Grado en Matemática Computacional

Proyecto Final de Grado

Aplicaciones algorítmicas del lema de Johnson-Lindenstrauss

Autor:
Javier Calahorra Tovar

Supervisor:
Carlos Pellicer Duato
Tutores académicos:
Alejandro Miralles Montolío
Jorge Galindo Pastor

Fecha de lectura: 26 de septiembre de 2016
Curso académico 2015/2016

A mi hermano.
A mis padres.
A mi pareja.

Agradecimientos

A mis compañeros, que me apoyaron durante estos últimos años. Por su compañía, las horas de estudio y los momentos compartidos en la pecera.

A The Netwizzy Company S.L., por la oportunidad que me ha brindado y haberme hecho sentir como uno más. A mis compañeros de almuerzo, por amenizar las mañanas y por las conversaciones escatológicas. Especialmente a Carlos Pellicer y Cristóbal Luque, por su continuo apoyo.

A todas las personas que, de una forma u otra, me han ayudado durante estos 4 años en la *Universitat Jaume I*. A todos mis profesores. De forma especial, a Alejandro Miralles Montolío y Jorge Galindo Pastor que, como directores de este Trabajo de Final de grado, me han guiado, aconsejado, apoyado y corregido con un afecto y una entrega que han sobrepasado, con mucho, todas las expectativas.

A mi familia. Por ellos soy lo que soy. A mis tíos y primos. A mi abuela Tere y a mi abuela María, con todo mi amor y a quienes tengo siempre en mi pensamiento. A Amparo e Irene, por haberos convertido en una parte tan fundamental de mi vida. De forma especial, a mi hermano. Por muy lejos que estés siempre tendrás mi cariño y mi amor. Sé que siempre te sentiré cerca, que tus consejos seguirán llegando a mis oídos. Son los recuerdos y las ganas de estar contigo lo que, sin duda, hará que volvamos a vivir cerca. Gracias compañero. Amigo. Hermano.

A mis padres, por su apoyo, consejos, comprensión y ayuda en los momentos difíciles. Me habéis dado todo lo que soy como persona, mis valores, mis principios, mi carácter y el coraje para sobreponerme, siempre. Sobre todo, gracias por vuestra fe. Gracias por creer en mí cuando no era sencillo. Gracias por creer en mí cuando nadie más lo hacía. Especialmente a tí, mamá. Gracias por ser como eres. Gracias por escucharme, por ser tan comprensiva, por haber tenido siempre tiempo para hablar conmigo. No sabes lo importante que eres para mí. Especialmente a tí, papá. Tan distintos pero tan iguales. Cuánto amor con tan pocas palabras. Gracias por entenderme, aún cuando no era fácil.

A Carina, a quien amo con locura. Por su esfuerzo y sacrificio. Por creer en mí y ser fuente de motivación, inspiración y luz para poder superarme cada día más y así poder luchar para que la vida nos depare un futuro mejor. Por nuestro proyecto de vida juntos.

Resumen

Este trabajo está estructurado en seis capítulos. En primer lugar, en el *Capítulo 1* desarrollaremos los conceptos previos que nos permitirán entender la teoría que se expondrá posteriormente. En el *Capítulo 2* enunciaremos y demostraremos el lema de *Johnson-Lindestrauss* ilustrando dicho lema con varios ejemplos teóricos. Las aplicaciones prácticas las desarrollaremos en el *Capítulo 3*: mostraremos la eficiencia computacional que proporciona el lema en algunos problemas clásicos como *el vecino más próximo* o el cálculo de la distancia entre puntos en espacios euclídeos de dimensión alta. En el *Capítulo 4* describiremos el trabajo realizado durante la estancia en prácticas en la empresa *The Netwizzy Company S.L.*. Detallaremos, de forma más precisa, las herramientas empleadas y la metodología. Por último, en el *Capítulo 5* y *Capítulo 6* expondremos los resultados obtenidos durante la estancia en *The Netwizzy Company S.L.* y las conclusiones de este Trabajo Final de Grado, respectivamente.

Abstract

This thesis is structured in six chapters. Firstly, in *Chapter 1* we will develop the preliminary concepts that will allow us to understand the theory which will be exposed afterwards. In *Chapter 2* we will enunciate and demonstrate the *Johnson-Lindenstrauss's* lemma, illustrating it with a few theoretical examples. Practical applications will be developed during *Chapter 3*: we will show the computational efficiency that the lemma provides in some classical problems such as *the nearest neighbor* or calculating the distance between points in high-dimensional Euclidean spaces. During *Chapter 4* we will describe our work during our internship at the company *The Netwizzy Company S.L.*. We will detail the tools used and the methodology in a precise way. Finally, in *Chapter 5* and *Chapter 6* we will discuss respectively the results during the stay in *The Netwizzy Company S.L.* and the conclusions of this Final Project.

Palabras clave

Lema de Johnson-Lindenstrauss, inmersión, The Netwizzy Company, Our.com.

Keywords

Johnson-Lindenstrauss Lemma, embedding, The Netwizzy Company, Our.com.

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

Un espacio métrico (X, d) sobre un conjunto X de n elementos puede expresarse como una matriz $n \times n$ de números reales (por la simetría de d y teniendo en cuenta que d(x,x)=0, podríamos trabajar con una matriz triangular con ceros en la diagonal, esto es, $\binom{n}{2}$ números reales). Por tanto, podemos expresar un espacio métrico como una matriz de distancias o de disimiltud entre los elementos de dicho espacio.

El siguiente ejemplo ilustra lo anterior. Sea $l_2(m)$ el espacio euclídeo m-dimensional. Sea $X = \{x_1, x_2, \ldots, x_n\}$ un conjunto de tuplas de $l_2(m)$ que contienen m variables observadas sobre n individuos que juegan en *Our.com*, empresa en la que realicé la estancia en prácticas y que desarrolla su labor en el sector de los juegos online. Entonces, para cada par de individuos de X podemos obtener su disimilitud, esto es, cuánto son de diferentes entre ellos. La disimilitud puede ser calculada atendiendo a una o varias variables: edad, nacionalidad, sexo, cantidad de dinero jugado,... Es difícil intuir estructuras o modelos en largas tablas de números; además, generalmente no es viable computacionalmente tratar dichos datos. Por todo ello, puede ser interesante tratar de sumergir el espacio $l_2(m)$ en otro espacio euclídeo de menor dimensión con el que podamos trabajar más fácilmente.

Supongamos que pudiéramos asignar a cada $x \in X$ un punto $f(x)$ en el plano de tal forma que $d(x, y)$ fuera igual a la distancia euclídea de $f(x)$ a $f(y)$. Dicha representación nos permitiría entender mejor la estructura del espacio métrico: puntos aislados, clusters,... Otra ventaja de dicho espacio es que quedaría unívocamente representado con $2n$ números reales en lugar de los $\binom{n}{2}$ números de la matriz de disimilitud. Todo esto nos permitiría, para valores de n grandes, aplicar algoritmos geométricos eficientes que no serían computacionalmente viables para espacios métricos arbitrarios. Sin embargo, es fácil intuir que, en general, no podremos sumergir un espacio arbitrario en dicho espacio métrico.

En los ejemplos anteriores hemos considerado sólo inmersiones isométricas, i.e., $f : (X, d) \rightarrow (Y, \sigma)$ tal que $\sigma(f(x), f(y)) = d(x, y)$. Sin embargo, existen aplicaciones en las que no es necesario que se preserven las distancias exactamente, es decir, sólo necesitamos obtener una apro-

ximación suficientemente buena. En lo que sigue, nos centraremos en este tipo de aplicaciones lo que nos llevará a desarrollar, de forma exhaustiva, el lema de *Johnson-Lindenstrauss*.

El lema de *Johnson-Lindenstrauss* establece que, dado un conjunto discreto perteneciente a un espacio vectorial de dimensión alta es posible sumergir dicho conjunto en un espacio euclídeo cuya dimensión es menor de tal forma que las distancias entre los elementos se -casi- preserven.

1.2. Conceptos previos

1.2.1. Espacios métricos

Definición 1.1. *Un* ***espacio métrico*** *es un par* (X, d) *compuesto de un conjunto* X *y una función distancia* $d : X \times X \rightarrow \mathbb{R}$ *que verifica:*

1. $d(x,y) \geq 0$, *para todo* $x,y \in X$;
2. $d(x,y) = 0 \leftrightarrow x = y$, *para todo* $x,y \in X$;
3. $d(x,y) = d(y,x)$, *para todo* $x,y \in X$;
4. $d(x,z) \leq d(x,y) + d(y,z)$, *para todo* $x,y,z \in X$.

Ejemplo 1.1. *El par* $(\mathbb{R}^n, d_D)$ *donde* d_D *verifica que:*

$$d_D(x,\ y) \ = \ \begin{cases} 0, & \text{si } x \ = \ y \\ 1, & \text{si } x \ \neq \ y \end{cases}$$

para todo $x,\ y \in \mathbb{R}^n$ *es un espacio métrico. A la distancia* d_D *la denominaremos distancia discreta.*

Ejemplo 1.2. *El conjunto* $C\,[a,b]$ *de todas las funciones reales continuas definidas en el intervalo* $[a,b]$ *con la distancia*

$$\rho(f,\ g) \ = \ \max_{a \leq t \leq b} \ |f(t) - g(t)|$$

también forma un espacio métrico.

Ejemplo 1.3. *En lo que sigue, denotaremos mediante* ℓ_2 *al espacio métrico cuyos elementos son todas las sucesiones de números reales que verifican la condición:*

$$\sum_{k=1}^{\infty} \ x_k^2 \ < \ \infty$$

y cuya distancia viene dada por:

$$\rho(x,\ y) \ = \ \sqrt{\sum_{k=1}^{\infty} \ (y_k - x_k)^2}$$

1.2.2. Espacios normados

Denotaremos por $\mathbb{K}$ al cuerpo de escalares de los espacios vectoriales que aparecen a continuación. Asumiremos que este cuerpo es siempre $\mathbb{R}$ o $\mathbb{C}$.

Definición 1.2. *Sea X un espacio vectorial sobre un cuerpo $\mathbb{K}$. Una* ***norma*** *en X es una aplicación $\|\cdot\| : X \to \mathbb{R}$ que para cada $x, y \in X$ y cada $\alpha \in \mathbb{K}$ verifica:*

1. $\|x\| \geq 0$;
2. $\|x\| = 0 \leftrightarrow x = 0$;
3. $\|\alpha x\| = |\alpha|\|x\|$;
4. $\|x + y\| \leq \|x\| + \|y\|$.

El par $(X, \|\cdot\|)$ se denomina ***espacio normado****.*

Dados $x, y \in \mathbb{R}^n$, el segmento que va de x a y se obtiene trasladando el que va del origen a $y - x$, luego la longitud del vector $y - x$ nos da la distancia de x a y. Por tanto, cada norma que usemos en $\mathbb{R}^n$ nos va a dar una posible definición de la distancia entre dos puntos de $\mathbb{R}^n$. De hecho, vamos a hacer lo mismo en cualquier espacio normado.

Si X es un espacio normado, se define la función distancia entre dos puntos de X por

$$d(x, y) = \|y - x\| \qquad \text{para todo } x, y \in X.$$

Obtenemos así una función distancia $d : X \times X \to \mathbb{R}$, a partir de la cual podemos recuperar la norma X inducida por dicha distancia ya que

$$\|x\| = d(0, x) \qquad \text{para todo } x \in X.$$

Observación. Todo espacio normado es, a su vez, un espacio métrico. De esta forma, todas las nociones de espacios métricos están definidas también para espacios normados. Notar que el recíproco no es cierto. Consideremos el par (X, d_D). Si d_D fuese inducido por una norma, $\|2x\| = d_D(2x, 0) = d_D(x, 0) = \|x\|$ lo que contradice el *axioma 3* de la definición de norma.

En general, si tenemos definida una métrica nos podemos preguntar cuándo esta métrica está inducida por alguna norma. El siguiente teorema resuelve esta cuestión.

Teorema 1.1. *Sea (X, d) un espacio métrico. Entonces d está inducida por una norma en X si y sólo si para cada $x, y, z \in X$ y cada $\alpha \in \mathbb{K}$ se verifica:*

1. $d(x+z, y+z) = d(x, y)$;
2. $d(\alpha x, \alpha y) = |\alpha| d(x, y)$.

Demostración. Supongamos que d está inducido por la norma $\|\cdot\|$. Entonces si $x, y, z \in X$ y $\alpha \in \mathbb{K}$ tenemos que:

- d(x+z, y+z) = $\|(x+z)-(y+z)\| = \|x-y\|$ = d(x, y);
- d(αx, αy) = $\|\alpha x,\ \alpha y\| = |\alpha|\|x-y\| = |\alpha|$d(x, y).

Supongamos, ahora, que la métrica d verifica las condiciones *1* y *2* del enunciado y definimos $\|x\|$ = d(x, 0), para cada x ∈ X. Demostraremos que $\|\cdot\|$ es norma en X. Está claro que dicha norma verifica las propiedades *1* y *2* de la definición de norma. Además, si x,y ∈ X y $\alpha \in \mathbb{K}$ entonces tenemos que:

- $\|\alpha x\|$ = d(αx, 0) = d(αx, α0) = $|\alpha|$d(x, 0) = $|\alpha|\|x\|$;
- $\|x+y\|$ = d(x+y, 0) = d(x, -y) ≤ d(x, 0) + d(0, -y) = $\|x\| + \|y\|$,

como queríamos demostrar. □

No es difícil demostrar que los siguientes espacios son normados[1]:

Ejemplo 1.4. *ℓ_p ($p \geq 1$) es el espacio de las sucesiones $x = (x_n)_{n=1}^{\infty}$, $x_n \in \mathbb{K}$ que satisfacen la condición $\sum_{n=1}^{\infty}|x_n|^p < \infty$. Se define:*

$$\|x\|_p \ = \ \left(\sum_{i=1}^{\infty}|x_i|^p\right)^{\frac{1}{p}}$$

y se verifica que $\|\cdot\|_p$ es norma en ℓ_p .

Ejemplo 1.5. *El par $(\mathbb{R}^n, \|\cdot\|_p)$ donde $\|\cdot\|_p$ se define como:*

$$\|x\|_p \ = \ \left(\sum_{i=1}^{n}|x_i|^p\right)^{\frac{1}{p}},$$

es un espacio normado y lo denotaremos mediante $\ell_p(n)$. Si $p = 2$, $\|\cdot\|_2$ se denomina norma euclídea.

Ejemplo 1.6. *ℓ_∞ es el espacio de las sucesiones acotadas $x = (x_n)_{n=1}^{\infty}$ con la norma dada por*

$$\|x\|_\infty = sup\,\{|x(n)| : n \in \mathbb{N}\} \qquad (x \in \ell_\infty)$$

Ejemplo 1.7. *Denotemos por c al espacio de las sucesiones escalares convergentes. Es claro que $c \subset l_\infty$ y por tanto c, con la norma del supremo, es un espacio normado.*

Ejemplo 1.8. *Sea c_0 el espacio de las sucesiones convergentes a cero. $(c_0, \|\cdot\|_\infty)$ es un espacio normado. Además, c_0 es un subespacio de ℓ_∞.*

Ejemplo 1.9. *Sea*

$$c_{00} \;=\; \{x \in c_0 \;:\; existe\ n_0 \in \mathbb{N}\ tal\ que\ x(n) = 0,\ n \geq n_0\}$$

Se tiene que c_{00} es un subespacio de c_0, que es conocido como el espacio de las sucesiones eventualmente nulas y c_{00} puede ser identificado como el espacio vectorial de los polinomios en una indeterminada. Además, dicho espacio es normado dotado con la norma del supremo.

Observación: Hemos visto que c_{00}, c_0 y c son subespacios de ℓ_∞:

$$c_{00} \subseteq c_0 \subseteq c \subseteq \ell_\infty.$$

Por este motivo, cuando heredan la norma de ℓ_∞ *-norma del supremo-* son espacios normados.

1.2.3. Normas equivalentes y distorsión

Todo espacio métrico (X, d) tiene asociado un espacio topológico (X, τ_d), donde τ_d es la familia de los conjuntos abiertos de X para la distancia d. Estos abiertos constituyen una topología que llamamos topología τ_d asociada a la métrica d [9].

Prestamos ahora atención a la posibilidad de que dos distancias diferentes en un mismo conjunto generen la misma topología. En tal caso tendremos dos espacios métricos distintos, pero con idénticas propiedades topológicas.

Se dice que dos distancias en un conjunto X son equivalentes cuando generan la misma topología. Por tanto, diremos que dos normas en un espacio vectorial X son equivalentes cuando lo sean las distancias asociadas, esto es, cuando las topologías de ambas normas coincidan. La equivalencia entre dos normas se caracteriza de forma muy sencilla, como vamos a ver.

Definición 1.3. *Para dos normas $\|\cdot\|$ y $\|\cdot\|'$ definidas en un mismo espacio vectorial X, las siguientes afirmaciones son equivalentes:*

1. *Existe una constante $\rho \in \mathbb{R}^+$ tal que $\|\cdot\|' \leq \rho\|\cdot\|$ para todo $x \in X$;*
2. *La topología de la norma $\|\cdot\|'$ está incluída en la de $\|\cdot\|$;*

Demostración. Para la demostración, dados x ∈ X y r ∈ $\mathbb{R}^+$, denotamos por B(x, r) y B'(x, r) a las bolas abiertas de centro x y radio r para las normas $\|\cdot\|$ y $\|\cdot\|'$, respectivamente.

(1) ⇒ (2). Si U es un conjunto abierto para la norma $\|\cdot\|'$, para cada x ∈ U existe $\epsilon > 0$ tal que B'(x, ϵ) ⊂ U. De 1 deducimos entonces que $B(x,\epsilon/\rho) \subset B'(x,\epsilon) \subset$ U, luego U es abierto para la norma $\|\cdot\|$, como queríamos.

(2) ⇒ (1). Como $B'(0, 1)$ es abierto para $\|\cdot\|'$, también lo será para $\|\cdot\|$, luego tiene que existir $\delta > 0$ tal que $B(0,\delta) \subset B'(0,\delta)$. Tomando $\rho = 1/\delta > 0$ conseguimos la desigualdad buscada. En efecto, si x ∈ X verficase que $\|x\|' > \rho\|x\|$, tomando y = x/$\|x\|'$ tendríamos:

$$\|y\| \;=\; \frac{\|x\|}{\|x\|'} \;<\; \frac{1}{\rho} = \delta$$

de donde $\|y\|' < 1$, que es una contradicción, puesto que claramente $\|y\|' = 1$. Por tanto, $\|x\|'$ $\leq \rho\|x\|$ para todo x $\in$ X, como queríamos. $\square$

Está claro ahora que dos normas serán equivalentes si y sólo si se verifica la desigualdad que aparece en la condición 1, junto con la que se obtendría intercambiando los papeles de ambas normas. Enlazando ambas desigualdades y dividiendo ambos miembros por la constante que en ellos aparezca, obtenemos un sencillo criterio para decidir si dos normas son equivalentes:

Proposición 1.1. *Dos normas $\|\cdot\|$ y $\|\cdot\|'$ en un espacio vectorial X son equivalentes si y sólo si existen $\lambda, \rho > 0$ tales que*

$$\lambda \|\cdot\| \ \leq \ \|\cdot\|' \ \leq \ \rho \|\cdot\| \qquad \text{para todo } x \ \in \ X.$$

Definición 1.4. *Sean $(X,\ \rho)$ e $(Y,\ \sigma)$ espacios métricos. Una función $f : (X, \rho) \ \rightarrow \ (Y, \sigma)$ se llama **D-inmersión**, donde $D \geq 1$ es un número real, si existe un número $r > 0$ tal que*

$$r \cdot \rho(x, y) \ \leq \ \sigma(f(x), f(y)) \ \leq \ D \cdot r \cdot \rho(x, y)$$

*para todo $x, y \in X$. El ínfimo de los números tales que f es un D-embedding se llama **distorsión de f**.*

El objetivo de este trabajo es D-sumergir espacios euclídeos de dimensión muy alta en otros de dimensión más baja. Es por otra parte conocido que todo espacio métrico de n puntos puede D-sumergirse en un espacio euclídeo con una cierta distorsión. Este es el Teorema de Jean Bourgain [4].

Teorema 1.2. *Todo espacio métrico discreto de n puntos (X, d) puede ser sumergido en $\ell_2(n)$ con una distorsión de orden O(log n), como máximo.*

1.2.4. Espacios de Banach

Definición 1.5. *Sea $\{x_n\}_{n\in\mathbb{N}}$ una sucesión. Diremos que $\{x_n\}_{n\in\mathbb{N}}$ es de Cauchy, si para todo número real $\epsilon > 0$ existe un entero positivo n_0 tal que para todos los números naturales $m, n > n_0$ $|x_m - x_n| < \epsilon$.*

Proposición 1.2. *Toda sucesión convergente es una sucesión de Cauchy.*

Definición 1.6. *Un espacio normado $(X, \|\cdot\|)$ se dice **espacio de Banach** si es completo para la topología de la norma, esto es, si toda sucesión de Cauchy es convergente en X [6].*

Proposición 1.3. *Si $(X, \|\cdot\|)$ es un espacio de Banach y $Z \subset X$ un subespacio cerrado entonces $(Z, \|\cdot\|)$ es también un espacio de Banach.*

Ejemplo 1.10. $(\mathbb{R}, |\cdot|)$ *es un espacio de Banach.*

Ejemplo 1.11. *El espacio $(\mathbb{Q}, |\cdot|)$ no es un espacio de Banach ya que dicho espacio no es completo.*

Demostración. Vamos a construir una sucesión de Cauchy de números racionales que no es convergente en $\mathbb{Q}$. Consideremos para cada $n \in \mathbb{N}$ la sucesión

$$s_n \;=\; 1 \;+\; \frac{1}{1!} \;+\; \cdots \;+\; \frac{1}{n!}.$$

En primer lugar, averiguaremos el carácter de la serie $s_n = \sum_{k=0}^{\infty} \frac{1}{k!}$. Aplicando el criterio del cociente de D'Alembert podemos ver que $\lim\limits_{k\to\infty} \dfrac{s_{k+1}}{s_k} < 1$ y, por tanto, la serie converge. Además, al límite de s_n se le llama numero e. Por la *Propiedad 1.2.*, como s_n es convergente en $\mathbb{R}$ es de Cauchy.

Por último, vamos a demostrar que $e \notin \mathbb{Q}$. Supongamos inicialmente que e es un número racional de la siguiente forma:

$$e \;=\; \frac{a}{b}.$$

Definamos x, tal que

$$x \;=\; b!\left(e - \sum_{n=0}^{b} \frac{1}{n!}\right).$$

Si sustituimos el número e en la definición de x obtenemos que:

$$x \;=\; b!\left(\frac{a}{b} - \sum_{n=0}^{b} \frac{1}{n!}\right) \;=\; a\,(b-1)! - \sum_{n=0}^{b} \frac{b!}{n!}.$$

El término $a\,(b-1)!$ es un número entero y, además, los términos de $\sum_{n=0}^{b} \frac{b!}{n!}$ son, también, números enteros ya que $n \leq b$ para cada término. Por lo tanto, x es un entero.

Ahora probaremos que $0 < x < 1$. Veamos que $x > 0$:

$$x = b!\left(e - \sum_{n=0}^{b} \frac{1}{n!}\right) = b!\left(\sum_{n=0}^{\infty} \frac{1}{n!} - \sum_{n=0}^{b} \frac{1}{n!}\right) = \sum_{n=b+1}^{\infty} \frac{b!}{n!} > 0.$$

Veamos que $x < 1$. Para todos los términos del sumatorio con $n \geq b+1$ tenemos una cota superior ya que:

$$\frac{b!}{n!} = \frac{1}{(b+1)(b+2)\cdots(b+(n-b))} \leq \frac{1}{(b+1)^{n-b}}.$$

Cambiando el índice del sumatorio a $k = nb$ y aplicando la fórmula de la serie geométrica infinita, obtenemos que:

$$x = \sum_{n=b+1}^{\infty} \frac{b!}{n!} < \sum_{k=1}^{\infty} \frac{1}{(b+1)^k} = \frac{1}{b+1}\left(\frac{1}{1-\frac{1}{b+1}}\right) = \frac{1}{b} \leq 1.$$

Como no hay ningún número entero comprendido entre 0 y 1 hemos llegado a una contradicción, y por lo tanto, e debe ser irracional. Concluimos que $\mathbb{Q}$ no es completo. □

Ejemplo 1.12. *El par* $(C[a,b], \|\cdot\|_\infty)$ *donde*

$$\|f\|_\infty = sup\{|f(x)| : x \in [a,b]\}$$

es un espacio de Banach.

Ejemplo 1.13. $(l_\infty, \|\cdot\|_\infty)$ *es un espacio de Banach.*

Ejemplo 1.14. $(c_0, \|\cdot\|_\infty)$ *es un espacio de Banach ya que es un subespacio cerrado de* l_∞.

Ejemplo 1.15. c_{00} *no es completo.*

Demostración. Consideremos la sucesión $x_1 = (1,0,0,0,\ldots)$, $x_2 = (1,1/2,0,0,0,\ldots)$, $x_3 = (1,1/2,1/3,0,0,0,\ldots)$, ... La sucesión $\{x_n\}$ es convergente y, por la *Propiedad 1.2.*, $\{x_n\}$ es de Cauchy. Veamos cuál es el límite de $\{x_n\}$:

Sea k_0 un índice cualquiera y supongamos que $a_{k_0} \neq 1/k_0$. Entonces, siempre que $n > k_0$ tenemos que:

$$\|x_n - a\|_\infty = sup_j \left\| \frac{1}{j} - a_j \right\| \geq \left\| a_{k_0} - \frac{1}{k_0} \right\| \neq 0$$

lo cual es una contradicción.

Por tanto, $a_k = 1/k$ para todo k pero $(1,1/2,1/3,\ldots,1/k,1/k+1,\ldots) \notin c_{00}$. □

1.2.5. Espacios de Hilbert

Definición 1.7. *Un espacio con* ***producto interior*** *o* ***pre-Hilbert*** *es un espacio vectorial* X *en el que se define una aplicación* $\langle\cdot,\cdot\rangle : X \times X \to \mathbb{K}$ *con las siguientes propiedades:*

1. *Aditiva:* $\langle x+y, z\rangle = \langle x,z\rangle + \langle y,z\rangle$;
2. *Homogénea:* $\langle \alpha x, y\rangle = \alpha\langle x,y\rangle$;
3. *Hermítica:* $\langle x,y\rangle = \langle \overline{y,x}\rangle$;
4. *Definida positiva:* $\langle x,x\rangle \geq 0$ *y* $\langle x,x\rangle = 0 \leftrightarrow x = 0$.

Los axiomas anteriores fueron establecidos por von Neumann en 1930 en sus trabajos sobre fundamentos matemáticos de la Mecánica Cuántica. En su definición se incluía también la separabilidad del espacio, axioma posteriormente eliminado cuando Lowing, Rellig y Riesz mostraron que en la práctica era innecesaria dicha restricción [5].

Todo espacio pre-Hilbert es en particular normado, donde la norma asociada se define como $\|x\| = \sqrt{\langle x,x\rangle}$, y por tanto es también métrico, con la distancia $d(x,y) = \|x-y\| = \sqrt{\langle x-y, x-y\rangle}$. Esto motiva la siguiente definición:

Definición 1.8. *Un **espacio de Hilbert** es un espacio pre-Hilbert completo para la norma asociada.*

Observación: Los espacios de Hilbert son, obviamente, espacios de Banach.

Ejemplo 1.16. *ℓ_2 es un espacio de Hilbert (además, separable) con el producto $\langle x, y\rangle = \sum_{n=1}^{\infty} x_n\overline{y_n}$.*

Ejemplo 1.17. *ℓ_p, con $p \neq 2$ no es espacio de Hilbert.*

Capítulo 2

El lema de Johnson-Lindenstrauss

Johnson y Lindenstrauss demostraron que cualquier subconjunto discreto de n puntos de $\ell_2(d)$ puede ser sumergido en otro espacio euclídeo de dimensión $k = O(\log n/\epsilon^2)$ sin distorsionar la distancia entre cualquier par de puntos más allá de un factor $1+\epsilon$ con $0 < \epsilon < 1$.

En los últimos años, se han hallado numerosas aplicaciones del lema de Johnson-Lindenstrauss entre las que se incluyen implementaciones más eficientes, entre otros, del problema del vecino más próximo en espacios euclídeos de dimensión alta y reducción de la dimensión de bases de datos.

La prueba original de Johnson y Lindenstrauss es probabilística, mostrando que la proyección de un subconjunto discreto de n puntos en otro subespacio aleatorio de dimensión $k = O(\log\ n/\epsilon^2)$ no distorsiona más de $1+\epsilon$ la distancia entre puntos con probabilidad positiva. En la demostración que realizaremos a continuación utilizaremos técnicas probabilísticas para obtener el resultado.

La prueba de Johnson y Lindestraus fue simplificada años después por Frankl y Maehara [7]. Posteriormente, otros autores como Indyk y Motwani [8], Arriaga y Vempala [10] y Achlioptas [2] también han dado pruebas similares del teorema utilizando algoritmos aleatorios sencillos.

Lema de Johnson-Lindestraus 2.1. Dados $0 < \epsilon < 1$, $n \in \mathbb{N}$. Si $k \geq \frac{4}{\epsilon^2/2-\epsilon^3/3} \cdot \log n$ entonces para todo espacio normado $V = \{v_1, v_2, \ldots, v_n\} \subseteq (\mathbb{R}^d,\ \|\cdot\|_2)$ existe una aplicación lineal $f : \mathbb{R}^d \to \mathbb{R}^k$ tal que para todo $u, v \in V$

$$(1-\epsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\epsilon)\|u-v\|^2 .$$

Mostraremos que el cuadrado de un vector aleatorio se concentra en torno a su media cuando el vector se proyecta sobre un espacio k-dimensional cualquiera; en concreto, su longitud no se distorsionará más de $1+\epsilon$ con probabilidad $O(1/n^2)$ [11].

Por lo tanto, el objetivo es estimar la longitud de un vector unitario en $\mathbb{R}^d$ cuando se proyecta sobre un subespacio k-dimensional aleatorio. Sin embargo, esta longitud sigue la misma distribución que la longitud de un vector unitario aleatorio proyectado sobre un subespacio

k-dimensional fijado. Por simplicidad, tomaremos el espacio formado por las k primeras componentes de los vectores.

Sean $X_1, X_2, \ldots, X_d$ variables aleatorias, independientes con $X_i \sim N(0,1)$ para $i = 1, 2, \ldots, n$ y sea $Y = \frac{1}{\|X\|}(X_1, X_2, \ldots, X_d)$. Es fácil ver que Y es un punto elegido de manera aleatoria y uniforme de la superfície de la esfera d-dimensional S^{d-1}. Sea el vector $Z \in \mathbb{R}^k$ la proyección de Y dado por sus primeras k componentes y sea $L = \|Z\|^2$. Claramente, la longitud al cuadrado de Z es $\mu = E[L] = k/d$. El *Lema 2.2.* prueba que L también se concentra en torno a μ [11]. Veamos antes un ejemplo.

