Índice general

1. Introducción **3**

2. Objetivos **5**

3. Desarrollo **7**
3.1. Matemáticas 8
3.1.1. Introducción al problema 8
3.1.2. Factibilidad del aprendizaje 11
3.1.3. Generalización: Teoría de Vapnik-Chervonenkis 11
3.1.4. Sobreajuste 11
3.1.5. Modelos de árboles: Boosting 11
3.2. Ingeniería Informática 12
3.2.1. Comprensión del problema y de los datos 12
3.2.2. Preprocesamiento de datos 15

4. Conclusiones **19**

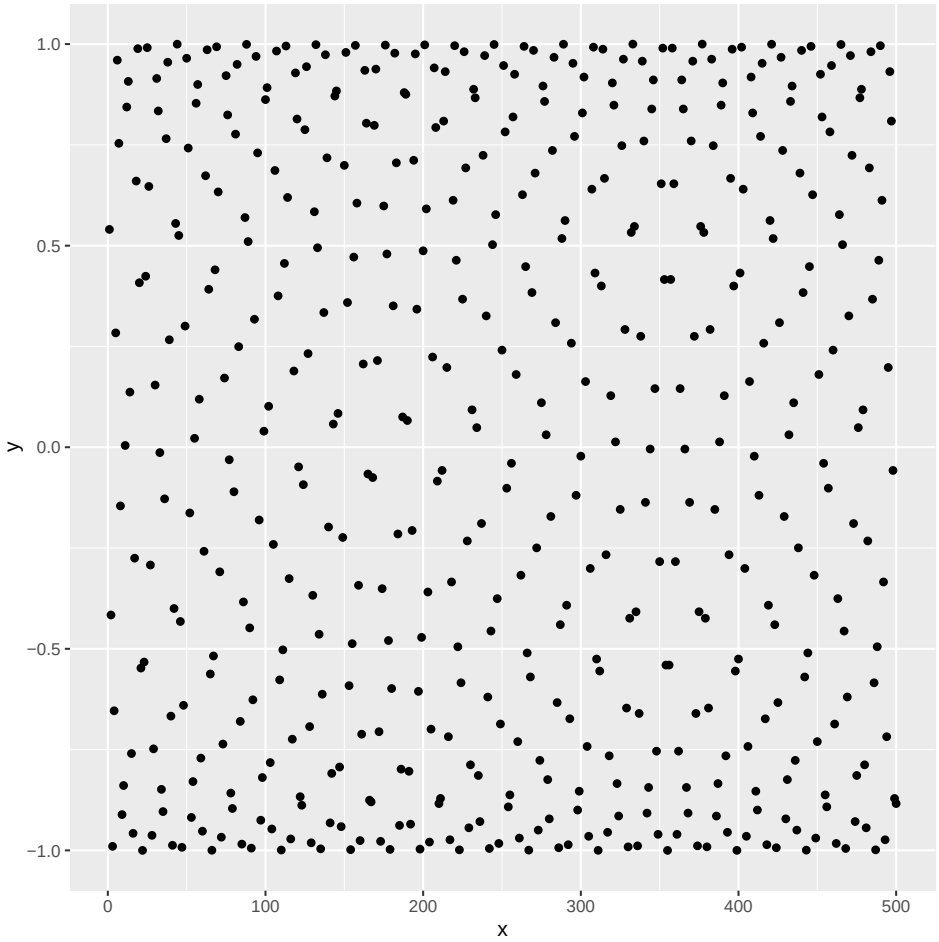

Figura 1: prueba

Capítulo 1

Introducción

Capítulo 2

Objetivos

Capítulo 3

Desarrollo

Este trabajo ha sido desarrollando utilizando las siguientes herramientas:

- LaTeX como procesador de textos científico.
- *R* como lenguaje de tratamiento de datos. Junto a *knitr*, se ha escrito este documento con el paradigma de *programación literaria*, donde el documento y los programas están en simbiosis, ofreciendo comodidad tanto para escribir como para leer.
- MariaDB como sistema gestor de bases de datos relacionales (SQL).

