UNIVERSITY OF OSLO
Department of Informatics

# On the predictability of server resources in online games, an investigative approach.

Master thesis

Jon-Erik Tyvand

Network and System Administration

Oslo University College

**May 24, 2011**

# On the predictability of server resources in online games, an investigative approach.

Jon-Erik Tyvand

Network and System Administration
Oslo University College

May 24, 2011

**Acknowledgements**

I owe my deepest gratitude to my supervisor, Kyrre Begnum. His extraordinary level of knowledge has always amazed me and motivated me in many ways. Without his unique abilities to encourage, guide and advice, I would not be able to achieve the same quality of work what I have today. He has from the very beginning of this project always been available and enormously helpful. Thank you.

I would like to thank the authority of Oslo University College (OUC) for providing me with a good environment and facilities to complete this project. Also, I would like to take this opportunity to thank the Department of Informatics in University of Oslo (UiO) for offering this Master degree program. In addition, I would like to thank Hugo Lewi Hammer which have provided valuable ideas on a important part of this project.

Finally, an honorable mention goes to my family and friends for their understandings and supports in completing this project.

## Abstract

*This Master thesis investigate the predictability in game server resource data by implementing and developing a predictive algorithm. Thorough testing of the algorithm has been performed and the results show that the game server resource data is predictable to some extent. The findings in this thesis introduce the possibilities to predict allocation of sufficient resources to game servers.*

# Contents

# List of Figures

# Chapter 1

# Introduction

Online games as a service is a challenging field in system administration, where providers compete for the best possible experience for their players. Just a slight insufficiency of resources will lead to a noticeable change and will result in a decreased quality of service.

A game server consumes resources when players are connected to the server. The more players connected, the more resources are consumed. At some point the saturation point for the server's resources will be met, and from there on players will experience reduced gaming quality. It is therefore crucial to always have available resources for the game servers. On the other hand, savings can be made to prevent the excessive allocation of possibly unused resources [14]. There is a pressure towards system administrators to optimize resources to reduce costs and at the same time having minimal impact on the environment.

At various times of day, the number of players connected will vary. The server resources are often set excessively high to guarantee satisfied players all the time. Some resources will therefore be unused parts of the day. An dynamic adjustment of resources would therefore optimize cost performance but there is no particular method on how this can be done on game servers today.

There are existing algorithms which respond to a tendency if necessary, these are called reactive algorithms. Reactive algorithms makes it possible to allocate resources depending on how many players that is currently connected. However, these algorithms do not anticipate the need for resources.

A predictive algorithm has the potential to secure enough resources in advance. A reactive algorithm may periodically offer too much or too little resources. By introducing a predictive algorithm, these periods might be reduced. Although a predictive algorithm assumes that you already have data and that the data has the property of being repetitive.

## 1.1 Problem statement

*Investigate the predictability of game server resource usage by developing and implementing a predictive algorithm.*

**The game server resource usage** is not simulated but real data extracted directly from a game server or applications such as Steam. This data shows how many people have been online over a period of time.

**Predictability** is to a certain degree how successful one is to investigate the predictability of the game server resource usage.

**Predictive algorithm** is a tool to determine the predictability of the game server resource usage.

# Chapter 2

# Background

This chapter will provide a short introduction to some of the topics that is relevant or in correlation with this project. First, Online gaming as a service. Then the issues of green computing and the power consumption in data centers. Furthermore, resource management, including research done in autonomic power management. Information about data mining, and data mining techniques. About predictability and methods to measure predictability, and finally, the idea behind a new predictive algorithm.

## 2.1 Online Gaming as a Service

Online gaming today place high demands on servers that are running the service. Gamers demands enough resources from the game servers so that they get an expected gaming experience. The game industry is forced to follow these demands in order to keep their customers. Online gaming has become such a large service that it takes many data centers with multiple servers in order to manage just one game [33]. Because of this, the power consumption of data centers has an huge impact on the environment.

Network Quality of Service can correspond to Server resource QoS in online gaming. This because the gaming experience will feel the same if the quality deteriorate. Several studies has evaluated the effect of network quality on online gamers. [2] [4] [9] [24] [40] In most of these referenced studies players have been graded either subjectively or objectively in controlled network environment. Most of these studies has the same conclusion, network quality has a great effect in gaming experience. [42] [57] [58]

In general, perception of a players decision to quit or continue playing a game is difficult to study because there are so many factors that affect human decisions. A study in Computing and Network Security Laboratory at National Taiwan University ask the question, how sensitive are online gamers to network quality? [10] This study has shown that game playing time is strongly related to network QoS and is a potential indicator of user satisfaction. This indicates the importance of QoS in online gaming.

## 2.2 Green Computing

Green computing has been a hot topic the last few years. Power consumption, waste disposal and environmental effects of production has become more apparent over the years. One of the reasons for environmental change is the growth in IT systems, so one of the main goals of green computing is to use the computer resources as efficiently as possible, while maintaining or increasing the computer performance [23].

Over the last few years, the technologies to reduce power consumption have improved, especially on laptops where the battery time has improved drastically [47]. However, over the same time period the overall power consumption has increased because of the increase in IT systems [6]. The focus on reducing the consumption of power has been on processing power [32]. Processors consume most of the power in the majority of computers or servers [35], by following the well-known Moore's law [39] the power consumption by processors has been reduces over time [17].

### 2.2.1 Power Consumption in Data Centers

Data centers has grown in both number and size, with the increasing need of power to run the server and the cooling systems, the main focus of green computing lies in methods to reduce the power demands in these data centers [32]. The wasteful energy consumption of a data center can easily account for more than half of both the electricity bill and the corporate carbon footprint in the most IT organizations [23].

A recent Internet Data Center report estimated the worldwide cost on enterprise power consumption exceeds $30 billion in year 2008 and is likely to surpass the worldwide spendings on new server hardware in 2008. The rated power consumptions of servers has increased by 10 times over the past ten years [45]. The huge amount of power consumption calls for the need of new energy efficient methods.

In Green Computing the IT energy management is the analysis and management of energy demand within the information technology arena. The IT energy management have found that the global IT energy demand accounts for approximately 2%

of global energy demand, this is approximately at the same level as aviation. [19] IT equipment can account for 25% of a modern office buildings energy cost. [37] The main sources of IT energy consumption are PCs and Monitors, accounting for 39% of energy use, followed by data centers and servers, accounting for 23% of energy use. [30]

However, the challenges of today does not necessarily lie in creating new power efficient technologies, but using knowledge and technology that we already have [23].

### 2.2.2 Resource Management

Recently, the focus in computer systems has shifted from purely performance to good performance at lowest possible power consumption [26]. A study performed at Intel Research show power control algorithms that attempt to reduce power consumption of a resource by taking advantage of available low power states. This study compared proactive power control algorithms to reactive power control algorithms. The study showed that proactive algorithms can provide some added benefits at moderate traffic loads [26].

Researchers try to find effective solutions to make data centers reduce power consumption while keeping the desired quality of service [34]. Researchers at IBM China Research Laboratory, McGill University and University of New Mexico have developed a Green Cloud architecture, which aims to reduce data center power consumption, while guarantee the performance from users perspective. They have verified the efficiency and effectiveness of the Green Cloud architecture by taking an online real-time game, Tremulous, as a VM application. The evaluation results show that they saved up to 27% of the energy when applying Green Cloud architecture to this game.

The need to improved power management in data centers is becoming essential, one of the must promising topics on this is improving Autonomic power management systems [28]. Autonomic power management is defined as a management system that to a certain degree can manage itself given a set of objectives from an administrators [27].

Kandasamy et al[25] propose a control mechanism on the processor to optimize expected behavior using a mathematical model based on a limited look-ahead prediction. This model can be applied to reduce power consumption in processors. Sharma et al [49] have implemented algorithms inside the Linux kernel that scale voltage dynamically in QoS-enabled web-servers, to minimize energy consumption without violating any quality of service constraints.

A well-studied technique for increasing data center energy efficiency is dynamic server consolidation. This method migrates application workload onto a minimal number of servers and putting the unused servers into a low-power state.[8] [15] [11] [5] [14]. however, the primary challenge of this technique is the decision-making aspects, which in enterprise data centers can be very complexed. Interesting work on this challenge has been done [22].

## 2.3 Predictability

Predictability is the degree to which a correct prediction can be made either qualitatively or quantitatively. In experimental physics, there are always observational errors determining variables such as positions and velocities. So perfect prediction is practically impossible. Moreover, in modern quantum mechanics, Werner Heisenberg's Uncertainty principle [18] puts limits on the accuracy with which such quantities can be known. So such perfect predictability is also theoretically impossible.

The ability to predict the future has been found useful in many situations. Although Werner Heisenberg's Uncertainty principle state that perfect predictability is theoretically impossible, in many cases the predictability is very accurate, and does not need to be 100% accurate to provide useful information. The variety in information that is being predicted is huge, all from the stock marked, sport results, to tomorrows weather. Some of these topics is easier to predict than others.

For example, predicting when solar or lunar eclipses occur. [53]. The Moon moves with a repetitive circular orbit around the Earth and the Earth moves around the sun. Because of this almost precise repetitive movement one can predict with a very good accuracy when and where a lunar or solar eclipse is going to occur on earth. However, predicting the weather is far more complex [41]. Weather predicting is made by gathering data about the current state of the atmosphere and using scientific understanding of atmospheric process to predict how the atmosphere will evolve. Predicting become less accurate over time because of incomplete understanding and errors involved in measuring the initial conditions of the atmosphere.

Predictive inference is an interpretation of probability that emphasizes the prediction of future observations based on past observations [20]. The more repetitive observations, the more accurate predictions. Using historical data for prediction is normally used when predicting a repetitive nature, e.g. Human behavior.

Techniques to determine predictability already exists today. In statistics, dependence is the statistical relationship between two sets of data. Correlation is the statistical relationship involving dependence. The most familiar measure of dependence between two quantities is the Pearson product-moment correlation coefficient [48].

Pearson's correlation is a measure of the correlation between two variables X and Y, giving a value between +1 and -1. The values between +1 and -1 is the correlation coefficient range. A value of 1 implies that a linear equation describes the relationship between X and Y perfectly, with all data point lying on a line for which Y increases as X increases. A value of -1 implies that all data points lie on a line for which Y decreases as X increases. A value of 0 implies that there is no linear correlation between the variables. This method could be used to correlation between existing data and predicted data, this would provide an easy understandable value to determine the predictability.

## 2.4 Data mining

In computer science, data mining is the process of discovering interesting, useful patterns and relationships from large data sets by combining methods from statistics with database management. [12] The data mining process is either automatic or manual, but is usually used as a cooperative work of both humans and computers. Best results are achieved by balancing the knowledge of human experts in describing problems and goals combined with the search capabilities of computers. [56]
There are several types of data mining, typically divided into some kind of information known and the type of knowledge sought from the data mining model.

*Predictive modeling* is a data mining model that can be used when the information known is a numeric value and the goal is to predict that value for new data. A predictive model is created or chosen to try to best predict the probability of an outcome. This model is used widely in information technology such as Spam filtering system. Predictive modeling is then used to identify the probability that a given message is Spam. [21] Several predictive models already exists today.

- Naive Bayes classifier [46]
- k-nearest neighbor algorithm [44]
- Logistic regression [29]
- Group method of data handling (GMDH) [38]

*Anomaly detection* is another modeling technique in data mining. Anomaly detection, also referred to as outlier detection [31] Anomaly detection is the problem of finding patterns in data sets that do not appear to be normal behavior. [7]

Leap detection test or LDT is another method to detect anomalies. LDT for time series data was first purposed by Cochran [13] as a modification of the Chi-Squared test for distribution free data. This test detects anomalies based on periodic data behavior. Furthermore, the Leap Detection Test was improved to LDTi by Kyrre Begnum [36] [3].

## 2.5 The idea behind a Predictive Algorithm

The idea of a new predictive algorithm was formed in collaboration with Hugo Lewi Hammer (Associate professor, Oslo University College) and Kyrre Begnum (Associate Professor, Oslo University College) in the autumn 2010. The idea was to combine statistics with server resource data, in order to determine whether it is possible to foresee the need of server resources.

The thought was that previous repetitive data of used server resources, could be used to form a profile that would represent the probable resource usage of a typical day. This profile could then be used to predict a following day by somehow adapting the profile to the day. This way, one could say something like, "tell me the amount of resources that is in use now, and I will show you how the rest of the day will look like".

By implementing this algorithm as a resource management tool, one could (in theory) dynamically scale the resources to reduce power consumption.

This idea formed the beginning of this project.

## 2.6 Related Work

In a long-term study of the popular MMORPG game EvE online [16], predictability results of the workload of the game has shown to be highly predictive in short-term. This study focus on characterize players to understand their playing behavior. By understanding factors that contribute to players joining the game, players continuing their subscriptions, and players leaving the game, the publisher can provide in-game incentives and game updates to keep the player active.

## 2.7 Software and Services

This section will provide information about the software and services used to complete this study.

| Name | Description |
|---|---|
| Steam | Application developed by Valve Corporation. Used to distribute games and related media online. In this project this application is used to collect the current amount of players that is playing the different games [52]. |
| Sqlite3 | SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. [51] |
| Perl | Perl is a high-level, general-purpose, interpreted, dynamic programming language [50]. In this project, most of the developed tools was created in this programming language. |
| Tikz | programming languages for producing graphics directly into latex generated formats [54]. In this project, this programming language was used to create a plotting tool. |
| Twitter | Twitter is a website/service which offers a social networking and micro-blogging service [55]. In this project, the service was used to collect repetitive data to be tested on the developed predictive algorithm. |

# Chapter 3

# Approach

In this chapter the approach will be explained. The complexity of this project calls for a systematic approach to complete the project in the given time constraints. The following is a brief summary of the issues that will be gone through in this chapter:

- Preparing the game data.
- Design and develop the predictive algorithm.

  Preparing the algorithm for testing.
- Develop a test framework for the algorithm.

  Perform testing on the algorithm.
- Establish how the analysis should be done.

## 3.1 Preparing the data

This section deals with the necessary preparations needed for the data to be ready for testing.

### 3.1.1 Data fetching

Steam is primarily a multi-player and communication platform developed by Valve Corporation.[52] It is used to distribute games online. Steam also comes with other features. One of these features provide statistics of the current number of players online on the different games in Steam.

By retrieving this data periodically with constant intervals, one will over time have historic data of how many players that are online playing the different games at any time. If this data is repetitive, one can attempt to predict it.

This data already exists, a script has been collecting data every 5 minutes from the Steam application from 17th of may 2010. This means that the script has collected data in about one year, from 31 games and has stored the data in a datafile. The datafile is now 2164389 lines and is about 70 MB large.

Data processing is important with such large amount of data. This is because the data should be processed in a way that makes is easier to work with. The data obtained from the steam application is stored in a format that is hard to process. The following will address the issue.

Here is an explanation of the format of the raw data collected from the data fetching tool.

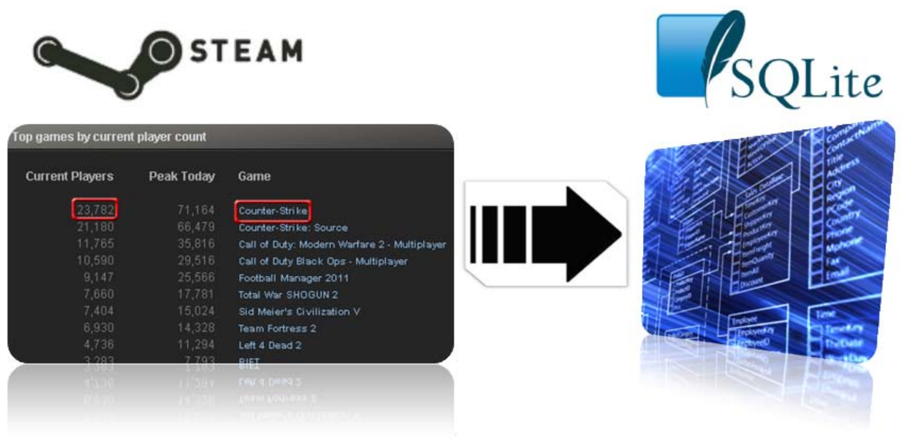
raw Steam data format:

```
<date>
<game1>, <value>
<game2>, <value>
<game3>, <value>
...
<gameX>, <value>
```

It would be easier to process the data if the data was stored in a database, this because the proper organization will allow for specific data extraction, something that will probably be of importance in this project. It will therefore be necessary to convert the data to a new format so that it fits into a database. The following is a purposed format:

purposed Steam data:

```
<date>,<yearday>,<weekday>,<game1>,<value>
<date>,<yearday>,<weekday>,<game2>,<value>
<date>,<yearday>,<weekday>,<game3>,<value>
...
<date>,<yearday>,<weekday>,<gameX>,<value>
```



Top games by current player count

| Current Players | Peak Today | Game |
|---|---|---|
| 23,782 | 71,164 | Counter-Strike |
| 21,180 | 66,479 | Counter-Strike: Source |
| 11,765 | 35,816 | Call of Duty: Modern Warfare 2 - Multiplayer |
| 10,590 | 29,516 | Call of Duty Black Ops - Multiplayer |
| 9,147 | 25,566 | Football Manager 2011 |
| 7,660 | 17,781 | Total War SHOGUN 2 |
| 7,404 | 15,024 | Sid Meier's Civilization V |
| 6,930 | 14,328 | Team Fortress 2 |
| 4,736 | 11,294 | Left 4 Dead 2 |
| 3,383 | 7,793 | BIET |

Figure 3.1: Illustration of Steam data collected to database.

Figure 3.1 illustrates how the data is to be inserted into the database. The current players online is collected for each game. This is done every five minutes. Date stamp, game name and the current player value is the variables that is stored in a database.

This section has addressed how the fetching of data should be conducted, the next task will be to prepare the data before it is tested on the algorithm.

### 3.1.2 Plan for gap analysis

It must be taken into account that the data fetching not always will be reliable. Instances can occur where the data is not collected or there are errors in the data. Early inspection of the data has revealed the presence of some gaps of data loss. This section will address the procedure for handling these data errors.

Some kind of data handling should always be taken into consideration before starting any test phase. If this is not taken into account, errors in the data can affect the results. This could cause that the entire analysis and the conclusion to become inaccurate. It is therefore critical in this situation that the data the algorithm will be tested on is checked so one is aware of the errors and is capable of some sort of repairing of data if desired. This because it could be interesting to run both processed and un-processed data on the algorithm to see how the algorithm perform with these errors.

## 3.2 Algorithm Outline

This section will explain the design of the predictive algorithm. The design reflects the idea behind the algorithm and the desired functionality.

The purpose of a predictive algorithm is to have the ability to calculate whats going to happen in the future. The idea is to design an algorithm that would create a profile based on statistical operations on data from previous days. The profile should reflect how a day in the near future would look like and be able to stay equal to or just above that day the whole time. By adapting the profile to data on a specific time on a following day, the profile should be able to predict how the rest of that day should look like. The illustrations below shows an example of how the profile adapts by using the value at present time. By using any value at any given time the profile should adapt and calculate how the rest of the day should look like.

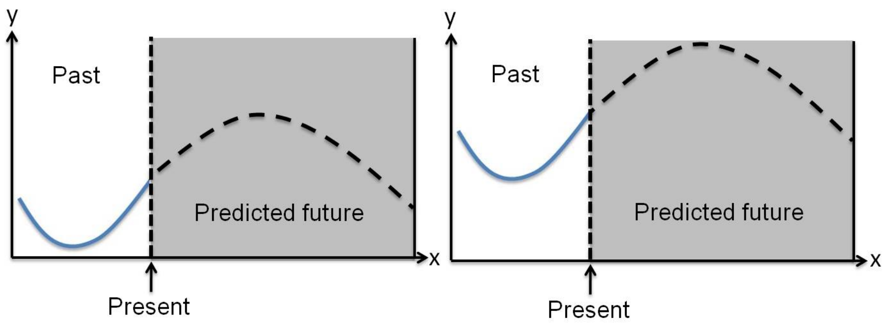


Figure 3.2: The algorithm adapts based on present value.

The figure 3.2 demonstrate two scenarios in which the algorithm is given different values at present time. The figure on the left side is given a lower value than the one on the right side, the algorithm then adjusts the profile based on what kind of value it receives and calculate the curve such it predicts how the future should look like.

Developing and testing this algorithm will be the basis of finding the predictability in the data it is tested against.

## 3.3 Test Framework

In this section, the procedure of how the testing of the algorithm should be performed will be explained.

Testing will be a comprehensive and large part of the project. A strategy must be structured to determine what to test, how to test it and in what form the results should be so that an analysis can be performed.

In the data fetching section some kind of data processing tool was mentioned to be required so that one could investigate the amount of errors in the data, and post-process the data if desired. This processing tool should therefore address these errors so that it will be possible to determine if further investigation can be performed with or without some kind of post-processing of the data. Since there should be an option to run the data with or without post-processing, the results should be displayed in a form that enables manual inspection. This process of data handling should therefore be done before the actual testing begins, so that one can better plan how the testing will be conducted.

A benchmarking tool should be developed that keeps track of how well the predictive algorithm is able to make predictions. In the previous section it was described how the profile should always remain equal to or just above the day it is tested against. A tool that keeps track of how often the profile manage this and how accurate the algorithm is able to predict would be necessary results for this investigation. The tool should therefore rate how well the algorithm is able to stay above or equal to the observed day, to determine if the algorithm is successful or not. The benchmarking tool should provide results in numeric and graphical form and the result must be in a format that is easy to analyze.