Ejemplo 2.1. *Sea* $P = (cos(t), sen(t)), t \in [0, 2\pi)$ *un punto cualquiera de* S^1. *Sea* $Z : S^1 \to [-1, 1]$ *la primera proyección* $Z(P) = cos(t)$ *y sea* $L = \|Z\|^2 = cos^2(t)$. *Entonces:*

$$Pr[L \geq \alpha] \;=\; Pr\left[cos^2(t) \geq \alpha\right] \;=\; Pr\left[cos(t) \geq \sqrt{\alpha}\right] \;=\; \frac{2}{\pi}\arcsin\sqrt{\alpha}$$

y, por tanto,

$$\begin{aligned} Pr[L \leq \alpha] &= 1 - \frac{2}{\pi}\arcsin\sqrt{\alpha} \\ Pr[L \geq \alpha] &= \frac{2}{\pi}\arcsin\sqrt{\alpha} \end{aligned}$$

Veamos si L se concentra en torno a $\frac{k}{d}$:

$$\begin{aligned} Pr[L \leq 0{,}45] &= 1 - \frac{2}{\pi}\arcsin\sqrt{0{,}45} = 0{,}4681157. \\ Pr[L \leq 0{,}375] &= 1 - \frac{2}{\pi}\arcsin\sqrt{0{,}375} = 0{,}4195694. \\ Pr[L \leq 0{,}25] &= 1 - \frac{2}{\pi}\arcsin\sqrt{0{,}25} = 0{,}3333333. \end{aligned}$$

$$\begin{aligned} Pr[L \geq 0{,}75] &= \frac{2}{\pi}\arcsin\sqrt{0{,}75} = 0{,}3333333. \\ Pr[L \geq 0{,}625] &= \frac{2}{\pi}\arcsin\sqrt{0{,}625} = 0{,}4195694. \\ Pr[L \geq 0{,}55] &= \frac{2}{\pi}\arcsin\sqrt{0{,}55} = 0{,}4681157. \end{aligned}$$

A pesar de estar trabajando con espacios de dimensión muy baja, acabamos de corroborar que L se concentra entorno a $\frac{k}{d} = \frac{1}{2}$. Notar que, en el caso anterior, el cálculo de las probabilidades es exacto.

Lema 2.2. Sea $k < d$. Entonces:

1. Si $\beta < 1$, entonces:

$$Pr\left[L \leq \frac{\beta k}{d}\right] \leq \beta^{k/2}\left(1 + \frac{(1-\beta)k}{(d-k)}\right)^{(d-k)/2} \leq e^{\left(\frac{k}{2}(1-\beta+\log\beta)\right)}.$$

2. Si $\beta > 1$, entonces:

$$Pr\left[L \geq \frac{\beta k}{d}\right] \leq \beta^{k/2}\left(1 + \frac{(1-\beta)k}{(d-k)}\right)^{(d-k)/2} \leq e^{\left(\frac{k}{2}(1-\beta+\log\beta)\right)}.$$

Ejemplo 2.2. *Veamos qué sucede cuando aplicamos el lema en el Ejemplo 2.1:*

- *Si $\beta < 1$, entonces:*

$$si\ \beta = 0{,}90\ \Rightarrow\ Pr\left[L \leq \frac{0{,}90 \cdot 1}{2}\right] = Pr\left[L \leq 0{,}45\right] \leq e^{\left(\frac{1}{2}(1-0{,}90+\log 0{,}90)\right)} = 0{,}9973233.$$
$$si\ \beta = 0{,}75\ \Rightarrow\ Pr\left[L \leq \frac{0{,}75 \cdot 1}{2}\right] = Pr\left[L \leq 0{,}375\right] \leq e^{\left(\frac{1}{2}(1-0{,}75+\log 0{,}75)\right)} = 0{,}9813353.$$
$$si\ \beta = 0{,}5\ \Rightarrow\ Pr\left[L \leq \frac{0{,}5 \cdot 1}{2}\right] = Pr\left[L \leq 0{,}25\right] \leq e^{\left(\frac{1}{2}(1-0{,}5+\log 0{,}5)\right)} = 0{,}9079431.$$

- *Si $\beta > 1$, entonces:*

$$si\ \beta = 1{,}50\ \Rightarrow\ Pr\left[L \geq \frac{1{,}50 \cdot 1}{2}\right] = Pr\left[L \geq 0{,}75\right] \leq e^{\left(\frac{1}{2}(1-1{,}50+\log 1{,}50)\right)} = 0{,}9538323.$$
$$si\ \beta = 1{,}25\ \Rightarrow\ Pr\left[L \geq \frac{1{,}25 \cdot 1}{2}\right] = Pr\left[L \geq 0{,}625\right] \leq e^{\left(\frac{1}{2}(1-1{,}25+\log 1{,}25)\right)} = 0{,}9866615.$$
$$si\ \beta = 1{,}1\ \Rightarrow\ Pr\left[L \geq \frac{1{,}1 \cdot 1}{2}\right] = Pr\left[L \geq 0{,}55\right] \leq e^{\left(\frac{1}{2}(1-1{,}1+\log 1{,}1)\right)} = 0{,}9976578.$$

Vemos como, al estar trabajando con espacios de dimensión tan baja, el lema no nos proporciona información relevante.

Ejemplo 2.3. *Sea $v = (x_1, x_2, x_3, \ldots, x_{10000})$ un vector aleatorio de $\mathbb{R}^{10000}$. Sea $Y : \mathbb{R}^{10000} \to S^{9999}$, definida como sigue:*

$$Y(v) = \frac{1}{\sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_{10000}^2}}(x_1, x_2, x_3, \ldots, x_{10000}).$$

$Y(v)$ es un punto de la superfície de la esfera 10000-dimensional S^{9999}. Sea $Z : S^{9999} \to \mathbb{R}^{2000}$,

$$Z\left(Y(v)\right) = \frac{1}{\sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_{10000}^2}}(x_1, x_2, x_3, \ldots, x_{2000})$$

la proyección de Y y $L = \|Z\|^2$. Entonces:

$$L\left(Z\left(Y(v)\right)\right) = \frac{x_1^2, x_2^2, x_3^2, \ldots, x_{2000}^2}{x_1^2 + x_2^2 + x_3^2 + \cdots + x_{10000}^2}.$$

Aplicando el lema,

- Si $\beta < 1$, entonces:

$$si\ \beta = 0{,}95\ \Rightarrow\ Pr\left[L \leq \frac{0{,}95 \cdot 2000}{10000}\right] \leq e^{\left(\frac{2000}{2}(1-0{,}95+\log 0{,}95)\right)} = 0{,}2743654.$$

$$si\ \beta = 0{,}925\ \Rightarrow\ Pr\left[L \leq \frac{0{,}925 \cdot 2000}{10000}\right] \leq e^{\left(\frac{2000}{2}(1-0{,}925+\log 0{,}925)\right)} = 0{,}0517391;$$

$$si\ \beta = 0{,}9\ \Rightarrow\ Pr\left[L \leq \frac{0{,}9 \cdot 2000}{10000}\right] \leq e^{\left(\frac{2000}{2}(1-0{,}9+\log 0{,}9)\right)} = 0{,}004698483;$$

$$si\ \beta = 0{,}875\ \Rightarrow\ Pr\left[L \leq \frac{0{,}875 \cdot 2000}{10000}\right] \leq e^{\left(\frac{2000}{2}(1-0{,}875+\log 0{,}875)\right)} = 0{,}0001971802;$$

- Si $\beta > 1$, entonces:

$$si\ \beta = 1{,}125\ \Rightarrow\ Pr\left[L \geq \frac{1{,}125 \cdot 2000}{10000}\right] \leq e^{\left(\frac{2000}{2}(1-1{,}125+\log 1{,}125)\right)} = 0{,}0007340273.$$

$$si\ \beta = 1{,}1\ \Rightarrow\ Pr\left[L \geq \frac{1{,}1 \cdot 2000}{10000}\right] \leq e^{\left(\frac{2000}{2}(1-1{,}1+\log 1{,}1)\right)} = 0{,}009188338;$$

$$si\ \beta = 1{,}075\ \Rightarrow\ Pr\left[L \geq \frac{1{,}075 \cdot 2000}{10000}\right] \leq e^{\left(\frac{2000}{2}(1-1{,}075+\log 1{,}075)\right)} = 0{,}06860853;$$

$$si\ \beta = 1{,}05\ \Rightarrow\ Pr\left[L \geq \frac{1{,}05 \cdot 2000}{10000}\right] \leq e^{\left(\frac{2000}{2}(1-1{,}05+\log 1{,}05)\right)} = 0{,}2982462;$$

Y vemos, claramente, como L se concentra entorno a $\frac{k}{d} = \frac{2000}{10000} = \frac{1}{5}$.

Lema de Johnson-Lindenstrauss. Si $d \leq k$ la demostración es trivial. Suponemos $d > k$. Sea S un subespacio vectorial aleatorio de $\mathbb{R}^d$ tal que $dim\,(S) = k$ y sea $P(v_i)$ la proyección de una d-tupla $v_i \in V$ -V queda perfectamente descrito en el enunciado del lema- sobre S. Entonces, tomando $L = \|P(v_i) - P(v_j)\|^2$ y $\mu = (k/d)\,\|v_i - v_j\|^2$ y aplicando el lema 2.2.1,

$$Pr\left[L \leq \beta \cdot \mu\right] = Pr\left[\|P(v_i) - P(v_j)\|^2 \leq \beta \cdot \frac{k}{d}\,\|v_i - v_j\|^2\right] = Pr\left[\frac{\|P(v_i) - P(v_j)\|^2}{\|v_i - v_j\|^2} \leq \beta \cdot \frac{k}{d}\right].$$

Teniendo en cuenta que la serie de Taylor de $\log(1-\epsilon) = -\epsilon - \frac{\epsilon^2}{2} - \frac{\epsilon^3}{3} + O(\epsilon^4)$ y tomando $\beta = 1 - \epsilon < 1$ tenemos que:

$$Pr\left[\frac{\|P(v_i) - P(v_j)\|^2}{\|v_i - v_j\|^2} \leq (1-\epsilon) \cdot \frac{k}{d}\right] \leq e^{\left(\frac{k}{2}(1-(1-\epsilon)+\log(1-\epsilon))\right)} = e^{\left(\frac{k}{2}(\epsilon+\log(1-\epsilon))\right)}$$

$$\leq e^{\left(\frac{k}{2}\left(\epsilon+\left(-\epsilon-\frac{\epsilon}{2}\right)\right)\right)} = e^{\left(-\frac{k\epsilon^2}{4}\right)} \leq e^{(2\log(n))=\frac{1}{n^2}}.$$

Análogamente y teniendo en cuenta que la serie de Taylor de $\log(1+\epsilon) = \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} + O(\epsilon^4)$, aplicamos el lema 2.2.2:

$$Pr\left[\frac{\|P(v_i) - P(v_j)\|^2}{\|v_i - v_j\|^2} \geq (1+\epsilon) \cdot \frac{k}{d}\right] \leq e^{\left(\frac{k}{2}(1-(1+\epsilon)+\log(1+\epsilon))\right)} = e^{\left(\frac{k}{2}(-\epsilon+\log(1+\epsilon))\right)}$$

$$\leq e^{\left(\frac{k}{2}\left(-\epsilon+\left(\epsilon-\frac{\epsilon}{2}+\frac{\epsilon}{3}\right)\right)\right)} = e^{\left(-\frac{k(\epsilon^2/2-\epsilon^3/3)}{2}\right)} \leq e^{(-2\log(n))} = \frac{1}{n^2}.$$

Es decir, $Pr\left[\frac{\|P(v_i)-P(v_j)\|^2}{\|v_i-v_j\|^2} \notin [(1-\epsilon)\cdot\frac{k}{d},(1+\epsilon)\cdot\frac{k}{d}]\right] \leq \frac{1}{n^2}+\frac{1}{n^2}=\frac{2}{n^2}$. Como en el espacio $V=\{v_1,v_2,\ldots,v_n\}$ hay $\binom{n}{2}$ parejas de d-tuplas, la probabilidad de que alguna de dichas parejas no esté contenida en el intervalo $[(1-\epsilon)\cdot\frac{k}{d},(1+\epsilon)\cdot\frac{k}{d}]$ es igual o menor que $\frac{2}{n^2}\binom{n}{2}=\frac{2}{n^2}\frac{n(n-1)}{2}=1-\frac{1}{n}$. Por tanto, las probabilidad de que las parejas (v_i,v_j) para todo $i,j=1,2,\ldots,n$ con $i\neq j$ estén contenidas en $[(1-\epsilon)\cdot\frac{k}{d},(1+\epsilon)\cdot\frac{k}{d}]$ es $1-\left(1-\frac{1}{n}\right)=\frac{1}{n}$.

Por todo ello, podemos concluir lo siguiente:

- Como la probabilidad de que todas las parejas de elementos de V pertenezcan a $[(1-\epsilon)\cdot\frac{k}{d},(1+\epsilon)\cdot\frac{k}{d}]$ es $\frac{1}{n}>0$, existe algún subespacio S verficando que $\frac{\|P(v_i)-P(v_j)\|^2}{\|v_i-v_j\|^2} \in [(1-\epsilon)\cdot\frac{k}{d},(1+\epsilon)\cdot\frac{k}{d}]$ para todo $i,j=1,2,\ldots,n$ con $i\neq j$. Basta tomar $f(v_i)=\sqrt{d/k}\cdot P(v_i)$ ya que:
$$(1-\epsilon)\cdot k/d \leq \frac{\|P(v_i)-P(v_j)\|^2}{\|v_i-v_j\|^2} \leq (1-\epsilon)\cdot k/d \;\leftrightarrow$$
$$\leftrightarrow\; (1-\epsilon)\|v_i-v_j\|^2 \leq \left\|\sqrt{d/k}\cdot P(v_i)-\sqrt{d/k}\cdot P(v_j)\right\|^2 = \|f(v_i)-f(v_j)\|^2 \leq (1+\epsilon)\|v_i-v_j\|^2.$$
- Como la probabilidad de que todas las parejas de elementos de V pertenezcan a $[(1-\epsilon)\cdot\frac{k}{d},(1+\epsilon)\cdot\frac{k}{d}]$ es $\frac{1}{n}>0$, en un tiempo polinomial deberíamos poder encontrar el subespacio S y la función correspondiente [11].

Comprobémoslo: Sea B_i el suceso *Hemos encontrado un subespacio válido en el intento i* y M_i el suceso *No hemos encontrado un subespacio válido en el intento i*. Notar que $B_i=\bar{M_i}$. Entonces, $P(M_1\cap M_2)\leq\left(1-\frac{1}{n}\right)^2$ y, en general, $P(M_1\cap M_2\cap\cdots\cap M_k)\leq\left(1-\frac{1}{n}\right)^k$ de donde se deduce que $P(B_1\cup B_2\cup\cdots\cup B_k)\geq 1-\left(1-\frac{1}{n}\right)^k$.

Fijemos $k=O\left(n^2\right)$ y sea $x_n=\left(1-\frac{1}{n}\right)^k$. Entonces:

$$\log(x_n)=O\left(n^2\right)\log\left(1-\frac{1}{n}\right)=O\left(n^2\right)\log\left(-\frac{1}{n}+\frac{1}{O\left(n^2\right)}-\ldots\right)=-n+O(1).$$

Por tanto, $x_n=e^{x_n}=e^{-n}\cdot e^{O(1)}$ y $\lim_{n\to\infty}(x_n)=0$. Si buscamos un subespacio válido $O\left(n^2\right)$ veces, la probabilidad de no encontrarlo en todos los casos es casi 0. □

Capítulo 3

Aplicaciones

En este capítulo veremos aplicaciones prácticas del *lema de Johnson-Lindenstrauss*. En primer lugar, comprobaremos si la aplicación del lema produce mejoras computacionales a la hora de calcular la distancia entre puntos en un espacio euclídeo de dimensión alta. En segundo lugar, comprobaremos de forma práctica como dicho lema produce mejoras computacionales en el clásico problema del *vecino más próximo*. Por último, comentaremos brevemente la reducción del coste de ejecución de dicho problema mediante el lema.

3.1. Aplicación 1

Ejemplo previo

Sea X un conjunto de n puntos en $\mathbb{R}^d$. En este ejemplo, trataremos de sumergir dicho conjunto en $\mathbb{R}^k$ donde $k < d$ de tal forma que se -casi- preserven las distancias entre los n puntos.

En este ejemplo previo, X será un conjunto de $n = 2500$ elementos en $\mathbb{R}^{5000}$ y fijaremos un nivel de tolerancia $\epsilon = 0{,}25$. Por tanto,

$$k \ \geq \ \frac{4}{\epsilon^2/2 - \epsilon^3/3} \cdot \log \ n \ = \ \frac{4}{(0{,}25)^2/2 - (0{,}25)^3/3} \cdot \log \ 2500 \ = \ 1201{,}773.$$

Por comodidad, tomaremos $k = 1500$.

Recordamos los valores $(n, d, k, \epsilon) = (2500, 5000, 1500, 0{,}25)$. El algoritmo -implementado en *R*- que seguiremos para 0.25-sumergir X en $\mathbb{R}^{1500}$ es el que sigue:

1. Simulamos una matriz aleatoria de dimensión $n \times d$. Los números simulados siguen una distribución uniforme. Hay paquetes de R que generan números absolutamente aleatorios pero no son capaces de generar tantos como necesitamos sin repetición. Observamos que

la función *rand(d,k)* de *MATLAB* también genera números aleatorios siguiendo una distribución uniforme. Por tanto, decidimos implementar la solución con R por la facilidad que nos proporciona dicho software para calcular la distancia entre elementos;

2. Simulamos k enteros entre 1 y n sin repetición mediante la función *sample* de R;
3. Generamos un nuevo conjunto, Y, formado por las k columnas seleccionadas de los n elementos;
4. Calculamos la matriz de distancias de los conjuntos X, Y con la función *dist* de R;
5. Multiplicamos todos los elementos de la matriz de distancias de Y por $\sqrt{d/k}$;
6. Observemos que la matriz de distancias de X,Y tiene dimension $n \times n$. Dividimos la matriz de distancias de X entre la matriz generada en el *paso 5*, elemento a elemento;
7. Por último, comprobamos que todos los elementos de dicha matriz están comprendidos entre $1-\epsilon$ y $1+\epsilon$, esto es, 0,75 y 1,25.

Resultados:

- Calcular la distancia entre los puntos en $\mathbb{R}^{5000}$ ha tenido un coste temporal de 290.22 segundos mientras que, en $\mathbb{R}^{1500}$, ha tenido un coste de 74 segundos.
- El factor máximo de distorsión entre puntos ha sido $0{,}1607496 < 0{,}25 = \epsilon$.

Por tanto, podemos concluir que hemos encontrado un subespacio de $\mathbb{R}^{1500}$ en el cual la distancia entre los puntos de X se distorsiona menos de ϵ reduciendo, además, el coste temporal en la ejecución del algoritmo casi en un 400 %.

Observación 1: en la demostración original de *Johnson-Lindenstrauss* hemos visto como la probabilidad de que todas las parejas de X estén dentro de $[(1-\epsilon)\cdot\frac{k}{d}, (1+\epsilon)\cdot\frac{k}{d}]$ es mayor o igual que $1/n$, en nuestro caso, $1/2500$. Con lo que sabemos hasta ahora, sorprende que el primer subespacio aleatorio de $\mathbb{R}^{1500}$ que hemos generado verifique el lema de *Johnson-Lindenstrauss* si bien, posteriormente, otros autores han mejorado las cotas del lema demostrando que con probabilidad mayor a $1/2$ podemos encontrar dicho subespacio.

Observación 2: El código que hemos creado para obtener los resultados figura en el anexo *-Ejemplo Previo-*. Puesto que hemos utilizado semillas para generar los números aleatorios, cualquier reproducción del experimento reflejará los mismos resultados.

Variante primera

Fijados $d = 5000$ y $k = 1500$, veamos cómo afecta el aumento de puntos de X a la eficiencia del algoritmo.

Número de puntos	**Tiempo en $\mathbb{R}^{5000}$**	**Tiempo en $\mathbb{R}^{1500}$**
2500	280.03 segundos $\approx$ 5 minutos	75.72 segundos $\approx$ 1 minuto
3000	402.38 segundos $\approx$ 7 minutos	107.27 segundos $\approx$ 2 minutos
3500	564.07 segundos $\approx$ 9 minutos	153.42 segundos $\approx$ 3 minutos
4000	782.21 segundos $\approx$ 13 minutos	208.75 segundos $\approx$ 3 minutos
4500	962.31 segundos $\approx$ 16 minutos	271.39 segundos $\approx$ 5 minutos
5000	1176.05 segundos $\approx$ 20 minutos	336.71 segundos $\approx$ 6 minutos

Cuadro 3.1: Tiempo invertido en calcular la matriz de distancias según número de puntos.

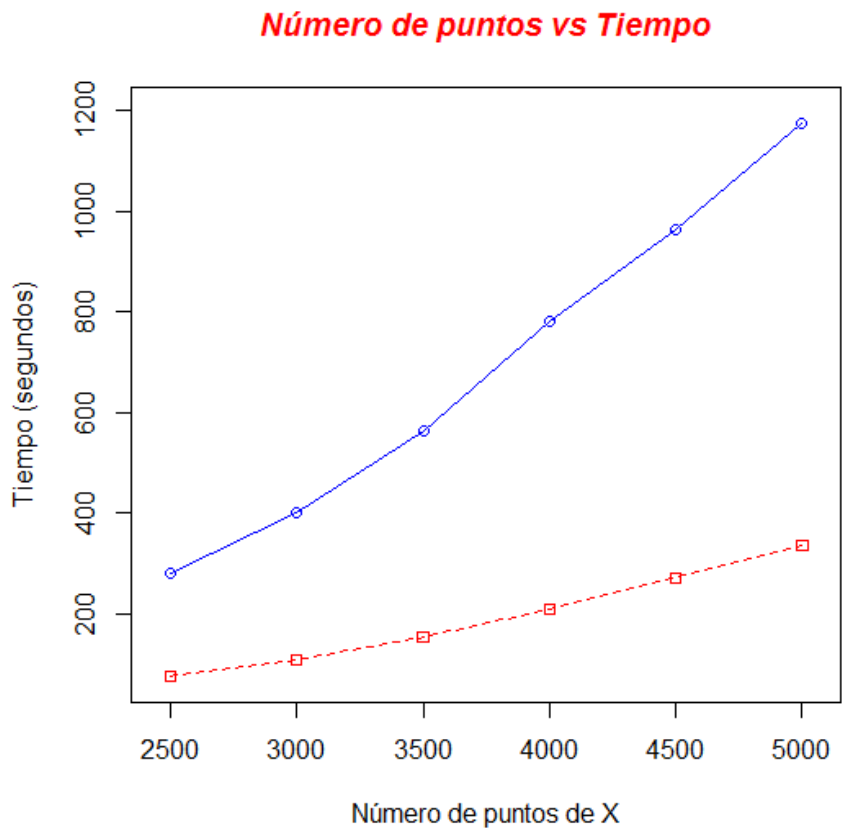


Figura 3.1: En azul, los tiempos según número de puntos en $\mathbb{R}^{5000}$. En rojo, en $\mathbb{R}^{1500}$.

Observación 1: Vemos como, a medida que aumentamos el número de puntos, la inmersión de X en $\mathbb{R}^{1500}$ genera mejoras computacionales mayores.

Observación 2: Como ya hemos comentado anteriormente, lo que limita el valor de k es tanto ϵ como n. Por tanto, fijado ϵ, aumentar el número de puntos debería incrementar el valor de k sustancialmente. Sin embargo, el término que influye en k es $\log n$ y, por esto, el aumento del valor de n tiene un influencia relativa en k. Veámoslo:

Número de puntos	**Valor mínimo de k**
2500	1203
3000	1231
3500	1254
4000	1275
4500	1293
5000	1309

Cuadro 3.2: Tiempo invertido en calcular la matriz de distancias según número de puntos.

Observación 3: El código que hemos creado para obtener los resultados figura en el anexo -*VARIANTE PRIMERA*-. Puesto que hemos utilizado semillas para generar los números aleatorios, cualquier reproducción del experimento reflejará los mismos resultados. Se adjuntan, además, los resultados obtenidos y el código de generación de las gráficas.

Variante Segunda

Fijado $n = 2500$, comprobaremos como la dimensión del espacio inicial d no influye en la inmersión. Es decir, podemos sumergir conjuntos de dimensión más alta en $\mathbb{R}^{1500}$. A prori, deberíamos obtener mejores prestaciones. Veámoslo:

Espacio inicial	**Tiempo en el espacio inicial**	**Tiempo en $\mathbb{R}^{1500}$**
$\mathbb{R}^{5000}$	279.25 segundos $\approx$ 5 minutos	73.24 segundos $\approx$ 1 minuto
$\mathbb{R}^{6000}$	339.22 segundos $\approx$ 6 minutos	73.90 segundos $\approx$ 1 minutos
$\mathbb{R}^{7000}$	397.79 segundos $\approx$ 7 minutos	73.50 segundos $\approx$ 1 minutos
$\mathbb{R}^{8000}$	457.42 segundos $\approx$ 8 minutos	73.54 segundos $\approx$ 1 minutos
$\mathbb{R}^{9000}$	516.53 segundos $\approx$ 9 minutos	73.00 segundos $\approx$ 1 minutos
$\mathbb{R}^{10000}$	575.77 segundos $\approx$ 10 minutos	72.40 segundos $\approx$ 1 minutos

Cuadro 3.3: Tiempo invertido en calcular la matriz de distancias según dimensión incial del problema.

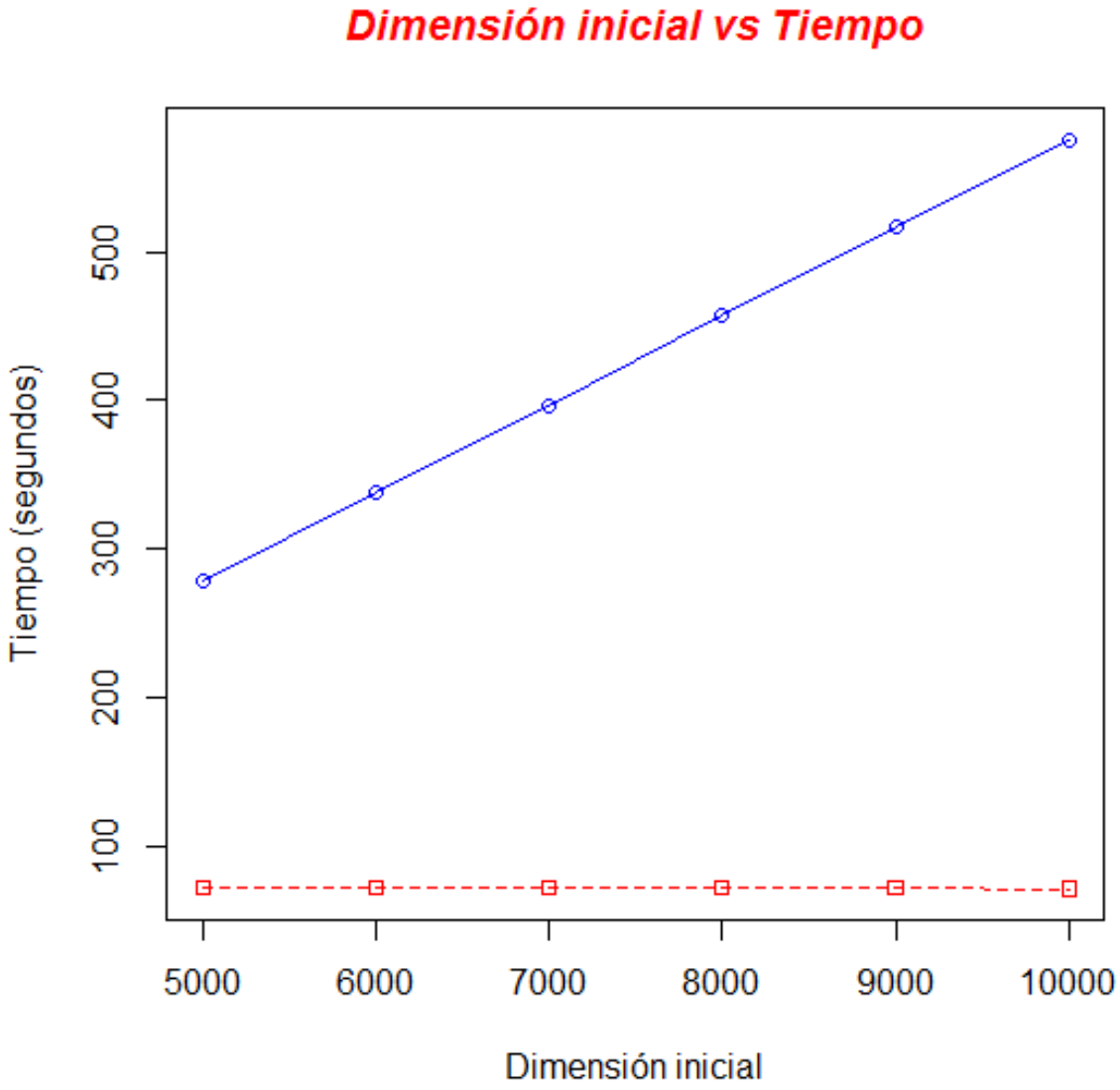


Figura 3.2: En azul, los tiempos según la dimensión incial del problema. En rojo, en $\mathbb{R}^{1500}$.

Observación 1: A medida que aumenta la dimensión del espacio incial d, calcular la matriz de distancias de un conjunto X de $\mathbb{R}^d$ es más costoso. Sin embargo, como podemos sumergir cualquiera de dichos conjuntos de dimensión distinta en $\mathbb{R}^{1500}$ -sin aumentar el valor de ϵ-, el tiempo utilizado es constante para cualquier d.

Observación 2: El código que hemos creado para obtener los resultados figura en el anexo -*VARIANTE SEGUNDA*-. Puesto que hemos utilizado semillas para generar los números aleatorios, cualquier reproducción del experimento reflejará los mismos resultados. Se adjuntan, además, los resultados obtenidos y el código de generación de las gráficas.

Número de veces que encontramos el espacio de forma aleatoria

Hemos encontrado un método que nos permite reducir el coste computacional de este tipo de problemas de forma sustancial. Pero, ¿cuánto nos puede costar encontrar un subespacio de verifique que $1-\epsilon = 0{,}75 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}25 = 1+\epsilon$ para todo $i,j = 1,2,\ldots,2500$ con $i \neq j$? Recordamos que $v_i \in \mathbb{R}^{5000}$ y $P(v_i) \in \mathbb{R}^{1500}$ para todo i.

Veámoslo:

		Valor de ϵ	**¿Subespacio válido?**
Intento 1:	$0{,}9388281 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}074431$	0,074431	Sí
Intento 2:	$0{,}9386575 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070887$	0,070887	Sí
Intento 3:	$0{,}9392014 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}068900$	0,068900	Sí
Intento 4:	$0{,}9389769 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}064403$	0,064403	Sí
Intento 5:	$0{,}9396454 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071320$	0,071320	Sí
Intento 6:	$0{,}9413918 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}067061$	0,067061	Sí
Intento 7:	$0{,}9400079 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070161$	0,070161	Sí
Intento 8:	$0{,}9383350 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070853$	0,070853	Sí
Intento 9:	$0{,}9404110 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070819$	0,070819	Sí
Intento 10:	$0{,}9392359 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071365$	0,071365	Sí
Intento 11:	$0{,}9386313 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}066601$	0,066601	Sí
Intento 12:	$0{,}9412088 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071227$	0,071227	Sí
Intento 13:	$0{,}9398468 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070392$	0,070392	Sí
Intento 14:	$0{,}9404493 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070267$	0,070267	Sí
Intento 15:	$0{,}9357975 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}073751$	0,073751	Sí
Intento 16:	$0{,}9395957 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}072688$	0,072688	Sí
Intento 17:	$0{,}9404958 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}064509$	0,064509	Sí
Intento 18:	$0{,}9435611 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070621$	0,070621	Sí
Intento 19:	$0{,}9282895 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071146$	0,0717105	Sí
Intento 20:	$0{,}9399594 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}072397$	0,072397	Sí
Intento 21:	$0{,}9422792 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}067393$	0,067393	Sí
Intento 22:	$0{,}9397705 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}069425$	0,069425	Sí
Intento 23:	$0{,}9417293 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}069579$	0,069579	Sí
Intento 24:	$0{,}9320003 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070277$	0,070277	Sí
Intento 25:	$0{,}9400761 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}066646$	0,066646	Sí

Cuadro 3.4: Intentos del 1 al 25.