Estas decisiones se han tomado, principalmente, por ser las mejores herramientas en cada uno de sus ámbitos, con la grandísima ventaja de ser software libre.

Además, la plataforma donde se ha realizado la computación, es la siguiente:

- Sistema operativo GNU/Linux - Archlinux x86_64
- Intel i7 6700K @ 4.2Ghz
- G.Skill Ripjaws V DDR4 2400Mhz 4x8GB
- eVGA GTX 960 4GB ACX2.0+
- 2x128GB SSD + 2x750GB HDD en RAID0 vía hardware

La decisión de usar un RAID0 es por la falta de discos para formar un sistema con redundancia. Con este sistema tenemos un sistema rápido y de gran capacidad, para albergar las grandes bases de datos que se precisan, aunque sin tolerancia a fallos. Al ser vía hardware, la tasa de fallo es mucho menor que uno vía software, por lo que en un entorno doméstico es suficiente.

3.1. Matemáticas

Se introducen las herramientas matemáticas necesarias para la consecución de este trabajo.

3.1.1. Introducción al problema

Parafraseando a [1], si a un niño pequeño se le pregunta si hay un árbol en una foto, lo más probable es que dé la respuesta correcta. Si a uno algo mayor se le pregunta "¿Qué es un árbol", posiblemente no sabrá contestar o su respuesta no sea muy útil. Los seres humanos no han aprendido qué es un árbol con una definición matemática, sino viendo árboles. Han aprendido desde los *datos*.

Este tipo de aprendizaje puede ser modelado matemáticamente para solucionar una gran cantidad de problemas en ciencia, ingeniería, economía, ..., etc. Esta rama de las matemáticas se llama *Aprendizaje Estadístico*.

Matemáticamente, el problema del aprendizaje es, dado un conjunto de puntos de entrada $\mathcal{X} \in \mathbb{R}^d$, un espacio de salida $\mathcal{Y}$, una función objetivo **desconocida** $f : \mathcal{X} \to \mathcal{Y}$ y N muestras $((x_1, y_1), \ldots, (x_N, y_N))$ tal que $f(x_i) = y_i,\ i = 1, \ldots, N$, se desea conseguir una función de un determinado espacio de hipótesis, $g \in \mathcal{H}$ que se asemeje todo lo posible a la función objetivo f, $g \approx f$.

En el aprendizaje se distinguen muchos problemas, siendo tres los más conocidos:

- Regresión.
- Clasificación.
- Estimación de densidad.

Un ejemplo de clasificación sería el siguiente: sea $\mathcal{X} \subseteq [0, 1] \times [0, 1]$ e $\mathcal{Y} = \{+1, -1\}$ el espacio de salida, el cual se puede considerar sí/no, f la función objetivo. Generando muestras aleatorias, un ejemplo sencillo sería el de la figura 3.1. En lenguaje natural, tenemos dos características con las que se puede decidir si un correo electrónico entrante es spam, por ejemplo. Se necesita una función g que cumpla que $\forall x \in \mathcal{X}$, $g(x) = f(x)$. Intuitivamente, en el caso de clasificación se busca una función que *separe* los conjuntos de datos con distintas etiquetas. ¿Existe un separador para este conjunto? En este caso, **sí**, siendo además el más sencillo, un separador *lineal*. Así se puede ver en la figura 3.2.