It will be necessary to automatically make adjustments to the algorithm. This will provide the ability to run numerous tests and provide results more effectively. Therefore a test framework for the algorithm should be developed. The algorithm itself is just a set of statistical operations that will run on the data. The test framework on the other hand should be designed with adjustable parameters to facilitate the algorithm. This way, the algorithm is an independent part of the framework and will allow easier modification of the algorithm and the opportunity to test other algorithms with the same framework. This will allow easier development in the future.

Based on the results provided by the data inspection, the framework should have parameters that can determine what kind of data processing, if any, that should be preformed to the data before it is supplied to the algorithm.

This summarize that the test framework should address these challenges:

- Ability to choose specific parts of the data to be tested
- Guarantee correct chosen type of data processing

- Run the data on the algorithm
- Provide result in formats that can be analyzed

The following figure 3.3 illustrate a overview of the test phase.

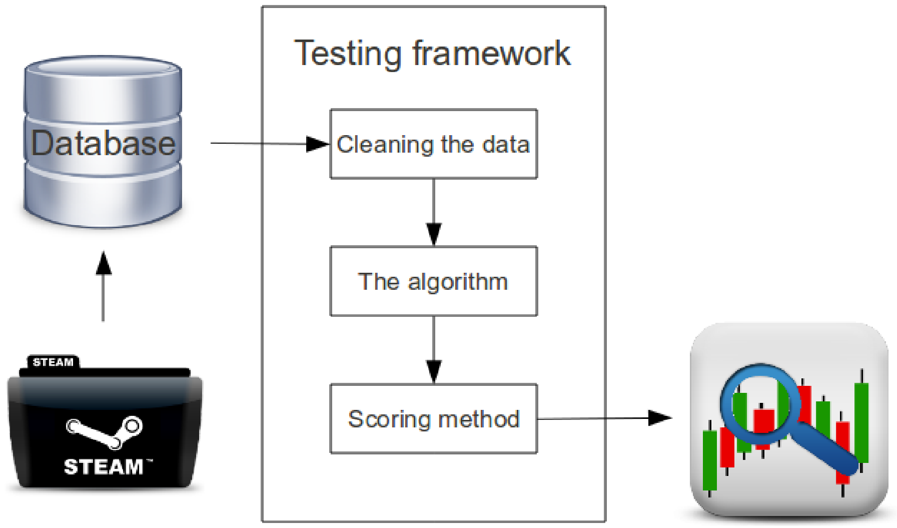


Figure 3.3: Overview on the test framework.

Here we see that data from the steam application is inserted into the database, the test framework query the database for specific data to be tested. The data is then processed and sent to the algorithm. The outcome of the algorithm will then be sent to a scoring method that will provide results that is ready to be analyzed.

### 3.3.1 Algorithm Test Plan

A plan should be structured to keep track of different combinations of days the algorithm should be tested on. This should be done to prevent the results to be unstructured and it would possibly increase the chance to cover the majority of interesting findings. It would also prevent a too long testing phase considering the time constraints. This plan should have a various set of test combinations, both in the number of days and specific days that the profile should be based on. It is also important to specify how many and which days the profile should be tested against.

The following table 3.1 lists the testing scenarios that should be tested on a selection of games:

| The Test Scenarios | | |
|---|---|---|
| ID | Profile learning days | day(s) tested |
| Week | 1 week | The 3 following weeks |
| Weekend | 10 weekends | The 10 following weekends |
| Monday | 5 Mondays | The 20 following Mondays |
| Tuesday | 5 Tuesdays | The 20 following Tuesdays |
| Wednesday | 5 Wednesdays | The 20 following Wednesdays |
| Thursday | 5 Thursdays | The 20 following Thursdays |
| Friday | 5 Fridays | The 20 following Fridays |
| Saturday | 5 Saturdays | The 20 following Saturdays |
| Sunday | 5 Sundays | The 20 following Sundays |
| 40days | 40 days in a row | The following 3 weeks |
| 60days | 60 days in a row | The following 3 weeks |

Table 3.1: Table of purposed test scenarios.

Number of days in the test scenarios are selected based on some assumptions. These assumptions are made based on time constraints and opinions about how large the scenarios should be to establish good results. In statistical theory, it is said that more than or equal to 30 (n> = 30) tests should be made for a sample mean to be considered close to the population mean.[43].

The reason for the variety of test scenarios is the increased chance of finding trends when different combination of training days is used to predict. These scenarios should provide results whether the variety of training day combinations have an influence on the predictability. A profile that is based on a specific day, e.g. Mondays, would determine if the algorithm is able to predict that specific day better if the profile only consist of that same type of day. While a profile based on several consecutive days will determine the predictability results not using specific days. The different variations of training days shown in the table above should provide enough results whether variation has an impact or not.

This plan should be manageable with the given time constraints and should function as a recipe for how the testing should be performed. The idea is to perform this recipe every time a change is suggested to the algorithm and every time a new game is tested. This way, the analysis can build on results that are structured.

## 3.4 The Analysis Structure

The analysis should be structured in a way that:

- give explanations to errors occurred in the data.
- provide reasoning to both bad and good predictions.
- address if the algorithms ability to predict is random or consistent.
- Data mining techniques should map the tendencies in the predictability.

This structure should investigate the predictability in the data and will also map the algorithms ability to predict.

# Chapter 4

# Result

In this chapter the results from the approach will be explained.

## 4.1 Database

First, the data needs to be inserted into the database, SQlite3[51] was used as database in this project. A script was developed to transform the existing data format (as explained in the approach chapter) into a format that is understandable for a database. The development of this script was done using simple Perl code.

The following is an excerpts of the raw data and shows retrieval of the first interval. The first line is the date at retrieval, the following lines show the game name ending with the value of current players.

Raw data format

```
1274109603
Call_of_Duty__Modern_Warfare_2___Multiplayer.value 63106
Counter_Strike__Source.value 58084
Counter_Strike.value 51917
Football_Manager_2010.value 13689
Team_Fortress_2.value 11671
Portal.value 10704
Left_4_Dead_2.value 10168
Call_of_Duty__Modern_Warfare_2.value 5414
Empire__Total_War.value 5345
Condition_Zero.value 5086
```

A script converts the raw data into the format that was purposed in the approach chapter. This format is a comma separated list. When executing the script a new file is created containing the new format. The following show the new format:

New data format

```
1274109603,0,1,Call_of_Duty__Modern_Warfare_2___Multiplayer,63106
1274109603,0,1,Counter_Strike__Source,58084
```

```
1274109603,0,1,Counter_Strike,51917
1274109603,0,1,Football_Manager_2010,13689
1274109603,0,1,Team_Fortress_2,11671
1274109603,0,1,Portal,10704
1274109603,0,1,Left_4_Dead_2,10168
1274109603,0,1,Call_of_Duty__Modern_Warfare_2,5414
1274109603,0,1,Empire__Total_War,5345
1274109603,0,1,Condition_Zero,5086
```

Here we see the same information as in the raw data only in a format understandable for the database. Also year day and week day is added for easier querying later on. A simple script was then developed to insert the data into the database. Both the scrip that creates the new format and the script that insert the data into the database can be found in the Appendix.

## 4.2 Data handling

In the approach chapter it was discussed about possible errors in the data. These errors are in form of gaps of data loss. In this section, the results from the search of these errors will be presented.
The following list is an example of how the data should look like when querying the database for data from a day on one of the games.

Example of desired data format

```
1, 35043
2, 36545
3, 42043
4, 44332
5, 56443
6, 57454
7, 56443
8, 58756
9, 59434
10, 59854
11, 64340
...
287, 24546
288, 23443
```

But as anticipated, there was something wrong in the dataset. Data loss was discovered after investigating the data. The reason for this is most likely that Steams statistics feature has been unreliable. The feature seems to be unavailable at times and therefore cause the fetching script to be unable to retrieve data.

The script that collects data from the Steam application is designed so that it only records the date stamp when the data is collected successfully. Therefore, the data is not visible as holes but as a large gap of data loss at the end of the dataset of a day.

The following is a list of unprocessed data from a day directly from the database, this illustrates an example of data loss:

Unproccessed Data format

```
1, 35043
2, 36545
3, 42043
4, 44332
5, 56443
6, 64340
7, 43534
8, 34330
9,
10,
11,
...
287,
288,
```

The example show that something is wrong, one would expect data all over the 288 lines, but a gap of data loss have occurred at the end of the dataset. In order to do something about this problem, it would be desirable to know where the data loss is located and the length of the loss. The following example demonstrates this.

Example of data format with data loss located

```
1, 35043
2,
3, 42043
4, 44332
5, 56443
6,
7,
8,
9,
10, 59854
11, 64340
...
287, 24546
288, 23443
```

Here we see where the gaps are located and the length. Detection of data loss so that the data is adjusted to the correct time position will be absolutely necessary to get accurate results and to do some kind of correction to the gaps of data.

A gap analysis was therefore conducted to locate the holes and to figure out what to do with them.

### 4.2.1 Gap Analysis

The following output retrieves data from one day from the database:

Database query

```
sqlite3 gamebasev2.sqlite "select date,count from games where game =
'Counter_Strike'"
```

This query was used to extract data in the first tests. As explained in the previous section this led to major data loss at the end of the data list. In order fix this problem and locate where data loss occurred a script was developed.

The gap location script retrieves date and data from a game. Then makes calculations on the difference between the dates to see if there is a bigger gap than it should be. For example, if the different between two date stamps are longer than 5 minutes but less then 10 minutes, this indicates that loss of one data retrial have occurred. Since the date is known, one can tell when the loss have occurred. The script can this way calculate how big the gap is and where it is located.

The following example query the database for date and data value from the game Counter Strike Source, and illustrate how the gap of data loss is located.

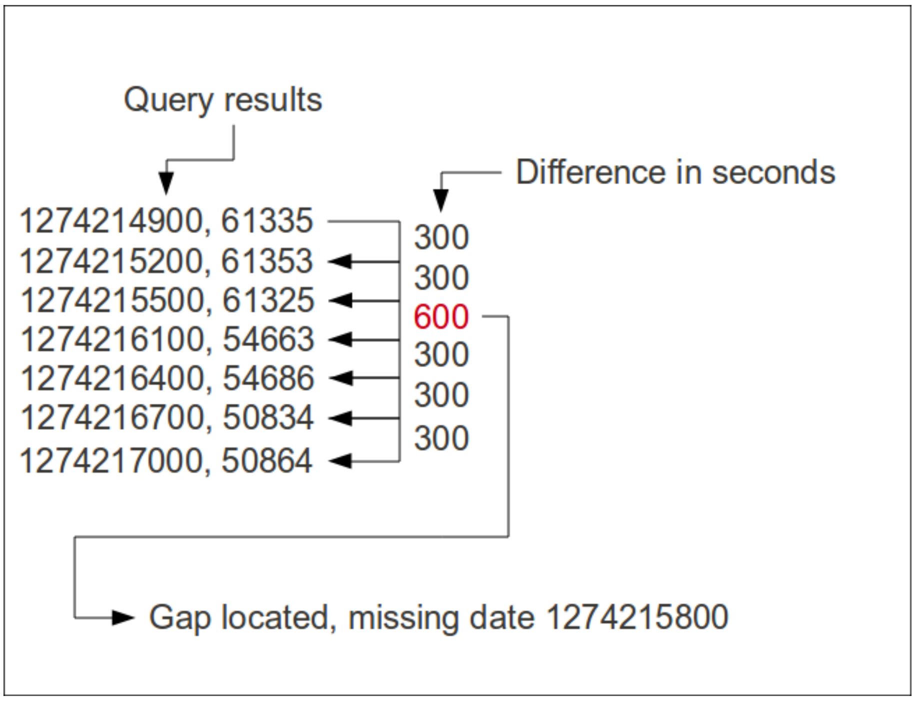


Figure 4.1: Illustrative explanation of the Gap locating tool

As we can see in figure 4.1 each line from the query is inspected by the gap location tool. If the tool finds anomalies in the difference between each line of data, and that this difference is longer then 300 seconds. This raises an alert and the gap is filled with the date stamp that is missing. This procedure will fix the data by locating the errors, but it will not fill the gap.

A gap analysis script was created to visually see the distribution of gap size. Not surprisingly, it turned out that the results were similar for all games. This supports the theory that Steam's statistics feature has been unavailable at times and therefore similar gaps have occurred on all the games in the database.

The following is a distribution of the data gap distance from the game Counter Strike.

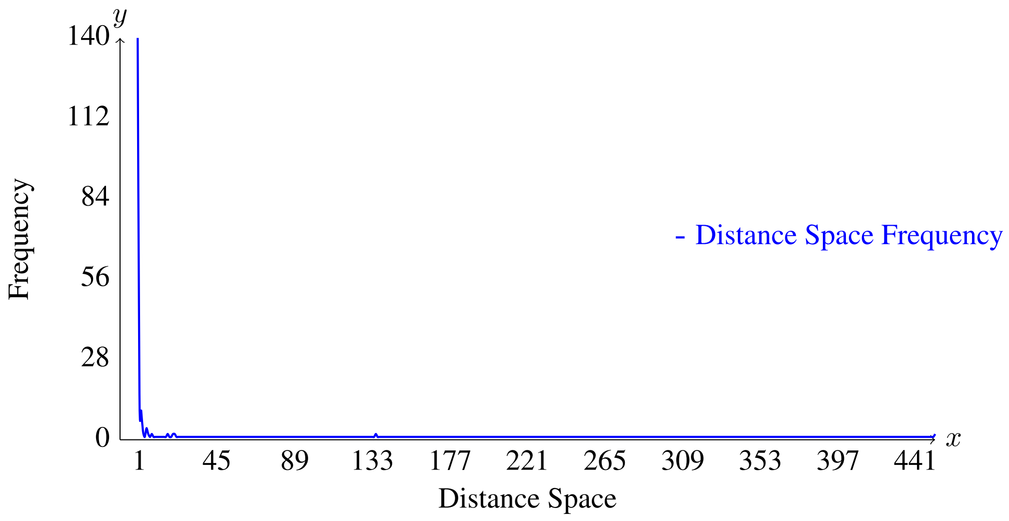


Figure 4.2: Distance Space Frequency in the game Counter Strike

The figure 4.2 show that there are definitely more small gaps than large in the dataset. This is positive because it is easier to fix small gaps than large ones. Small gaps can be sealed by means of statistics while large must likely be transplanted from other days of data.

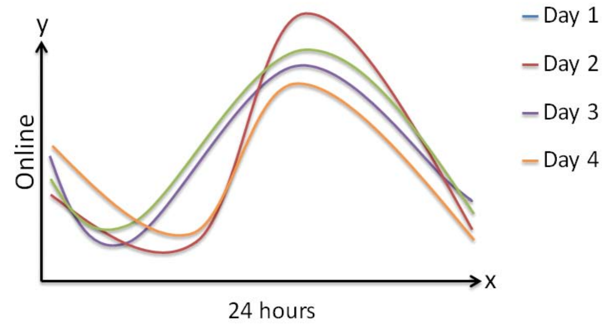
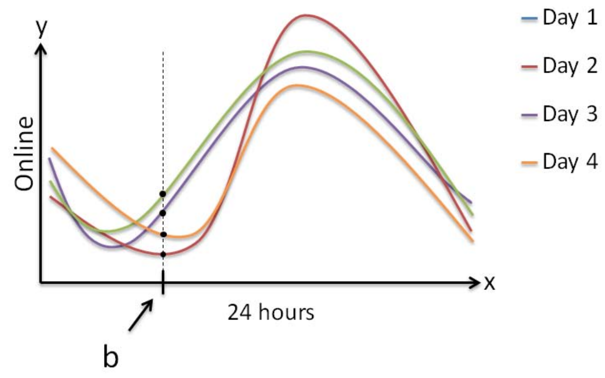
In the previous chapter it was suggested that it should be an option to seal the gaps of data loss before the data is fed to the algorithm. Therefore a tool was created to enable sealing of data loss. Since the distribution showed that the majority of gap length was 1, a decision was made to only enable sealing when gap length is 3 or less. Sealing small holes in the dataset can be done in a relatively simple and effective way. The script locates the small gaps and seal these by using a linear mathematical method.

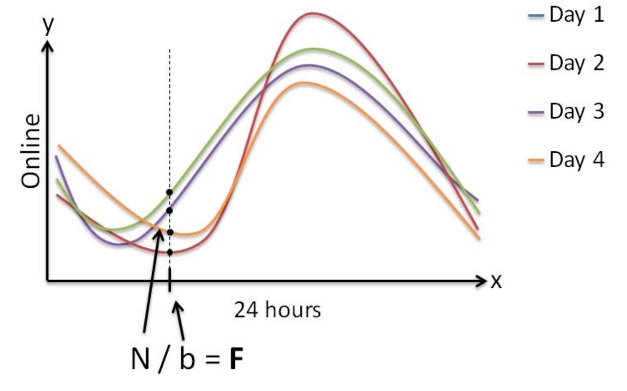
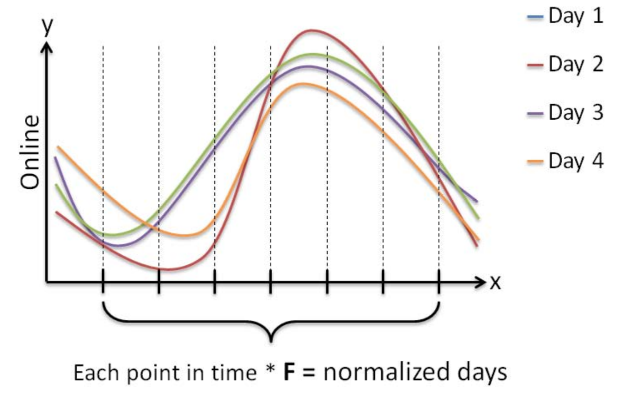
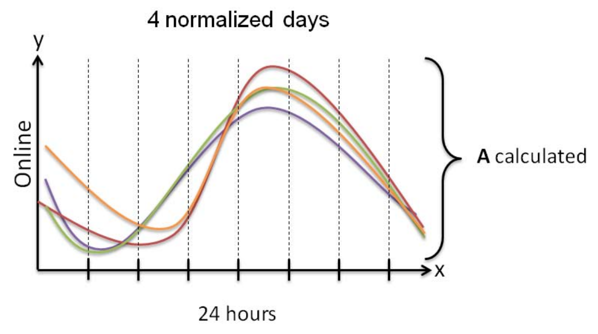
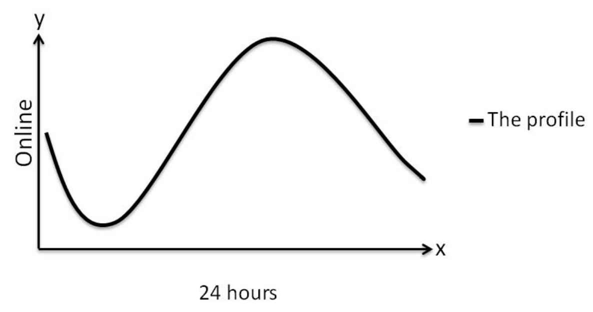
Both the gap locating and gap sealing scripts was created using Perl and is available in the Appendix.

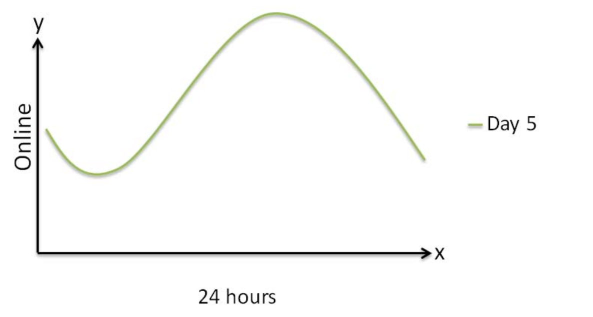
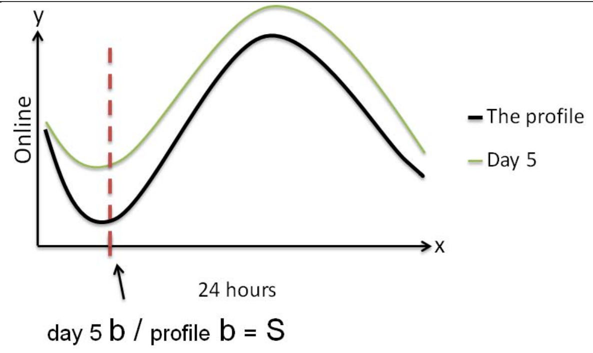
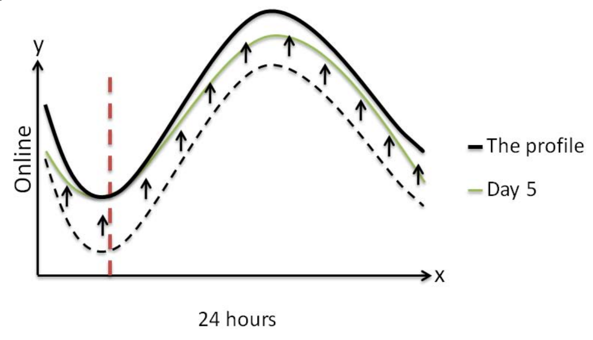
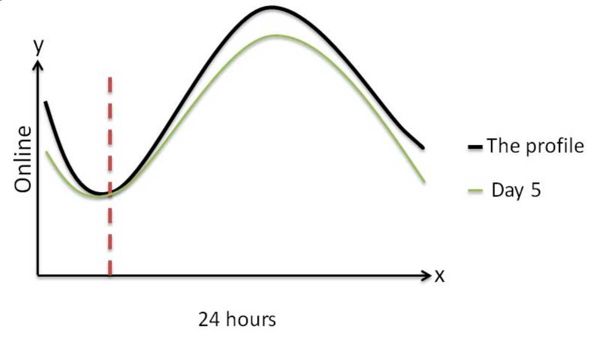
## 4.3 The Predictive Algorithm

Based on the algorithm outline from the approach chapter the developed predictive algorithm will be explained in this section.