		Valor de ϵ	**¿Subespacio válido?**
Intento 26:	$0{,}9416490 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}072900$	0,072900	Sí
Intento 27:	$0{,}9339683 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}068004$	0,068004	Sí
Intento 28:	$0{,}9373907 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}072913$	0,072913	Sí
Intento 29:	$0{,}9356557 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071467$	0,071467	Sí
Intento 30:	$0{,}9375267 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}067512$	0,067512	Sí
Intento 31:	$0{,}9397467 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}076800$	0,076800	Sí
Intento 32:	$0{,}9435947 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071484$	0,071484	Sí
Intento 33:	$0{,}9383287 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}073180$	0,073180	Sí
Intento 34:	$0{,}9372250 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071907$	0,071907	Sí
Intento 35:	$0{,}9403050 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070528$	0,070528	Sí
Intento 36:	$0{,}9423634 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070528$	0,070528	Sí
Intento 37:	$0{,}9423677 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}069598$	0,069598	Sí
Intento 38:	$0{,}9364676 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071891$	0,071891	Sí
Intento 39:	$0{,}9418236 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}074232$	0,074232	Sí
Intento 40:	$0{,}9338670 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070481$	0,070481	Sí
Intento 41:	$0{,}9381978 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}076242$	0,076242	Sí
Intento 42:	$0{,}9415986 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070910$	0,070910	Sí
Intento 43:	$0{,}9383465 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}070227$	0,070227	Sí
Intento 44:	$0{,}9404189 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071500$	0,071500	Sí
Intento 45:	$0{,}9397721 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071279$	0,071279	Sí
Intento 46:	$0{,}9385408 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}066882$	0,066882	Sí
Intento 47:	$0{,}9415142 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}076661$	0,076661	Sí
Intento 48:	$0{,}9440974 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}064643$	0,064643	Sí
Intento 49:	$0{,}9407316 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}078298$	0,078298	Sí
Intento 50:	$0{,}9405227 < \frac{d(v_i,v_j)}{d(P(v_i),P(v_j))} < 1{,}071065$	0,071065	Sí

Cuadro 3.5: Intentos del 26 al 50.

Observación 1: El código que hemos creado para obtener los resultados figura en el anexo -*Número de veces que encontramos el espacio de forma aleatoria*-. Puesto que hemos utilizado semillas para generar los números aleatorios, cualquier reproducción del experimento reflejará los mismos resultados. Se adjuntan, además, los resultados obtenidos y el código de generación de las gráficas. Notar que la ejecución de dicho código puede tardar, según las prestaciones del ordenador en el que se ejecute, más de dos horas.

Conclusión: Acabamos de ver que los 50 subespacios aleatorios generados son válidos. Como ya hemos dicho anteriormente, la probabilidad de encontar un subespacio válido es mucho mayor que la cota que proporcionaban Johnson y Lindenstrauss en la demostración original. Por último, notar que las aproximaciones obtenidas son mejores de lo esperado: $\epsilon < 0{,}08$.

3.2. Aplicación 2. *El vecino más próximo.*

El problema del vecino más próximo es un problema clásico que, generalmente, trata con datos de dimensión alta. Sea $\mathrm{P} = p_1, p_2, \ldots, p_m \in \mathbb{R}^d$. Dado $q \in \mathbb{R}^d$, buscamos hallar eficientemente el valor i que minimize $\|q - p_i\|$.

Sin tratar previamente los datos, el coste de una búsqueda es O(dn). Sin embargo, este problema puede ser resuelto en tiempo polinómico. Veamos primero un ejemplo.

Sea X un conjunto de n elementos en $\mathbb{R}^d$ donde $n = 2500$ y $d = 10000$. Llamaremos *nodo* i-ésimo del grafo g al elemento i-ésimo de X. Primero, trataremos de sumergir dicho conjunto en $\mathbb{R}^k$ donde $k = 2000$ de tal forma que, fijado un nodo, podamos encontrar de forma eficiente *el vecino más próximo*, esto es, el elemento de g que diste menos de dicho nodo. Posteriormente, fijados dos nodos cualesquiera, buscaremos cuál es el camino de menor longitud entre ellos. Fijaremos un nivel de tolerancia $\epsilon = 0{,}25$. Por comodidad, tomaremos el mismo conjunto X de observaciones que en la *Aplicación 1.* Creamos el siguiente grafo *no dirigido* en R mediante el paquete *igraph* con los 13 primeros elementos de la matriz:

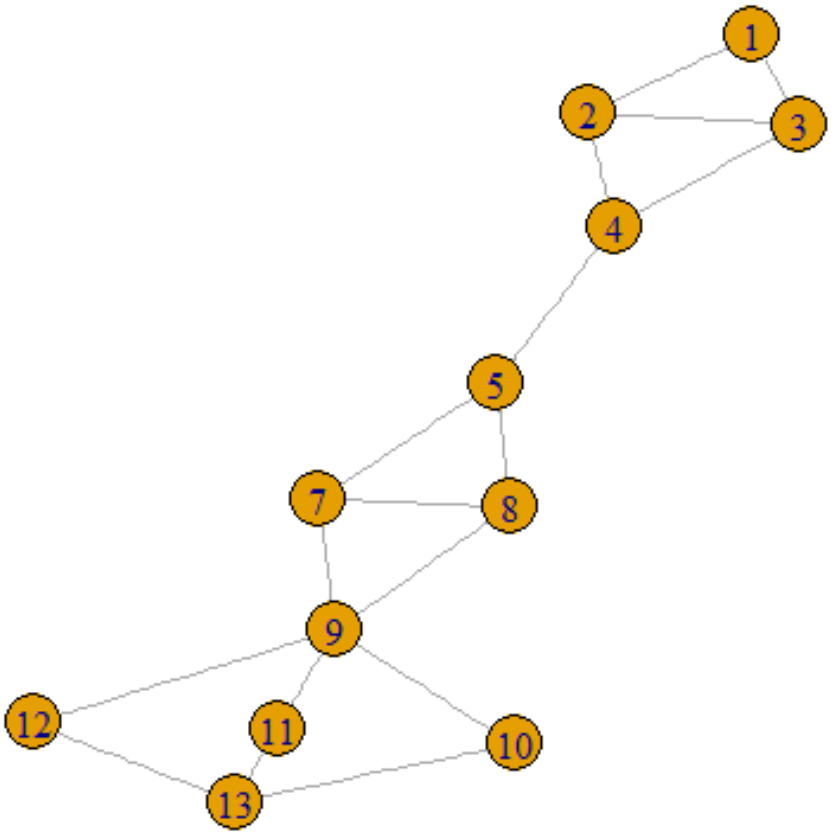


Figura 3.3: Grafo no dirigido sobre el que trabajaremos el problema del *vecino más próximo.*

El paquete *igraph* de R [12] es muy completo. Nosotros usaremos dos funciones llamadas *graph.knn*() y *get.shortest.paths*(). La primera, dado un grafo y un nodo, devuelve *el vecino más próximo*; la segunda, dado un grado y dos nodos, devuelve el camino de menor longitud que les une.

Mejora computacional

Tiempo en $\mathbb{R}^{10000}$	**Tiempo en $\mathbb{R}^{2000}$**
575.99 segundos	69.78 segundos

Cuadro 3.6: Tiempo global de ejecución.

Observación 1: El código que hemos creado para obtener los resultados figura en el anexo -*Aplicación Segunda*-.

Observación 2: Vemos cómo se ha reducido el coste computacional del problema inicial de forma considerable. Notar que, una vez calculada la matriz de distancias, la diferencia en el coste temporal que se produce a la hora de trabajar con un subconjunto o todos los elementos de X es despreciable.

Observación 3: Para generar el nuevo subespacio en $\mathbb{R}^{2000}$ hemos escodigo las 2000 primeras componentes de los elementos de X. Comprobemos si esta inmersión de X en el nuevo subespacio de $\mathbb{R}^{2000}$ es válida.

El vecino más próximo

En este apartado comprobaremos como tanto en el espacio inicial $\mathbb{R}^{10000}$, como en el subespacio en el que realizaremos la inmersión mediante el lema de *Johnson-Lindenstrauss* $\mathbb{R}^{2000}$, el cálculo del *vecino más próximo* es -casi- idéntico en todos los nodos. Veámoslo:

Nodo	**Vecino más próximo en $\mathbb{R}^{10000}$**	**Vecino más próximo en $\mathbb{R}^{2000}$**
1	3	3
2	3	4
3	4	4
4	5	5
5	4	4
6	-	-
7	8	8
8	9	9
9	10	10
10	9	9
11	9	9
12	13	13
13	12	12

Cuadro 3.7: Problema del vecino más próximo.

Camino de menor longitud que une dos nodos.

Ahora comprobaremos que, fijados dos nodos cualesquiera, el camino de menor longitud que les une es el mismo tanto en $\mathbb{R}^{10000}$ como en $\mathbb{R}^{2000}$.

Nodo Origen	**Nodo Destino**	**Camino de menor peso**
1	13	1 - 2 - 4 - 5 - 7 - 9 - 10 - 13
3	11	3 - 4 - 5 - 7 - 9 - 11
8	1	8 - 5 - 4 - 2 - 1

Cuadro 3.8: Camino de menor longitud que une Nodo Origen y Nodo Destino en $\mathbb{R}^{10000}$.

Nodo Origen	**Nodo Destino**	**Camino de menor peso**
1	13	1 - 2 - 4 - 5 - 7 - 9 - 10 - 13
3	11	3 - 4 - 5 - 7 - 9 - 11
8	1	8 - 5 - 4 - 2 - 1

Cuadro 3.9: Camino de menor longitud que une Nodo Origen y Nodo Destino en $\mathbb{R}^{2000}$.

Conclusión: Ha quedado comprobado que, mediante el *lema de Johnson-Lindestrauss*, podemos generar buenas aproximaciones del conjunto X en $\mathbb{R}^{2000}$ de tal forma que el orden entre los nodos se preserve mejorando, así, el coste computacional.

Como ya hemos dicho anteriormente, la búsqueda exhaustiva requiere $O(nd)$ comparaciones. Actualmente, existen algoritmos de búsqueda más eficientes como el PLEB o los kd-árboles pero dichos algoritmos dejan de ser eficientes en problemas donde la dimensión del espacio es muy alta, pues ambos métodos requieren un preprocesado (PLEB requiere $O(1/\epsilon)^d$ y kd-árboles $O(d \cdot n^{1-1/d})$. Sin embargo, pueden seguir siendo eficientes si, mediante el *lema de Johnson-Lindestrauss*, reducimos la dimensión del espacio a $\log n$. [3].

Capítulo 4

Estancia en prácticas

4.1. La empresa

La empresa *THE NETWIZZY COMPANY S.L.* fue fundada en 2008 y, desde entonces, desarrolla su labor en el sector de los juegos online. En su plataforma, *Our.com*, juegan decenas de miles de usuarios cada día y, en la actualidad, está compuesta por 36 juegos online multijugador. Los juegos que más éxito tienen en dicha plataforma son *Our Bingo* y *Our Slots*.

Figura 4.1: Logo Our.com

THE NETWIZZY COMPANY S.L. recoge gran cantidad de datos diariamente. La mayoría se guardan en documentos de texto plano con extensión *csv*. Dichos documentos se almacenan en carpetas según temática (por ejemplo, información sobre el registro de los usuarios, partidas de *OurBingo*, partidas de *OurSlots*,...) teniendo en cuenta además, que:

- se puede jugar tanto a *Our Bingo* como a *Our Slots* desde móviles Android, móviles Ios, el sito web *Our.com* y Facebook;
- *Our.com* está presente, también, en dos redes sociales rusas *(OK y VK)* y en otra japonesa

(YM).

4.1.1. Our Bingo

Our Bingo es un juego en el cual cada jugador compra uno o varios cartones con 15 números -comprendidos entre el 1 y 90- que tendrá que ir tachando conforme vayan saliendo de un bombo virtual de forma aleatoria. Algunas de las características de *Our Bingo* son:

- Bingo multijugador en tiempo real;
- Chat para interactuar con el resto de jugadores;
- Tres modalidades de juego:
  - Bingo clásico, de 75 ó 90 bolas;
  - Sala Viajes. Esta sala dispone de cuatro escenarios: París, Hawaii, Tokio y Egipto. El atractivo de la Sala Viajes reside en completar los puzzles en los que aparecen lugares de interés de los distintos escenarios: a medida que el usuario juega y se fideliza obtiene piezas de los distintos puzzles que debe ir completando;
  - Vídeo Bingo.
- Compra automática de cartones;
- Interacción con Facebook u otras plataformas, pudiendo compartir y publicar los premios ganados.

Figura 4.2: Bingo clásico

4.1.2. Our Slots

Un slot, coloquialmente una máquina tragaperras, simula las máquinas de juegos que funcionan introduciendo monedas y dan premios en metálico al azar. *Our Slots* ofrece *Vídeo Slots* y *Tragaperras Clásicas* con diferentes temáticas y funcionalidades:

- Temáticas:
  - *Vídeo Slots*: Dulce Tentación, Noche en Nueva York, Las 1001 Noches, Misterios de Egipto, Lobo Siberiano, Sherlock Files, El Legendario Dragón Rojo,...
  - *Tragaperras Clásicas*: Retro, Lucky Wheel, Super 5 Estrellas, Two Diamonds, Burning 777,...
- Funcionalidades:
  - Supera la barra de nivel para conseguir más monedas;
  - Diferentes líneas de apuesta;
  - Gran variedad de premios: sticky wild, wild y bonus;
  - Torneos: competiciones contra otros usuarios;
  - Interacción con Facebook u otras plataformas, pudiendo compartir y publicar los logros.

Figura 4.3: Tragaperra clásica: Burning 777.

4.2. Tareas previas

4.2.1. Movimientos de piezas de la Sala Viajes

El objetivo principal era implementar en *php* una función que facilitara al departamento de Marketing -encargado de atender las quejas de los clientes- las consultas acerca de las piezas de la Sala Viajes de *Our Bingo*. Además, esta tarea permitió que me familiarizara con el lenguaje de programación usado por el departamento de Estadística de la empresa, la forma en la cual la empresa almacena la información, la manera de acceder a ella,...

Figura 4.4: Puzzle de la colección Hawaii

Dicha función (llamada *Dame_MovimientosPiezas_SalaViajes*) mostraba, dado un *nombre de usuario*, *fechaInicio* y *fechaFin*, todos los movimientos de piezas que había realizado el usuario durante ese período de tiempo.

Los movimientos que un usuario puede realizar son:

- ganar una pieza;

- comprar una pieza;
- enviar una pieza a otro usuario;
- recibir una pieza de otro usuario;
- completar un puzzle.

Para facilitar la realización de la consulta implementamos, en *html4*, un formulario que permitiera introducir los datos de dicha consulta de una forma sencilla y devolviera la información acerca de los movimientos de piezas de una forma clara e intuitiva.

Figura 4.5: Formulario web

Puede suceder que, ocasionalmente, la información sita en la base de datos no coincida con la almacenada en los *csv* y, por ello, producirse errores en las consultas. Para solventar esto, implementamos una nueva versión de la función *Dame_MovimientosPiezas_SalaViajes* que, además, hiciera la consulta a la base de datos y comparase la información obtenida por ambos medios mostrando en verde las filas en las que son coincidentes y en rojo las que no.

Por último, añadimos la función y el formulario a *Estadistica Our* para que el personal de Marketing puediera acceder a dicha consulta.

4.2.2. Sentiment Analysis

Podemos definir el *Sentiment Analysis* como el tratamiento computacional de las opiniones, sentimientos y fenómenos subjetivos en los textos. Dicho de otro modo, consiste en implementar

Ganados:

fecha	collection	photo	item	messageId	coinsWon
2016-03-09 00:42:20	PARIS	4		awardCollectionCompleted	5000
2016-03-09 00:43:33	HAWAII	79		awardCollectionCompleted	5000
2016-03-09 00:43:58	HAWAII	83		awardCollectionCompleted	5000
2016-03-09 05:21:26	HAWAII	80	12	awardCollectionPhotos	
2016-03-09 06:03:06	HAWAII	80	4	awardCollectionPhotos	
2016-03-09 06:39:51	PARIS	5	4	awardCollectionPhotos	
2016-03-09 07:21:31	PARIS	5	2	awardCollectionPhotos	
2016-03-09 19:51:00	HAWAII	80	8	awardCollectionPhotos	
2016-03-09 20:30:12	HAWAII	80	12	awardCollectionPhotos	
2016-03-09 21:16:46	HAWAII	80	4	awardCollectionPhotos	
2016-03-09 21:58:26	HAWAII	80	8	awardCollectionPhotos	
2016-03-09 10:59:55	PARIS	5	4	awardCollectionPhotos	
2016-03-09 11:42:19	HAWAII	80	1	awardCollectionPhotos	
2016-03-09 12:28:52	HAWAII	80	5	awardCollectionPhotos	
2016-03-09 13:19:39	PARIS	5	8	awardCollectionPhotos	
2016-03-09 14:51:06	PARIS	5	8	awardCollectionPhotos	
2016-03-09 17:10:05	PARIS	5	10	awardCollectionPhotos	
2016-03-09 17:56:40	HAWAII	80	10	awardCollectionPhotos	

Comprados:

fecha	collection	photo	item	messageId
2016-03-09 00:42:20	PARIS	4	3	buyCollectionItem
2016-03-09 00:43:29	HAWAII	79	1	buyCollectionItem
2016-03-09 00:43:33	HAWAII	79	8	buyCollectionItem
2016-03-09 00:43:54	HAWAII	83	11	buyCollectionItem
2016-03-09 00:43:58	HAWAII	83	8	buyCollectionItem

Recibidos:

Recibido de	fecha	collection	photo	item	messageId	coinsWon
brigittedel817	2016-03-09 06:42:59	PARIS	5	12	sendCollectionItemFromUserToUser	200

Enviados:

Enviado a	fecha	collection	photo	item	messageId	coinsWon
anaolaranpena	2016-03-09 19:42:46	PARIS	1	2	sendCollectionItemFromUserToUser	200
anaolaranpena	2016-03-09 19:42:50	PARIS	1	7	sendCollectionItemFromUserToUser	200
anaolaranpena	2016-03-09 19:42:52	PARIS	1	10	sendCollectionItemFromUserToUser	200
anaolaranpena	2016-03-09 19:42:54	PARIS	1	9	sendCollectionItemFromUserToUser	200
brigittedel817	2016-03-09 11:00:22	PARIS	5	4	sendCollectionItemFromUserToUser	200
eddyvarin14	2016-03-09 18:04:37	PARIS	4	6	sendCollectionItemFromUserToUser	200

Figura 4.6: Ejemplo de ejecución. Este usuario ha ganado items, ha completado colecciones, ha comprado items, ha recibido items de otros usuarios y también los ha enviado. Además, la información de la base de datos y de los *csv* coincide. Por privacidad, se oculta el nombre de usuario.

un programa que sea capaz de entender e interpretar el significado de un texto digital haciéndolo operativo para su análisis y clasificación (objetivo/subjetivo, positivos/negativos,...).

Mediante el *Sentiment Analysis* las empresas pueden seguir la valoración de sus productos por parte de los clientes y evaluar el nivel de reputación de la compañía a partir de los comentarios de la red, conocer su nivel de posicionamiento y resonancia en los medios sociales, elaborar estrategias de marketing digital mucho más elaboradas y efectivas,...

R tiene varios paquetes que permiten hacer este tipo de estudios. Tras analizar en *R* -usando las librerías plyr, twitteR, rjason, tm, sentiment (sólo reconoce lenguaje natural en inglés) y wordcloud (librería gráfica)- el hashtag #UCLfinal obtenemos lo que sigue:

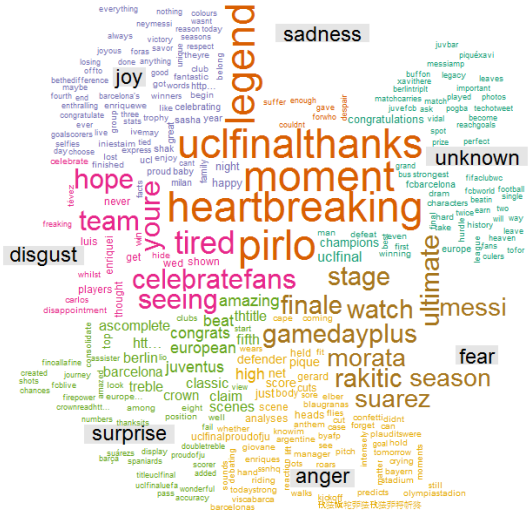


Figura 4.7: Ejemplo de uso de la librería wordcloud tras realizar *Sentiment Analysis* del hashtag #UCLfinal

Tras este ejemplo previo, analizaremos el chat de las salas de *Our Bingo* intentando extraer conclusiones beneficiosas para la empresa. Analizando sólo los comentarios en castellano del chat de las salas de *Our Bingo* los días 08/03/2016 y 09/03/2016 pudimos concluir que las palabras más utilizadas por los usuarios eran:

Palabra	**Frecuencia**
gratis	520
cartones	351
xfavooor	212
monedas	211
xfavor	208

Cuadro 4.1: Resultados obtenidos *Sentiment Analysis*

El resultado obtenido es lógico ya que, durante esos dos días, tuvimos un *bug* en *Our Bingo* y los usuarios que se quedaban sin saldo recibían cartones gratis, es decir, podían seguir jugando de forma ilimitada. Analizando las frases del chat que contienen la palabra *gratis* podemos ver como los usuarios se estaban avisando y comentado dicha eventualidad.

Ampliamos el estudio anterior escribiendo un script en *R* que mostrara las N-tuplas más frecuentes en un texto. Veamos un ejemplo de ejecución del script sobre el chat de *Our Bingo*:

Palabra	**Frecuencia**
(mandar, cartones, xfavor)	210
(foto, hawaii, 2)	126
(hawaii, 2, 11)	126

Cuadro 4.2: Ejemplo de ejecución para N=3

Analizando los datos vemos que hay un usuario que, reiteradamente, pide que le manden la pieza 11 de la segunda foto de Hawaii, esto es, *foto hawaii 2 11*. Además, la 3-tupla más repetida en el chat es *mandar cartones xfavor*, una expresión muy habitual de los usuarios para pedir a otros usuarios cartones de *Our Bingo* para poder seguir jugando.

4.3. Propuesta técnica

4.3.1. Descripción y objetivos

El objetivo de la estancia en prácticas en *THE NETWIZZY COMPANY S.L.* es, mediante técnicas de análisis de minería de datos, intentar entender y deducir por qué los usuarios cambian de slot en *Our Slots*. Para ello, nos centraremos en el estudio de 10 sujetos propuestos por la empresa.

La motivación de este estudio es clara: tener un conocimiento mayor de por qué un usuario cambia de slot ayudará a la empresa a conocer y entender mejor las necesidades y deseos del usuario pudiendo, de esta forma, satisfacer mejor a dicho usuario.

Para realizar este estudio haremos uso de toda la información que la empresa recaba diariamente de la interacción con el usuario en *Our.com*.

4.3.2. Tareas

1. Buscar toda la información necesaria y extraer sólo la parte que será útil para el estudio. Este será el paso más costoso ya que deberemos procesar archivos muy pesados y, además, la información está dispersa e inconexa. Para obtener toda la información acerca de los usuarios (datos personales, fecha de registro, información sobre las partidas jugadas,...) deberemos consultar varios archivos con extensión *csv* y un archivo con extensión *log* que contiene información sobre las sesiones y las partidas jugadas. Algunas de las cuestiones que deberemos resolver son:

    - ¿Cuál es la forma más eficiente de procesar los archivos con extensión *log*?
    - ¿Cómo filtrar dichos archivos?

- ¿Cómo relacionar la información personal de cada usuario (sita en los ficheros *csv* de la carpeta Registros) con la información de las partidas jugadas de los archivos *rpc_OurSlotsAwardCoins* de una forma eficiente?

2. Realizar el estudio estadístico mediante varias técnicas;
3. Entender los resultados obtenidos y extraer de ellos conclusiones válidas para *THE NETWIZZY COMPANY S.L.*

4.3.3. Preparación de los datos

Empezamos revisando las funciones que tiene ya implementadas el departamento de Estadística por si existiera alguna que pudiera sernos de utilidad. Destacamos tres:

- *Usuarios_DameSesiones_EntreFechas(...)*: este método devuelve los datos de las sesiones de los usuarios en los juegos seleccionados entre las fechas *FechaInicial* y *FechaFinal*. La información se extrae de *−CSVs−/app/app_fecha*;
- *RPC_OurSlots_DameListaTiradas(...)*: este método devuelve los datos de las tiradas de slots de los usuarios que le indicamos, tomando la información de *RPC/rpc_OurSlots−−AwardsCoins_fecha.log*;
- *RPC_DameLineas_Usuario(...)*: este método devuelve las líneas de *RPC/rpc_fecha.log* donde aparece el nombre o el *id* del usuario para ver toda la actividad que ha generado.

Número de sesiones necesarias para realizar el estudio

Estimamos, a priori, que necesitaríamos unas 500 sesiones en *Our Slots* para realizar un estudio de calidad. Para fijar las fechas del estudio realizamos el siguiente script en php:

```
$FechaInicial = '2015-10-01';
$FechaFinal = '2016-03-31';
$ListaNombres['sujeto1']=1;

$n = $OurStats -> Usuarios_DameSesiones_EntreFechas ('NumSes',
$FechaInicial, $FechaFinal, 'OurSlots',"","","","", $ListaNombres);
```

Este script muestra como el usuario *sujeto1* ha iniciado un total de 442 sesiones en *OurSlots* desde 2015-10-01 hasta 2016-03-31. Éstas son, por tanto, las fechas definitivas del estudio.

Instalación de un terminal Linux y acceso al servidor

Una vez seleccionadas las fechas del estudio debíamos, de una forma eficiente, generar nuevos ficheros en los que sólo apareciera la información relevante para llevar a cabo el proyecto. Para ello instalamos el siguiente sofware: *PuTTy* y *FileZilla*:

FileZilla es un gestor de FTP para la administración de archivos:

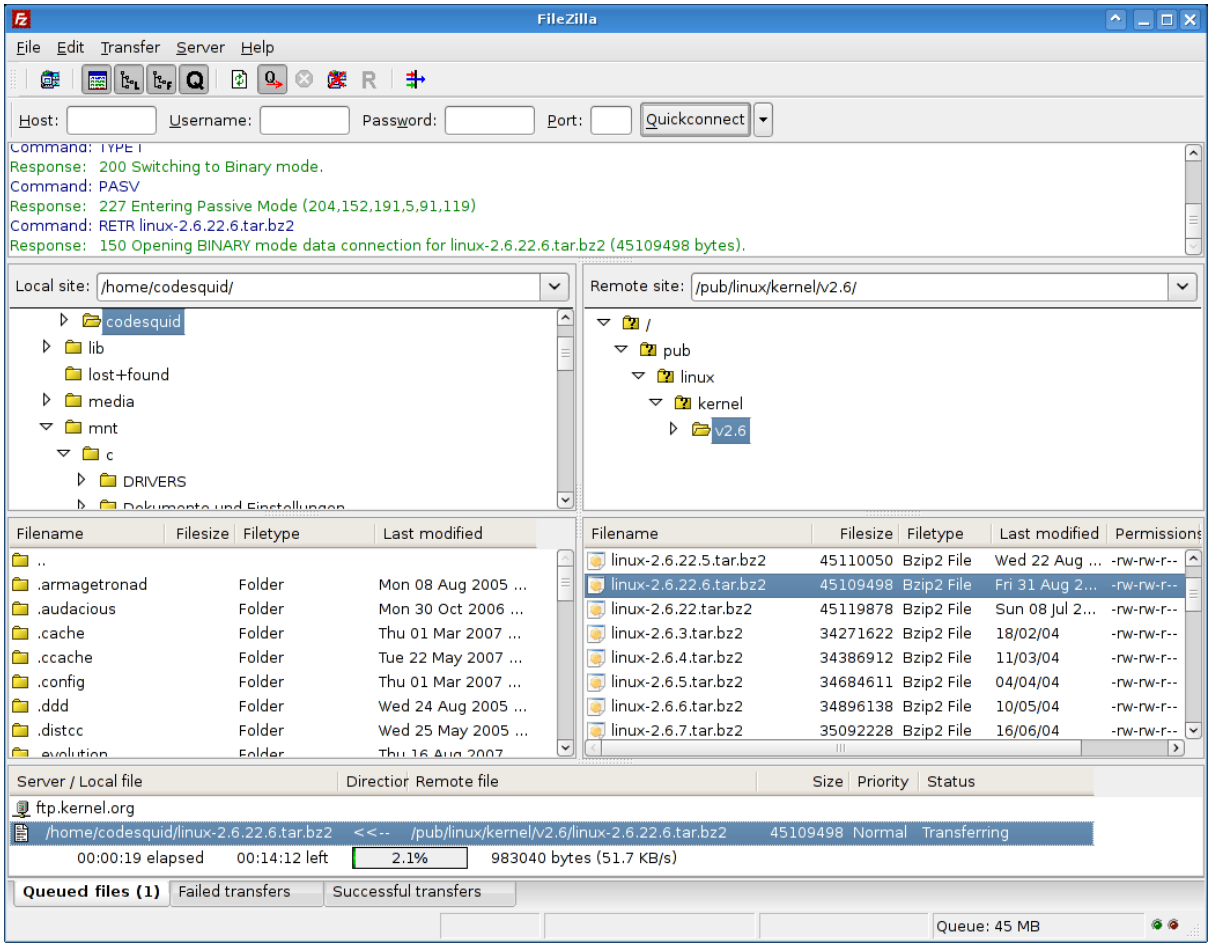


Figura 4.8: Interfaz de usuario del gestor de archivos FileZilla.

El uso de FileZilla es imprescindible puesto que sólo tengo permisos de escritura en *mi home*. Por tanto, deberemos generar los archivos en dicho directorio y descargarlos con dicho gestor de archivos.

PuTTy es un cliente de red que soporta el protocolo *SSH* y sirve, principalmente, para iniciar una sesión remota en otro servidor -en nuestro caso, el de la empresa-.

Una vez configurado el software, disponemos de un terminal linux conectado de forma remota con el servidor de la empresa:

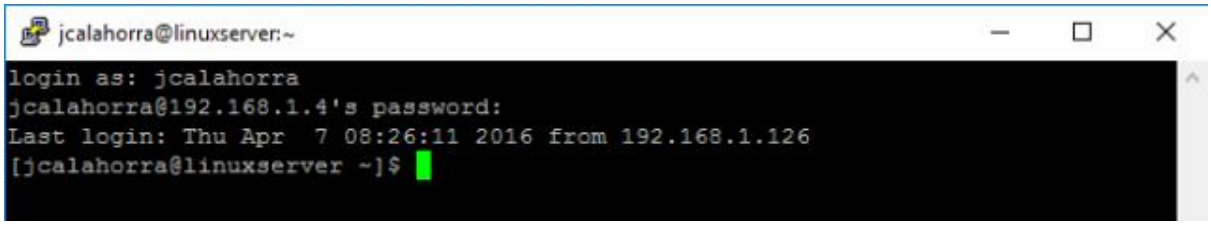


Figura 4.9: Terminal linux conectado al servidor de la empresa con mi usuario.

