StABLE: Making Player Modeling Possible for Sandbox Games

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Luana da Silva Fragoso

Permission to Use

Abstract

Digital games are increasingly delivered as services. Understanding how players interact with games on an ongoing basis is important for maintenance. Logs of player activity offer a potentially rich window into how and why players interact with games, but can be difficult to render into actionable insights because of their size and complexity. In particular, understanding the sequential behavior in-game logs can be difficult. In this thesis, we present the String Analysis of Behavior Log Elements (StABLE) method, which renders location and activity data from a game log file into a sequence of symbols which can be analyzed using techniques from text mining. We show that by intelligently designing sequences of features, it is possible to cluster players into groups corresponding to experience or motivation by analyzing a dataset containing Minecraft game logs. The findings demonstrate the validity of the proposed method, and illustrate its potential utility in mining readily available data to better understand player behavior.

Acknowledgements

I am immeasurable grateful for my supervisor Dr. Kevin Stanley for his insightful guidance during my Master's program. I am very happy of how much I have learned from him and how much he encouraged me to seek the research I have done for this thesis.

I could not have gone so far in this program without the support of my family, friends, and my boyfriend. They were very patient with me on my stressful days and helped me as best they could. I do not have words to say how grateful I am to have these people around me. This thesis exists because of you and I hope you all enjoy it (I will need to translate this thesis to Portuguese for my family though). Thank you!

Contents

List of Tables

List of Figures

List of Abbreviations

StABLE	String Analysis of Behavior Log Elements
GUR	Game User Research
BoW	Bag of Words
NLP	Natural Language Processing
TF-IDF	Term Frequency - Inverse Document Frequency
RQ	Research Question

Chapter 1

Introduction

Interactive digital entertainment is part of life in the Information Age. In America, 64% of households own a device used to play games and 60% of Americans play video games daily [4], with more than 2.5 billion gamers in the world [65]. All of this gaming activity is a significant economic driver, with content sales alone accounting for 29.1 billion USD in 2017, and predicted to reach 138 billion worldwide in 2018 [38]. Video games have moved into the mainstream creating additional pressure on developers to provide high-quality games on strict schedules, and to maintain them after launch to protect in-game monetary streams, which can pass 1 billion USD annually for games like Fortnite [38].

Developers use several methods to obtain data on how their game is performing. Social media and the developer's own feedback boards provide viable methods of collecting self-reported problems from the player base, but data can be distorted by bias. Targeted surveys and in-house testing can provide more select data, such as player reaction based on what they say during the game, but are more expensive and may not identify emerging issues. Companies can collect detailed game logs in either controlled or uncontrolled environments, but these logs, while accurate representations of player behaviour, can be difficult to parse into anything more meaningful than a heat map [26].

Improving game log analysis is an attractive target for developers interested in understanding how players interact with their games. Game logs are entirely under the control of the developers, can include substantial detail about player activities, and can unobtrusively collect data from actual play sessions. However, the data collected can be difficult to parse and even more difficult to obtain actionable insight from. For example, a heat map of player behaviour on a particular map can tell designers where players went, but not why they chose to go there, or logs of in-game purchases can tell you what items were popular, but not how they are being used.

For games which simulate worlds, and movement through those worlds, the traces of where a player went, how they got there, and what they did while in transit could be mined for player behaviours. For example, adventure games, action games, racing games, open world simulators, and role playing games all have moving through the world and performing actions as primary mechanics. If one accepts the hypothesis that why you play is reflected in how you play [68], proper analysis of these logs could access aspects of player motivation and affect. However, the long strings of location IDs and action labels are difficult if not impossible for a human to process directly, and not amenable to traditional statistical analysis.

1.1 Problem Statement

Practitioners of Geographic information science (GIScience) have proposed features that can be extracted from location and action strings [23, 28], and other researchers have attempted to parse these features to extract insight from game logs [39, 46, 48], but no one has proposed a general solution for the comparison of game log location or action strings for unstructured games, also known as sandbox games [35]. Sandbox games are becoming extremely popular, for example, Minecraft has more than 100 million active players monthly [31].

In this thesis, we propose the String Analysis of Behavior Log Elements (StABLE) technique to directly compare sequential movement and/or action records, and demonstrate that it is versatile, efficient, and able to logically discern important play style and play motivation patterns amongst users. Our method keys on the observation that location and activity patterns through time can be represented as long strings, and therefore could be amenable to Natural Language Processing (NLP) techniques. NLP involves a set of mature machine learning techniques for the analysis and comparison of long strings.

We implemented StABLE with the canonical Bag of Words, N-gram, and cosine similarity tools [1] to compare different aspects of player behaviour based on three different string constructions: location construction, dwell-trip construction, and dwell-trip-action construction. These string constructions are based on movement metrics from real world applications, such as visit frequency, dwell time, and trip duration [23, 28, 49]. Visit frequency is the number of times a player visited a location. Dwell time is the duration a player stayed in the same location. Trip duration tracks how long a player moved from a location to another. The distributions for each metric is generated through Bag of Words and N-grams and are used to compare player behavior with the cosine similarity equation. We evaluated StABLE by using Minecraft game play data collected to address the following research questions:

RQ1 Do players have similar visited frequency behaviors in the game?

RQ2 Do players who explore more of the game present similar visited frequency?

RQ3 Do players change their visited frequency behavior through the game?

RQ4 Can we cluster the players based on the self reported motivation?

RQ5 Do players who explore more of the game present more similar movements and activities than players who did not?

RQ6 Do players have similar movements, with and without activity, when compared to themselves at the beginning and end of the game?

Statistical tests were used for each question to validate the results by identifying significant differences between-player and within-players behavior. A within player analysis investigates the behavior of a single player as the game proceed, whereas a between focusses on comparing the behaviors of different players. These six questions are examples of the kinds of analysis enabled by the technique we are proposing, demonstrating its breadth, but are not exhaustive. At its core StABLE is about comparing time series or sequential data, and creating similarity graphs from the results. The meaning of the comparison is embedded in the features which comprise the series, and the prior or external information available. The following results showed that StABLE can, indeed, be used for analyzing players behavior in open world and endless games. While the findings in the next section constitute contributions in and of themselves, our primary contribution is the technique.

1.2 Contribution

An open-world player modeling can not be based on a domain knowledge since players are free to play how they want to [35]. In this thesis, we proposed StABLE, a technique that models players behavior and identifies similarity between and within players with minimal designer input, making player modeling possible for sandbox games. We were able to achieve such a model by encoding players behavior as strings of movements and actions. Natural Language Processing (NLP) techniques could then be easily applied to quantify the similarity between and within players. While we are not the first to propose cosine similarity in game log analysis [39], we are the first to demonstrate that by altering the string construction, one can answer different questions about players behaviour. How strings are created has a significant impact on the ultimate interpretation of the comparisons. We created three different string constructions in this thesis: location-construction, dwell-trip-construction, and dwell-trip-action-construction. Because of the generality of StABLE, we are also the first ones to propose a method for player modeling for open world and endless games from mobility game log analysis. In summary, we found the following results by using the Minecraft game log:

- Direct comparisons of visited location are possible.
- By aggregating to trip properties (trip duration and dwell time) players cluster according to their self-reported motivation for playing the game.
- By aggregating to trip properties and including action state, players cluster according to depth of the game explored. Players who explore more of the game have more similar trip patterns than players who not.
- By stratifying by time, changes in player behavior as the game progresses can be determined. Players with more experience have more stable play patterns.

The process of rendering log patterns and sequences of features, and subsequently performing similarity analysis on those features is the core methodological contribution of this work. The findings we described here have strong internal validity, but weaker generalizability given the single game, limited demographic, and relatively small number of participants. However, the intent of this thesis was not to provide strongly generalizable findings relating movement patterns to experience, but to demonstrate that StABLE, as a tool for game log analysis, shows utility. The tool itself, and more importantly the concept underlying the tool, has broad applicability and can be used to analyze many types of sequential log data, including data from sandbox games. While we focused on movement data in an open world game in this work, many other genres of games and streams of data, and player attributes could be analyzed. This technique is not limited to games, and could be broadly applicable to Human Computer Interaction in general. For example, eye tracking data or galvanic skin response data could also be meaningfully analyzed using this approach. Essentially, the string represents features of interest of player behavior through time, from which designers hope to gain insight through comparison.

1.3 Organization of Thesis

This thesis consists of six other chapters besides the introduction where the motivation of the study, problem statement, and contributions were stated. Chapter 2 presents the literature review for player modeling, movement modeling, and behavior similarity. Chapter 3 provides theoretical background for the main concepts used in this thesis. Chapter 4 describes the methodology proposed. Chapter 5 explains how we built the experiment to evaluate StABLE. Chapter 6 shows the results. Chapter 7 discusses the results as well as the shortcomings and future works, and concludes the thesis.

Chapter 2

Literature Review

To obtain the knowledge to build StABLE, three main topics had to be explored: player modeling, movement behavior, and behavior similarity. The applicability of StABLE is on player modeling where designers can use it for player behavior analysis. The literature review for this topic is covered in two main approaches: subject-driven and data-driven. Subject-driven approaches provide an analysis of why players play games whereas data-driven approaches uncover player behavior patterns. Hybrid methods are also mentioned in this chapter, which are the combination of both approaches and the type used in StABLE. Existing methods provide player modeling for games with goals and an end, but not for unstructured games. Sandbox games cannot benefit from current player modeling methods. With open world maps, we addressed this gap by incorporating movement behavior studies from real world to virtual world. Therefore, we explored how such studies defined and described movements and applied these same concepts to virtual movements. The last step on StABLE is to compare player movements. Behavior similarity approaches were reviewed in this section where some of them encode behaviors as strings to apply Natural Language Processing (NLP) techniques, which offer a variety of methods to quantify similarity between text documents. By encoding player behavior as strings, we could quantify and compare their similarities based on NLP methods. We revised NLP techniques to know the pros and cons of the existing approaches.

In summary, this chapter provides a literature review in player modeling, movement behavior, and behavior similarity. We divided the *Player modeling* section into three topics: subject-driven, data-driven, and hybrid methods. In the subsection *Movement behavior*, we extended our literature review beyond game user research (GUR) papers to include real world papers since movement behavior is well explored in human science fields. The discussion of the similarity metrics are found in the section *Behavior Similarity.* Finally, a summary is provided at the end of this chapter to highlight the contributions of this thesis for player modeling.

2.1 Player Modeling

Player modeling has multiple applications in game design, [12] from dynamic difficulty adjustments [29] to churn prediction [53], to discovering in-game patterns (behaviors) to classifing players' actions, motivation, and personality [58]. Understanding why and how players behavior in a specific manner is an important field in Game User Research (GUR) [27,44]. There are two main approaches to model players behavior: subject-driven and data-driven [67]. The first, and still a popular approach to player behavior modeling, is based on the collection of subjective data through surveys and observational data [6,68]. With the availability of data and data analytic tools, data-driven approaches have began to be explored [27,35,44]. While subject-driven approaches address why players play videogames, they are difficult to collect and might not be accurate as many questionnaires are based on player's mood, which often changes during the game. Data-driven approaches show the opposite. Hybrid methods combine these approaches with the attempt to answer why and how players behavior in a particular way. [12,56,57,61].

2.1.1 Subject-driven

Bartle [6] introduced one of the first papers in player modeling by observing a discussion of players of a multi-user dungeon (MUD). He noticed that there were four type of players: Achievers, Socializers, Explorers, and Killers. For example, a player is an achiever because he or she would say to another player "*Sure, I'll help you. What do I get?*". However, only 30 participants took part in the study and only 15 of them were actively participating. Bartle did not show if the four types were actually independent. Yee [68] addressed these problems by surveying 3,000 participants over 40 questions related to the four types found by Bartle [6]. An example of one of the presented question is "*How important is it you to level up as fast as possible?*", using Principal Component Analysis (PCA), which identifies the dimensions that account for most of the variance between them, Yee found three main player behaviors related to achievement, socializing, and immersion.

Selecting the right questions to collect subjective data is not an easy task and requires domain knowledge [27]. Therefore, survey-based player modeling studies commonly build their questionnaires based on consolidated personality tests from psychology [21,61,62,69]. Among the psychological models, The Five Factor Model (FMM) is the most popular one and can be evaluated through different tests such as the Big Five Inventory questionnaire (BFI) [37] and the NEO Personality Inventory-Revised (NEO PI-R) [17]. The FFM classify individual's personality into five dimensions: Extraversion, Agreeableness, Conscientiousness, Neuroticism, and Openness. However, only five personality traits may not be enough to describe an individual's personality [42] and may not be suitable to obtain actionable information of what drives an individual to behave in a such manner [14,67].

Motivation models, on the other hand, describes why a behavior happened based on a desire that triggered such a behavior, divided into internal (intrinsic) motivation and external (extrinsic) motivation [20,52,60]. In their canonical studies [19,51], Deci and Ryan extended motivation into four categories: intrinsic motivation, identified regulation, external regulation, and amotivation, where the identified regulation and external regulation are a finer classification for the extrinsic motivation. A popular motivation test is The Situational Motivation Scale (SIMS) by Guay, Vallerand, and Blanchard [32] which evaluates an individual's motivation into these four categories.

Although subject-driven player modeling allows to interpret the data to understand *why* players behavior in such a way, there is a list of limitations with this approach. As observed in Bartle [6], not all the participants are active during the questionnaires. The lack of participation is commonly found in subject-driven studies where the response rate for questionnaires is very low [34]. Player behavior analysis relies on the selected questions, which varies depending on what meaningful information one requires [27]. To gather more information, a more complex questionnaire would need to be constructed [34]. Another challenge is that player behavior changes through time [5]. In this case, multiple surveys would need to be answered in different moments, which can be time-consuming and a burden for players [27]. These limitations inhibit the creation of a general method since each game would need a specific set of questions and strategy on how and when to apply them.

2.1.2 Data-driven

With the huge amount of events being collected through video-games, researchers began to model player behavior based on log data. In contrast to the subjective approach, data-driven player modeling does not require as much expert domain knowledge. Player behavior patterns can be retrieved from the data itself and allows the creation of more general models [35, 66]. Different methods have been proposed to model players behavior with a data-driven approach, especially data visualization and goal prediction.

Data Visualization is one of the most popular analysis tool for modeling in-game behaviors. Heatmaps are commonly used in Game User Research (GUR) to visualize the frequency of a feature, usually death and visited locations [15, 25, 45]. Because of its limitation of only showing a single feature, some research has proposed more complex visualizations. Recently, Wallner et al. [64] improved data frequency visualization by showing more than one aggregated feature in the virtual map. However, the addition of such features polluted the outcome and designers started to feel overwhelmed by the information, especially for a between-player behavior analysis [64].

Playtracer is another tool proposed by Andersen et al. [2]. Their goal was to create a tool to be used independently of a game's structure. They were able to construct such a tool by creating a graph where the nodes represent a state in the game and the edges are the player's path, showing the direction from one state to another. The graph captures the aggregated data for all players. To project the aggregated data into a 2D graph, designers need to specify a state-distance metric so that similar states are placed close to each other and dissimilar states are placed far apart. Although Playtracer made a significant contribution in the literature, it had limitations. Besides the challenge of choosing important states and a distance metric for them, it is also not intuitive to answer finer question such as "*What are the top 10 most popular sequences?*" as Nguyen et al. [46] argued in their work.

Playtracer motivated the creation of Glyph [46], which provides two visualizations: a state graph and a sequence graph. In contrast with Playtracer, Glyph's state graph summarizes the nodes with transition to each other into a single node, providing a cleaner representation. Glyph offers a tool to show the popularity and similarity of sequences, which can answer finer grained questions. Each node in the sequence graph represents a sequence. The closer the nodes, the more similar the sequences are. The distance between the nodes for the state graph is determined by the force directed placement algorithm [18], whereas for the sequence graph, the distance is calculated by Dynamic Time Warping (DTW) [9]. DTW finds the best alignment between time series, overcoming speed differences between them. For example, DTW is commonly applied in speech recognition because people speak at different speeds. The best alignment is the optimal match based on the sum of the minimal distances of the points, and the following restrictions:

- Every point from the first time series must be matched to one or more points in the second time series and vice versa.
- The first point from the first time series must match the first point from the second time series. The same restriction applies for the last point.
- If j and i are points from the first time series, where j happened before i ($j < i$), and l and k are points from the second time series, where l happened before k ($l < k$), the following scenario cannot happen: j matches with k, and i matches with l.

Osborn et al. [48] proposed a similar visualization to Glyph but instead of showing a sequence of game states, their visualization shows a sequence of game actions. They claim that with an action based model, they can build a more general and efficient model. Instead of using DTW to group similar behaviours, they used a tool they created in a previous work called Gamalyzer [47] by using the Constraint Continuous Edit Distance (CCED). The Edit Distance (ED) algorithm counts the number of minimal operations (change, insert, and delete) that are needed to transform a sequence X to a sequence Y. The lesser the total of operations, the higher the similarity between the sequences. CCED is a variant of ED with constrains. Gamlyzer requires three conditions: only the change operation can be used, the changes have different weight (e.g., changing from 1 to 10 costs more than 1 to 2), and the Sakoe-Chiba Band constrain. The latter establishes a window where the points beyond this window are not compared, preventing the algorithm to compare too different sequences. Both Glyph [46] and the Gamalyzer visualization [48] can be hard to read for a large amount of data, and are limited to games with defined goals.

Player behavior prediction and classification has been recently explored in the literature using machine learning. Most of the research built their predictive and classification models based on goals and player achievements [35, 66]. Min et al. [43] started addressing goal prediction in open world games with an end using a deep-learning algorithm. In their approach, goals and locations need to be labeled beforehand to train the model, which might not be an easy task for more complex games. They evaluated their model with the Crystal Island game and they found promising results with the proposed algorithm. However, deep learning algorithms are known to be black-box algorithms because it is extremely difficult for one to understand how the model fitted the data, which can limit the understanding of why players behavior were clustered in a particular way.

Harrison and Roberts [34] developed a data-driven approach to predict players' behavior based on past observations of other players. They assumed that players with similar behaviors in the past would show similar behaviors in the future. By creating a set of actions as cliques from other players, they determine the likelihood of a player to complete one of the cliques given his/her past actions. Although they achieved a better prediction accuracy than a base line algorithm, finding cliques can be a costly task.

2.1.3 Hybrid methods

Other research used both data-driven and subject-driven approaches to find patterns in player's personality. Bunian et al. [12] modeled player's actions using a hybrid method with a Hidden Markov Model (HMM) to predict individual differences related to their expertise and personality. Each state of the chain can be understood as player's behavior (not observable) which is derived from a set of actions (observable) that happened during the game. For example, the behavior *Social* includes any action related to messaging or interacting with other players. A list of pre-defined actions is needed. From the resulting model, they generated a new feature by collecting the frequency of each state in the HMM and compared these states to the top and low player's personality traits and expertise, which were collected through surveys. They obtained a better prediction accuracy with an HMM than by just aggregating the data.