The idea behind the design is to use mathematical calculations on existing repetitive data to predict the future. The following example will be used to explain how the algorithm has been developed. Data from four days will form a profile, this profile will then be used to predict a subsequent day.

| | |
|---|---|
|  | 1: Four days is chosen as training days that will form a profile. |
|  | 2: The variable *beta* signifies a single point in time, future predictions will be made based on data from this particular point in the next phase. The entire profile will therefore be generated around this point. Variable *N* is a percent percentile of all values located at point *beta* from all four days. |

| | |
|---|---|
|   y<br>Online<br>x<br>24 hours<br>N / b = **F**<br>Day 1<br>Day 2<br>Day 3<br>Day 4 | 3: The variable *F* is defined by dividing *N* to the value located at *beta* for each day. |
|   y<br>Online<br>x<br>Each point in time * **F** = normalized days<br>Day 1<br>Day 2<br>Day 3<br>Day 4 | 4: The days can now be normalized by multiplying each point in time on each day with *F*. |
|   4 normalized days<br>y<br>Online<br>x<br>24 hours<br>**A** calculated | 5: After the normalization, for each point in time a percent percentile is calculated on the values between each day. These values will shape the profile. |
|  | 6: The profile is formed based on the operations above. |

| | |
|---|---|
|  | 7: The profile can now be tested against a observed day. |
|  | 8: Now, the profile must be normalized to the observed day. Variable *S* is defined by dividing the *beta* value on the subsequent day with the *beta* value on the profile. |
|  | 9: The profile will be normalized according to the observed day by multiplying each value in the profile by *S*. |
|  | 10: The profile is normalized and should be able to predict equal to or just above the observed day. |

## 4.4 Developing a benchmarking tool

In this section the benchmarking tool will be explained. In the previous chapter, a need to rate the performance of the algorithm was mentioned. Therefore, a tool is developed to:

- rate how often the profile remains equal to or above the observed day
- calculate the average distance between the profile and the day
- display the algorithm in a graphical format

A score tool was developed to simply calculate how often the profile have a equal or higher value than the observed day at each point in time. Every time the profile have a equal or higher value, the tool provides one score point. There are 288 points in a day, so a score of 288 means that the profile was able to stay above the day all the time, while a score of 0 means that the profile was under the observed day the whole time.

The score tool also calculate how close the profile is to the observed day. It does this by calculating the distance between the profile and the observed day for each point in time. The average distance is calculated and displayed as results.

### 4.4.1 Developing the plotting tool

To examine how the algorithm behaves, a plotting tool was developed. This tool uses Tikz which is a drawing tool for latex. By combining Tikz and Perl programming, this plotting tool has become customized for this project and has made it possible to plot quickly and efficiently.

This tool was designed so it was able to provide the results in a visual form. Tikz works as a drawing program, the biggest difference is that one have to use programming instead of a cursor to draw. The plots that where created by this tool will be shown in the next chapter when the results from the testing will be presented. Later in this chapter an explanation of the design will also be described. To explain how Tikz programming work the following example will show the code to draw X and Y lines.

```
\begin{tikzpicture}[x=1pt,y=1pt]
\draw[->] (0,0) -- (50,0) node[right] {$x$};
\draw[->] (0,0) -- (0,50) node[above] {$y$};
\end{tikzpicture}
```

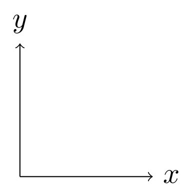


Figure 4.3: Example of drawing X and Y lines using Tikz.

This example also highlight the difficulty differences between using the cursor and Tikz programming for drawing.

The plotting tool makes it possible to plot every beta value on a day. The approach section described how the testing should be performed using different scenarios. With this plotting tool one is able to do a deep examination on all the days that the profile is tested against. For example, If the algorithm is tested against 20 days, a plot can be created on every 288 beta values for all the 20 days. This is 288 * 20 = 5760 plots, and that is just from one scenario. This plotting tool makes it possible to manually inspect how the algorithm performs in each day. Although this manual inspection process takes a huge amount of time, it provides a very unique overview of the algorithms ability to predict.

## 4.5 The test framework script

The approach suggested a test framework that would automate the process of testing. This section will describe the test framework script.

The test framework script is a collection of the many tools developed in this project and handles the entire process of testing. The script accept parameters that chose:

- -e | the option to seal data loss.
- -T | the number of training days.
- -D | the number of testing days.
- -w | to only test on weekends.
- -n | to only test on weekdays.
- -0 (0 - 6) | to only test on specific days.
- -A | the percent percentile used to create the profile.
- -t | the topic of the test.

- -m | the name of the directory the test results are to be placed.

The following is a example of an execution on the test framework script.

Test framework script execution example

```
./TestFramework.pl -A 85 -O 2 -T 5 -D 20 -e -m Counter_Strike_TuesdayGapSealed
-t                              "Tuesday Gap Sealed                              "
```

This execution tests the data on the algorithm with the following parameters. -A 85 specifies that the percentile used is 85%. -0 2 specifies that only Tuesdays are to be tested. 0 is Sundays, 6 is Saturdays. -T 5 specific that the profile is created by using 5 days, 5 Tuesdays in this example. -D 20 specifies that the profile should be tested on the following 20 Tuesdays. -e specifies that data gaps is sealed. -m specifies the name of the directory that all results will be placed and -t specifies the topic of the test.

## 4.6 The algorithm test results

Testing has been done as described in the algorithm test plan section. Not all the scenarios mentioned have been tested, because the extent of each scenario has greater than initially anticipated. A choice was made to go deeper into certain selected scenarios instead of scratching the surface of all. This decision was made based on two factors.

- An interest to perform a deep inspection of scenarios, using the graphical results provided by the plotting tool. It would take to much time to do this on all suggested scenarios in the approach section.
- Several of the scenarios tested indicated that the majority of important findings have been found so the necessity to continue the testing became smaller.

The results shown in the following chapters are in graphical form. In order to better understand these, an explanation of the various important elements will be illustrated in the following figure.

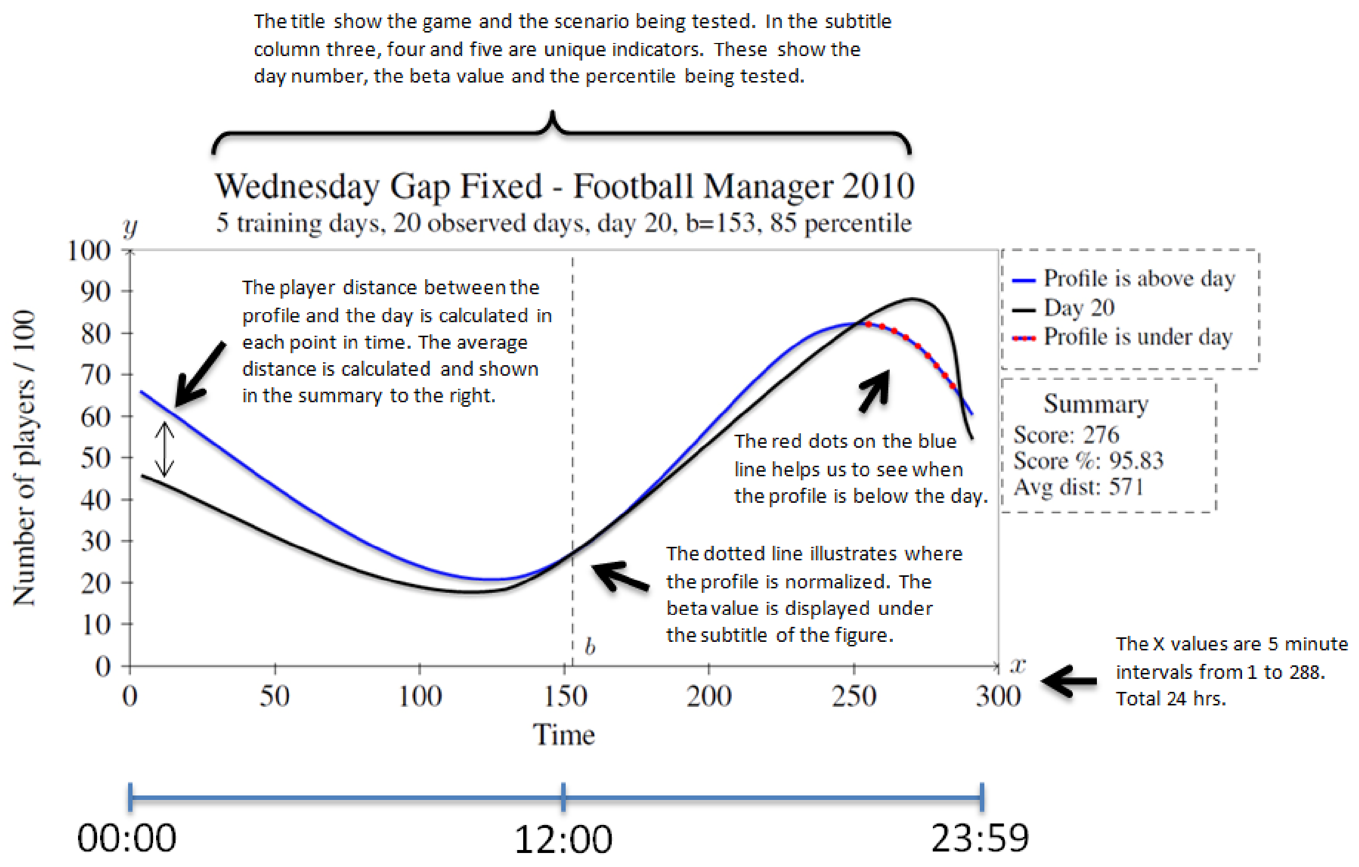


Figure 4.4: Explanation of the graphical result format

In addition to the explanations in figure 4.4 some information will be further highlighted to avoid any confusion. The summery box on the right side shows the score, the score in percentage and the average distance. The blue time-line is not located on the graphs, this is just an indicator to help understand the time period on the X axis.

# Chapter 5

# Game specific results

In this chapter the algorithm will be tested on the post-processed game data. Three games have been chosen to be tested, these are:

- Counter Strike Source
- Football Manager 2010
- Supreme Commander 2

These games has not been chosen randomly. They are of entirely different game type and should therefore also have some difference in the human behavior. An introduction to the various games will be made under each section and there will also be an explanation to the scenario tested.

## 5.1 Results: Counter Strike Source

### 5.1.1 Introduction

In this section results from the game Counter Strike Source will be presented. The algorithm has been tested on the game data for some of the scenarios described in the algorithm test plan section.

Counter Strike Source is a first-person shooter game where players frequently can log on to play whenever it suits them. You choose a server from the Steam application and log on from there. Data from such a type of game can be thought to be more difficult to predict than games were one are “bound“ to play for a certain period of time. This is because one has less control over how long the players are online, and when they play.

### 5.1.2 Scenario: Week

In this section the first scenario will be tested. The algorithm will have a learning period of a week and will be tested against the three following weeks. Seven consecutive days will therefore be tested against twenty-one continuous days.

The following figure 5.1 illustrates the frequency distribution of the best *beta* for every observed days. The best is a score over 260 with the lowest average distance.

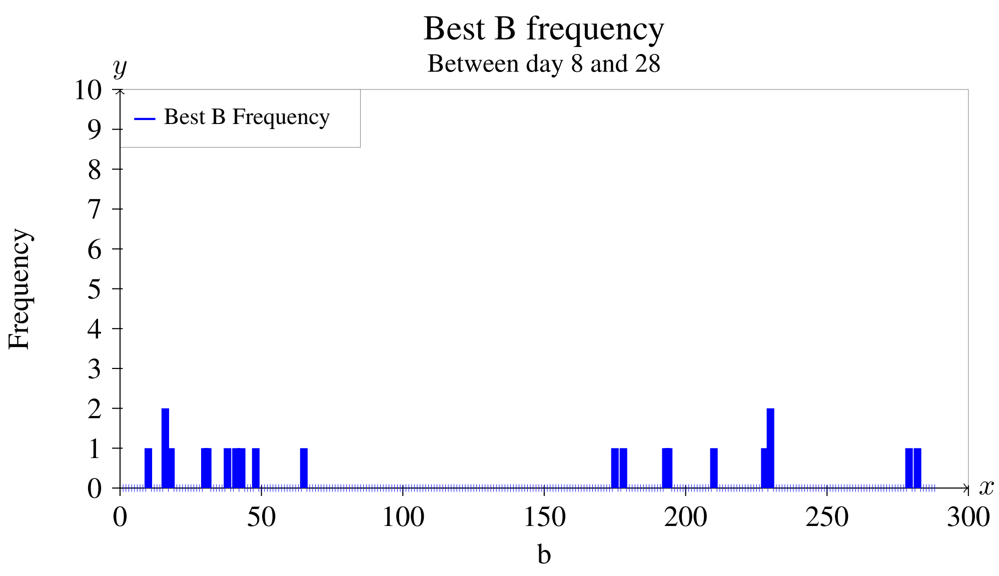


Figure 5.1: Frequency distribution of the best *beta* values from all days in scenario: Week, in game Counter Strike Source

One can see here that there is no clearly superior *beta* values. However, *beta* values of 16 and 230 have occurred as the best value twice. It also seems that the best *beta* values occur in groups. One can see two groups in the distribution where the best *beta* values occur close to each other. The first group occurs at *beta* values between 10 and 48, this seems to be the largest group of most *beta* values close together. The second group occurs at*beta* values between 175 and 279, these are a little more spread out than the first group.

There are long periods in the distribution where there is no best values. This may indicate that it is hard to predict in these periods.

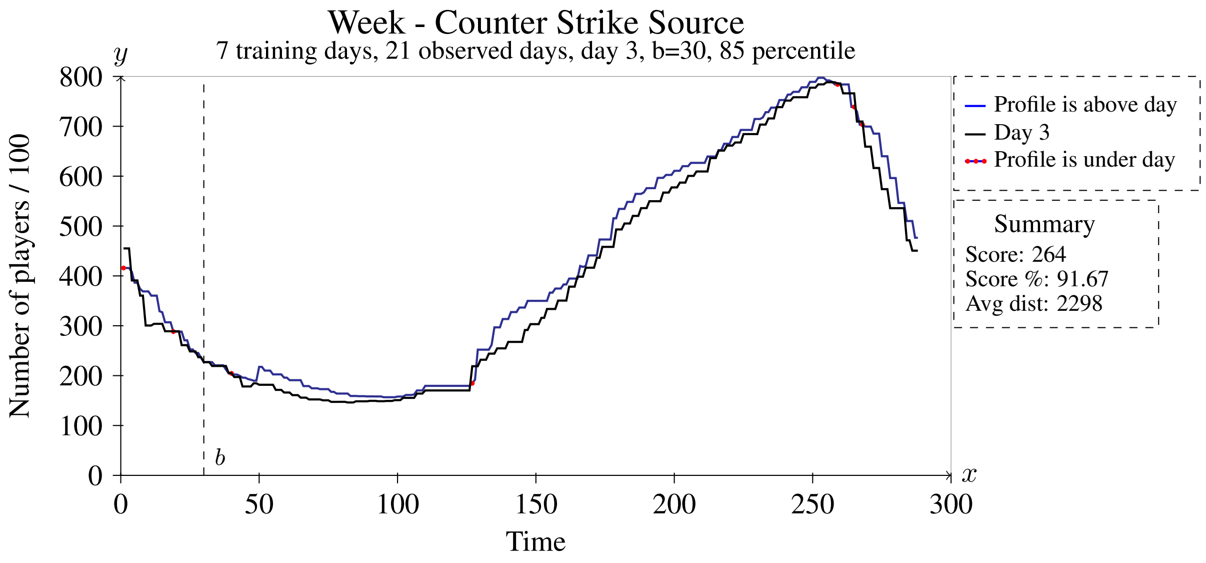


Figure 5.2: Results from the game Counter Strike Source of scenario: Week, using *beta* value 30 on day 3

The next figure 5.2 shows an instance in the first group from the distribution. Here one can see that the profile has predicted the observed day very well. This day has a distance of three days from the last day the profile is based on. A score of 264 means that the profile has remained above the observed day 91.67% of the time. With a very low average distance of 2298 would indicate that the predicted profile is very similar to the observed day. This is also very clearly presented by observing the graph. In the few occasions the profile score lower than the observed day one can see that the value is just slightly below the observed day. This is a accurate predication.

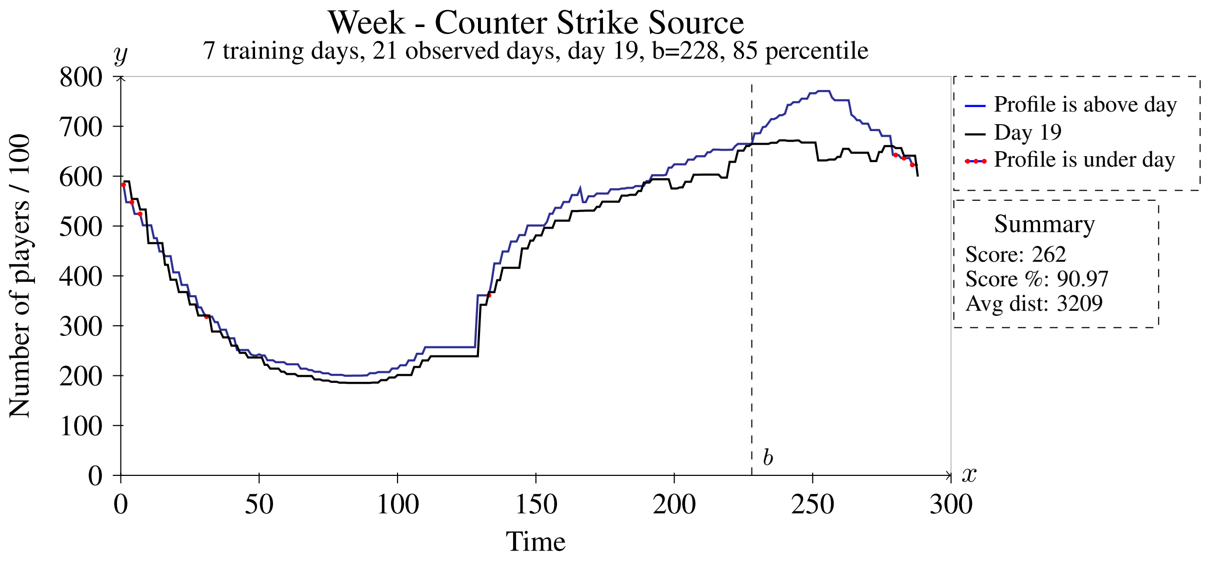


Figure 5.3: Results from the game Counter Strike Source of scenario: Week, using *beta* value 228 on day 19

Figure 5.3 shows an instance from the second group where the *beta* values were more spread. Here we also see that there's been an accurate predication where the profile has managed to remain close to and above the observed day. This day has a distance of nineteen days from the last day the profile is based on, so this is a longer distance than the previous example. With a score of 262 means that this profile has been able to predict with a value above or equal to the observed day 90.97 % of the time. The average distance between the profile and the observed day is at 3209 which indicates that the predicted profile is quite similar to the observed day. This is also shown clearly by observing the graph.

The two previous figures show two occurrences where the algorithm has been able to predict days that are of long and short distance away from the seven training days.

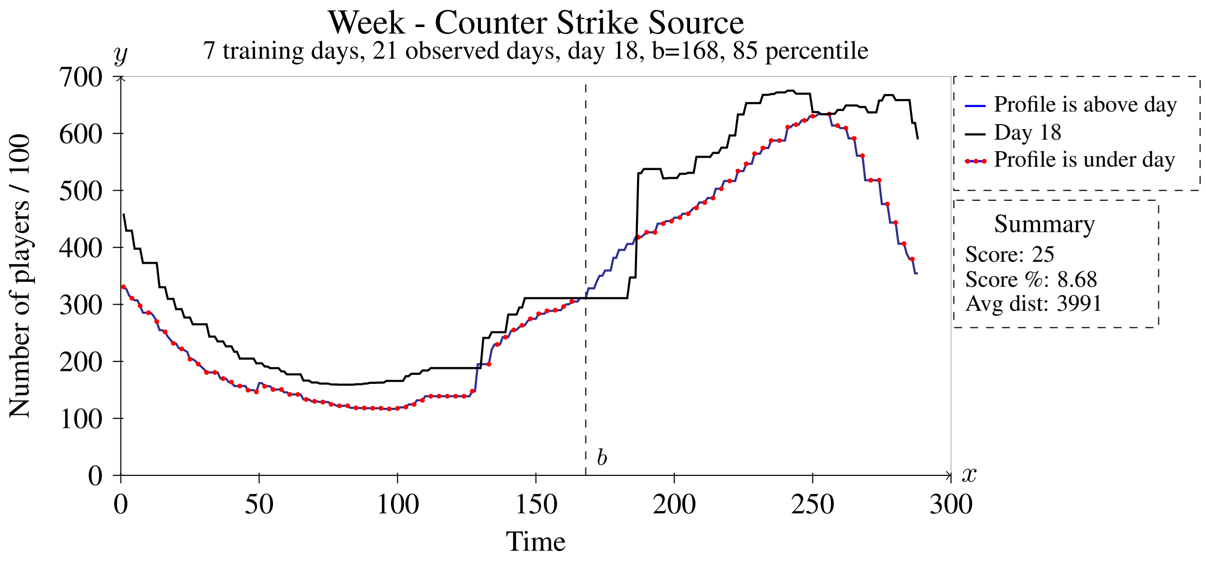


Figure 5.4: Results from the game Counter Strike Source of scenario: Week, using *beta* value 168 on day 18

Figure 5.4 shows an example from the long period when there was no best *beta* values. The figure illustrates how the profile was unable to predict the observed day. We see here that the profile generally stays below the observed day, just in a period right after the *beta* value, we see that the profile remains over. Although the profile scores as low as 25 and is therefore only over the observed day 8.68 % of the time, the average distance is as low as 3991. This is because it manages to remain close even though it is below most of the time. At the same time, we may also wonder about the strange human behavior in the period just before and after the *beta* value. It may seem like it's the same value over a period of time, although one might anticipate that the value should increase during this period. If the values had increased as one might have anticipated the profile would probably have risen to a level above the observed day, but this is only an assumption.