Para comprobar que todo el software funcionaba correctamente hicimos la siguiente prueba: mostrar un archivo por pantalla, de forma controlada, mediante el comando cat y less:

- El comando cat (por concatenar) es un programa de Unix usado para concatenar y mostrar archivos;
- El comando less permite paginar texto por pantalla y una completa navegación por el contenido del archivo, utilizando un mínimo de recursos del sistema.

Si no usáramos el comando less sería una ejecución suicida ya que se mostarían todas las líneas del fichero (hay demasiadas) por pantalla a una velocidad ininteligible y no podríamos navegar sobre ellas.

```
[jcalahorra@linuxserver -RPC-]$ cat rpc_OurSlotsAwardsCoins_2016-01-01.log | less

[01-Jan-2016 00:26:49] general.php OurSlotsAwardsCoins {"result":null,"levelCoin
s":0,"jackPot":0,"keys":0,"type":"result","slotsId":"luckywheel","extraCoins":0,
"slotsNumId":34,"coins":0,"userId":13146555,"bet":30,"endCoins":119095,"userName
":"jocelynegus","lines":5,"levelPoints":12,"initialCoins":118945} 90001 1ms[01-Jan-
2016 00:26:49] general.php OurSlotsAwardsCoins {"result":null,"coins":-2
000,"bet":50,"userId":15518777,"userName":"araratks5","endCoins":36456,"lines":4
0,"slotsId":"buffalo","initialCoins":38456,"type":"spin","slotsNumId":27} 90001
1ms[01-Jan-2016 00:26:49] general.php OurSlotsAwardsCoins {"result":null,"levelCoin
s":0,"jackPot":0,"keys":0,"type":"result","slotsId":"buffalo","extraCoins":0,"sl
otsNumId":27,"coins":2000,"userId":15518777,"bet":50,"endCoins":38456,"userName"
:"araratks5","lines":40,"levelPoints":132,"initialCoins":34456} 90001 1ms[01-Jan-2016
00:26:49] general.php OurSlotsAwardsCoins {"result":null,"coins":-1
000,"bet":20,"userId":6802455,"userName":"simone1978","endCoins":41487,"lines":5
0,"slotsId":"dragonfortunes","initialCoins":42487,"type":"spin","slotsNumId":35}
90001 1ms[01-Jan-2016 00:26:49] general.php OurSlotsAwardsCoins
```

Figura 4.10: Extracto del contenido de *rpc_OurSlotsAwardsCoins_2016-01-01.log*.

Archivos *rpc_OurSlotsAwardsCoins_XXXX-XX-XX.log*

Los archivos *rpc_OurSlotsAwardsCoins_XXXX-XX-XX.log* son ficheros de texto que almacenan la información registrada por el servidor de las partidas jugadas. Cada una de las líneas se guarda en formato *JSON* (JavaScript Object Notation), un formato de texto ligero y fácil de evaluar.

Como podemos ver, hay información redundante. El servidor guarda dos líneas por cada *spin*: apuesta y resultado. Lo hace de la siguiente forma:

```
Apuesta---> [17-Jan-2016       06:27:19]       general.php       OurSlotsAwardsCoins
{"result":null,"coins":0,"bet":6,"userId":9451499,"userName":"sujeto1","endCoins":3151
,"lines":30,"slotsId":"lucky-dragon","initialCoins":3151,                "type":"spin",
"slotsNumId":21}900011msResultado---> [17-Jan-2016       06:27:19]       general.php
OurSlotsAwardsCoins{"result":null,"levelCoins":0,"jackPot":0,"keys":0,"type":"result",
"slotsId":"luckydragon","extraCoins":0,"slotsNumId":21,"coins":0,"userId":9451499,"bet
":6,"endCoins":3151,"userName":"sujeto1","lines":30,"levelPoints":30,"initialCoins":31
51} 90001 1ms
```

Figura 4.11: Información registrada por el servidor en un *spin*

Podemos observar como toda la información relevante de la línea *Apuesta* está en *Resultado*. Por tanto, disponiendo del terminal ya estábamos en disposición de cribar los archivos de una de las formas más eficientes posibles: utilizando los comandos *grep* y *zgrep*.

El comando grep nos permite buscar, dentro de los archivos, las líneas que concuerdan con un patrón. Por tanto, pudimos buscar dentro de los ficheros las líneas que contenían el nombre del sujeto de estudio y que, además, eran líneas resultado, despreciando el resto. Finalmente, volcamos el resultado del comando en un fichero para, posteriormente, poder acceder a dicha información. Veámoslo:

Con el comando *cd /* cambiamos del directorio actual *mi home* al directorio raíz del servidor

```
[jcalahorra@linuxserver ~]$ cd /
```

Accedemos al directorio del servidor en el que se encuentran los archivos .log:

```
[jcalahorra@linuxserver  /]$  cd  var/www/lighttpd/estadisticas/Our-Datos2/./-
RPC-/
```

Una vez dentro del directorio cribamos, primero, los archivos sin comprimir:

```
for j in {1..3}; do
      for i in {..}; do
            grep sujeto1 rpc_OurSlotsAwardsCoins.log-20150$j$i.zip | grep levelCoins >
~/2015-0$j-$i;
      done;
done;
```

Ahora, los comprimidos:

```
for j in {10..12}; do
      for i in {..}; do
            zgrep sujeto1 rpc_OurSlotsAwardsCoins.log-2015$j$i.zip | grep levelCoins >
~/2015-$j-$i;
      done;
done;
```

Comprobemos, por último, que los archivos se han generado correctamente en el directorio. Para ello, listaremos el directorio *mi home*:

```
[jcalahorra@linuxserver -RPC-]$ ls ~

2015-10-01  2015-11-01  2015-12-02  2016-01-02  2016-02-02  2016-03-04
2015-10-02  2015-11-02  2015-12-03  2016-01-03  2016-02-03  2016-03-05
2015-10-03  2015-11-03  2015-12-04  2016-01-04  2016-02-04  ...
```

Posteriormente, haciendo uso de *FileZilla*, descargamos los archivos del directorio del servidor Unix a una carpeta en Windows pudiendo, de esta manera, aplicar técnicas de estudio a los datos.

Ya estamos en disposición de reducir los datos a un sólo archivo *csv* que nos permita tratar los datos en el software matemático R mediante un script en *php*.

Pero antes, ¿qué contienen las líneas *JSON*?

Posición dentro del *JSON*	**Nombre de la variable**	**Contenido**
[0]	*result*	Contiene el valor null. Variable en desuso.
[1]	*levelCoins*	Monedas recibidas por haber subido de nivel.
[2]	*jackPot*	Monedas recibidas por ganar el jackPot.
[3]	*keys*	Contiene el valor 0. Variable en desuso.
[4]	*type*	Contiene el valor result. Variable en desuso.
[5]	*slotsId*	Varibale String con el identificativo del slot.
[6]	*extraCoins*	Monedas ganadas en los freeSpins (tiradas gratis).
[7]	*slotsNumId*	Variable entera con el identificativo del slot.
[8]	*coins*	Cantidad ganada en la tirada (incuyendo levelCoins, jackPot, extraCoins). Notar que esta variable es positiva o cero.
[9]	*userId*	Variable entera con el identificativo del usuario.
[10]	*bet*	Cantidad de monedas apostadas por línea.
[11]	*endCoins*	Cantidad de monedas del usuario tras la tirada.
[12]	*userName*	Varibale String con el identificativo del usuario.
[13]	*lines*	Número de líneas apostadas. Notar que la cantidad total apostada será: lines * bet.
[14]	*levelPoints*	Puntos de nivel ganados.
[15]	*initialCoins*	Cantidad de monedas del usuario antes de la tirada.

Cuadro 4.3: Contenido del *JSON*.

Generación del nuevo fichero

Ahora vamos a comprimir todos los ficheros de texto que hemos generado en sólo un *csv* donde cada línea contendrá la información de cada sesión del *sujeto1* dentro del mismo slot, esto es, una sesión en la cual dicho usuario haya jugado en tres slots difrentes, generará tres líneas dentro del fichero.

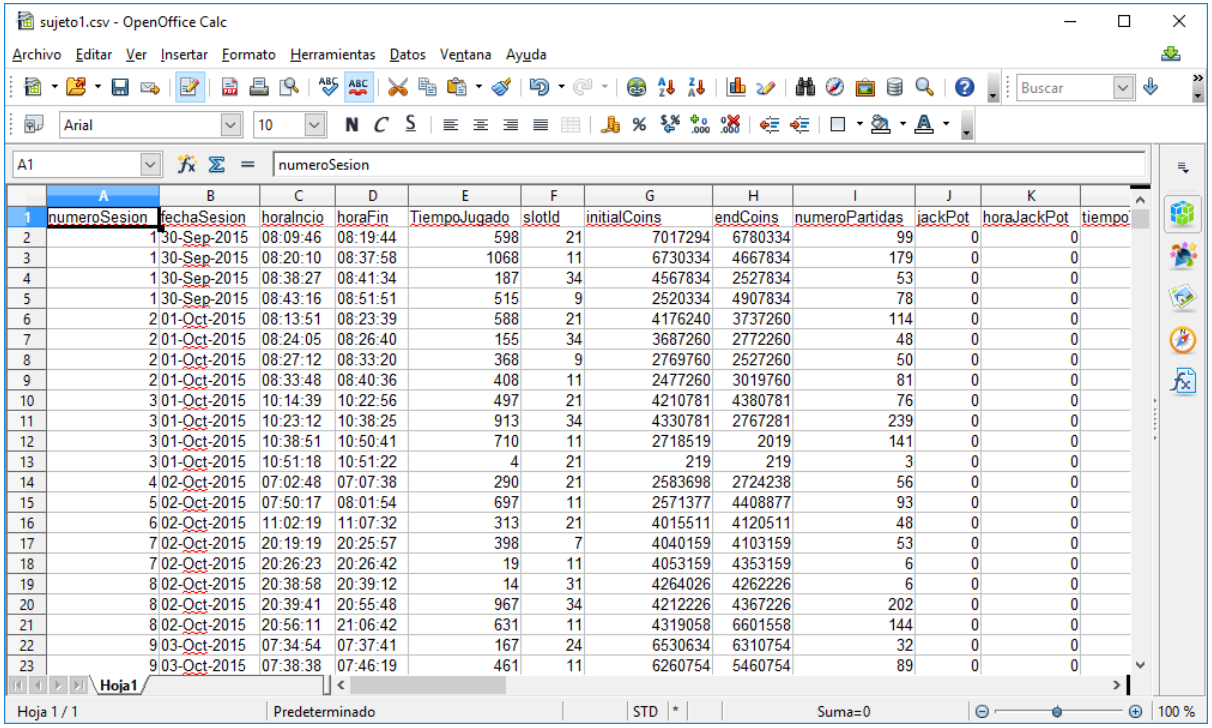
Además, añadiremos al *csv* estados de sesiones intermedias donde, obviamente, no se han producido cambios de slot ni cierres de sesión. El objetivo es predecir los motivos que provocan un cambio de slot y, para ello, intentaremos buscar patrones o eventualidades que diferencien las líneas del csv que son sesiones intermedias de las que son cambios de slot. Notar que un cambio de sesión implica un cambio de slot. Sin embargo, un cambio de slot no implica, necesariamente, un cambio de sesión.

Guardaremos 5 sesiones intermedias por cada cambio de slot o cierre de sesión de forma idénticamente distribuida.

	Nombre de la variable	**Contenido**
1	*numeroSesion*	Número de la sesión.
2	*fechaSesion*	Fecha de la sesión.
3	*horaInicio*	Hora de inicio de la sesión.
4	*horaFin*	Hora de fin de la sesión.
5	*TiempoJugado*	Duración de la sesión en mlseg.
6	*slotId*	Variable entera que identifica unequívocamente el slot.
7	*initialCoins*	Número de monedas al iniciar la sesión.
8	*endCoins*	Número de monedas al finalizar la sesión.
9	*numeroPartidas*	Número de partidas jugadas durante la sesión.
10	*jackPot*	Variable binaria que muestra si se ha ganado el *jackPot* durante la sesión en *slotId*.
11	*horaJackPot*	Hora a la que se ha conseguido el jackPot.
12	*partidasJackPot*	Partidas que se han jugado después de ganar el jackPot.
13	*levelCoins*	Variable binaria que muestra si se ha ganado el levelCoins durante la sesión en slotId.
14	*horaLevelCoins*	Hora a la que se ha conseguido el levelCoins.
15	*partidasLevelCoins*	Partidas que se han jugado después de ganar el levelCoins.
16	*extraCoins*	Variable binaria que muestra si se ha ganado el extraCoins durante la sesión en slotId.
17	*horaExtraCoins*	Hora a la que se ha conseguido el extraCoins.
18	*partidasExtraCoins*	Partidas que se han jugado después de ganar el extraCoins.
19,22 25,28 31,34	*winCoins(i)*	Total de dinero ganado durante las últimas i partidas de las sesión con $i = \{1, 3, 5, 10, 20, T\}$ siendo T el total de partidas de la sesión.
20,23 26,29 32,35	*betCoins(i)*	Total de dinero apostado durante las últimas i partidas de las sesión con $i = \{1, 3, 5, 10, 20, T\}$ siendo T el total de partidas de la sesión.
21,24 27,30 33,36	*balanceSesion(i)*	Diferencia entre initalCoins(i) y endCoins(i).
37	*racha*	Número de partidas seguidas perdiendo antes de cerrar sesión ó 0 si termina ganando
38-42	*partidasExtraCoins*	Porcentaje de partidas perdidas durante las últimas i partidas de las sesión con $i = \{3, 5, 10, 20, T\}$
43	*cambioSlot*	Variable Booleana que indica si se ha producido un cambio de slot.
44	*cambioSesión*	Variable Booleana que indica si se ha producido un cambio de sesión.

Cuadro 4.4: Contenido de cada línea del *csv*.

Tras implementar la función *php* -sita en el Anexo-, el *csv* sobre el cual realizaremos el estudio queda como sigue:



	A	B	C	D	E	F	G	H	I	J	K	
1	numeroSesion	fechaSesion	horaIncio	horaFin	TiempoJugado	slotId	initialCoins	endCoins	numeroPartidas	jackPot	horaJackPot	tiempo
2	1	30-Sep-2015	08:09:46	08:19:44	598	21	7017294	6780334	99	0	0	
3	1	30-Sep-2015	08:20:10	08:37:58	1068	11	6730334	4667834	179	0	0	
4	1	30-Sep-2015	08:38:27	08:41:34	187	34	4567834	2527834	53	0	0	
5	1	30-Sep-2015	08:43:16	08:51:51	515	9	2520334	4907834	78	0	0	
6	2	01-Oct-2015	08:13:51	08:23:39	588	21	4176240	3737260	114	0	0	
7	2	01-Oct-2015	08:24:05	08:26:40	155	34	3687260	2772260	48	0	0	
8	2	01-Oct-2015	08:27:12	08:33:20	368	9	2769760	2527260	50	0	0	
9	2	01-Oct-2015	08:33:48	08:40:36	408	11	2477260	3019760	81	0	0	
10	3	01-Oct-2015	10:14:39	10:22:56	497	21	4210781	4380781	76	0	0	
11	3	01-Oct-2015	10:23:12	10:38:25	913	34	4330781	2767281	239	0	0	
12	3	01-Oct-2015	10:38:51	10:50:41	710	11	2718519	2019	141	0	0	
13	3	01-Oct-2015	10:51:18	10:51:22	4	21	219	219	3	0	0	
14	4	02-Oct-2015	07:02:48	07:07:38	290	21	2583698	2724238	56	0	0	
15	5	02-Oct-2015	07:50:17	08:01:54	697	11	2571377	4408877	93	0	0	
16	6	02-Oct-2015	11:02:19	11:07:32	313	21	4015511	4120511	48	0	0	
17	7	02-Oct-2015	20:19:19	20:25:57	398	7	4040159	4103159	53	0	0	
18	7	02-Oct-2015	20:26:23	20:26:42	19	11	4053159	4353159	6	0	0	
19	8	02-Oct-2015	20:38:58	20:39:12	14	31	4264026	4262226	6	0	0	
20	8	02-Oct-2015	20:39:41	20:55:48	967	34	4212226	4367226	202	0	0	
21	8	02-Oct-2015	20:56:11	21:06:42	631	11	4319058	6601558	144	0	0	
22	9	03-Oct-2015	07:34:54	07:37:41	167	24	6530634	6310754	32	0	0	
23	9	03-Oct-2015	07:38:38	07:46:19	461	11	6260754	5460754	89	0	0	

Figura 4.12: Extracto del *csv* generado para el sujeto1

Por tanto, ya estamos en disposición de realizar en análisis estadístico con el fin de extraer resultados válidos para el empresa.

4.3.4. Técnicas estadísticas

Veamos, primero, la matriz de correlaciones. En dicha matriz comprobaremos si se puede descartar alguna variable por ser combinacion lineal de otras facilitando, de este modo, el estudio.

Matriz de correlaciones: (sólo la parte relevante de dicha matriz)

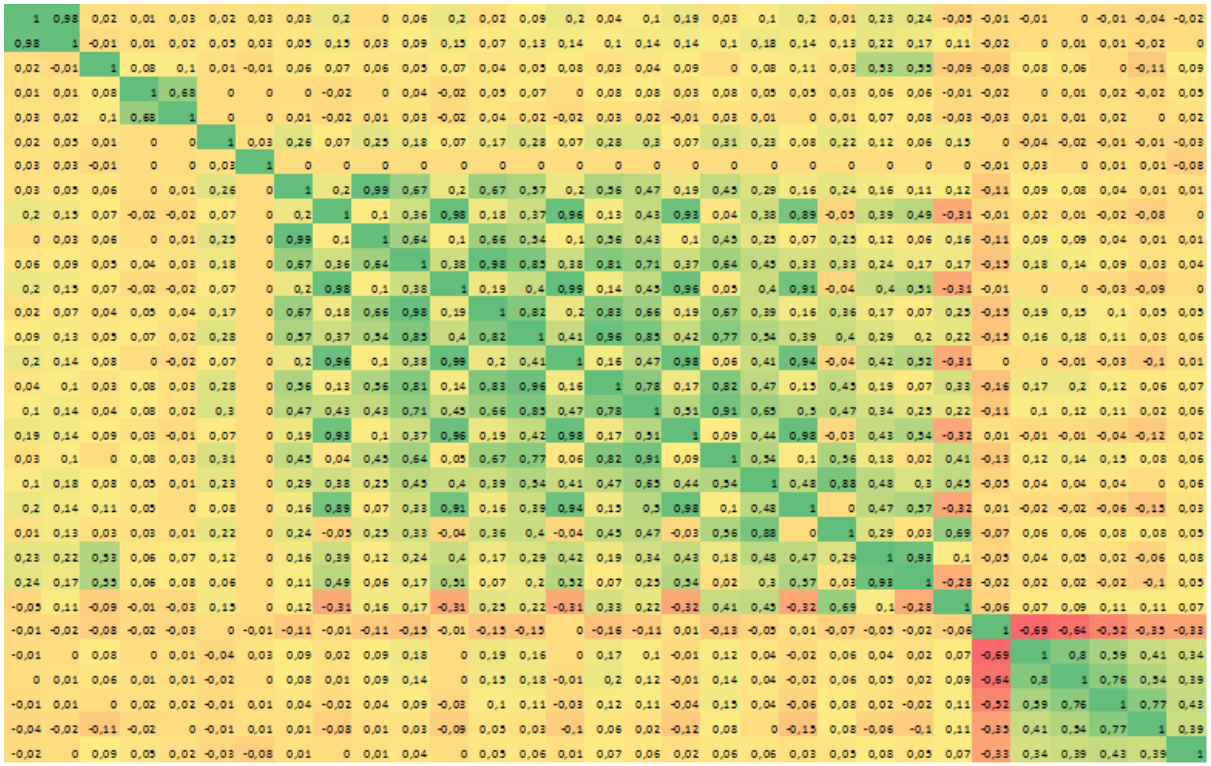


Figura 4.13: Matriz de correlaciones.

Conclusiones:

- Como cabía esperar, existe un correlacion directa y fuerte entre winCoins(i), betCoins(i) y balanceSesion(i) para todo *i*. Esto se debe a que la variable balanceSesion(i) la podemos expresar como combinacion lineal de winCoins(i) y betCoins(i) y, por este motivo, podemos omitir estas ultimas dos variables para todo *i*;
- Existe una correlacion positiva entre las variables %rachaT, %racha20, %racha10, %racha5 y %racha3 y, a la vez, todas ellas tienen correlacion negativa con la variable aleatoria racha. Este resultado extraño podria explicar el hábito que tiene el usuario de abandonar la sala con una tirada ganadora pese a tener un balance total de la sesión negativo.

Primera técnica empleada: Redes neuronales

```
# Leemos el csv:
datos = read.csv("sujeto1.csv", header = T)

# Cargamos el paquete: library(nnet)

# Nos quedamos sólo con las variables no redundantes y, por cada línea de
cambio de slot, otra que no lo sea (de forma idénticamente distribuida):
x = datos[,c(1,5:43)]
x = x[c(1:864,seq(865,5222,by=5)),]

# decay: se usa para evitar el sobreajuste.
# Entrenamiento de la red neuronal con size = tres capas ocultas:
red <- nnet (formula=x$cambioSlot~., data = x, size = 3, linout = TRUE)
# weights:  841
initial  value 673.865241
final  value 432.946086
converged

# Predicción. Se crea un data frame con las probabilidades reales
y estimadas. La red neuronal no genera buenas predicciones:
c=data.frame(real=x[,40], estimado=red$fitted.values)
c[1:8,]
   real  estimado
1     1 0.4959491
2     1 0.4959491
3     1 0.4959491
4     1 0.4959491
5     1 0.4959491
6     1 0.4959491
7     1 0.4959491
8     1 0.4959491
```

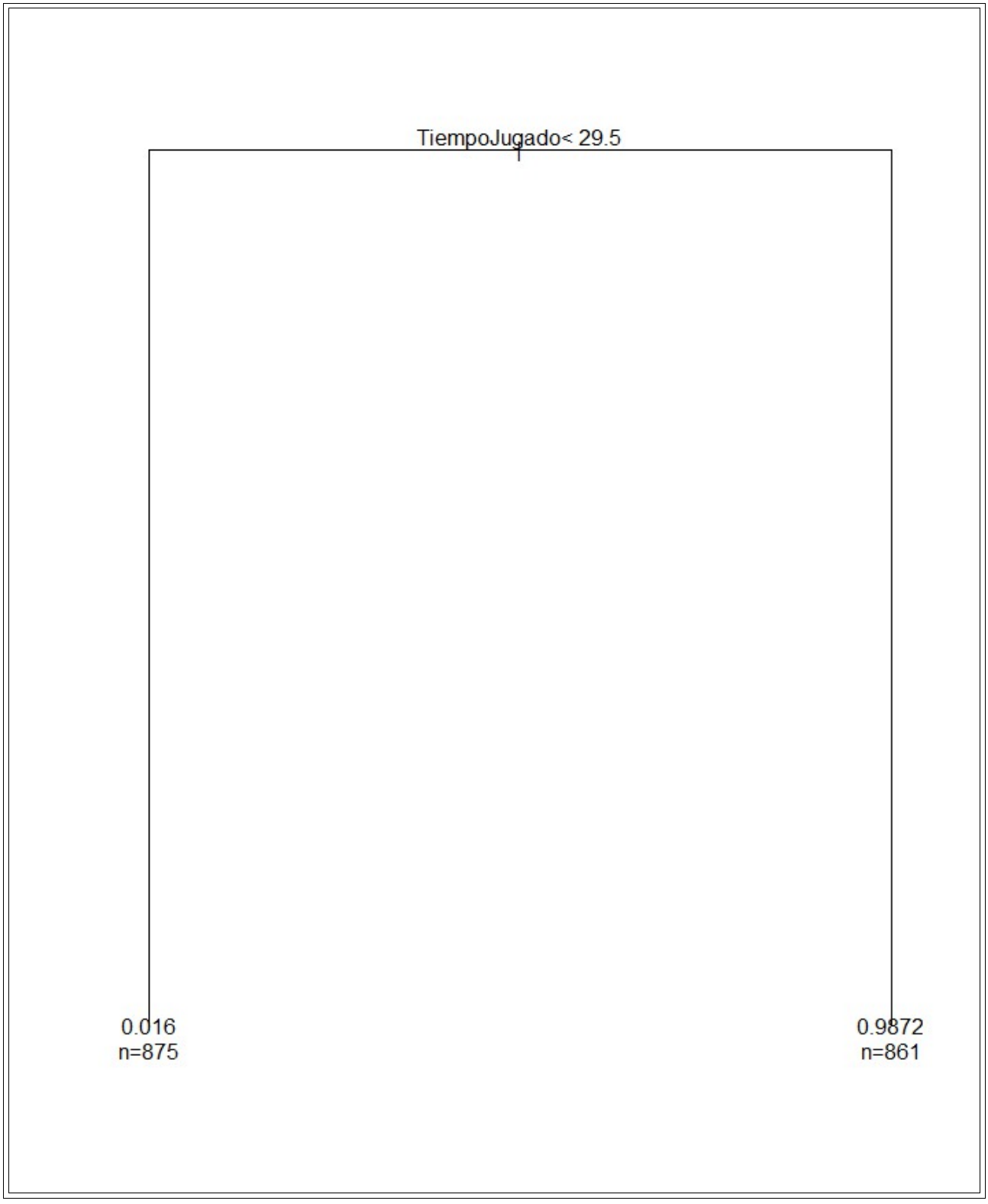
Nota: descartamos esta técnica estadística puesto que la red no diferencia cambios de slot de sesiones intermedias.

Segunda técnica empleada: Árboles de regresión

Primer intento:

```
# De la misma forma que antes, nos quedamos sólo con la parte del
data.frame que nos interesa para realizar el estudio:
x = datos[,c(1,5:9,13,15,16,18,21,24,27,30,33,36,40:42,43)]
x = x[c(1:864,seq(865,5222,by=5)),]

# Tenemos las siguientes variables:
names(x)
 [1] "numeroSesion"      "TiempoJugado"
 [3] "slotId"            "initialCoins"
 [5] "endCoins"          "numeroPartidas"
 [7] "levelCoins"        "tiempoTranscurridoHastaCierreLevelCoins"
 [9] "extraCoins"        "tiempoTranscurridoHastaCierreExtraCoins"
[11] "balanceSesion1"    "balanceSesion3"
[13] "balanceSesion5"    "balanceSesion10"
[15] "balanceSesion20"   "balanceSesionT"
[17] "X.racha10"         "X.racha20"
[19] "X.rachaT"          "cambioSlot"

# Intentamos deducir cambioSlot mediante el resto de variables
aleatorias:
arbol = rpart (cambioSlot ~ numeroSesion + TiempoJugado + slotId +
initialCoins + endCoins + numeroPartidas + levelCoins +
tiempoTranscurridoHastaCierreLevelCoins + extraCoins +
tiempoTranscurridoHastaCierreExtraCoins + balanceSesion1 + balanceSesion3
+ balanceSesion5 + balanceSesion10 + balanceSesion20 + balanceSesionT +
X.racha10 + X.racha20 + X.rachaT,  data=x)


# Resultado:
arbol
n= 1736

node), split, n, deviance, yval
        •  denotes terminal node
        •

1) root 1736 433.99080 0.4976959
  2) TiempoJugado< 29.5 875  13.77600 0.0160000 *
   3) TiempoJugado>=29.5 861  10.85947 0.9872242 *

# Obtenemos un árbol con un sólo nodo. Esto se debe a que la varibale
TiempoJugado es excesivamente influyente. Veámoslo:
> mean(datos[c(1:864),5])
[1] 550.5231
> mean(datos[c(865:5222),5])
[1] 35.62758
```

```
# Gráficamente, el árbol es como sigue:
plot(arbol); text(arbol, use.n=TRUE)
```



El anterior árbol pone de manifiesto una obviedad: si el usuario ha jugado menos de 29 segundos, seguirá jugando. No es válido, tampoco, este estudio.

Segundo intento:

```
# Realizamos el estudio anterior, descartando TiempoJugado como variable
explicativa.

arbol = rpart (cambioSlot ~ numeroSesion + slotId + initialCoins +
endCoins + numeroPartidas + levelCoins +
tiempoTranscurridoHastaCierreLevelCoins + extraCoins +
tiempoTranscurridoHastaCierreExtraCoins + balanceSesion1 + balanceSesion3
+ balanceSesion5 + balanceSesion10 + balanceSesion20 + balanceSesionT +
X.racha10 + X.racha20 + X.rachaT,  data=x)


# Resultado:
```

```
arbol
n= 1736

node), split, n, deviance, yval
      * denotes terminal node

 1) root 1736 433.9908000 0.49769590
   2) balanceSesion20< 151250 1308 317.4098000 0.41437310
     4) extraCoins< 725 1198 282.4307000 0.38063440
       8) slotId=alice-in-wonderland,arabian-nights,new-york-
nightlife,sweet-temptations 161   0.9937888 0.00621118 *
       9)
slotId=atlantis,bang,bigtimespay,buffalo,burningsevens,camelot,caribbean
-treasure,christmas,dragonfortunes,dualdiamond,egypt,fairy,farm,hansel-
and-gretel,little-red-riding-hood,lucky-
dragon,luckywheel,megaestelar,pet-a-
porter,quickhitplatinum,retro,sanvalentin,sherlock,threediamondsadvance,
timeslot,tumbling-diamonds,wolf 1037 255.3616000 0.43876570
        18) endCoins>=24571.5 947 227.5185000 0.40126720
          36) balanceSesionT>=-1588500 779 179.0757000 0.35815150
            72) slotId=atlantis,bang,camelot,christmas,hansel-and-
gretel,pet-a-porter,retro,sherlock,timeslot,tumbling-diamonds 445
92.0224700 0.29213480 *
            73) slotId=bigtimespay,buffalo,burningsevens,caribbean-
treasure,dragonfortunes,dualdiamond,egypt,fairy,farm,little-red-riding-
hood,lucky-
dragon,luckywheel,megaestelar,quickhitplatinum,sanvalentin,threediamonds
advance,wolf 334  82.5299400 0.44610780 *
          37) balanceSesionT< -1588500 168  40.2797600 0.60119050 *
        19) endCoins< 24571.5 90  12.5000000 0.83333330 *
     5) extraCoins>=725 110  18.7636400 0.78181820
      10) numeroPartidas< 32.5 20   0.9500000 0.05000000 *
      11) numeroPartidas>=32.5 90   4.7222220 0.94444440 *
   3) balanceSesion20>=151250 428  79.7476600 0.75233640 *
```

```
# Gráficamente:
```

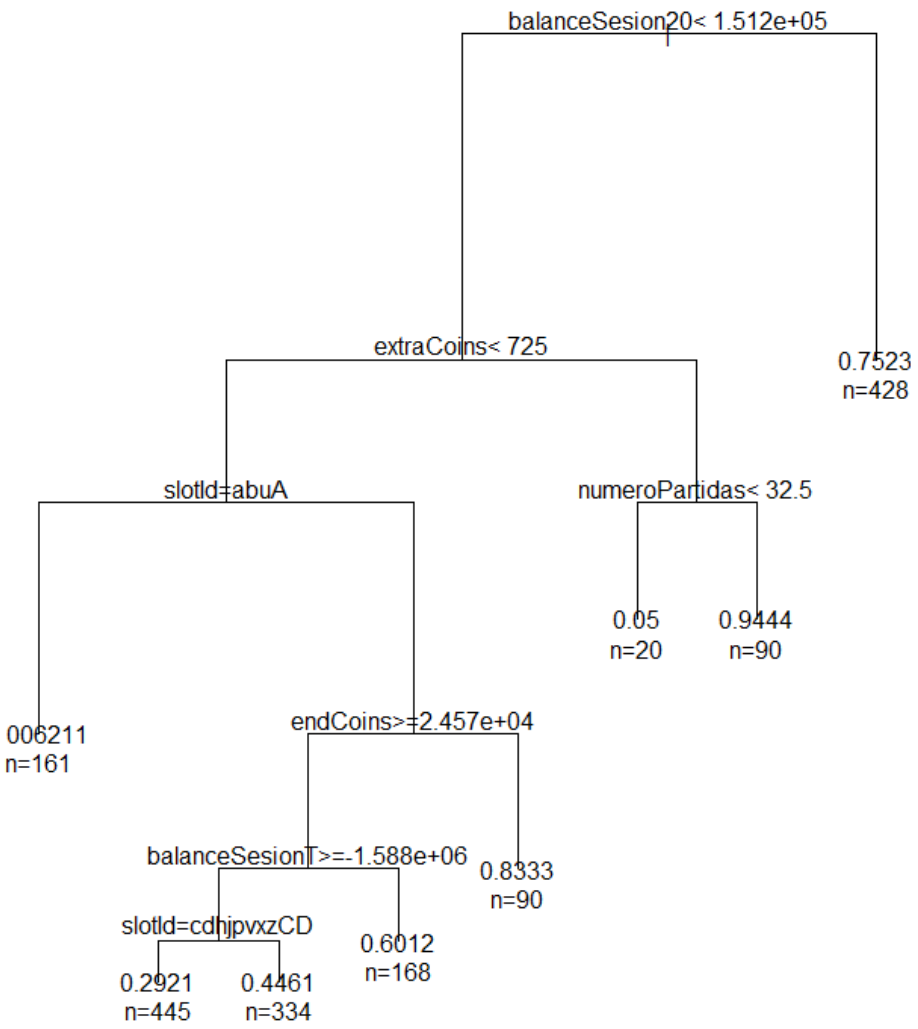


Figura 4.14: Árbol de regresión. Primer resutlado válido.