Canossa et al. [14] captured in-game behaviors to understand what drives players to play Minecraft. Using game features like location and actions to correlate with data collected through surveys, they were able to investigate what kind of player plays Minecraft. The motivation test used in their work was the Reiss Motivation Profile [50]. Reiss [50] assigned intrinsic motivation to mind related behaviors (e.g., curiosity) and extrinsic motivation to physical behaviors (e.g., hunger). The majority of the motivation that triggered participants to play Minecraft were curiosity, saving items, independence, honor, and idealism. They also predicted life motives (hunger, tranquility, etc.) beyond other variables such as age, hours played, and education from the 20 in-game features with highest correlation to the motivation attributes. Among the variables, gender was the easiest one to predict because of the highly skewed gender dataset.

Although many papers claimed to provide more general player modeling, most of them can only be applied for goal oriented games [12, 34, 35, 66] or games with a small world [43]. Sandbox games cannot benefit from these models since players are free to explore a huge open-world game as they want, without any goal in mind [35]. Even though one may think that such methods can still be applied in sandbox games, most of them require labels for actions, states and locations. It is challenging to select the most important features or even to label them in sandbox games because players usually have a variety of play options. As stated by Hooshyar et al. [35], a more general and fully automatic player modeling is needed for modeling players in infinitely open and replayable worlds.

2.2 Movement behavior

We believe we can capture essential features of player behavior based only on movements and actions. Some research focused on clustering players based on their movements [13]. They analyzed different clustering algorithms and metrics to cluster similar player's path where they found that the Fréchet distance metric [3] worked the best with the clustering algorithm they tested. Fréchet distance is similar to DTW, but instead of attributing the optimal path as the sum of the minimal distances of the points, it uses the maximum distances. They used a high dimensional vector containing in-game behavior information to cluster players. Finally, all the trajectories were plotted in a game world map with different colors based on the groups found in the previous step to understand how each group explored the world.

Bauckhage et al. [7] proposed a clustering technique to group players' behavior based on their movements for more complex 3D maps. They accounted for the problem of asymmetric relations between locations in 3D world. For example, jumping from a ledge can quickly lead the player to a location, but the other way around takes longer. By showing the final cluster in a 3D plot, the authors could identify hotspots in the game and gain insight of players behavior. However, by analyzing such behaviors as virtual plots can result in information overload that can overwhelm designers, a problem also found in Playtracer, Glyph, and Gamalyzer [2, 46, 48].

Movement behavior is extensively studied for real world applications from using data from geographic information systems (GIS) or the global positioning system (GPS) [41, 71]. Farrahi et al. [30] leverage human locations as strings to find similarity between-human and within-human routines. From cell tower signals, they could classify individual's locations into three labels: *home*, *work*, and *or out* to create a Bag of Words that counts how many of each label was visited per day. Besides the location, they also included the time (day of week and hour) to their analysis. By representing participant routine behaviors as strings and using a text mining technique called Latent Dirichlet Allocation (LDA) [10], the authors could answer questions regarding routines, for example, where and what time participants usually work. They found that the weekday work patterns differ from weekend work patterns. Farrahi et al. [28] used the bag-of-word technique described in their previous work [30] for an urban planning perspective from location-based social networks data. They found patterns that identified hotspots in the city and recurrent crowd behaviours in these places.

Yuan and Raubal [70] also created strings and applied the Edit Distance algorithm to uncover similar mobile trajectories. Based on cell tower phone calls, they tracked the position of participants and assigned a label for each cell to describe their trajectories. Similar to Faharri et al. [30], they included temporal information. They adapted the classical Edit Distance algorithm, similar to the Dynamic Time Warping (DTW) algorithm, to incorporate locations, with changes to adapt the algorithm to balance the influence of spatial and temporal dimensions. They found that people tend to visit regular locations and tend to show smaller distance movements during weekends.

Movement behavior can be modeled beyond visited locations as shown in research using GIS and GPS log data [24, 49, 55, 72]. Scherrer et al. [54] used 32 features derived from absolute locations and time to describe human movement behavior, such as the total distance the participant has covered, the average area covered daily, and the average stop duration. By applying clustering methods, they could distinguish groups of travelers and locals in Sydney.

Tuhin et al. [49] analyzed only five aggregated metrics (visit frequency, dwell time, trip length, trip duration, and radius of gyration) for human movement and found that except for radius of gyration, all of them conserve the power law distribution shape for different spatio-temporal resolutions and datasets. This trend was also found in other researches [40,59]. As a result, these metrics showed to be promising to be used for any location-based dataset. Dwell time and trip duration have been used in other papers [24,40,55,59,72].

The most similar movement modeling to our work is Dogde et al. [23]. They found trajectory patterns from hurricanes based on movement parameters (MP) such as speed, acceleration, and direction, instead of the locations and time directly. Next, they created a new path based on labels of these MP. For example, if S indicates speed and A accelaration, the new path could be a sequence of S's and A's: [SAAASSAA]. Finally, they computed the similarity score based on the DTW metric. They compared their work with previous method and found that their method is more effective in extracting similarities. Our work is similar in terms of the novelty of using movement parameters (MP) instead of the spatio-temporal data. However, we used different MP: dwell time and trip duration. We thought that speed and acceleration are less relevant in virtual world games where a lot of games present constant velocity, such as Minecraft.

2.3 Behavior Similarity

Behaviour similarity is important for behavior analysis since almost all the methods use a similarity metric to find interesting patterns. Similarity metrics are commonly used in Natural Language Processing (NLP) to address problems in text and speech recognition. Some research have shown that the similarity score can be a good feature to predict player behavior [12,48]. Alain et al. [58] tested different similarity metrics from NLP to cluster player behavior and found that each of them has a singular drawback – sensitivity to outliers. They also found that some common distance metrics, such as the Fréchet distance [3] did not provide a satisfactory performance.

A popular similarity metric is based on the Edit Distance algorithms [9,16,63]. Glyph [46] and Gamalyzer [47], respectively, used a variant of the Edit Distance algorithm called Dynamic Time Warping (DTW) [9] and Constraint Continuous Edit Distance (CCED) [16], DTW was one of the similarity metrics tested by Alain et al. [58]. DTW quantifies the difference of unequal sequence length by finding the optimal alignment that provides the best match between them [9]. Glyph [46] uses DTW to calculate the similarity between states in the game, where the distance metric used for DTW in Glyph was the Edit Distance. In other words, the difference between two states is the amount of actions required to lead from a state to another. The higher this value, the lesser similar are the states. This approach is similar to the one used in Gamalyzer [48].

Gamalyzer is based on CCED but with some adaptions for goal modeling. Gamalizer requires as input a lexicographical list ordered by the importance of the feature for the game. For example, in Chess, a piece is more significant than its location, however, X and Y coordinates have the same importance. In this scenario, the input is $(piece, [X, Y])$, where the parenthesis orders the features by importance and the brackets indicates that features have the same importance. To quantify the similarity between inputs, the number of minimal operations to transform an input to be exactly the same as the another is calculated, giving more weight to the most important features. One of the main drawbacks of Gamalyzer is that it requires designers to provide a threshold to limit the length of the sequences to be compared, i.e., the window size for the Sakoe-Chiba Band constrain. Also, both DTW and CCED can have a heavy computational cost [46, 48, 70].

Other similarity metrics commonly found in the literature are based on probabilistic models. One of the most popular probabilistic algorithm is the Latent Dirichlet Allocation (LDA) [10]. This algorithm determines the probability of a document belonging to a topic by calculating the probability of the words in a document being part of the existent topics. This algorithm needs three parameters: the number of possible topics, and two numbers between 0 and 1 for the *alpha* and *beta* parameters. A high *alpha* value leads to documents being more similar in terms of topics, whereas a high *beta* value leads to topics being more similar in terms of words they contain. LDA, therefore, requires a priori knowledge that is not necessary intuitive.

Popular and basic similarity metrics in NLP include Euclidean Distance, Jaccard Similarity, and Cosine Similarity [36]. Euclidean distance is one of the most famous geometrical metrics and calculates the shortest distance between two points. Each element in these vectors represents the frequency of a word in a sentence. A distance closer to 0 means that the sequences are more similar. Jaccard similarity calculates the similarity of common words by dividing the number of the same words in both sequences to the total number of the words. A similarity of 0 means that the texts are completely different, where a value of 1 means they are identical. Cosine similarity measures the cosine of angle between two vectors. The vectors are the same for the Euclidean distance and the similarity score is the same as the Jaccard similarity, between 0 and 1. In comparison to Euclidean distance, cosine similarity is not affected by the magnitude of the vector. On the other hand, in contrast to Jaccard similarity, duplicate words affect the result in cosine similarity.

Cosine similarity was used in the work done by Lee et al. [39]. They reported a general approach to identify bots based on player self-similarity. They created vectors to represent the amount of activity a player performed over a specific time frame. They employed cosine similarity to evaluate the self-similarity within players through the game. They found that bots have fewer variations in their similarities through time, whereas players showed more variation. Similar to them, we chose to use cosine similarity to quantify players' behavior because no additional parameters are required.

2.4 Summary

We believe we can capture essential features of players behavior in a general manner based on movements and actions by using NLP techniques. The resulting similarities can be used for between-player and within-player behavior analysis. A within-player analysis investigates the behavior of a single player as the game proceed, whereas a between-player analysis focus on comparing the behaviors of different players. To the best of the author's knowledge, none of the current papers presented a player modeling which uses N-grams with Bag of Words based on movement and action sequences, aiming to make analysis possible in sandbox games. Although Gamalyzer [48] also uses lexographical tokens to represent actions, it does not use Bag of Words nor N-grams to model players behavior and the purpose of the technique is to find goal similarity, whereas in our work we look for players similarity. Min et al. [43] analyzed the accuracy of their deep learning goal recognition algorithm with different N-grams. They found that higher N-grams provides higher accuracy. Similar to their work, we also analyzed the sensitivity of N-grams, but for player similarity, not goal recognition. Another difference is that their approach only works for games that pursue an end. To extend player modeling for endless games, we modeled players behavior based on the movement parameters (MP) dwell time and trip duration. A similar movement modeling as done by Dodge et al. [23] for hurricanes, but using different MPs. We encoded these behaviours as strings like in many papers for real world applications [23, 28, 70] and GUR [46, 48]. We then employed cosine similarity as Lee et al. [39] to quantify player similarities. While Lee et al. [39] provided a general modeling for bot detection, we focused on addressing a more global scope of identifying similar players' behavior. To obtain an insight of *why* these players are similar or different, we analyzed the similarity results by applying statistical tests based on motivational surveys and player expertise like in Bunian et al. [12] and Canossa et al [14]. However, different to them, we not only applied a within-player analysis but also a between-player analysis. A within-player analysis investigates the behavior of a single player as the game proceed, whereas a between-player analysis focus on comparing the behaviors of different players.

Chapter 3
Background

This thesis is built on three concepts: movement behavior, text similarity, and clustering. Movement behavior is how we modeled players behavior based on dwell time and trip duration. Text similarity quantifies similar sequence of strings, in our case, players' behavior. We used tokenization, N-grams, and Bag of Words to encode players behaviors into a vector of words. These vectors were compared using the cosine similarity which quantifies how similar these strings are. Finally, we evaluated the similarity scores by using a clustering algorithm to group players and evaluated cluster properties. From the many clustering algorithms available, we used the unsupervised algorithm Louvain to cluster players. We chose Louvain over other clustering methods because it does not require a priori knowledge of the graph shape.

3.1 Movement behavior

We first need to define movement. We used dwell time and trip duration to detect player movement. We also included visit frequency to show that our methodology can still be used with classical analysis, such as heatmaps. Figure 3.1 and Figure 3.2 illustrate visit frequency, dwell time, and trip duration. The first figure shows where the Pacman was in a sequence of six timestamps. There are a total of four possible locations: A, B, C, and D. The second figure shows the resulting movement distributions. Based on the work by Tuhin et al. [49], visit frequency, dwell time, and trip duration can be defined as following:

Visit frequency is the distribution of the number of times each player visited a specific location. This metric represents the popularity of a location.

Dwell time is the distribution of the duration spent at the same location. This metric offers the detail of visited places for short and long times.

Trip duration is the distribution of the duration spent on a trip. A trip is triggered if a player changes location and dwells for no more than three consecutive timestamps. The same threshold was used by Paul et al [49].

In Figure 3.1, Pacman visited the location A four times, D two times, and B and C never. Because we consider a dwell time for 3 timestamps or less as part of a trip, in this example, Pacman only dwelled in location A since it stayed in this location for more than 3 timestamps in sequence. Pacman took a trip of 2

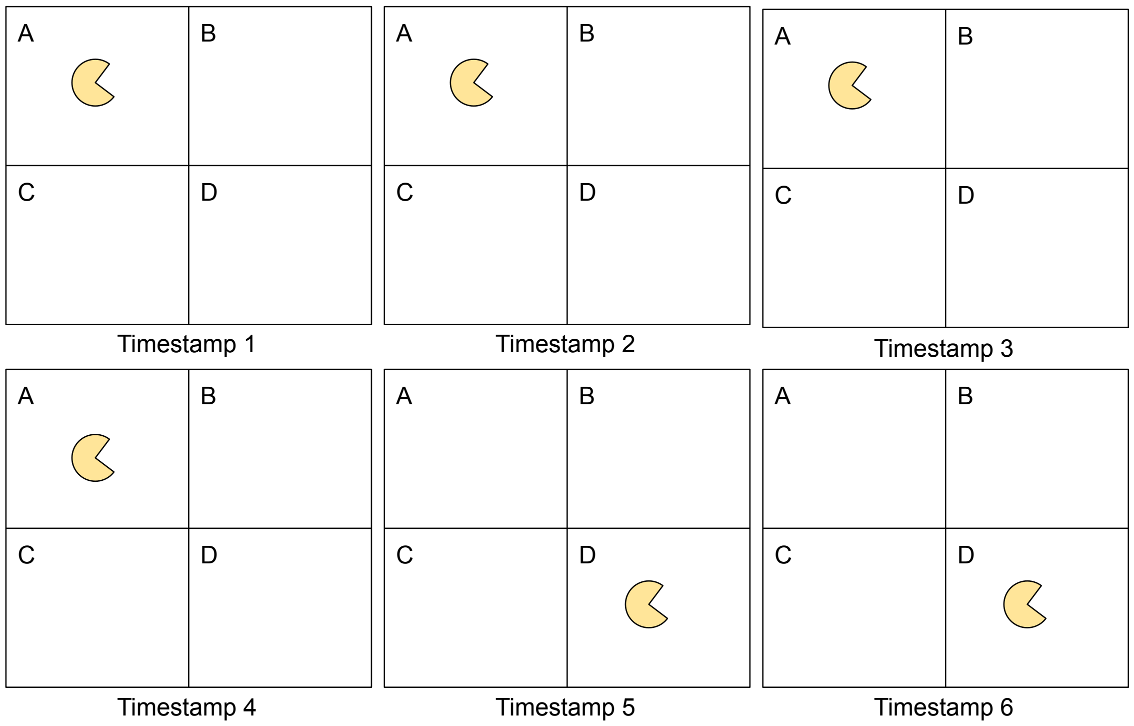


Figure 3.1: Example of movement in six timestamps and four different cells: A, B, C, and D.

timestamps. One timestamp from A to D, and another timestamp staying in D, since this dwell is considered part of the trip. The resulting visit frequency, dwell time, and trip duration distributions can be seen in Figure 3.2.

3.2 Text Similarity

A common problem in text similarity is that each piece of text has different size and content, and most of the similarity techniques, including cosine similarity, require vectors with same size as input. Existing methods in NLP to overcome this problem include tokenization, N-gram expression, Bag of Words (BoW), and term frequency and inverse document frequency (TF-IDF) [1]. To illustrate these techniques we consider these two lower case strings:

keep calm and play minecraft. next, keep calm and craft on.

keep calm and mine on.

Tokenization is the process of breaking up a string into unique smaller units (tokens), typically words. A word starts when a blank space occurs and ends when the next character is a blank space. The unique words are then added into a vector. From our first string, the tokenization produces 8 tokens: [“*keep*”, “*calm*”, “*and*”, “*play*”, “*minecraft.*”, “*next,*”, “*craft*”, “*on.*”]. The second string results in a vector of 5 tokens: [“*keep*”, “*calm*”, “*and*”, “*mine*”, “*on.*”]. Notice that each vector does not contain duplicate tokens.

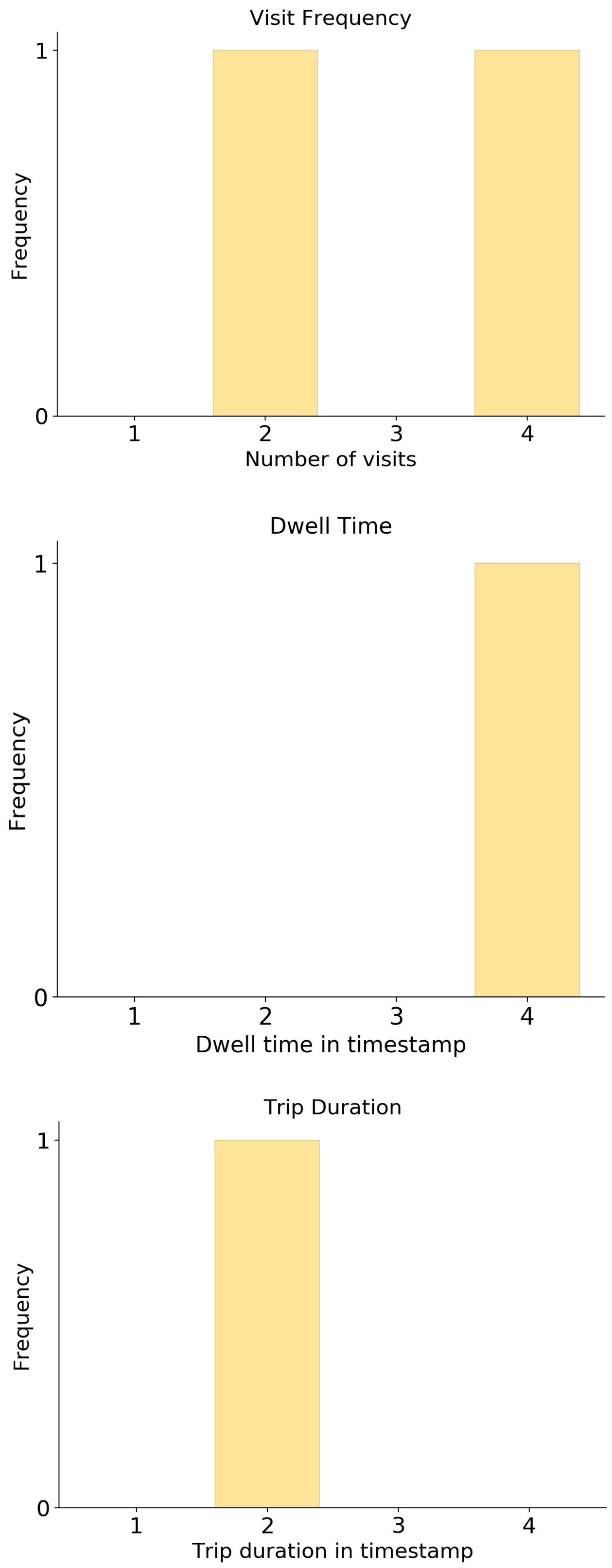


Figure 3.2: Movement distributions from top to bottom: visited frequency, dwell time, and trip duration.