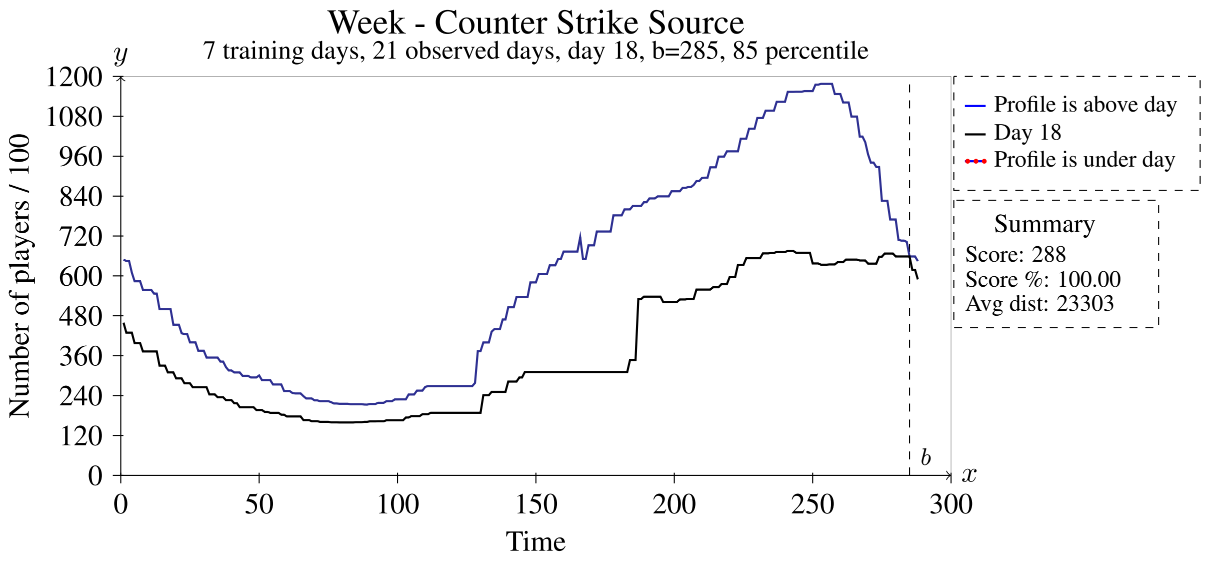


Figure 5.5: Results from the game Counter Strike Source of scenario: Week, using *beta* value 285 on day 18

The figure 5.5 below illustrates an example where the profile always manage to remain above the observed day. With a score on 288 the profile are able to keep above the observed days 100 % of the time. But as one will notice by observing the graph the profile is far above the observed day. An average distance of 23303 implies that the distance between the profile and the observed day is large, this is also confirmed by observing the graph. In the course against the current prime time, we see that the profile remains far above. In the period where *beta* is between 140 and 250 the number of players is increasing and reaches a peak point around *beta* 250. During this period, we see that the distance between the profile and the observed day is largest.

## Summary

In the first figure 5.1 in which the frequency distribution of the best *beta* for each day was shown, it was interesting so see that the distribution was divided into two groups. These results were also evident in the huge job of going through the large amounts of graphs. In this scenario, we see a trend of a much lower performance of the algorithm when it is based on a *beta* value in the period between the two best groups. This may indicate that it is harder to predict in this period. It was also interesting to see how accurate the algorithm managed to predict the observed days as illustrated in the two figures 5.3 and 5.2

In figure 5.4 it was interesting to see how the algorithm predicts completely inaccurate during the period when it may indicate that the value has been outdated over a period of time. This could be a bug in the Steam application where the statistics is not updated at all times. What happens is that the profile is then normalized by a factor which is of a lower value than anticipated. We see that the curve is drawn lower than expected because of this.

### 5.1.3 Scenario: Wednesdays

In this section the results from the testing on scenario Wednesday will be presented. The algorithm had a learning period of 5 consecutively Wednesdays, these days formed the profile that will be tested against 20 continuous Wednesdays. This test is performed to see if there is a pattern to get better results by choosing specific days of the week. and to find unique results that can influence the ability to develop an even better algorithm.

The following figure 5.6 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

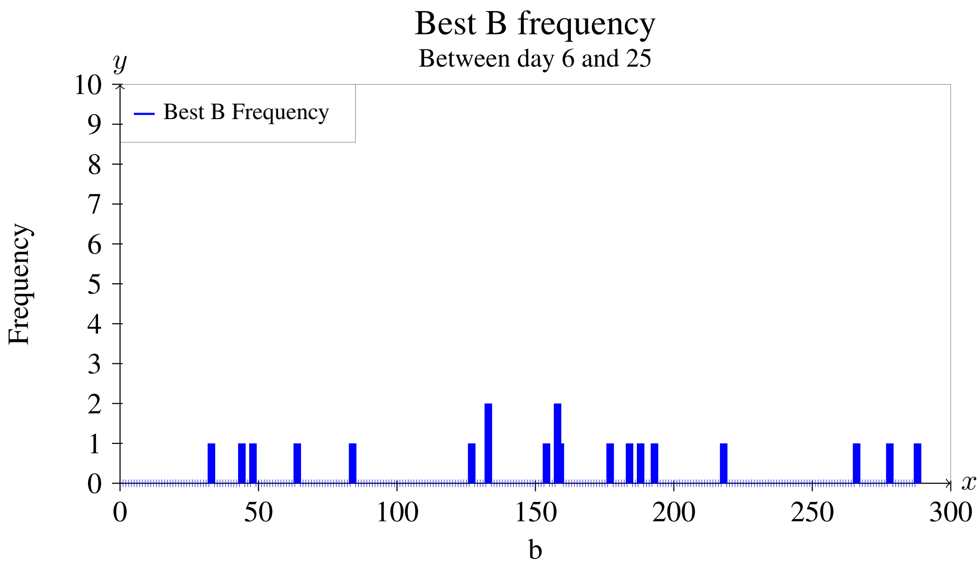


Figure 5.6: Frequency distribution of the best *beta* values from all days in scenario: Wednesday, in game Counter Strike Source

In the frequency distance we see that occurrences are fairly spread out but that there are two occurrences that have two frequencies and these are quite close to each other. These two are located in the area where an escalation of the players tend to happen.

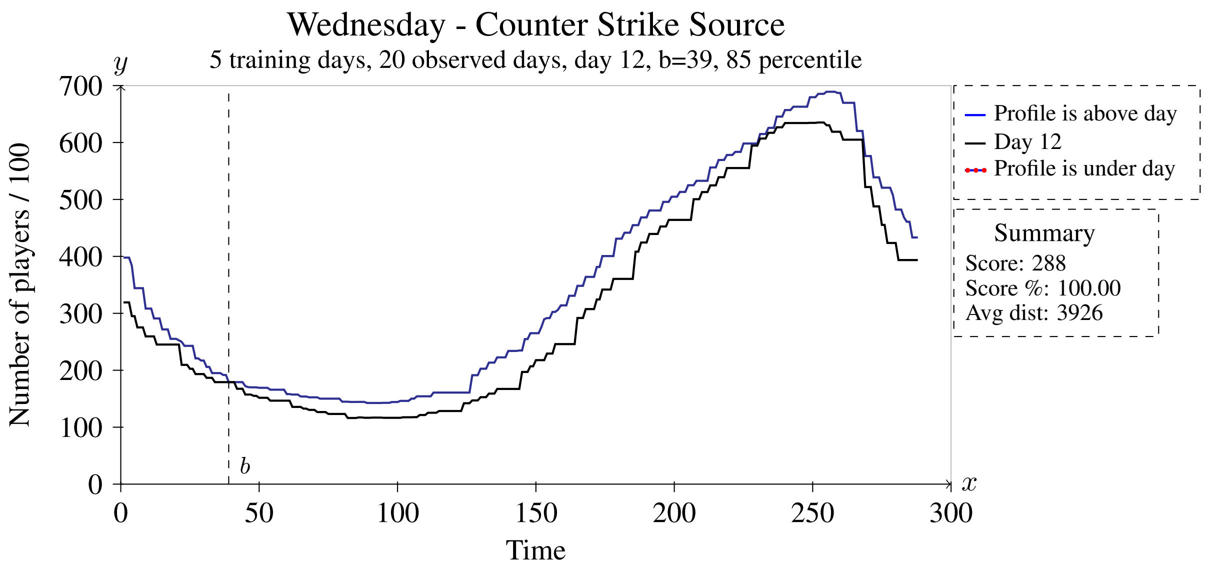


Figure 5.7: Results from the game Counter Strike Source of scenario: Wednesday, using *beta* value 39 on day 12

This first figure 5.7 illustrates the lowest average distance with a score over 90%. We see that the profile have managed to stay above the observed day 268 times or 93.06% of the time and have a average distance of 1963. Counter Strike Source have a huge amount of players playing every day compared to most of the other games. So as we can see on the graph it indicates that an average distance of 1963 reflects a fairly good predictability.

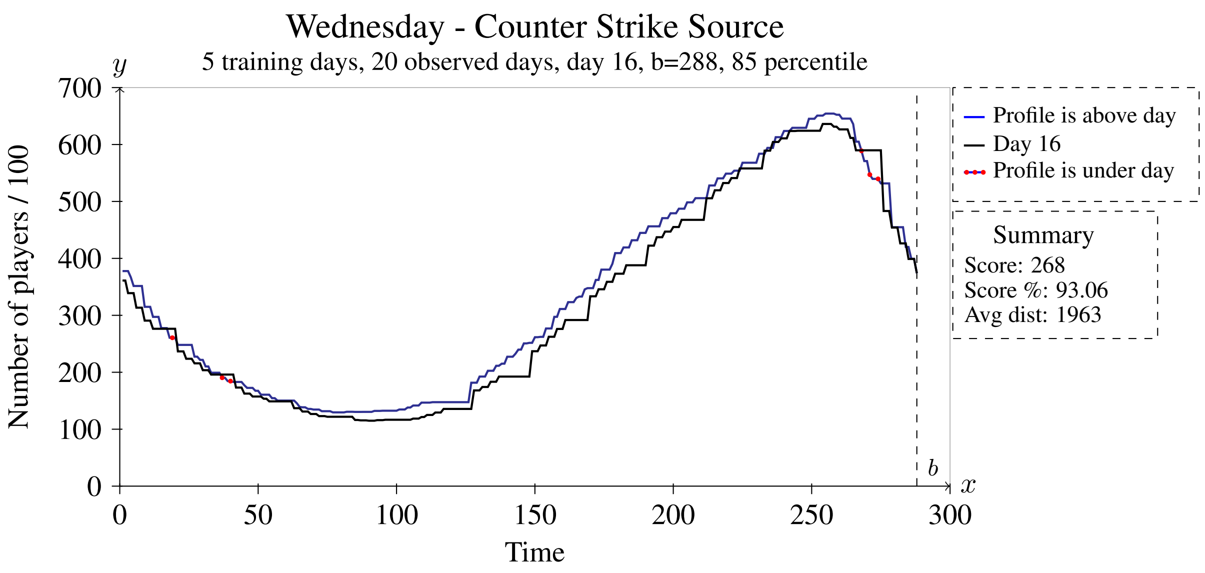


Figure 5.8: Results from the game Counter Strike Source of scenario: Wednesday, using *beta* value 288 on day 16

This next figure 5.8 also illustrate a good prediction. But using a different *beta* value. This example illustrate a instance where the profile have managed to stay above the observed day 100% of the time. And with a average distance of 3926 also managed to predict fairly close to the observed day as well.

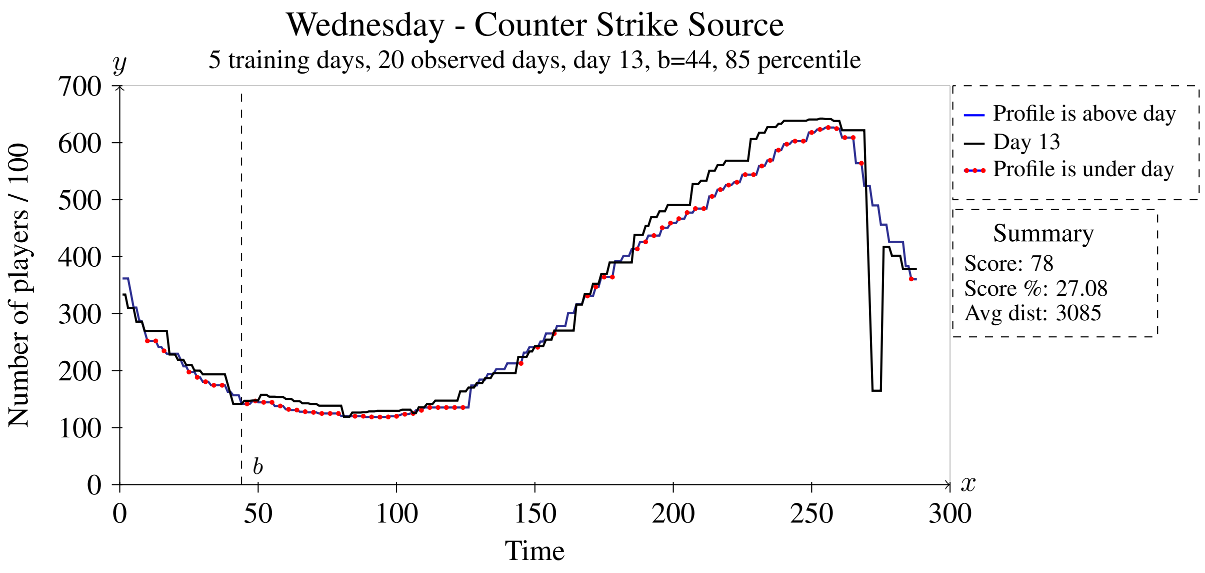


Figure 5.9: Results from the game Counter Strike Source of scenario: Wednesday, using *beta* value 44 on day 13

Figure 5.9 shows that it seems that the observed day has small drops of player loss throughout the day. At the point where the *beta* value is placed, we see one of these dumps of player loss. When the profile is normalized with this value, this results in the profile is unable to remain above the observed day much of the time. We see that the profile has only managed to remain above 27.08 % of the time. We also notice that the few times the profile remain above the observed day is during these drops.

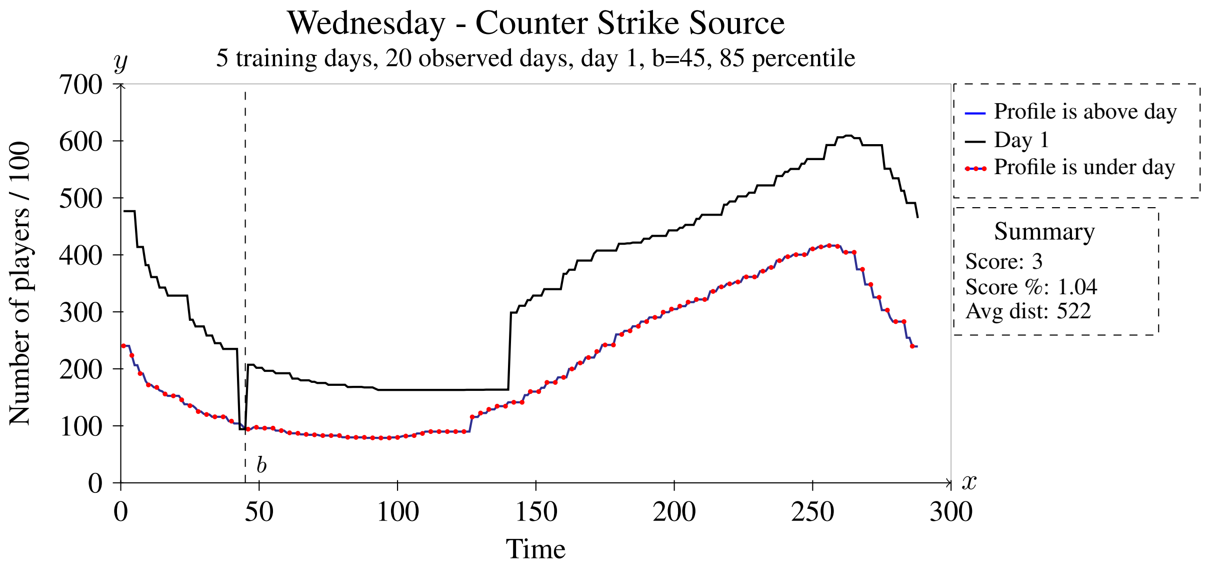


Figure 5.10: Results from the game Counter Strike Source of scenario: Wednesday, using *beta* value 45 on day 1

In figure 5.10 we see that the observed day is predicted wrong when a large loss of players or data error have occurred. This behavior has been observed several times and does not seem to be unique. The problem happens when the profile is normalized by a *beta* with a lower value than anticipated. This will result in a bad prediction the rest of the day. This drastic player loss probably happens when a update is performed on the game or the steam application. If this drop in players had not occurred one can assume that the profile would have predicted closer to the observed day.

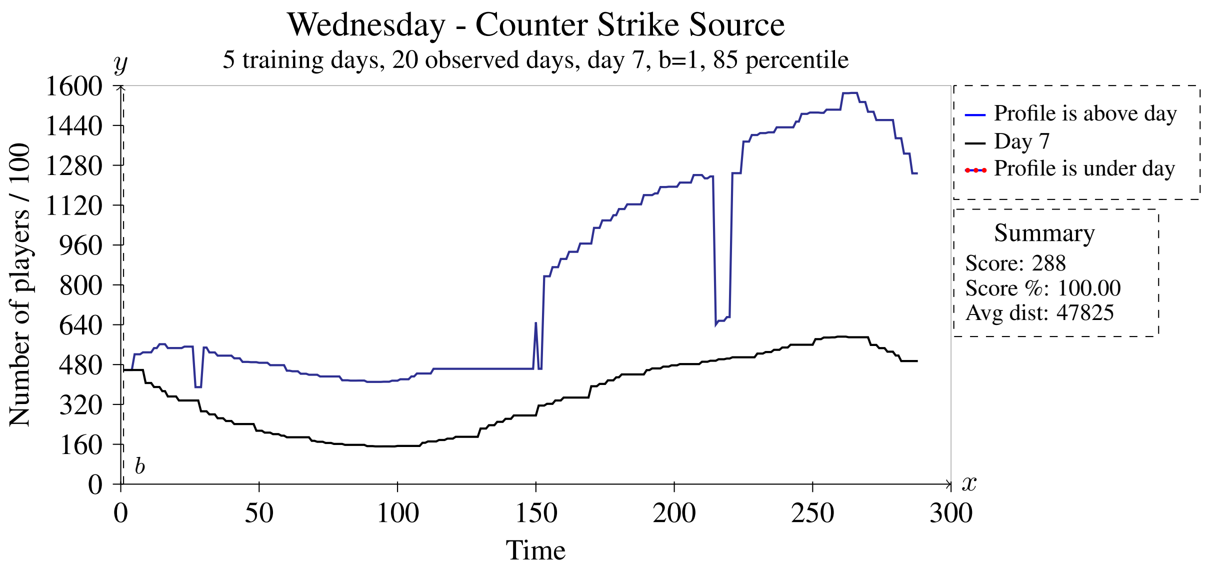


Figure 5.11: Results from the game Counter Strike Source of scenario: Wednesday, using *beta* value 1 on day 7

In this last figure 5.11 we see that the profile totally transform to a curve that does not correspond to any of the above figures. This phenomenon occur when the profile is normalized with *beta* values from range 1 to about 15.

### Summary

It was interesting to find out that the algorithm managed to predict the day 12 by normalizing the profile of almost all *beta* values with good results. This may indicate that day 12 did not have so many data errors or that the human behavior was more similar to those that the profile is based on.

When drastic player loss occur, it is interesting to see how inaccurate the prediction becomes.

### 5.1.4 Scenario: Saturdays

In this section the results from the scenario, Saturdays, will be presented. The algorithm have a learning period of 5 Saturdays and will be tested against the following 20 Saturdays. One will in this scenario be able to see if one can get better results from predicting the human gaming behavior by testing the algorithm with a specific day of the week.