Esta técnica explica por qué un usuario cambia de slot con una bondad de ajuste cercana al 80 %. Es decir, en la mayoría de ocasiones predice si una línea es cambio de sesión o no y, además, describe qué eventualidades producen que esto suceda. Por tanto, tras probar las anteriores técnicas estadísticas y otras como regresión logística, redes bayesianas,... concluimos usar ésta para realizar el estudio de los sujetos.

Capítulo 5

Resultados

El resultado del estudio del primer usuario se detallará de forma exhaustiva. Del resto de usuarios, si bien el estudio se realizará de la misma forma, los resultados obtenidos se expondrán de forma breve.

5.1. Primer usuario

5.1.1. Información previa

- Sexo: mujer;
- Nacionalidad: francesa;
- Edad: 46;
- Plataforma de juego: Facebook;
- Característica principal: usuario que más dinero ha invertido en comprar *coins* para jugar en *Our.com*.

5.1.2. Análisis descriptivo

Leemos el archivo, almacenamos su contenido en datos y comprobamos que se ha leído correctamente:

```
datos = read.csv("sujeto1.csv", header = T, nrows = 900)
```

```
#Nombre de las columnas:
> names(datos)
 [1] "numeroSesion"
 [2] "fechaSesion"
 [3] "horaIncio"
...
[42] "X.rachaT"

#Dimensión de datos:
> dim(datos)
[1] 864  44

#¿Tenemos datos faltantes?
> complete.cases(datos)
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
...
 [856] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Efectivamente, datos es una matriz *(n x p)* en la cual el número de observaciones es $n = 864$ y el número de variables observadas es $p = 44$. Podemos concluir que el archivo se ha leído correctamente.

Veamos, para empezar, algunas conclusiones inmediatas que nos proporciona R:

Evolución del número de partidas por sesión:

Script en R:

```
i = 1 #Contador auxiliar
numeroPartidasPorSesion = rep(0, 408); #R necesita que inicialicemos el
vector, por tanto, creamos un vector con 408 posiciones (hay 408
sesiones). A todas ellas les asignamos el valor 0.

#Tras el siguiente bucle, el vector numeroPartidasPorSesion contendrá en
la posición i el total de partidas jugadas durante la sesión i. Veamos
cómo:
while ( i < 866 ) { #Recorremos las 806 filas del csv
     j = datos$numeroSesion[i];
     numeroPartidasPorSesion[j] = (numeroPartidasPorSesion[j] +
     datos$numeroPartidas[i]);
     i=i+1; }

#Calculamos su media:
mediaPartidasPorSesion = rep(mean(numeroPartidasPorSesion),408);

#Gráficos:
plot(numeroPartidasPorSesion, col="blue", xlab='Número de Sesión')
lines(mediaPartidasPorSesion, col="red") #Dibujamos, también, la media.
title(main="Evolución de Número Partidas (Por sesión)", col.main="red",
font.main=4) #Añadimos un título en color rojo y letra cursiva/negrita.
```

Veamos el resultado obtenido:

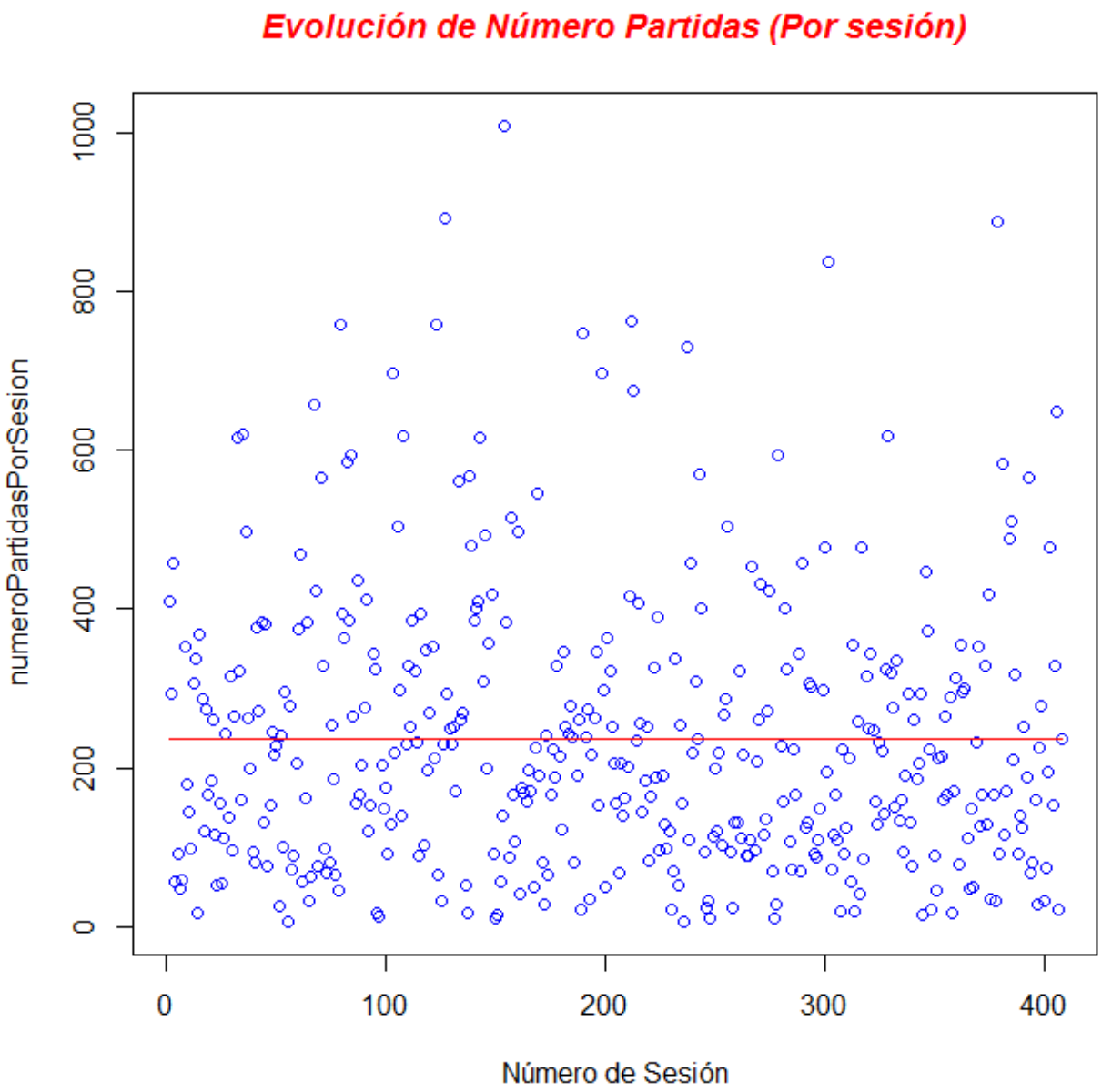


Figura 5.1: Número de partidas o spins por sesión

Podemos ver que la variable número de partidas por sesión tiene una varianza grande (nube de puntos sin patrones). Parece que dicha variable podría seguir una distribución normal con media = 237.1152 y una varianza parecida.

Tabla de Frecuencias de slotId:

De la siguiente imagen se deduce que el *sujeto1* ha jugado a un total de *31 slots* y que su *favorito* es *timeslot:* ha jugado un total de *240 veces* en los seis meses que dura el estudio:

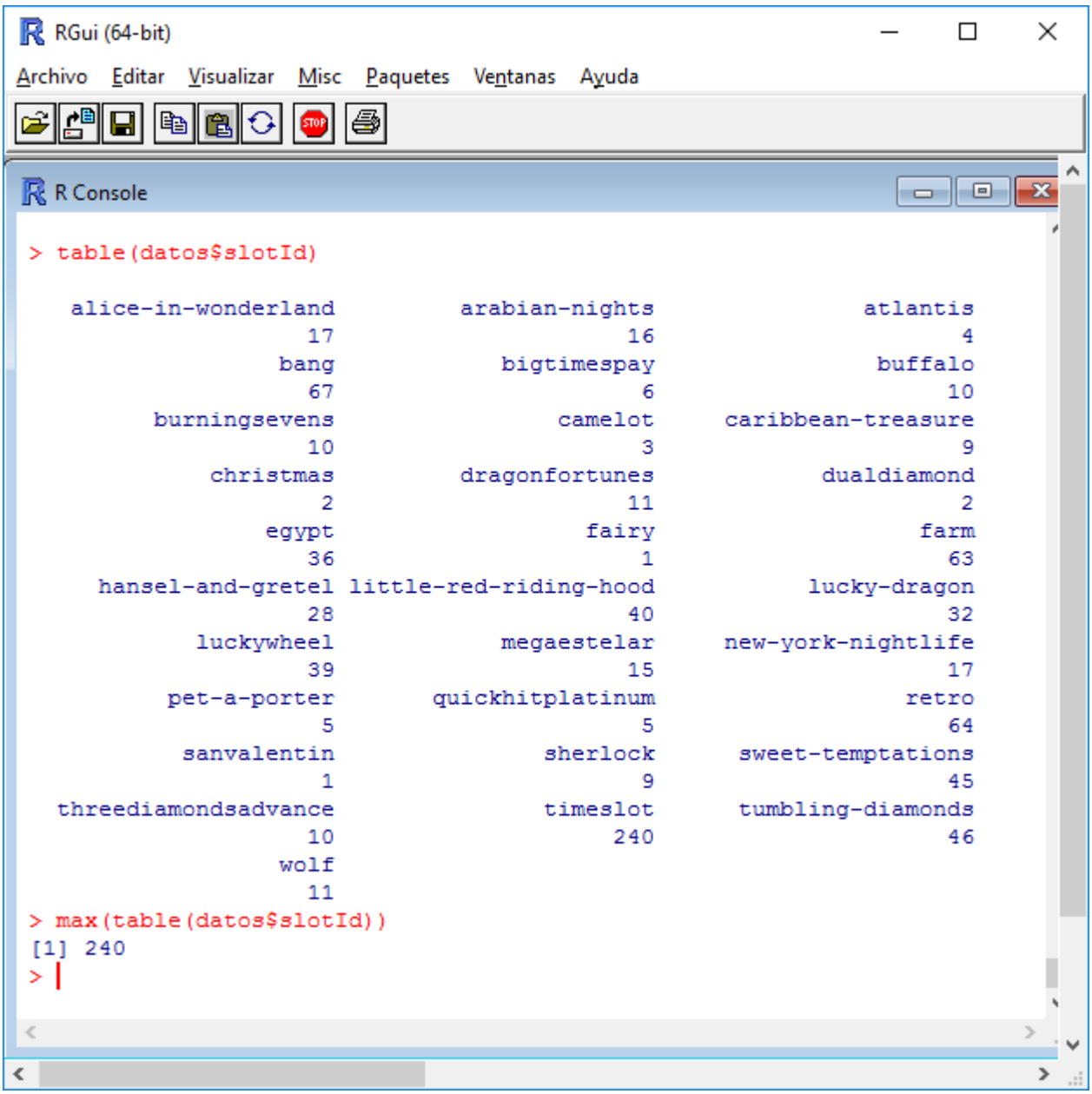


Figura 5.2: Tabla de frecuencias *slotId*.

Balance Total de la Sesión Versus Tiempo Jugado:

```
mediaTiempoJugado = mean(datos$TiempoJugado)
mediaBalanceSesionT = rep(mean(datos$balanceSesionT),5000)

#Gráficas:
#Aquí, en negro, dibujamos los puntos de la gráfica:
plot(datos$TiempoJugado, datos$balanceSesionT, ylim=c(-17000000,
15000000), xlab = 'Tiempo Jugado (seg.)', ylab = 'Balance de la Sesión
(coins)')

# Dibujamos las respectivas medias:
lines(mediaBalanceSesionT, col="red")
```

```
lines(x=c(mediaTiempoJugado,mediaTiempoJugado), y=c(-17000000,15000000),
col="blue")
title(main="Balance de la sesión VS Tiempo Jugado", col.main="red",
font.main=4) #Añadimos un título en color rojo y letra cursiva/negrita.
```

- La mediaTiempoJugado es de 550.5231 segundos, aproximadamente, 9 minutos y 6 segundos.
- La mediaBalanceSesion es de -152.539,5 coins.

Veamos el gráfico:

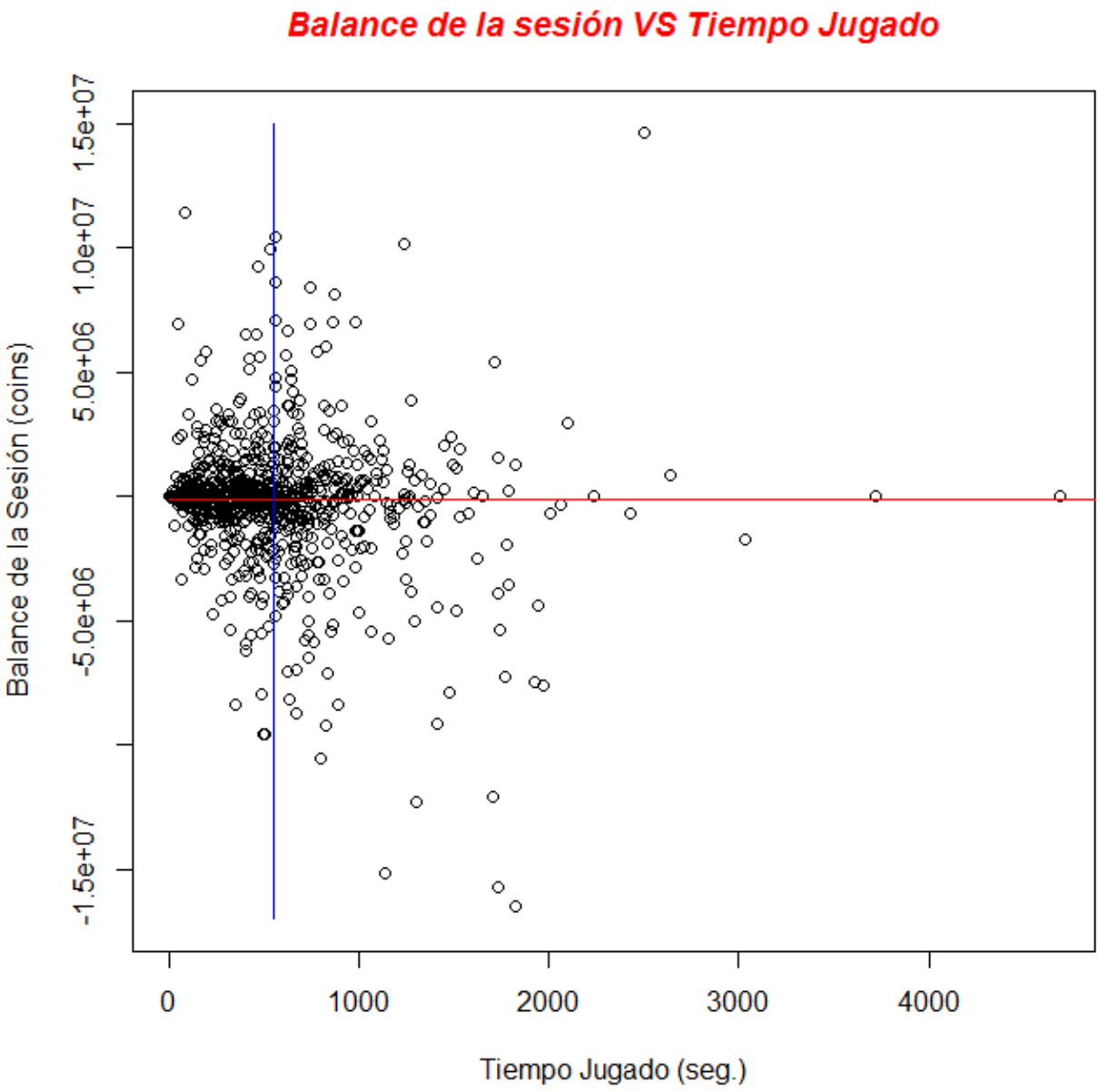


Figura 5.3: A medida que la sesión se dilata en el tiempo, el balance de la sesión se hace negativo.

Veamos, ahora, el mismo gráfico ampliando la zona de la nube de puntos:

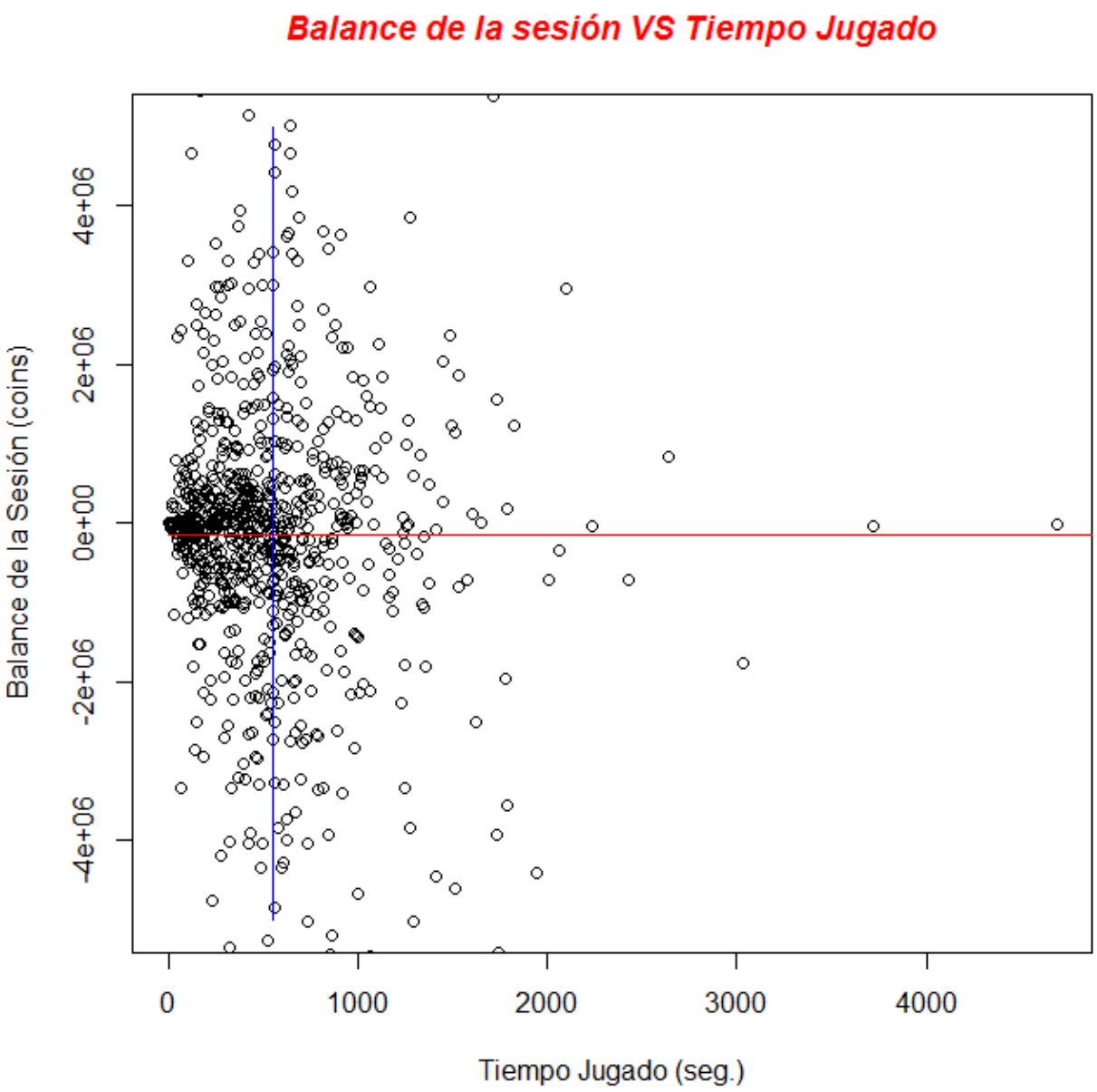


Figura 5.4: Ampliamos la Figura 5.3.

Conlusiones:

- hasta los 550 segundos (línea azul) abandona el s*lot* tanto con balance positivo como negativo;
- si la partida dura entre 550 - 2000 segundos, abandona un mayor número de veces el sl*ot* con balance negativo, pudiendo ser ésta la causa;
- el *sujeto1* sólo juega más de 2000 segundos en un slot cuando el balance NO es negativo. Esto se debe a que ha logrado un premio *considerable*.

Notar que estas conclusiones son previas al estudio (estamos, sólo, describiendo la muestra). Después habrá que corroborar lo que aseveramos a priori mediante técnicas estadísticas (y no descriptivas).

¿Cómo afecta al comportamiento del sujeto ganar FreeSpins?

```
#En 154 ocasiones ha ganado tiradas gratis:
> sum(datos$extraCoins!=0)
[1] 154

#Creamos una nueva tabla de datos en la que sólo están dichas 154
#líneas:
datos2=datos[datos$extraCoins!=0,]

#De las 154 veces en las cuales el sujeto ha ganado tiradas gratis, en
#64 ocasiones ha seguido jugando y ¡en 90 ha abandonado el SLOT!
table(datos2$tiempoTranscurridoHastaCierreExtraCoins==0)

     FALSE TRUE
     64              90

#De media tarda sólo 2 minutos en abandonar el SLOT desde que gana el
#premio:
> mean(datos2$tiempoTranscurridoHastaCierreExtraCoins)
[1] 149.8117

#Veamos una tabla de frecuencias:
> table(datos2$tiempoTranscurridoHastaCierreExtraCoins)

Segundos que tarda en abandonar la sala: 0  73 86 87 99 115 125 136 140
Número de ocurrencias:                   90  1  1  1  2   1   1   1   1
141 143 146 151 180 186 192 193 194 204 210 215 223 230 238 239 246 248
  1   2   1   2   1   2   1   1   1   1   1   1   1   1   1   1   1   2
252 257 321 328 331 334 342 352 376 378 380 382 402 408 465 478 479 490
  1   1   1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   1
513 517 520 542 554 569 648 721 793 827 871 934 1065 1647
  1   1   1   1   1   1   1   1   1   1   1   1    1    1
```

Para ver este resultado gráficamente:

```
mediaTiempoTranscurridoHastaCierre =
rep(mean(datos2$tiempoTranscurridoHastaCierreExtraCoins), 154)
plot(datos2$tiempoTranscurridoHastaCierreExtraCoins, xlab = 'Número de
Partida', ylab = 'Tiempo transcurrido hasta cierre (seg)')
lines(mediaTiempoTranscurridoHastaCierre, col="red")
title(main="Tiempo transcurrido desde FreeSpins hasta Cierre",
col.main="red", font.main=4) #Añadimos un título en color rojo y letra
cursiva/negrita.
```

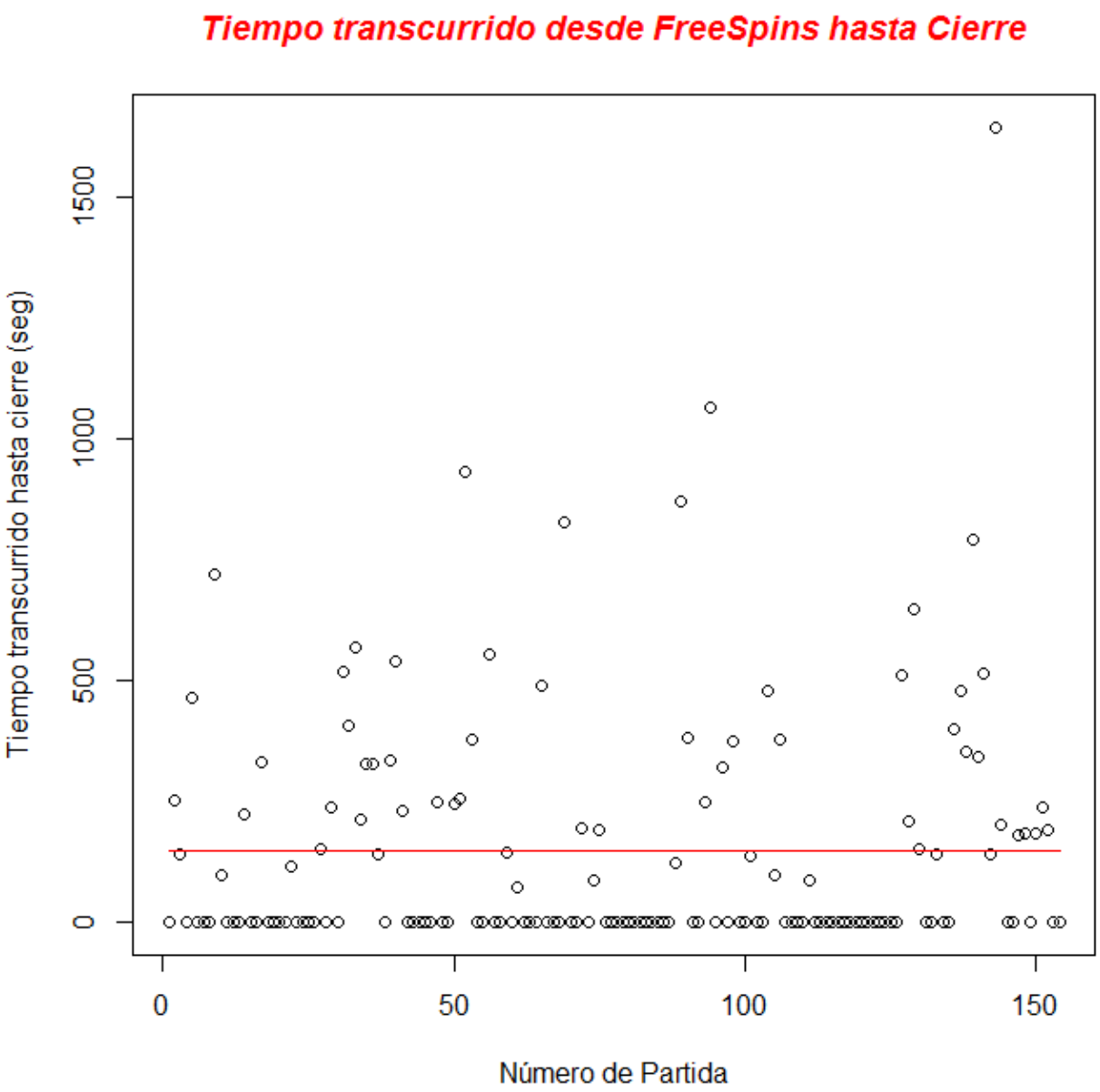


Figura 5.5: Tiempo transcurrido desde que el usuario consigue el premio hasta que se produce el cambio de slot.

Lo primero que podemos observar en esta gráfica es que hay datos atípicos (mayores de 750 segundos) que pueden hacer que perdamos información importante. Repitamos el estudio sin dichos valores:

```
#Quitamos esos seis datos atípicos:
> datos3=datos2[datos2$tiempoTranscurridoHastaCierreExtraCoins<600,]
#Vemos como, quitando sólo 6 valores atípicos, la media desciende hasta
#los 106.6 segundos poniendo aún más de manifiesto que cuando el sujeto
#gana FreeSpins, cambia de Slot o cierra sesión
> mean(datos3$tiempoTranscurridoHastaCierreExtraCoins)
[1] 106.6096
```

Veámoslo gráficamente:

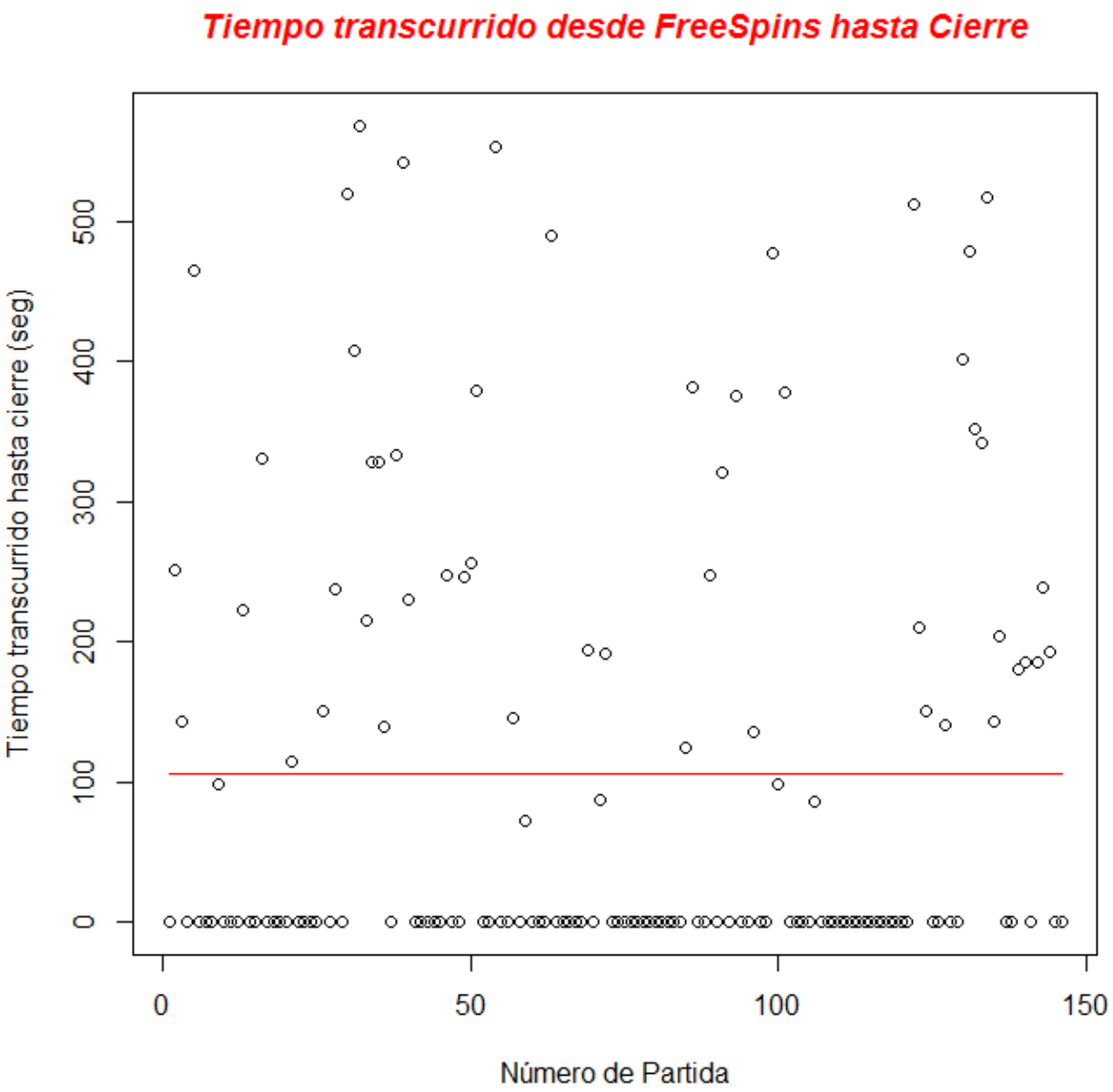


Figura 5.6: Obviamos los datos atípicos y repetimos la Figura 5.5.

Conclusiones:

- tras ganar tiradas gratis, el usuario en más de la mitad de las ocasiones cambia de slot o cierra sesión;
- en las ocasiones en las que sigue jugando, lo hace durante un período relativamente largo de tiempo.

¿Cómo afecta al comportamiento del sujeto ganar el premio por cambio de nivel?

Siguiendo el proceso anterior tenemos que:

```
datos4=datos[datos$levelCoins!=0,]
#En 23 ocasiones ha cambiado de nivel:
> dim(datos4)
[1] 23 42
mediaTiempoTranscurridoHastaCierre =
rep(mean(datos4$tiempoTranscurridoHastaCierreLevelCoins),23)
plot(datos4$tiempoTranscurridoHastaCierreLevelCoins, xlab = 'Número de
Partida', ylab = 'Tiempo transcurrido hasta cierre (seg)')
lines(mediaTiempoTranscurridoHastaCierre, col="red")
title(main="Tiempo transcurrido desde CambioNivel hasta Cierre",
col.main="red", font.main=4)
```

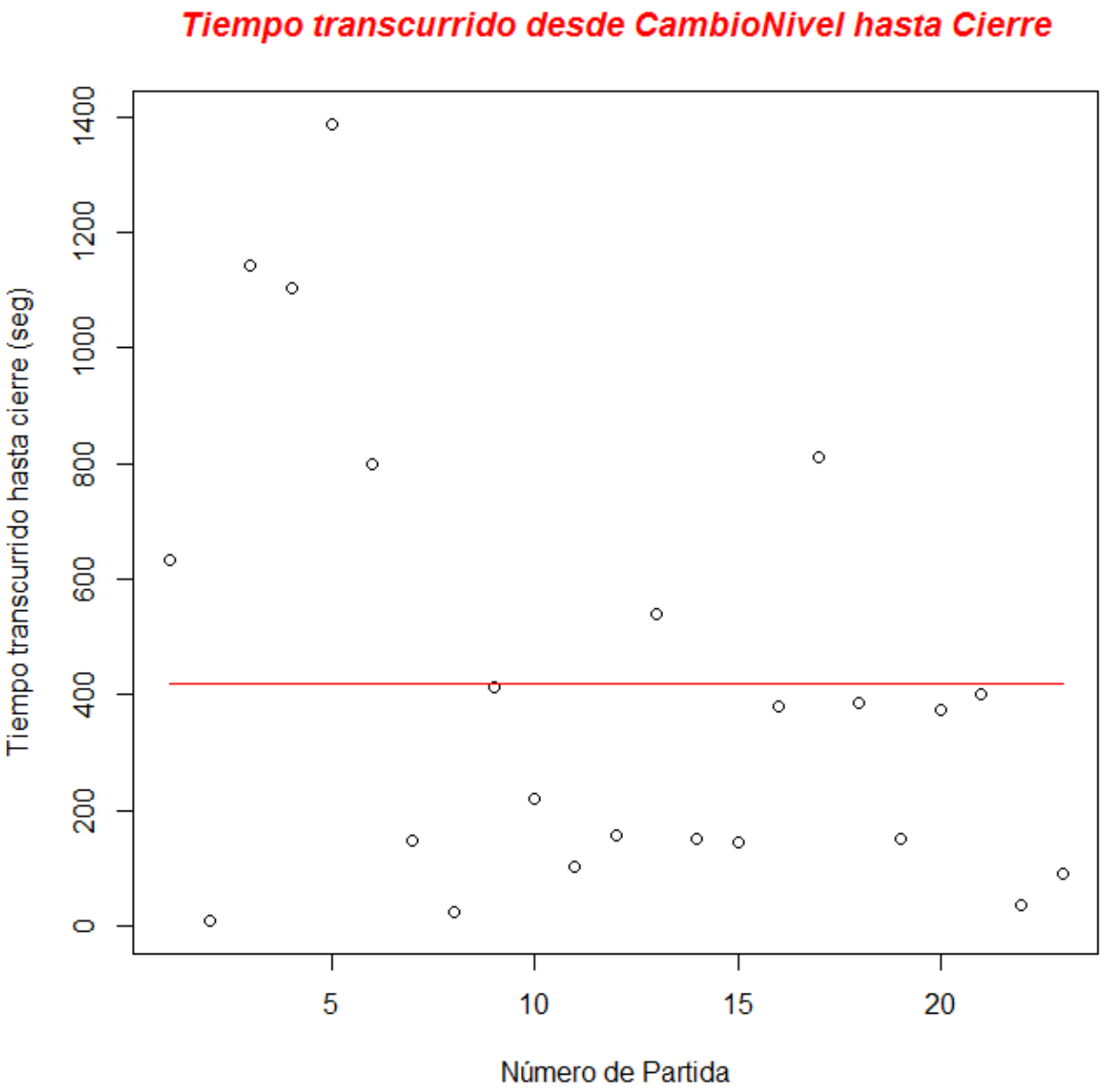


Figura 5.7: Tiempo transcurrido desde que el usuario aumenta de nivel hasta que se produce el cambio de slot o cierre de sesión.

Conclusión: podemos ver como el sujeto de estudio tiene interés por cambiar de nivel e intenta jugar hasta conseguirlo. De hecho, varias veces, tras lograrlo cambia de slot o cierra la sesión. Sin embargo, esto no es tan determinante como en el caso anterior y, en otras ocasiones, sigue jugando hasta más de 10 minutos. Por tanto, podemos aseverar que es más probable que el usuario cierre el slot si consigue *freeSpins* que si cambia de nivel.

Análisis de la variabe racha:

Recordemos que *racha* es una variable entera en la que se almacena el número de partidas perdidas de forma consecutiva antes de cerrar sesión ó 0 si el usuario termina la sesión ganando. Veamos la tabla de frecuencias de dicha variable:

```
> table(datos$racha)
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
334 235 117  66  33  23  15  16   5   4   8   1   3   2   2
```

Podemos inferir que el usuario tiende a cerrar sesión cuando en las últimas partidas el retornado (es decir, la cantidad de coins que recupera o gana tras la tirada) es positivo ya que, en 569 de las 864 sesiones abiertas, ha cambiado de slot o cerrado la sesión cuando ha conseguido un premio en la última y/o penúltima partida. Veámoslo gráficamente:

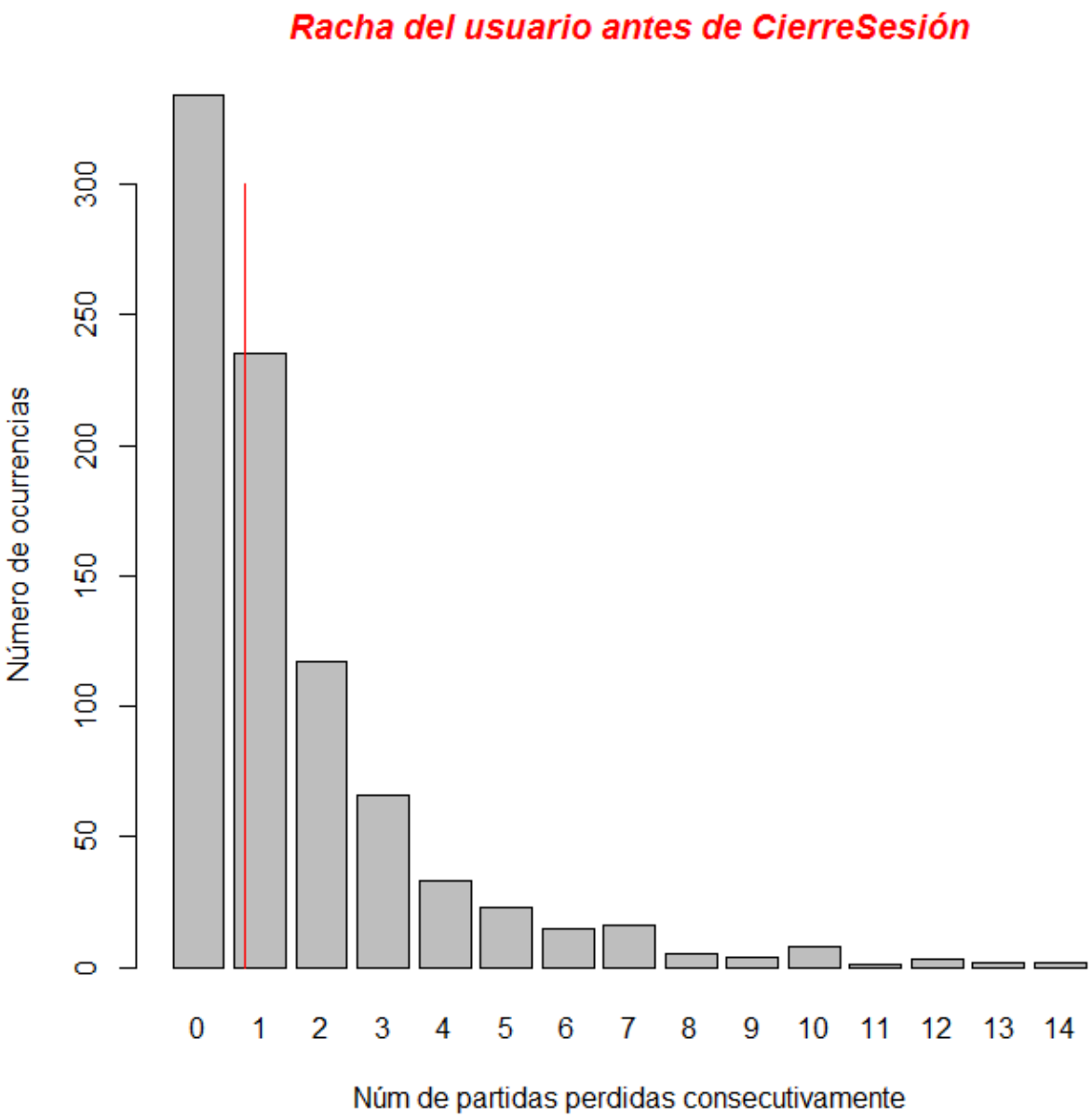


Figura 5.8: Número de partidas perdidas antes de cambiar de slot.

5.1.3. Resultados análisis estadístico

Recordamos la representación gráfica del árbol de regresion para este usuario:

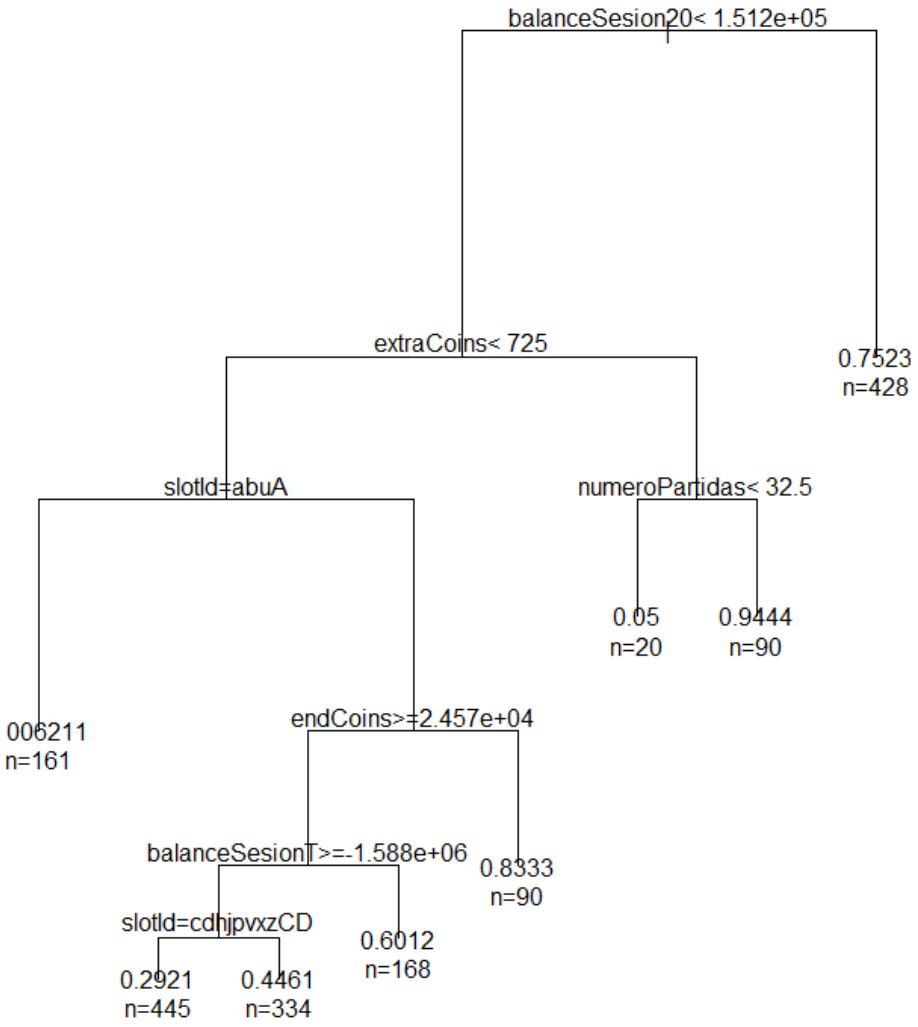


Figura 5.1: Árbol de regresión. Primer resutlado válido.

Explicación del árbol y, por ende, de las causas de abandono del slot por parte del usuario

Si el balance de la sesión en las últimas 20 tiradas es mayor que 151250 coins, el usuario cambia de slot o cierra sesión con probabilidad 0.75233640. Esto pone de manifiesto algo que ya habíamos visto mediante la estadística descriptiva: **si el balance de la sesión es positivo, abandona el slot**. En caso contrario, sigue jugando.

Si el balance de la sesión en las últimas 20 tiradas es menor que 151250 coins y el usuario gana *freeSpins*, sólo se queda jugando si lleva 32 tiradas o menos. En caso contrario, sigue jugando. Es decir, **cuando el usuario gana tiradas gratis abandona la sala** casi siempre -sólo permanece en la misma cuando lleva poco tiempo jugado y, probablemente, sienta que no ha jugado suficiente-. Esto concuerda con las conclusiones extraídas mediante técnicas de estadística descriptiva.

Suponiendo, ahora, que en las últimas 20 tiradas el balance del usuario es menor que 151250 coins y que dicho usuario no ha ganado *freeSpins*, la variable que determina si permanece, o no, en el slot es el slotId:

1. Si slotId es alice-in-wonderland, arabian-nights, new-york-nightlife o sweettemptations, abandona.

2. Si, por el contrario, slotId es atlantis, bang, bigtimespay, buffalo, burningsevens, camelot, caribbean-treasure, christmas, dragonfortunes, dualdiamond, egypt, fairy, farm, hansel-and-gretel,l ittle-red-riding-hood, lucky-dragon, luckywheel, megaestelar, pet-a-porter, quick-hitplatinum, retro, sanvalentin, sherlock, threediamondsadvance, timeslot, tumbling-diamonds o wolf, sigue jugando.

Esto se debe a que en los slots del *grupo 1* se ganan los *freeSpins* por acumulación de tiradas y, por tanto, si el balance del usuario es negativo y dicho usuario prevé que no va a poder llegar a acumular la cantidad suficiente para ganar tiradas gratis, abandona el slot. En los slots del *grupo 2*, si el usuario tiene un saldo actual lo suficientemente grande (24571 coins o más), abandona el slot. En caso contrario, sigue jugando, poniendo de manifiesto otra conclusión de la estadística descriptiva: a dicho usuario **no le gusta abandonar cuando va perdiendo**.

Finalmente, hay otras causas menores de abandono como, por ejemplo, que el balance de la sesión sea extremandamente bajo (-1 588 500 coins o menor).

Conslusiones:

- Si el usuario tiene un balance de la sesión positivo (mayor que 150.000 coins), cambia de slot o cierra sesión.

- El objetivo final (casi único) del usuario es lograr ganar *freeSpins*. Generalmente, lo hace en slots en los cuales se puede lograr de forma acumulativa y, tras lograrlo, abandona el slot.

- Si el balance de la sesión no es bueno y está jugando en slots en los cuales se consiguen los freeSpins de forma acumulativa, abandona, tal vez, previendopertencen que no lo va a lograr.
- En el resto de slots y con balance negativo, sigue jugando. Si continúa perdiendo y/o tiene un saldo actual bajo, sigue jugando de forma cada vez más constante y tenaz.

5.2. Segundo usuario

5.2.1. Información previa

- Sexo: mujer;
- Nacionalidad: francesa;
- Edad: 46;
- Plataforma de juego: Facebook;
- Ha comprado un total de 103 millones de coins en los últimos 6 meses;
- No tiene movimientos desde el 21 de septiembre hasta 27 de octubre de 2015;
- De la misma forma que el primer usuario, juega en todos los slots -especialmente, *timeslot*-;
- Característica principal: Es el primer usuario con otra cuenta y otro pseudónimo. El objetivo es comprobar que ambas cuentas pertenecen al mismo usuario analizando la conducta de juego.

5.2.2. Resultados análisis estadístico

Abandona el slot cuando:

- 71 % slots con freeSpins acumulativos:
  - 40.4 % se queda sin monedas (8022 coins o menos);
  - 30.6 % balance positivo en las últimas 10 tiradas (450.000 coins o más) o extraCoins;
- 12 % resto de Slots:
  - 11 % se queda sin monedas (1050 coins);
  - 1 % balance positivo en las últimas 10 tiradas (203.000 coins);
- 17 % sin clasificar o mal clasificado.

La variable balanceT no influye y podemos corroborar que ambos usuarios tienen la misma conducta de juego.

5.3. Tercer usuario

5.3.1. Información previa

- Sexo: mujer;
- Nacionalidad: española;
- Edad: 42;
- Plataforma de juego: Facebook;
- Ha comprado un total de 51.4 millones de coins en los últimos 6 meses;
- Juega en todos los slots -especialmente, *new-york-life*-;
- En un 30 % de las ocasiones ha ganado extraCoins;
- La cantidad de total que apuesta por tirada ha variado mucho y ha fluctuado periódicamente;
- Todas las apuestas elevadas las ha hecho en *bang*, *new-york-life* y *arabian nights*;
- Característica principal: No se queda sin monedas. Este es un comportamiento atípico.

5.3.2. Resultados análisis estadístico

Abandona el slot cuando:

- 31 % tras ganar freeSpins;
- 35 % new-york-life:
  - 26 % balance total de la sesión negativo (70.000 coins o menos);
  - 9 % tiene una última tirada muy buena (retornado de 24.000 coins o más);
- 20 % resto de slots:
  - 12 % balance total de la sesión negativo (30.000 coins o menos);
  - 8 % en las últimas 5 tiradas el retornados ha sido 0;
- 14 % sin clasificar o mal clasificado.

5.4. Cuarto usuario

5.4.1. Información previa

- Sexo: mujer;

- Nacionalidad: canada;
- Edad: 80;
- Plataforma de juego: Facebook;
- Ha comprado un total de 45.6 millones de coins en los últimos 6 meses;
- Juega en todos los slots -especialmente, *bang*-;
- Es más fácil explicar cuando sigue jugando;
- Característica principal: apuestas altas y sesiones largas.

5.4.2. Resultados análisis estadístico

Abandona el slot cuando:

- 61 % se queda sin monedas (5022 coins o menos);
- 22 % balance negativo en las últimas 10 tiradas (-53.000 coins o menos)
- 7 % balance total de la sesión muy negativo (-445.000 coins o menos);
- 10 % sin clasificar o mal clasificado.

5.5. Quinto usuario

5.5.1. Información previa

- Sexo: mujer;
- Nacionalidad: brasileña;
- Edad: 35;
- Plataforma de juego: Facebook;
- Ha comprado un total de 35.8 millones de coins en los últimos 6 meses;
- Sólo juega en *sherlock*. No prueba otros slots;
- Nunca ha ganado extraCoins;
- Característica principal: El 71.1 % de las veces dejar de jugar porque se quedan sin monedas.

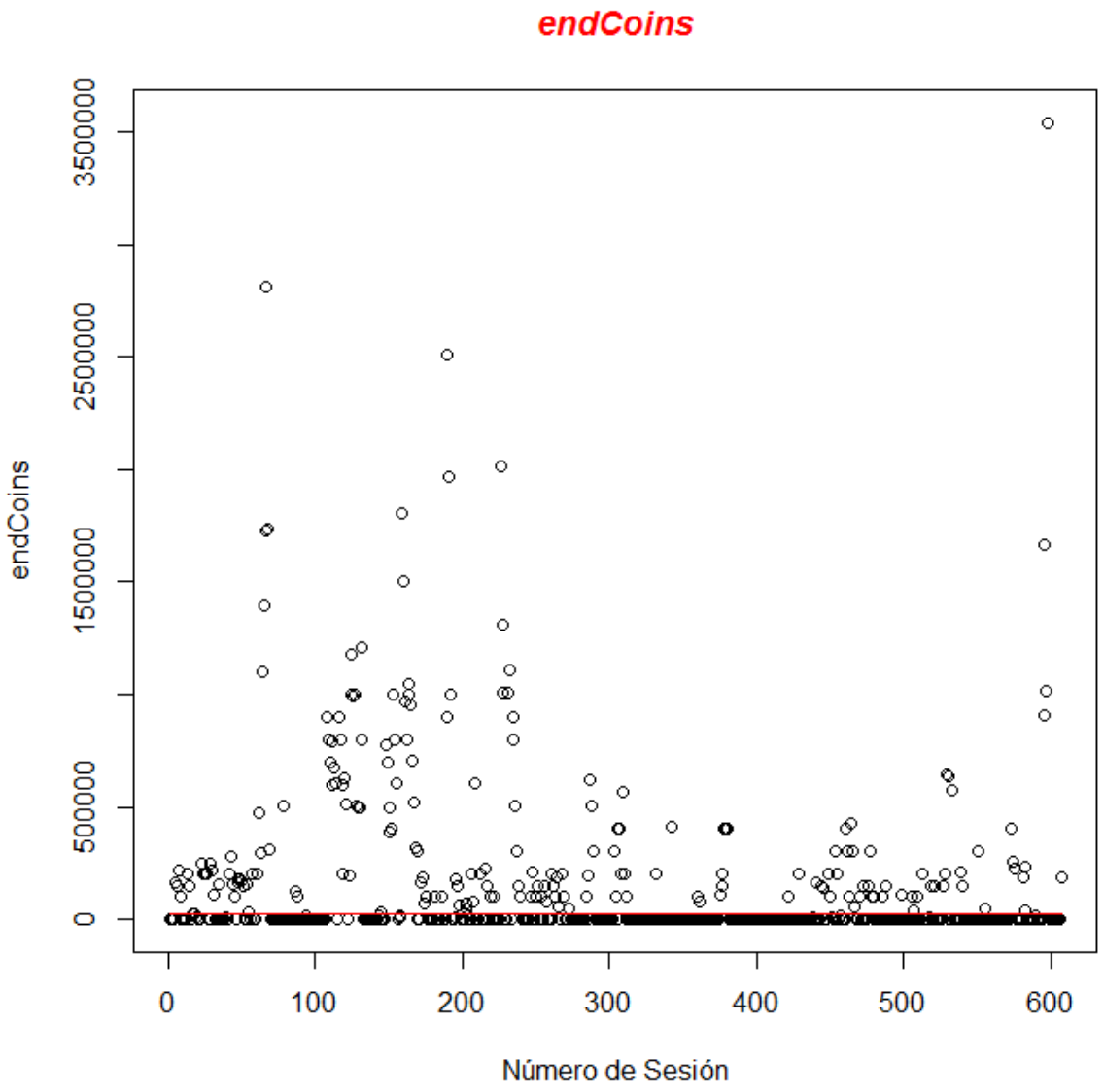


Figura 5.2: Vemos como la variable endCoins está cerca de 0 al final de cada sesión.

5.5.2. Resultados análisis estadístico

Abandona el slot cuando:

- 71.1 % se queda sin monedas (1088 coins o menos);
- 16.3 % balance total de la sesión negativo (-48.200 coins o menos);
- 12.6 % Sin clasificar o mal clasificado.

endCoins es la variable más influyente y, además, La variable *slotId* no influye.

5.6. Sexto usuario

5.6.1. Información previa

- Sexo: mujer;
- Nacionalidad: serbia;
- Edad: 56;
- Plataforma de juego: Facebook;
- Ha comprado un total de 35.5 millones de coins en los últimos 6 meses;
- Sesiones muy largas;
- Juega en slots en los cuales se gana freeSpins de forma acumulativa y torneos *Chili*;
- Característica principal: ha aumentado de forma considerable la compra de monedas.

5.6.2. Resultados análisis estadístico

Abandona el slot cuando:

- 49 % slots con torneos *Chili*:
  - 30 % se queda sin monedas (1012 coins o menos);
  - 19 % balance de la sesion muy negativo (-153.000 coins o menos);
- 30 % Slots con freeSpins acumulativo:
  - 16 % se queda sin monedas (809 coins o menos);
  - 14 % gana freeSpins;
- 11 % sin clasificar o mal clasificado.

5.7. Séptimo usuario

5.7.1. Información previa

- Sexo: hombre;
- Nacionalidad: francesa;
- Edad: 41;
- Plataforma de juego: Facebook;

- Juega, de media, 15 minutos en cada slot y algo más de 30 minutos en cada sesión;
- Ha ganado 12 jackpots en *farm* y 1 en *buffalo*;
- Característica principal: actualmente, es el usuario que más está comprando en *Our.com*.

5.7.2. Resultados análisis estadístico

Abandona el slot cuando:

- 35 % slots con torneos:
  - 5 % se queda sin monedas (menos 522 coins);
  - 15 % con una balance en las últimas 20 partidas de +174 500 o mayor;
  - 15 % con un balance de la sesión de -691 000 o menor;
- 52 % slots con jackpot:
  - 29 % se queda sin monedas: 13 680 coins o menos;
  - 23 % con un balance de la sesión de -1 202 500 coins o menos;
- 13 % sin clasificar o mal clasificado.

5.8. Octavo usuario

5.8.1. Información previa

- Sexo: mujer;
- Nacionalidad: española;
- Edad: 44;
- Plataforma de juego: Facebook;
- Juega, de media, 11 minutos por slot y sesión.;
- Característica principal: compra packs diariamente pero no superiores a 30.000 monedas. Sin embargo, está en el top histórico de compras en la empresa.

5.8.2. Resultados análisis estadístico

Abandona el slot cuando:

- 72 % retro:

    - 56 % se queda sin monedas (menos 540 coins);
    - 9 % balance en las últimas 20 partidas de +20 000 o mayor (incluye extraCoins);
    - 7 % balance de la sesión de -25 500 o menor;
- 20 % resto de slots:
    - 19 % se queda sin monedas: 146 coins o menos;
    - 1 % con un balance de la sesión de + 13 500 coins o más;
- 8 % sin clasificar o mal clasificado.

5.9. Noveno usuario

5.9.1. Información previa

- Sexo: hombre;
- Nacionalidad: estadounidense;
- Edad: 52;
- Plataforma de juego: Facebook;
- Tiene valores de initalCoins y extraCoins siempre entorno a los 3 000 000 coins;
- Intenta moderar mucho lo que juega y cómo lo hace para tener siempre un saldo alto;
- En 200 ocasiones (19 %) ha ganado **extraCoins**. Abandona la partida inmediatamente;
- Ha jugado en todos los slots;
- Característica principal: el comportamiento de los jugadores estadounidenses es distinto al europeo. El objetivo es entender mejor a este tipo de jugadores para abrir el mercado.

5.9.2. Resultados análisis estadístico

Abandona el slot cuando:

- 55 % balance en las últimas 20 tiradas es +550 000 o mayor.
- Si no ha tenido una buena racha:
    - 18 % con un balance de las sesión de -385 500 coins o menor;
    - 14 % con un balance de las sesión de -95 500 coins o menor;
- 13 % sin clasificar o mal clasificado.

5.10. Décimo usuario

5.10.1. Información previa

- Sexo: mujer;
- Nacionalidad: española;
- Edad: 37;
- Plataforma de juego: Facebook;
- Juega sólo en un slot por sesión;
- El 80 % de las veces juega a *bang*;
- Característica principal: juega estacionalmente, es decir, compra gran cantidad de monedas durante meses y, después, deja de jugar. Ampliamos el estudio: 01-May-15 a 30-Apr-16. Los meses en los que más ha jugado: mayo, junio, julio, septiembre y, sobre todo, agosto.

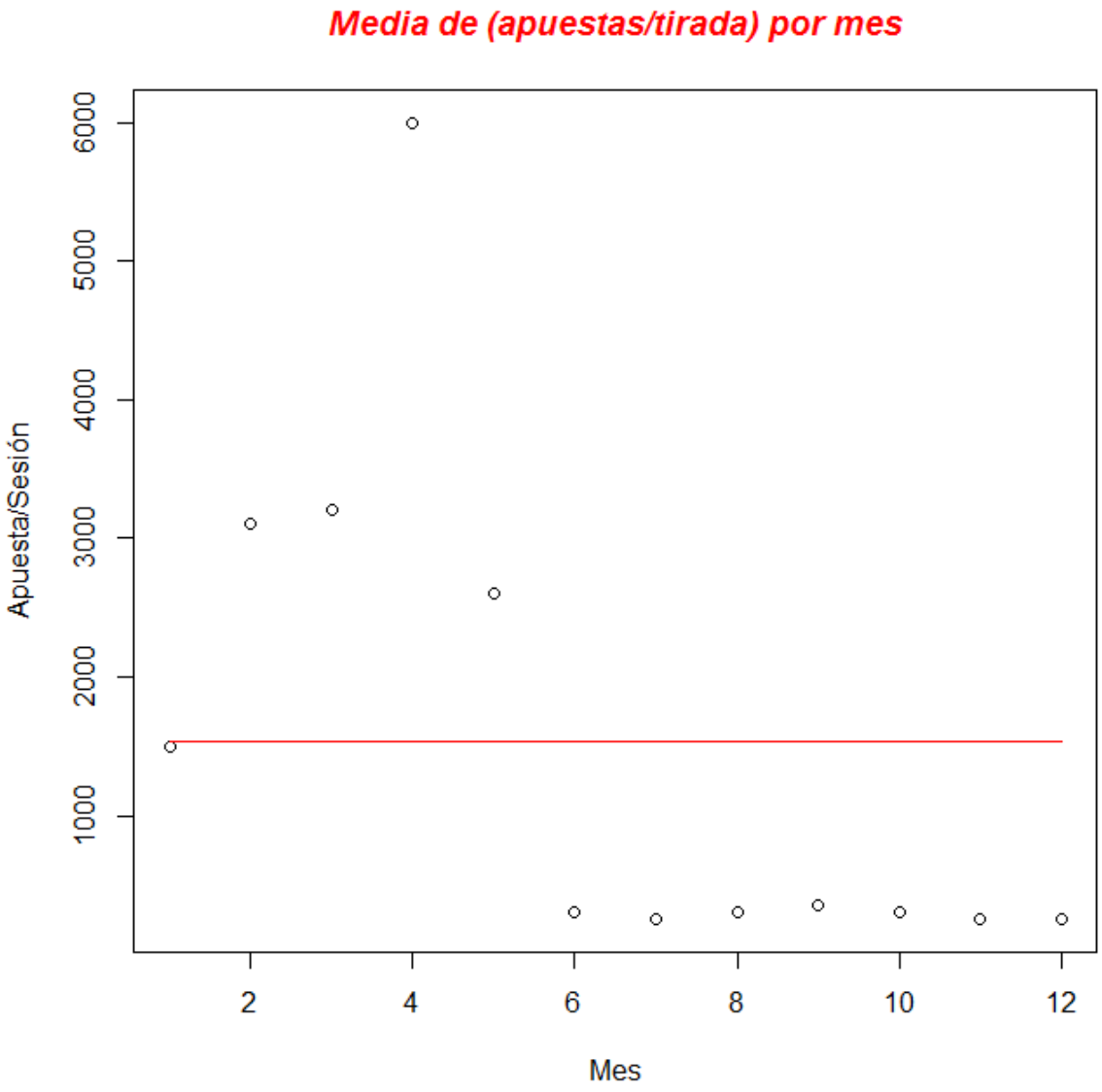


Figura 5.3: Cantidad que apuesta de media en cada spin por mes.

5.10.2. Resultados análisis estadístico

Abandona el slot cuando:

- 53.7 % de mayo a septiembre:
  - 27.4 % se queda sin monedas (1022 coins o menos);
  - 17.7 % balance positivo en las últimas 10 tiradas (51.000 coins o más);
  - 8.6 % balance positivo en el total de la sesión (450.000 coins o más);
- 35.3 % de octubre a abril:

    - 30.2 % se queda sin monedas (149 coins);
    - 5.1 % en las últimas 5 tiradas ha ganado 0 ó 1 veces;
- 11 % sin clasificar o mal clasificado.

Capítulo 6

Conclusiones

Para *THE NETWIZZY COMPANY S.L.* es fundamental entender el comportamiento de los jugadores estudiados ya que, aunque la masa de usuarios de *Our Slots* es muy grande, sólo unos cuantos de ellos son los que generan -casi- la totalidad de benficios de la empresa. Cribando los datos hemos podido comprobar como algunos de dichos usuarios pueden llegar a realizar hasta el 50 % del total de los spins que se realizan diariamente en *Our Slots*.

Tras analizar los resultados del estudio, podemos definir tres tipos básicos de jugadores:

- El primer grupo de usuarios de *Our Slots* se caracteriza por comprar diariamente packs pequeños de monedas y jugar hasta consumirlos;
- El segundo grupo de usuarios de *Our Slots* se caracteriza por adquirir packs de monedas grandes -invierten cantidades elevadas de dinero- con el objetivo de participar en torneos o lograr otros objetivos como conseguir fases de bonus, *freespins* o ganar el *jackpot* de algún slot. Es el grupo de usuarios que más benificios aporta a *THE NETWIZZY COMPANY S.L.*;
- El tercer grupo de usuarios de *Our Slots*, menos frecuente, se caracteriza por comprar monedas y jugar de forma moderada. Intenta, tanto minimizar las pérdidas, como no quedarse nunca sin monedas. Suele abandonar tras una buena racha.

Toda esta información permitirá a la empresa personalizar las ofertas y, por ello, cubrir mejor las necesidades del usuario maximizando, de esta forma, los beneficios.

Comentaremos, también, algunas mejoras interesantes para el estudio que durante la estancia en prácticas no nos ha dado tiempo a poner en práctica:

- Aumentar el período de estudio a dos años, introduciendo variables estacionales y temporales que contemplen, por ejemplo, si es principio o final de mes, víspera de día festivo, día festivo, horas diurnas o nocturnas,... Y desechar otras variables que no han sido de interés para el estudio;

- Muchos usuarios juegan en slots porque el valor del jackpot es alto y, cuando éste se da, abandonan el slot. Por tanto, en futuros estudios deberíamos implementar una nueva función en *php4* que contemplara este caso.
- Implementar otra función que muestre o contemple el resultado de los torneos a los que está jugando el usuario estudiado ya que, otro motivo por el cual dicho usuario puede abandonar el slot es que no esté ganando torneos.

Por último, dejar constancia de lo gratificante que ha sido esta experiencia en *THE NETWIZZY COMPANY S.L.*. He podido realizar una primera inmersión en el mundo laboral, interactuando con profesionales de capacidades muy diversas -gráficas, musicales, informáticas, matemáticas- con un fin común: el juego online. He recorrido un camino en el cual siempre he estado acompañado y he podido descubrir como, matemáticamente, se puede predecir el comportamiento de los usuario usando la informática como herramienta indispensable.

Bibliografía

[1] Alejandro Alonso Fúster, Lucía Arguelles Cortés. Teoría, procedimientos de demostración y ejercicios de análisis funcional. http://www.red-mat.unam.mx/foro/volumenes/vol017/Analisis_Funcional.pdf. [Consulta: 24 de junio de 2016].

[2] D. Achlioptas. Database friendly random projections. Proc 20th ACM Symp Principles of Database Systems, Santa Barbara, 2001, pp 274â281. [Consulta: 06 de septiembre de 2016].

[3] Nick Harvey. Randomized algorithms. Lecture 9, 2014-15 Term 2, University of British Columbia. [Consulta: 10 de septiembre de 2016].

[4] Jirí Matousek. Lectures on discrete geometry. Editorial Springer. [Consulta: 04 de julio de 2016].

[5] mtpalezp. Universidad del País Vasco. Producto escalar interior y espacios de hilbert. http://www.ehu.eus/~mtpalezp/libros/anafun3.pdf. [Consulta: 05 de julio de 2016].

[6] Oscar Blasco. Universidad de Valencia. Análisis funcional. http://www.uv.es/oblasco/Docencia/Teoria/AnalsisFuncional.pdf. [Consulta: 27 de junio de 2016].

[7] P. Frankl and H. Maehara. The johnson-lindenstrauss lemma and the sphericity of some graphs. Journal of Combinatorial Theory, 1988, pp 355â362. [Consulta: 01 de septiembre de 2016].

[8] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality,. Proc 30th Annu ACM Symp Theory of Computing, Dallas, 1998, pp 604â613. [Consulta: 02 de septiembre de 2016].

[9] Pedro Jos3́ Herrero Piñeyro. Universidad de Murcia. Topología. capítulo 3. http://www.um.es/docencia/pherrero/esp-top.pdf. [Consulta: 19 de junio de 2016].

[10] R. I. Arriaga and S. Vempala. An algorithmic theory of learning: Robust concepts and random projection. Proc 40th Annu IEEE Symp Foundations of Computer Science, New York, 1999, pp 616â623. [Consulta: 04 de septiembre de 2016].

[11] Sanjoy Dasgupta, Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. http://cseweb.ucsd.edu/~dasgupta/papers/jl.pdf. [Consulta: 11 de julio de 2016].

[12] https://cran.r-project.org/web/packages/igraph/AUTHORS. Paquete igraph de r. https://cran.r-project.org/web/packages/igraph/igraph.pdf. [Consulta: 10 de septiembre de 2016].

Anexo A

Códigos

En este apartado podemos ver los códigos implementados para resolver los problemas que hemos visto anteriormente. Veremos, por orden:

1. *Ejemplo previo*, hoja *número 29*;
2. *Variante primera*, hoja número 30;
3. *Variante segunda*, hoja *número 32*;
4. *Número de veces que encontramos el espacio de forma aleatoria*, hoja *número 34*;
5. *Aplicación 2. Vecino más proximo*, hoja *número 36*;
6. Código implementado en *THE NETWIZZY COMPANY S.L.*, *capítulo tercero.*

