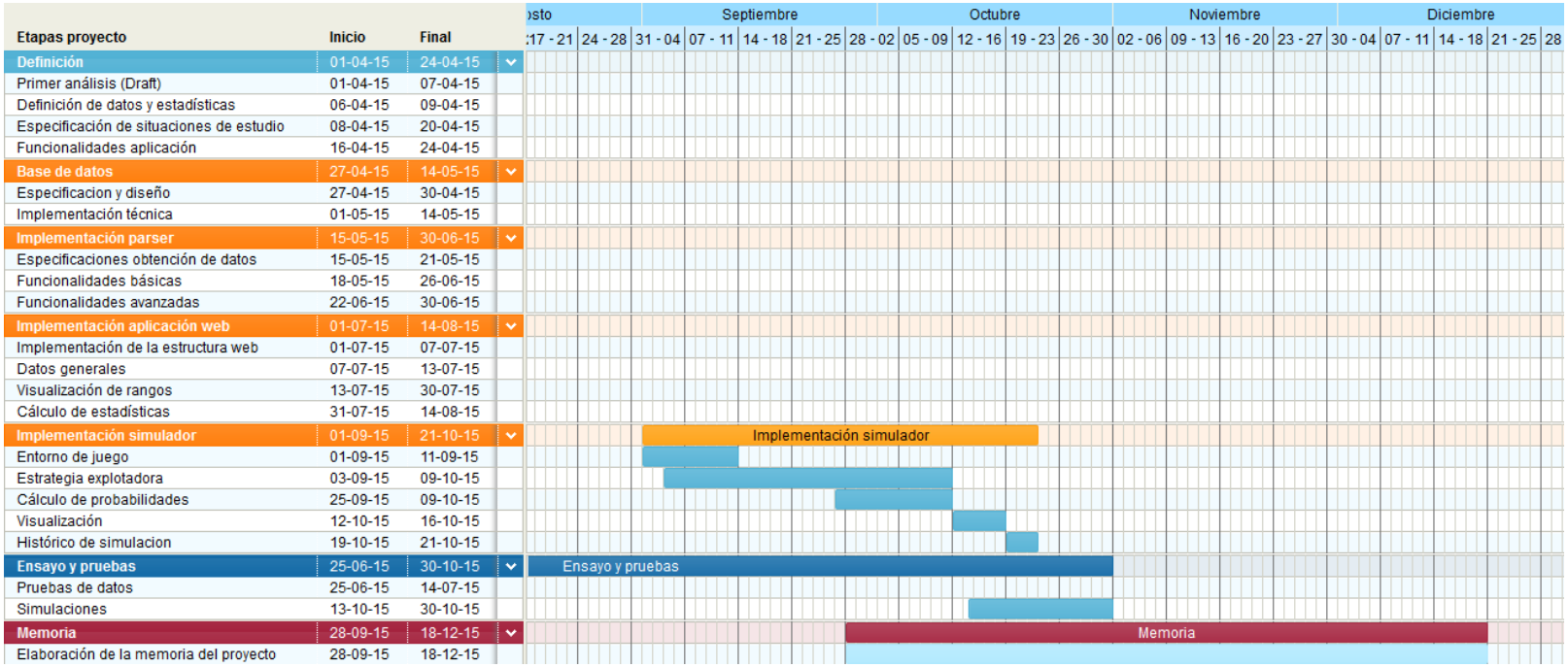
El coste en dólares representados en euros (a tipo actual) es de 120 EUR.

Por lo tanto, el coste total del proyecto son 20645 EUR.