The following figure 5.12 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

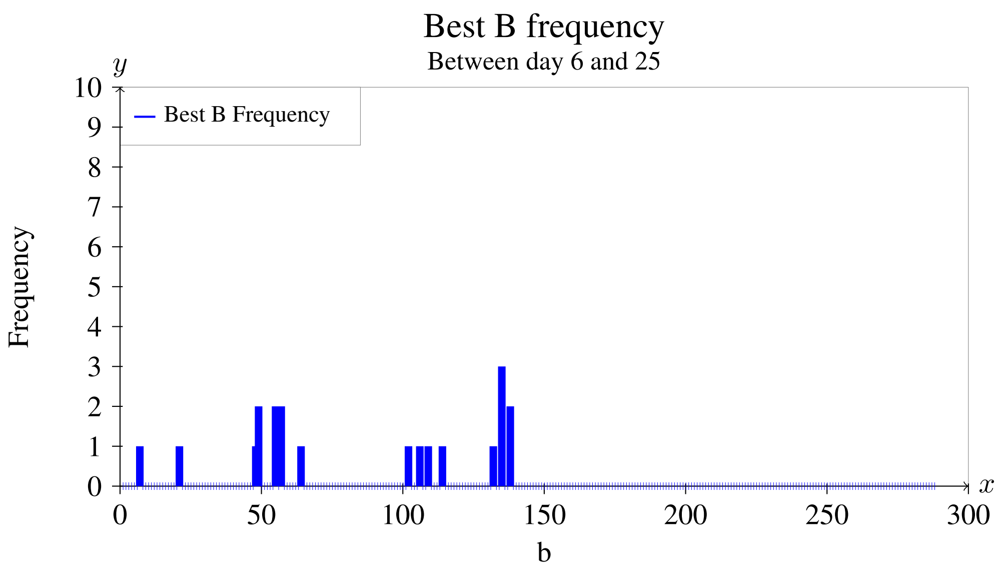


Figure 5.12: Frequency distribution of the best *beta* values from all days in scenario: Saturday, in game Counter Strike Source

In this scenario we see that the distribution is gathered on one side of the distribution. Its should indicate that the algorithm get the best results by normalizing the profile with *beta* values in range from 1 to about 140. We notice that there are several days that have the same *beta* value as the best value. These are stacked in two groups, one at about *beta* 50 and the other one at about *beta* 140. It seems like there are three groups with best *beta* values. These groups have values that are located very close to each other.

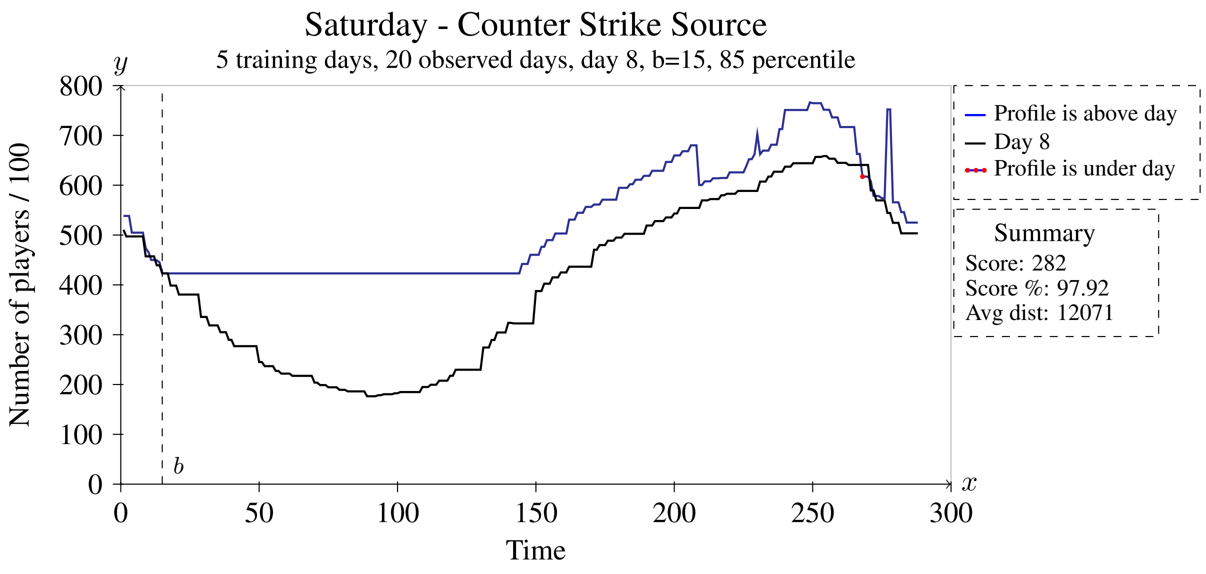


Figure 5.13: Results from the game Counter Strike Source of scenario: Saturday, using *beta* value 15 on day 8

In this first figure 5.13 we immediately see an abnormal shape of the profile. In the period between the *beta* value and about 150 then number of players is constant. This suggests that one or more of the Saturdays the profile is created of have data errors. This affects the predictions. Although the profile is able to remain above that observed day most of the time the distance between the the profile and the day is larger than it should be.

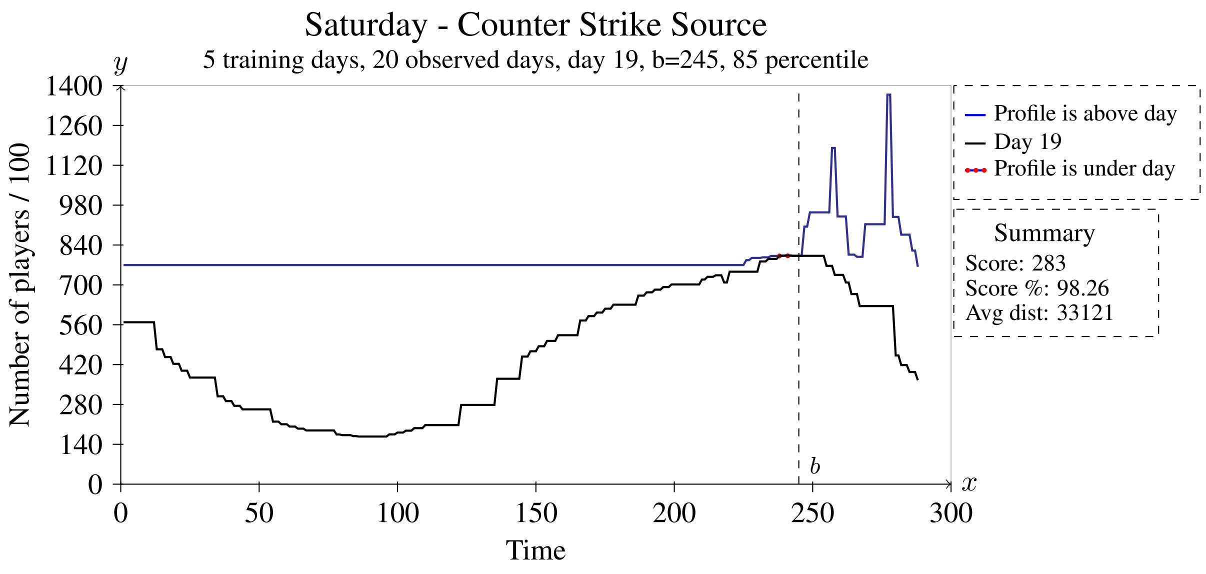


Figure 5.14: Results from the game Counter Strike Source of scenario: Saturday, using *beta* value 245 on day 19

This next figure 5.14 illustrate the results when the profile is normalized with a *beta* value at the peak of number of players. We see here that the phenomenon from the previous figure is still there. In this scenario, it seems like the higher the number of players become the longer the error line in the profile also become. The profile is above the observed day most of the time, but a average distance of 33121 makes the predictions very imprecise. This emphasizes the importance of correct data for the algorithm to work properly.

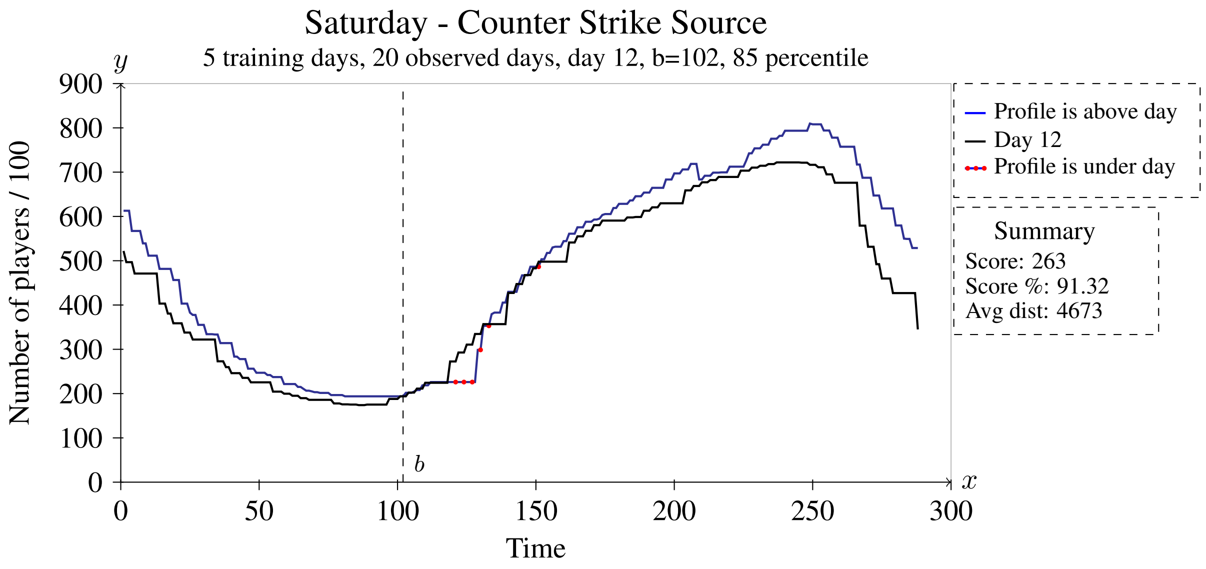


Figure 5.15: Results from the game Counter Strike Source of scenario: Saturday, using *beta* value 102 on day 12

Figure 5.15 shows an example where the profile is normalized with a *beta* value where the number of players value is low. Here we notice that the profile is completely different and actually predicts the observed very well. Although we already know about the errors in the profile, so we see here that in some cases, the algorithm still are able to create a profile with good results. In this example the profile remain above the observed day 263 times out of 288, this is 91.32% of the time. With a average distance 4673 we can see on the graph that the profile have remained close to the observed day most of the time.

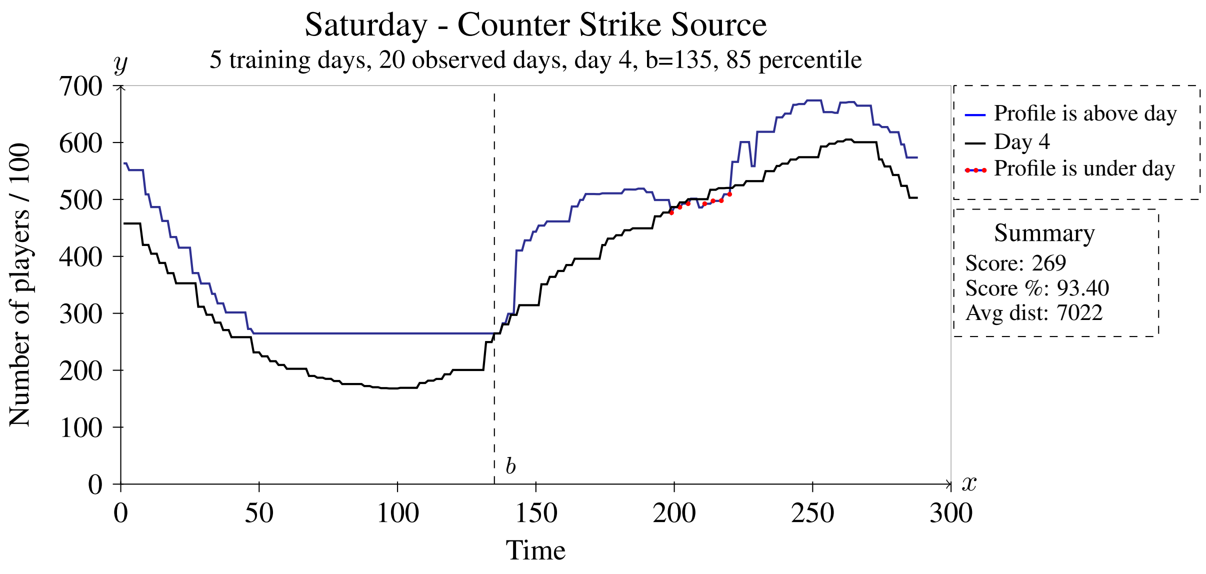


Figure 5.16: Results from the game Counter Strike Source of scenario: Saturday, using *beta* value 135 on day 4

The last figure 5.16 in this scenario illustrates the best *beta* values that occurred in three of the days that was tested. Here we see that it seems that the error in the profile does not make a big impact on the predictions results.

## Summary

It's a very interesting finding to see how different the profile behaves when provided incorrect data. It is incredibly important to create the profile based on a correct data flow of already existing data.

## 5.2 Results: Football Manager 2010

### 5.2.1 Introduction

In this section results from the game Football Manager 2010 will be presented. The algorithm has been tested on the game data for all scenarios as described in the algorithm test plan section.

Football Manager 2010 is a football manager simulation game where you as a player act as a manager for a team. This is a type of game that does not require full attention all the time. This is a type of game that one can have running on the computer while doing other activities simultaneously. The human behavior should therefor behave somewhat different than in for example Counter Strike Source which has a more unforeseen human behavior.

### 5.2.2 Scenario: Week

In this section the first scenario will be tested. The algorithm will have a learning period of a week and will be tested against the three following weeks. Seven consecutive days will therefore be tested against twenty-one continuous days.

The following figure 5.17 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

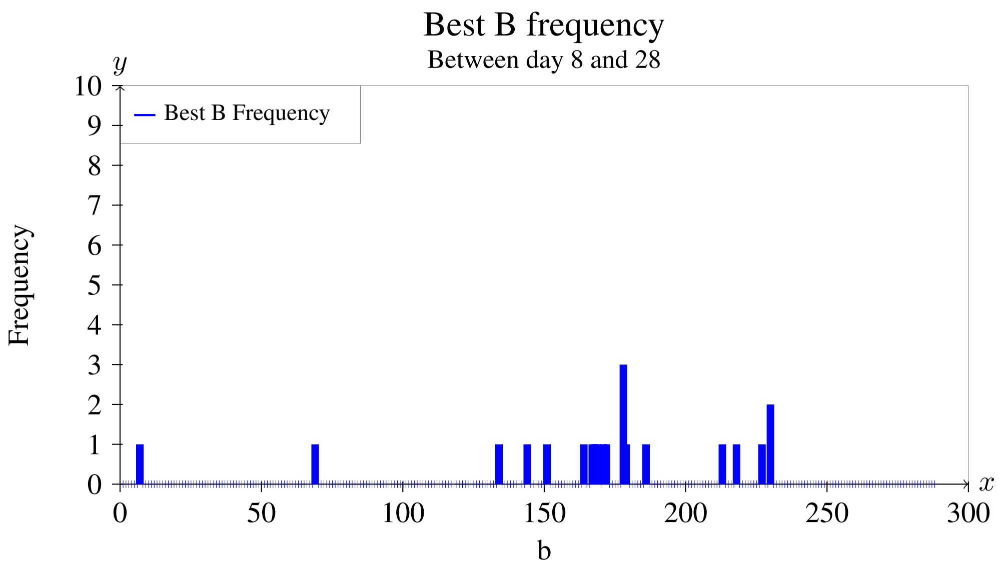


Figure 5.17: Frequency distribution of the best *beta* values from all days in scenario: Week, in game Football Manager 2010

On figure 5.17 we see that the majority of the distribution is stacked between *beta* value 140 and 230. We also notice that a *beta* value of 178 have occurred three times and *beta* value 230 have occurred two times. It may also appear that the *beta* values in this area is divided into two groups. But these groups are very close to each other. There are also two cases where the best outcomes have ended up with a *beta* values of 7 and 69. These are outside the majority group.

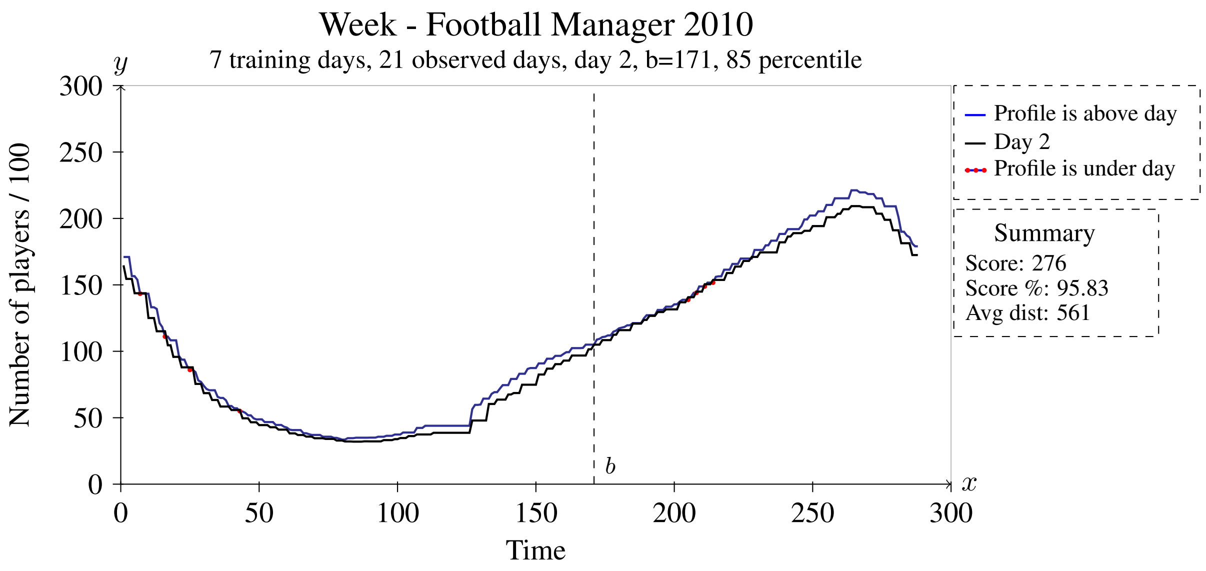


Figure 5.18: Results from the game Football Manager 2010 of scenario: Week, using *beta* value 171 on day 2

Figure 5.18 illustrate how the algorithm predicted the observed day by normalizing the profile to one of the *beta* values in the majority group. We see here that the average distance is as low as 561, this means that the profile was able to predict very similar to the observed day. A score of 276 proves that the profile managed to remain above the day 95.83 % of the time. By observing the graph we can clearly see that the curve of the profile and the observed day is almost equal. This is a accurate prediction.

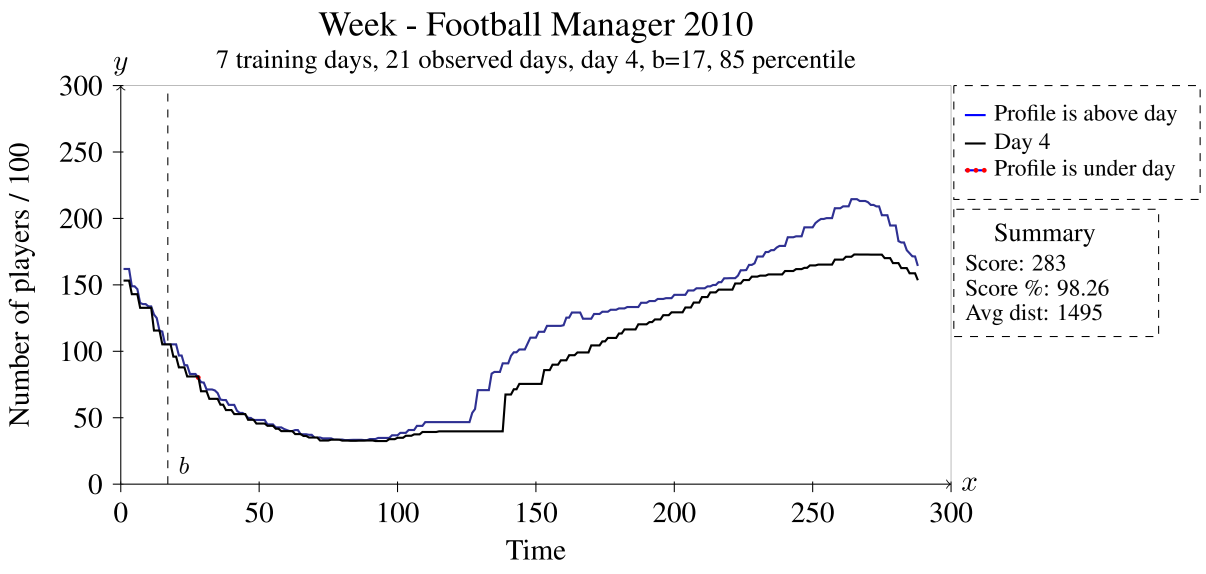


Figure 5.19: Results from the game Football Manager 2010 of scenario: Week, using *beta* value 174 on day 4

This next figure 5.19 illustrate the profile using a *beta* value outside the majority group. Here we see that the algorithm have predicted above the observed day 283 times which is 98.26% of the time. With a average distance of 1495 indicates that the profile is fairly close to the observed day, this is also ascertainable by studying the figure. This tells us that it is not necessary to use a *beta* value that is set by the majority group to get accurate predictions.