```
##########################
###                    ###
###   EJEMPLO PREVIO   ###
###                    ###
##########################

#Dimensión inicial del problema (d):
d = 5000;

#Número de observaciones (n):
n = 2500;

#Fijamos el nivel de tolerancia (epsilon):
epsilon = 0.25;

#La dimensión del nuevo espacio debe ser igual o mayor que k:
k = round(4/(epsilon^2/2-epsilon^3/3)*log(n),0)+1;

#k=1416. Por simplicidad, fijaremos k=1500.

###################################
### Creamos la matriz de puntos ###
###################################

#Fijamos una semilla:
set.seed(111);

#Inciamos el cronómetro:
t = proc.time()

#Simulamos el primer punto:
X = runif(d);

i=1;
while(i<n){
      #Simulamos una nuevo punto:
      newPoint = runif(d);

      #Concatenamos la matriz de puntos y el nuevo punto:
      X = rbind(X, newPoint);
      i=i+1;
}

proc.time()-t
# Tardamos 343.92 segundos en crear la matriz de puntos.

#Calculamos la matriz de distancias y convertimos el objeto
#distanceMatriz de tipo dist en otro de tipo matrix para poder trabajar
#con dicho objeto:

t = proc.time()
```

```
distanceMatrix = as.matrix(dist(X, method = "euclidean"))

proc.time()-t
# Ya tenemos la matriz de distancias de la matriz de puntos original.
# Se tarda 290.22 segundos en calcular dicha matriz.

#Fijamos una nueva semilla:
set.seed(32078)

#Seleccionamos las 1500 columnas que generarán el nuevo subespacio de
#forma aleatoria:
selectedColumns =  sample(1:5000, size=1500, replace=FALSE);

#Creamos la nueva matriz de puntos con las 2500 observaciones pero sólo
#con las 1500 columnas seleccionadas anteriormente:
Y = X[,selectedColumns]

#Creamos la nueva matriz de distancias:
t = proc.time()

newDistanceMatrix = as.matrix(dist(Y, method = "euclidean"))

proc.time()-t
# Ahora tardamos sólo 74 segundos.

#Calculamos la constante:
cte=sqrt(d/k);

#Multiplicamos la matriz de distancias newDistanceMatrix por la constante
#sqrt(k/d)
newDistanceMatrixDef = cte*newDistanceMatrix;

#Eliminamos los ceros de la diagonal:
A=diag(n);
newDistanceMatrixDef=newDistanceMatrixDef+A;
distanceMatrix=distanceMatrix+A;

#Dividimos, elemento a elemento, ambas matrices:
matrixResult = distanceMatrix/newDistanceMatrixDef;

##################
### Resultados ###
##################

# max(matrixResult)
# [1] 1
# min(matrixResult)
# [1] 0.8392504
# 1-min(matrixResult)
# [1] 0.1607496
# Epsilon vale, por tanto, 0.1607496.
```

```
############################
###                      ###
###   VARIANTE PRIMERA   ###
###                      ###
############################

#Dimensión inicial del problema (d):
d = 5000;

#Número de observaciones (n):
n1 = 2500;
n2 = 3000;
n3 = 3500;
n4 = 4000;
n5 = 4500;
n6 = 5000;

#Fijamos el nivel de tolerancia (epsilon):
epsilon = 0.25;

#La dimensión del nuevo espacio debe ser igual o mayor que k:
k1 = round(4/(epsilon^2/2-epsilon^3/3)*log(n1),0)+1;
k2 = round(4/(epsilon^2/2-epsilon^3/3)*log(n2),0)+1;
k3 = round(4/(epsilon^2/2-epsilon^3/3)*log(n3),0)+1;
k4 = round(4/(epsilon^2/2-epsilon^3/3)*log(n4),0)+1;
k5 = round(4/(epsilon^2/2-epsilon^3/3)*log(n5),0)+1;
k6 = round(4/(epsilon^2/2-epsilon^3/3)*log(n6),0)+1;

# Los valores de k, según el número de puntos son:
# > k1
# [1] 1203
# > k2
# [1] 1231
# > k3
# [1] 1254
# > k4
# [1] 1275
# > k5
# [1] 1293
# > k6
# [1] 1309
# Por simplicidad, fijaremos k=1500.

###################################
### Creamos la matriz de puntos ###
###################################

#Fijamos una semilla:
set.seed(111);

t = proc.time()
```

```
#Simulamos el primer punto:
X = runif(d);

i=1;
while(i<n6){
      #Simulamos un nuevo punto:
      newPoint = runif(d);

      #Concatenamos la matriz de puntos y el nuevo punto:
      X = rbind(X, newPoint);
      i=i+1;
}

proc.time()-t # Tardamos algo más de 24 minutos en calcular esta matriz.

# Generamos las matrices correspondientes según número de puntos:
X1 = X[1:2500,];
X2 = X[1:3000,];
X3 = X[1:3500,];
X4 = X[1:4000,];
X5 = X[1:4500,];
X6 = X[1:5000,];

#Fijamos otra semilla:
set.seed(32078)

#Seleccionamos las 1500 columnas que generarán el nuevo subespacio de
forma aleatoria:
selectedColumns =  sample(1:5000, size=1500, replace=FALSE);

#Creamos las nuevas matrices de puntos con las mismas observaciones pero
#sólo con las 1500 columnas seleccionadas anteriormente:
Y1 = X1[,selectedColumns];
Y2 = X2[,selectedColumns];
Y3 = X3[,selectedColumns];
Y4 = X4[,selectedColumns];
Y5 = X5[,selectedColumns];
Y6 = X6[,selectedColumns];

#Calculamos la matriz de distancias de todas las matrices de puntos
-tanto en dimensión 5000, como en dimensión 1500-:

#Dimensión: 5000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X1, method = "euclidean");
proc.time()-t

#Dimensión: 5000. Número de puntos: 3000.
t = proc.time()
distanceMatrix = dist(X2, method = "euclidean");
proc.time()-t
```

```
#Dimensión: 5000. Número de puntos: 3500.
t = proc.time()
distanceMatrix = dist(X3, method = "euclidean");
proc.time()-t

#Dimensión: 5000. Número de puntos: 4000.
t = proc.time()
distanceMatrix = dist(X4, method = "euclidean");
proc.time()-t

#Dimensión: 5000. Número de puntos: 4500.
t = proc.time()
distanceMatrix = dist(X5, method = "euclidean");
proc.time()-t

#Dimensión: 5000. Número de puntos: 5000.
t = proc.time()
distanceMatrix = dist(X6, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y1, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 3000.
t = proc.time()
distanceMatrix = dist(Y2, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 3500.
t = proc.time()
distanceMatrix = dist(Y3, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 4000.
t = proc.time()
distanceMatrix = dist(Y4, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 4500.
t = proc.time()
distanceMatrix = dist(Y5, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 5000.
t = proc.time()
distanceMatrix = dist(Y6, method = "euclidean");
proc.time()-t
```

```
##################
### Resultados ###
##################

#Dimensión: 5000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X1, method = "euclidean");
proc.time()-t
   user  system elapsed
 280.02    0.02  280.03

#Dimensión: 5000. Número de puntos: 3000.
t = proc.time()
distanceMatrix = dist(X2, method = "euclidean");
proc.time()-t
   user  system elapsed
 402.31    0.06  402.38

#Dimensión: 5000. Número de puntos: 3500.
t = proc.time()
distanceMatrix = dist(X3, method = "euclidean");
proc.time()-t
   user  system elapsed
 564.02    0.03  564.07

#Dimensión: 5000. Número de puntos: 4000.
t = proc.time()
distanceMatrix = dist(X4, method = "euclidean");
proc.time()-t
   user  system elapsed
 782.15    0.04  782.21

#Dimensión: 5000. Número de puntos: 4500.
t = proc.time()
distanceMatrix = dist(X5, method = "euclidean");
proc.time()-t
   user  system elapsed
 962.22    0.09  962.31

#Dimensión: 5000. Número de puntos: 5000.
t = proc.time()
distanceMatrix = dist(X6, method = "euclidean");
proc.time()-t
   user  system elapsed
1175.95    0.05 1176.05

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y1, method = "euclidean");
proc.time()-t
```

```
   user  system elapsed
  75.67    0.05   75.72

#Dimensión: 1500. Número de puntos: 3000.
t = proc.time()
distanceMatrix = dist(Y2, method = "euclidean");
proc.time()-t
   user  system elapsed
 107.20    0.06  107.27

#Dimensión: 1500. Número de puntos: 3500.
t = proc.time()
distanceMatrix = dist(Y3, method = "euclidean");
proc.time()-t
   user  system elapsed
 153.32    0.08  153.42

#Dimensión: 1500. Número de puntos: 4000.
t = proc.time()
distanceMatrix = dist(Y4, method = "euclidean");
proc.time()-t
   user  system elapsed
 208.65    0.08  208.75

#Dimensión: 1500. Número de puntos: 4500.
t = proc.time()
distanceMatrix = dist(Y5, method = "euclidean");
proc.time()-t
   user  system elapsed
 271.33    0.06  271.39

#Dimensión: 1500. Número de puntos: 5000.
t = proc.time()
distanceMatrix = dist(Y6, method = "euclidean");
proc.time()-t
   user  system elapsed
 336.65    0.06  336.71


####################
###   Gráfica   ###
####################

x1=c(280,402,564,782,962,1176);
x2=c(75,107,153,208,271,336);
y=c(2500,3000,3500,4000,4500,5000)

plot(y, x1, type = "o", col = "blue", xlab = "Tiempo (segundos)", ylab =
"Número de puntos de X" , xlim = c(2450,5050), ylim = c(70, 1200))
lines(y, x2, type="o", pch=22, lty=2, col = "red")
title(main="Número de puntos vs Tiempo", col.main="red", font.main=4)
```

```
############################
###                      ###
###   VARIANTE SEGUNDA   ###
###                      ###
############################

#Dimensión inicial del problema (d):
d1 = 5000;
d2 = 6000;
d3 = 7000;
d4 = 8000;
d5 = 9000;
d6 = 10000;

#Número de observaciones (n):
n = 2500;

#Fijamos el nivel de tolerancia (epsilon):
epsilon = 0.25;

#La dimensión del nuevo espacio debe ser igual o mayor que k:
k = round(4/(epsilon^2/2-epsilon^3/3)*log(n),0)+1;

# Por simplicidad, fijaremos k=1500.

###################################
### Creamos la matriz de puntos ###
###################################

#Fijamos una semilla:
set.seed(111);

t = proc.time()

#Simulamos el primer punto:
X = runif(d6);

i=1;
while(i<n){
      #Simulamos un nuevo punto:
      newPoint = runif(d6);

      #Concatenamos la matriz de puntos y el nuevo punto:
      X = rbind(X, newPoint);
      i=i+1;
}

proc.time()-t
# Tardamos 666 segundos, es decir, algo más de 11 minutos.
```

```
# Generamos las matrices correspondientes según número de puntos:
X1 = X[,1:5000];
X2 = X[,1:6000];
X3 = X[,1:7000];
X4 = X[,1:8000];
X5 = X[,1:9000];
X6 = X[,1:10000];

#Fijamos otra semilla:
set.seed(32078)

#Seleccionamos las 1500 columnas que generarán el nuevo subespacio de
forma aleatoria:
selectedColumns1 =  sample(1:5000, size=1500, replace=FALSE);
selectedColumns2 =  sample(1:6000, size=1500, replace=FALSE);
selectedColumns3 =  sample(1:7000, size=1500, replace=FALSE);
selectedColumns4 =  sample(1:8000, size=1500, replace=FALSE);
selectedColumns5 =  sample(1:9000, size=1500, replace=FALSE);
selectedColumns6 =  sample(1:10000, size=1500, replace=FALSE);


#Creamos las nuevas matrices de puntos con las mismas observaciones pero
#sólo con las 1500 columnas seleccionadas anteriormente:
Y1 = X1[,selectedColumns1];
Y2 = X2[,selectedColumns2];
Y3 = X3[,selectedColumns3];
Y4 = X4[,selectedColumns4];
Y5 = X5[,selectedColumns5];
Y6 = X6[,selectedColumns6];


#Calculamos la matriz de distancias de todas las matrices de puntos:

#Dimensión: 5000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X1, method = "euclidean");
proc.time()-t

#Dimensión: 6000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X2, method = "euclidean");
proc.time()-t

#Dimensión: 7000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X3, method = "euclidean");
proc.time()-t

#Dimensión: 8000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X4, method = "euclidean");
```

```
proc.time()-t

#Dimensión: 9000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X5, method = "euclidean");
proc.time()-t

#Dimensión: 10000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X6, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y1, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y2, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y3, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y4, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y5, method = "euclidean");
proc.time()-t

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y6, method = "euclidean");
proc.time()-t


##################
### Resultados ###
##################

#Dimensión: 5000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X1, method = "euclidean");
proc.time()-t
```

```
   user  system elapsed
 279.21    0.03  279.25

#Dimensión: 6000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X2, method = "euclidean");
proc.time()-t
   user  system elapsed
 339.17    0.03  339.22

#Dimensión: 7000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X3, method = "euclidean");
proc.time()-t
   user  system elapsed
 397.77    0.04  397.79

#Dimensión: 8000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X4, method = "euclidean");
proc.time()-t
   user  system elapsed
 457.42    0.00  457.42

#Dimensión: 9000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X5, method = "euclidean");
proc.time()-t
   user  system elapsed
 516.53    0.00  516.53

#Dimensión: 10000. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(X6, method = "euclidean");
proc.time()-t
   user  system elapsed
 575.75    0.01  575.77

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y1, method = "euclidean");
proc.time()-t
   user  system elapsed
  73.24    0.00   73.24

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y2, method = "euclidean");
proc.time()-t
   user  system elapsed
  73.85    0.05   73.90
```

```
#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y3, method = "euclidean");
proc.time()-t
   user  system elapsed
  73.45    0.03   73.50

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y4, method = "euclidean");
proc.time()-t
   user  system elapsed
  73.49    0.03   73.54

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y5, method = "euclidean");
proc.time()-t
   user  system elapsed
  72.96    0.03   73.00

#Dimensión: 1500. Número de puntos: 2500.
t = proc.time()
distanceMatrix = dist(Y6, method = "euclidean");
proc.time()-t
   user  system elapsed
  72.38    0.03   72.40

####################
###   Gráfica   ###
####################

tiempos1=c(279,339,397,457,516,575);
tiempos2=c(73,73,73,73,73,72);
dim=c(5000,6000,7000,8000,9000,10000)

plot(dim, tiempos1, type="o", col="blue", ylab = "Tiempo (segundos)",
xlab = "Dimensión inicial" ,xlim=c(4995,10005), ylim=c(72, 576))
lines(dim, tiempos2, type="o", pch=22,lty=2, col="red")
title(main="Dimensión inicial vs Tiempo", col.main="red", font.main=4)
```

```
#############################################################################
###                                                                       ###
###   Número de veces que encontramos el espacio de forma aleatoria       ###
###                                                                       ###
#############################################################################

#Dimensión inicial del problema (d):
d = 5000;

#Número de observaciones (n):
n = 2500;

#Fijamos el nivel de tolerancia (epsilon):
epsilon = 0.25;

#La dimensión del nuevo espacio, k, debe ser igual o mayor que newDim:
k = round(4/(epsilon^2/2-epsilon^3/3)*log(n),0)+1;
# k=1416. Por simplicidad, fijaremos k=1500.

k=1500;

###################################
### Creamos la matriz de puntos ###
###################################

#Fijamos una semilla:
set.seed(111);

#Simulamos el primer punto:
X = runif(d);

i=1;
while(i<n){
      #Simulamos una nuevo punto:
      newPoint = runif(d);

      #Concatenamos la matriz de puntos y el nuevo punto:
      X = rbind(X, newPoint);
      i=i+1;
}

#Calculamos la matriz de distancias:
distanceMatrix = as.matrix(dist(X, method = "euclidean"));

#Eliminamos los ceros de la diagonal:
A=diag(n);
distanceMatrix=distanceMatrix+A;
```

```
#############################################################################
###                                                                       ###
### Creamos una matriz de 50 filas. Cada fila está formada por 1500 ###
### columnas  seleccionadas de forma aleatoria y que generan el       ###
### nuevo subespacio.                                                   ###
###                                                                       ###
#############################################################################

set.seed(32078)

selectedColumns =  sample(1:5000, size=1500, replace=FALSE);

j=1;
while(j<50){
      #Simulamos 1500 columnas de nuevo:
      newSelectedColumns = sample(1:5000, size=1500, replace=FALSE);

      #Concatenamos selectedColumns con la nueva fila:
      selectedColumns = rbind(selectedColumns, newSelectedColumns);
      j=j+1;
}

vectorMin=c();
vectorMax=c();

i=1;

while(i<51){
      Y = X[,selectedColumns[i,]]

      #Creamos la nueva matriz de distancias:
      newDistanceMatrix = as.matrix(dist(Y, method = "euclidean"));

      #Calculamos la constante:
      cte=sqrt(d/k);

      # Multiplicamos la matriz de distancias newDistanceMatrix por la
      # constante sqrt(d/k)
      newDistanceMatrixDef = cte*newDistanceMatrix;

      #Eliminamos los ceros de la diagonal:
      newDistanceMatrixDef=newDistanceMatrixDef+A;

      # Dividimos, elemento a elemento, ambas matrices:
      matrixResult = distanceMatrix/newDistanceMatrixDef;

      vectorMax[i]=max(matrixResult)
      vectorMin[i]=min(matrixResult)

      i=i+1
}
```

```
##################
### Resultados ###
##################


> vectorMin
 [1] 0.9388281 0.9386575 0.9392014 0.9389769 0.9396454 0.9413918 0.9400079
 [8] 0.9383350 0.9404110 0.9392359 0.9386313 0.9412088 0.9398468 0.9404493
[15] 0.9357975 0.9395957 0.9404958 0.9435611 0.9282895 0.9399594 0.9422792
[22] 0.9397705 0.9417293 0.9320003 0.9400761 0.9416490 0.9339683 0.9373907
[29] 0.9356557 0.9375267 0.9397467 0.9435947 0.9383287 0.9372250 0.9403050
[36] 0.9423634 0.9423677 0.9364676 0.9418236 0.9338670 0.9381978 0.9415986
[43] 0.9383465 0.9404189 0.9397721 0.9385408 0.9415142 0.9440974 0.9407316
[50] 0.9405227

> vectorMax
 [1] 1.074431 1.070887 1.068900 1.064403 1.071320 1.067061 1.070161 1.070853
 [9] 1.070819 1.071365 1.066601 1.071227 1.070392 1.070267 1.073751 1.072688
[17] 1.064509 1.070621 1.071146 1.072397 1.067393 1.069425 1.069579 1.070277
[25] 1.066646 1.072900 1.068004 1.072913 1.071467 1.067512 1.076800 1.071484
[33] 1.073180 1.071907 1.070528 1.070528 1.069598 1.071891 1.074232 1.070481
[41] 1.076242 1.070910 1.070227 1.071500 1.071279 1.066882 1.076661 1.064643
[49] 1.078298 1.071065
```

```
################################
###   APLICACIÓN SEGUNDA     ###
################################


d = 10000;      # Dimensión inicial del problema (d).
n = 2500;       # Número de observaciones (n).
epsilon = 0,25. # Fijamos el nivel de tolerancia (epsilon):

# La dimensión del nuevo espacio debe ser igual o mayor que k:
k = round(4/(epsilon^2/2-epsilon^3/3)*log(n),0)+1;
# k=2000.

###################################
### Creamos la matriz de puntos ###
###################################

install.packages("igraph") # Descargamos e instalamos igraph:
library("igraph") # Cargamos igraph:

set.seed(111) # Fijamos una semilla:
# Simulamos la matriz de elementos X igual que la apliación primera.

t = proc.time()

# Calcumas la matriz de distancias que, en este caso, serán los
# diferentes pesos de las aristas:
pesos = as.matrix(dist(X))

# Generamos el grafo. Se puede hacer así o leyéndolo de un archivo:
data = matrix(c(1, 2, pesos[1,2], 1, 3, pesos[1,3], 2, 3, pesos[2,4], 2,
4, pesos[2,4], 3, 4, pesos[3,4], 4, 5, pesos[4,5], 5, 7, pesos[5,7], 5,
8, pesos[5,8], 7, 8, pesos[7,8], 7, 9, pesos[7,9], 8, 9, pesos[8,9], 9,
10, pesos[9,10], 9, 11, pesos[9,11], 9, 12, pesos[9,12], 10, 13,
pesos[10,13], 11, 13, pesos[11,13], 12, 13, pesos[12,13]), nrow = 17,
ncol=3, byrow = TRUE)

# Nombramos las filas y columnas:
colnames(data)=c('V1','V2','W')
rownames(data)=c('1','2','3','4','5','6','7','8','9','10','11','12','13',
'14','15','16','17')

# Empezamos a usar igraph:
# Creamos el grafo:
g <- graph.data.frame(data, directed = FALSE)
# Gráfico:
plot(g, edge.label = paste(E(g)$weight, sep = ""))

proc.time()-t
```

```
##################
### RESULTADOS ###
##################

# Calculamos el vecino más próximo de i:
graph.knn(g, i)

# Calculamos el camino de menor peso que un i,j:
gsp <- get.shortest.paths(g, from = "i", to = "j")

# Los nodos del camino más corto del nodo i a nodo j son:
V(g)[gsp$vpath[[1]]]

#############################################################################
```

Aplicando el **lema de Johnson-Lindestrauss** nos quedamos sólo con 2000 filas y repetimos el proceso:

```
Y = X[,2000]
cte = sqrt(10000/2000); #sqrt(d/k)

t = proc.time()

# Calculamos la matriz de distancias que, en este caso, serán los
# diferentes pesos de las aristas:
pesos2 = as.matrix(dist(Y))

# Multiplicamos la matriz de distancias por la constante:
Y=pesos2*cte;

# Generamos el grafo:
data2 = matrix(c(1, 2, pesos2[1,2], 1, 3, pesos2[1,3], 2, 3, pesos2[2,4],
2, 4, pesos2[2,4], 3, 4, pesos2[3,4], 4, 5, pesos2[4,5], 5, 7,
pesos2[5,7], 5, 8, pesos2[5,8], 7, 8, pesos2[7,8], 7, 9, pesos2[7,9], 8,
9, pesos2[8,9], 9, 10, pesos2[9,10], 9, 11, pesos2[9,11], 9, 12,
pesos2[9,12], 10, 13, pesos2[10,13], 11, 13, pesos2[11,13], 12, 13,
pesos2[12,13]), nrow = 17, ncol=3, byrow = TRUE)

# Nombramos las filas y columnas:
colnames(data2)=c('V1','V2','W')
rownames(data2)=c('1','2','3','4','5','6','7','8','9','10','11','12','13'
,'14','15','16','17')

# Empezamos a usar igraph:
# Creamos el grafo:
g2 <- graph.data.frame(data2, directed = FALSE)

# Gráfico:
plot(g2, edge.label = paste(E(g2)$weight, sep = ""))

proc.time()-t
```

```
##################
### RESULTADOS ###
##################

# Calculamos el vecino más próximo de i:
graph.knn(g2, i)

# Calculamos el camino de menor peso que un i,j:
gsp2 <- get.shortest.paths(g2, from = "i", to = "j")

# Los nodos del camino más corto del nodo i a nodo j son:
V(g2)[gsp2$vpath[[1]]]
```

Tardamos sólo 69.78 segundos en hacer los cálculos.

