UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Visual Analytics and Team Strategies in Online Games

Sónia dos Santos Portugal Alves da Costa

Mestrado em Informática

Dissertação orientada por:
Prof.ª Doutora Ana Paula Pereira Afonso
Prof.ª Doutora Maria Beatriz Duarte Pereira do Carmo

2022

Acknowledgements

I would like to thank my supervisors, professors Ana Paula Afonso and Maria Beatriz Carmo, as they gave me the opportunity to develop this thesis, and their continuous support allowed me to finish it.
Secondly, I would like to acknowledge the team *Muito Riso Pouco Siso* for their help in the final stage of this project; Thank you for taking the time to evaluate my work and give your feedback.
Finally, I would like to thank all my friends and family who endured this long process with me, directly or indirectly helping me with their company, emotional support, and care (and helping me review this very report).

Acknowledgements.

Resumo

O fenómeno dos *e-sports* (desportos eletrónicos) tem vindo a crescer e, com este, também o interesse por videojogos *online*, por parte de jogadores e espectadores. Com a evolução tecnológica tem-se tornado cada vez mais fácil utilizar técnicas de recolha de dados sobre os eventos que decorrem durante um jogo gerando grandes volumes de dados que podem ser utilizados para análise do desempenho de jogadores e de equipas. Este tipo de análise é de grande importância tanto em contextos pessoais como em contextos profissionais. Os jogadores casuais procuram métodos que permitam perceber que erros estão a cometer e qual a forma ótima de jogar com determinadas personagens ao passo que num contexto profissional, o foco é maioritariamente perceber que estratégias são utilizadas por outras equipas e como combatê-las.

Atualmente as ferramentas existentes focam-se em dados estáticos, como número de mortes, rácio vitória/derrota ou ouro obtido por jogo, por exemplo. Podem também ser restritas à análise de jogos únicos, o que não é útil na reflexão do comportamento geral de um jogador. Para obter conclusões mais realistas de como um jogador se comporta normalmente, deve-se abordar os dados espácio-temporais e também a análise multi-jogos. Mesmo quando oferece análise multi-jogos, perdem na interatividade, onde o utilizador não pode ajustar as visualizações presentes aos seus objetivos de análise, como por exemplo escolher quantos jogos quer analisar.

Há que considerar que para ter uma perspetiva do comportamento de um jogador a nível profissional, será preciso um grande volume de dados. Para que a análise deste volume de dados seja eficaz, é fundamental explorar mecanismos de análise de dados combinadas com técnicas de visualização (visualização analítica) aplicadas a dados espácio-temporais e aos vários tipos de eventos durante uma partida e que são de interesse para os jogadores, treinadores e analistas. Estes eventos podem variar desde a posição de um jogador (espaço) num determinado instante (tempo), a eventos mais específicos do jogo, tais como a posição onde o jogador morreu ou quando e onde cumpriu determinado objetivo.

Focando neste objetivo, concluiu-se que algoritmos de *clustering*, onde os dados agrupados em *clusters*(agrupamentos) onde todos os pontos que pertencem ao mesmo agrupamento tem características semelhantes, será o mais indicado para a análise espácio-temporal, tendo sido escolhido o algoritmo DBSCAN (*Density-based spatial clustering of applications with noise*), que para além das características já referidas, também ignora

outliers (ie. pontos da base de dados que fogem aos *clusters* criados pelo algoritmo), o que permite uma visualização mais realista e clara dos dados. Também este algoritmo é ideal para visualizações, visto que estes *clusters* podem ser visualizados em mapas (animados ou não). Em relação a técnicas de visualização analítica, os vários artigos estudados neste projeto apontam para que visualizações como mapas animados e *heatmaps* são boas escolhas para abordar dados espácio-temporais.

O trabalho relacionado estudado neste relatório também conclui que a análise espácio-temporal não só é importante para jogadores, como para treinadores e *game designers*, como analisado no sistema Echo [26], onde comparado com gravações de ecrã, este tipo de abordagem permite uma análise mais rápida e mais precisa, visto que retira o fator do erro humano. Contextualizando as visualizações em análises de videojogos, os mapas animados e *heatmaps* podem permitir a *designers* perceber como é que os utilizadores das suas aplicações se comportam, podendo até detetar erros de desenho nas mesmas. Também temos o trabalho de Camilo [12], onde se utilizou dados espácio-temporais para criar *dashboards* com o objetivo de ajudar o Regimento de Sapadores Bombeiros de Lisboa, criando ferramentas visuais que contribuem para a deteção e compreensão dos padrões nas ocorrências na cidade para melhorar a gestão operacional dos bombeiros. O estudo realizado conclui que para a utilização deste tipo de ferramentas não é necessária experiência prévia para que se consiga obter a informação pretendida de forma rápida e eficaz.

No âmbito de teses de mestrado anteriores [10], [28], [42] e [13], foram exploradas técnicas de visualização e de análise de dados para um dos jogos mais populares, do tipo MOBA (*Multiplayer Online Battle Arena*), League of Legends[31] (LoL). Nestes projetos foram desenvolvidos e avaliados protótipos que analisaram a adequação de técnicas de visualização (por exemplo, mapas animados) para explorar os dados fornecidos pela API (Application Programming interface) da Riot Games [7] e tirar conclusões relativamente ao desempenho dos jogadores e das estratégias utilizadas. Na atual versão VisuaLeague3.0 [10] foram exploradas técnicas de visualização para jogos de equipas de LoL, de modo a que possam ser utilizadas por treinadores de equipas profissionais para análise de estratégias de jogo, e análise da evolução espácio-temporal.

Neste projeto pretende-se dar continuidade ao trabalho iniciado num trabalho anterior de mestrado [10] na aplicação de algoritmos de *clustering* na análise espácio-temporal de dados de jogos e jogadores, combinando técnicas de visualização analíticas e algoritmos de *clustering* ao jogo LoL, com o foco em análise multi-jogos e interatividade da plataforma/ferramenta com o utilizador, seja ele jogador principiante ou treinador profissional.

A informação utilizada foi a de jogos e jogadores fornecida pela API da Riot de partidas de Legue of Legends (LoL). LoL é um jogo MOBA onde duas equipas de cinco jogadores batalham com o objetivo de destruir a base adversária. Cada jogador, que são identifcados pelo seu *summoner's name*, escolhe um *champion* (personagem do jogo), e

assumirá um *role* (papel) na equipa. Existem cinco *roles* principais, *Top*, *Jungler*, *Middle*, *Bottom* e *Support*, sendo estes definidos pelas características do *champion* escolhido e a posição que o jogador toma no mapa. Cada *role* tem um comportamento expectável durante o jogo, como por exemplo o *Top* tem uma abordagem mais defensiva e o *Jungler* normalmente encontra-se na *Jungle* (zona do mapa) com o objetivo de poder viajar entre as várias zona do mapa para suportar os colegas de equipa.

Para obter os dados necessários recorreu-se à Riot API em conjunto com a biblioteca riotwatcher, e fez-se o pré-processamento inicial dos dados em Python para obter e organizar toda a informação necessária relativamente aos dados estáticos e espácio-temporais de jogadores com apenas o seu *summoner's name* (nome de utilizador). Com isto criou-se, para cada jogador a analizar, três ficheiros CSV (*Comma-separated values*), para que posteriormente os dados possam ser inseridos nas respetivas visualizações.

A ferramenta desenvolvida utiliza o Tableau Desktop para criar uma série de *dashboards* que representam o comportamento de jogadores de LoL em múltiplos jogos, dependendo do seu *role*, utilizando dados fornecidos pela Riot API em conjunto com técnicas de *clustering*. Nesta ferramenta, um utilizador pode interagir com as diferentes visualizações através de filtros, e personalizar a sua experiência de acordo com os seus objetivos de análise.

Para desenhar as visualizações necessárias utilizou-se a informação obtida por Afonso [10], que no seu estudo entrevistou dois treinadores profissionais, com o objetivo de rastrear os seus processos de análise de jogadores singulares e quais as informações que achavam mais relevantes para identificar o bom comportamento e resultados do mesmo. De uma forma mais geral, concluiu-se que delinear o movimento de um jogador ao longo do tempo em vários jogos e verificar se estes cumprem determinados objetivos são métricas importantes a analisar. Adicionalmente, e consoante a situação, pode ser importante variar o número de jogos a analisar em cada instante. Com esta informação base, em conjunto com as ferramentas já fornecidas pelo Tableau e dados fornecidos pela Riot API, desenhou-se os *dashboards* finais apresentados neste projeto.

A ferramenta desenvolvida foi avaliada por uma equipa semi-profissional de modo a perceber se as técnicas de visualização e descrição dos dados foi adequada, útil e inovadora comparado com as ferramentas já existentes para a análise de jogos e para as necessidades dos jogadores. Os resultados foram maioritariamente positivos, com os participantes realçando a inovação na perspetiva espácio-temporal, visto que não existem até à data ferramentas que apresentem estes dados. Para além disso, mencionaram a capacidade de poder interagir e personalizar as visualizações, como no exemplo da escolha do número de jogos que pretende agregar nas visualizações apresentadas. Os *dashboards* também se mostraram bastante úteis e relevantes para a análise pessoal dos participantes.

Podemos concluir neste estudo que existe procura para este tipo ferramentas de análise de jogos MOBA, visto que dados espácio-temporais ainda não estão a ser implemen-

tada noutras ferramentas de análise já existentes, apesar de já disponíveis por alguns desenvolvedores destes mesmos jogos. Também podemos inferir que a interação com as visualizações é um aspeto crucial a focar em futuras versões da ferramenta, já que foi um dos aspetos de interesse mencionado pelos jogadores.

Palavras-chave: Visualização analítica, dados espácio-temporais, video jogos, Tableau, DBSCAN

Abstract

The eSports (electronic sports) phenomenon has been growing and so does the interest in online video games, from players and spectators. With technological advancements it has become easier to use techniques to retrieve data about the events occurring during a game, generating big volumes of data that can be used for a performance analysis. Casual players are looking for methods to better themselves overall or with specific characters, whereas, in a professional context, the focus is to study other teams and how to defeat them. For efficiency, it is imperative to explore data analysis mechanisms combined with visualisation techniques (visual analytics) applied to spatiotemporal data and to various relevant events during a match such as a player's position (space) in a given instant (time) or, for example, the position where the player died.

The goal of this project is the study of previous work and the development and application of the acquired knowledge in analytic visualisation techniques to League of Legends[31] (LoL) spatiotemporal datasets. The developed tool used Tableau Desktop[24] to create a series of dashboards depicting the behaviour of multiple LoL matches, using the Riot API (Application Programming Interface) provided dataset, and clustering algorithms.

The tool was evaluated by a team of semi-professional players in order to understand if the visualisation techniques and data used was adequate, useful or innovative compared to already existing tools for game analysis and the players' needs. The results were mostly positive, with the participants pointing out the interactivity of the visualisations and ability of analysing multiple games as an advantage compared to existing tools. To conclude, even though spatiotemporal data is not yet implemented in MOBA (Multiplayer Online Battle Arena) videogame analysis tools, it is still relevant for the players' personal goals and overall an interesting approach.

Keywords: Visual Analytics, spatiotemporal datasets, video games datasets, Tableau, DBSCAN

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface. 17, 27–35, 48, 52, 53

CSV Comma-separated values. 22, 33, 52

DBSCAN Density-based spatial clustering of applications with noise. 7, 19, 23, 25, 26, 35, 36, 48, 51, 52

LoL League of Legends. iv, 1, 11, 16, 19, 27, 31, 32, 45, 47, 49, 51, 52

MOBA Multiplayer Online Battle Arena. iv, 1, 15, 19, 51

Chapter 1

Introduction

This chapter describes the main motivations and goals of this project, focusing on the importance of videogame analytics, followed by the motivation, goals, and contributions of this work, ending with the structure of the report.

1.1 Motivation

ESports is the phenomenon of sports competition using videogames, usually held in online tournaments between professional players (individually or organized in teams). By 2019, the total audience of eSports has grown up to 454 million viewers with revenue increasing to over US$1 billion [32].

Safe to say, eSports competitions are as popular as traditional sports. With this comes the curiosity for match analysis. Both professional and amateur players, in addition to competitive team coaches, need to retrieve data from previous games to review strategies, get players' behaviour patterns, or understand the opposite team's game plan.

There is already a market for tools that get this information, where some game developers have released their own as well as independent researchers [26], [10]. These tools most of the time only have statistical data (e.g. number of wins per player or team, time per match, which team a player belonged to), although they usually do not provide methods to analyse spatial temporal data. Being eSports extremely popular in MOBA (Multiplayer Online Battle Arena) games, where teams are against each other on a map, it is easy to conclude that having data from each point in time or space is more complete than just one point of data regarding an entire match. Spatiotemporal data does not lose the game's context, which is important when analysts are reviewing a match (or multiple).

VisuaLeague3.0 [10] is one of these tools, created to analyse League of Legends (LoL) games. LoL is a MOBA game where two teams of five players go against each other, and the team that destroys the opponent's base first wins. This tool uses different visualisation techniques, like animated maps, charts and maps with timelines and clustering for spatiotemporal data aggregation. The main goal of this project is to continue to explore

other data visualisation techniques to extract knowledge from spatiotemporal data from multiple videogame matches using clustering algorithms.

1.2 Goals

The main objectives of the thesis can be enumerated as follows:

1. Understand previous work and the study of related work regarding clustering algorithms, visual analytics and visual analysis of videogames.

2. Propose interactive visualisations and dashboards to analyse multiple games or multiple spatiotemporal variables.

3. Test and evaluate the visualisations with competitive players, the target audience of this project.

This work will be loosely based on previous work, specifically VisuaLeague3.0's [10] section of multi-game analysis, where there was a first attempt to develop this type of analytic review, however, the objective is not to continue with the platform, but to have and independent tool focusing on multi-game analysis, while still keeping an interactive approach with the users.

1.3 Contributions

The main contributions of this work can be enumerated as follows:

1. Study and discussion of previous research focused on the relevance of eSports, virtual games and resultant telemetry data gathered granting the ability to create visualisations for different analysis types for different games' genres and respective objectives emphasizing player performance and spatiotemporal data.

2. Development of a series of dashboards that analyse spatiotemporal data using clustering algorithms to extract knowledge regarding team strategies in the videogame League of Legends.

3. A user study with a team of players addressing the effectiveness of the resultant dashboards as a tool for scouting and analysing player performance according to League of Legends' competitive environment needs for each of the available analysis types in multiple matches.

1.4 Document's Structure

This document is organised as follows: in Chapter 2 the related work will be described, including data visualisation techniques, clustering algorithms (the focus of the project), and the state of the art of data visualisation in videogames. Then Chapter 3 presents the used technologies for this project, followed by Chapter 4, which is divided into sections where each one follows the path of building the final dashboards used for this project. After that, in Chapter 5 some user tests are done and their results are provided and discussed. The final discussion and future work of this report will be presented in Chapter 6.

Chapter 2

Related Work

This chapter describes the related work, starting with unsupervised learning algorithms, followed by data visualisation techniques, and ending with some examples of state of the art on the subject of videogame analytics.

2.1 Unsupervised Learning Algorithms

Unsupervised Learning algorithms in Machine Learning are designed to model the underlying structure or distribution in the data to learn more about the data, meaning that no labels are given to the learning algorithm, leaving it on its own to find structure in its input [27]. The goal in unsupervised learning problems is to discover groups of similar examples within the data, where it is called **clustering**, or to determine how the data is distributed in the space, known as **density estimation**.

Clustering

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in that same group than those in other groups, meaning, to segregate groups with similar traits and assign them into clusters [23]. A cluster is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters [27].

Unsupervised learning is then divided into different models, such as:

- **Partitioning** where it constructs a partition of a database D of n objects into a set of k clusters. The partitioning algorithm typically starts with an initial partition of D and then uses an iterative control strategy to optimize an objective function. Each cluster is represented by the gravity center of the cluster (k-means algorithms) or by one of the objects of the cluster located near its center (k-medoid algorithms) [15].

- **hierarchical** creates a hierarchical decomposition of D, represented by a **dendrogram**, a tree that iteratively splits D into smaller subsets until each subset consists of only one object. In such a hierarchy, each node of the tree represents a cluster of D. The dendrogram can either be created from the leaves up to the root (top-down - agglomerative approach) or from the root down to the leaves (bottom-up - divisive approach) by merging or dividing clusters at each step [15].

- and **density-based**. Being this type the focus of this project it will be explained in depth further in the report.