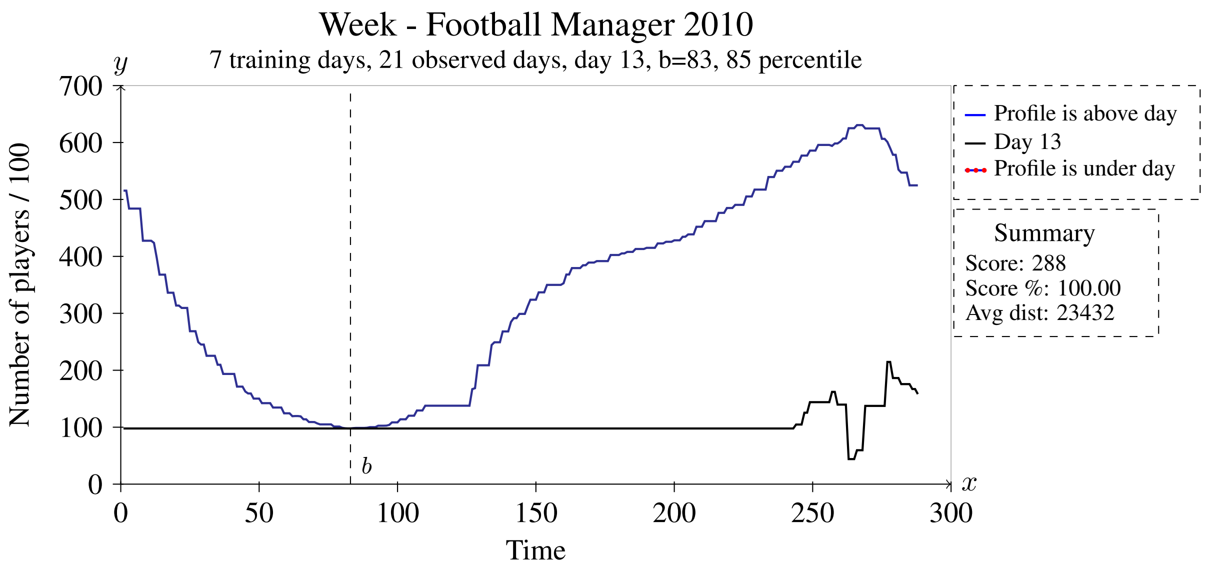


Figure 5.20: Results from the game Football Manager 2010 of scenario: Week, using *beta* value 83 on day 13

This next figure 5.20 shows how the algorithm tries to predict a day of data errors. Here it is clear that the data-collection script has collected data from the Steam application statistics feature, and that this data has not updated the current players over a long period of time. This results in a profile that remains over the observed day 100 % of time. With a very high average distance of 23432 the profile is completely different than the observed day, this is very clearly indicated by observing the graph. The profile is normalized by the lowest *beta* value.

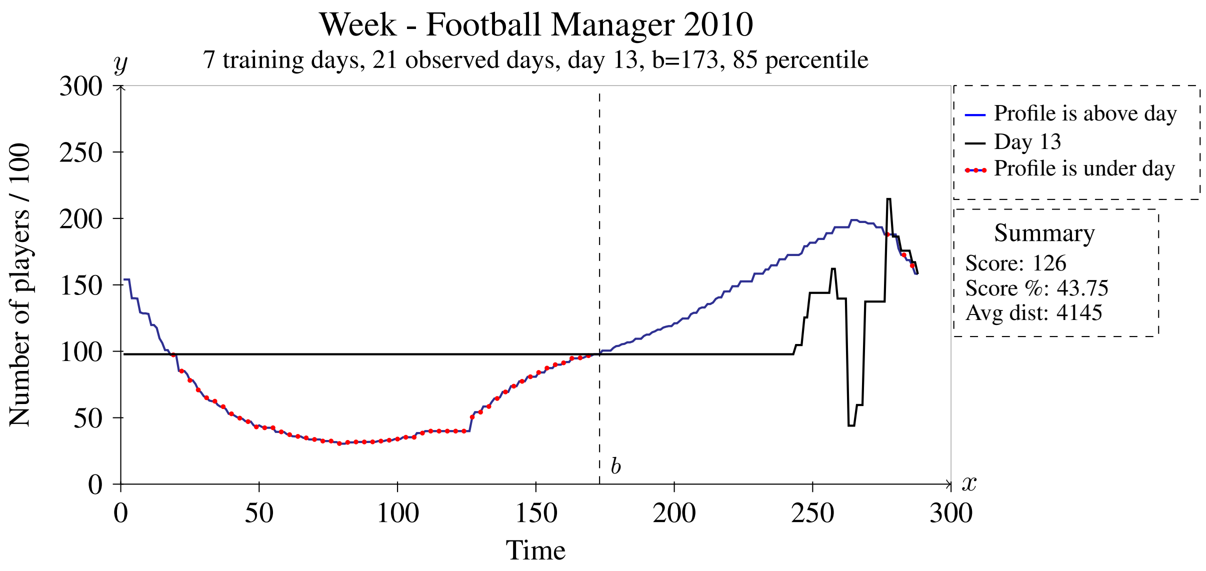


Figure 5.21: Results from the game Football Manager 2010 of scenario: Week, using *beta* value 173 on day 13

Figure 5.21 show the result when normalizing on a higher *beta* value in the same day with errors.

### Summary

It's very interesting to see how the data from the game Football Manager 2010 seems easier to predict than data from, for example, Counter Strike Source. This suggests that the type of game provides a different impact on the predication results. After studying the large amounts of results, it was interesting to see that the algorithm managed to predict the day regardless of *beta* value used to normalize the profile, but only on observed days with few data errors. It should be mentioned that the best results came from tests done where the profile was normalized with *beta* value located in the majority group. Figure 5.18 illustrates how accurate the algorithm can predict days that are not effected by any data errors. It is important to highlight that this example is not a single case, but that almost all days tested without errors using all *beta* values provides accurate prediction results in this test scenario.

When it comes to days with data errors, we see on the two figures that the algorithm fails to predict. It may therefore indicate that the algorithm must rely on a data stream that provides the correct value on current players online.

In this scenario, data from the game Football Manager 2010 was tested. The algorithm had a training period of 7 days and was tested against the 21 ongoing days. Some of these ongoing days have data errors in them. These in form of both data loss and incorrect updated data. This has resulted in inaccurate predications on those days. By looking away from these days, the algorithm managed to predict very accurately by the use of large parts of the *beta* value spectrum. But very well where the majority of the best *beta* values are found.

### 5.2.3 Scenario: Wednesdays

In this section the results from the scenario Wednesday will be presented. The algorithm had a learning period of 5 consecutively Wednesdays, these days will form the profile that will be tested against 20 continuous Wednesdays. This test is performed to see if there is a pattern to get better results by choosing specific days of the week.

The following figure 5.22 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

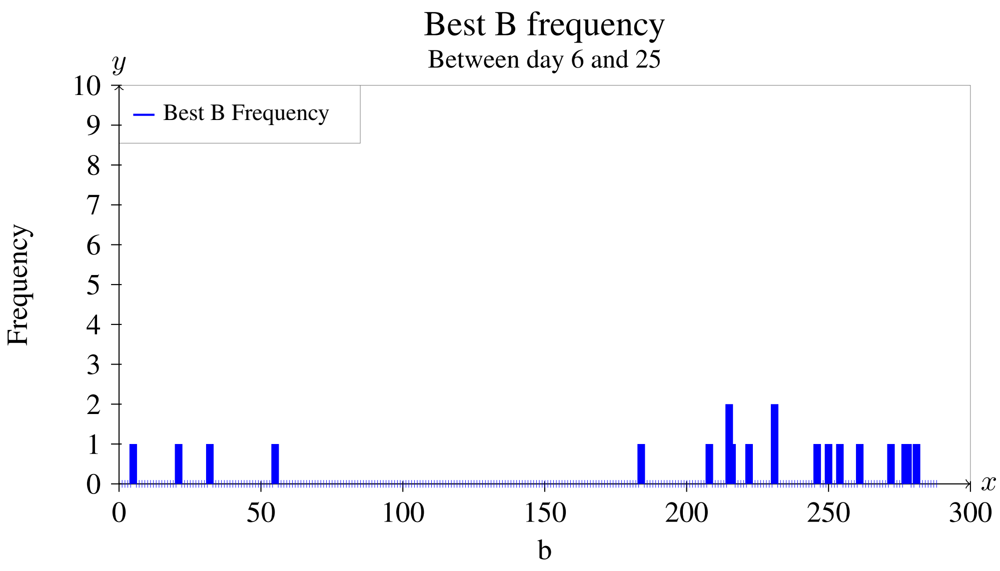


Figure 5.22: Frequency distribution of the best *beta* values from all days in scenario: Wednesday, in game Football Manager 2010

In the frequency distribution one can notice that the majority of frequencies occur in range between *beta* value 180 and 280. In this area *beta* value 215 and 231 have occurred as the best *beta* value twice. There is a large area in the middle of the *beta* range where there has been no best beta. It is in this area, where the number of players usually increases. Then there are a few occurrences of the best *beta* values in the beginning of the *beta* range. These have a small distance from each other and are in the period in which the number of players tend to decrease somewhat.

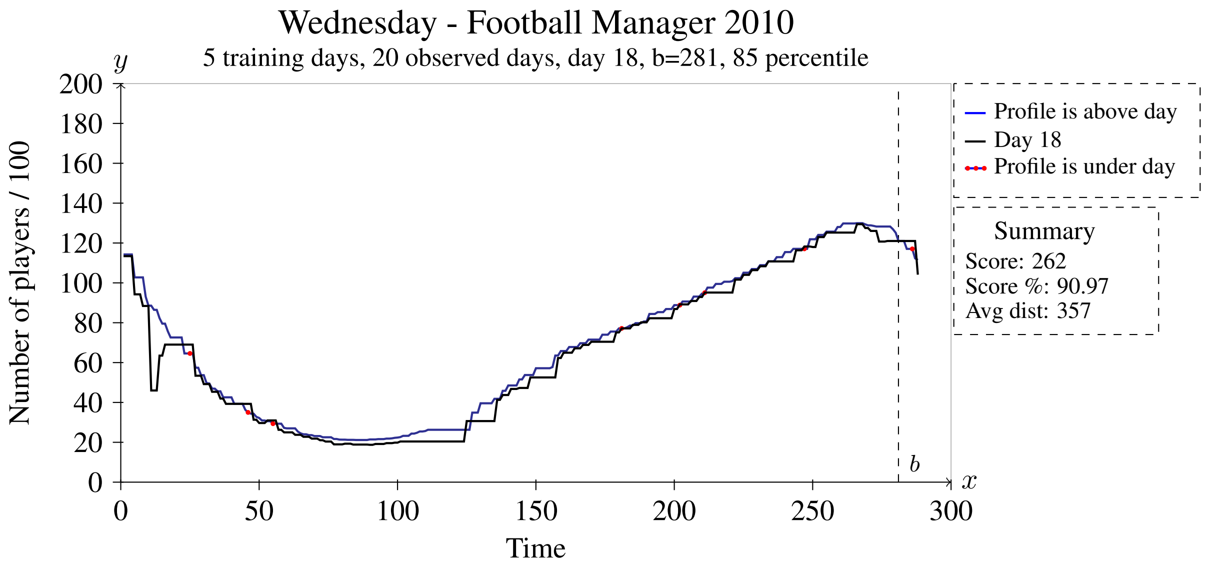


Figure 5.23: Results from the game Football Manager 2010 of scenario: Wednesday, using *beta* value 281 on day 18

This first figure 5.23 illustrates how the algorithm performed using the best *beta* value on day 18. This is also the example with the lowest average distance of 357 in this scenario. We can see how accurate the profile have predicted the observed day, and we can probably estimate that the sudden decrease of players in the beginning on the observed day contributes in a greater average distance than the rest of the day. We notice that some times the profile get a value belove the observed day, but these occurrences are very scattered over the entire area. This is knowledge telling us that the profile have not failed in large periods, and has remained close to the observed day.

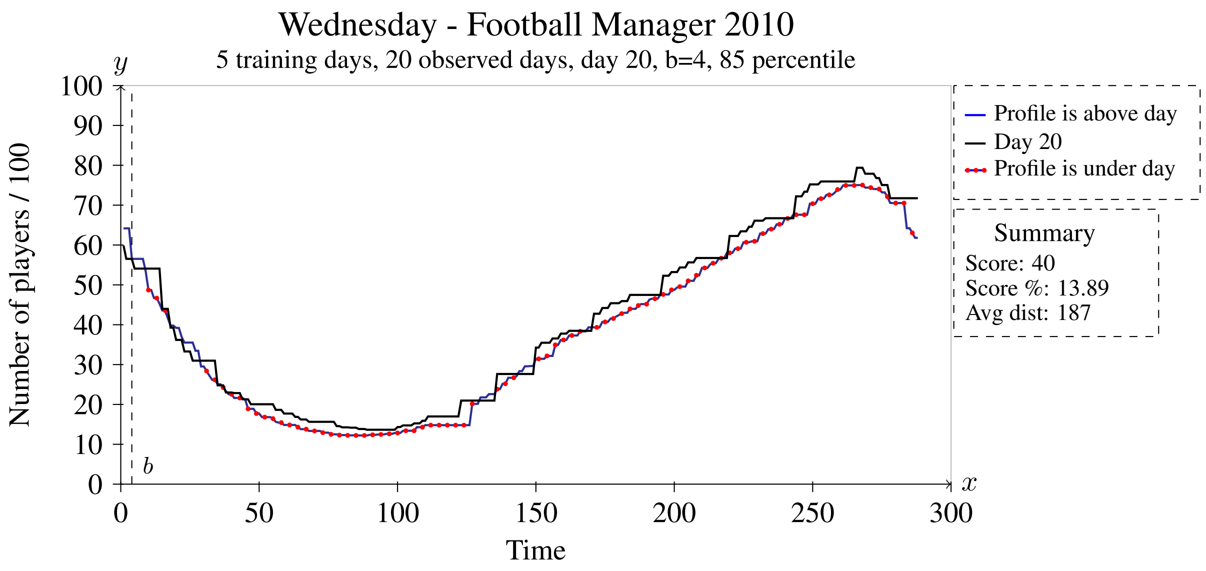


Figure 5.24: Results from the game Football Manager 2010 of scenario: Wednesday, using *beta* value 4 on day 20

In figure 5.24 we see results that the profile have not been able to stay above the observed day, but has been able to remain close to the day the whole time. With a average distance of only 187 we see that even tho the score is not that good, the algorithm have been able to predict close to the day.

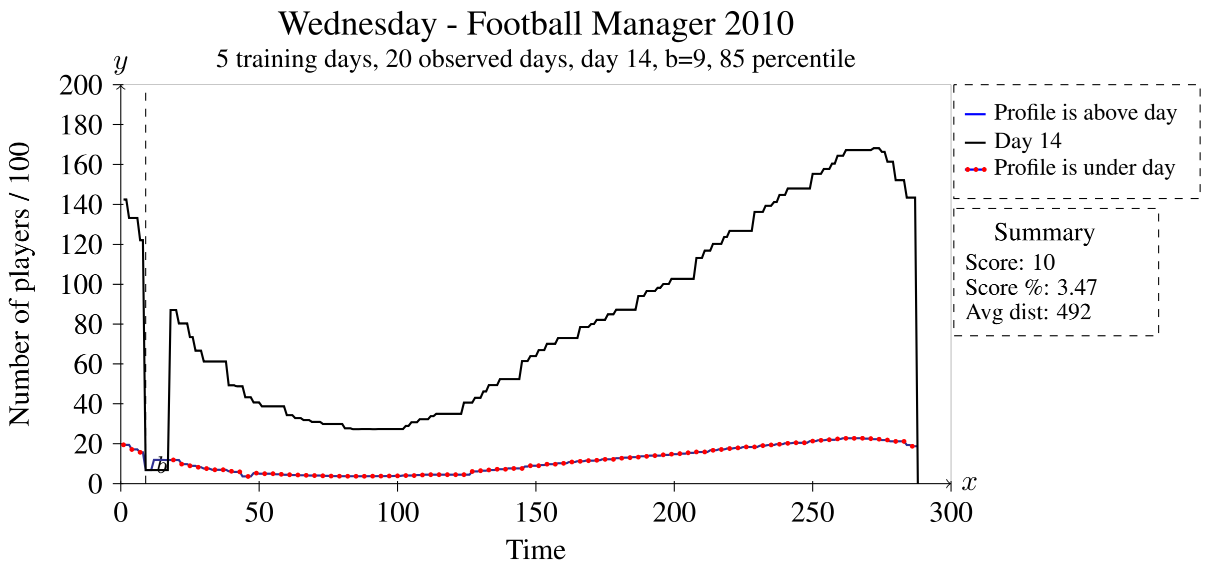


Figure 5.25: Results from the game Football Manager 2010 of scenario: Wednesday, using *beta* value 9 on day 14

This next figure 5.25 demonstrate the devastating effect unforeseen player loss or data error is for the algorithm. Here we see that the observed day is predicted wrong when a large loss of players or data error have occurred. It may well be that the this is a case where an update has been preformed, it will certainly explain the drastic loss of players. By normalizing the profiles with a *beta* value that happen to occur in a time period where a update have been performed, will provoke a very bad predication for the rest of the day. This is revealed by observing the figure.

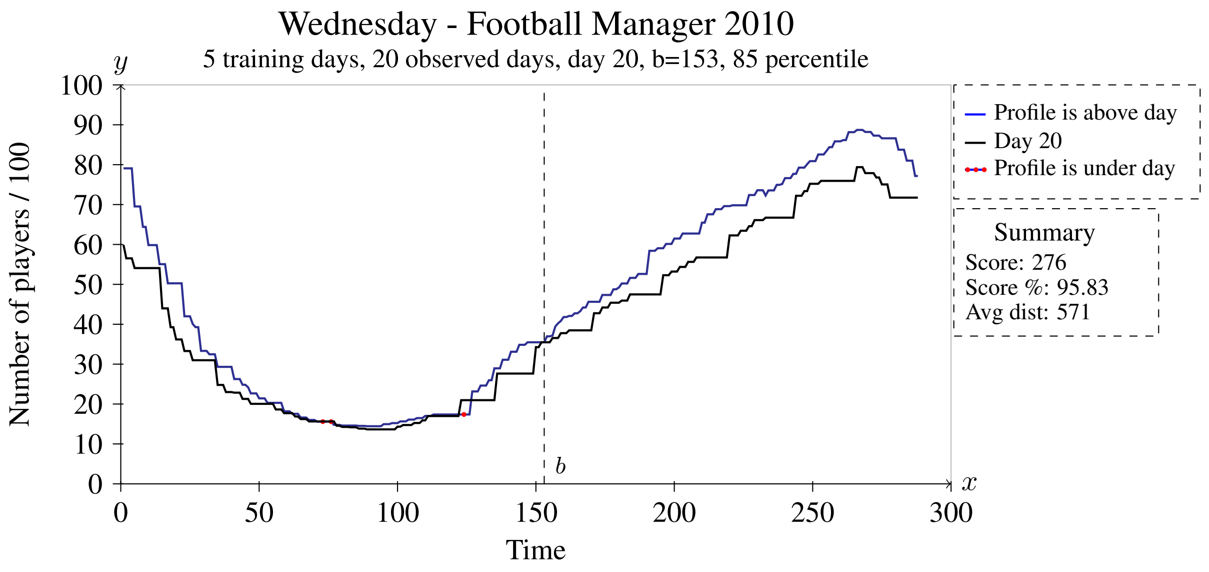


Figure 5.26: Results from the game Football Manager 2010 of scenario: Wednesday, using *beta* value 153 on day 20

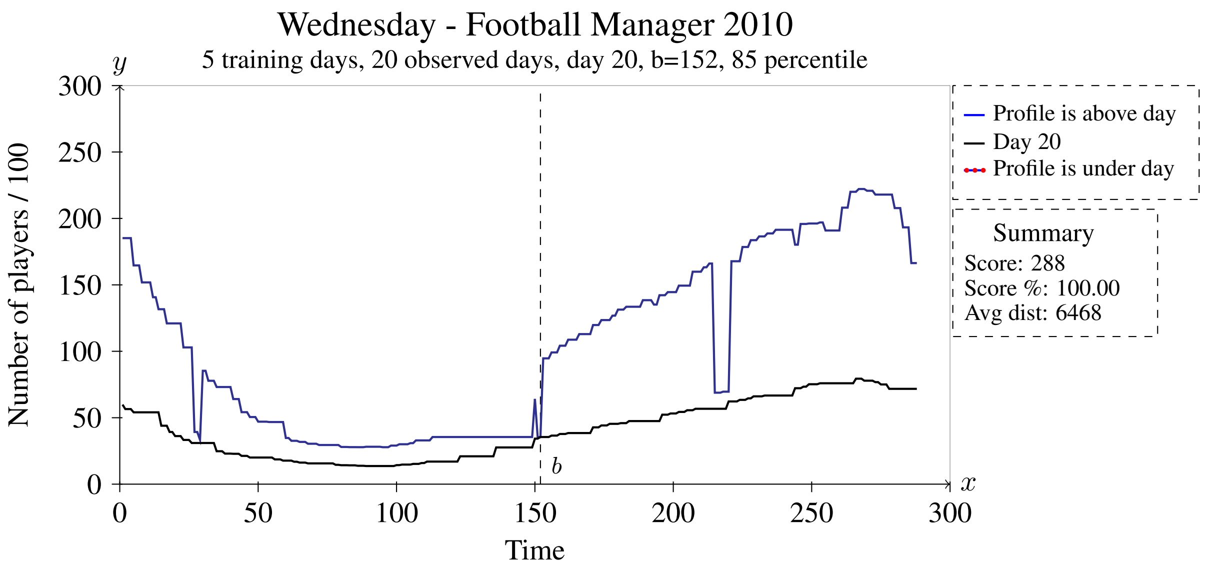


Figure 5.27: Results from the game Football Manager 2010 of scenario: Wednesday, using *beta* value 152 on day 20

These two last figures in this scenario illustrate how two *beta* values right next to each other can completely re-transform the profile. In figure 5.26 we see a good prediction using *beta* value 153 on day 20. On figure 5.27 the profile has completely re-transformed by using the *beta* value 152 on day 20. These beta values are next to each other. The player value is only 100 more on *beta* value 153 than 152. This provide valuable information on how small the changes can be to completely re-transform the profile. This is probably caused by errors in the profile.