```
////////////////////////////////////////////////////////////////////////////
///                                                                      ///
///                  FUNCIONES AUXILIARES IMPLEMENTADAS                  ///
///                                                                      ///
////////////////////////////////////////////////////////////////////////////

//////////////////////////////
// Primera función auxiliar //
//////////////////////////////

//La función dameSegundos_EntreFechas devuelve, dadas dos fechas (en formato
hh:mm:ss) tal que fecha1 >= fecha2, el número de segundos entre ambas fechas:

function dameSegundos_EntreFechas ( $fecha1, $fecha2 ) {

      $aux = array();

      //La función explode devuelve un array (de Strings), siendo cada substring
el String formado por la división realizada por el delimitador ':'.
      $aux = explode( ":", $fecha1 );

      //Almacenamos en 'h1', 'm1' y 's1', horas, minutos y segundos de fecha1,
respectivamente:
      $h1 = $aux[0];
      $m1 = $aux[1];
      $s1 = $aux[2];

      $aux = explode( ":", $fecha2 );

      //Almacenamos en 'h2', 'm2' y 's2', horas, minutos y segundos de fecha2,
respectivamente:
      $h2 = $aux[0];
      $m2 = $aux[1];
      $s2 = $aux[2];

      $res=0; //Definimos la variable resultado

      //Si la fecha1 y fecha2 coinciden en horas y minutos la función devuelve,
directamente, la diferencia entre los segundos:
      if ($h1 == $h2 && $m1 == $m2) {     return ($s1 - $s2); }

      //Si sólo las horas coinciden:
      if ($h1 == $h2) {
      //Calculamos la diferencia de minutos (en segundos) entre ambas fechas y
sumamos o restamos la diferencia de segundos teniendo en cuenta que s1 puede ser
mayor, menor o igual que s2:
            $res = ($m1 - $m2)*60;
            if ($s2 > $s1) { $res = $res - ($s2 - $s1); }
            else { $res = $res + ($s1 - $s2); }
      }

      //Si las horas de ambas fechas no coinciden:
      if ($h1 != $h2) {
            //Calculamos la diferencia de horas (en minutos)
            $res = ($h1 - $h2)*60;
            //Incrementamos o decrementamos la cantidad anterior según si m1 <,
            = ó > que m2 (en segundos):
            if ($m2 > $m1) { $res = ($res - ($m2 - $m1)) * 60; }
            else { $res = ($res + ($m1 - $m2)) * 60; }
            //Por último, contemplamos la diferencia de segundos:
            if ($s2 > $s1) { $res = $res - ($s2 - $s1); }
            else { $res = $res + ($s1 - $s2); }
      }
      return $res; }
```

```
////////////////////////////
// Segunda función auxiliar //
////////////////////////////

/* La función contarCerosArraySinUltimo recibe como parámetro un $array y un entero $numPosiciones y devuelve el número de ceros (número de partidas perdidas o sin ganar) que tiene dicho array en las últimas $numPosiciones (sin contabilizar el último elemento del array).

Ejemplo de uso:
	$array = [1000, 100000, 25000, 0, 0, 1000];
	$numPosiciones = 3;

Llamamos a la función:
	$resultado = $this->contarCerosArraySinUltimo( $array, $numPosiciones );

Y cuando mostramos:
	$resultado ---> print('Valor de resultado: '.$resultado.'.') vemos que:
	Valor de resultado: 2.
*/

function contarCerosArraySinUltimo( $array, $numPosiciones ) {
	$resultado = 0;
	$j = 0; //Variable auxiliar

	//Si el número de posiciones a contar NO es mayor que la longitud del array:
	if( count($array) > $numPosiciones) {
		//Mientras aún nos queden posiciones que contar:
		while ( $j < $numPosiciones ) {
			//Si la posición del array es cero, incremento el valor de resultado en una unidad:
			if ($array[(count($array) -2 - $j)] == 0 ) { $resultado++; }
			$j++;
		}
		return $resultado;
	}

	//Si el número de posiciones a contar es mayor que la longitud del array, volvemos a llamar a la función siendo numPosiciones el número máximo de elementos del array que podemos contar:
	else {
		return  $this->contarCerosArraySinUltimo($array,(count($array)-1) );
	}
}


////////////////////////////
// Tercera función auxiliar //
////////////////////////////

/* La función contarCerosSeguidosArraySinUltimo recibe como parámetro un $array y devuelve el número de ceros seguidos (número de partidas perdidas de forma consecutiva) que tiene dicho array en las últimas posiciones (sin contabilizar el último elemento del array).

Ejemplo de uso:
	$arrayPrueba1 = [1000, 100000, 25000, 0, 0, 1000];
	$arrayPrueba2 = [1000, 0, 0, 0, 0, 0];

Llamamos a la función:
	$resultado1 = $this->contarCerosSeguidosArraySinUltimo($arrayPrueba1);
	$resultado2 = $this->contarCerosSeguidosArraySinUltimo($arrayPrueba2);
```

```
Y cuando mostramos los resultados: print('Valor de resultado1: '.$resultado1.' y
valor de resultado2: '.$resultado2) vemos que:
      Valor de resultado1: 0 y valor de resultado2: 4.
*/

function contarCerosSeguidosArraySinUltimo($array) {
      //Recorremos el array desde el final hasta el inicio:
      for ( $i=0; i <= count($array) - 1 ; $i++ ) {
            //Si no hay un cero, devuelvo el número de posiciones que he
recorrido:
                  if ( $array[count($array) - 2 - $i] != 0 ) {
                        return $i;
                  }
            }
            //Si llego a este punto, es que todos los elementos del vector son
ceros y devuelvo la longitud del vector
            return ( count($array) - 1 );
      }

////////////////////////////////////////////////////////////////////////////////
///                                                                          ///
///                           FUNCIÓN PRINCIPAL                              ///
///                                                                          ///
////////////////////////////////////////////////////////////////////////////////

function EstudioComportamientoUsuario_OurSlotsRPC ( $FechaInicial, $FechaFinal,
$CadNombreUsuario ){
```

Vamos a usar una función que tiene ya implementada la empresa que se llama *I_VuelcaArray_A_CSV($CadNomColumnas, $Datos, $RutaArcCSV, $Separador)*. Dicha función recibe los siguientes parámetros:

$CadNomColumnas: lista de nombres que asignaremos a cada columna;
$Datos: array de datos;
$RutaArcCSV: ruta en la que queremos generar el documento (incluye el nombre de dicho documento);
$Separador, generalmente serán " " ó ",".

Ejemplo de uso: `$this->I_VuelcaArray_A_CSV ( $CadNomColumnas, $Datos, '/var/www/lighttpd/estadisticas/-MineriaDatos-/PHPJavierCalahorra/Estudio_Comportamiento_Fatimamokht/Jugadas_Fatimamokht_vCor/fatimamokht.csv', ",");`

y la función genera un archivo .csv con todos los datos del array. Por tanto, el formato en que almacenaremos los datos será el que sigue:

```
      $Datos[contador][0]='numeroSesion';
      $Datos[contador][1]='fechaSesion';
      $Datos[contador][2]='horaIncio';
      $Datos[contador][3]='horaFin';
      $Datos[contador][4]='TiempoJugado';
      $Datos[contador][5]='slotId';
      $Datos[contador][6]='initialCoins';
      $Datos[contador][7]='endCoins';
      $Datos[contador][8]='numeroPartidas';
      $Datos[contador][9]='jackPot';
      $Datos[contador][10]='horaJackPot';
      $Datos[contador][11]='tiempoTranscurridoHastaCierreJackPot';
      $Datos[contador][12]='levelCoins';
      $Datos[contador][13]='horaLevelCoins';
      $Datos[contador][14]='tiempoTranscurridoHastaCierreLevelCoins';
```

```
        $Datos[contador][15]='extraCoins';
        $Datos[contador][16]='horaExtraCoins';
        $Datos[contador][17]='tiempoTranscurridoHastaCierreExtraCoins';
        $Datos[contador][18]='winCoins1';
        $Datos[contador][19]='betCoins1';
        $Datos[contador][20]='balanceSesion1';
        $Datos[contador][21]='winCoins3';
        $Datos[contador][22]='betCoins3';
        $Datos[contador][23]='balanceSesion3';
        $Datos[contador][24]='winCoins5';
        $Datos[contador][25]='betCoins5';
        $Datos[contador][26]='balanceSesion5';
        $Datos[contador][27]='winCoins10';
        $Datos[contador][28]='betCoins10';
        $Datos[contador][29]='balanceSesion10';
        $Datos[contador][30]='winCoins20';
        $Datos[contador][31]='betCoins20';
        $Datos[contador][32]='balanceSesion20';
        $Datos[contador][33]='winCoinsT';
        $Datos[contador][34]='betCoinsT';
        $Datos[contador][35]='balanceSesionT';
        $Datos[contador][36]='racha';
        $Datos[contador][37]='%racha3';
        $Datos[contador][38]='%racha5';
        $Datos[contador][39]='%racha10';
        $Datos[contador][40]='%racha20';
        $Datos[contador][41]='%rachaT';
```

```
        //DameArray_ListaFechas($FechaInicial,$FechaFinal)  es un método que devuelve un array unidimensional con la lista de fechas entre $FechaInicial, $FechaFinal. $ListaFechas[0] contiene el tamaño de ($ListaFechas):

        $ListaFechas=$this->DameArray_ListaFechas($FechaInicial,$FechaFinal);
```

En el caso que nos ocupa, $ListaFechas es de la siguiente forma:

```
[0] => 183
[1] => 2015-10-01
[2] => 2015-10-02
[3] => 2015-10-03
...
[183] => 2016-03-31
```

```
        //Ahora, obtengo la ruta de todos los archivos ($Lista_URL):
        $Lista_URL[0] = $ListaFechas[0];
        for ( $f = 1; $f <= $ListaFechas[0]; $f++ ) { $Lista_URL[$f]='/var/www/
              lighttpd/estadisticas/-MineriaDatos-/PHPJavierCalahorra/  Estudio_
              _Comportamiento_'.$CadNombreUsuario.'/Jugadas_'.$CadNombreUsuario.
              .'_vCor/'.($ListaFechas[$f]); }
```

Si mostramos por pantalla $Lista_URL:

```
[0] => 183
[1] => /var/www/lighttpd/estadisticas/-MineriaDatos-/PHPJavierCalahorra/
Estudio_Comportamiento_fatimamokht/Jugadas_fatimamokht_vCor/2015-10-01
[2] => /var/www/lighttpd/estadisticas/-
MineriaDatos-/PHPJavierCalahorra/Estudio_Comportamiento_sujeto1/Jugadas_sujeto1_
vCor/2015-10-02
```

```
...
[183] => /var/www/lighttpd/estadisticas/-
MineriaDatos-/PHPJavierCalahorra/Estudio_Comportamiento_sujeto1/Jugadas_fsujeto1
_vCor/2016-03-31
```

```
//Inicializamos algunas variables:
      $winCoins = array();          //<--- winCoins es un array en el que
                                    asignaremos a la posición 'i' el dinero
                                    retornado durante la partida 'i' de la sesión
                                    actual.
      $betCoins = array();          //<--- betCoins es un array en el que
                                    asignaremos a la posición 'i' el dinero
                                    apostado (bet*numLines) durante la partida
                                    'i' de la sesión actual.
      $contador = 0;                //<--- La variable auxiliar contador equivale
                                    a la variable columna del array de datos.
      $numeroSesion = 1;            //<--- Variable que contabilizar el número de
                                    sesiones que ha abierto el usario durante el
                                    periodo de estudio.
      $numPartidasActual = 1;       //Usaremos numPartidasActual para
                                    contabilizar el número de partidas que juega
                                    el sujeto1 en cada sesión.

//Recorremos la $Lista_URL:
for ($AuxArchivos = 1; $AuxArchivos <= $Lista_URL[0]; $AuxArchivos++) {
      $URL = $Lista_URL[$AuxArchivos];
      //Abrimos cada archivo:
      if (($handle = fopen($URL, "r")) !== FALSE) {
            $Fila=0;
            $primeraVez = 1;
            //Mientras haya líneas no vacías, leemos:
            while (($data = fgetcsv($handle, 500000, ",")) !== FALSE) {

      /* data es el siguiente array asociativo:
            Array
                  (
                  [0] => [30-Sep-2015 08:09:46] general.php OurSlotsAwardsCoins
                  {"result":null
                  [1] => levelCoins:0
                  [2] => jackPot:0
                  [3] => keys:0
                  [4] => type:"result"
                  [5] => slotsId:"lucky-dragon"
                  [6] => extraCoins:0
                  [7] => slotsNumId:21
                  [8] => coins:30
                  9] => userId:9451499
                  [10] => bet:6
                  [11] => endCoins:7017324
                  [12] => userName:"fatimamokht"
                  [13] => lines:30
                  [14] => levelPoints:30
                  [15] => initialCoins:7017294} 90001 1ms
                  )
      */
```

Durante la implementación de la función detectamos un error en los Slots *"hansel-and-gretel"* y *"tumbling-diamonds"*: no se registraba el jackPot en el JSON. Por tanto, el array asociativo es (ligeramente) distinto:

```
/* bug:
      Array
            (
            [0] => [02-Oct-2015 20:19:19] general.php OurSlotsAwardsCoins
            {"result":null
            [1] => levelCoins:0
            [2] => keys:0
            [3] => type:"result
            [4] => slotsId:"hansel-and-gretel
            [5] => extraCoins:0
            [6] => slotsNumId:7
            [7] => coins:0
            [8] => userId:9451499
            [9] => bet:500
            [10] => endCoins:4040159
            [11] => userName:"fatimamokht
            [12] => lines:20
            [13] => levelPoints:132
            [14] => initialCoins:4040159} 90001 1ms
            )
```

Nota: a día de hoy, en la empresa dicho error está subsanado.

```
//Si acabo de abrir una sesión ($primeraVez == 0) o un archivo ($Fila == 0) tengo que salvar algunas variables que, al leer la nueva fila, se sobreescribirán:

      if( $primeraVez == 0 || $Fila == 0) {
            $fechaDeSesion = $fechaSesion;      //Fecha de la sesión
            $horaFinSesion = $hora;             //horaFin de la sesión
            $slotsNumIdSesion = $slotsNumId;    //Slot de la sesión
            $endCoinsSesion = $endCoins;        //endCoins de la sesión
            $numPartidas = $numPartidasActual; //numPartidas de la sesión
            $horaAnterior = $hora;
      }

//Si la longitud del array asociativo data es 16 (todos los Slots menos 'hansel-and-gretel' y 'tumbling-diamonds'):

      if ( count($data) == 16 ) {
            //'fecha' y 'hora':
            $fechaHoraJuego = explode(" ", $data[0]);
            $fechaSesion = substr($fechaHoraJuego[0],1,11);
            $hora = substr($fechaHoraJuego[1],0,8);

            //'levelCoins', 'jackPot', 'extraCoins', 'slotsNumId', 'retornado', 'apuestaPorLinea', 'numLineas', 'endCoins', 'initialCoins':
            $aux = array();
            $aux2 = array();
            for ($i = 1; $i <= 14; $i++) {
                  $aux = explode(":", $data[$i]);
                  $aux2[$i] = $aux[1];
            }

            $levelCoins = $aux2[1];
            $jackPot = $aux2[2];
            $extraCoins = $aux2[6];
            $slotsNumId = $aux2[7];
            $retornado = $aux2[8];
            $apuestaPorLinea = $aux2[10];
            $numLineas = $aux2[13];
            $endCoins = $aux2[11];

            //'initialCoins':
            $initialCoins = explode("}", $data[15]);
```

```
            $initialCoins = $initialCoins[0];
            $initialCoins = explode(":", $initialCoins);
            $initialCoins = $initialCoins[1];
      }
```

Ahora comprobaremos que hemos conseguido, mediante la función *explode*, eliminar todos aquellos caracteres que debíamos eliminar del JSON y asignar correctamente los valores a las nuevas variables:

Tras la sentencia *print($fechaSesion.'-'.$hora.'-'.$levelCoins.'-'.$jackPot.'-'.$extraCoins.'-'.$slotsNumId.'-'.$retornado.'-'.$apuestaPorLinea.'-'.$numLineas.'-'.$endCoins.'-'.$initialCoins)* se muestra lo que sigue:

(30-Sep-2015-08:09:46-0-0-0-21-30-6-30-7017324-7017294)

```
      //Consideramos, siguiendo un proceso similar, 'hansel-and-gretel' y 'tumbling-diamonds':

      else if ( count($data) == 15 ) {
            //'fecha' y 'hora':
            $fechaHoraJuego = explode(" ", $data[0]);
            $fechaSesion = substr($fechaHoraJuego[0],1,11);
            $hora = substr($fechaHoraJuego[1],0,8);

            //'levelCoins', 'jackPot', 'extraCoins', 'slotsNumId', 'retornado', 'apuestaPorLinea', 'numLineas', 'endCoins', 'initialCoins':
            $aux = array();
            $aux2 = array();
            for ($i = 1; $i <= 14; $i++) {
                  $aux = explode(":", $data[$i]);
                  $aux2[$i] = $aux[1];
            }

            $levelCoins = $aux2[1];
            $jackPot = 0;
            $extraCoins = $aux2[5];
            $slotsNumId = $aux2[6];
            $retornado = $aux2[7];
            $apuestaPorLinea = $aux2[9];
            $numLineas = $aux2[12];
            $endCoins = $aux2[10];

            //'initialCoins':
            $initialCoins = explode("}", $data[14]);
            $initialCoins = $initialCoins[0];
            $initialCoins = explode(":", $initialCoins);
            $initialCoins = $initialCoins[1];
      } else {
            print ('Error: la longitud de la línea levelCoins del RPC no es la adecuada'); //<--- Control de errores
      }

//Añadimos el valor de retornado a $winCoins y $bet*numLíneas a $betCoins:
      array_push($winCoins, $retornado);
      array_push($betCoins, ($apuestaPorLinea*$numLineas));

      if( $primeraVez == 0 || $Fila == 0) {
```

```
      //Si cambia el día, también es nueva sesión de usuario. Por tanto,
escribiremos el resumen de la sesión en Datos y reiniciaremos el valor de las
variables auxiliares:
      if(($Fila==0 || $fechaDeSesion!=$fechaSesion) && ($fechaDeSesion!=null)) {
            //Guardo datos de la sesión:
            $tiempoJugado = $this->dameSegundos_EntreFechas ( $horaFinSesion,
            $horaIncioSesion);
            $Datos[$contador][0]=$numeroSesion;
            $Datos[$contador][1]=$fechaDeSesion;
            $Datos[$contador][2]=$horaIncioSesion;
            $Datos[$contador][3]=$horaFinSesion;
            $Datos[$contador][4]=$tiempoJugado;
            $Datos[$contador][5]=$slotsNumIdSesion;
            $Datos[$contador][6]=$initialCoinsSesion;
            $Datos[$contador][7]=$endCoinsSesion;
            $Datos[$contador][8]=$numPartidas;
            //Calculo el tiempo desde que se gana el evento hasta
            cierreSesión/cambioSlot:
            if ($Datos[$contador][9] != null){
                  $tiempoJugadoDesdeJackPot = $this -> dameSegundos_EntreFechas
                  ($horaFinSesion,$Datos[$contador][10]);
                  $Datos[$contador][11]=$tiempoJugadoDesdeJackPot; }
      if ($Datos[$contador][12] != null){
            $tiempoJugadoDesdeLevelCoins=$this->dameSegundos_EntreFechas
            ($horaFinSesion,$Datos[$contador][13]);
            $Datos[$contador][14]=$tiempoJugadoDesdeLevelCoins;
      }
      if ($Datos[$contador][15] != null){
            $tiempoJugadoDesdeExtraCoins=$this->dameSegundos_EntreFechas
            ($horaFinSesion,$Datos[$contador][16]);
            $Datos[$contador][17]=$tiempoJugadoDesdeExtraCoins;
      }
```

Métodos importantes:

array_sum(): devuelve la suma de los valores de un array.

array_slice(): devuelve la secuencia de elementos de un array según los parámetros *offset* y *length* donde:

1. *offset*: Si el índice dado por offset no es negativo, la secuencia empezará en esa posición del array. Si el offset es negativo, la secuencia empezará en esa posición empezando por el final del array;
2. *length:* longitud de la secuencia.

Teniendo en cuenta que el último elemento del array pertenece a la próxima sesión y NO a la actual, una forma sencilla de calcular las monedas apostadas en las últimas *i* tiradas es la siguiente:

```
array_sum(array_slice($betCoins, -(i+1), i))
```

```
$Datos[$contador][18] = array_sum(array_slice($winCoins, -2, 1));
$Datos[$contador][19]=array_sum(array_slice($betCoins, -2, 1));
$Datos[$contador][20]=($Datos[$contador][18]-$Datos[$contador][19]);
$Datos[$contador][21]=array_sum(array_slice($winCoins, -4, 3));
$Datos[$contador][22]=array_sum(array_slice($betCoins, -4, 3));
$Datos[$contador][23]=($Datos[$contador][21]-$Datos[$contador][22]);
$Datos[$contador][24]=array_sum(array_slice($winCoins, -6, 5));
$Datos[$contador][25]=array_sum(array_slice($betCoins, -6, 5));
$Datos[$contador][26]=($Datos[$contador][24]-$Datos[$contador][25]);
Datos[$contador][27]=array_sum(array_slice($winCoins, -11, 10));
$Datos[$contador][28]=array_sum(array_slice($betCoins, -11, 10));
$Datos[$contador][29]=($Datos[$contador][27]-$Datos[$contador][28]);
$Datos[$contador][30]=array_sum(array_slice($winCoins, -21, 20));
$Datos[$contador][31]=array_sum(array_slice($betCoins, -21, 20));
$Datos[$contador][32]=($Datos[$contador][30]-$Datos[$contador][31]);
$Datos[$contador][33]=array_sum($winCoins)-winCoins[(count($winCoins)-1)]);
$Datos[$contador][34]=(array_sum($betCoins)-$betCoins[(count($betCoins)-1)]);
$Datos[$contador][35]=($Datos[$contador][33]-$Datos[$contador][34]);
$Datos[$contador][36]=$this->contarCerosSeguidosArraySinUltimo($winCoins);
$Datos[$contador][37]=100-($this->contarCerosArraySinUltimo($winCoins,3)*100/3);
$Datos[$contador][38]=100-($this->contarCerosArraySinUltimo($winCoins,5)*100/5);
$Datos[$contador][39]=100-($this->contarCerosArraySinUltimo($winCoins,10)*100/10);
$Datos[$contador][40]=100-($this->contarCerosArraySinUltimo($winCoins,20)*100/20);
$Datos[$contador][41]=100-($this->contarCerosArraySinUltimo
($winCoins,count($winCoins)-1))*100/(count($winCoins)-1));
```

```
//Una vez guardados todos los datos de la sesión, actualizamos los datos apra la
nueva:


//Eliminamos todos los elementos de $winCoins y $endCoins excepto el último,que
pertenece, ya, a la sesión actual:


            $winCoins = array($winCoins[(count($winCoins)-1)]);
            $betCoins = array($betCoins[(count($betCoins)-1)]);

            $numPartidasActual=1;
            $numeroSesion++;
            $contador++;
            $primeraVez = 1;
      }

      else {
            if ($fechaDeSesion != null) {
            //Si el usuario no ha jugado durante 10+ minutos, entendemos que ha
cambiado de sesión:
            $seg = $this->dameSegundos_EntreFechas($hora,$horaAnterior);
                  if ( $seg > 600) {

                  //Guardamos, de la misma forma que antes, los datos de la
                  sesión e inciamos una nueva: datos de la sesión:
```

GUARDAMOS LA SESIÓN CON EL MISMO CÓDIGO

```
                        $numPartidasActual=1;
                        $numeroSesion++;
                        $contador++;
                        $primeraVez = 1;
                  }
```

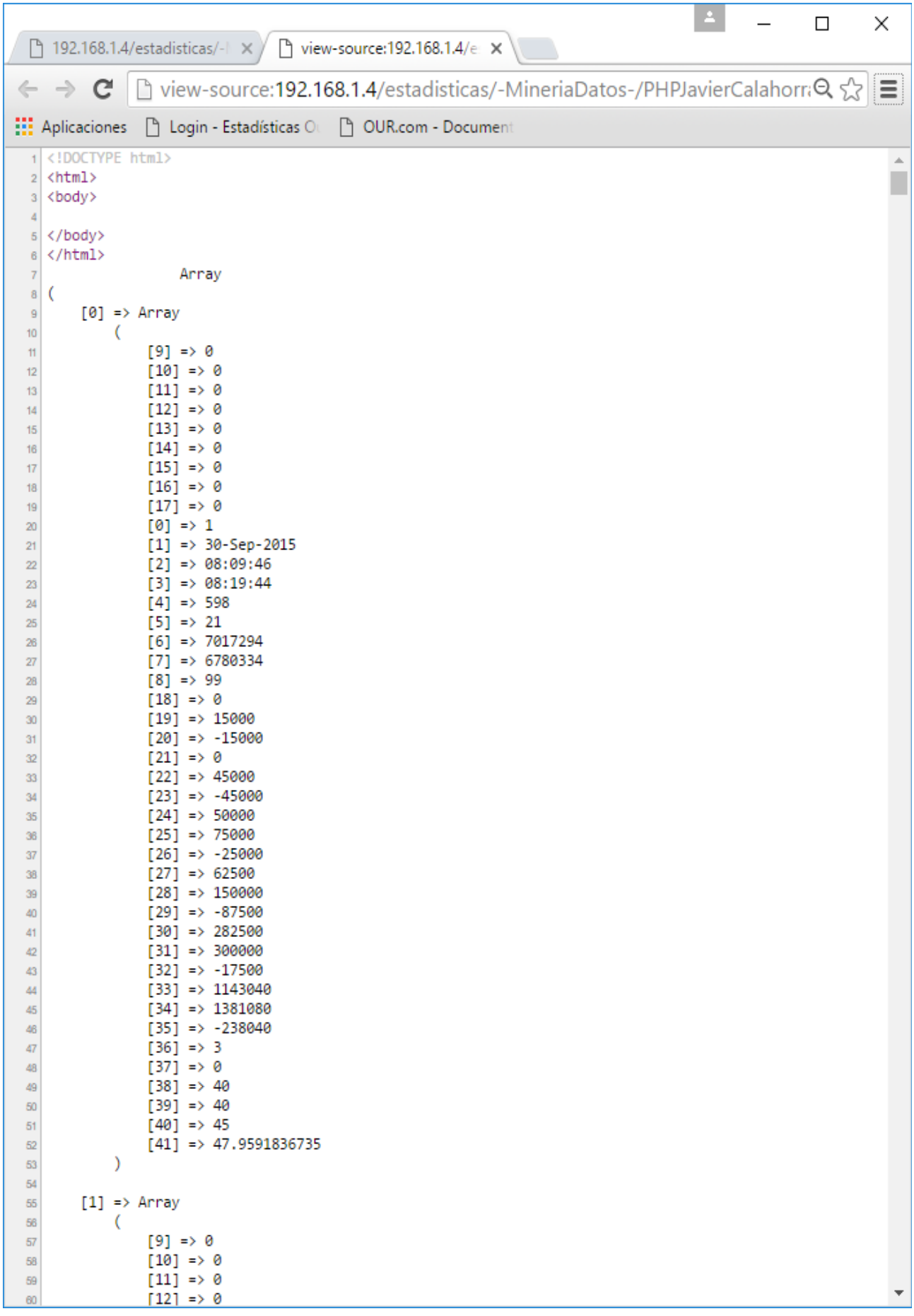
```
                    //El usuario, en la misma sesión, ha cambiado de Slot:
                    if ( $seg <= 600 && $slotsNumIdSesion != $slotsNumId ) {

                    //Guardamos, también, la información recaba en el array Datos
                    e inicializamos las variables . los datos de la sesión e
                    inciamos una nueva:
```

GUARDAMOS LA SESIÓN CON EL MISMO CÓDIGO

```
                    $numPartidasActual=1;
                    $contador++;
                    $primeraVez = 1;
                    }
            }
      }
}

//Notar que, tras leer todos los archivos, tendremos que almacenar en Datos la
última sesión registrada:
```

GUARDAMOS LA SESIÓN CON EL MISMO CÓDIGO

La matriz Datos, mostrada por pantalla, es como sigue:




```
<!DOCTYPE html>
<html>
<body>

</body>
</html>
                Array
(
    [0] => Array
        (
            [9] => 0
            [10] => 0
            [11] => 0
            [12] => 0
            [13] => 0
            [14] => 0
            [15] => 0
            [16] => 0
            [17] => 0
            [0] => 1
            [1] => 30-Sep-2015
            [2] => 08:09:46
            [3] => 08:19:44
            [4] => 598
            [5] => 21
            [6] => 7017294
            [7] => 6780334
            [8] => 99
            [18] => 0
            [19] => 15000
            [20] => -15000
            [21] => 0
            [22] => 45000
            [23] => -45000
            [24] => 50000
            [25] => 75000
            [26] => -25000
            [27] => 62500
            [28] => 150000
            [29] => -87500
            [30] => 282500
            [31] => 300000
            [32] => -17500
            [33] => 1143040
            [34] => 1381080
            [35] => -238040
            [36] => 3
            [37] => 0
            [38] => 40
            [39] => 40
            [40] => 45
            [41] => 47.9591836735
        )

    [1] => Array
        (
            [9] => 0
            [10] => 0
            [11] => 0
            [12] => 0
```

Por último y, teniendo en cuenta que el objetivo es hacer un análisis estadístico de los datos (con software especializado en minería de datos como *R* o *Weka*), volcaremos dicha matriz de datos en un archivo:

Primero, creamos la cadena con el nombre de las columnas:

```
$CadNomColumnas = "numeroSesion, fechaSesion, horaIncio, horaFin, TiempoJugado,
slotId, initialCoins, endCoins, numeroPartidas, jackPot, horaJackPot,
tiempoTranscurridoHastaCierreJackPot, levelCoins, horaLevelCoins,
tiempoTranscurridoHastaCierreLevelCoins, extraCoins,horaExtraCoins,
tiempoTranscurridoHastaCierreExtraCoins, winCoins1, betCoins1, balanceSesion1,
winCoins3, betCoins3, balanceSesion3, winCoins5, betCoins5, balanceSesion5,
winCoins10, betCoins10, balanceSesion10, winCoins20, betCoins20,
balanceSesion20, winCoinsT, betCoinsT, balanceSesionT, racha, %racha3, %racha5,
%racha10, %racha20, %rachaT";
```

Y, ahora, generamos el archivo mediante la función I_VuelvaArray_A_CSV:

```
    $this->I_VuelcaArray_A_CSV($CadNomColumnas, $Datos, '/var/www/lighttpd/
estadisticas/-MineriaDatos-/PHPJavierCalahorra/Estudio_Comportamiento_sujeto1
/Jugadas_sujeto1_vCor/sujeto1.csv', ",");
```