N-grams capture N words in sequence by treating them as new tokens. The first token is built by taking the first N words. The next tokens are built by repeating the second word of the previous token followed by the next (N-1) words. This process is executed until the end of the string. In the previous paragraph, we showed the unigram result ($N = 1$) for the first string, which led to 8 tokens. If a bigram ($N = 2$) was used instead, the resulting tokens would be [“*keep calm*”, “*calm and*”, “*and play*”, “*play minecraft.*”, “*minecraft. next,*”, “*next, keep*”, “*and craft*”, “*craft on.*”]. As a result, in a bigram, each token has 2 consecutive words from the string. On the other hand, an N-gram of 11 would result in the original string since this string has 11 words.

Bag of Words (BoW) combines two or more strings by taking the union of the observed tokens from each string. As a result, from our two strings, we have the following BoW with an unigram: [“*keep*”, “*calm*”, “*and*”, “*play*”, “*minecraft.*”, “*next,*”, “*craft*”, “*on.*”, “*mine*”]. Finally, each string has a histogram vector representing the count of the tokens. The vector of the first string is [2, 2, 2, 1, 1, 1, 1, 1, 0], and the vector of the second string is [1, 1, 1, 0, 0, 0, 0, 1, 1]. BoW normalizes the length of the vectors for comparison by transforming the strings into token histograms. When BoW is combined with N-grams, the histogram can be built for a specific N-gram, or for a range of N-grams. For example, a BoW built from trigrams ($N = 3$) would be a histogram of all three word sequences in the text, but a BoW built over a range $N = \{1, 2, 3\}$ would contain a histogram containing all single words, all two word sequences and all three word sequences in the text. In our example, the BoW would have 29 tokens in total: [“*keep*”, “*calm*”, “*and*”, “*keep calm*”, “*calm and*”, “*keep calm and*”, *...*].

Term Frequency - Inverse Document Frequency (TF-IDF) normalizes the count of the token histograms to fairly compare strings with unequal length. Term Frequency (TF) is the ratio of the count of a token T in a string S divided by the number of tokens in the same string.

$$TF = (S_t)/(S_n) \tag{3.1}$$

where S_t is the number of occurrences that T appears in S and S_n is the total number of tokens in the string. Unusual tokens (*craft*), that are present in the documents should have more weight than common tokens (*and*). The Inverse Document Frequency (IDF) measure addresses this problem.

$$IDF = \log(D_n/D_t) \tag{3.2}$$

where D_n is the total number of strings to be compared and D_t is the number of strings where T is present. The log is part of the equation to dampen the effect that the relevance of a document does not increase proportionally by the presence of T. Term Frequency - Inverse Document Frequency (TF-IDF) is the product of TF and IDF.

$$TFIDF = TF \times IDF \tag{3.3}$$

Cosine Similarity is a canonical equation to calculate similarity between vectors as the cosine angle between the vectors. Using the angle instead of the magnitude avoids increasing the similarity significantly for longer strings as common words tend to appear more. The cosine similarity of two documents ranges from 0 to 1 [36], where 0 means they are not equal and 1 means they are completely similar. Cosine similarity is represented as:

$$similarity = cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||||\mathbf{B}||} \tag{3.4}$$

Because of the normalization when using TF-IDF, the vectors present a magnitude of 1. In this case, the cosine similarity is only the dot product between the vectors:

$$similarity = cos(\theta) = \mathbf{A} \cdot \mathbf{B} \tag{3.5}$$

The output from cosine similarity is a matrix of N x N, where N is the number of vectors (strings) to be compared. Each element in this matrix it the similarity value between each string. This matrix is a symmetric matrix with a diagonal populated with values 1.

3.3 Situational Motivation Scale (SIMS)

The SIMS test measures the motivation of an individual. The test identifies four motivational categories: intrinsic motivation, identified regulation, external regulation, and amotivation. *Intrinsic motivation* is when the behavior is performed for the pleasure and satisfaction of performing an activity by itself, without any external influence. *External regulation* occurs when the behavior is affected by rewards or pressure to avoid a negative outcome such as a posted high score. *Identified regulation* happens when a behavior is showed to be valuable for an outcome. *Amotivation* behaviors are when individuals feel incompetent and without any control, which does not fit as an intrinsic nor extrinsic motivation.

Based on the question *Why are you currently engaged in this activity?*, participants evaluate 16 items as enumerated in Table 3.1 with the following scale: strongly disagree (0 point), disagree (1 point), slightly disagree (2 points), neutral (3 points), slightly agree (4 points), agree (5 points), and strongly agree (6 points). Each question assess a motivation. Intrinsic motivation is assessed on questions 1, 5, 9, 13; identified regulation: 2, 6, 10, 14; external regulation: 3,7, 11, 15; amotivation: 4, 8, 12, 16. To calculate the final SIMS score, the sum of the points answered by the participant is divided by the maximum possible points (96 points).

Table 3.1: SIMS test questionnaire

Why are you currently engaged in this activity?
1. Because I think that this activity is interesting
2. Because I am doing it for my own good
3. Because I am supposed to do it
4. There may be good reasons to do this activity, but personally I don't see any
5. Because I think that this activity is pleasant
6. Because I think that this activity is good for me
7. Because it is something that I have to do
8. I do this activity but I am not sure if it is worth it
9. Because this activity is fun
10. By personal decision
11. Because I don't have any choice
12. I don't know; I don't see what this activity brings me
13. Because I feel good when doing this activity
14. Because I believe that this activity is important for me
15. Because I feel that I have to do it
16. I do this activity, but I am not sure it is a good thing to pursue it

3.4 Louvain Algorithm

The Louvain algorithm [11] is a greedy community detection method that maximizes the modularity in each community. A greedy algorithm is defined as making locally optimal choices in each step in an attempt to reach a globally optimal result. Modularity quantifies how many connections the nodes in a community present (community density) compared to the density of a community would be if a new random node is assigned to this community. If the modularity is higher, the new node is attached to the community. Modularity has a value between -1 and 1 [11]. Louvain is greedy because it first detects small communities by finding the maximal modularity locally, and then calculates the maximum modularity between communities to find the global optimal communities. Therefore, Louvain works in two steps: community detection and community aggregation. First, the algorithm detects communities from a graph network. An arbitrary node is chosen and aggregated to each of its neighbours. This node and its neighbour that shows a higher modularity are grouped in a community. The algorithm repeats these processes until a local modularity maximum is reached [8]. Once all nodes were merged, some communities will have been formed. Next, community aggregation is applied, where the communities from the previous step are treated as nodes. The algorithm in the first step is then applied to group the communities in to larger communities. Louvain is faster than other community detection algorithms based on modularity optimization [11]. We used Louvain to create communities from the relationships derived from cosine similarity.

Figure 3.3 illustrates the above explanation. The algorithm starts with a graph network without any community assigned, which is represented by the graph with all nodes gray (graph 1). The node A is chosen to start a community (green) and its neighbours' modularity, nodes B and C, are checked as if they belonged to the same community. In this case, node B yields a higher modularity because node C is linked to node D that is not linked to any other green node. The resulting community can be seen in the third graph. The only neighbour of B that does not belong to a community is C. Node C could be part of the green community or might create a new community with node D. The last option would decrease the modularity of the green community since there would be two edges from the green nodes A and B linking to the new community on node C. In the first option, however, where node C is part of the green community, the modularity would increase since it would result in all of the nodes in this community to be linked to each other (graph 4). The next node to assign, node D, would decrease the modularity of the green community since it would break the scenario of all the nodes in the community being connected to each other. In this case, creating a new community with its neighbour (node E) that does not belong to any community would result in two communities with higher modularity than the first case, resulting in the graph 6. All the nodes were finally assigned to a community and the community aggregation step can now be applied. Because only two communities were created, the aggregation has only one option which is to merge both communities into a single community. However, this aggregation would result in a community with a lower modularity than the modularity of the two communities. Therefore, the algorithm stops with the two communities.

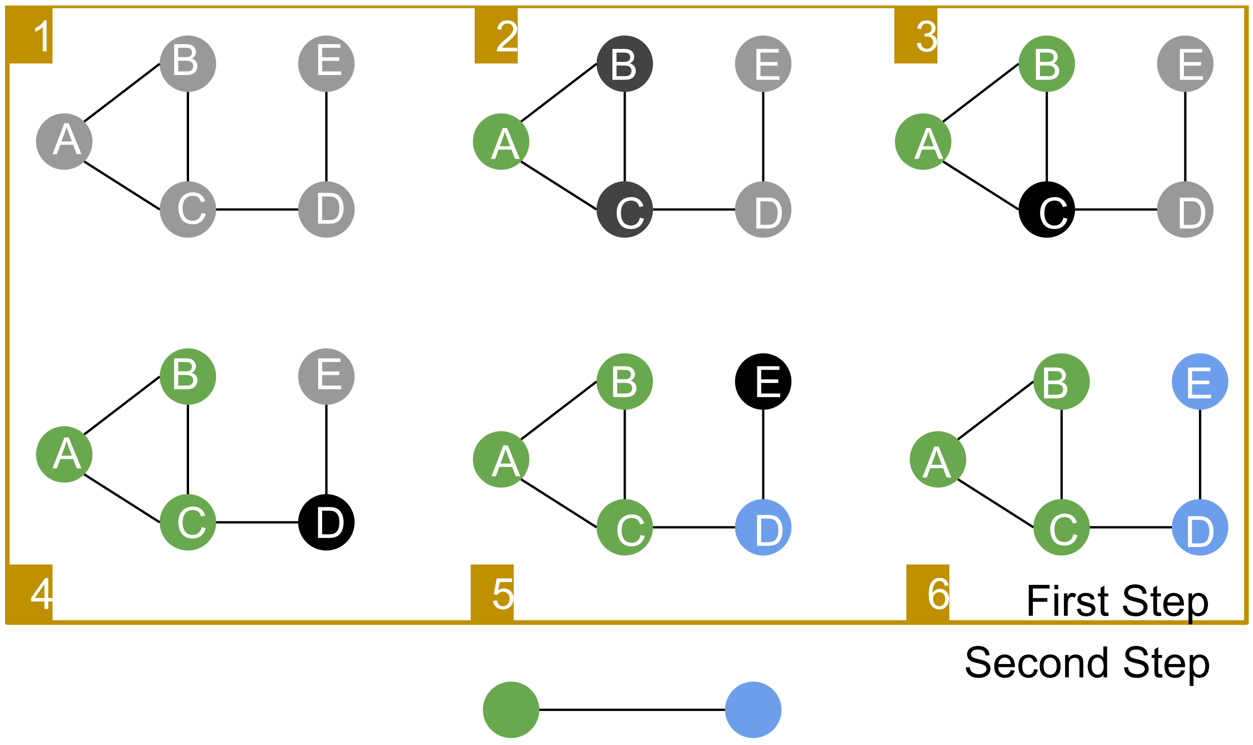


Figure 3.3: Louvain algorithm example. The numbers represent each step and the letters each node. Gray nodes are nodes without a community. Black nodes are neighbour nodes being checked to a new community. Green and blue nodes are nodes belonging to a community.

Chapter 4
Methods

The method described in this chapter – String Analysis of Behavior Log Elements (StABLE) – opens the possibility for designers to analyze with minimal input open world games. StABLE was developed to automatically stratify locations from a given dataset into bins, extract movement behavior, compare them, and provide the behavior similarities. The core of the method is to intelligently aggregate movements and encode them as strings. Natural Language Processing (NLP) techniques can be applied to compare these strings within and between players. Depending on how these strings are built, they can answer different questions. In this thesis, we built strings for visit frequency (location-construction), dwell time, trip duration, and action analysis (dwell-trip-construction and dwell-trip-action-construction). Finally, StABLE returns a similarity matrix which allows to differentiate players for a variety of questions.

The advantage of StABLE over other methods is the minimal input requirement and reasonable run time. From the perspective of a designer, only a dataset preprocessed containing the player ID, locations, and activity is required. While other methods require designers to select features or other non-intuitive parameters, StABLE compares player behaviors based solely on their movements, making possible the analysis of sandbox games due to their feature complexity. We believe that modeling players based on their movements and actions can provide a more general player modeling than the existing approaches, which are primarily based on actions. With the similarity matrix returned by StABLE, designers have the opportunity to analyze player behavior in different levels of information which addresses the problem of information overload commonly found in the literature. The next section describes in detail how StABLE was built and how it can be applied for player behavior analysis.

4.1 StABLE

StABLE uncovers similar behaviors among players for analysis by following the steps depicted in Figure 4.1. *Stratification* groups location data points into bins. In the *Feature Extraction* step, players behavior is encoded into strings. These strings are compared with NLP techniques in the *Comparison* step. At the end of the StABLE process (*Model*), a similarity matrix is returned for behavioral analysis. Each step presents different possible implementations. In the following paragraphs, we explain the concept of each step and how we implemented them.

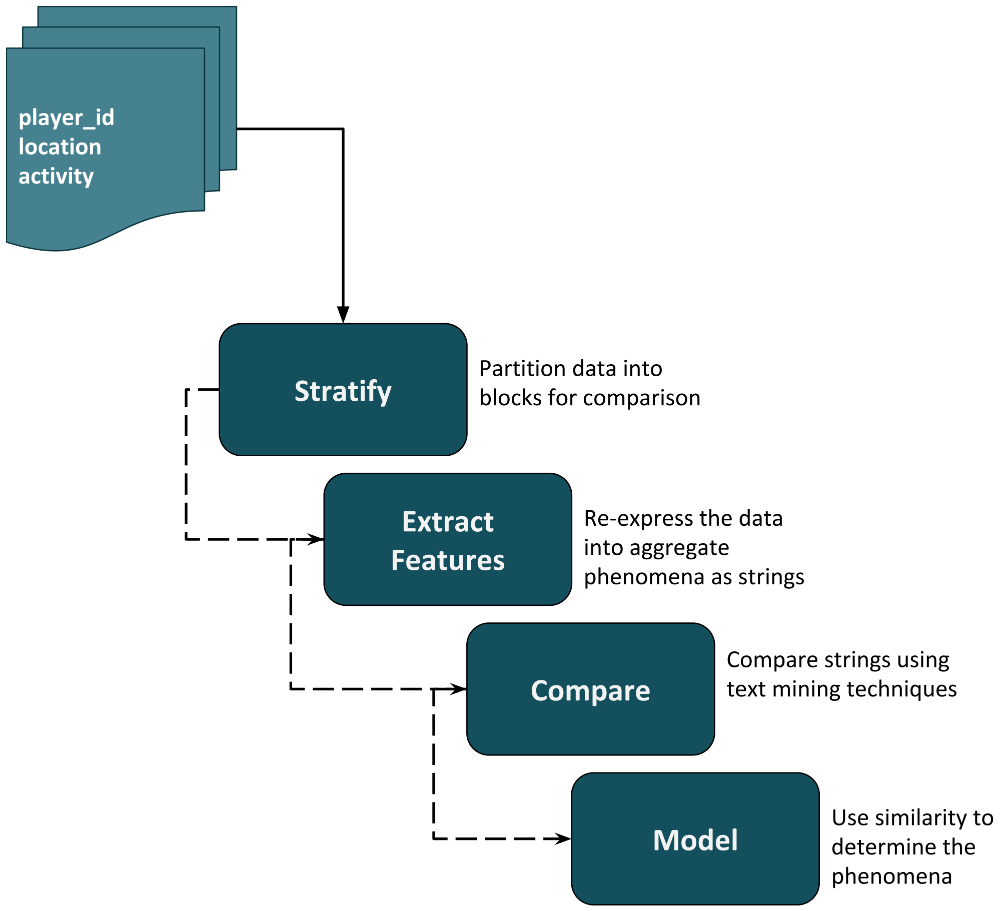

player_id
location
activity
Stratify
Partition data into blocks for comparison
Extract Features
Re-express the data into aggregate phenomena as strings
Compare
Compare strings using text mining techniques
Model
Use similarity to determine the phenomena

Figure 4.1: StABLE process

Data stratification, the first step in StABLE, sorts location data points into bins. The locations are stratified to simplify the detection of dwell time, trip duration, and visited locations. Our implementation requires a dataset with a column named `loc_bin`, containing the locations preprocessed into bins, or a dataset with columns containing locations in four dimensions: pos_x, pos_y, pos_z, and world, because many video games have different worlds or levels as the game advances. If the `loc_bin` column was provided instead of the locations, StABLE will not bin them, giving the option of designers to stratify the data as they wish. In this thesis, we implemented StABLE to be able to stratify the locations into a grid by concatenating the locations of each position x, y, and z plus the level/world. For example, if the player was in world 0 and at the location $(217, 914, 493)$, the final cell label would be the string `0217914493`. The cells can be any size that makes sense to the designer as a parameter to StABLE. By default, it builds the cells as the smallest unit in the game, resulting in each location being its own cell, as shown in the simplified 2D example in Figure 4.2. In this example, Pacman visited cells 11, 21, and 32. If the designer opts for a cell size of 2 units, all the locations that are apart from 1 unit would be located in the same cell. The resulting cells are shown in Figure 4.3. As a result, Pacman visited cells 11, and 21. The relocation of the data points into new cells happens according to:

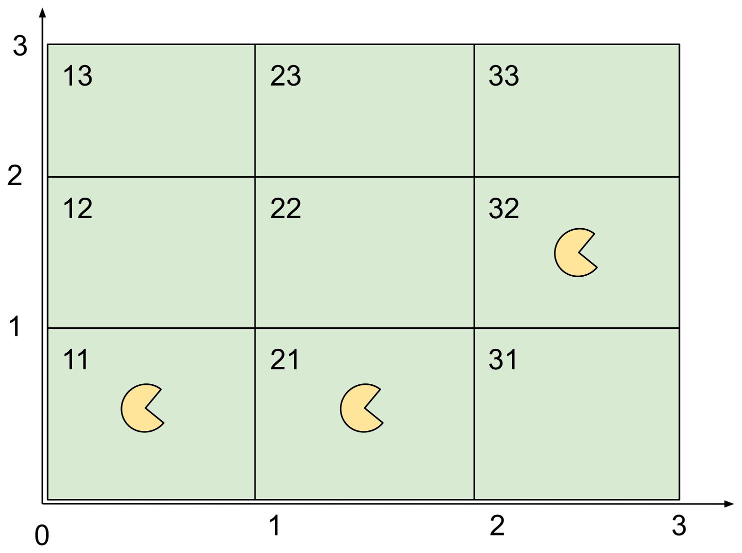


Figure 4.2: Location data points from game log are stratified as grids in StABLE. This figure illustrates a grid with cells of 1x1.

$$cell_x = ceil\left(\frac{x - origin_x}{cell\_size}\right) \tag{4.1}$$

$$cell_y = ceil\left(\frac{y - origin_y}{cell\_size}\right) \tag{4.2}$$

$$cell_z = ceil\left(\frac{z - origin_z}{cell\_size}\right) \tag{4.3}$$

$$loc\_bin = world + cell_x + cell_y + cell_z \tag{4.4}$$

where x, y, and z are the positions in their respective axis; $origin_x$, $origin_y$, and $origin_z$ are the start point in their axis; $cell\_size$ is the size of each cell which results in a grid of $cell\_size \times cell\_size$; and "+" is the concatenation of the cells into the string `loc_bin`. The `ceil` function rounds the fraction to the closest number higher than the result of the fraction. For example, to recalculate the cell 32 in Figure 4.2 to 21 in Figure 4.3, the Equation 4.1 and Equation 4.2 would be used, where $x = 2.5$, $origin_x = 0$, $y = 1.5$, $origin_y = 0$, and $cell\_size = 2$. As a result, $cell_x = ceil(1.25) = 2$ and $cell_y = ceil(0.75) = 1$. The last step (Equation 4.4) is to concatenate all the cell results plus the world. If these cells were located in world 0, $loc\_bin = 021$.