## Summary

It is particularly interesting to find many examples where the algorithm works very well and manages to draw a curve that corresponds to the day observed. Examples shown in the first figures clearly show that it is possible to predict Wednesdays of long distance into the future by training the profile just by using the 5 Wednesdays as described in the introduction of this scenario.

It was also very interesting to find an example on a period of drastic loss of players who do not necessarily needs to be data errors. Figure 5.25 have truly a lot of knowledge on how the profile might behave if it is normalized at such a time.

In the period of the escalation of players, it was interesting to see how the profile re-transformed totally by a small change in time.

### 5.2.4 Scenario: 60 days

In this scenario 60 sequential days will be used to create the profile. The following 20 days will be tested against the profile. The results from this scenario will determine if the profile perform more accurate if it gets fed with a large amount of training days.

The following figure 5.28 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

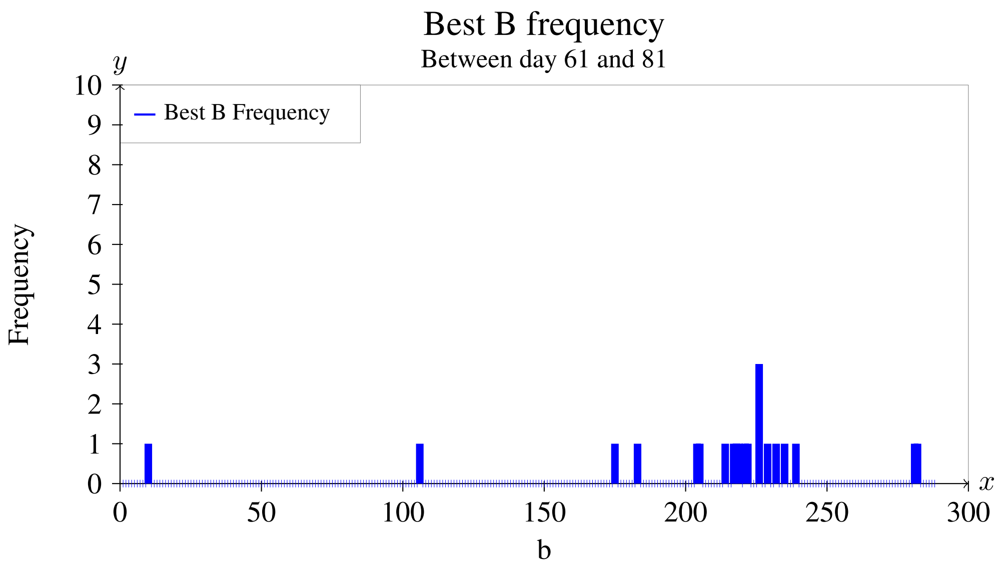


Figure 5.28: Frequency distribution of the best *beta* values from all days in scenario: 60 Days, in game Football Manager 2010

In the frequency distribution we definitely see majority group with most occurrences. We see that most of the best *beta* values is gathered very tightly around *beta* value 226 which has occurred as best *beta* value on three of the days. In the area where this majority group gathered the number of players tend to be at a peak. This distribution tells us that the best results might be found in the area where this majority group is located. It will be interesting to see if the *beta* values that are not in this group have some kind of abnormalities or not.

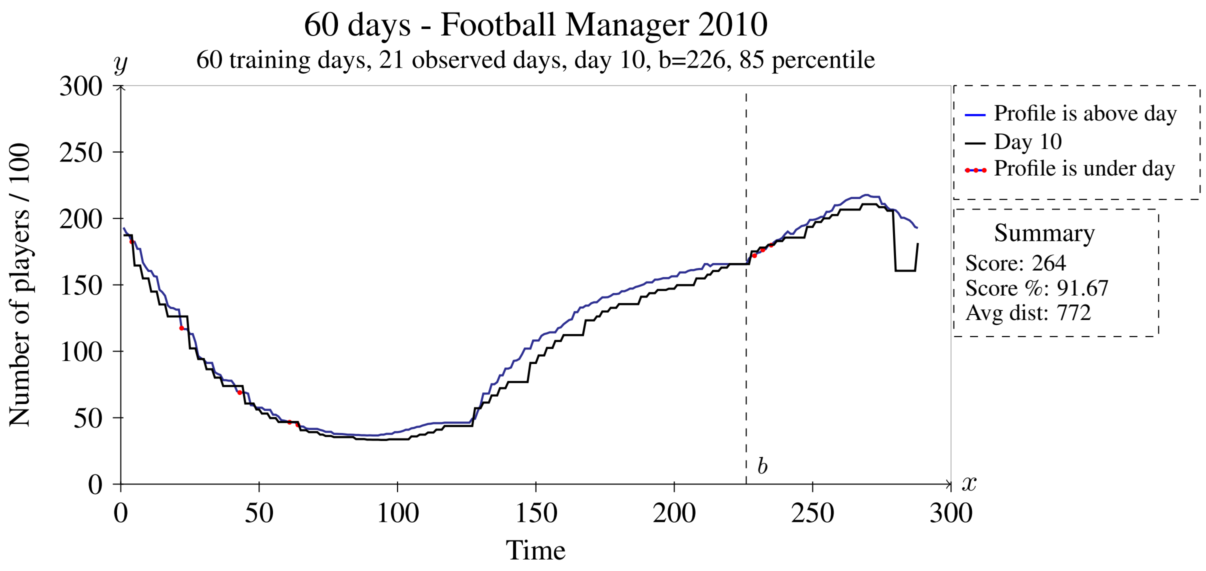


Figure 5.29: Results from the game Football Manager 2010 of scenario: 60 Days, using *beta* value 226 on day 10

In this first figure 5.29 the lowest average distance with a score over 260 is shown. This example have a score of 264 out of 288. This means that the profile was able to predict and remain above the observed day 91.67% of the time. The average distance 772 is the lowest average distance with a score over 260 in this entire scenario. We can see at the figure that this is a very similar predication of the observed day. Profile is close to the observed day most of the day, only towards the end, we observed that the day gets a sudden player loss that profile is not able to predict.

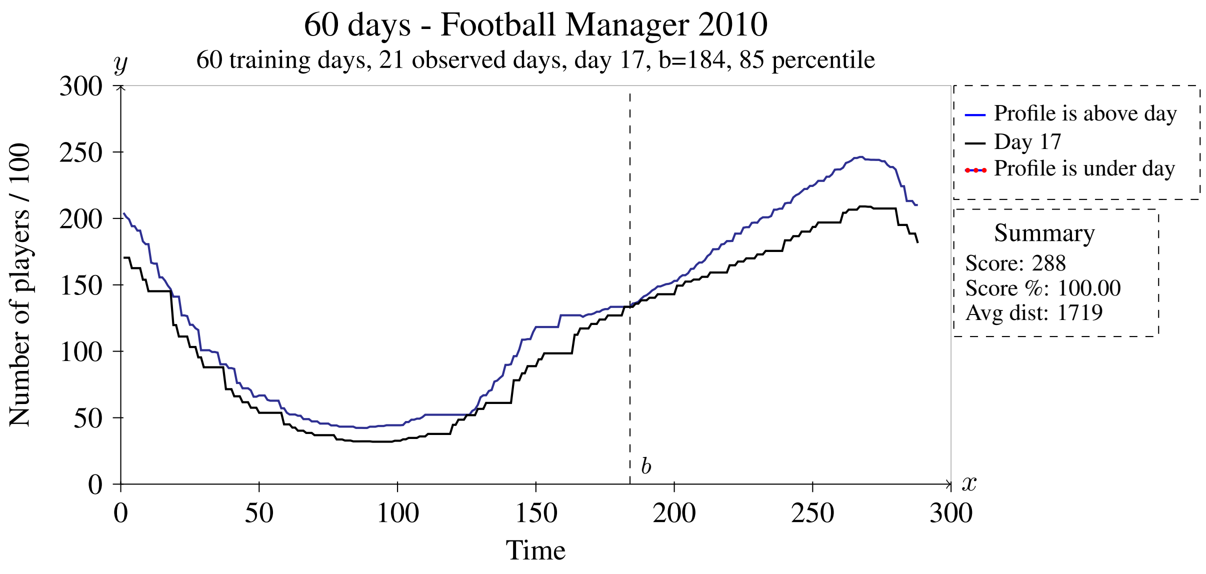


Figure 5.30: Results from the game Football Manager 2010 of scenario: 60 Days, using *beta* value 184 on day 17

This next figure 5.30 illustrate an example with the lowest average distance with a perfect score of 288. Be studying the graph we see clearly that the profile always stay above the observed day and at the same time it stay pretty close as well. Beside a slightly higher increase after the *beta* value, the profile manage to stay close to the observed day most of the time. The *beta* value used to normalize the day is in the area where most of the best *beta* values here found in the frequency distribution.

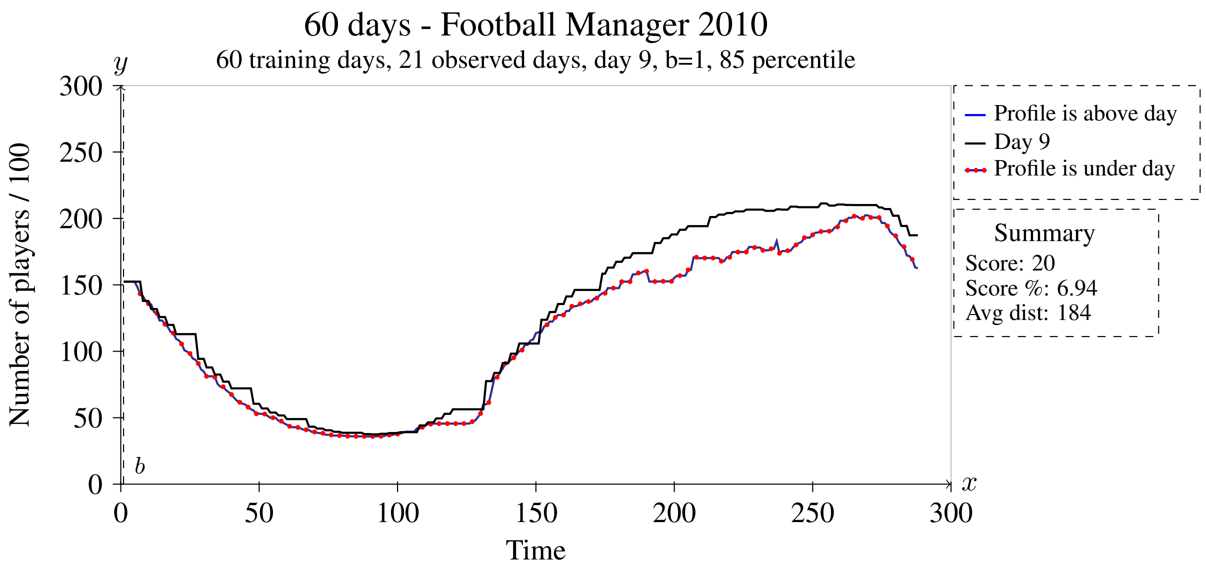


Figure 5.31: Results from the game Football Manager 2010 of scenario: 60 Days, using *beta* value 1 on day 9

In figure 5.31 the profile is normalized with *beta* value 1. Here we see that profile is under the observed day most of the time resulting in a very bad score. But we also notice that the profile still remain close to observed day. This suggests that by a slightly higher value by the *beta* will place the profile to be more desirable place closer to the observed day. In this case, it might be that the value given at *beta* 1 is lower because of an error.

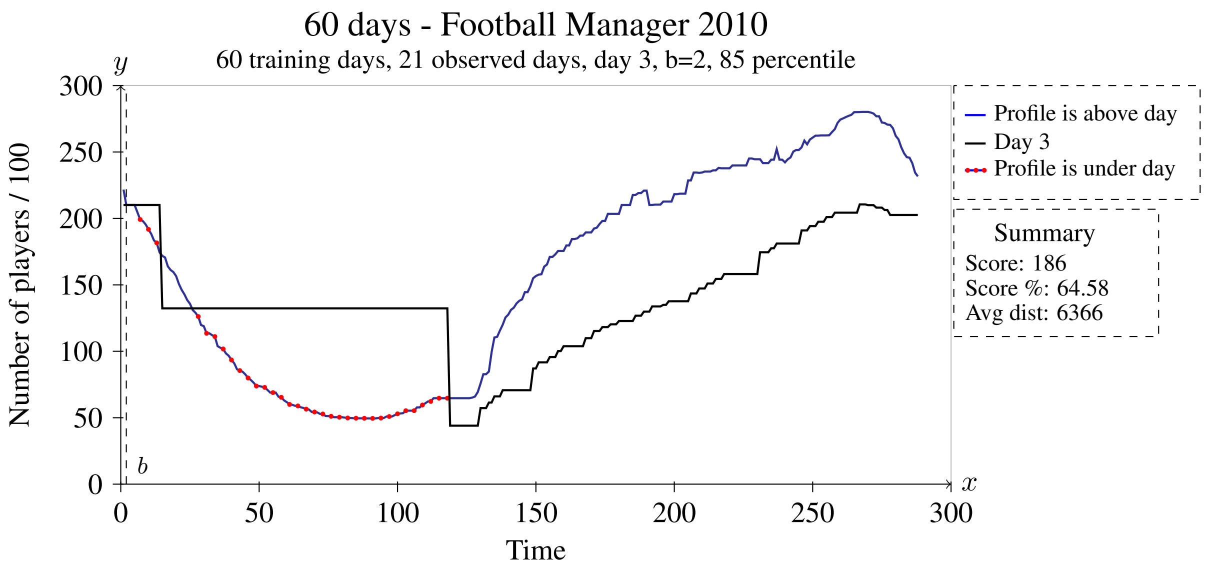


Figure 5.32: Results from the game Football Manager 2010 of scenario: 60 Days, using *beta* value 2 on day 3

Figure 5.32 shows another example of a day with data errors. This phenomenon might occurs because the Steam application is not able to update the current players online value in some periods. As the algorithm is now it does not take these types of errors into account. The algorithm is probably depending on a accurate flow of data.

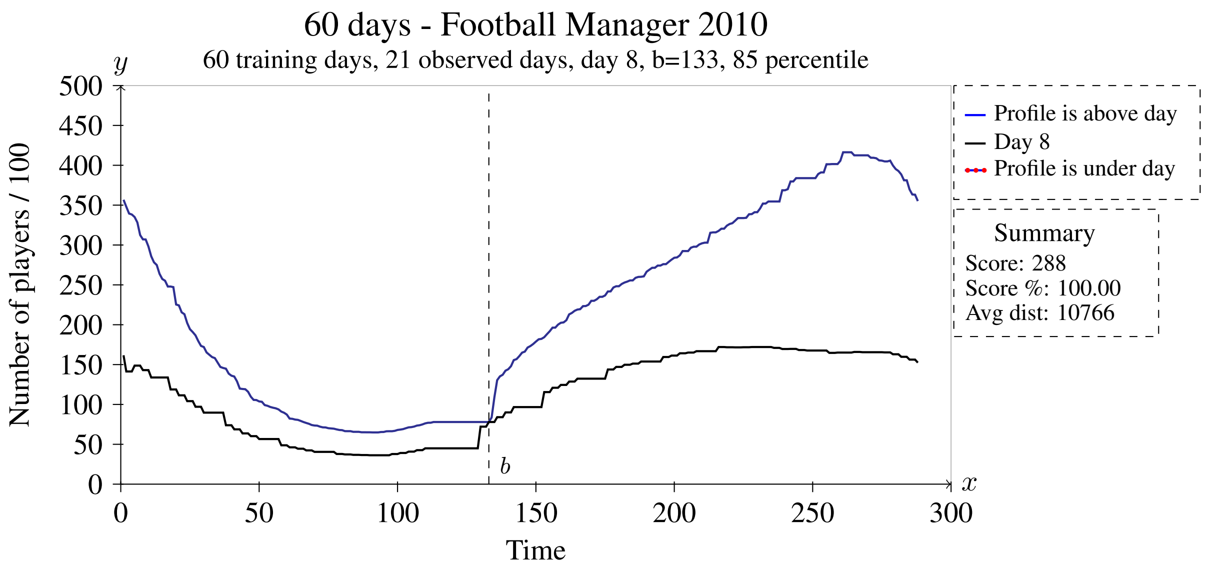


Figure 5.33: Results from the game Football Manager 2010 of scenario: 60 Days, using *beta* value 133 on day 8

In this last figure 5.33 we see that the profile stay way above the observed day most of the time. It is desirable to always be above the observed day but the average distance is very large and should therefor be considered as a bad prediction. It might also be that the observed day have less players online during peak hours and that this is the reason for the errors in the prediction.

## Summary

It was interesting to see how close the distribution of good predictions was in the frequency distribution. The inspection of the results showed that many good predictions was found in this area. It should also be mentioned that most of the findings in this scenario was very repetitive from other scenarios.

## 5.3 Results: Supreme Commander 2

### 5.3.1 Introduction

In this section results from the game Supreme Commander 2 will be presented. The algorithm has been tested on the game data for all scenarios as described in the algorithm test plan section.

The game Supreme Commander 2 is a real-time strategy game. This type of game that require players to set aside plenty of time to complete a game, but usually a game does not last longer then 30-60 minutes of play time. Players are usually fully aware of this, so they does not start a game if they do not have enough time. On the basis of this, one can say that the data for this game should be more predictable because players are “bound“ to play over a period of time. This game is much less popular than both Counter Strike Source and Football Manager 2010, this is evident by looking at the number of players in the graphs of this game.

### 5.3.2 Scenario: Week

In this section the first scenario will be tested. The algorithm will have a learning period of a week and will be tested against the three following weeks. Seven consecutive days will therefore be tested against twenty-one continuous days.
The following figure 5.34 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

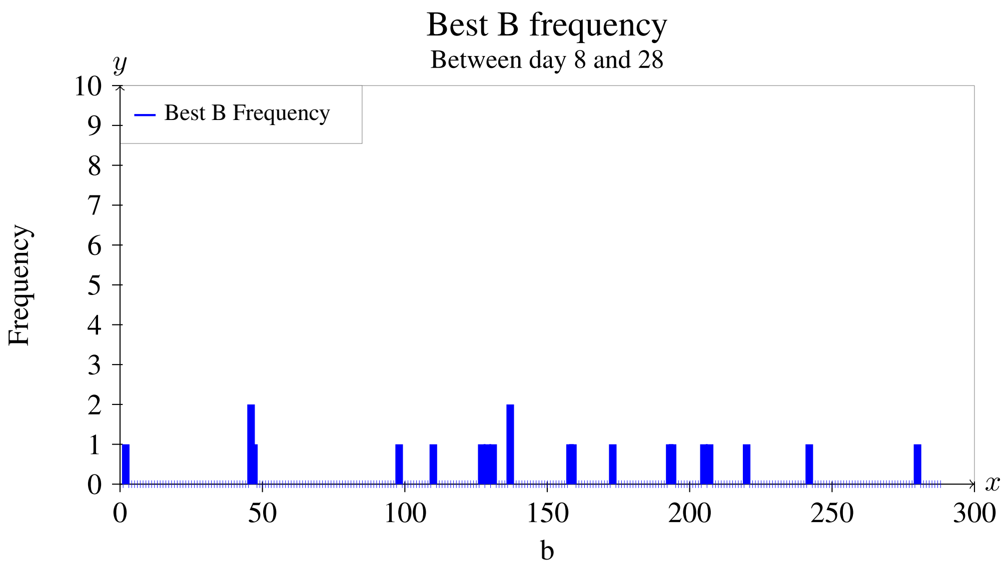


Figure 5.34: Frequency distribution of the best *beta* values from all days in scenario: Week, in game Supreme Commander 2

By studying the frequency distribution, we see that the distribution is spread over the *beta* range. We also notice that the *beta* values 46 and 137 occur twice. It also seems that there are frequent occurrences in the area between *beta* values 100 and 210 This area has proven to be a good area for accurate prediction on the same scenario in the two other games that have been tested as well.