En este sencillo caso se puede deducir lo siguiente: en un lado de la recta obtenida se tienen todos los datos con etiquetas 1 y en el otro los que tienen etiquetas -1. A los conjuntos de datos que cumplen esto los llamaremos *linealmente separables*. Este hecho no es lo usual en un problema de aprendizaje real. Ni siquiera se sabe a priori si, ante nuevos datos, esta

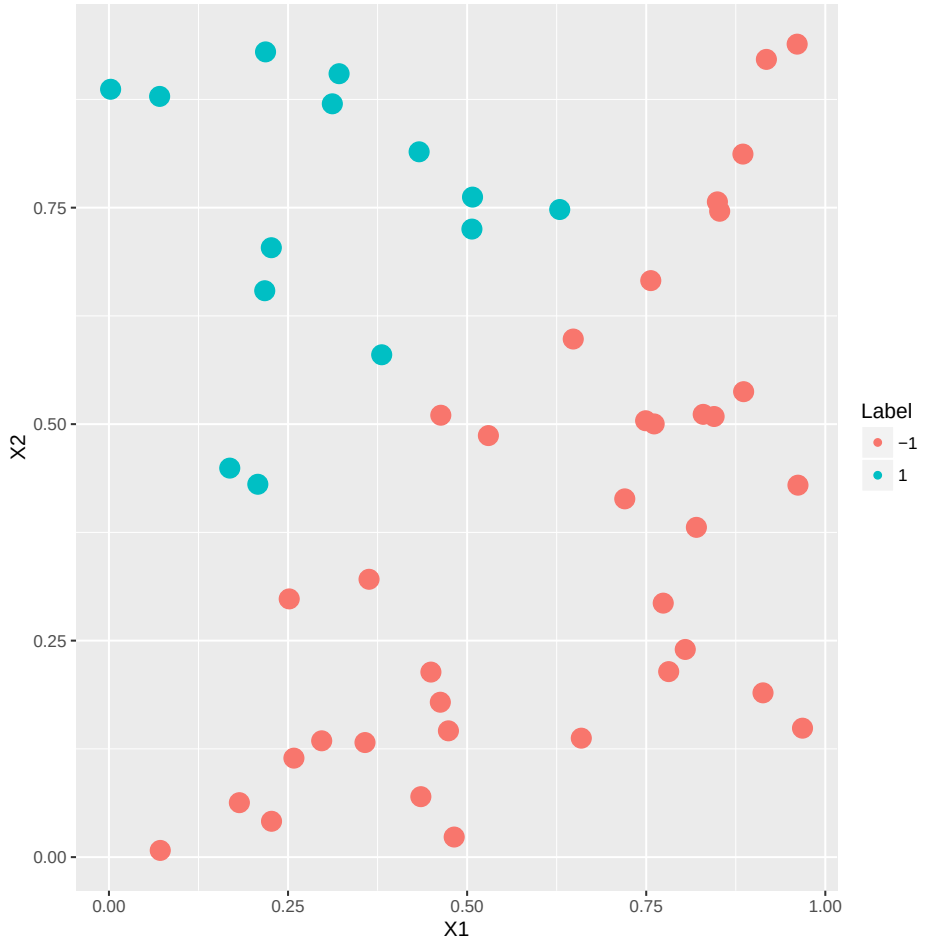

Figura 3.1: Datos de ejemplo

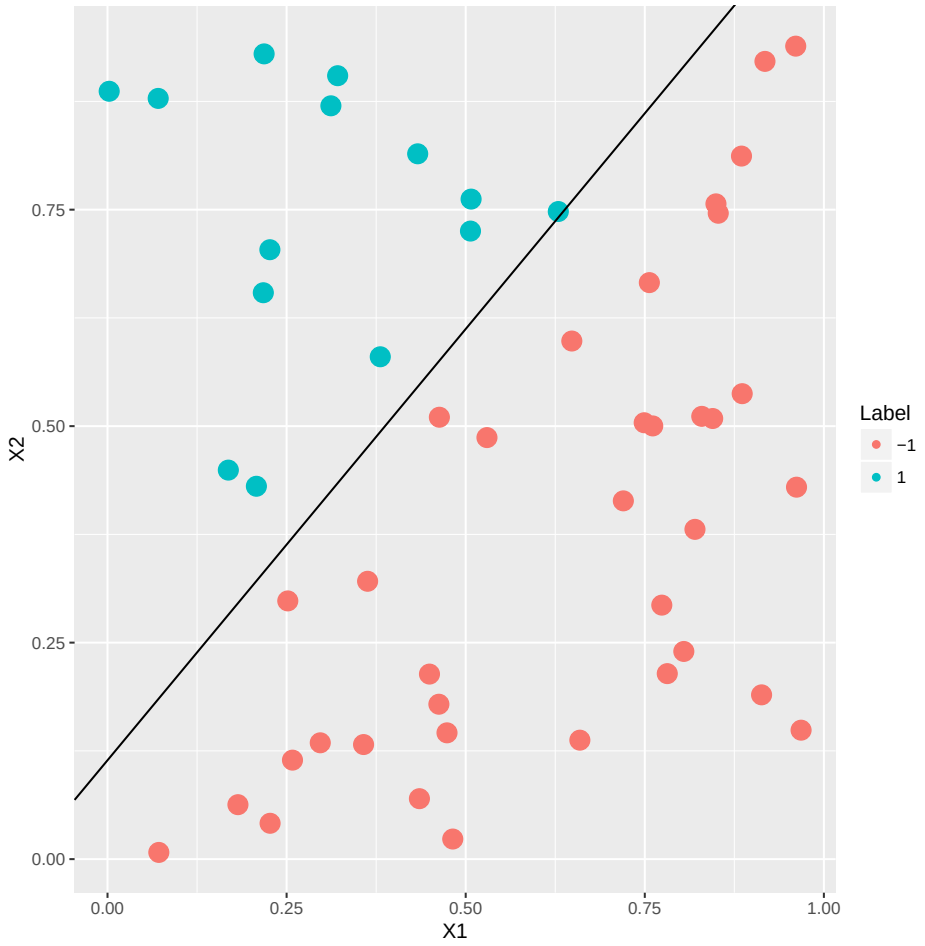

Figura 3.2: Primer clasificador

recta los seguirá separando correctamente. Incluso, ante nuevos datos, el conjunto podría pasar a ser no linealmente separable.

3.1.2. Factibilidad del aprendizaje

El aprendizaje estadístico busca *obtener información desde los datos*, utilizando para ello técnicas estadísticas. Se empezará estudiando una cuestión que se ha planteado antes por encima: ante la llegada de nuevos datos, ¿se tiene un modelo adecuado? ¿Es factible aprender?

Definición 3.1. Error dentro de la muestra (E_{in}). Es la fracción de $\mathcal{D}$ donde la función escogida h y la objetivo f difieren.

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N} \chi(h(x_n) \neq f(x_n))$$

Definición 3.2. Error fuera de la muestra (E_{out}). Es la probabilidad, basada en la distribución de $\mathcal{X}$, de que la función escogida h y la objetivo f difieren.