The feature extraction step is the step in StABLE where player behavior is encoded into strings. The core insight underlying the StABLE method is that player behaviors can be sensibly rendered as strings. As player behavior unfolds through time, a record of symbols representing that behavior can be generated. The nature of the symbols selected determines the type of analysis conducted. Changing these parameters

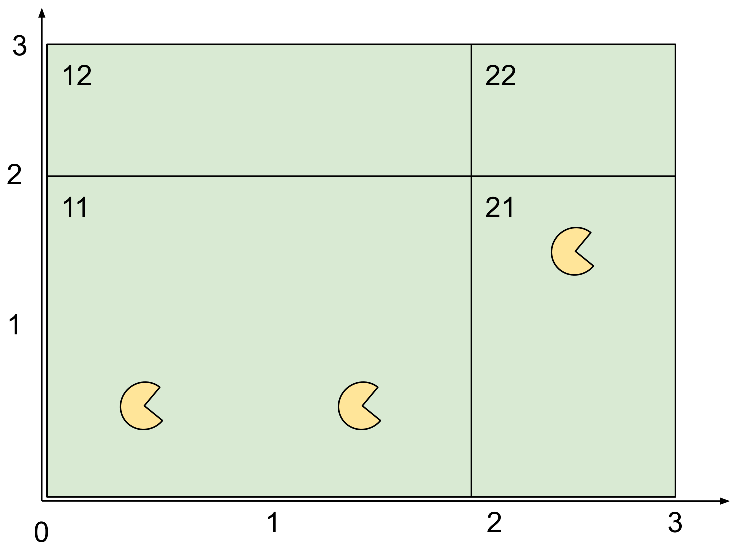


Figure 4.3: Location data points from game log are stratified as grids in StABLE. This figure illustrates a grid with cells of 2x2.

changes the operationalization of the comparison. For example, comparing two strings based only in the location bins, the comparison will answer the question "*Does player A follow the same trajectories as player B?*". Players who followed similar trajectories through the world would have similar strings, regardless of when they followed the path. If we added the timestamp to the trajectory record as individual words `timestamp loc_bin`, and rendered this as a sequence `timestamp1 loc_bin1 timestamp2 loc_bin2...` the comparison would prefer co-located play, that is players who were at the same place at the same time would be more similar than players who visited the same places at different times. Because timestamp and location are individual words, players who visited the same locations at different times would be more similar to co-located players than players who never visited those location, as the Bag of Words (BoW) would treat location and time as independent words in the histogram. However, if location and time were merged into a single word `timestamp1-loc_bin1 timestamp2-loc_bin2...` co-located play would be required for similarity, as bins in the BoW histogram require both the time and location to be the same. To illustrate the power and flexibility of this technique we address different analyses with different strings based on a game log in Table 4.1. In this thesis, we developed StABLE with three different possible string constructions: location construction, dwell-trip construction, and dwell-trip-action construction. In addition, we also show how these strings are formatted for between-player and within-player analysis.

Location construction We chose to create the location construction to show that StABLE can be used for popular and basic analysis based solely on visited locations such as visited frequency. To probe use of space using StABLE, we encoded the location bins as separate words. Players who visited the same places in the same order more often would be more similar. The `loc_bin` attribute resulting from the previous *Stratification* step is enough to describe this behavior. Given a game log of player, as shown in Table 4.1, the resulting string would be: `0423 0423 0423 0423 0423 -1887 -1886 -1886 -1886`

Table 4.1: A simplified mock-up example of a player log

loc_bin	n_activity
0423	0
0423	0
0423	0
0423	0
0423	0
-1887	1
-1886	1
-1886	2
-1886	3
-1885	1
-1885	0

`-1885 -1885`.

Dwell-trip construction Player movement can be driven by gameplay mechanics as well as the level design. For example, action RPGs involve a series of traversals and encounters, and the movement patterns within those encounters can depend on character class and player playstyle, but might be independent of the actual location in the game world. To render strings which capture movement patterns independent of location we aggregate movement into periods of motion (trips) and stationary periods (dwells), as is common in GIScience [23, 49, 59, 70]. To detect the dwell time and trip duration, we determine if the player changed location. For every sequential record where player location is unchanged, dwell time is incremented. For every sequential record where player location is different than trip duration is incremented, until the player remains at the same location for at least four timestamps, breaking the trip. This aggregation scheme naturally transforms the location records into a sequence of paired dwells and trips.

Dwell time is transformed to `dX`, where `X` is the number of timestamps the player stayed in the same location in sequence, and the trip duration to `tY`, where `Y` is the number of timestamps the player moved from one location bin to another in sequence. Each player's trajectory can be expressed as a string of the form: $\texttt{dX}_1\texttt{, tY}_1\texttt{, dX}_2\texttt{, dY}_2\texttt{....}$ We have opted to separate the trips and dwells as words, meaning that players who had similar trip durations, but different dwell times would be more similar to each other than players whose trip and dwell profiles were completely different.

For example, from Table 4.1, the player started at the bin 0423 and stayed in the same bin for 4 consecutive duty cycles, meaning that they had a dwell time of 4. The player then changed bins twice and stayed in bin -1886 for 2 consecutive timestamps. Finally, the player moved to bin -1885 and stayed for another duty cycle. As a result, the string that represents this behavior with no threshold for movement is `d4 t2 d2 t1 d1`. Because we defined a trip as changing location within four consecutive times, any dwell time of 3 or less consecutive timestamps also belonged to the trip, so the string is `d4 t6`.

Dwell-trip-action construction In the action RPG example, designers might not only be interested in how players move, but what actions they undertake while moving (for example attacking, using a skill, or casting a spell). Actions can be readily incorporated into the purely location or in the dwell-trip construction. We created a string incorporating whether or not an action had occurred during each dwelling or movement period. If at least one `n_activity` has occurred, the word `a1` is added after the dwell time or trip duration, otherwise, `a0` is added. The resulting string is $\texttt{dX}_1\texttt{, aZ}_1\texttt{, tY}_1\texttt{, aZ}_2\texttt{, dX}_2\texttt{, aZ}_3\texttt{, tY}_2\texttt{, aZ}_4\texttt{...}$, where Z can be 0 or 1, indicating if the player was doing some activity (1) or not (0) during the preceding dwell time or trip duration. From Table 4.1, the resulting string is `d4 a0 t6 a1`. For games which do not typically involve many actions, inclusion of a binary activity parameter to the string could increase similarity, as everyone will have similar amounts of non-activity; however, the TF-IDF helps diminish this effect by reducing the importance of the repeated null activity value.

Between-player and Within-player analysis Comparing players with each other can be used to highlight or segment players into clusters of playstyles, but comparing players with themselves can also be useful to describe how gameplay changes with time or through phases (tutorial, main game, end game) or in within player experiments (for example before and after a patch is deployed). To demonstrate the utility of our approach, we illustrate how similarity can be employed to compare players with themselves, and also between groups of players where the comparison is within player but the effect is between players. In the first case, we are interested in the extent of an effect of a change overall, and in the second case, we are interested if the extent of a change differently impacts groups. For the next step, *Feature Comparison*, where we use cosine similarity to calculate the similarities, a list of strings is required so that each string can be compared to each other. In the between-player analysis, the resulting list of strings from one of the string construction is enough. However, to illustrate the concept of a change in a within-player perspective, we differentiate the beginning of the game from the rest for each participant. We performed a simple per-player median split on their play log to obtain a list of two elements to be compared. For example, Table 4.1 has 11 records in total for a player. The first half (from the 1st row to the 5th) which results in a behavior with only dwell time `d4` would be compared to the second half (from the 6th row to the 11th) which results in a trip duration `t6`. With the addition of activity, the first half would be `d4 a0`, and the second would be `t6 a1`.

To compare the constructed strings, the BoW transformation is performed with the specified N-gram number, using the TF-IDF to determine the final bin size. The N-gram selection is linked to the construction of the string, and subsequently how the similarity is interpreted. For constructions like the simple location string, the interpretation of the N-gram is the sequence of movements (visits to A, visits to B, visits to A then B). For the dwell and trip construction, the interpretation is the sequential movement patterns and naturally follows pairs of labels (`dX tY`), which implies that N should increase by twos to capture subsequent motion. Similarly, N should increase by fours in the case of the dwell-trip-action construction. In our implementation, StABLE starts using N with the size of the string construction: $N = 1$ for the location analysis, $N = 2$ for the dwell-time analysis, and $N = 4$ for the dwell-time-action analysis. StABLE then doubles the N value for each analysis and also uses an N-gram ranging from one to the doubled N values.

After constructing the BoW histogram using TF-IDF, the similarity can be calculated between the time series data that has been encoded in the string. We applied the cosine similarity in this thesis for the two different types of comparisons, between and within players. In the within-player analysis, the strings at the beginning and at the end of the same player were compared as explained in Between-player and Within-player analysis. Within-player evaluations could be useful for studying learning or playstyle evolution in deployed games, or for probing differences in player behaviors under controlled conditions in an experiment. In the between player analysis, the strings of all players were compared to each other. Between player comparisons could be used in large experiments, or in cluster analysis in deployed games.

The previous step returns a similarity matrix (sim) for both between and within-player analysis which is the final model of StABLE. Each element in the matrix sim_{ij} is the similarity score between the behaviors of players i and j. In the between analysis, the matrix is a symmetric matrix $P \times P$, where P is the total number of players, with the diagonal populated by ones. In the within-player analysis, the matrix has the shape $1 \times P$. Figure 4.4 shows an example for both matrices with 3 players.

$$\begin{array}{c|ccc|} & \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{1} & 1 & 0.4782 & 0.8347 \\ \mathbf{2} & 0.4782 & 1 & 0.1258 \\ \mathbf{3} & 0.8347 & 0.1258 & 1 \end{array} \qquad \begin{array}{|ccc|} \mathbf{1} & \mathbf{2} & \mathbf{3} \\ 0.9981 & 0.2567 & 0.6712 \end{array}$$

a) b)

Figure 4.4: Example of similarity matrix for between-player (a) and within-player (b) analysis with a total of 3 players. The columns and rows represent players.

Chapter 5
Experimental Setup

We used the Minecraft game log dataset with StABLE to demonstrate its efficacy with endless and open world games. From the perspective of a designer using StABLE, the only information he or she would need to provide are the columns of the dataset representing player's location and activity. If selection and filtering are required, the dataset should be preprocessed before feeding StABLE since it does not offer a preprocessing step as shown in Figure 4.1. We describe how we filtered and selected the columns from a Minecraft game log and how we used the similarity model from StABLE to gain insights of player behavior to address the following research questions:

RQ1 Do players have similar visited frequency behaviors in the game?

RQ2 Do players who explore more of the game present similar visited frequency?

RQ3 Do players change their visited frequency behavior through the game?

RQ4 Can we cluster the players based on the self reported motivation?

RQ5 Do players who explore more of the game present more similar movements and activities than players who not?

RQ6 Do players have similar movements, with and without activity, when comparing themselves at the beginning and end of the game?

5.1 Minecraft

Minecraft is one of the most popular games in history, with more than 100 million active players monthly. Minecraft is a creative and survival game where players pay attention to their health while they build from resources in the world. One of the novelties when the game was released in 2011 was that the Minecraft world is never the same. Every time a player starts a new game, a new world is generated, unless they provide a fixed seed number. The Minecraft world is procedurally generated with voxels, cubes that represent any resource in the world: wood, grass, water. Even animals are made of multiple voxels. A view of a Minecraft world can be seen in Figure 5.1. Minecraft has no end, and no defined goal. Minecraft is a sandbox game, if not the canonical sandbox game. Players can play in the same game for how long they want to without any script, which results in a range of possible player behaviors [14]. Even though their character might die because of monsters, the character is respawned in a place where the player has constructed a bed. Players gain experience levels as they explore the map in the game [14]. Minecraft is composed of three worlds (dimensions): The Overworld, The Nether, and The End. All players started in The Overworld. Players can only access the next world by exploring the current one until they find the items needed to open a portal. To reach The Nether, players have to further explore The Overworld whereas to reach The End, players have to further explore The Nether.

Figure 5.1: Minecraft world is made by boxes called voxels.

5.2 Study Procedure and Apparatus

We recruited 40 participants (7 female) with ages ranging from 18 to 34 ($\mu = 20.97, \sigma = 3.63$). Participants already had a Minecraft account prior to participating in the study. Thirty six of the participants reported playing digital games at least a few times per week. Participants completed informed consent, and demographics questionnaires before playing. Enrollment happened via a website, and under the authorization of our institutional ethics review board. After completing the surveys, participants were added to the server white-list and received an email on how to install Minecraft Forge in their computers and connect to the server. Participants were instructed to play normally without any restriction. Data collection occurred over 2.5 weeks. At the end of the study, participants completed the Situational Motivation Scale (SIMS) [32]. Participants received an honorarium of CAD $25.

We collected the game log using Minecraft Forge [22], a framework to facilitate the interface between Minecraft and player designed mods. We built a custom data logging mod that collects 25 different events from the game in each second, which we refer as a duty cycle. At the end of the experiment, each event in the game log was uploaded to a table in a MySQL database. A Minecraft multiplayer server was installed on a Virtual Machine running Linux (RHEL 6.10) with 2 CPUs and 4 GB of RAM. The server ran Minecraft Forge v13.20.0.2228 on Minecraft v1.11.2 using our mod. Participants accessed the server from their own personal computers, but they had to install the same Minecraft Forge version to avoid conflicts. All players started the game in the same position and played with the same randomly selected world seed (5791568963867681365).

5.3 Preprocessing

Preprocessing the data is the only step where the designer's input is explicitly required in the StABLE analysis pipeline. Preprocessing is important in dealing with the redundancy, incompleteness, and inconsistency of data that is commonly found in real databases. Appropriate columns must be selected, erroneous data points removed, data points discretized, and datasets aggregated and/or merged. These preprocessing steps are known as data selection, data filtering, data aggregation and data merging, respectively. They are commonly applied before any data analysis. Data selection reduces the dataset dimensionality by using only relevant attributes as many of them are not interesting for analysis. Data filtering is the process of finding data points that are discrepant (outliers). Their presence ends up negatively interfering in the data analysis. Data stratification groups data points into bins for better discretization. Data aggregation and merging are part of the data transformation process that aims to prepare the data for model implementation and to make it easier to understand results [33]. We applied data selection, filtering, stratification, aggregation, and merging over the Minecraft dataset.

We first selected the necessary data. StABLE requires a dataset containing player locations and activity. The game log should be in a dataset format. In total, we selected four tables to preprocess the data. Italicized columns were employed in our analysis.

PlayerTick provides *time, player ID, position x, position y position z, current dimension, experience level*, camera yaw, camera pitch, inventory, momentum x, momentum y, momentum z, current dimension, health, hunger, food saturation, experience, total experience, light level of current block, and local difficulty of area. This dataset has 1.6 million records.

PlayerInteract provides data when the player is about to interact with a block. It has 5.1 million records of *time, player ID*, position x, position y, position z, block, block position x, block position y, block position z, and item in hand. The other columns were not selected to our final dataset.

PlayerLoggedIn provides data when a player connects to the server. We recorded 466 records of time and player ID. We used this table for filtering purpose only.

PlayerLoggedOut provides data when a player disconnects from the server. We recorded 448 records of time and player ID. We used this column for filtering purpose only.

For the first three questions (RQ1, RQ2, and RQ3), we did not select the position y (altitude) because we used heatmaps to evaluate these questions, which only account for longitude (position x) and latitude (position z) points. As mentioned before, Minecraft has three worlds: The Overworld (dimension 0), The Nether (dimension -1), and The End (dimension 1). Because few players visited dimensions -1 and 1, as shown in Figure 5.2, we only included locations from dimension 0.

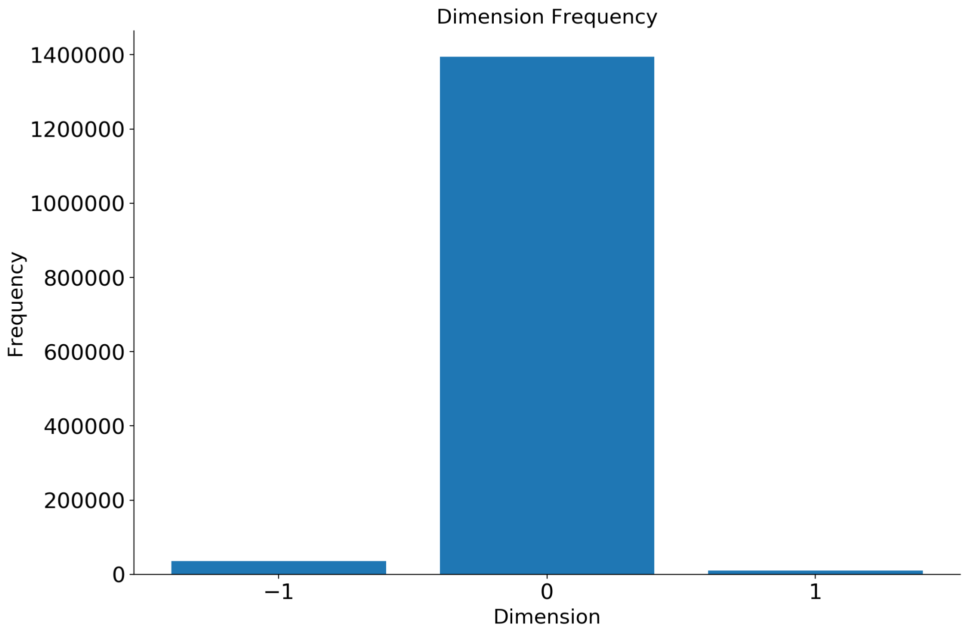


Figure 5.2: Minecraft has three different worlds (dimensions): The Overworld (0), The Nether (-1), and The End (1). This figure illustrates the dimension distribution. Dimensions -1 and 1 were rarely visited and filtered out to answer questions RQ1, RQ2, and RQ3.