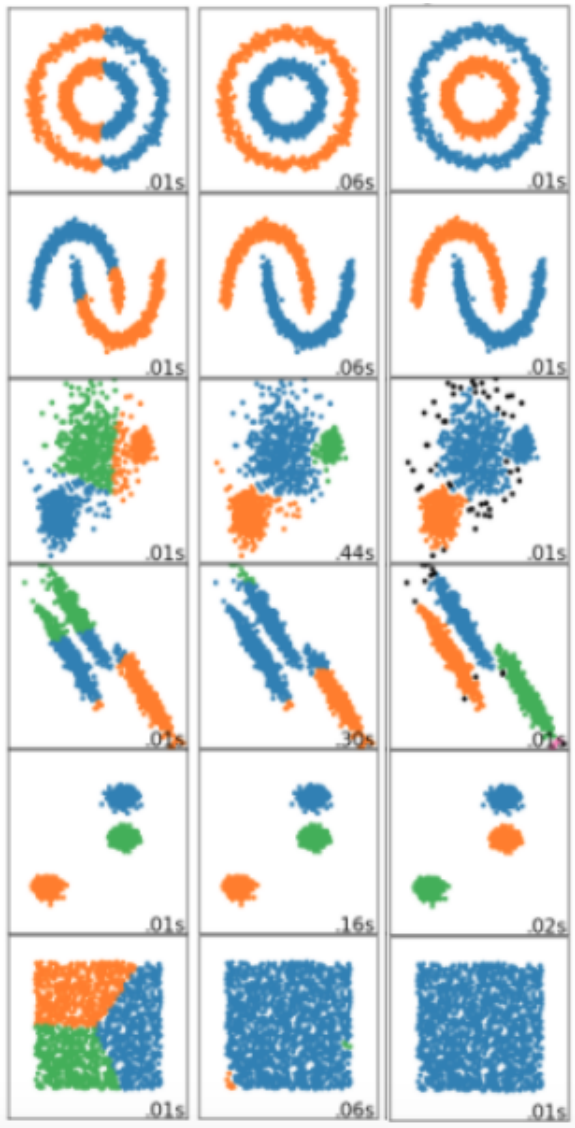

Figure 2.1: A comparison of clustering algorithms from Sclikit-learn [36]. From left to right, K-means, Agglomerative Clustering and DBSCAN

Density-based Algorithms

Density-Based Clustering refers to unsupervised learning methods that identify distinctive groups/clusters in the data, where a cluster in a data space is a contiguous region of high point density [33], separated from other such clusters by contiguous regions of low point density. The data points in the separating regions of low point density are typically considered noise/outliers. The first time a density based approached was explored was to identify clusters in k-dimensional point sets (Jain, 1988) [22].

The reason we are focusing in these algorithms is based on their performance and our intended results. Considering that the datasets to be analysed are very voluminous, efficiency in terms of memory and time is of upmost importance.

Partitioning methods only return convex-shaped clusters (as seen in Figure 2.1), which may be extremely restrictive. The created clusters may change depending on the order in which the data is organized, meaning that it is inconsistent with results. Also, usually requires an initial partition of the data, which means, it requires previous knowledge of the dataset, and in this particular case that is not possible due to size. In terms of memory, for example, CLARANS [29] assumes that all objects to be clustered can reside in main memory at the same time which does not hold for large databases (only works on databases of thousands of objects). On the other hand, hierarchical clustering does not need an input of k (initial partition of data), but instead a termination condition (again, requires previous knowledge of objects to clusters). *Ejcluster* [16] was developed in order to automatically derive a termination condition, that way not needing initial parameter input (and returning non-circular clusters). However the computational cost of Ejcluster is $O(n^2)$, meaning again that these methods are not ideal for the size of data in question.

DBSCAN

Density Based Spatial Clustering of Applications with Noise (DBSCAN) is an algorithm proposed in 1996 by Ester et. al [15].

Next is a listing of the core concepts, definitions and lemmas that are involved in the algorithm.

Definition 1: (Eps-neighborhood of a point) The *Eps-neighborhood* of a point **p**, denoted by $N_{Eps}(p)$, is defined $N_{Eps}(p) = \{q \in D | dist(p,q) \leq Eps\}$.
For each point of a cluster the neighborhood of a given radius has to contain at least a minimum number of points, i.e. the density in the neighborhood has to exceed some threshold. dist(p,q) is the distance function for two points p and q.

There are two kinds of points in a cluster, points inside of the cluster (**core points**) and points on the border of the cluster (**border points**)(see Figure 2.2 as an example).

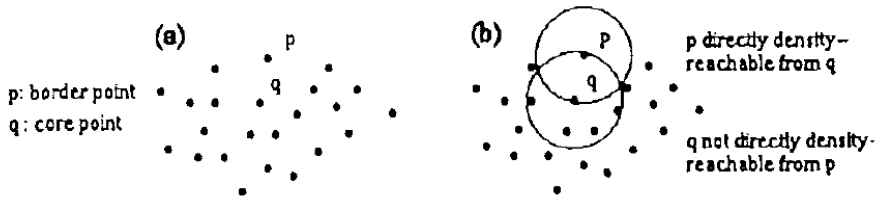

Figure 2.2: Core points and border points in DBSCAN [15]

Definition 2: (directly density-reachable) A point **p** is *directly density-reachable* from a point **q** wrt[1]. **Eps**, **MinPts** if

1. $p \in N_{\text{Eps}}(q)$ and

2. $|N_{\text{Eps}}(q)| \geqslant MinPts$ (core point condition).

For every point p in a cluster C there is a point q in C so that p is inside of the Eps neighborhood of q and NEps(q) contains at least MinPts points (see Figure 2.2).

Directly density-reachable is symmetric for pairs of core points. In general, however, it is not symmetric if one core point and one border point are involved.

Definition 3: (density-reachable) A point **p** is *density reachable* from a point **q** wrt. **Eps** and **MinPts** if there is a chain of points p_1, ..., p_n, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

Density-reachability is a canonical extension of direct density-reachability. This relation is transitive, but it is not symmetric. Although not symmetric in general, it is obvious that density-reachability is symmetric for core points (Figure 2.3 shows an example of this).

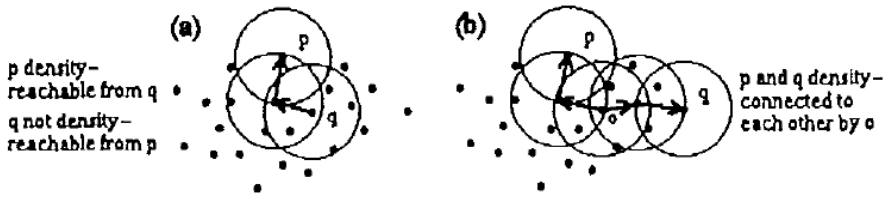

Figure 2.3: Density-reachability and density-connectivity in DBSCAN [15]

Definition 4: (density-connected) A point **p** is *density connected* to a point **q** wrt. **Eps** and **MinPts** if there is a point o such that both, **p** and **q** are density-reachable from **o** wrt. **Eps** and **MinPts**. *There must be a core point in C from which both border points of C are density-reachable (example in Figure 2.3).*

[1] with respect to

Density-connectivity is a symmetric relation. For density reachable points, the relation of density-connectivity is also reflexive.

Definition 5: (cluster) Let **D** be a database of points. cluster **C** wrt. **Eps** and **MinPts** is a non-empty subset of **D** satisfying the following conditions:

1. $\forall p, q$: if $p \in C$ and q is density-reachable from p wrt. Eps and MinPts, then $q \in C$. (Maximality)

2. $\forall p, q \in C$: p is density-connected to q wrt. EPS and MinPts. (Connectivity)

A cluster is defined to be a set of density-connected points which is maximal wrt. density-reachability.

Definition 6: (noise) Let C_1, ..., C_k be the clusters of the database **D** wrt. parameters $\mathbf{Eps_i}$ and $\mathbf{MinPts_i}$, i = 1, ..., k. Then we define the *noise* as the set of points in the database **D** not belonging to any cluster C_i, i.e. noise = $\{p \in D | \forall i : p \notin C_i\}$.

Noise is the set of points in D not belonging to any of its clusters.

Lemma 1: Let p be a point in **D** and $|N_{\text{Eps}}(p)| \geqslant MinPts$. Then the set **O** = {o | $o \in D$ and o is density-reachable from p wrt. Eps and MinPts } is a cluster wrt. **Eps** and **MinPts**. *A cluster C wrt. Eps and MinPts is uniquely determined by any of its core points. However, each point in C is density-reachable from any of the core points of C and, therefore, a cluster C contains exactly the points which are density-reachable from an arbitrary core point of C.*

Lemma 2: Let **C** be a cluster wrt. **Eps** and **MinPts** and let **p** be any point in **C** with $|N_{\text{Eps}}(p)| \geqslant MinPts$. Then **C** equals to the set **O** = {o | o is density-reachable from p wrt. Eps and MinPts}.

Pseudo-code

Next is a simplified pseudo-code of the algorithm. For the complete algorithm, see Appendix A.

Given the parameters Eps and MinPts, we can discover a cluster in a two-step approach. First, choose an arbitrary point from the database satisfying the core point condition as a seed. Second, retrieve all points that are density-reachable from the seed obtaining the cluster containing the seed (as seen in Algorithm 1). In the appendices are the extended pseudo-codes representing these two phases (Appendix A).

Runtime

Let us consider a database consisting of *n* points and the Eps-neighborhood of a point is

Algorithm 1 DBSCAN(Simplified)

```
for all o ∈ D do
  if o is not yet classified then
    if o is a core-object then
      collect all objects density-reachable from o and assign then to a new cluster
    end if
  end if
end for
```

a list of points. If we use and efficient way to access this list, let us assume it takes on average O(log n) to traverse this list for all n points(since the region is expected to be small compared to the whole database), so the average runtime complexity of DBSCAN is $O(n \log n)$.

Optimal values of Eps (ϵ) and MinPts

When looking at examples of DBSCAN, the values of the parameters *Eps* (radius of the neighborhood) and *MinPts* (minimum number of points in said neighborhood) are chosen via trial and error. This doesn't work for this project, as we want the algorithm to recalculate every time we change the data points to cluster. This means that we can't use a single value for these parameters, since they may not always be optimal. When we focus on *MinPts* there are not a lot of options. In [35] concludes that $minPts = 2 * dim$, 4 in this case. The authors have proposed an optimal way to calculate *Eps*, using the point of maximum curvature, also known as *knee* or *elbow*. Adapting to this project, we plot of the k-nearest-neighbor distances (using the library *sklearn.neighbors*). Usually we could visualize the plot and choose the knee value, but again this is not ideal. So we use the library kneed that gets this value from a plot. where $k = 2 * dim - 1$, in this case 3.

2.2 Data visualisation

Data visualisation is the graphical representation of information and data [39]. By using visual elements like charts, graphs and maps, data visualisation tools provide an accessible way to see and understand trends, outliers and patterns in data. In the world of Big Data, data visualisation tools and technologies are essential to analyse massive amounts of information and make data-driven decisions.

2.2.1 Visual Game Analysis

As mentioned previously, analysis of videogame data can be used for analysis of patterns or strategies post-match by players, coaches or analysts or for playtesting [34]. With competitive gaming on the rise, there is also a growing market for tools for e-sports team coaches to see other team's strategies and mitigate their own team's weaknesses.

The most widely available videogame analysis tools still revolve round statistics and static data, as in the example of OP.GG [2], a tool for LoL matches. Some examples of static data representation are **choropleth** maps (Figure 2.4), where a colour is associated to a specific location, and **heatmaps** (Figure 2.5) that aggregate that information and represent it with a gradient [9].

Although easy to access and extract some knowledge from, one user study found that compared to some dynamic data extraction tools, statistics are sometimes not enough to get a grasp of the game context and to evaluate a player's performance, since they lose the game's context and don't allow a temporal analysis of the data.

Some other existing tools for analysis exist, like replay systems for matches, but most are restricted to a certain game and/or are privately owned, like FIFA 20, PUBG and LoL [9].

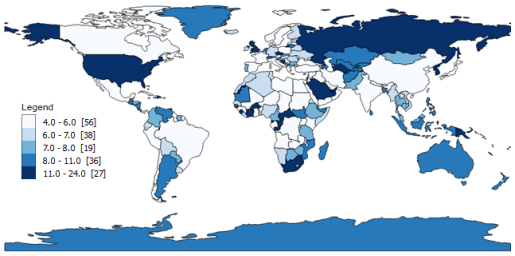

Figure 2.4: An example of a choropleth map [18]

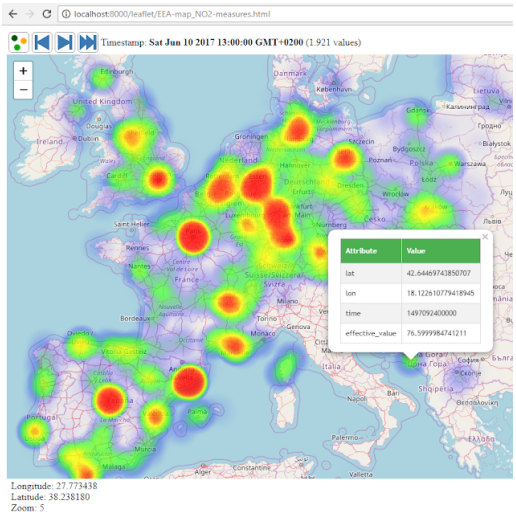

Figure 2.5: An example of a heatmap [1]

2.2.2 Spatiotemporal Data Visualization

Spatiotemporal data analysis can be represented, for example, in **spacetime cubes** (Figure 2.6), where using solid lines, segmented lines or point symbols over the geographical representation, and the widths or colours may encode movement information/attributes, or **animated maps**, that are singular images in a sequence, enabling the analysis of the variation of events over time in the geographical area [9]. Analysing games with this technique may be beneficial since it doesn't lose the context (can use in-game map or graphics) and, as the name suggests, allows a temporal analysis. This means that we are able to extract knowledge in various points in time and evaluate a player or team's performance throughout a match, allowing for a better pattern recognition, which is the goal for coaches or analysts.

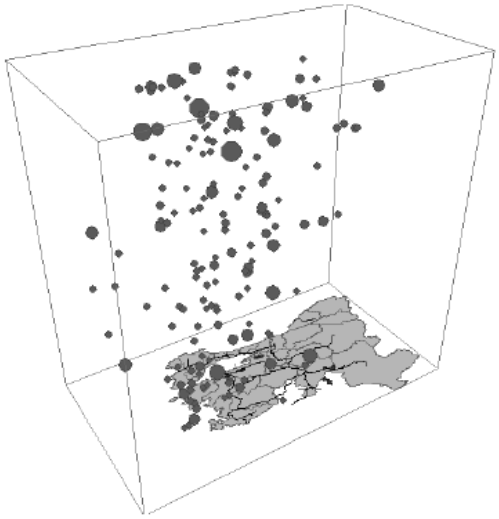

Figure 2.6: A spacetime cube. z-axis is the time when an event occurred and x and y-axis are the position in space of the event [17].

An example of a tool that allows this type of examination is **Echo**, developed by MacCormick et al. [26], a Unity [40] plugin used to record game data for later review and examination. It records temporal data of dynamic and static objects of the user's choosing, recording their status, location and appearance, in addition to camera angles. It then represents the chosen objects in a 3D landscape and the user is able to watch a match unravel (see Figure 2.7). This way, there is no loss of context since the objects are represented physically so it's easier to understand what data are we looking at. The user study concluded that Echo is preferred to screen recording in terms of gameplay analysis; It is less stressing and troublesome to the users and more versatile to their work style and speed, thanks to the ability to stop, fast-forward or rewind the time stamp and the different data that can collect.

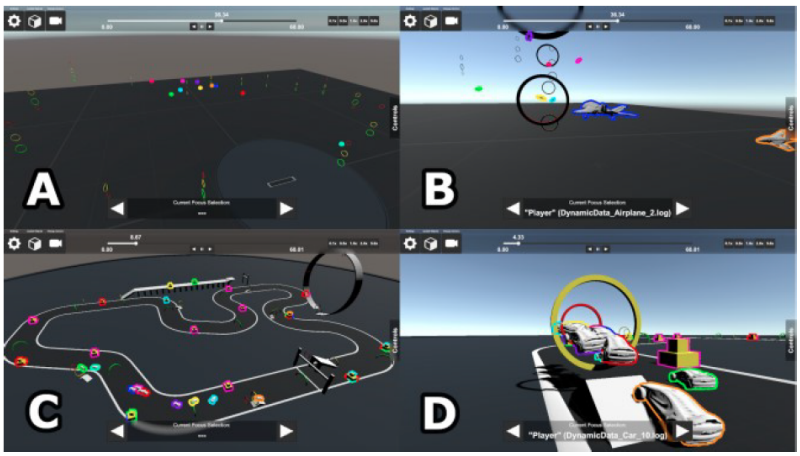

Figure 2.7: A car (top) and plane (bottom) game viewed with Echo, with left: game from afar and right: game when focused in on a player [26].

Wallner et. al [44] explored both aggregated and non-aggregated visualisation techniques in the context of playtesting. Using the game Infinite Mario (a procedurally generated platformer inspired by the Nintendo game *Super Mario Bros.*), the authors created seven static levels and collected in-game, physiological and observational data from six playtesters. Then, they displayed the data both non-aggregated and using aggregation techniques and derived a user study, concluding that compared to the individual data visualisation, the aggregated visualisation has significantly better readability (due to a less cluttered screen), is considered more useful for identifying areas of struggle, major paths, and scales better for larger datasets. We can see in Figure 2.9, they used:

1. **Clustering of discrete events** using the clustering algorithm DBSCAN to cluster the observational data and then represent the cluster with the symbol of the event and its size representing how many events there are in the cluster;

2. **Trajectory tessellation** to separate areas of the level, using a choropleth map to colour the average arousal value within each cell and the transparency to correlate to more or less traversed areas;

3. and **Trajectory aggregation**, reusing the previous trajectory tessellation and count how often a player crosses the border from one cell to another to draw the used trajectories and the thickness of the line representing amount of movement.