$$E_{out}(h) = \mathbb{P}[h(\mathbf{x}) \neq f(\mathbf{x})]$$

3.1.3. Generalización: Teoría de Vapnik-Chervonenkis

3.1.4. Sobreajuste

3.1.5. Modelos de árboles: Boosting

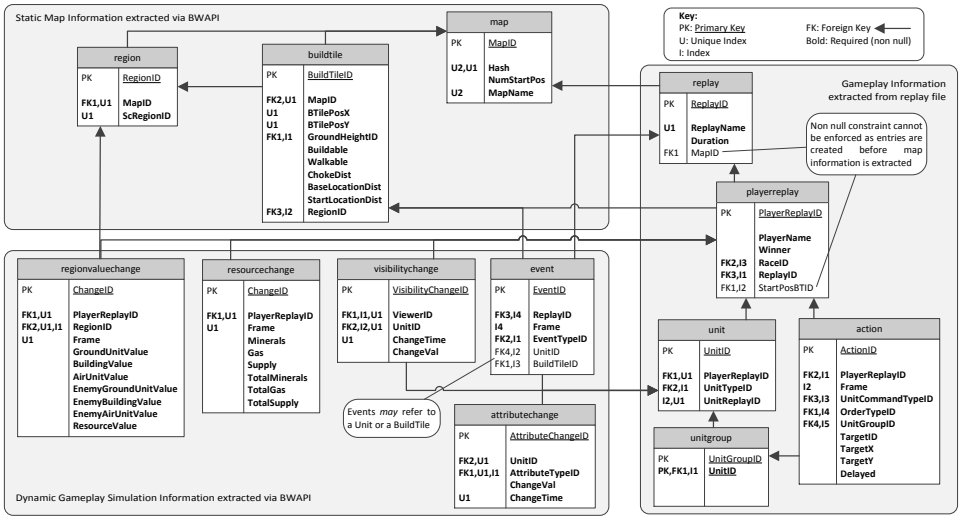

Figura 3.3: Base de datos de partidas de StarCraft

3.2. Ingeniería Informática

Una vez se dispone de todas las herramientas matemáticas necesarias, se puede comenzar con el desarrollo de la parte de Ingeniería Informática que se aborda en el trabajo.

3.2.1. Comprensión del problema y de los datos

En primer lugar en cualquier problema de Ciencia de Datos, el primer paso es *comprender el problema y los datos que se disponen.*

El problema a abordar es el siguiente: a partir de un conjunto de partidas antiguas de StarCraft [2], se busca predecir el momento en el que la partida está decidida con una confianza determinada. Para ello, partimos de 6 bases de datos relacionales (SQL) con gran cantidad de partidas almacenadas, cada una con muchas características a observar. Cada una de ellas posee las características presentes en la figura 3.3.

Una vez se tiene conocimiento del problema y un conjunto de datos, se debe decidir qué datos y características van a ser usados y de qué forma. El principal problema de este paso es conocer el conjunto de datos del que se dispone, ya que usualmente no es extraído por los investigadores.

Las características están sacadas casi en su totalidad directamente de valores que proporciona la API que permite interactuar con StarCraft, *BWAPI*. Otros son datos derivados, como la distancia de un jugador a la base más cercana, por ejemplo.

Las características que van a ser usados en este trabajo son, principalmente, los recursos de cada jugador, sus batallones (que son medidos de

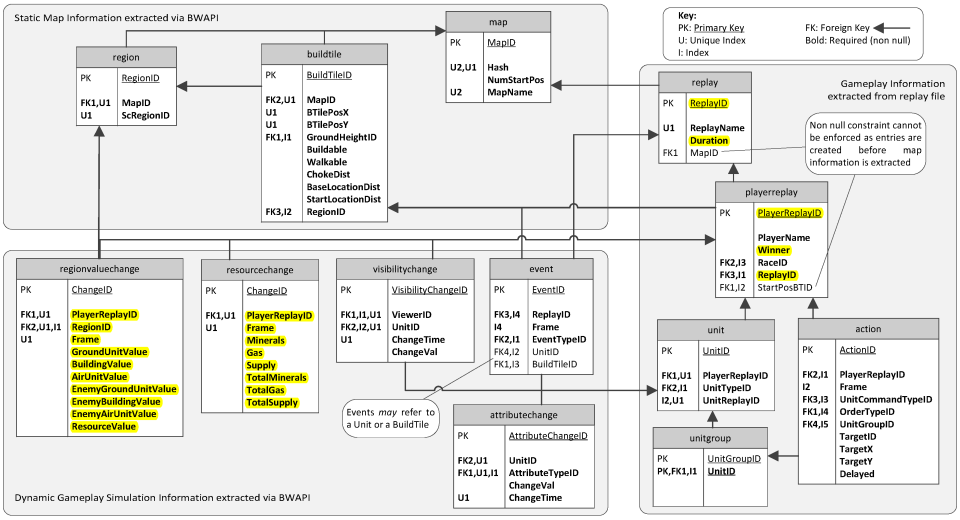

Figura 3.4: Características seleccionadas de las bases de datos

una manera determinada que se explican con más detalle más adelante), sus construcciones, y los valores estimados de batallones y contrucciones que tienen un jugador del otro. Además, también se tiene en cuenta los recursos restantes del mapa que cada jugador estima que quedan. Estas características quedan reflejadas en la figura 3.4.