Besides the previous tables, we also selected the scores for each of the SIMS assessment: intrinsic motivation, identified regulation, external regulation, and amotivation. We used SIMS scores to answer RQ4. The player distribution for each motivation (intrinsic motivation, identified regulation, external regulation, and amotivation) can be seen in Figure 5.3. The blue bars represent the number of players who had a score equal or greater than 50% (strong), whereas the orange bars indicate players who had a score less than 50% (weak). Intrinsic motivation showed a strong score for all participants meaning that playing Minecraft by itself is an enjoyable activity. Another strong score among players was from identified regulation, where players were motivated to play Minecraft as a means to an end, possibly for helping the experiment. Although players received an honorarum of $25, this did not seen to impact play motivation as external regulation scores were weak. The low amotivation scores demostrate that most of the players had an intrinsic or extrinsic motivation for playing Minecraft.

After selecting the necessary data, we filtered the dataset to remove unwanted data. Because we were not policing on how players were playing Minecraft, there was a chance of some of them used bots to

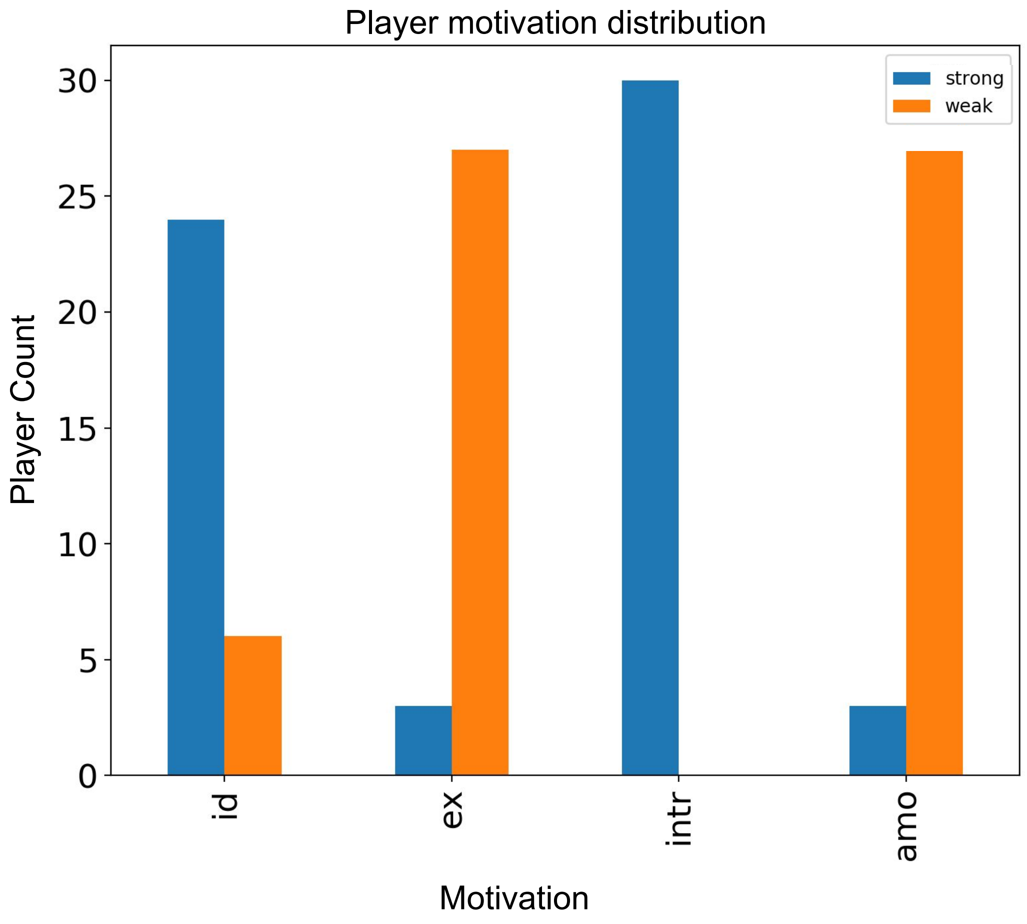


Figure 5.3: Players answered the Situational Motivation Scale (SIMS) questionnaires. This figure illustrates the player motivation distribution. Identified regulation: id, external regulation: ex, intrinsic motivation: intr, and amotivation: amo.

advance in the game. With *PlayerLoggedIn* and *PlayerLoggedOut* tables, we could calculate how long players played per session (period of being logged in) to possibly eliminate these bots. *PlayerLoggedIn* shows more record (466) than *PlayerLoggedOut* (448). We suspect this discrepancy happened because the server crashed at the beginning of the study giving no opportunity for players to log out properly. Therefore, for each *PlayerLoggedOut* data point, we subtracted its time to the closest time before in *PlayerLoggedIn* with the same player ID. We then took the average hour per session. Figure 5.4 shows the distribution of hours played on average. Most of the players played for less than 8 hours, only two players player more than that. We filtered *PlayerTick* dataset to eliminate these two players who played more than 8 hours per session on average, as we suspected bots were playing.

Based on the distribution of dwell times in Figure 5.5, most of the players showed interaction with the game for less than 33 minutes (2000 seconds), with a few interacting for more than 1.5 hours (6000 seconds) in sequence. If players had dwell times in excess of 1.5 hours, the corresponding data was removed because we suspected that the game had been left open. Notice that while in the bots filtering we removed all the

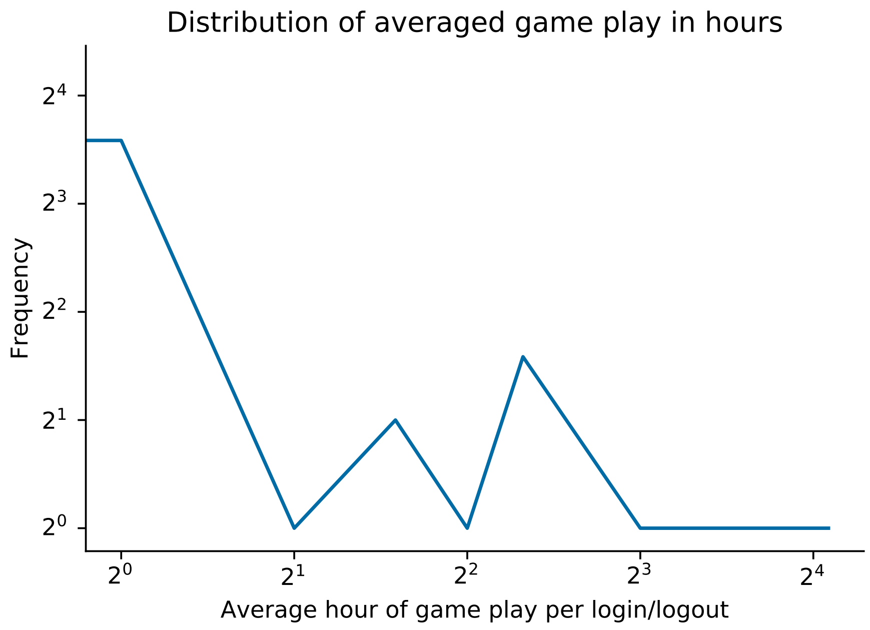


Figure 5.4: Distribution of play hours where players who played for more than 8 hours were considered outliers as we suspected bots were playing instead.

player's data, i.e., the player was discarded from the study, in the dwell filtering, we only removed the data points in *PlayerTick* that were collected for more than 1.5 hours in sequence. The other data points for the same player remained in the dataset.

Finally, because we verify the resulting player clustering based on their motivation, only the players in *PlayerTick* who were also present in the SIMS dataset were selected. At the end of the filtering, *PlayerTick* had a total of 30 players with 1.4 million data points.

We also stratified the data since some of these questions classified players into advanced (who gained a lot of experience) and beginner. As mentioned before, all players started in The Overworld. Players can only access the next world by exploring the current one until they find the items needed to open a portal. To reach The Nether, players have to further explore The Overworld whereas to reach The End, players have to further explore The Nether. We established an approximation for experience where players with 21 *xp_level* or more by the end of the study were labelled as advanced because that was the lowest level of any player who reached the Nether. As a result, we had 18 advanced players and 12 beginner players.

The final step in our data preprocessing was to combine the location data and the activity data by aggregating and merging them. The datasets *PlayerInteract* and *PlayerTick* were parsed for this purpose. Because we may have multiple records for each duty cycle, we aggregated both *PlayerInteract* and *PlayerTick* to the level of a duty cycle for each player. When aggregating, we took the mean location and the total number of activities. With single data points for each duty cycle, we were able to merge both datasets based on their player ID and duty cycle. The resulting dataset G had a total of 30 participants and around 1.1 million

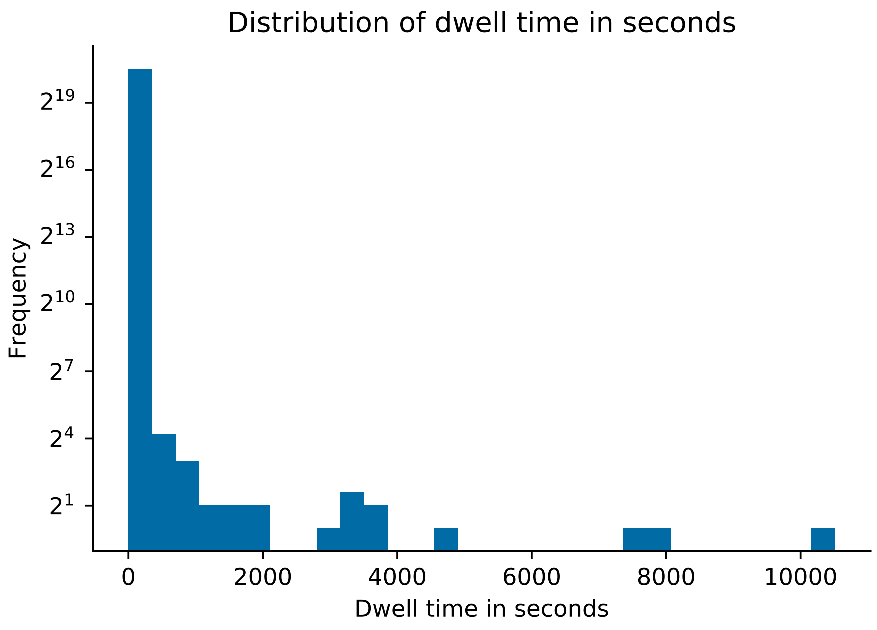


Figure 5.5: Distribution of dwell times. Few players dwelled for more than 6000 seconds. Dwells over 6000 seconds were removed from the data where we suspected players left the game open.

records with 7 columns. The dataset G was used as input for StABLE, and included a column identifying each player (*name* in our dataset), a list of columns for location (*pos_x, pos_y, pos_z*), a list of columns for additional location information, in our case *dimen*, and a column for the number of activities in a duty cycle (*n_activity*). The data points in G must be in order to preserve the time series.

Table 5.1: Final schema of game log G dataset with around 1.1 million records

Attribute	Description
name	player's ID
dc	duty cycle number
dimen	dimension (world) where the player is located
pos_x	coordinate for position in x
pos_y	coordinate for position in y
pos_z	coordinate for position in z
n_activity	number of activities

In addition, the SIMS score and expertise information were merged in a dataset P that is used to interpret the model from StABLE. The final P dataset containing the player ID, expertise, and SIMS scores is shown in (Table 5.2), with 30 records and 6 attributes. With this dataset and the similarity models, six research questions posed by this thesis can be addressed.

Table 5.2: Final schema of players P dataset with 30 records

Attribute	Description
name	player's ID
in_motivation	SIMS intrinsic motivation score based on [32]
id_regulation	SIMS identified regulation score based on [32]
ex_regulation	SIMS external regulation score based on [32]
amotivation	SIMS amotivation score based on [32]
xp	experience classification: *Advanced* or *beginner*

5.4 Using StABLE

With the similarity matrix provided by StABLE, we could answer the six research questions. All the questions were run with the default location stratification with cells of size of 1, which is the smallest unit in Minecraft. The only parameter given to StABLE was the dataset G. The first three research questions are based on visit frequency behavior analysis. We answered the first question (*Do players have similar visited frequency behaviors in the game?*) with the similarity matrix by retrieving the highest similarity score. If the highest score is below 50%, we can conclude that players tend to visit unique places. We also plotted all the paths on a heatmap, highlighting the most similar paths to see where players mostly visited. The second question (*Do players who explore more of the game present similar visited frequency?*) requires the similarities between players. To plot all the similarities in a concise manner, we represented the similarity matrix as a box plot. The higher the boxes, the more similar is a player to others. Because each player has a distribution of similarities with all other players, we took the distribution of the mean similarity of each player, and tested whether there were differences between beginner and advanced players in mean behavioral similarity. The third question (*Do players change their visited frequency behavior through the game?*) is a within-player analysis, and we represented the similarity matrix as a bar plot. We applied a statistical test to check if advanced players were significantly more stable (higher scores) than beginners.

We answered the fourth question (*Can we cluster the players based on the self reported motivation?*) with the dwell-trip-construction and the Louvain algorithm. We used Louvain to naturally group the data into clusters based on the graph network structure only. We ran the Louvain method with predefined community labels to fix the number of resulting communities to avoid clustering into insignificantly small communities. For each node, we assigned its weight as being the sum of the similarities incident to that node divided by the degree of that node (weight average). We divided the graph into two communities based on the median of weights, where nodes with equal or more weight average than the median were given an initial label in one community and nodes with lower weight average than the median the other community. Louvain adjusted these initial communities to maximize the modularity. The network graph shows how players are connected based on their behaviors. We established a similarity score as a threshold to obtain a more concise network representation that is easier to read. We then applied statistical tests for each of the SIMS motivation scores to verify if the groups were significant different, demonstrating that StABLE can also be used to illustrate underlying drivers of differences in behavior.

For the fifth question (*Do players who explore more of the game present more similar movements and activities thanplayers who did not?*), we applied statistical tests to differentiate the similarity box plot distributions between advanced and beginner players with the dwell-trip-action construction. We applied a statistical test between advanced and beginner players over their similarity means to check if advanced players show a more similar behavior than beginners. A network graph was employed to visualize player similarity relationships.

Finally, the sixth question (*Do players have similar movements, with and without activity, when compared to themselvesat the beginning and end of the game?*) could be answered with the dwell-trip and dwell-trip-action constructions. These constructions were used to analyze playstyle stability within players. Players were grouped into advanced and beginner players and we applied statistical tests to verify if advanced players had more stable playstyles (more similar to themselves) than beginner players from the similarity matrix. We visualized the results with the similarity bar plots. This analysis demonstrates that StABLE can not only provide more complex insight between players, but also within players.

5.5 Experiment Environment

We analyzed the similarities within and between players using the StABLE technique. The Mann-Whitney U statistical test was applied because similarity based partitioning does not necessarily return normal distributions. We reported the results with a significant effect when $p < .05$. We used the library *stats* under *scipy* (v 0.19.1) in Python for the statistical tests. A Mac-Book Pro 2.7 GHz Intel Core i7 with 16 GB 1600 MHz DDR3 RAM was used to perform the analysis. The run time and memory use of StABLE for each research question did not exceed 2 minutes or 200MB respectively, not including the time and memory required to download and store the original data.

Chapter 6

Results

We evaluated the results from StABLE for six research questions (Table 6.1). While we are not design experts, and do not claim to have made a strong contribution to game studies, we intend to demonstrate the flexibility of StABLE as a method to answer many different questions around gameplay, especially for an open and endless game like Minecraft. Of course, more complex questions could be addressed with StABLE if one of the string constructions described in the previous Chapter 4 is representative of the phenomena.

Table 6.1: Research questions applied in this study with StABLE

RQ1:	Do players have similar visited frequency behaviors in the game?
RQ2:	Do players who explore more of the game present similar visited frequency?
RQ3:	Do players change their visited frequency behavior through the game?
RQ4:	Can we cluster the players based on the self reported motivation?
RQ5:	Do players who explore more of the game present more similar movements and activities than players who did not?
RQ6:	Do players have similar movements, with and without activity, when compared to themselves at the beginning and end of the game?

The results for RQ1, RQ2, and RQ3 are described in Section 6.1. RQ4 is addressed in Section 6.2. In Section 6.3 shows the results for RQ5. Finally, RQ6 results can be seen in Section 6.4.

6.1 Location Analysis

We only considered locations in the axis-x and z of the Minecraft world, which correspond to the virtual longitude and latitude coordinates. Because few players visited dimensions -1 and 1, we only included locations from dimension 0. The location string construction was used in this analysis with $N = 1$ which directly reproduces standard aggregate location distributions. Figure 6.1 shows the visited frequency similarity distributions ranging from 0% to 1%. Only 6 of the pairs were bigger than 10%. We applied the Mann-Whitney U test where the independent variable was the player experience with 2 levels (advanced and beginner), and the dependent variable was the mean similarity distribution, which did not find a significant difference between advanced and beginner players ($p > .05$). In Figure 6.2, we can see all the visited locations in the game log and the most similar visited frequency between two players (players 4 and 11), with a similarity score of 24.36%. As expected for a large open-world game such as Minecraft, trajectories through the game were substantially different for each player. In contrast we would expect the location similarity of a tightly constrained game like an on-rails shooter to approach 100% for all players as there is no freedom of movement. This result demonstrates that StABLE can be used to directly probe visited locations, but that this is a less interesting question for games like Minecraft. Stratifying location into cells makes it possible to analyze players behavior in different resolution, a common approach in GPS and GIS research [49, 59]. In this thesis, we used the default cell size to capture the highest movement resolution in Minecraft. After stratifying the locations, we were left with approximately 1.1 million records and 30 participants, but with only 4 attributes (Table 5.1).

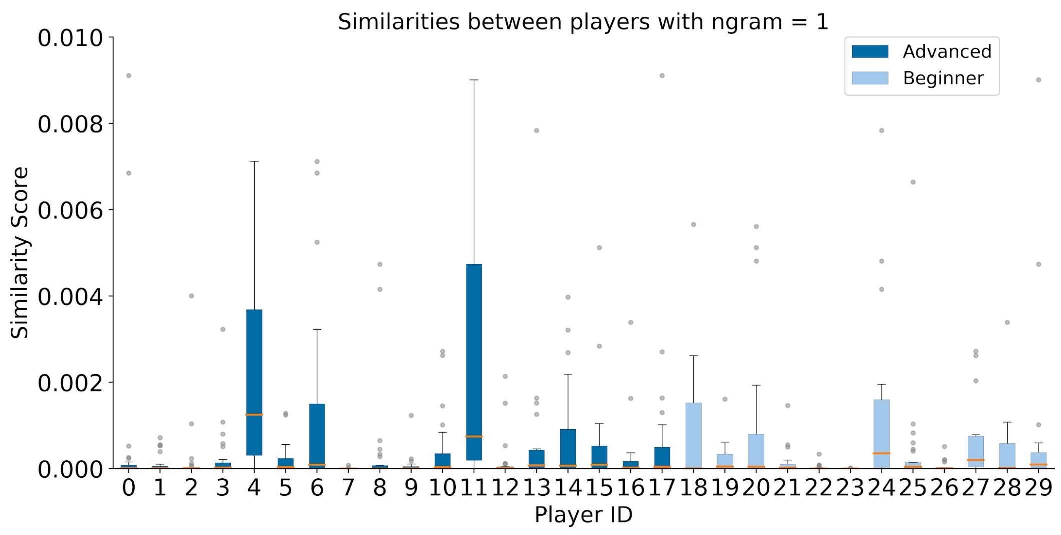


Figure 6.1: The lower and upper quartile values of the similarities (box), the median (line), and the range of the similarities (whiskers) are illustrated, using the location construction. Players are in descending order of experience level.