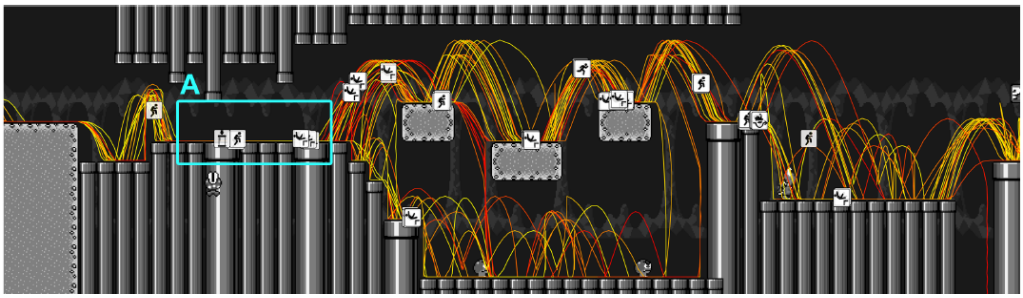

Figure 2.8: Non-aggregated visualisation of player's performance in Infinite Mario [44].

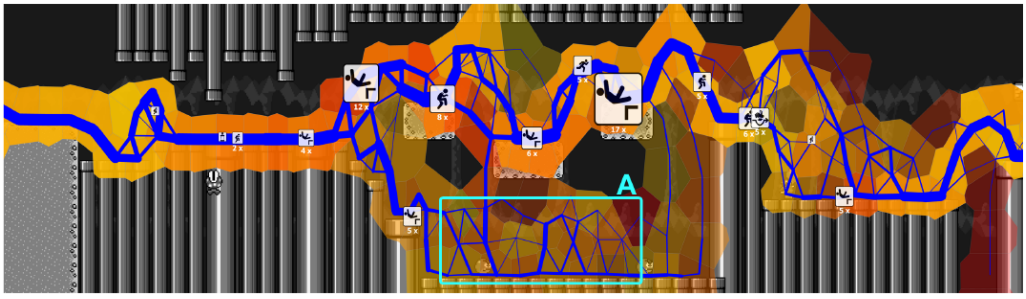

Figure 2.9: Aggregated visualisation of the data depicted in Figure 2.8 [44].

In regards with just the spatial data visualisation area, Wallner [43] proposed a way to merge individual troop movements into combined flows in order to improve the readability of troop movements, their splitting and merging, and their overall strength in the context of battle maps. Focusing on the game *World of Tanks* [41], an online warfare game where two teams of players compete against each other, with each player controlling a single tank. By using a flow tree (as shown in Figure 2.10), with the direction of the arrows representing the direction of movement and the splitting of the branches represented is the landmark where it occurs, as well as the thickness of the arrows correlating to the size of the troop, this representation reveals a valuable tool for players to reflect on their gameplay. It was mentioned in the paper that this tool lacks in temporal analysis, as it only takes into consideration the position of the players, and sometimes merging movements may take place at very different points in time, but the resulting flow tree has no way to express this.

Figure 2.10: The resulting flow tree regarding one match played on the map *Cliff* in *World of Tanks* [43].

Camilo et. al [12] also applied data exploration in spatiotemporal datasets. They used pattern analysis to build dashboards and analysing the data referring to emergencies in the Lisbon area the that require the help of the Firefighters, helping with quick responses and area forecasting with more incidences. They also carried out an experimental study and concluded that the existence or not of previous experience with visualisations does not make a significant difference in the success rate of the using the tool, and also that most of the reproduced dashboards included intuitive graphs that do not require previous knowledge from the user to interpret them. Although not in a videogame context, her work shows how to use tools such as Tableau Desktop to design views and Dashboards that use spatiotemporal datasets to observe patterns and make assumptions about the data's

behaviour.

2.2.3 VisuaLeague and League of Legends

This subsection will first describe briefly the videogame League of Legends [31], followed by the analysing tool VisuaLeague 3.0 [10].

League of Legends

League of Legends (Riot Games [21]) is a team-based strategy game where two teams of five players face off and destroy each other's base.

Each player, referred to as a summoner, can choose a character (champion) from a roster with already over one hundred possibilities and continuously increasing, each one with different strengths and weaknesses, representing a different playstyle and experience. The game offers different game modes and maps, but we are going to focus on the Summoner's Rift.

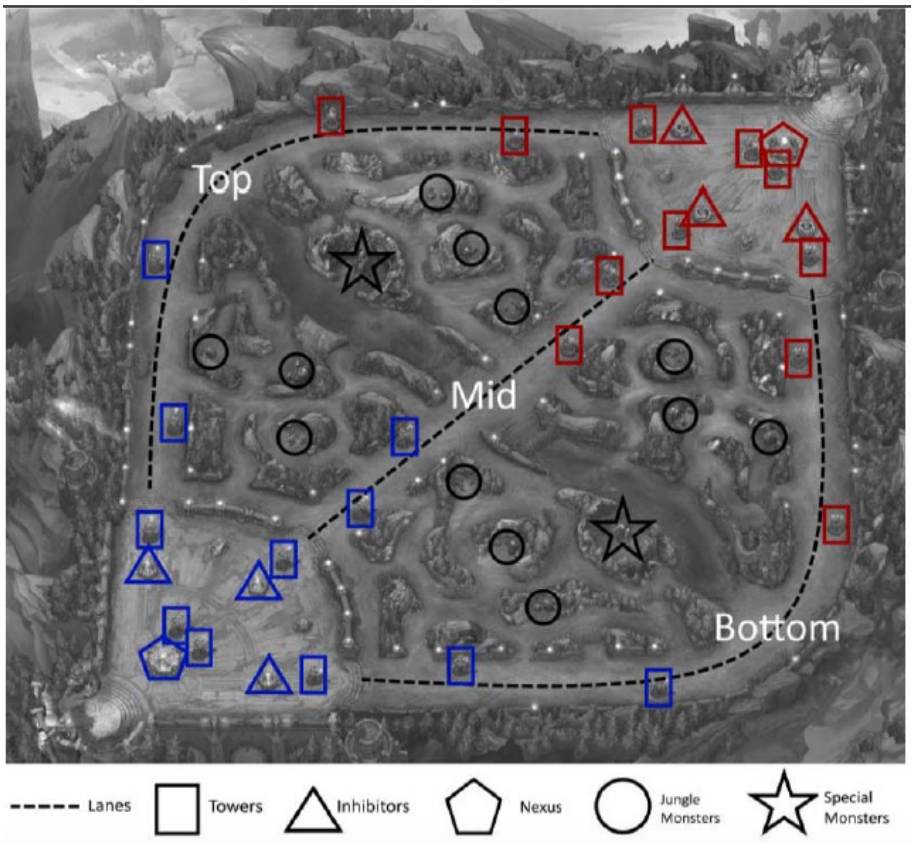

Figure 2.11: League of Legends Summoner's Rift map with important locations [19].

Summoner's Rift is the standard MOBA mode (and the name of its map) and the only one featured in professional level play. A match takes place in a squared map (Figure 2.11) composed of three **lanes** (top, middle, and bottom) that connect the player's base

to the enemy's. To win a game, the player needs to push down their lane into the enemy base and destroy the main structure (**Nexus**) at the center of their **base** (down-left and top-right corners of the map). Between the lanes lies the Jungle, an area filled with neutral monsters, called mobs. These monsters will not attack the enemy base, but they can be defeated for additional gold and the champion gains temporary bonuses (e.g, Dragons and Baron). Periodically, AI-controlled soldiers (**minions**) spawn at the Nexus and march down each lane towards the enemy base, attacking enemies they encounter along the way. When minions die, they give champions experience points (XP). **Towers** are defensive structures that protect each lane at specific intervals, hitting enemies that come within range. Usually tower prioritize minions, but if a player attacks another player, they will be the target instead. Some other structures, known as **Inhibitors**, are located where each lane meets the base on both sides of the map. Players see the full map, but only have updated vision in locations with an allied minion, structure, player, or some special ability, item or when a ward is used.

Before a match starts, each player can use a unique champion to fulfill a specific role in the team: one player goes to the top lane (Top) and another to the middle lane (Mid), two players go to the bottom lane (Attack Damage Carry (ADC or Bot) and Support) and the remaining player goes to the jungle (Jungler). However, this is the recommended method and different strategies can result in a change of roles. After this the match begins, with each team starting on their own base, on opposite sides of the map.

Normally, a match's gameplay can be divided into three stages: **early-game** (until 15 minutes), where players mostly stick to their lanes; **mid-game** (until 25 minutes), where the team starts grouping to help push a lane and helping allies by invading enemy lanes; and **late-game** (until the end), where team fights are predominant, trying to invade the enemy base. The specific time for each phase can change between version of the game, but the three game stages are always present. Summoner's Rift matches typically last between 30 and 40 minutes, until a Nexus is destroyed.

Next let us focus on the competitive environment of LoL. There are two perspectives: solo queue and team competitions.

Solo queue

This first one is accessible to everyone, where a summoner plays various matches with random players in order to go up in the rank system. This ranking system shows player competitive level as an individual. A player's rank is determined by their number of victories over defeats, League points (LP), with and additional calculation named Match Making Rating (MMR) that helps determine the player quality and if they are in the correct League. After earning 100 LP, a player must win 3 out of 5 matches in a trial in order to go up a rank. The MMR helps this by making it easier or harder to get points depending on the rank the player is currently. Each time they go up or down a rank, they

will play against others with similar ranks. The rank system basically allows player to show off their skills and allow them to play against other with similar experience. As a final note, if a player desires they can play against random players unranked, where they will be assigned with an unranked placeholder.

Team Competitions

The main difference between this mode and Solo queue is that each match is created by a player with availability restrictions, making it possible for only specific players to join the match. Teams playing at a competitive level are usually composed of five players with high solo rankings and a variant amount of substitutes that practice together in custom games (*scrims*). A team's constitution may vary and members are constantly changing, not only players but other staff such as coaches or analysts, and also players can having more than one account and change them while playing competitively. This does create an issue regarding having access to team information. To solve this issue, coaches usually make manual annotations while watching a match, or save the match data (Id, date, time) in order to search for it later on other web applications (mentioned in 2.2.1 and 2.2.3).

Riot's API (Application Programming Interface) does not keep track of this type of data, so in this project we are not going to focus on matches played by the entire team, but rather the behavior of each player. It would be ideal if we could get a list of team matches, but since Riot API saves the matches in chronological order and only saves up to 100 matches, that would only be possible if the team plays exclusively with each other, which doesn't happen in reality.

VisuaLeague3.0

Afonso [10] developed VisuaLeague3.0, a prototype for analysing League of Legends matches. By specifying a player's Summoner (a player's username/ID) and filtering by number of games, type or champion (in-game character) used, VisuaLeague displays a series of tabs regarding data from the matches selected: "Selected Player", "All players" and "Map & Timelines" respectively.

Regarding the last one, in Figure 2.12 we see that in the center there are two maps, each one filtering a different data (Left: Maps with kill events and Right: tower destruction events). The maps show the position of major events and players' positions over matches. On the left of the screen there is a series of filters that can be enabled of disabled according to the goal of the analysis, such as showing data for a specific player or all players, showing or hiding wins or defeats, with events in the game to display (tower destruction, kill). By clicking with the mouse on an event in the map, the user can see which players were involved in the event. Bellow each map there is a timeline with a slider with timestamps for every 5 minutes plus the 8 minute mark. This filter can be activated and

deactivated at will, showing the data for that timestamp and the 30 seconds before and after (if activated) or for the overall match (if deactivated).

Figure 2.12: Multi-match analysis in VisuaLeague3.0 [10].

Concerning the use of clustering algorithms to aggregate data, there is also a filter, ”group points”, which changes the view of the map between individuals points or clusters. Since the prototype is using DBSCAN (mentioned in Section 2.1), there is a need to specify the parameter Eps, which in VisuaLeague is done by using a slider to choose the radius of the neighborhood. By doing this, the user can experiment and see which radius is ideal for the amount of data/matches chosen to analyse (see resulting maps in Figure 2.13).

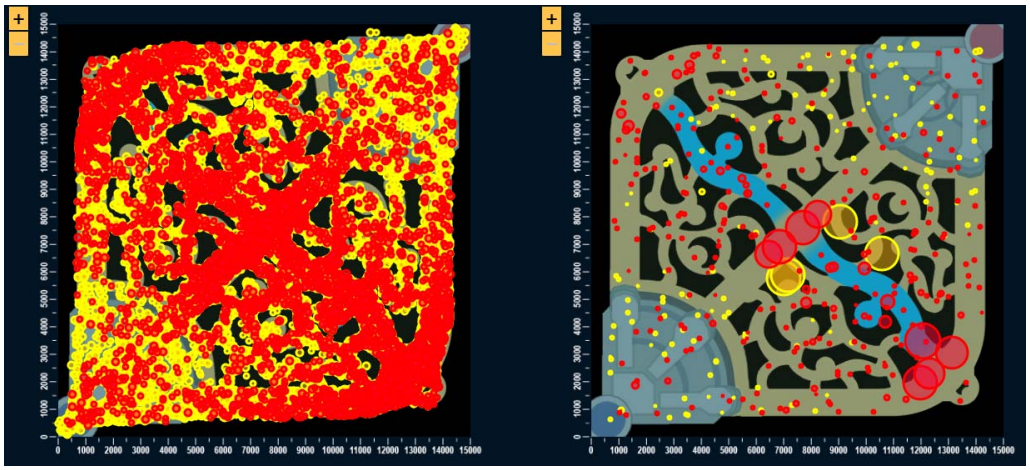

Figure 2.13: Player analysis for multiple games (one hundred games). Left: Individual positions. Right: clustered positions, in VisuaLeague3.0 [10].

2.3 Summary

Unsupervised learning is used to model the underlying data structure with no previous knowledge on labels, which is and advantage for Big data because this way there is no need to prepare training sets, which would be impossible. One unsupervised learning technique is clustering, recommended for distributing data in the space. Density-based clustering helps group datapoints with similar characteristics between each point of a same cluster, and the goal is that the members of a same clusters are more similar to each other than members of different clusters. Many algorithms of this type have been developed but due to the fact that this project works with larger amounts of data, we chose DBSCAN, which applies density-based clustering with consistent results while also ignoring outliers from the clustering, as well as having a reasonable computational cost. This algorithm requires super-parameters ϵ and *MinPts* that can be calculated on runtime.

Regarding spatiotemporal data visualisation, the related work has many user studies that applied these techniques to videogame analysis. All studies' conclusions go in the same direction: an aggregated, spatiotemporal approach is less stressful and faster to use, with better readability, no loss of context and better in general for large amounts of data.

In terms of platforms to create dashboards with the desired visualisations, Tableau Desktop [24] was chosen. Camilo's work [12] also used Tableau for visual analysis, and their user study concluded that is easy to use even for people with no experience and provides useful results.

To finalize, the related work describes VisuaLeague 3.0 [10], a tool for analysing League of Legends matches. The future work mentioned on this project suggests further study in clustering and spatiotemporal analysis on multiple LoL matches, in a way to make more relevant conclusions to a player's behaviour during their playtime. In this section we also have a small summary of League of Legends. LoL is a MOBA game where two teams of five players battle in a map, and the goal is to destroy the opponent's base. It has two main modes: Solo queue and Team competitions, with the first allowing player to go up the rank system and show off their skills, playing with and against random players in the same rank, and the last the matches played with specific players of their choosing. For this project, the game mode is not relevant, even though the user study (later in Chapter 5) has participants with experience in both modes.

Chapter 3

Technologies

This chapter describes what technologies were used while developing this project, those being Datalore [20] for the development of Python code and data manipulation and extraction and Tableau Desktop [24] for the development of visualisations.

3.1 Datalore

Datalore is an online environment tool for Jupyter [5] notebooks from JetBrains that enables users to execute and share code more productively. Datalore is used for data collection and exploration, machine learning, deep learning, and interactive visualisation [38]. In Datalore, all the computations are performed in the cloud. To run Python and Kotlin code, it is sufficient to open a browser, register at datalore.jetbrains.com and create a Notebook with no additional setup required, since the top data science libraries are already pre-installed in Datalore. Datalore was designed to help data scientists and analysts complete their day-to-day tasks, namely:

- Collect and explore data,
- Create machine learning and deep learning models and
- Visualize results and share them with others.

One of the features of Datalore is a smart coding assistance, which includes code completion, quick fixes, auto-imports, inspection and refactoring options, tips, rename, and reformat options, helping make a coding experience more productive. Some other features are:

- Distraction-free mode and 'Split view' option,
- Cell-toolbar to quickly add Markdown and Code cells and
- Cell context-menu to generate a table of contents, access the Variable viewer, etc.

Datalore also allows its users to share their work with others, track all progress using the built-in version control system, and provide comments in text cells with Markdown and LaTex support. There are several ways to collaborate with other users using Datalore:

- Sharing Notebooks to work with a team in real-time,
- Publishing Notebooks and share them by a link to receive comments,
- Sharing whole workspaces with multiple Notebooks and Datasets, and
- Publishing PyCharm Notebooks using the Datalore plugin for further editing and collaboration with a team.

In short, Datalore is an online platform that works with notebooks, that is designed for data scientists, with ready-to-use data science tools with code assistance and editing in the cloud that allows sharing and collaboration between teams [38].

Datalore is not completely free (pricing going from $19.90/month for bigger storage, more collaboration options and greater monthly machine used and computation power) but provides a free version to every registered user, which is the one used for this project [37].

3.2 Tableau Desktop

Tableau Desktop [24] is a software for users to create visualisations and dashboards for data analysis, allowing them to save the results and share them with other people. This software is not free (with a monthly cost of 70$, which includes access to all other Tableau products like Tableau Prep Builder, Tableau Server and Tableau Online), but for this project we used a student license which grants access to all Tableau products for a year.