Estas características son, según cada tabla:

- replay: Esta tabla contiene datos asociados a cada partida.
  - ReplayID: Identificador de cada partida.
  - Duration: Duración (en frames) de cada partida. 15 frames equivalen a 1 segundo.
- playerreplay: Esta tabla contiene datos asociados a un jugador en una partida.
  - PlayerReplayID: Identificador de un jugador en una partida.
  - ReplayID: Identificador de partida asociado.
  - Winner: Ganador de cada partida.
- resourcechange: Esta tabla contiene datos asociados a cambios en los recursos de un jugador.

- PlayerReplayID: Identificador del jugador que produce un cambio.
- Frame: Frame en el que se produce un cambio.
- Minerals: Cantidad de minerales que tiene un jugador en ese momento.
- Gas: Cantidad de gas que tiene un jugador en ese momento.
- Supply: Capacidad de carga del jugador.
- TotalMinerals: Cantidad total de minerales que ha obtenido un jugador, sin contar gastos.
- TotalGas: Cantidad total de gas que ha obtenido un jugador, sin contar gastos.
- TotalSupply: Capacidad que ha obtenido un jugador, sin contar gastos.

- regionvaluechange: Esta tabla contiene datos asociados a cambios de un jugador en una región del mapa determinada. Cada *value*, que llamaremos de aquí en adelante *valor*, es la suma del precio de una unidad en Minerales y Gas.

  - PlayerReplayID: Identificador del jugador que produce un cambio.
  - RegionID: Identificador de la región del mapa donde se produce un cambio.
  - Frame: Frame en el que se produce el cambio.
  - GroundUnitValue: Valor de las unidades terrestres en esta región.
  - BuildingValue: Valor de las construcciones en esta región.
  - AirUnitValue: Valor de las unidades aéreas en esta región.
  - EnemyGroundUnitValue: Valor de las unidades terrestres del enemigo en esta región. Este valor es estimado, sólo se conoce lo que el jugador puede ver del enemigo.
  - EnemyBuildingValue: Valor de las contrucciones del enemigo en esta región. Este valor es estimado, sólo se conoce lo que el jugador puede ver del enemigo.
  - EnemyAirUnitValue: Valor de las unidades aéreas del enemigo en esta región. Este valor es estimado, sólo se conoce lo que el jugador puede ver del enemigo.
  - ResourceValue: Valor de los recursos en esta región. Este valor es estimado, sólo se conoce lo que el jugador puede ver del mapa. Si el jugador no conoce una zona, estima que los recursos restantes es la totalidad de lo disponible en la región.

Una vez se ha decidido qué vamos a usar, hay que pasar al cómo. Esta fase es el *preprocesamiento de los datos*, que es la fase donde se organizan los datos, se corrigen si hubiera datos perdidos o ruido... para poder abordar el problema a resolver.

3.2.2. Preprocesamiento de datos

Aquí meter toda la lógica que se ha usado para extraer los datos. Qué y cómo se ha hecho.