Diagrama de Gannt



Etapas proyecto	Inicio	Final
Definición	01-04-15	24-04-15
Primer análisis (Draft)	01-04-15	07-04-15
Definición de datos y estadísticas	06-04-15	09-04-15
Especificación de situaciones de estudio	08-04-15	20-04-15
Funcionalidades aplicación	16-04-15	24-04-15
Base de datos	27-04-15	14-05-15
Especificacion y diseño	27-04-15	30-04-15
Implementación técnica	01-05-15	14-05-15
Implementación parser	15-05-15	30-06-15
Especificaciones obtención de datos	15-05-15	21-05-15
Funcionalidades básicas	18-05-15	26-06-15
Funcionalidades avanzadas	22-06-15	30-06-15
Implementación aplicación web	01-07-15	14-08-15
Implementación de la estructura web	01-07-15	07-07-15
Datos generales	07-07-15	13-07-15
Visualización de rangos	13-07-15	30-07-15
Cálculo de estadísticas	31-07-15	14-08-15
Implementación simulador	01-09-15	21-10-15
Entorno de juego	01-09-15	11-09-15
Estrategia explotadora	03-09-15	09-10-15
Cálculo de probabilidades	25-09-15	09-10-15
Visualización	12-10-15	16-10-15
Histórico de simulacion	19-10-15	21-10-15
Ensayo y pruebas	25-06-15	30-10-15
Pruebas de datos	25-06-15	14-07-15
Simulaciones	13-10-15	30-10-15
Memoria	28-09-15	18-12-15
Elaboración de la memoria del proyecto	28-09-15	18-12-15


Etapas proyecto
Inicio
Final
osto
Septiembre
Octubre
Noviembre
Diciembre
:17 - 21
24 - 28
31 - 04
07 - 11
14 - 18
21 - 25
28 - 02
05 - 09
12 - 16
19 - 23
26 - 30
02 - 06
09 - 13
16 - 20
23 - 27
30 - 04
07 - 11
14 - 18
21 - 25
28
Definición 01-04-15 24-04-15
Primer análisis (Draft) 01-04-15 07-04-15
Definición de datos y estadísticas 06-04-15 09-04-15
Especificación de situaciones de estudio 08-04-15 20-04-15
Funcionalidades aplicación 16-04-15 24-04-15
Base de datos 27-04-15 14-05-15
Especificacion y diseño 27-04-15 30-04-15
Implementación técnica 01-05-15 14-05-15
Implementación parser 15-05-15 30-06-15
Especificaciones obtención de datos 15-05-15 21-05-15
Funcionalidades básicas 18-05-15 26-06-15
Funcionalidades avanzadas 22-06-15 30-06-15
Implementación aplicación web 01-07-15 14-08-15
Implementación de la estructura web 01-07-15 07-07-15
Datos generales 07-07-15 13-07-15
Visualización de rangos 13-07-15 30-07-15
Cálculo de estadísticas 31-07-15 14-08-15
Implementación simulador 01-09-15 21-10-15
Implementación simulador
Entorno de juego 01-09-15 11-09-15
Estrategia explotadora 03-09-15 09-10-15
Cálculo de probabilidades 25-09-15 09-10-15
Visualización 12-10-15 16-10-15
Histórico de simulacion 19-10-15 21-10-15
Ensayo y pruebas 25-06-15 30-10-15
Ensayo y pruebas
Pruebas de datos 25-06-15 14-07-15
Simulaciones 13-10-15 30-10-15
Memoria 28-09-15 18-12-15
Memoria
Elaboración de la memoria del proyecto 28-09-15 18-12-15

Referencias

Información general

- Wikipedia (https://en.wikipedia.org/wiki/Texas_hold_'em)
- Foros de 2+2 (http://forumserver.twoplustwo.com/)
- Educapoker (http://www.educapoker.com)

Información de software

- Hold'em Manager (http://www.holdemmanager.com/)
- EquiLab (http://www.pokerstrategy.com/poker-tools/equilab-holdem/)
- ProPokerTools (http://propokertools.com/simulations)
- Pokenum (http://download.gna.org/pokersource/)
- PostgreSQL (https://www.postgresql.org/)

Literatura

- The Mathematics of Poker (Bill Chen, Jerrod Ankenman)