To use Tableau Desktop we start by connecting it to our dataset. This can be in the form of local files or in a server (Tableau Server, Dropbox, MondoDB or Google Drive just to name a few) and the files could be a CSV (Comma-separeted value), Excel, text files, PDF, JSON and many more. In this case we connected to multiple local CSV files which contain all the user and match data we need. If adding multiple files, it automatically suggests a relation between files (similar to connecting relational databases), in order to help the user build visualisations in the future, that is, it identifies common column names and types (they can be adjusted by the user if needed) such as string, number, boolean or date. After that, it separates its data into **dimensions** (qualitative) and **measures** (quantitative). This can also be adjusted by the user if identified incorrectly. This distinction is necessary because measures can be aggregated and calculated upon (e.g. maximum and minimum value, average, sum) and it helps make sure that the developed views make sense from a data point of view. After connecting the data, we can start building views. Tableau Desktop has a drag-and-drop feature where the users select the measures/dimensions available

on their dataset on a list to the left of the screen, and drag them to the desired axis to form views. Without any need for writing code we can change aggregations, filters, orientation of the axis, type of charts (bar, line, map, pie, gantt bar, etc.), colours, labels and size, among others.

The user can also add their own calculations as fields. The advantage to this opposed to calculating these fields before adding the files to Tableau (in a preprocessing phase) is that it updates its results every time we update the data by filters. For example, it would be interesting to know the average times a players dies during a game. If we were to calculate this outside of Tableau, we would need to make a calculation for every match available (at this point, we know that means over 100 matches). If then we use that data on Tableau, and later one user decided to filter that data somehow (for example, filter the positions that occur in the first 10 minutes), the results are not accurate. Another example is the aggregation of the last 5 matches, or only the matches were a certain team wins. So we can see that there are so many iterations of the same calculation, that the intractability of the tool would be compromised. This is why there is an effort in calculating fields and applying algorithms and aggregations in Tableau, since then even if we refresh the data set, all the calculations will make sense and be accurate.

One of the goals of this project is to apply clustering algorithms to the dataset, and looking at the options provided by Tableau Desktop, the "clustering" option in the Analytics section is very ambiguous in what method or parameters it uses, so we decided to add a clustering field manually. This means that we are going to use TabPy [25], which will be described in the next section, in order to write a Python script that applies the DBSCAN algorithm to a set of measures (X and Y positions on a map) and returns another measure with a list of the designated cluster each position is aggregated with.

3.3 TabPy

Tableau can work with Python scripts, provided that it has a connection to a TabPy Server [25]. The server receives the script as a string, followed by the aggregated fields of the dataset that we want to use as "arguments", and then returns a field with whatever the scripts returned. Depending on the type of script we use (**script_int, script_string or script_bool**) the returning field has a different data type.

In the DBSCAN case, we wrote the script and added the X and Y coordinates of the player's positions, to return the Id of cluster where each position is allocated, as seen in Figure 3.1. To calculate the optimal ϵ we also needed to add the library **KneeLocator**.

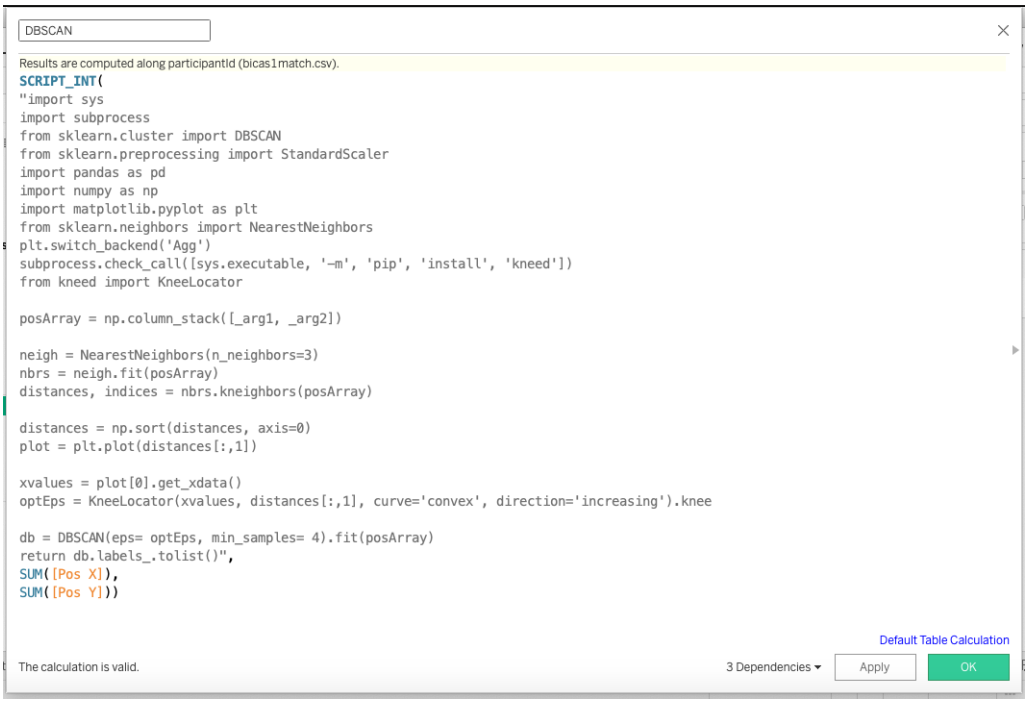

Figure 3.1: Script in Tableau Desktop to apply DBSCAN to player's positions

To compile and execute any scripts we may have, we need to do the following:

1. Install TabPy on the machine's shell and initialize it (follow instructions available on Figure 3.2);
2. On Tableau, select Help > Settings and Performance > Manage Analytics Extension Connection... (see Figure 3.3);
3. On the pop-up window that opens, select the connection type "TabPy" and fill the required information (by default, it should be localhost on port 9004) and click "Save" (see Figure 3.4).

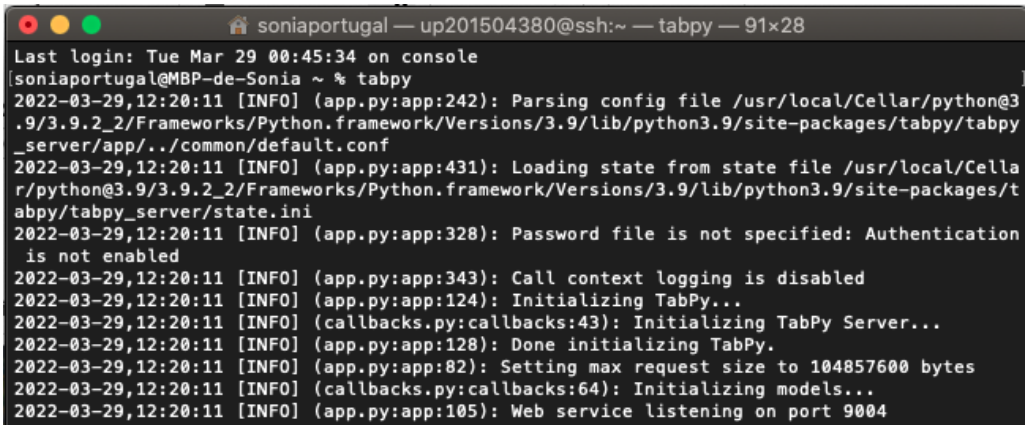

Figure 3.2: How to initialize TabPy on Shell

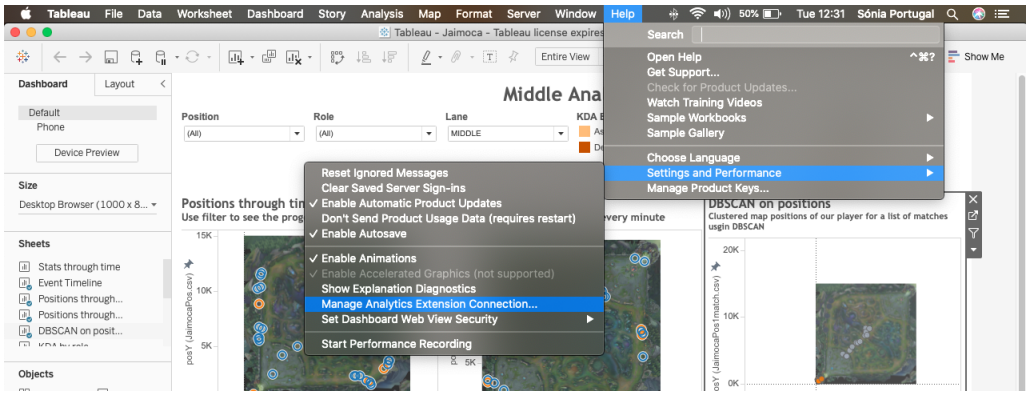

Figure 3.3: How to open Connection options on Tableau

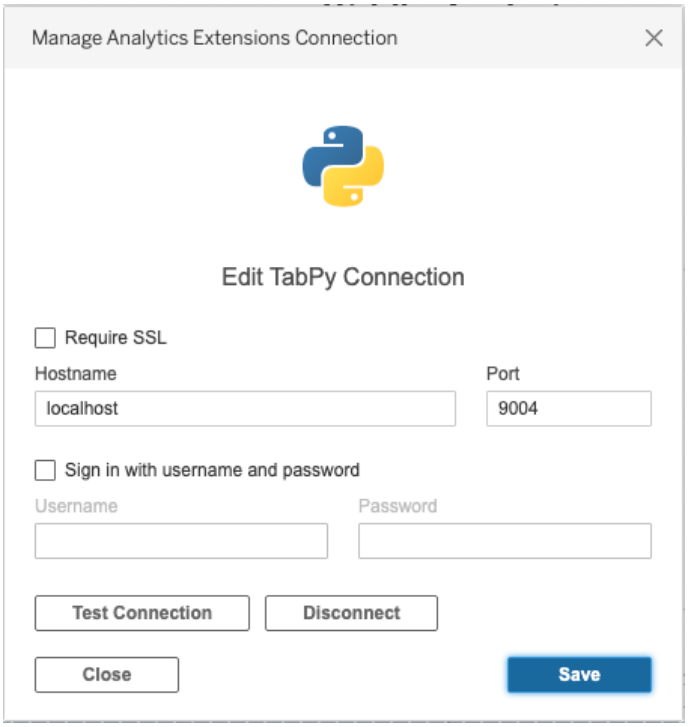

Figure 3.4: TabPy connection settings on Taleau

The Tableau page then reloads (it might take some time depending on how many scripts need to be compiled and executed). If any errors arise, a pop-up window appears with the error messages, or you can check your shell for more detailed information.

Here we encountered some issues. Since TabPy uses only aggregated measures as arguments for the script, and error occurred while calculating the optimal *Eps*, since it would received only one value per coordinate (for example, the average of X and average of Y), so it could not calculate the k-nearest neighbours, although when using a standard value, the DBSCAN would not have any issues running. This was solved by disabling an option (Analysis > Aggregate Measures at the top of the screen) that prevents automatic aggregation on measures. The problem then was that if we wanted to apply a filter with

a dimension (quantitative datapoint), such as a date filter, sometimes an error occurs and it cannot load the result of the script. This is obviously an issue, since later on while building dashboards we need to make sure that other filters used in other views do not "corrupt" the DBSCAN map. Although it would have been ideal if this did not happen and users could interact with the DBSCAN with multiple filters. The same principle was applied to events, where a map is created with clusters of positions where events occur in a match (only some apply such as champion kill, for example).

Chapter 4

Approach

In this chapter we are going to detail our approach to this project. After introducing the technologies used in Chapter 3, we are going to follow the general way of approaching any machine learning project, as illustrated in Figure 4.1. Starting from left to right, we are going to describe first how we extract our dataset, using the Riot API, then preprocess it (Data Extraction & Preprocessing) in a way that makes it ready for the next step (Knowledge Extraction), where we are going to use said dataset to build views, which will be shown in dashboards in Tableau Desktop (Visualization). To note that the Evaluation step will be described in Chapter 5.

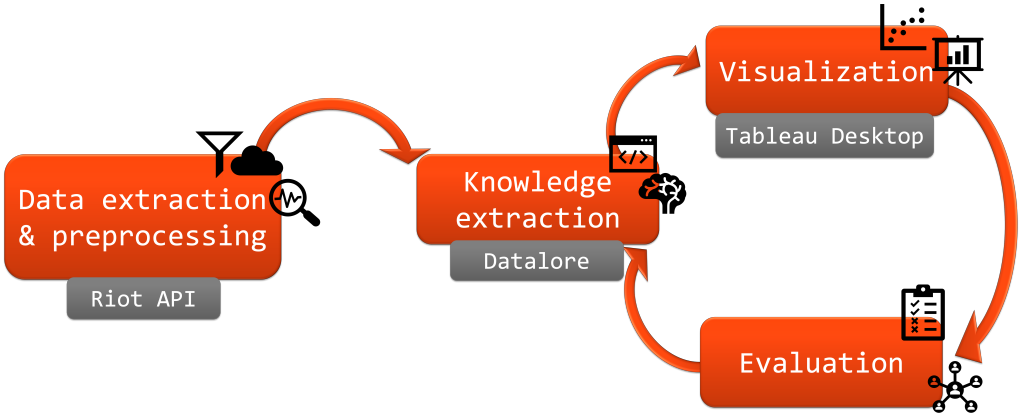

Figure 4.1: Methodology of this project

4.1 Data extraction & Preprocessing

Riot Developer Portal

Riot Games, the developers of League of Legends, developed an API [7] with free access to a lot of information regarding LoL, such as Service status, Summoner details (name, level, profile icon, account ids), Match History, Match details and timelines by Summoner name, Someone's current game info (if someone is in the game) and more. To access the

Riot Games API, a user always needs a key. For that, all we need is an existing (or create a new) Riot account. After logging in, we will see the DEVELOPMENT API KEY, which expires every 24 hours (for this not to happen, we can register a personal Riot product, to get a PRODUCTION API KEY). The upper limit of requests for a development API key and personal product key is:

- 20 requests every 1 second;
- 100 requests every 2 minutes.

Next in Listing 4.1 is how to use Riot API and Python to make a request to fetch the last match's information from an account, showing the resulting DataFrame object in Figure 4.2. To note that we are using the **riotwatcher** library (line 1), as explained later. Also we need to pick a region (line 7), since the API is regionalized, so Summoner and account IDs are only unique per region.

```
1  from riotwatcher import LolWatcher, ApiError
2  import pandas as pd
3
4  # global variables
5  api_key = 'RGAPI-xxxxx'
6  watcher = LolWatcher(api_key)
7  my_region = 'na1'
8
9  # see account information
10 me = watcher.summoner.by_name(my_region, 'Doublelift')
11
12 my_matches = watcher.match.matchlist_by_account(my_region, me['
       accountId'])
13
14 # fetch last match detail
15 last_match = my_matches['matches'][0]
16 match_detail = watcher.match.by_id('AMERICAS', last_match['gameId'])
17
18 participants = []
19 for row in match_detail['participants']:
20     participants_row = {}
21     participants_row['champion'] = row['championId']
22     participants_row['spell1'] = row['spell1Id']
23     participants_row['spell2'] = row['spell2Id']
24     participants_row['win'] = row['stats']['win']
25     participants_row['kills'] = row['stats']['kills']
26     participants_row['deaths'] = row['stats']['deaths']
27     participants_row['assists'] = row['stats']['assists']
28     participants_row['totalDamageDealt'] = row['stats']['
       totalDamageDealt']
29     participants_row['goldEarned'] = row['stats']['goldEarned']
30     participants_row['champLevel'] = row['stats']['champLevel']
31     participants_row['totalMinionsKilled'] = row['stats']['
       totalMinionsKilled']
32     participants_row['item0'] = row['stats']['item0']
33     participants_row['item1'] = row['stats']['item1']
34     participants.append(participants_row)
```

```
df = pd.DataFrame(participants)
df
```

Listing 4.1: Example to fetch account information and match data

[74]:

	assists	champLevel	champion	deaths	goldEarned	item0	item1	kills	spell1	spell2	totalDamageDealt	totalMinionsKilled	win
0	6	12	62	6	8298	1036	2031	2	11	4	104001	34	False
1	4	15	58	5	11954	3053	3153	5	4	12	126899	218	False
2	1	13	145	3	9879	1055	3006	3	7	4	116909	198	False
3	2	11	111	7	5537	2055	3860	1	4	14	19054	33	False
4	5	13	68	7	8072	3020	3151	2	14	4	94328	136	False
5	14	13	555	0	11023	3117	3179	3	4	14	31552	44	True
6	4	15	81	2	14497	1055	3153	12	4	12	167204	246	True
7	10	15	39	5	11418	3153	2033	3	4	12	137770	204	True
8	13	14	60	5	10769	1414	3115	4	4	11	154551	18	True
9	8	15	7	1	11035	2033	3285	6	4	14	104808	174	True

Figure 4.2: Output example of the request made in 4.1

The development key must be manually changed in the source code. The limit on the requests can become restrictive since multiple requests are required to extract all information needed for the analysis of a single match. So analysing aggregated matches will become challenging as the queries scale when searching for multiple games from multiple players.

Riot API separates data into different requests:

- Requests with a player's summoner name to retrieve the accountId,
- Requests for a match-list (list of matchIds), with a specific number of games, from an accountId,
- Requests for statistical information for a matchId, and
- Requests for spatiotemporal data for a matchId (we would not be talking about rank data in this project, but to note that rank information per accountId is also available).

The main queries utilized in the project follow the approach of identifying an accountId since this project focuses on a player analysis. With the accountId is possible to extract a match list that contains a defined number of the player's most recent matches (up to one hundred matches in a single match list request), in a very summarized way. Through the game id acquired from that match list, it is possible to request match data, where general statistics of that game are described like total gold for each team or total kills for a player. In a separate request (with the same match id) called timeline data, it is possible to get spatiotemporal data with information specific to players and events positions in certain time frames. The three main datasets created for the project are explained with the data process further in this Section.