Análisis exploratorio de datos

Lectura de datos

```
print(getwd())

## [1] "/home/antonio/Git/TFG/Memoria/capitulos"

data.pvp <- read.csv("../datos/data_pvp.csv")

## Warning in file(file, "rt"): no fue posible abrir el archivo
'../datos/data_pvp.csv': No existe el fichero o el directorio
## Error in file(file, "rt"): no se puede abrir la conexión

data.pvt <- read.csv("../datos/data_pvt.csv")

## Warning in file(file, "rt"): no fue posible abrir el archivo
'../datos/data_pvt.csv': No existe el fichero o el directorio
## Error in file(file, "rt"): no se puede abrir la conexión

data.pvz <- read.csv("../datos/data_pvz.csv")

## Warning in file(file, "rt"): no fue posible abrir el archivo
'../datos/data_pvz.csv': No existe el fichero o el directorio
## Error in file(file, "rt"): no se puede abrir la conexión

data.tvt <- read.csv("../datos/data_tvt.csv")

## Warning in file(file, "rt"): no fue posible abrir el archivo
'../datos/data_tvt.csv': No existe el fichero o el directorio
## Error in file(file, "rt"): no se puede abrir la conexión

data.tvz <- read.csv("../datos/data_tvz.csv")

## Warning in file(file, "rt"): no fue posible abrir el archivo
'../datos/data_tvz.csv': No existe el fichero o el directorio
## Error in file(file, "rt"): no se puede abrir la conexión
```

```
data.zvz <- read.csv("../datos/data_zvz.csv")

## Warning in file(file, "rt"): no fue posible abrir el archivo
'../datos/data_zvz.csv': No existe el fichero o el directorio
## Error in file(file, "rt"): no se puede abrir la conexión

data.full <- rbind(cbind(data.pvp, Races = "PvP"), cbind(data.pvt, Races = "PvT
                   cbind(data.pvz, Races = "PvZ"), cbind(data.tvt, Races = "TvT
                   cbind(data.tvz, Races = "TvZ"), cbind(data.zvz, Races = "ZvZ

## Error in cbind(data.pvp, Races = "PvP"): objeto 'data.pvp' no
encontrado

data.full$ReplayID  <- as.factor(paste(data.full$Races,data.full$ReplayID,sep =

## Error in paste(data.full$Races, data.full$ReplayID, sep = "_"):
objeto 'data.full' no encontrado

metadata <- data.full[, colnames(data.full) %in% c("ReplayID","Duration", "Race

## Error in eval(expr, envir, enclos): objeto 'data.full' no encontrado

metadata <- unique(metadata[order(-metadata$Duration),])

## Error in unique(metadata[order(-metadata$Duration), ]): objeto
'metadata' no encontrado

metadata <- transform(metadata, ReplayID=reorder(ReplayID, -Duration) )

## Error in transform(metadata, ReplayID = reorder(ReplayID, -Duration)):
objeto 'metadata' no encontrado

# Change all ReplayIDs because there are duplicates
# metadata£ReplayID <- 1:nrow(metadata)
# data.full£ReplayID <- metadata£ReplayID

# clean_data <- data.full[,!(colnames(data) %in% c("ReplayID"))]
system.time(data.full.transformed <- transformData(data.full))

## Error in system.time(data.full.transformed <- transformData(data.full)):
no se pudo encontrar la función "transformData"

## Timing stopped at: 0 0 0

data.full.transformed.auc <- transformData(data.full, auc = T)

## Error in eval(expr, envir, enclos): no se pudo encontrar la función
"transformData"
```

Histograma

```
ggplot(data=metadata) +
geom_bar(aes(x=ReplayID,y=Duration,fill=Duration), stat="identity") +
geom_hline(yintercept = mean(metadata$Duration))

## Error in ggplot(data = metadata): objeto 'metadata' no encontrado

print(mean(metadata$Duration))

## Error in mean(metadata$Duration): objeto 'metadata' no encontrado
```

Capítulo 4

Conclusiones

Bibliografía

[1] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*. AMLBook, 2012.

[2] Glen Robertson and Ian Watson. An improved dataset and extraction process for starcraft ai. 2014. Proceedings of the Florida Artificial Intelligence Research Society Conference (FLAIRS).