Figure 6.3 shows within-player location analysis. Players are more similar to themselves through time than with each other. We applied the Mann-Whitney U test where the independent variable was player experience with 2 levels (advanced and beginner), and the dependent variable was the similarity scores, which showed that the advanced and beginner were distinguishable to a level of significance ($p < .05$). Advanced players were more self-similar than beginner ones because they know how to play Minecraft.

6.2 Player similarity and motivation

Applying the dwell-trip construction, it is possible to compare player movement profiles over different values of N. Figure 6.4 shows the similarity distributions between players with different N-grams, and the total number of words per player. We used all N-grams from 1 to 4 (Figure 6.4.a). We can see that most of the players showed similarities over 50%. We then plotted the similarities with an N-gram of exactly 2 (6.4.b),

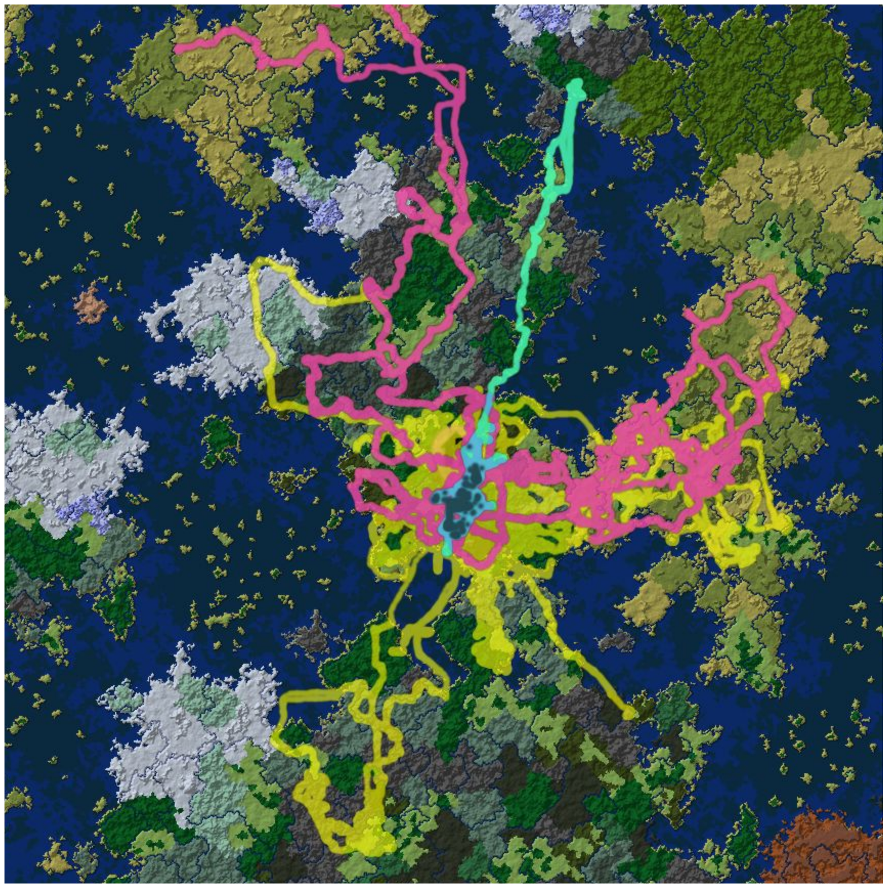


Figure 6.2: Heat map with locations from all players (yellow locations). Green path represents player 4, and the pink locations represent player 11, where the dark blue portions are the common locations between them, with a similarity of 24.36%.

which corresponds to a complete movement (dwell time plus trip duration), and an N-gram of exactly 4 (6.4.c). The latter figure showed lower similarity distributions, indicating that this N-gram is likely too constrained for this analysis. There appear to be canonical dwell-trip rhythms in Minecraft, but these rhythms do not typically chain together. Note that StABLE is not influenced by the length of the strings (Figure 6.5). For example, even though player 0 had a string with a more than double then next largest, their similarity was equivalent to others (3, 4, 5 and 17).

We used the Louvain method for community detection to cluster players based on their similarities with a bigram, $N = 2$, (Figure 6.4.b), to determine if the groups created from play similarity data encoded information about play motivation. Binary graphs were created by applying a threshold of similarity ($T = 0.7$). Similarity values above the threshold created an edge between the players in the graph. Players with no edges to any other nodes were discarded. The start node for the Louvain method was the node with highest weight average in the network, which was player 0 who had a degree of 19 and a similarity average of 84.25%. The resulting network graph can be seen in Figure 6.6. The closer the nodes, the more similar they are. We then mapped the SIMS scores (construct of intrinsic motivation, identified regulation, external regulation,

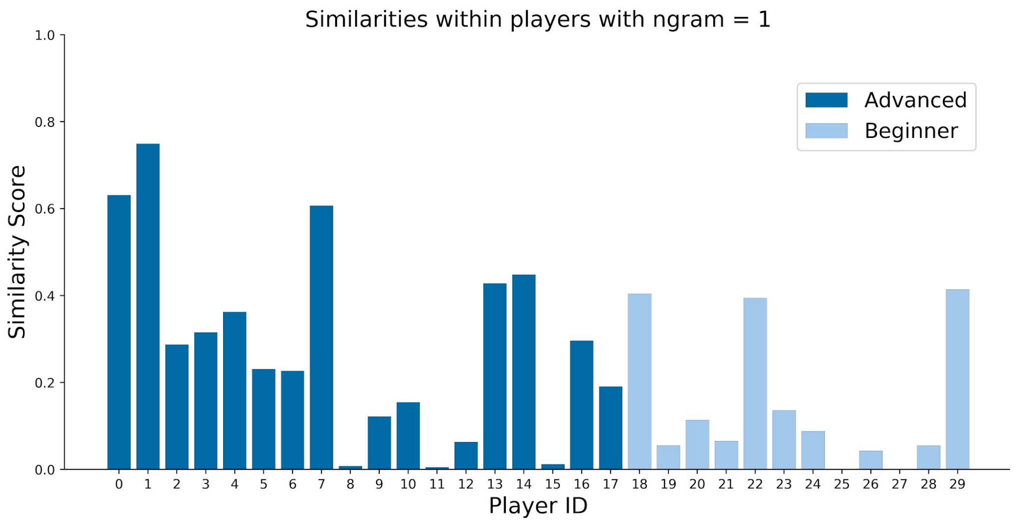


Figure 6.3: Similarity scores within players using the location string construction. Players are in descending order of experience level.

and amotivation) to the groups identified through Louvain clustering and applied the Mann-Whitney U test. Identified regulation ($p < .05$), and amotivation ($p < .01$) showed a significant difference across the player clusters.

6.3 Player similarity with experience

Advanced and beginner players may present different behaviors based on how they play. We investigated this difference by using dwell-trip-action construction. Figure 6.7 shows the similarity distributions among players and the number of words for each player. Figure 6.7.a shows the similarities with a N-gram ranging from 1 to 8. Most of the advanced players had mean similarities over 80%, visibly higher than beginner distributions. We applied the Mann-Whitney U test where the independent variable was the player experience with 2 levels (advanced and beginner), and the dependent variable mean similarity distribution. Advanced and beginner players had significantly different mean similarity scores ($p < .01$). Analyzing with a N-gram of 4 (Figure 6.7.b), advanced players seem to present higher similarity than with less experienced players. Using the same independent and dependent variable, Mann-Whitney U returned that advanced players are different than beginner players ($p < .01$). An N-gram of 8 (Figure 6.7.c) was characterized by small similarities but still showed a significant difference between classes ($p < .01$). We can see in Figure 6.8 that the number of words had a limited influence on similarity scores as expected.

Figure 6.9 shows the network graph for players who had a similarity of greater than 0.5 using an N-gram of 4, with connection to at least one other player. The shorter the edges between the players, the more similar they are. Advanced players were similar to each other, but beginner players were more likely to show similar behaviors to advanced players than to the other beginner players. We can also depict from the graph that the most experience player (player 0, with an experience level of 68) as well as player 4 (experience level of 35) presented the highest degree in the graph, both with 18 edges. On the other hand, the lowest similarity were between players 10 and 23, with a score close to 50%. Almost all the advanced players appeared in the graph, except by players 8 and 9, who showed a low similarity distributions in Figure 6.7.b, compared to other advanced players.

6.4 Playstyle stability

The dwell-trip construction was used to plot Figures 6.10.a-c but was partitioned per player with a median split, and each player was only compared to themselves. In Figure 6.10.a with the N-gram ranging from 1 to 4, we can see that all the advanced players showed similarities over 60%, with most of them over 80%. On the other hand, most of the beginner players showed a similarity closed to 60%. Mann-Whitney U test showed that advanced and beginner players had different sensitivity to playstyle changes ($p < .001$), with player experience as the independent variable (2 levels) and similarity scores as the dependent variable. Experienced players had more stable (more self-similar) movement patterns than less experienced players. With an N-gram of 2, (Figure 6.10.b) 83% of the advanced players had a similar trip behavior over 50% while most of the beginner players showed similarities below it. With the same independent and dependent variables, a Mann-Whitney U test showed a significant difference between advanced and beginner players ($p < .001$). Doubling the N-gram to 4 (Figure 6.10.c), the similarities were small but still showed a significant effect between groups ($p < .05$).

When adding activity, the pattern remains (Figures 6.11). If we set a minimum N-gram of 1 and a maximum N-gram of 8 (Figure 6.11.a), similarities for advanced players (except for player 8) were close to 100%. Using the same independent and dependent variables, a Mann Whitney-U test showed that the self-similarity of the two groups were significantly different ($p < .001$). In (Figure 6.11.b), with a N-gram of 2, the Mann-Whitney U test showed that the self-similarity of advanced and beginner players were significantly different ($p < .001$). With an N-gram of exactly 8 (Figure 6.11.c), while presenting low similarities, advanced and beginner players showed different self-similarities ($p < .05$).

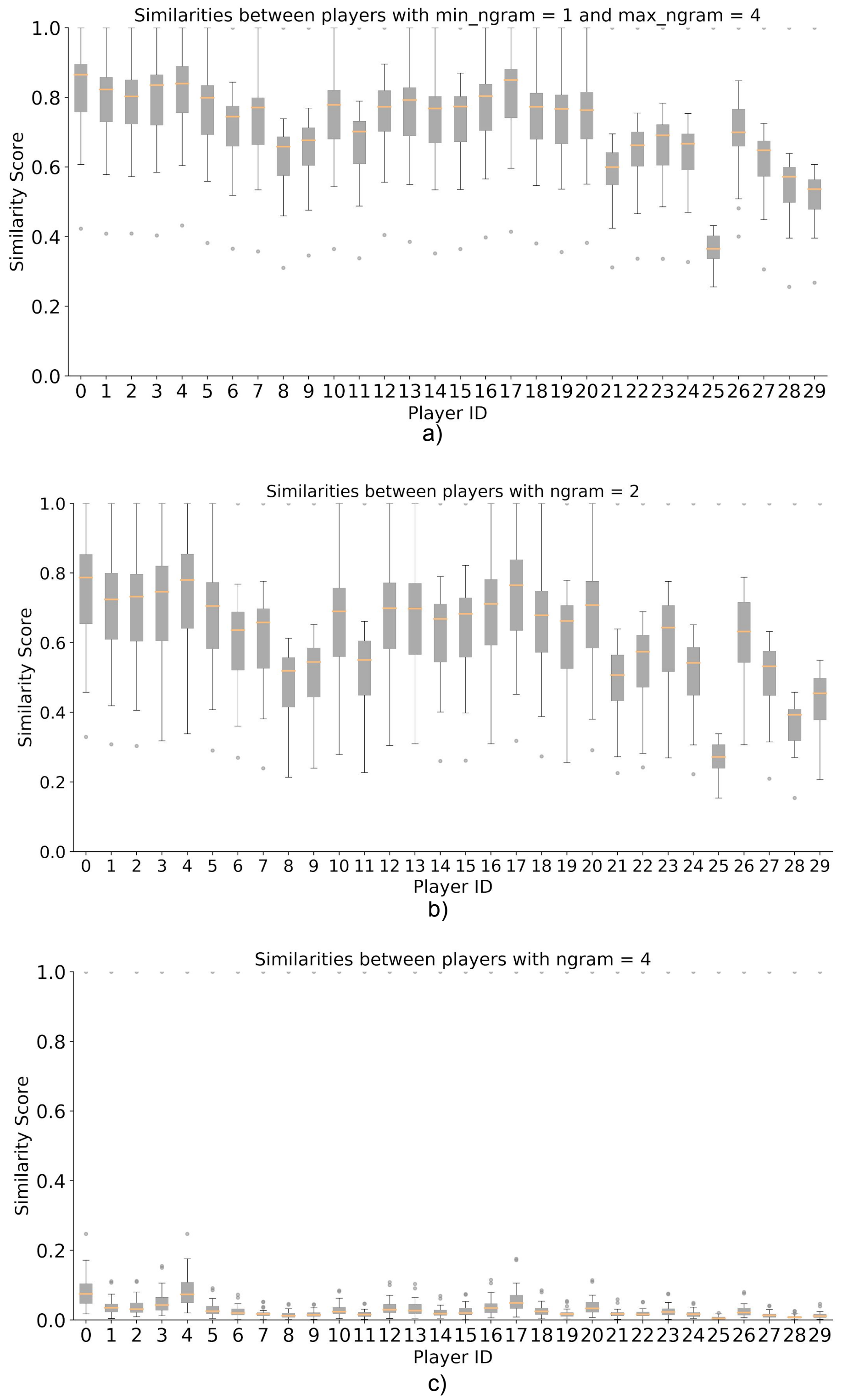


Figure 6.4: The lower and upper quartile values of the similarities (box), the median (line), and the range of the similarities (whiskers) are illustrated with different N-grams (a to c), using the dwell-trip construction. Players are in descending order of experience level.

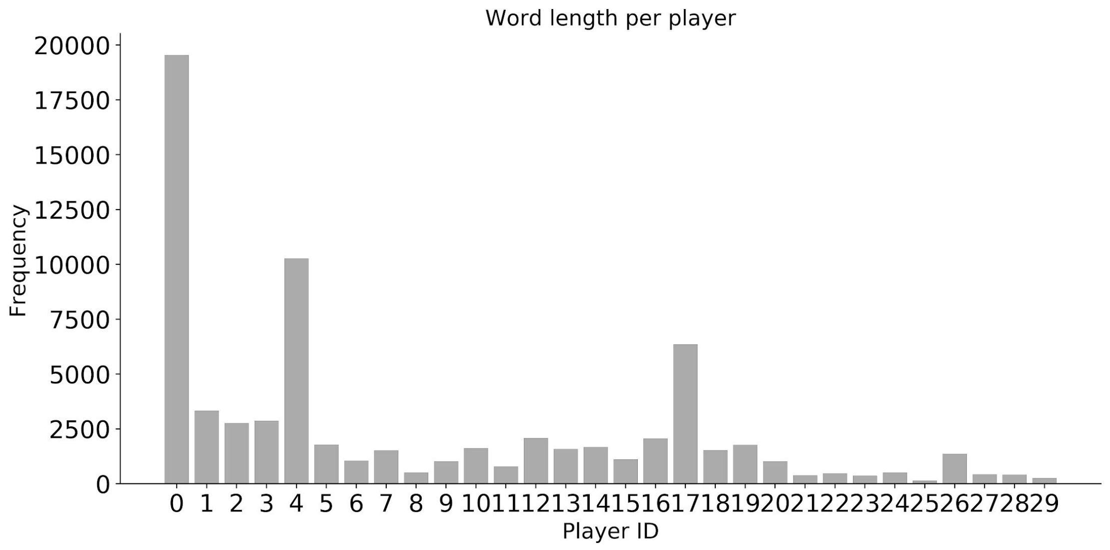


Figure 6.5: Total number of words per player. Players are in descending order of experience level.