To access the API data, it is necessary to do a request. In Python, we can do that using a library named **riotwatcher** [8]. There are multiple libraries available, depending on the programming language used or the kinds of requests needed. In [6] there is a list of the available libraries. We chose riotwatcher because the other options either did not work properly (cassiopeia's example would not run), could not find the library in the programming environment used (pyot), or the requests restricted the type of data available (pantheon does not support static data).The **riotwatcher** library needs the development API key retrieved from Riot Developer Portal as an argument to every request. All responses are JSON objects, which are easy to manipulate in Python since they are equivalent to dictionaries.

Preprocessing

One of the difficulties of this project is that we are dealing with large amounts of data. This is a problem in computer science because even though it is great for works where the more information the better, like knowledge extraction or prediction in machine learning, Big Data is hard to handle, since we have to consider security and growth of information.

In any Machine Learning process, Data Preprocessing is a step in which the data gets transformed, or encoded, to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm [30].

In this project, we can separate the preprocessing stage into two perspectives: one that is independent of the analysis we want to make and another which is dependent on the data. The first one guarantees that the data is stored correctly and easily read by any machine, and the second ensures that only data that is relevant for our research purposes is kept, and that is organized in such a way that the desired analysis is as optimized as possible.

The one that is independent of the analysis can be broken down by the following [30]:

- **Data Quality Assessment**: there is always a chance that large amounts of data have mistakes (either from human error, limitations on measuring devices or flaws in the data collecting process). This can result in missing, inconsistent or duplicate values, but this can be resolved by removing missing and duplicate values or verifying the data types to prevent inconsistencies (e.g. text where there should be a number). In this case, there are some values in the lists that are sometimes missing, the position of players on the map on the last timestamp of every match, so in this case, we are going to remove these errors since it would not affect the final result (this error may be due to a game not always finishing in a 60-second interval, so the game has trouble getting a position of a player that has left the game already).

- **Feature Aggregation**: this is the step that helps put the data in a better perspective.

This not only reduces memory combustion and processing time as well as provide a more stable behaviour in the group than if we were looking at individual data. In this context, we are going to aggregate events in a timestamp that refers to the same minute. Since LoL matches last on average 34 minutes, having a line data for every millisecond that an event happens makes it impossible to track any patterns. Aggregating it to very minute (since timestamps in events do already have an interval of a minute), would make for a more realistic view of the game, since the exact millisecond when a player buys an item is not that relevant, compared to what phase of the game or minute. Also, some timestamps are not exactly on the minute, even though all start at 0 milliseconds, the second timestamp may be at 6000, 6001 or 6230. Another aggregation that will be made is aggregating all accounts from the same player, since is a bit redundant to track 4 accounts separately if they refer to the same person. This, again, gives a more high-level view of the dataset, since for each player there can be hundreds of matches to evaluate.

- **Dimensionality Reduction**: This refers to reducing the number of features (dimensions) on a dataset Data analysis tasks become significantly harder as the dimensionality of the data increases, since the number of planes occupied by the data increases, adding more sparsity to the data which becomes difficult to model and visualize. In this context, if we use all the features that the requests receive (we do not even make every request possible to the API for this very reason), we would have X features. Some of these are not even relevant to our research, some are redundant, others do not always have a value per timestamp or line of data, so now we are going to reduce that number to Y features. Also, depending on the extraction, we will make a feature subset selection.

We can say that the points mentioned above can involve both parts (independent of the analysis and dependent of the data), since we have to understand the data to choose what data types are correct per feature, but also it is obvious that duplicate data points should be deleted, using data quality assessment as an example.

For the second perspective, let us keep in consideration what ”questions” we want to ask our dataset. In VisuaLeague3.0 [10], Afonso conducted two interviews with two professional LoL coaches, to learn what knowledge extractions were interesting to make, know what already exists (or does not exist) in the market, and how do these coaches proceed in doing their own analysis. From that interview, we can conclude the following:

- Recording when and what happens after certain events (e.g. tower destructions) as well as if the team fulfils certain objectives (e.g. killing the dragon or baron), is very indicative of a player’s behaviour and if a match is going well for their team.,

- Tracking Jungler’s movements were referenced multiple times as very important,

- A Heatmap of Ward placement by time was also suggested more than once (especially for the analysis of the Support role),
- Tracking of all the player's locations through a match, with the ability to choose a timestamp or time range to analyse, and
- It was confirmed that to review a single player's behaviour and development, it would be necessary around 100 to 200 matches.

Knowing that we have hundreds of matches per player, where each could be with different roles, we decided to have the extraction focus on the roles, i.e., have a separate analysis per role taken in matches, and also a global one for all matches. Since previously mentioned, there are 5 roles that a player could take in League of Legends: Top, Mid(dle), Bot(tom), Jungler and Support, where each one has an expected behaviour associated with (since the champions - characters in the game - that could be chosen for that role have common characteristics).

Why do we consider multiple accounts per player? As mentioned before, some players (professional or not) have more than one account that they used actively. Sometimes is to use different champions/roles or to start from level one as a challenge. It is easy to conclude that using these other accounts and matches will help give a more dense review of the players play-style and behaviour, since some patterns may still be common even with different champions chosen, such as players may have a similar behaviour even with different playmates. There is no intuitive way to know the multiple accounts of a player and it has to be done "manually". One way is to ask the player in question but, if that's not possible we could go to LOLPros.GG [3] and check a team or a player's profile, which usually has a list of alternate accounts. This website also has information like what role does the player take on the team, previous teams and even a tab for a live match.

Why do we want to evaluate an entire team performance? Why is it hard to get an entire team list of matches manually? In data analysis, context is everything. It was previously mentioned how spatiotemporal awareness of the data adds context that existing platforms for LoL matches lack. Right now the only way to have it into consideration is if coaches watch a live streaming or recording, which is very time-consuming and may still lose some information due to human mistakes. As previously mentioned in 2.2.3, a lot of team training is done in Section *scrims*, and unfortunately those matches are not trackable in the Riot API. Also, let us consider VisuaLeague [10][28] previous iterations; multi-game analysis of one player is way more complete and balanced compared to looking at one game at a time. Making this for each member of a team is, again, time-consuming, and also, we can assume that in random games with random players, the team members may change their behaviour.

Considering this information when designing the user scenarios and dashboards (detailed in Section 4.3.2) we have constructed three files for each player or team:

- **Match**: static data of each match such as which team won or lost, what player made part of which team (blue or red), what were the stats of the players, finals statistics of each team and the attribution of a participant Id (number from 1 to 10) to each player.

- **Events**: timeline regarding game events. Each timestamp (1 minute between each), there's is a list of events that happen during that time, like champion kills, buying and using items, ward placement and destruction, skill level up just to name a few. To each applicable event, there is associated a map location, exact timestamp, list of participants, victim and killer Ids, ward type and some more.

- **Pos**: similar to events, but with just a list of the stats of the player at the moment of the timestamp. Some features include a position on the map, level, experience points, total gold and team score.

4.2 Knowledge Extraction

Knowledge extraction is the process where is created knowledge from structured an unstructured sources, and the result is in a machine-readable and machine-interpretable format that facilitates inference. So what we are doing during this phase is taking our structured data (our CSV file that resulted from the preprocessing of the API requests), and use Machine Learning algorithms to infer knowledge and reach some conclusions.

Knowing that we have hundreds of matches per player, where each could be with different roles, we decided to have the extraction focus on the roles, i.e., have a separate analysis per role taken in matches, and also a global one for all matches. Since previously mentioned, there are 5 roles that a player could take in League of Legends: Top, Middle, Bottom, Jungler and Support, where each one has an expected behaviour associated with (since the champions - characters in the game - that could be chosen for that role have common characteristics).

In [14] it is propose a method to identify the role of a player in a match using a style of mapping (described bellow), using Riot API's role and lane data, in order to try and correct a player's style to the five roles as accurately as possible (this example being about 87.5% accurate). The definitions of "role", "lane", and "position" are all similar, but for the sake of this explanation we use the Riot API's definition of them. The values Riot API provides for Role, Lane, and Position are below:

- Role: DUO, DUO_CARRY, DUO_SUPPORT, NONE, and SOLO;

- Lane: TOP_LANE, MID_LANE, BOT_LANE, and JUNGLE;

- Position: TOP, MIDDLE, JUNGLE, BOTTOM, SUPPORT, APEX, and NONE.

For a given matchId, the API gives gives one value for Role and Lane, per player. To try to get an accurate description of the position of a player, we need to combine these two values. The proposed mapping solution is as follows:
{
(MID_LANE, SOLO): MIDDLE,
(TOP_LANE, SOLO): TOP,
(JUNGLE, NONE): JUNGLE,
(BOT_LANE, DUO_CARRY): BOTTOM,
(BOT_LANE, DUO_SUPPORT): SUPPORT
}

This mapping solution takes the lane and role values, and gives a position value. Of course this is an estimate, since even the role and lane values are estimates as well (a player does not pick these values before starting a match, they are calculated by the API).

Using player's coordinates during a game and analysing using that data would be more accurate, but there is not a need for that right now, since is too complicated and the developed dashboards do not need to be that precise. This mapping will be used throughout this project when referring to a player's role in game. To note that while building the final dashboards for evaluation, this mapping solution did not work as well, as most positions were mapped as NONE, so we decided to also use the data from Role and Lane.

4.3 Visualization

4.3.1 Dashboard Design

The overall idea of the developed dashboards is to be divided into 3 areas: at the very top there are the filters/highlights (Figure 4.3 area [A]), where the user can change the look of the dashboard according to their preferences. In the middle is the position tracking area ((Figure 4.3 area [B]), where there are at all times three maps with views related to positions of the players throughout a match. At the bottom is the event area (Figure 4.3 area [C] and Figure 4.4), where there are multiple views that relate to events during matches, such as levels gained or placement of wards. This last area has different visualisations from role to role.

To keep in consideration that the dashboards tried to follow design best practices as much as possible, with instructions written clearly in titles and filters, using the tooltip feature (a small window that appears while hovering/selecting a datapoint in a view) as a way to add more context to a data point, and matching colour/font that both makes the data cohesive and also draws attention to important areas of the dashboard/view. Next, we are going to describe the dashboards in more detail.

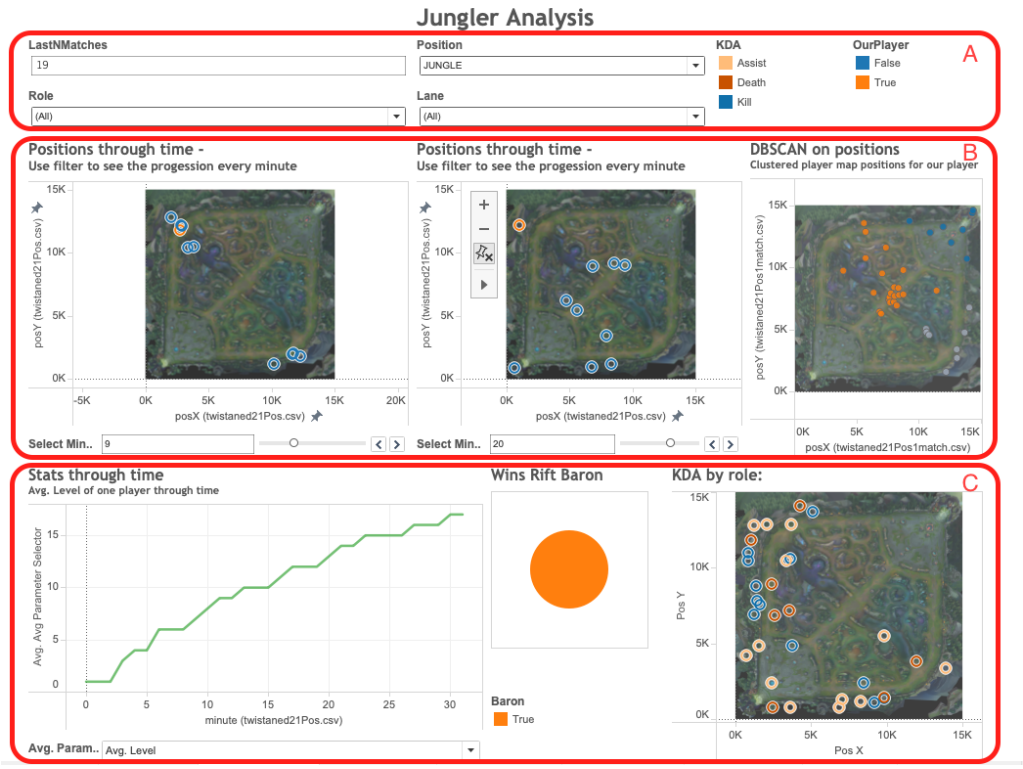

Figure 4.3: Example of Filter area [A], Position Tracking area [B], and Event Area [C]

Following Figure 4.3, starting at the top there is the title of the dashboard, in the form of *[Role] Analysis*. Bellow is the Filter/Highlighter area [A], which contains the following, from left to right:

- *LastNMatches* filter: select last N matches by date (users types a number between 1 to 19) to analyse. This restriction in number of matches is due to limitations of the Riot API, which only saves spatiotemporal data of the last 19 matches. Affects all views except *DBSCAN on positions*;
- *Role/Lane/Position* selection filters: select one or multiple roles the player takes in the present dataset (only shows relevant values); Affects all views except *DBSCAN on positions*.
- *KDA* highlighter: color code ”kill”, ”death” and ”assist” event of our player. Only affects *KDA by role* view.
- *OurPlayer* highlighter: color codes player datapoints vs. other players. Only affects *Positions through time* view;

Below the filter area [A] is the Position Tracking [B] area. There are two *Position through time* views, with each accompanied by a time slider at the bottom of the maps, in order to compare all players' positions in two different instants. There is also on the right-side a map chart, *DBSCAN on positions*, where the rows are the Y coordinate and

the columns are the X coordinate of a player's position, and we see the resulted DBSCAN clusters and their designated colour. The Position tracking area is the same throughout all dashboards.

At the bottom of the dashboard is the Event area (consider both area [C] of Figure 4.3 and all of Figure 4.4). Here the presented views may vary according to the analyst preference. We have (from left to right, top to bottom):

- *Stats through Time*, a line chart that represents the average of a selected stat through time. We can choose trough the Avg. Parameter filter what stat we want to see, and the chart aggregates all games played on the selected days (date range filter) and then calculates the average of the stat (for example, level or gold collected).

- *KDA by role* is another map chart, this time showing, with different colours, where a kill, death, or assist has happened, according to a player's selected role and range of dates.

- *Ward placement by role* uses a map of the game to create a Heatmap of the positions of wards placed during matches that occurred on the select date range. On the right we see a highlighter feature on the roles (showing only the available ones on the dataset), with each having its own colour. Selecting one turns the other ones less opaque (almost white in colour) in order to better make out their position.

- *Event timeline*, a circle chart where the columns represent the exact timestamp (milliseconds in game) where a certain event occurs and on the rows we see the event type. Every time a certain event occurs, a circle appears on the exact timestamp.

- *Wins by [X]*, a pie chart that show the percentage of wins where the player's team fulfilled an objective, such as the first to kill the dragon or baron.

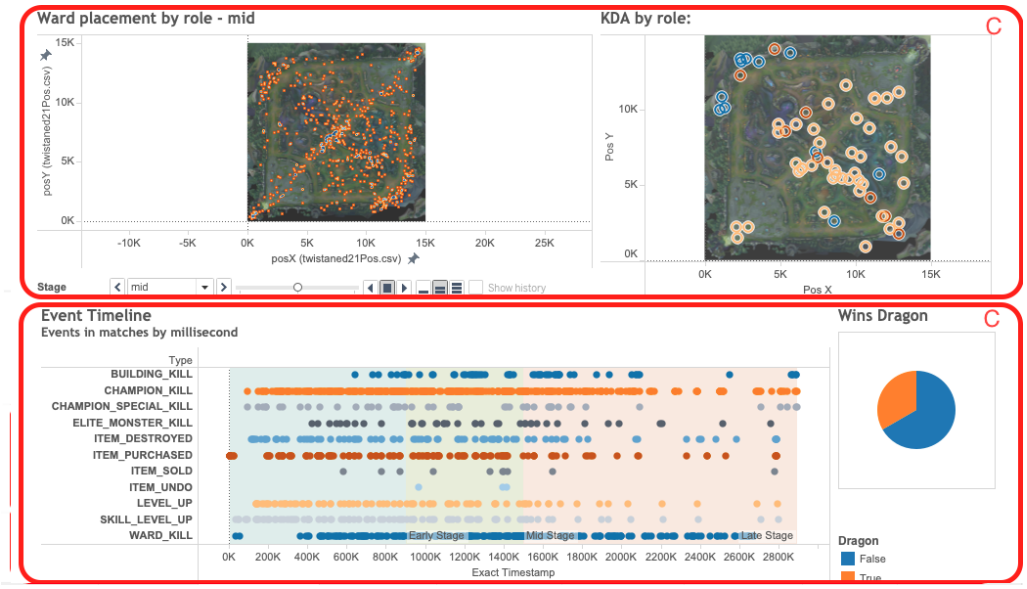

Figure 4.4: Examples of other views present in Event area [C]

4.3.2 Developed Dashboards and Stories

This section will focus on the design and development of the dashboards and stories. Each dashboard has a description of of its goals and a small list of possible knowledge extractions, what contents does it have, being filters, highlighters or views, followed by a screenshot of said dashboard and an example of a usage scenario, describing how a user can interact with it. For the latter, we assume that Tableau already has the player's information loaded into the views.

Top Analysis

Following the Figure 4.5 we can see in area [A] the filters *LastNMatches*, *Position*, *Role* and *Lane*, and the highlighters *OurPlayer* and *KDA*. Next is the Position Tracking [B] Area, with two *Position through time* views side by side, and on the right *DBSCAN on positions*. And at the bottom, in Area [C], the view *Event Timeline* on the left, and *KDA by role* on the right.

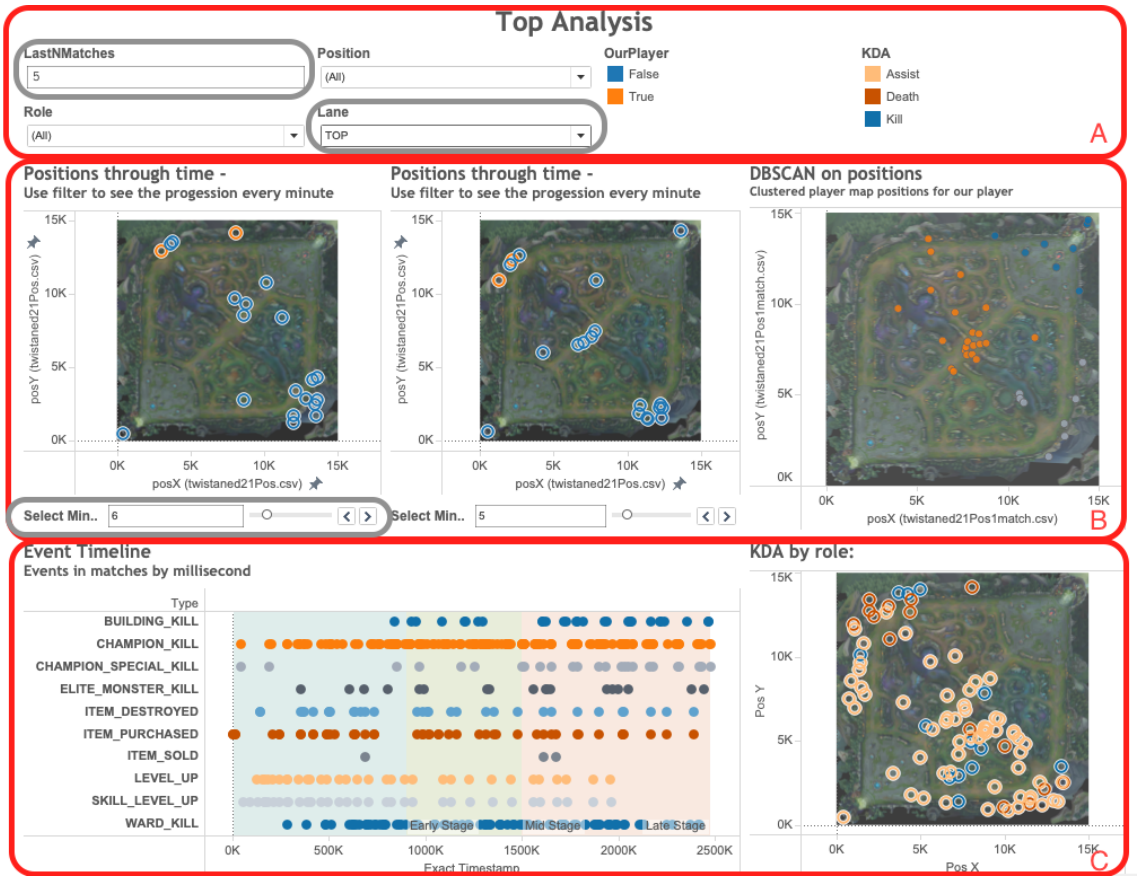

Figure 4.5: *Top Analysis* Dashboard

The main goal of this dashboard is to provide an analysis of multiple matches (up to 19) of a summoner that plays in the Top role. Some of the possible knowledge extractions could be identifying what are the major areas of the map where the player is located, where the player executes kills or when it starts destroying towers, to name a few.

Scenario: User wants to know when, on the last 5 matches, the player starts advancing to the opponents' side. Follow Figure 4.5 for details.

1. User types on *LastNMatches* filter "5" (Area [A]);

2. User selects on *Lane* filter "TOP" (Area [A]);

3. On *Positions through time* view (Area [B]), the user slides the timeline from left to right until they find the players position moving to opponents' side;

4. User checks in which timestamp that map lands, and that is the answer (in this case, at the 6 minute mark);

5. (Optional) user can click on "True" in *OurPlayer* highlighter (Area [A]) to only show the player's positions (instead of all players').

Jungler Analysis

Following the Figure 4.6 we can see in area [A] the filters *LastNMatches*, *Position*, *Role* and *Lane*, and the highlighters *OurPlayer* and *KDA*. Next is the Position Tracking [B] Area, with two *Position through time* views side by side, and on the right *DBSCAN on positions*. And at the bottom, in Area [C], the view *Stats through time* on the left, *Wins Rift Baron* at the middle, and *KDA by role* on the right. At the bottom left is the filter Avg.Parameter, which selects one of the list of parameters available (avg. Level, avg. XP, avg. Gold)to analyse. Only affects Stats through time view;

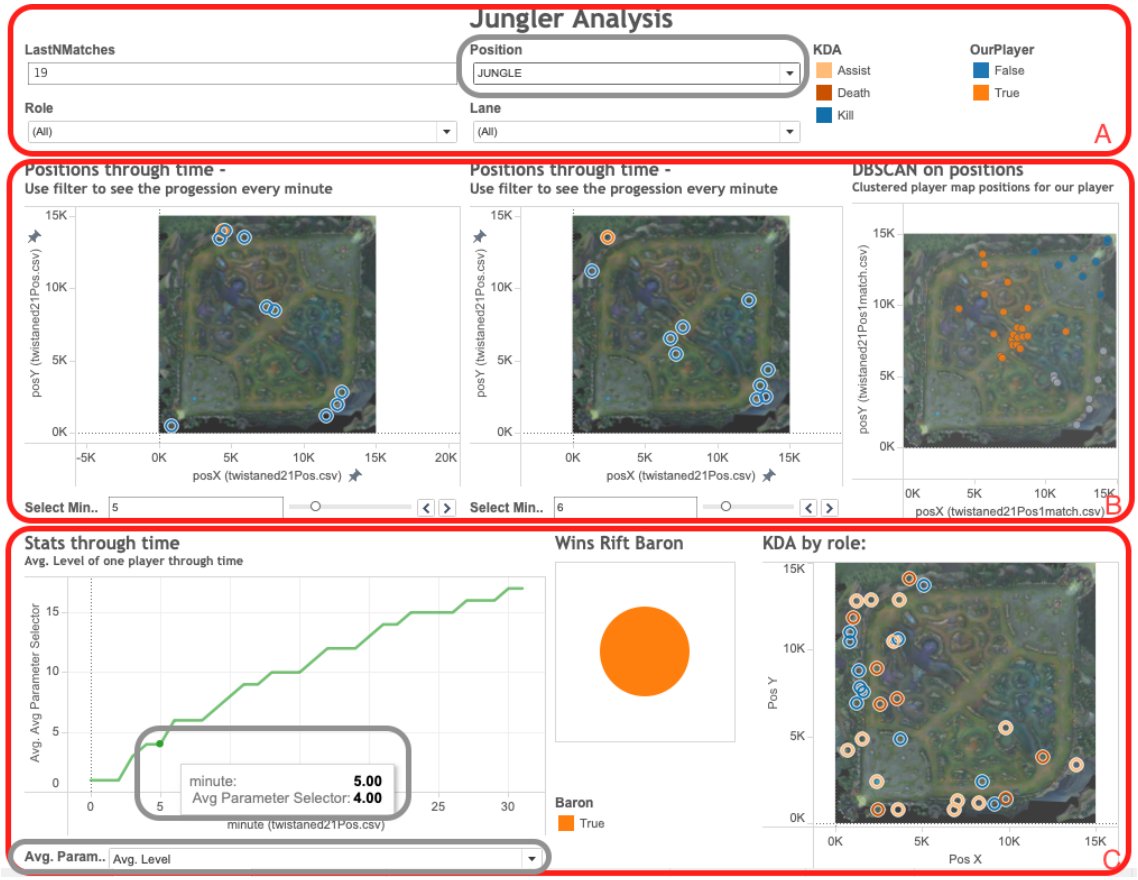

Figure 4.6: *Jungler Analysis* Dashboard

The main goal of this dashboard is to provide an analysis of multiple matches (up to 19) of a summoner that plays in the Jungler role. Some of the possible knowledge extractions could be where was the player located when Baron spawned (20 minute mark), where the player dies or what is the average amount of goal the players has throughout time.

Scenario: User wants to know, on average, what level does the player reach, at the 5-minute mark. Follow Figure 4.6 for details.

1. User selects on *Position* filter ”JUNGLE” (Area [A]);

2. On *Stats through time* view (Area [B]), the user select in the *Avg.Parameter* filter "Avg. Level";

3. Following the line chart, user checks in which timestamp that event lands (x-axis), that is the answer (in this case, level 4);

Middle Analysis

The main goal of this dashboard is to provide an analysis of multiple matches (up to 19) of a summoner that plays in the Middle role. Some of the possible knowledge extractions could be when does the player level up, where the player executes assists or what other roles does it play has.

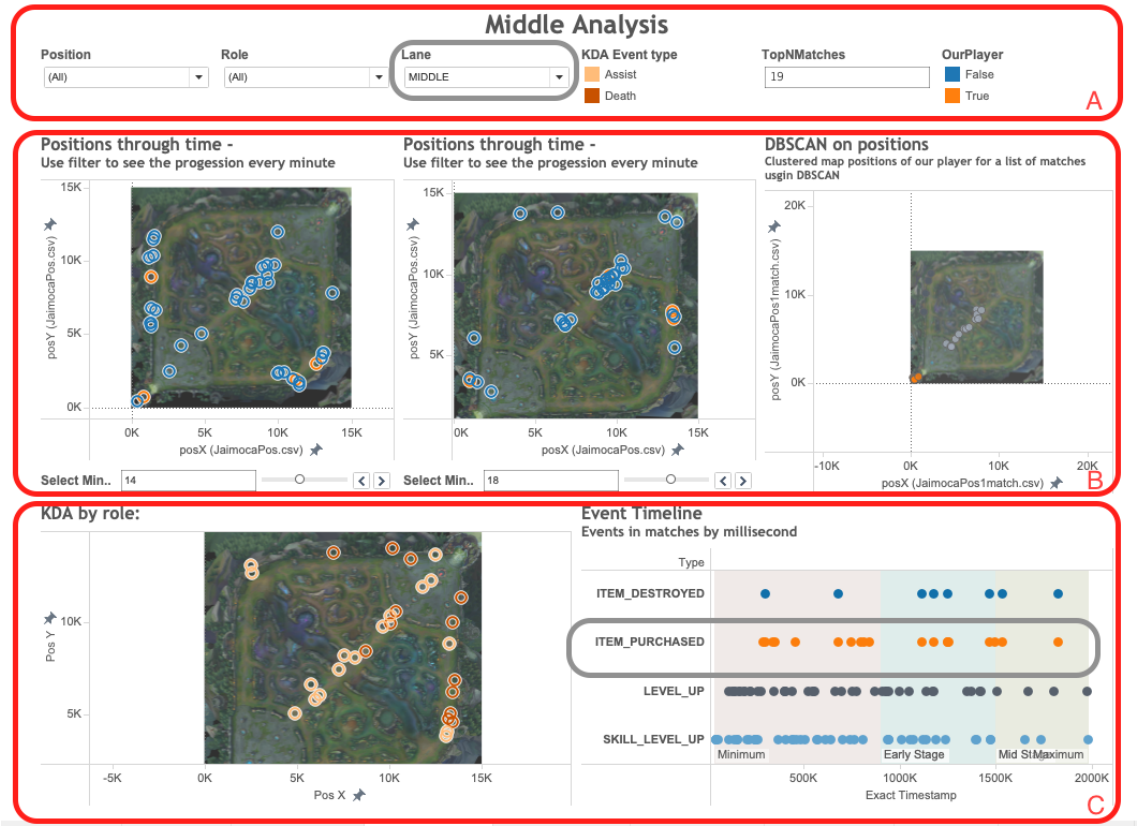

Figure 4.7: *Middle Analysis* Dashboard

Following the Figure 4.7 we can see in area [A] the filters *LastNMatches*, *Position*, *Role* and *Lane*, and the highlighters *OurPlayer* and *KDA*. Next is the Position Tracking [B] Area, with two *Position through time* views side by side, and on the right *DBSCAN on positions*. And at the bottom, in Area [C], the view *Event Timeline* on the left, and *KDA by role* on the right.

Scenario: User wants to know when, on average, does the player go to their base. Follow Figure 4.7 for details.

1. User selects on *Lane* filter ”MIDDLE” (Area [A]);
2. On *Event Timeline* view (Area [B]), the user checks on the y-axis the event ”ITEM_PURCHASED”;
3. (Optional) user can select that line in order to highlight that event;
4. Following the line, whenever there is a dot it means that event occurs, the user checks the x-axis for timestamp, and that is the answer;

Note: To purchase an item, a player needs to be at their base.

Bottom Analysis

The main goal of this dashboard is to provide an analysis of multiple matches (up to 19) of a summoner that plays in the Bottom role. Some of the possible knowledge extractions could be knowing what it the ratio of wins to loses when their team kills the Dragon, when does it start killing other players or when do they go to their base, to name a few.

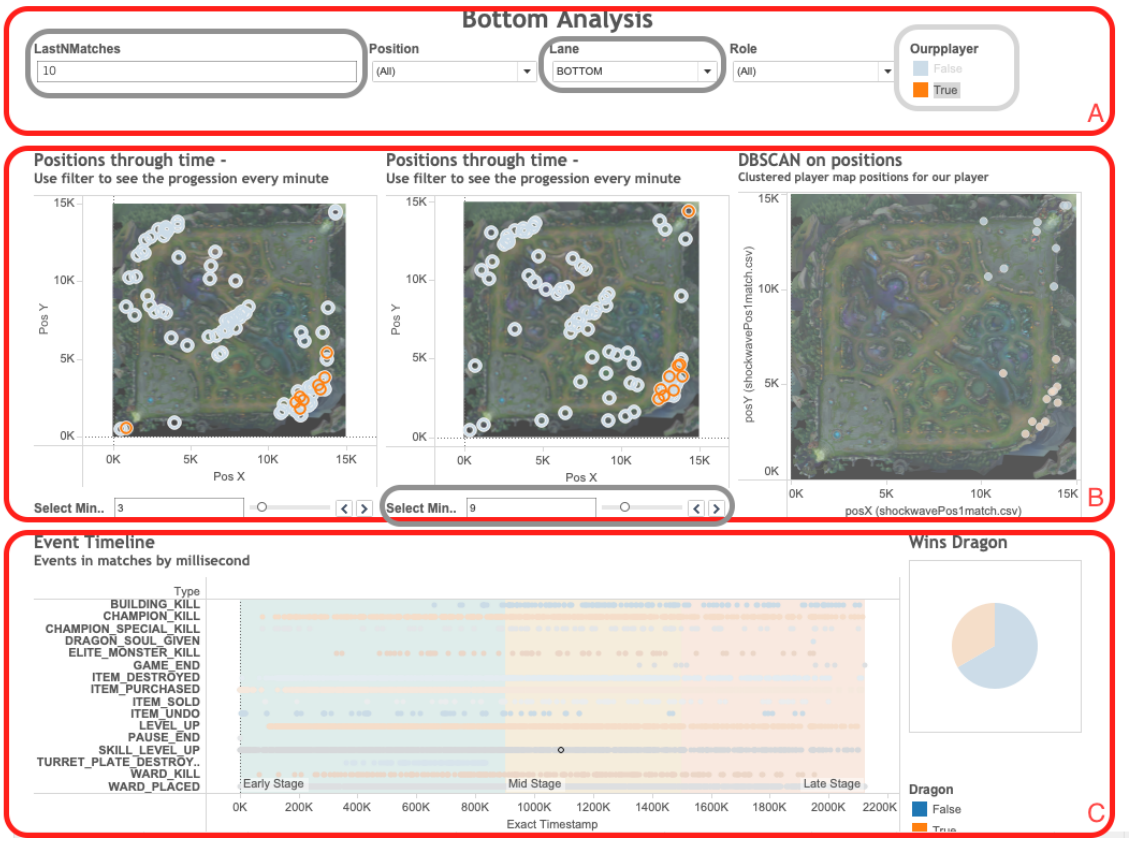

Figure 4.8: *Bottom Analysis* Dashboard

Following the Figure 4.8 we can see in area [A] the filters *LastNMatches*, *Position*, *Role* and *Lane*, and the highlighter *OurPlayer*. Next is the Position Tracking [B] Area, with two *Position through time* views side by side, and on the right *DBSCAN on positions*.

And at the bottom, in Area [C], the view *Event Timeline* on the left, and *Wins Dragon* on the right.

Scenario: User wants to know where is the player at the 9-minute mark, during the last 10 matches. Follow Figure 4.8 for details.

1. User types on *LastNMatches* filter ”10” (Area [A]);

2. User selects on *Lane* filter ”BOTTOM” (Area [A]);

3. On *Positions through time* view (Area [B]), the user types on the timeline ”9”, or uses the slider until the 9 minute mark;

4. User checks in the map where the player’s position are, and that is the answer (in this case, the bottom-right corner - bottom lane);

5. (Optional) user can click on ”True” in *OurPlayer* highlighter (Area [A]) to only show the player’s positions (we did this on Figure 4.8).