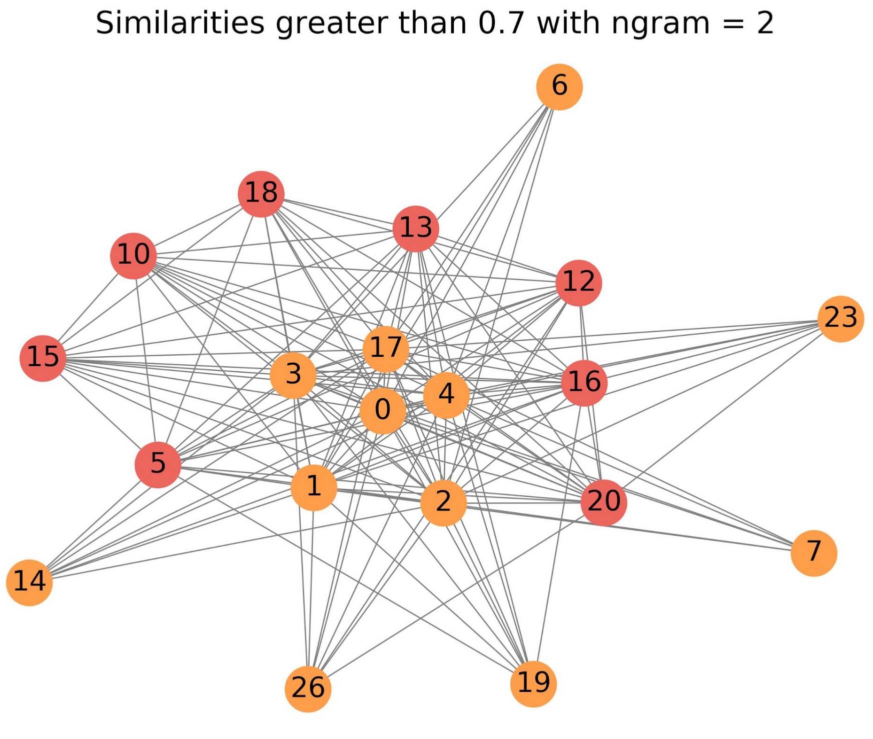
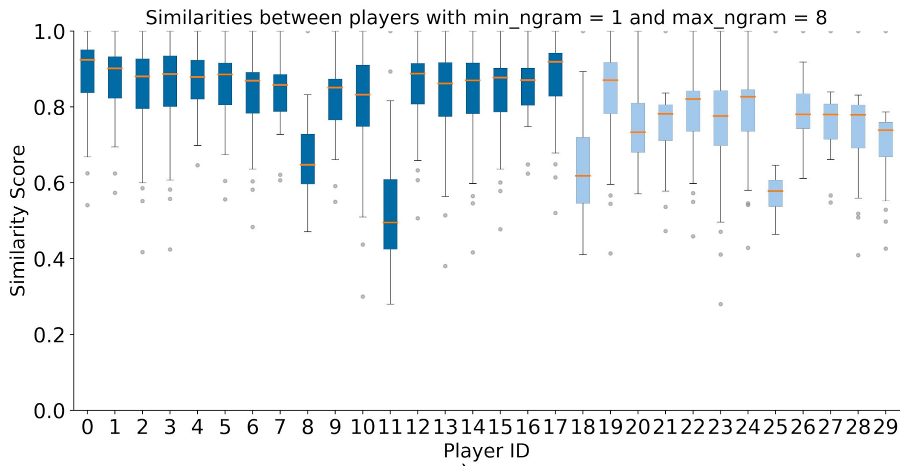
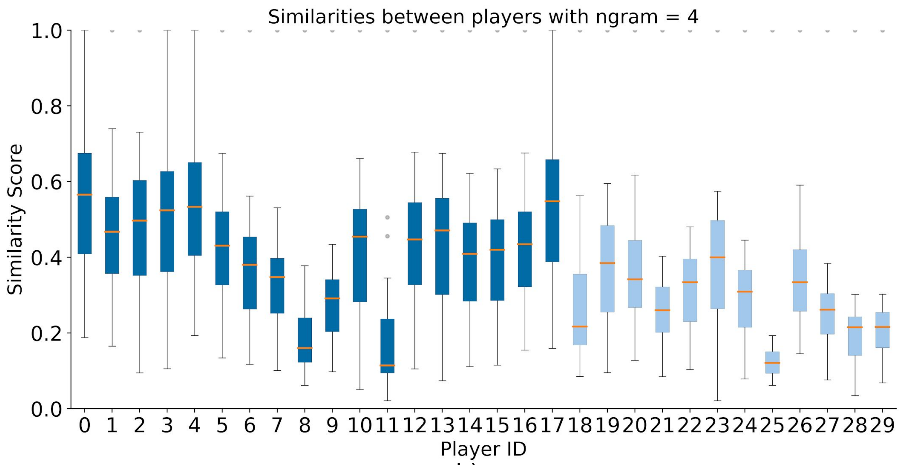


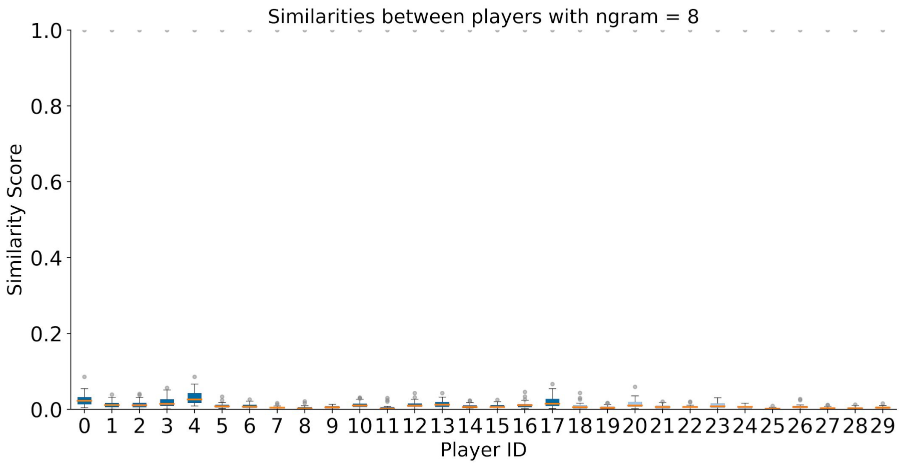
Figure 6.6: Network graph of players who are over than 70% similars with the dwell-trip construction, where each color shows a community by Louvain method.



a)



b)



c)

Figure 6.7: The lower and upper quartile values of the similarities (box), the median (line), and the range of the similarities (whiskers) are illustrated with different N-grams (a to c), using the dwell-trip-action construction. Players are in descending order of experience level.

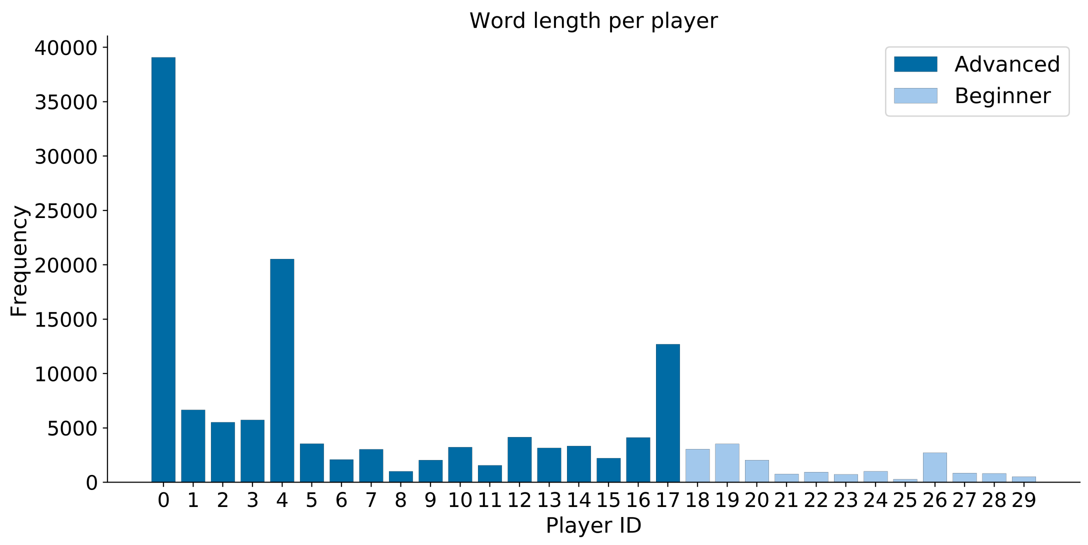


Figure 6.8: Total number of words per player. Players are in descending order of experience level.

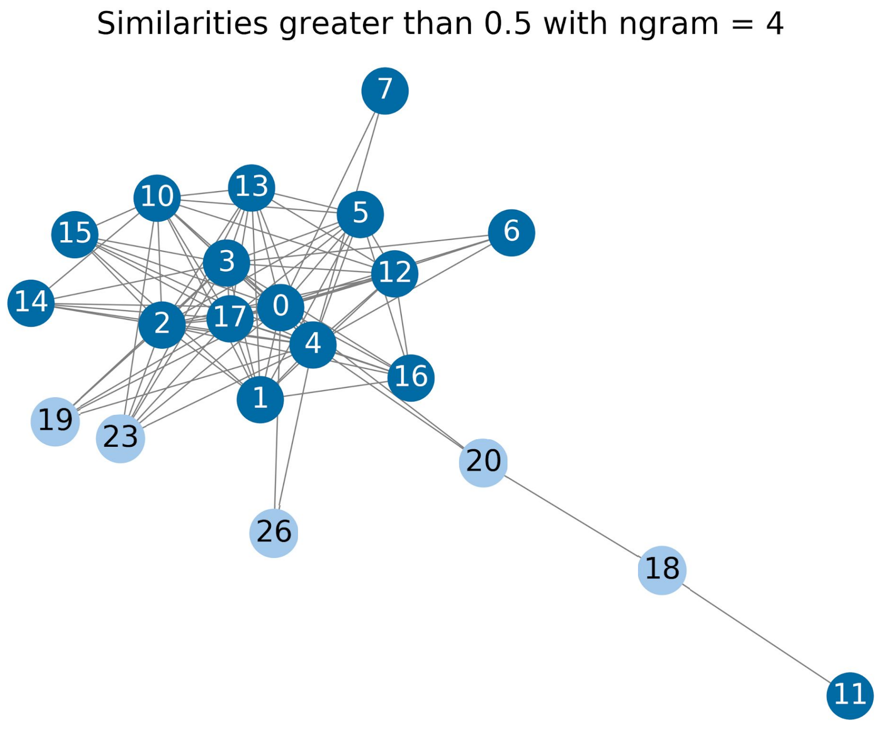


Figure 6.9: Graph network between players that are more than 50% similar between each other, using the dwell-trip-action construction. Advanced players are the dark blue nodes whereas beginner players are the light blue nodes.

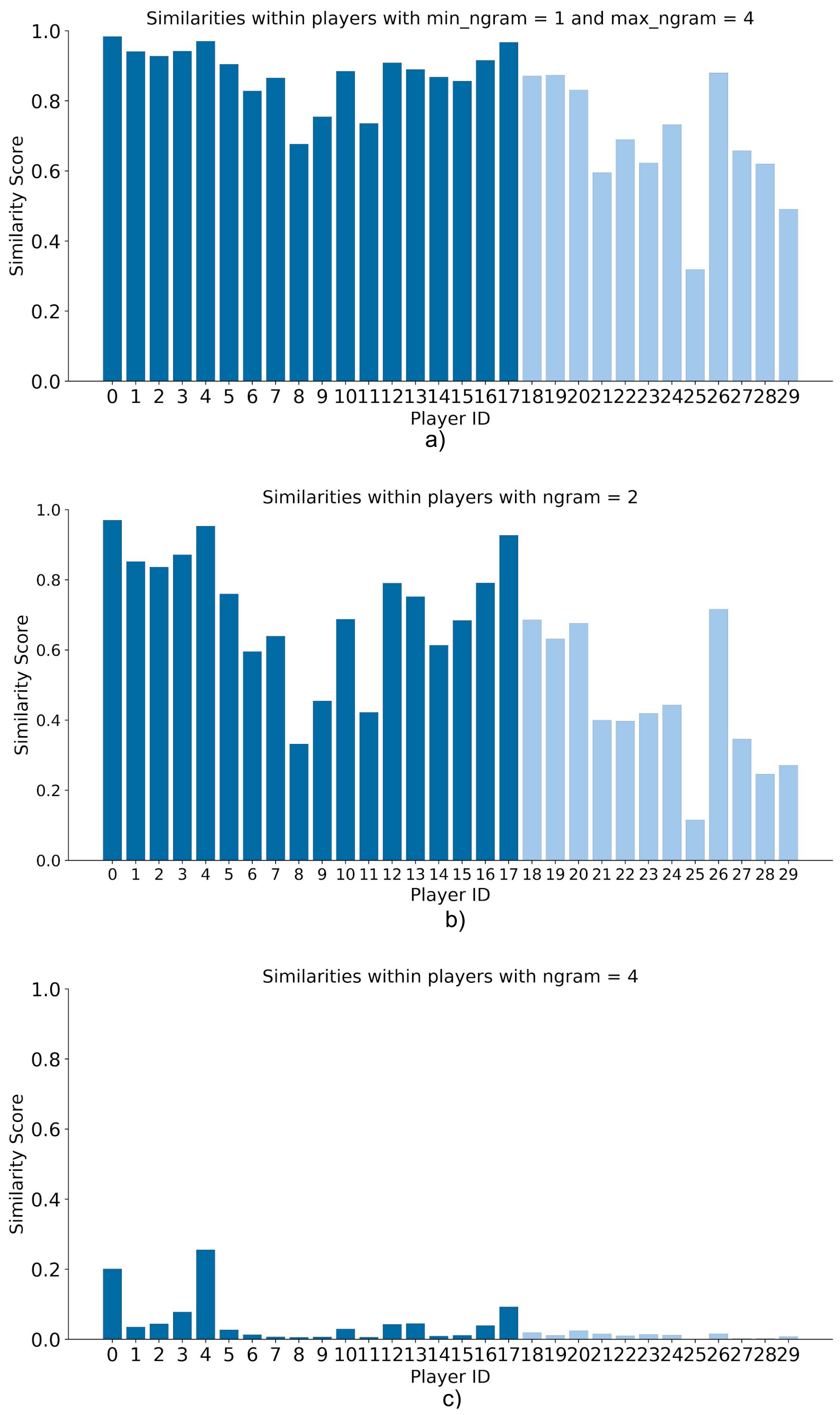


Figure 6.10: Similarity scores within players using the dwell-trip construction. Players are in descending order of experience level.

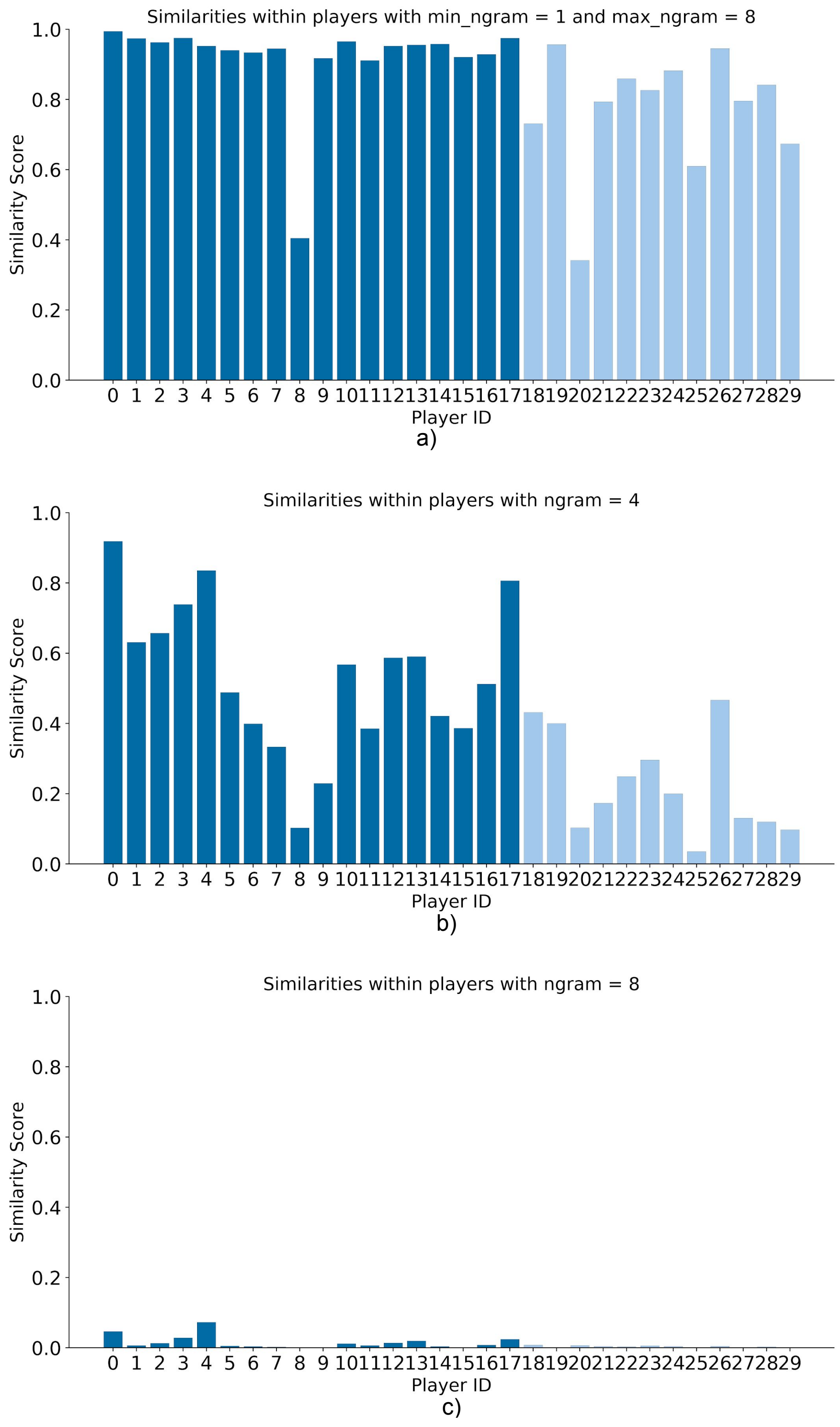


Figure 6.11: Similarity scores within players using the dwell-trip-action construction. Players are in descending order of experience level.

Chapter 7
Discussion

Researchers have proposed different methods to analyze player behavior; however, there is a lack of a player modeling approach for open and endless games (sandbox games). Although many research claim to provide a general player modeling, they usually assume that the game has defined goals and an end. In this thesis, we developed a technique named StABLE that makes it possible to analyze player behavior in sandbox games. StABLE requires minimal interference from designers while encoding player behavior into strings and applying similarity techniques from Natural Language Processing (NLP) to uncover player behavior similarity. Our results demonstrated that it is possible to use the StABLE technique to compare strings of movement and activity patterns to answer a number of important questions in game analytics for open and endless games. With StABLE, we could answer the six proposed research questions for a sandbox game.

RQ1 Do players have similar visited frequency behaviors in the game?

RQ2 Do players who explore more of the game present similar visited frequency?

RQ3 Do players change their visited frequency behavior through the game?

RQ4 Can we cluster the players based on the self reported motivation?

RQ5 Do players who explore more of the game present more similar movements and activities than players who did not?

RQ6 Do players have similar movements, with and without activity, when compared to themselves at the beginning and end of the game?

The discussion of results are listed below. These findings demonstrate the scope of utility of the StABLE approach to game log analysis.

RQ1, RQ2, and RQ3 We found that players did not have similar location strings to each other, which is expected in an open world game like Minecraft. In addition, the similarities between advanced and beginner players were not significantly different. Both results might be related to the fact that Minecraft world is procedurally generated and the resources are randomly placed in the world. The same resources can be found in different areas in the game, leading to a lack of patterns on how players visited places. However, players did have similarity within themselves, which is also anticipated, as each player's home location and preferred resource collection grounds should have been somewhat stable over the play period.

RQ4 By aggregating away location, and examining patterns of travel and dwelling, we demonstrated that we could probe how people play. Unsupervised clustering of the similarities partitioned players by motivation, as encoded by the SIMS response, demonstrating that our technique can be used to identify why people play from how people play, a canonical question in the literature [14, 68]. We found that players can naturally be grouped based on their identified regulation and amotivation. Further interpretation for these results would require a psychological expert in game user research.

RQ5 and RQ6 Finally, by analyzing movement and activity within and between individuals, we could group players into beginner and advanced categories. This analysis further showed that the movement and activity patterns associated with expert players were more likely to be more similar than beginner players with each other. This finding is interesting in the sense of pursuing movements similar to the most advanced players can result in progress in the game. Intriguingly, movement and activity pattern differentiability between advanced and beginner players seems to be independent of N. While similarity decreased substantially with N, the groups of advanced and beginner players were still significantly different. Advanced players also demonstrated more stable play patterns from movement pattern perspective.