Support Analysis

The main goal of this dashboard is to provide an analysis of multiple matches (up to 19) of a summoner that plays in the Support role. Some of the possible knowledge extractions could be knowing at what game stage does it put wards in the middle lane, where the player do more assists or where is the player usually located in the late stage, to name a few.

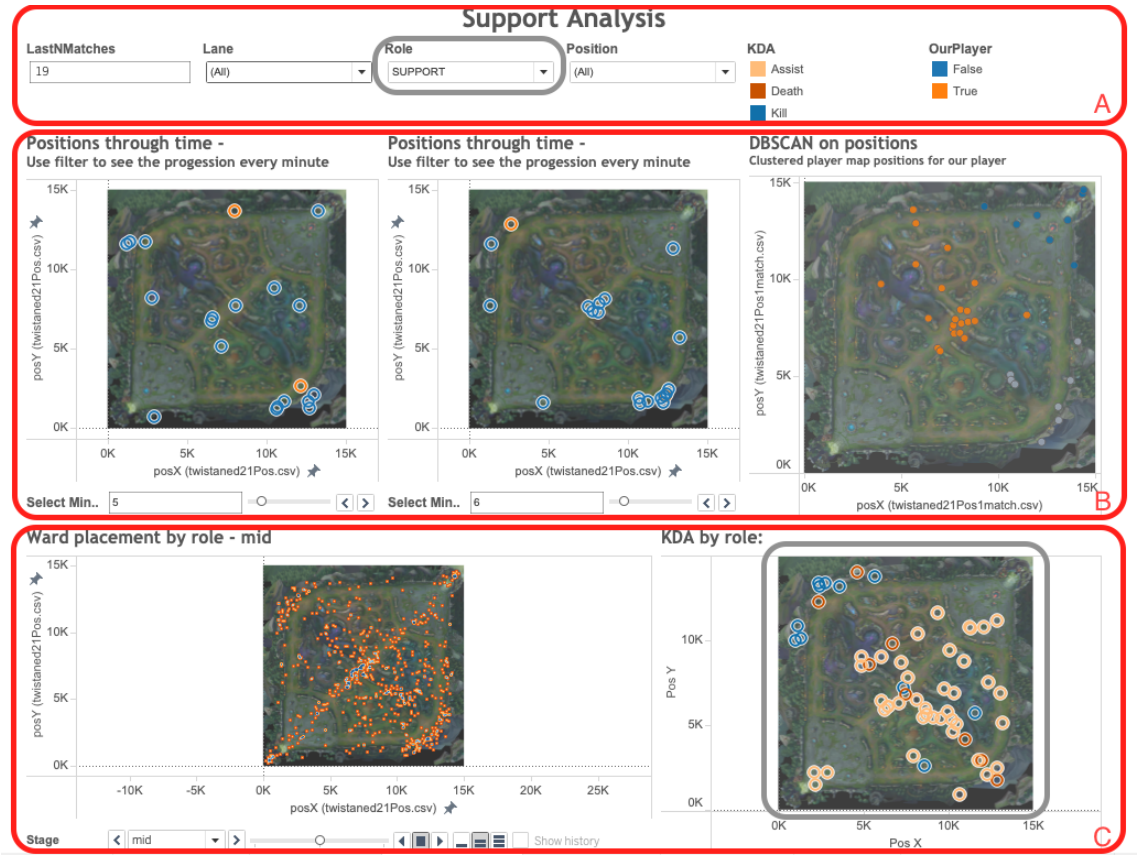

Figure 4.9: *Support Analysis* Dashboard

Following the Figure 4.9 we can see in area [A] the filters *LastNMatches*, *Position*, *Role* and *Lane*, and the highlighters *OurPlayer* and *KDA*. Next is the Position Tracking [B] Area, with two *Position through time* views side by side, and on the right *DBSCAN on positions*. And at the bottom, in Area [C], the view *Ward placement by role - mid* on the left, and *KDA by role* on the right.

Scenario: User wants to know in in which area of the map does the player do more assists. Follow Figure 4.9 for details.

1. User selects on *Role* filter ”SUPPORT” (Area [A]);
2. On *KDA by role* view (Area [C]), the user checks in the map where the assists’ datapoints are located, and that is the answer (in this case, between the mid and bot lane);
3. (Optional) user can click on ”Assist” in the *KDA* highlighter (Area [A]) to only show the player’s assists.

Player Behaviour Story

After all of the dashboards were designed, we approach the ”Story” in Tableau. This is basically a slideshow of dashboards, were we can add written text an comments, and still

interact with the dashboards displayed. In Figure 4.10 is the ”Player Behaviour” Story, which just lists all the previous dashboards. This helps players to compare all their roles in one place.

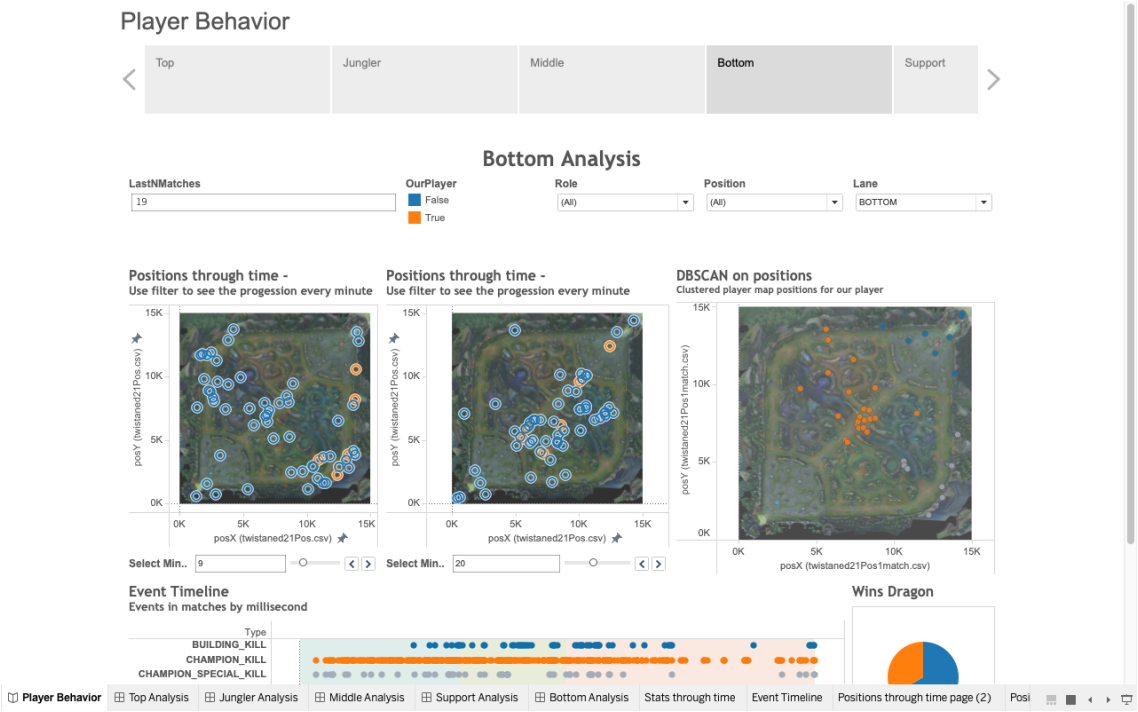

Figure 4.10: *Player Behaviour* Story

Chapter 5

Evaluation

This chapter presents the evaluation of the tool developed for analysing games in the League of Legends' competitive matches environment. Starting by describing the methodology of the user study (Section 5.1), followed by the description of the participants (Section 5.2), and ending with the respective results in Section 5.3. Finally, the discussion of the results is presented in Section 5.4.

5.1 Methodology

To assess the dashboards created, interviews were conducted with five members of an amateur LoL team.

Due to the present Covid-19 pandemic, we decided to do these interviews remotely, through Zoom meetings. These Zoom meetings were set up so that the players had remote access to my shared screen, where their dashboards were displayed so that they could explore the tool at will. For every interviewee, a Google Docs file was shared that contained a description of the evaluation process, the questions to answer and a group of tables to fill in with their grading. There were 5 different documents, one per player, and each player only had access to their own file.

To start, all players answered some demographic questions, regarding their age, how long have they been playing LoL and its frequency, what role to they usually play as, what play style they prefer and if they have previous experience in using match analysing tools. All of this information will be described in the following section. A short introduction was given to Tableau and the dashboards used, and each player only had a view of their own games (from their **Summoner name** which was provided earlier). After that, each interview was slightly different, and each participant answered three questions about their behaviour according to the their role in the team. The goal of these questions was to guide the participants through the different views, allowing them explore Tableau and views/filters and understand if they can extract meaningful information from the developed dashboards. The questions involve the analysis of multiple matches and were all

different (see Appendix B).

The following list compiles all the different sets for questions asked during the evaluation process, separated by role. To re-iterate that each player only answered one set of questions, according to the role they played in the team (this information is provided in the demographic questions at the beginning of each interview).

- Top:
    1. When does the player start advancing to the opponent's side?
    2. Which roles does the player usually take on?
    3. On the "Top" role, on the last 5 matches, list 2 main areas where the player was positioned on the map.
- Jungler:
    1. On average, what level does the player reach, at the 5-minute mark?
    2. Where do they die more often?
    3. In the last 8 matches, did their team kill Baron first?
- Middle:
    1. Where do they kill the most?
    2. When do they go to the base/buy items?
    3. Does the player move to other lanes?
- Bottom:
    1. Where are they at the 9-minute mark?
    2. In which stage do they start destroying towers?
    3. When, on average, do they start buying items?
- Support:
    1. In which area of the map do they do more assists?
    2. Where are wards placed?
    3. Which roles does the player usually take on?

There was no time limit and we would advance to the next phase whenever they were ready. On average, the whole interview lasted less than 30 minutes.

Each player, after answering the initial questions, filled out some tables regarding their experience with the platform, evaluating its **Intuitiveness**, **Usefulness**, **Efficiency** and

Innovation, ending with a general grading and were also able to write an open comment with regards to their experience and/or adding some suggestions. All tables were graded from 1-5, with 1 being the worst grade and five being the best grade. These choices of parameters and evaluation method was inspired by VisuaLeague's [10], where the same parameters where evaluated in order to conclude if the tool is easy to use and reveals relevant information.

5.2 Participants

All five players from this team have participated together in UPorto inter-faculty tournaments, so they are a good sample of our focus demographics, since they all play LoL frequently, have experience playing in the solo queue and in competitions as well as all members already used match analysis tools before the study. Their ages are between 21 and 24, all have at least 7 years of experience playing LoL and play on a weekly basis. In regards with the type of games they play, it varies: some of them have more experience in tournaments and professional teams than others but all play in ranked matches on their own. All have previous experience in tracking their in-game behaviour using other tools, with some of the most mentioned being *Porofessor* [4] and *Op.gg* [2].

5.3 Results

In this section we are going to list the results of the evaluation explained previously, listing the players' results and opinions and ending with some conclusions. The team interview is listed in its entirety in Appendix B.

After exploring the tool, the participants had to evaluate their experience in 4 parameters (intuitiveness, usefulness, efficiency and innovation) plus a general grading of the tool (evaluation). To note that if the players were able to reach an answer to the proposed questions, the answers were all correct.

The four parameters (as mentioned before) are intuitiveness, usefulness, efficiency and innovation. To describe what they mean:

- Intuitiveness corresponds to how easy it was to answer each proposed questions,
- Usefulness, to how useful was the information provided from each question,
- Efficiency is how fast were the users able to get each of the answers, and
- Innovation asks if the proposed analysis method offers anything new compared to already existing tools.

The Table 5.1 presents the ratings per parameter of the evaluation, per player. For intuitiveness, usefulness and efficiency, the values presented on the following table are

the averages of the grading each player gave. For e.g., the participant that analysed the Top Dashboard graded all three questions as 5 on intuitiveness, so the average is 5. For innovation and evaluation, since these parameters were directed to the whole dashboard and not the proposed questions, only have one grade, and that values is the one presented.

Role/Parameter	Intuitiveness	Usefulness	Efficiency	Innovation	Evaluation
Top	5	4.6	5	4	4
Jungler	5	4.6	5	5	4
Middle	4	5	5	4	4
Bottom	4.6	4.6	5	4	4
Support	4	1.6	3.3	3	2

Table 5.1: Evaluation Results: average rating (from 1 to 5) per parameter per role

5.4 Discussion

This section will describe all the players' comments and a discussion of the results of this evaluation.

The final comments could be separated into three main categories: positives, additions, and negatives. Starting with the positives, they said that:

- A spatio-temporal approach is innovative and very useful. Their usual platforms/-tools do not show this type of data;
- Choosing the number of matches to analyse is beneficial, since existing tools have a static number of matches.

Following with the suggested additions:

- Wished to see team comparison (between all players of a single team);
- Show when is their first death, compare Jungler's positioning with Jungler proximity (metric that says how much time a Jungler spend in Top/Mid/Bot lane per match);
- Wished to see this type of approach applied to other datapoints, such as champion information or player build (not really the focus of this project).

And then the negatives they said:

- Missing ability to drag maps using mouse; DBSCAN not being interactive (Tableau restrictions);
- Needed better data accuracy, specifically positions and ward placement (API restriction);

- Better tooltip descriptions: show accumulative xp, change in gold amount, win with first herald/dragon (my own design fault, some were shown in other dashboards, other the player may have not found on their own).

To summarize this small study, the feedback was overall very positive and it helped us conclude that a spatiotemporal approach focused on interactivity is the way to go in terms of building behaviour-analysing tools for players. The tool yielded significant results for the goal of this project, as it both proved that it is possible to make spatiotemporal analysis from already available information from LoL matches, and also that players are interested in these type of data extractions, since the participants responded positively to the dashboards, mentioning how unique and pertinent the analysis were.

There is still a long way to go but there is a market and an audience for spatiotemporal analysis.

Chapter 6

Conclusion and Future Work

This chapter finishes the report by describing its conclusions from the studied related work and developed tool, and then the possible future work, which includes useful improvements and what could become of this perspective of videogame analysis.

6.1 Conclusion

While studying the related work we could see that clustering algorithms and other machine learning techniques are suited for the analysis of big amounts of data. Specifically unsupervised learning is good when there is no training data available for testing the created models and labels. For spatiotemporal data analysis, we have chosen the clustering algorithm DBSCAN due to its ability to clear out noise points (outliers), making the final cluster map free of clutter without losing pertinent information.Since the resulting clusters can easily be translated into maps (animated or not), it is a good fit for a analytic visualisation approach, as it it easier to use the output in a view or chart.

Concerning videogame visual analytics, there is potential in terms of developing interesting visualisations using spatiotemporal data. Multiple studies were made where it was concluded that a spatiotemporal approach could prove useful for both players and game designers in terms of exploring both the players' behavior during gameplay and game performance. Also the use of maps (animated or not) and passive retrieval of spatiotemporal information from strategic points in game maps (with the use of cameras) is proven to be a much faster and accurate way of getting information than traditional ways, such as live note-taking and screen recordings.

Specifically regarding LoL, VisuaLeague [10] has concluded that spatiotemporal data gives important information regarding a player's movement during a MOBA game. Its integration with other existing analysis tools could make the job of coaches and professional players of viewing multiple matches easier, efficient and accurate, and that a whole-team approach could prove useful combined with single-player analysis.

The tool created in this project used Tableau Desktop, a platform that takes datasets

and creates visualisations and dashboards, turning simple CSV files into interesting and interactive experiences that allows the analysis and extraction of information from data to be viewed in a more clear and pleasing way. The data used was extracted from Riot API and preprocessed using Python and Datalore. The data used referred to players' timeline (behaviour throughout time) and match information (for e.g. which team won and what time was the match held).

Most of the created views were chosen due to the interview provided in VisuaLeague [10], where coaches gave some leads on what they are looking for when analysing a player's behaviour during matches. To summarize, 5 different dashboards were developed with the goal of helping to extract knowledge regarding a player's behaviour during multiple matches, and the different dashboards are separated by the role which the player enrolls. The dashboards track such metrics as movement, stats evolution or kills/deaths/assists through time, and can be interacted with in order to change the views and have a personalized analysis defined by the user's goals. Some of the suggested views in those interviews weren't able to be fulfilled 100%, since inaccuracies in Riot API and data limitations could not allow it, for example in the Ward Placement Map, where the datapoints are were the player is located in that minute and not when they placed the ward, so the map looks a little awkward since some placement is done in the middle of a lane, which does not make a lot of sense if zoomed in.

Tableau also allows to some extent the use of Python scripts, with the use of TabPy, in order to create more measures to be used in their views, which we took advantage of to create a dynamic DBSCAN algorithm, ie. takes the existing datapoints for a player's positions and adapts its parameters values (ϵ and MinPts) to best fit the input data. This turned out to not be a perfect solution, since there is some restrictions with TabPy, so the "DBSCAN on positions" view became less interactive then expected. So even though the parameters were adjusted as needed, the user could not change how many matches were used in the algorithms and any sort of aggregation would lead to errors in the scripts. This is because there was a limit on the size of the data that could be sent to TabPy, and that limited the types of views that could be made with this approach.

After this, the tool was evaluated by a team of LoL players. These players all had previous experience in both using tracking tools and tournament matches. All player's interviews were held via Zoom, due to the pandemic situation.