How and why people play digital games is a core question that game designers and researchers seek to answer. In this thesis, we have provided a novel methodology to help answer these questions. Like others before [23, 28, 30, 40, 48], we observed that the sequential records of movement and actions encapsulated in a log could be rendered as strings, and probed using tools from semantic analysis. Similar to prior work by Dodge [23], we realized that feature engineering of the sequence allows analysts to probe questions that cannot be addressed by examining locations or actions alone. In particular, by intelligently aggregating away strict dependence on time, space, or activity type, we could compare similarities in movement quality as well as the realized movements. Combining similarity calculations with graph-based clustering algorithms allowed us to differentiate players by experience and motivation. Stratifying within groups allowed us to identify experience, as more experienced players had more stable movement qualities. While strict location-based analysis yielded little actionable insight, we were able to uncover patterns of motivation and experience by crafting meaningful feature sequences from the game logs.

7.1 Contributions

The study conducted to this thesis is an attempt to extend player modeling to make analysis of open and endless games possible. The findings we described here have strong internal validity, but weaker generalizability given the single game, limited demographic, and relatively small number of participants. However, the intent of this work was not to provide strongly generalizable findings relating movement patterns to experience, but to demonstrate that StABLE, as a tool for game log analysis, has substantial utility. Our work has the following contributions:

Provide unsupervised player behavior analysis algorithm. Other methods such as Glyph [46] and Gamalyzer [48] require designers to provide parameters that are not intuitive. StABLE, on the other hand, only needs a clean dataset containing the player ID, location, and activity. By default, StABLE stratifies these locations into a grid with the cells as the smallest unit in the game which only works for games with non-continuous space. Designers have the option to provide other cells sizes, or a dataset with a column named `loc_bin` with their own stratification criteria, addressing the problem of discretization for continuous space.

Perform within-player and between-player analysis. Most of the studies perform either within-player or between-player analysis but not both together. Few studies, however, addressed this problem but for goal modeling, not player modeling [34, 43]. Another common problem found in the existent methods is that they are focused on map visualization, which can lead to a polluted outcome and difficult analysis [2, 7, 13, 46, 48]. StABLE can be used for both within-player and between-player analysis by returning the similarity matrix, which designers can represent in different manners (e.g., box/bar plots and network graphs) to avoid information overload.

First to model players behavior as a pair of dwell time and trip duration. To the best of the author's knowledge, no other study in game user research (GUR) used derived parameters from locations to describe players behavior. The use of derived features from locations can be found in GIS and GPS studies such as [23, 24, 54]; in our findings, we showed that it is possible to obtain insights from players behavior by only using dwell times and trip durations.

First to render players behavior as sentences. The work done by Bunian et al. [12] also focused on player modeling. However, they developed a predictive model with Hidden Markov Models (HMM), where we were focused on creating a feature (the similarity score) as a way to model players. We were able to create StABLE by encoding players behavior as sentences. Other studies in GUR showed that encoding actions into strings is promising for goal modeling purpose [46, 48]. Nonetheless, we believe that StABLE is the first technique to use sentences to model players behavior, not goals.

First to model players behavior for a sandbox game. As described in the literature review, existing player modeling is focused on goal modeling and on games with an end and a lack of research on player modeling for open world and endless games (sandbox games) [35]. Applying StABLE with Minecraft, one of the most popular sandbox games, we were able to show that StABLE can be used to analyze player behavior for sandbox games. We believe we are the first one to provide such a method.

Provide broad applicability. We were able to address six different research questions with the three string constructions we developed in this thesis: visited locations, dwell-trip, and dwell-trip-action. While we focused on movement data in an open world game in this work, many other genres of games and streams of data, and player attributes could be analyzed. This technique is not limited to games, and could be broadly applicable to Human Computer Interaction. For example, eye tracking data or galvanic skin response data could also be meaningfully analyzed using this approach. Essentially, the string represents features of interest of player behavior through time, from which designers hope to gain insight through comparison. When rendering strings, analysts must make two key decisions: the nature of the features in the strings, and the definitions of words.

7.2 Shortcomings and Future Work

Because we focused on providing broad evidence of applicability rather than focusing the application of StABLE to a single problem, this work raises more questions than answers with respect to game behaviors. We established that movement pattern similarity could cluster players a way that reflected their SIMS scores, but did not dig into which playstyle patterns were diagnostic. We demonstrated that, for Minecraft at least, advanced players had greater movement pattern similarity to each other than to beginner players, but did not probe the nature of those movement patterns. We showed that advanced players had more stable movement patterns than beginner players, but did not attempt to describe those patterns. Answering the game play question, as opposed to the methodological utility addressed here, is a potentially fruitful and fascinating avenue for future research. Another interesting future work is to explore the use of StABLE with different randomly generated Minecraft worlds instead of a common world for all players as we addressed in this thesis.

The primarily limitation of this work is the scope of the data analyzed. By only analyzing a relatively small population over a single game, we cannot make strong conclusions about our game-related findings, other than the method was able to generate them. As a methodological work, this proof is sufficient; however, more detailed analysis of larger datasets, preferably obtained from actual commercial games is an obvious next step. Within the method itself, many potential avenues for further improvement exist. We employed the simplest (and most canonical) string comparison method from the text mining literature. Extending this work to include more advanced and nuanced string comparison algorithms could be potentially fruitful. The core contribution and insight in this work is the use of feature sequences to selectively probe specific behavior issues. We have only explored three simple feature constructions: location, dwell-time and dwell-time-action. Many more sophisticated feature strings could be envisioned, and should be investigated. We also implemented a single location stratification and other discretization techniques can be added to StABLE. Finally, we have focused on game log data, in part because it is limited and precise. However, these techniques could be employed to examine human or animal behavior in the real world where GPS or other locational datasets are available.

7.3 Conclusion

Game logs contain rich information on player experience, but this data can be difficult to mine for insights. We have presented the StABLE method for log analysis, which leverages techniques from text mining to perform comparisons of sequence of behavioral features. Using StABLE we were able to mine insights about player behavior, experience and motivation from Minecraft play logs collected over a two week period. The findings demonstrate the utility of our approach in game log analytics, and point the way towards both additional research in methods, and application to game log analysis in both commercial and research settings.

References

[1] Charu C. Aggarwal and Cheng Xiang Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[2] Erik Andersen, Yun Enedu Liu, Ethan Apter, François Boucher-Genesse, and Zoran Popović. Gameplay analysis through state projection. In *FDG 2010 - Proceedings of the 5th International Conference on the Foundations of Digital Games*, pages 1–8, 2010.

[3] Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet distance for curves, revisited. In *European Symposium on Algorithms*, pages 52–63. Springer, 2006.

[4] ESA The Entertainment Software Association. Industry facts at `http://www.theesa.com/about-esa/industry-facts/`, 2011.

[5] Jeremy N. Bailenson and Nick Yee. A longitudinal study of task performance, head movements, subjective report, simulator sickness, and transformed social interaction in collaborative virtual environments. In *Presence: Teleoperators and Virtual Environments*, volume 15, pages 699–716, 2006.

[6] Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD Research*, 1(1):19, 1996.

[7] Christian Bauckhage, Rafet Sifa, Anders Drachen, Christian Thurau, and Fabian Hadiji. Beyond heatmaps: Spatio-temporal clustering using behavior-based partitioning of game levels. In *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2014.

[8] Marya Bazzi, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. Community detection in temporal multilayer networks, with an application to correlation networks. *Industrial and Applied Mathematics*, 14(1):1–41, 2016.

[9] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[10] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[11] Vincent D Blondel, Jean Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008.

[12] Sara Bunian, Alessandro Canossa, Randy Colvin, and Magy Seif El-Nasr. Modeling Individual Differences in Game Behavior using HMM. *Proceedings, The Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2018.

[13] Jonathan C Campbell, Jonathan Tremblay, and Clark Verbrugge. Clustering Player Paths. In *FDG 2015 - Proceedings of the 10th International Conference on the Foundations of Digital Games*. McGill, 2015.

[14] Alessandro Canossa, Josep B. Martinez, and Julian Togelius. Give me a reason to dig Minecraft and psychology of motivation. In *IEEE Conference on Computatonal Intelligence and Games, CIG*, pages 1–8. IEEE, 2013.

[15] Alessandro Canossa, Truong-Huy D. Nguyen, and Magy Seif El-Nasr. G-Player: Exploratory Visual Analytics for Accessible Knowledge Discovery. In *Proceedings of the First International Joint Conference of DiGRA and FDG*, 2016.

[16] Van M Chhieng and Raymond K Wong. Adaptive distance measurement for time series databases. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4443 LNCS, pages 598–610, 2007.

[17] Paul T Costa and Robert R Mac Crae. *Neo personality inventory-revised (NEO PI-R)*. Psychological Assessment Resources Odessa, FL, 1992.

[18] Paul Coulton, Will Bamford, Keith Cheverst, and Omer Rashid. 3d space-time visualization of player behaviour in pervasive location-based games. *International Journal of Computer Games Technology*, 2008:2, 2008.

[19] Edward L Deci and Richard M Ryan. Intrinsic motivation. *The Corsini Encyclopedia of Psychology*, pages 1–2, 2010.

[20] Edward L Deci and Richard M Ryan. Self-determination theory. In *P. A. M. Van Lange, A. W. Kruglanski, E. T. Higgins (Eds.), Handbook of theories of social psychology*, pages 416–436. Sage Publications Ltd, 2012.

[21] Mouna Denden, Ahmed Tlili, Fathi Essalmi, and Mohamed Jemni. Does personality affect students' perceived preferences for game elements in gamified learning environments? In *Proceedings - IEEE 18th International Conference on Advanced Learning Technologies, ICALT 2018*, pages 111–115, 2018.

[22] Forge Development LLC. Minecraft forge at `https://files.minecraftforge.net/`, 2018.

[23] Somayeh Dodge, Patrick Laube, and Robert Weibel. Movement similarity assessment using symbolic representation of trajectories. *International Journal of Geographical Information Science*, 26(9):1563–1588, 2012.

[24] Somayeh Dodge, Robert Weibel, and Ehsan Forootan. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environment and Urban Systems*, 33(6):419–434, nov 2009.

[25] Anders Drachen and Alessandro Canossa. Analyzing spatial user behavior in computer games using geographic information systems. In *Proceedings of the 13th international MindTrek conference: Everyday life in the ubiquitous era*, pages 182–189. ACM, 2009.

[26] Anders Drachen and Matthias Schubert. *Game Analytics*, chapter Spatial Game Analytics, pages 365–402. Springer, 2013.

[27] Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa. *Game Analytics*. Springer, 2013.

[28] Katayoun Farrahi and Daniel Gatica-Perez. Discovering routines from large-scale human locations using probabilistic topic models. *ACM Transactions on Intelligent Systems and Technology*, 2(1):1–27, 2011.

[29] Henry D Fernández, Mikami Koji, and Kunio Kondo. Adaptable game experience based on player's performance and EEG. In *Proceedings - 2017 NICOGRAPH International, NICOInt 2017*, pages 1–8, 2017.

[30] Laura Ferrari, Alberto Rosi, Marco Mamei, and Franco Zambonelli. Extracting urban patterns from location-based social networks. In *ACM LBSN '11*, page 1. ACM, 2011.

[31] Ben Gilbert. 'Minecraft' has been quietly dominating for over 10 years, and now has 112 million players every month — Business Insider at `https://www.businessinsider.com.au/minecraft-monthly-player-number-microsoft-2019-9`, 2019.

[32] Frédéric Guay, Robert J Vallerand, and Céline Blanchard. On the assessment of situational intrinsic and extrinsic motivation: The Situational Motivation Scale (SIMS). *Motivation and Emotion*, 24(3):175–213, 2000.

[33] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, Third Edition.* Morgan Kaufmann, Waltham, Mass., 3rd edition edition, July 2011.

[34] Brent Harrison and David L Roberts. Using sequential observations to model and predict player behavior. In *Proceedings of the 6th International Conference on the Foundations of Digital Games, FDG 2011*, pages 91–98, 2011.

[35] Danial Hooshyar, Moslem Yousefi, and Heuiseok Lim. Data-driven approaches to game player modeling: A systematic literature review. *ACM Computing Surveys*, 50(6), 2018.

[36] Anna Huang. Similarity measures for text document clustering. In *New Zealand Computer Science Research Student Conference, NZCSRSC 2008 - Proceedings*, pages 49–56, 2008.

[37] Oliver P John, Eileen M Donahue, and Robert L Kentle. The big five inventory: Versions 4a and 54. Technical report, University of California, Institute of Personality Assessment and Research, 1991.

[38] Kellie Ell. Video game industry is booming with continued revenue at `https://www.cnbc.com/2018/07/18/video-game-industry-is-booming-with-continued-revenue.html`, 2018.

[39] Eunjo Lee, Jiyoung Woo, Hyoungshick Kim, Aziz Mohaisen, and Huy Kang Kim. You are a Game Bot!: Uncovering Game Bots in MMORPGs via Self-similarity in the Wild. In *NDSS*, 2016.

[40] Kyunghan Lee, Seongik Hong, Seong Joon Kim, Injong Rhee, and Song Chong. SLAW: A mobility model for human walks. In *Proceedings - IEEE INFOCOM*, pages 855–863, 2009.

[41] Jed A Long and Trisalyn A Nelson. A review of quantitative methods for movement data. *International Journal of Geographical Information Science*, 27(2):292–318, 2013.

[42] Robert R. McCrae and Oliver P John. An Introduction to the Five-Factor Model and Its Applications. *Journal of Personality*, 60(2):175–215, 1992.

[43] Wookhee Min, Eun Young Ha, Jonathan Rowe, Bradford Mott, and James Lester. Deep learning-based goal recognition in open-ended digital games. In *Proceedings of the 10th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2014*, pages 37–43, 2014.

[44] Pejman Mirza-Babaei, Lennart Nacke, and Anders Drachen. Games User Research. pages 1–4. Oxford University Press, 2018.

[45] Stephan Mueller, Barbara Solenthaler, Mubbasir Kapadia, Seth Frey, Severin Klingler, Richard P Mann, Robert W Sumner, and Markus Gross. HeapCraft: Interactive data exploration and visualization tools for understanding and influencing player behavior in Minecraft. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, MIG 2015*, pages 237–241, 2015.

[46] Truong-Huy Dinh Nguyen, Magy Seif El-Nasr, and Alessandro Canossa. Glyph: Visualization Tool for Understanding Problem Solving Strategies in Puzzle Games. In *Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG 2015)*, 2015.

[47] Joseph C Osborn and Michael Mateas. A Game-Independent Play Trace Dissimilarity Metric Categories and Subject Descriptors. In *FDG 2014 - Proceedings of the 9th International Conference on the Foundations of Digital Games*, 2014.

[48] Joseph Carter Osborn, Benjamin Samuel, and Michael Mateas. Visualizing the strategic landscape of arbitrary games. *Information Visualization*, 17(3):196–217, 2018.

[49] Tuhin Paul, Kevin Stanley, Nathaniel Osgood, Scott Bell, and Nazeem Muhajarine. Scaling behavior of human mobility distributions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9927 LNCS, pages 145–159. Springer, Cham, 2016.

[50] Steven Reiss. *The normal personality: A new way of thinking about people.* Cambridge University Press, 2008.

[51] Richard M Ryan and Edward L Deci. A motivational approach to self: Integration in personality. *Perspectives on Motivation*, 38(237):237–288, 1991.

[52] Richard M Ryan and Edward L Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, 55(1):68, 2000.

[53] Alain Saas, Anna Guitart, and Africa Perianez. Discovering playing patterns: Time series clustering of free-to-play game data. In *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2017.

[54] Luca Scherrer, Martin Tomko, Peter Ranacher, and Robert Weibel. Travelers or locals? Identifying meaningful sub-populations from human movement data in the absence of ground truth. *EPJ Data Science*, 7(1):19, 2018.

[55] Kelly M Searle, Jailos Lubinda, Harry Hamapumbu, Timothy M Shields, Frank C Curriero, David L Smith, Philip E Thuma, and William J Moss. Characterizing and quantifying human movement patterns using gps data loggers in an area approaching malaria elimination in rural Southern Zambia. *Royal Society Open Science*, 4(5):1–12, 2017.

[56] Manu Sharma, Santiago Ontañón, Manish Mehta, and Ashwin Ram. Drama management and player modeling for interactive fiction games. *Computational Intelligence*, 26(2):183–211, 2010.

[57] Manu Sharma, Santiago Ontañón, Christina Strong, Manish Mehta, and Ashwin Ram. Towards player preference modeling for drama management in interactive stories. In *Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2007*, pages 571–576, 2007.

[58] Sam Snodgrass, Omid Mohaddesi, and Casper Harteveld. Towards a generalized player model through the PEAS framework. In *FDG '19*, pages 1–7, 2019.

[59] Chaoming Song, Tal Koren, Pu Wang, and Albert László Barabási. Modelling the scaling properties of human mobility. *Nature Physics*, 6(10):818–823, oct 2010.

[60] Robert J Vallerand. Toward a hierarchical model of intrinsic and extrinsic motivation. In *Advances in Experimental Social Psychology*, volume 29, pages 271–360. Elsevier, 1997.

[61] Giel van Lankveld, Sonny Schreurs, and Pieter Spronck. Psychologically verified player modelling. In *10th International Conference on Intelligent Games and Simulation, GAME-ON 2009*, pages 12–19, 2009.

[62] Giel Van Lankveld, Sonny Schreurs, Pieter Spronck, and Jaap Van Den Herik. Extraversion in games. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6515 LNCS, pages 263–275, 2011.

[63] Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

[64] Günter Wallner, Nour Halabi, and Pejman Mirza-Babaei. Aggregated visualization of playtesting data. In *Conference on Human Factors in Computing Systems - Proceedings*, volume 12. ACM, 2019.

[65] Tom Wijman. The Global Games Market Will Generate $152.1 Billion in 2019 as the U.S. Overtakes China as the Biggest Market, 2019.

[66] Georgios N Yannakakis, Pieter Spronck, Daniele Loiacono, and Elisabeth André. Player Modeling. *Dagstuhl Follow-Ups*, 6:59, 2013.

[67] Georgios N. Yannakakis and Julian Togelius. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, 2(3):147–161, jul 2011.

[68] Nick Yee. Motivations for play in online games. In *Cyberpsychology and Behavior*, volume 9, pages 772–775, 2006.

[69] Nick Yee, Helen Harris, Maria Jabon, and Jeremy N Bailenson. The expression of personality in virtual worlds. *Social Psychological and Personality Science*, 2(1):5–12, 2011.

[70] Yihong Yuan and Martin Raubal. Measuring similarity of mobile phone user trajectories–a spatio-temporal edit distance method. *International Journal of Geographical Information Science*, 28(3):496–520, 2014.

[71] Rui Zhang, Kevin G Stanley, Daniel Fuller, and Scott Bell. Differentiating Population Spatial Behavior using Representative Features of Geospatial Mobility (ReFGeM). *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 1(1):1–25, 2019.

[72] Guanghu Zhu, Tao Liu, Jianpeng Xiao, Bing Zhang, Tie Song, Yonghui Zhang, Lifeng Lin, Zhiqiang Peng, Aiping Deng, Wenjun Ma, and Yuantao Hao. Effects of human mobility, temperature and mosquito control on the spatiotemporal transmission of dengue. *Science of the Total Environment*, 651:969–978, Feb 2019.