From this study, we can conclude that regardless if a player has experience in using game analysis tools or not, they can usually understand and explore the developed dashboards, saying that the use of maps and the interactivity that they have from the use of filters allows a more in depth analysis of their behavior that other existing web-tools. Generally, the players that evaluated the tool had a positive response, commenting that the dashboards show pertinent information to their own goals, adding that it was fast and also intuitive to use.

6.2 Future Work

For this tool to grow, these would be the next possible steps:

- Approach a whole-team analysis, where the dashboards and views would allow the user compare all players of the same team;
- Incorporate the additions and corrections suggested by the players that performed the user study, for example the Jungler proximity metric or better tooltip descriptions on the views;
- Approach *scrims*, which currently is impossible due to API restrictions, which would be specifically useful for coaches;
- Incorporate these dashboards into another platform that allows direct data extraction and injects the data automatically into the dashboards, without the need to create files with needed datasets previously;
- Change to a platform better suited for python scripting integration or clustering algorithms in general, in order to better/add more interesting views;
- The chosen platform should be easy to install and/or use by everyday people, since Tableau requires a paid license to use or even open a Tableau flow;
- More testing involving players, professional teams and coaches, in order to curate better visualisations and understand other possible approaches to the problem.

Bibliography

[1] gisconletras: Testing big spatial data software (hadoop + hbase + geowave + geoserver) without dying in the attempt :-). http://gisconletras.blogspot.com/2017/10/testing-big-spatial-data-software.html. visited on 2022-03-14.

[2] LoL Stats, Record Replay, Database, Guide - OP.GG. https://euw.op.gg/. visited on 2020-12-19.

[3] LOLPros.GG. https://lolpros.gg. visited on 2021-08-05.

[4] Porofessor.gg - league of legends live game search and real-time player statistics. https://porofessor.gg/. visited on 2022-02-19.

[5] Project jupyter — about us. https://jupyter.org/about. visited on 2022-03-09.

[6] Riot API Libraries — Riot API Libraries documentation. https://riot-api-libraries.readthedocs.io/en/latest/libraries.html. visited on 2021-08-05.

[7] Riot Developer Portal. https://developer.riotgames.com/. visited on 2020-12-19.

[8] Welcome to riotwatcher's documentation! — riotwatcher 3.2.0 documentation. https://riot-watcher.readthedocs.io/en/latest/. visited on 2022-03-09.

[9] A. P. Afonso, M. B. Carmo, and T. Moucho. Comparison of visualization tools for matches analysis of a moba game. In *2019 23rd International Conference Information Visualisation (IV)*, pages 118–126, 2019.

[10] Rafael Afonso. Visual League III: Visual Analytics of Multiple Games. Master's thesis, DI-FCUL, 2019.

[11] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. 28(2):49–60, June 1999.

[12] Mafalda Filipa Caldeira Camilo. Análise de padrões dos pedidos de intervenção do Regimento de Sapadores Bombeiros. Master's thesis, FCT-UNL, 2020.

[13] Nuno Carreiro. Técnicas de Visualização para Melhorar o Desempenho em Jogos Online. Master's thesis, DI-FCUL, 2016.

[14] Riot API Community. Identifying champion positions — riot api libraries documentation. `https://riot-api-libraries.readthedocs.io/en/latest/roleid.html`. visited on 2021-10-20.

[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. KDD'96. AAAI Press, 1996.

[16] J.A. García, J. Fdez-Valdivia, F.J. Cortijo, and R. Molina. A dynamic approach for clustering data. *Signal Processing*, 44(2):181 – 196, 1995.

[17] P. Gatalsky, N. Andrienko, and Gennady Andrienko. Interactive analysis of event data using space-time cube. pages 145– 152, 08 2004.

[18] GISGeography. Choropleth maps - a guide to data classification - gis geography. `https://gisgeography.com/choropleth-maps-data-classification/`. visited one 2022-03-14.

[19] T. Goncalves, P. Vieira, A. Afonso, M. Carmo, and T. Moucho. Analysing player performance with animated maps. In *2018 22nd International Conference Information Visualisation (IV)*, pages 103–109, Los Alamitos, CA, USA, jul 2018. IEEE Computer Society.

[20] Alena Guzharina. What is Datalore? | The JetBrains Datalore Blog. `https://blog.jetbrains.com/datalore/2020/08/19/what-is-datalore/`, aug 2020. visited on 2021-08-26.

[21] Riot Games Inc. How to play - league of legends. `https://www.leagueoflegends.com/en-us/how-to-play/`. visited on 2021-10-21.

[22] Jain Anil K. Algorithms for clustering data. Prentice Hall, 1988.

[23] Saurav Kaushik. Clustering | Types Of Clustering | Clustering Applications. `https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/`. visited on 2020-12-19.

[24] TABLEAU SOFTWARE LLC. Tableau Desktop. https://www.tableau.com/products/desktop. visited on 2021-08-17.

[25] TABLEAU SOFTWARE LLC. Use Python scripts in your flow. https://help.tableau.com/current/prep/en-us/prep_scripts_TabPy.htm. visited on 2021-08-05.

[26] Daniel MacCormick and Loutfouz Zaman. Echo: Analyzing gameplay sessions by reconstructing them from recorded data. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '20, page 281–293, New York, NY, USA, 2020. Association for Computing Machinery.

[27] Sanatan Mishra. Unsupervised Learning and Data Clustering | by Sanatan Mishra | Towards Data Science. https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a. visited on 2020-12-19.

[28] Tiago Moucho. VisuaLeague II - Animated maps for performance analysis in games. Master's thesis, DI-FCUL, 2018.

[29] Raymond Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 14:1003–1016, 10 2002.

[30] Pranjal Pandey. Data Preprocessing : Concepts. https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825, November 2019. visited on 2021-08-05.

[31] Inc. Riot Games. League of legends. https://www.leagueoflegends.com/. visited on 2022-03-09.

[32] Hilary Russ. Global esports revenues to top $1 billion in 2019: report | Reuters. https://https://reut.rs/2tlBbns. visited on 2020-12-19.

[33] Claude Sammut and Geoffrey I. Webb. Density-Based Clustering | SpringerLink. https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_211. visited on 2020-12-19.

[34] Ruben Schertler, Simone Kriglstein, and Günter Wallner. User guided movement analysis in games using semantic trajectories. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '19, page 613–623, New York, NY, USA, 2019. Association for Computing Machinery.

[35] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3), July 2017.

[36] scikit-learn developers. 2.3. Clustering — scikit-learn 0.23.2 documentation. `https://scikit-learn.org/stable/modules/clustering.html`. visited on 2020-12-19.

[37] JetBrains s.r.o. Pricing and plans | Datalore. `https://www.jetbrains.com/help/datalore/dl-pricing.html`. visited on 2021-08-26.

[38] JetBrains s.r.o. Quick start tutorial | Datalore. `https://www.jetbrains.com/help/datalore/datalore-quickstart.html`. visited on 2021-08-26.

[39] LLC TABLEAU SOFTWARE. What is data visualization? A definition, examples, and resources. `https://www.tableau.com/learn/articles/data-visualization`. visited on 2020-12-19.

[40] Unity Technologies. Unity Real-Time Development Platform | 3D, 2D VR & AR Engine. `https://unity.com/`. visited on 2020-12-19.

[41] BigWorld Technology. World of Tanks — Free Online War Game. `https://worldoftanks.eu/`. visited on 2020-12-19.

[42] Pedro Vieira. Visualization of spatial-temporal Information for Personal Performance Analysis in Games. Master's thesis, DI-FCUL, 2017.

[43] Günter Wallner. Enhancing battle maps through flow graphs. *CoRR*, abs/1906.04435, 2019.

[44] Günter Wallner, Nour Halabi, and Pejman Mirza-Babaei. Aggregated visualization of playtesting data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–12, New York, NY, USA, 2019. Association for Computing Machinery.

Appendix A

Appendix: DBSCAN

DBSCAN's Main function.

Algorithm 2 DBSCAN($SetOfPoints$,Eps,$MinPts$)

```
// SetOfPoints is UNCLASSIFIED
ClusterId := nextId(NOISE);
for i from 1 to SetOfPoints.size do
   Point := SetOfPoints.get(i);
   if Point.CiId = UNCLASSIFIED then
      if ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts) then
         ClusterId := nextId(ClusterId)
      end if
   end if
end for
```

Auxiliar function of DBSCAN to expand each cluster.

Algorithm 3 ExpandCluster($SetOfPoints$,$Point$,$ClusterId$,Eps, $MinPts$): Boolean

```
seeds := SetOfPoints.regionQuery(Point, Eps);
if seeds.size < MinPts then
  // no core point
  SetOfPoints.changeClId(Point, NOISE);
  RETURN False;
else
  //all points in seeds are density-reachable from Point
  SetOfpoints.changeClIds(seeds, C1Id);
  seeds.delete(Point);
  while seeds <> Empty do
    currentP := seeds.first();
    result := setOfPoints.regionQuery(currentP, Eps);
    if result.size >= MinPts then
      for i from 1 to result.size do
        resultP := result.get(i);
        if resultP.ClId in UNCLASSIFIED, NOISE then
          if resultP.ClId = UNCLASSIFIED then
            seeds.append(resultP);
          end if
          SetOfPoints.changeCiId(resultP, ClId);
        end if
        //UNCLASSIFIED or NOISE
      end for
    end if
    // result.size >= MinPts
    seeds.delete(currentP);
  end while
  //seeds <> Empty
  RETURN True;
end if
```

Appendix B

Appendix: Team's Interview

B.1 Top

What is your age? 22
How long have you been playing LoL? 11 years
Gaming frequency (per week) 5 days
Do you use tools/websites/apps to analyze your previous games? If so, which ones? Porofessor
What game mode do you usually play? Solo queue, with a designated team, professional tournaments or both? Both
What are your preferred roles to play? Everything except jungle

B.1.1 Tests

A When does the player start advancing to the opponent's side?

B Which roles does the player usually take on?

C On the "Top" role, on the last 5 matches, list 2 main areas where the player was positioned on the map.

How easy was it to answer the proposed questions using the dashboards?

Question	Couldn't Answer	1 (Very Hard)	2	3	4	5 (Very Easy)
A						**X**
B						**X**
C						**X**

Table B.1: Intuitiveness Top

How useful was the information extracted from this analysis process?

Question	Couldn't Answer	1 (Useless)	2	3	4	5 (Very Useful)
A						**X**
B						**X**
C					**X**	

Table B.2: Usefulness Top

How fast could you get to what you need for the analysis?

Question	Couldn't Answer	1 (Took too long)	2	3	4	5 (It was fast)
A						**X**
B						**X**
C						**X**

Table B.3: Efficiency Top

Compared to other tools, did this analysis method offer anything new?

1 (Very Similar)	2	3	4	5 (Very innovative)
			X	

Table B.4: Innovation Top

How would you evaluate the platform?

1 (Very bad)	2	3	4	5 (Very good)
			X	

Table B.5: Evaluation Top

Additional thoughts? (Suggestions, faults, comments, things to improve)
Quando se morre pela primeira vez Quando a diferença de gold muda

B.2 Jungler

What is your age? 22
How long have you been playing LoL? 8 years
Gaming frequency (per week) 40 matches
Do you use tools/websites/apps to analyze your previous games? If so, which ones? Yes Op.gg U.gg Mobalytics Porofessor Mymmr
What game mode do you usually play? Solo queue, with a designated team, professional tournaments or both? I play for a semi professional team in official tournaments, soloq and also ARAM's.
What are your preferred roles to play? Jungle and Support

B.2.1 Tests

A On average, what level does the player reach, at the 5-minute mark?

B Where do they die more often?

C In the last 8 matches, did their team kill Baron first?

How easy was it to answer the proposed questions using the dashboards?

Question	Couldn't Answer	1 (Very Hard)	2	3	4	5 (Very Easy)
A						**X**
B						**X**
C						**X**

Table B.6: Intuitiveness Jungler

How useful was the information extracted from this analysis process?

Question	Couldn't Answer	1 (Useless)	2	3	4	5 (Very Useful)
A					**X**	
B						**X**
C						**X**

Table B.7: Usefulness Jungler

How fast could you get to what you need for the analysis?

Question	Couldn't Answer	1 (Took too long)	2	3	4	5 (It was fast)
A						**X**
B						**X**
C						**X**

Table B.8: Efficiency Jungler

Compared to other tools, did this analysis method offer anything new?

1 (Very Similar)	2	3	4	5 (Very innovative)
				X

Table B.9: Innovation Jungler

How would you evaluate the platform?

1 (Very bad)	2	3	4	5 (Very good)
			X	

Table B.10: Evaluation Jungler

Additional thoughts? (Suggestions, faults, comments, things to improve)
Could show the accumulated XP for the game. Know if you win after the first Herald (1st drake is not that impactful), after the dragon soul or even after the elder dragon buff (rare). Get Jungler proximity (how much time per game you are around top, mid or bot lane) to verify where you normally play.

B.3 Middle

What is your age? 21
How long have you been playing LoL? 7 years
Gaming frequency (per week) 2 days
Do you use tools/websites/apps to analyze your previous games? If so, which ones? U.gg
What game mode do you usually play? Solo queue, with a designated team, professional tournaments or both? Both
What are your preferred roles to play? Mid

B.3.1 Tests

A Where do they kill the most?

B When do they go to the base/buy items?

C Does the player move to other lanes?

How easy was it to answer the proposed questions using the dashboards?

Question	Couldn't Answer	1 (Very Hard)	2	3	4	5 (Very Easy)
A	**X**					
B					**X**	
C					**X**	

Table B.11: Intuitiveness Middle

How useful was the information extracted from this analysis process?

Question	Couldn't Answer	1 (Useless)	2	3	4	5 (Very Useful)
A	**X**					
B						**X**
C						**X**

Table B.12: Usefulness Middle

How fast could you get to what you need for the analysis?

Question	Couldn't Answer	1 (Took too long)	2	3	4	5 (It was fast)
A	**X**					
B						**X**
C						**X**

Table B.13: Efficiency Middle

Compared to other tools, did this analysis method offer anything new?

1 (Very Similar)	2	3	4	5 (Very innovative)
			X	

Table B.14: Innovation Middle

How would you evaluate the platform?

1 (Very bad)	2	3	4	5 (Very good)
			X	

Table B.15: Evaluation Middle

Additional thoughts? (Suggestions, faults, comments, things to improve)

B.4 Bottom

B.4.1 Pre-tests Interview

What is your age? 24
How long have you been playing LoL? 8 years
Gaming frequency (per week) 4 days
Do you use tools/websites/apps to analyze your previous games? If so, which ones? Yes, https://u.gg, https://porofessor.gg/, https://op.gg
What game mode do you usually play? Solo queue, with a designated team, professional tournaments or both? Solo queue and sometime collegiate tournaments
What are your preferred roles to play? AD Carry

B.4.2 Tests

A Where are they at the 9-minute mark?

B In which stage do they start destroying towers?

C When, on average, do they start buying items?

How easy was it to answer the proposed questions using the dashboards?

Question	Couldn't Answer	1 (Very Hard)	2	3	4	5 (Very Easy)
A					**X**	
B						**X**
C						**X**

Table B.16: Intuitiveness Bottom

How useful was the information extracted from this analysis process?

Question	Couldn't Answer	1 (Useless)	2	3	4	5 (Very Useful)
A						**X**
B						**X**
C			**X**			

Table B.17: Usefulness Bottom

How fast could you get to what you need for the analysis?

Question	Couldn't Answer	1 (Took too long)	2	3	4	5 (It was fast)
A						**X**
B						**X**
C						**X**

Table B.18: Efficiency Bottom

Compared to other tools, did this analysis method offer anything new?

1 (Very Similar)	2	3	4	5 (Very innovative)
			X	

Table B.19: Innovation Bottom

How would you evaluate the platform?

1 (Very bad)	2	3	4	5 (Very good)
			X	

Table B.20: Evaluation Bottom

Additional thoughts? (Suggestions, faults, comments, things to improve)

Nos gráficos do mapa, devia ser possível filtrar as posições da equipa.

B.5 Support

B.5.1 Pre-tests Interview

What is your age? 24
How long have you been playing LoL? 11 years
Gaming frequency (per week) 14 matches per week(2 per day)
Do you use tools/websites/apps to analyze your previous games? If so, which ones? Sometimes: Porofessor and Blitz
What game mode do you usually play? Solo queue, with a designated team, professional tournaments or both? Solo queue, ARAM and Flex queue
What are your preferred roles to play? By order: Top → Support → Jungle

B.5.2 Tests

A In which area of the map do they do more assists?

B Where are wards placed?

C Which roles does the player usually take on?

How easy was it to answer the proposed questions using the dashboards?

Question	Couldn't Answer	1 (Very Hard)	2	3	4	5 (Very Easy)
A					**X**	
B					**X**	
C					**X**	

Table B.21: Intuitiveness Support

How useful was the information extracted from this analysis process?

Question	Couldn't Answer	1 (Useless)	2	3	4	5 (Very useful)
A			**X**			
B		**X**				
C			**X**			

Table B.22: Usefulness Support

How fast could you get to what you need for the analysis?

Question	Couldn't Answer	1 (Took too long)	2	3	4	5 (It was fast)
A					**X**	
B					**X**	
C			**X**			

Table B.23: Efficiency Support

Compared to other tools, did this analysis method offer anything new?

1 (Very Similar)	2	3	4	5 (Very innovative)
		X		

Table B.24: Innovation Support

How would you evaluate the platform?

1 (Very bad)	2	3	4	5 (Very good)
	X			

Table B.25: Evaluation Support

Additional thoughts? (Suggestions, faults, comments, things to improve)

Drag mouse move on the map; Better tooltip descriptions; More data accuracy;