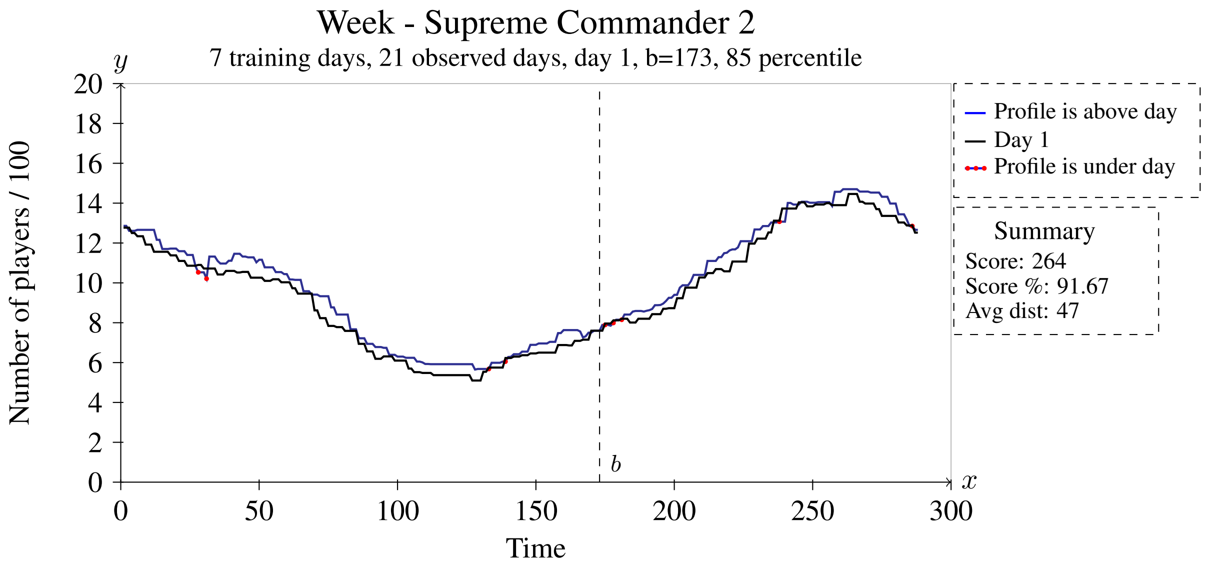


Figure 5.35: Results from the game Supreme Commander 2 of scenario: Week, using *beta* value 173 on day 1

Figure 5.35 illustrate the example with the lowest average distance with a score over 90% of the time. The example is taken from one of the instances with b value of 173, this is the best *beta* value for day 1. The average distance is as low as 47, this is very a very good prediction. The figure is somewhat deceptive because the figures from previous games in the same scenario looks more precise than this one. This is because the player values are different, Supreme Commander 2 have peaks of about 2100 players simultaneously while Counter Strike Source have peaks of 70000.

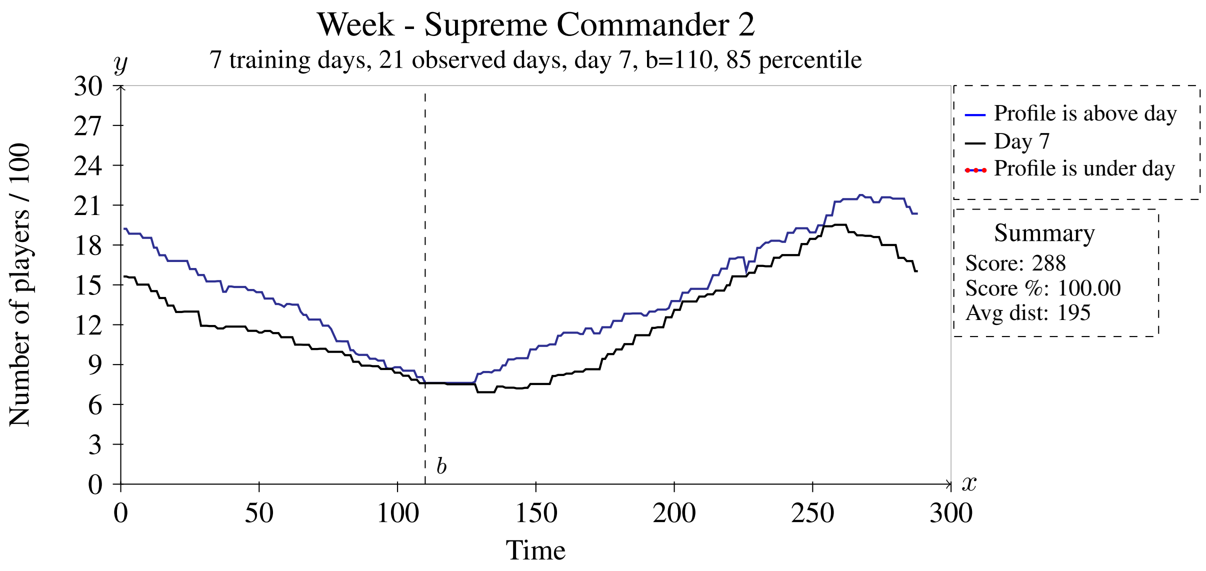


Figure 5.36: Results from the game Supreme Commander 2 of scenario: Week, using *beta* value 110 on day 7

This next example is also taken from one of the best values. Figure 5.36 shows a *beta* value of 110 which is the best *beta* value for day 7. Here one can see that the profile scores a rating of 288, something that indicates that the profile is above the observed day 100% of the time. The average distance is 195, again the figure looks misleading but only because the number of players are lower than in previous examples from other games tested.

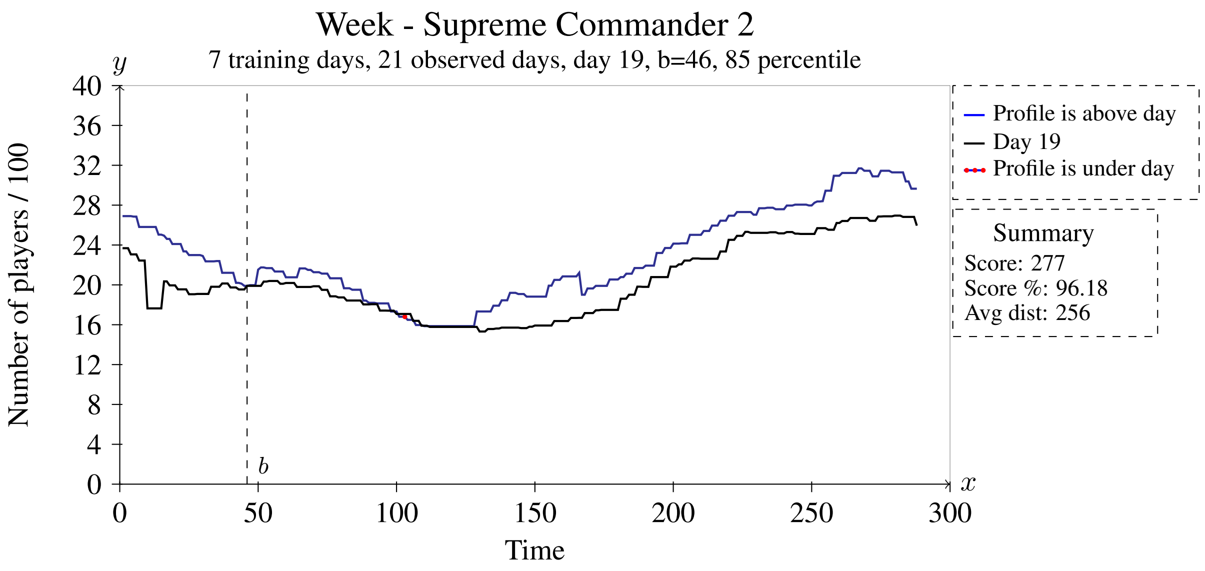


Figure 5.37: Results from the game Supreme Commander 2 of scenario: Week, using *beta* value 46 on day 19

The purpose of showing another example of a good prediction (figure 5.37), is to show that the algorithm works with three different *beta* values and on different days with a few days of distance from each other. It should also be mentioned that these days have been predicted with good results over all *beta* values in this scenario.

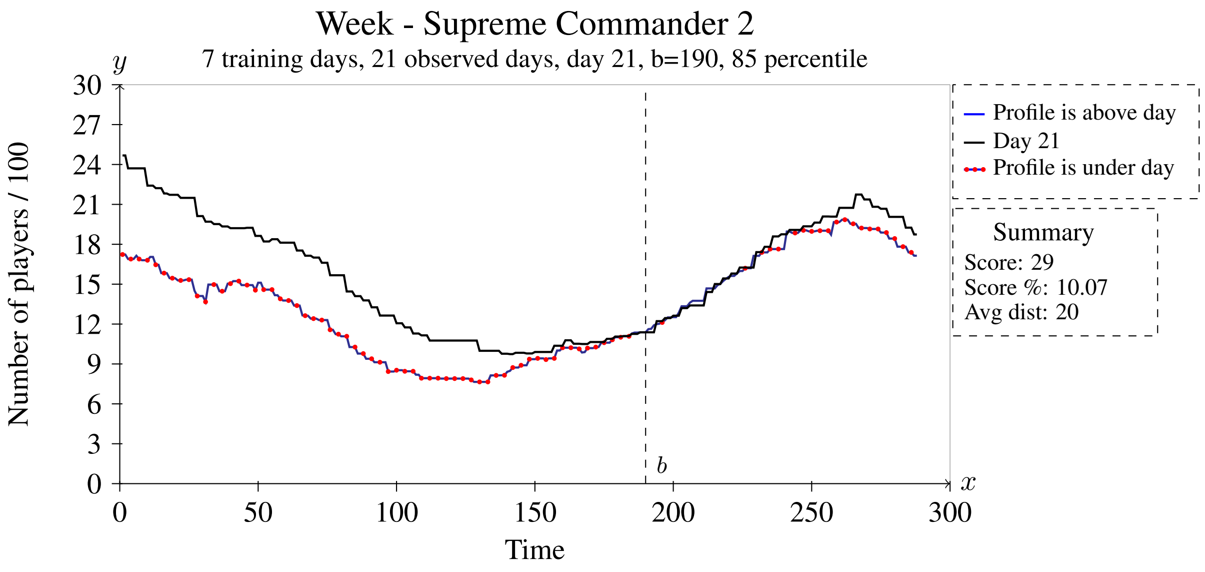


Figure 5.38: Results from the game Supreme Commander 2 of scenario: Week, using *beta* value 190 on day 21

This last figure 5.38 shows how the algorithm fails to predict correctly even though the observed day has few errors. Here we see that in a period where the number of players is between the lowest point and the way up to the highest peak, the profile only manage to predict above the observed day 10.07 % of the time. But the average distance is so low that this might not be of great importance in a situation where the results would determine the number of servers running to manage the amount of resources available. The profile had the same difficulties predicting this day when normalized using *beta* values between 50 and 288. The *beta* values in range 1 to 50 provoked the profile to stay above the observed day but with somewhat large average distance.

### Summary

It was interesting to see how spread the frequency distribution was in figure 5.34. This was different from the other games tested in this same scenario. There was also an interesting discovery to see that the algorithm was able to be so accurate in the cases shown in the first few figures. The reason for this might be that it perhaps is easier to predict data from games that are ”bound” to a period of time.

In figure 5.38 an example where the profile was not able to stay above the observed day most of the time was shown. The reason for this might be because of change in human behavior towards that game. One can observe on the figure that for some reason the distance between the profile and the day is larger between *beta* 0 and 150. This indicates that there are more players online during this time than the profile has been based on. Maybe a new update or a certain day that players can stay up longer to play is the reason for this.

### 5.3.3 Scenario: Wednesdays

In this section the results from the scenario Wednesday will be presented. The algorithm had a learning period of 5 consecutively Wednesdays, these days will form the profile that will be tested against 20 continuous Wednesdays. This test is performed to see if there is a pattern to get better results by choosing specific days of the week.

The following figure 5.39 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

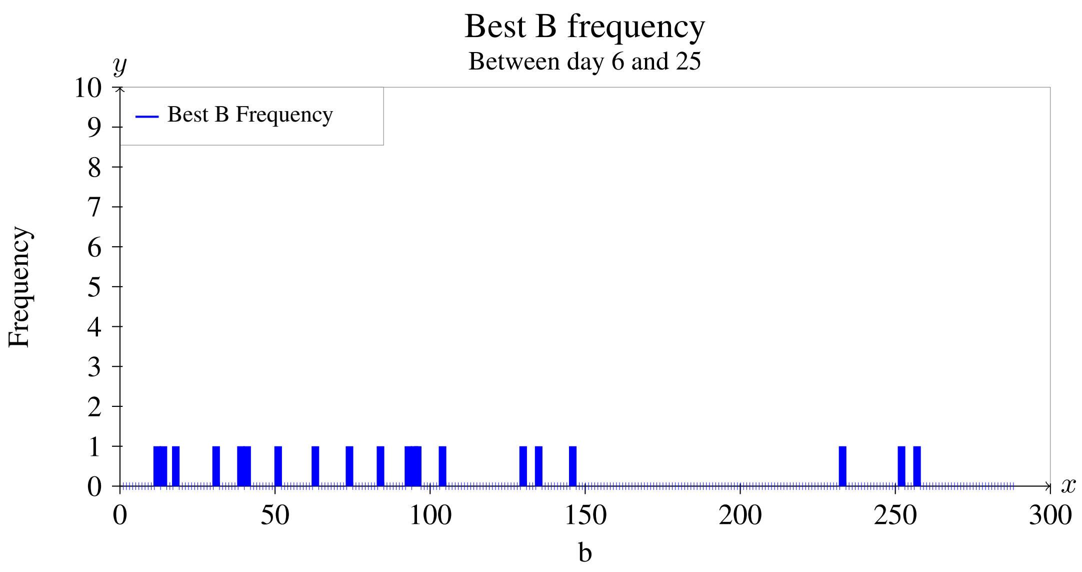


Figure 5.39: Frequency distribution of the best *beta* values from all days in scenario: Wednesdays, in game Supreme Commander 2

We see here on the frequency distribution that the majority of occurrences have occurred in the *beta* range between 1 and 150.

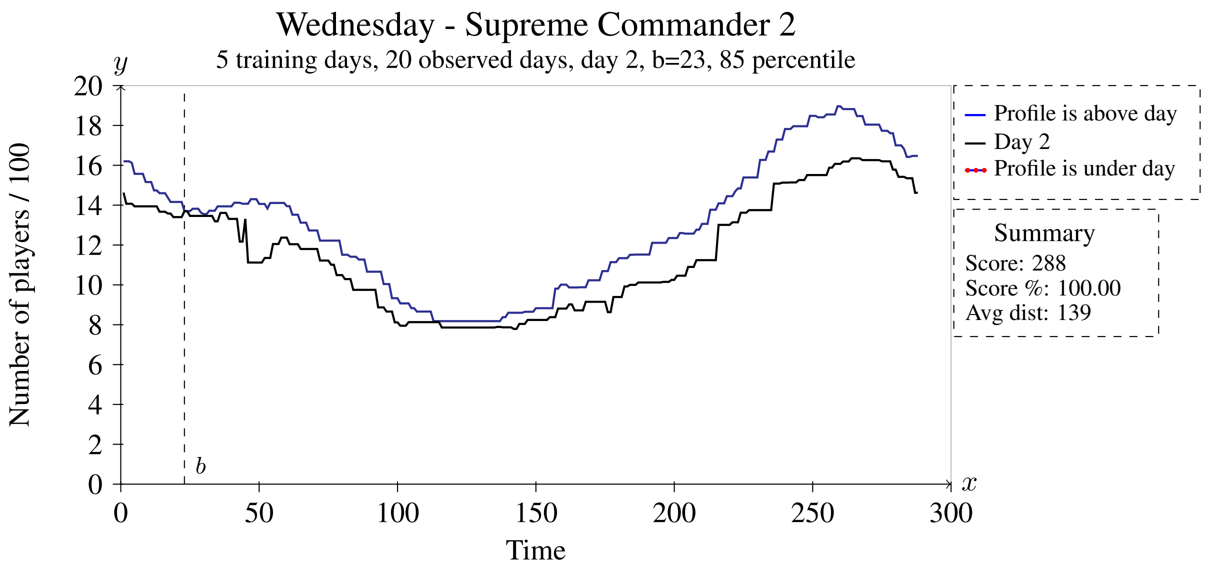


Figure 5.40: Results from the game Supreme Commander 2 of scenario: Wednesday, using *beta* value 23 on day 2

In this first figure 5.40 we see an example with a perfect score. The profile have remained above the observed day 100% of the time. The average distance is 139, this indicate that the profile have predicted somewhat according to the day, we can confirm this by observing the graph. The profile is normalized with the beta value 23, when all the results were studied thoroughly, it appeared that in the area around *beta* value 23 turned out to have many good predictions as foreseen by the frequency distribution.

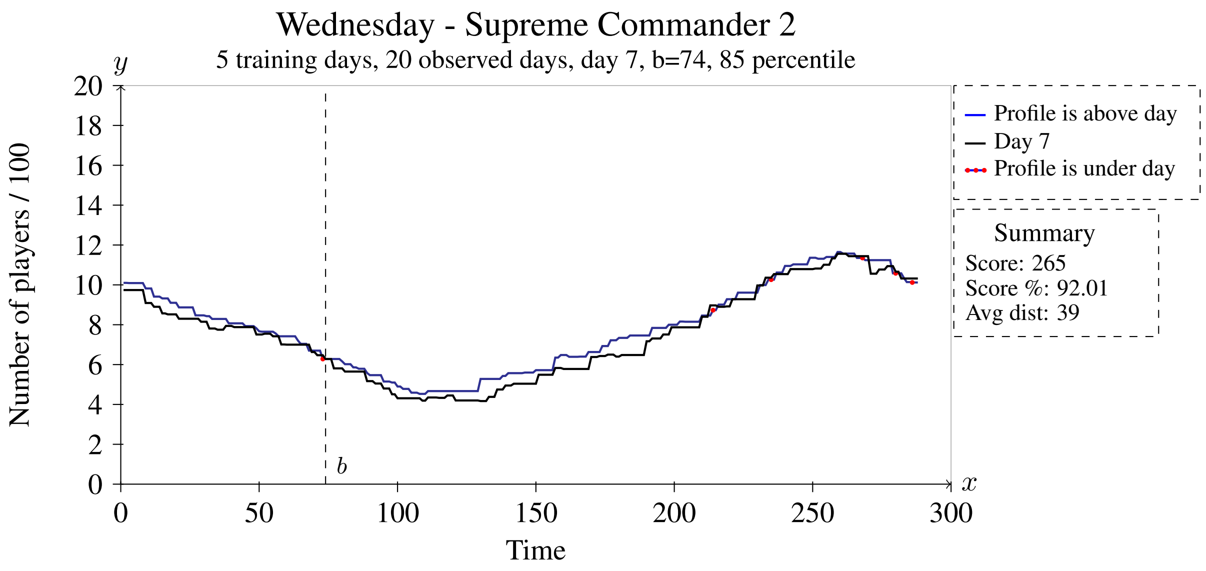


Figure 5.41: Results from the game Supreme Commander 2 of scenario: Wednesday, using *beta* value 74 on day 7

Figure 5.41 illustrates the best prediction in this scenario with score over 90%. Here wee see that the profile is very close to the observed day and most of the time remain above as well. We also notice that the *beta* value used is in the range where the most best *beta* values where found. We see that there are only a few occurrences where the profile has a lower value than the observed day and that these values are close to the observed day.

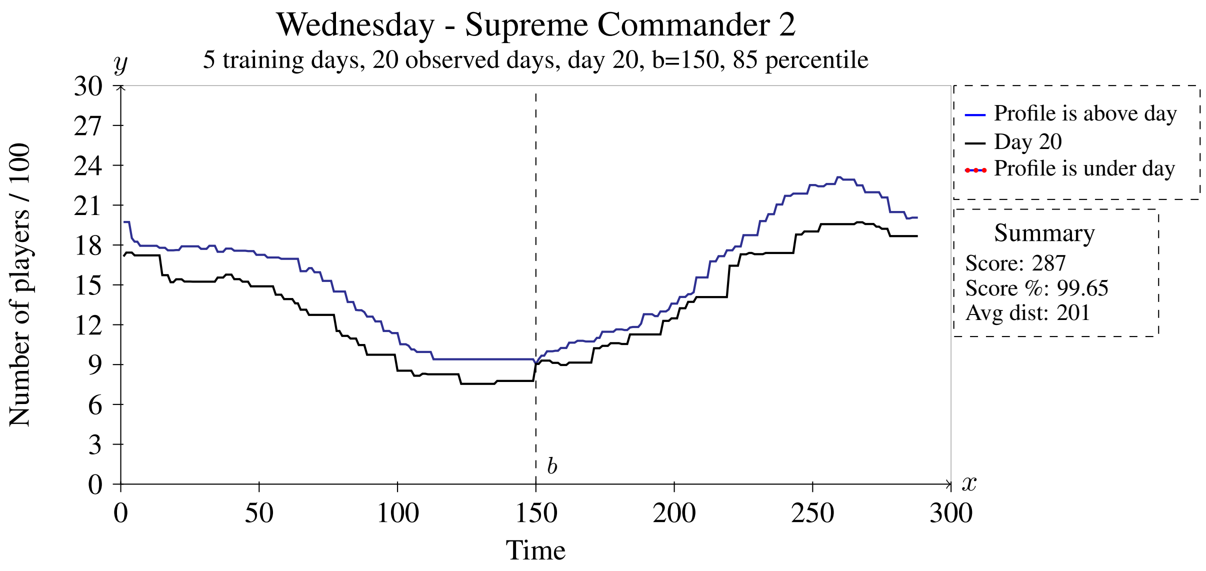


Figure 5.42: Results from the game Supreme Commander 2 of scenario: Wednesday, using *beta* value 150 on day 20

In the next two figures, a comparison be made. This is to be able to see result differences just by a small change in the number of players. Figure 5.42 shows that the profile has remained above the observed day 99 % of the time, and we also see that it has predicted good by staying close throughout.

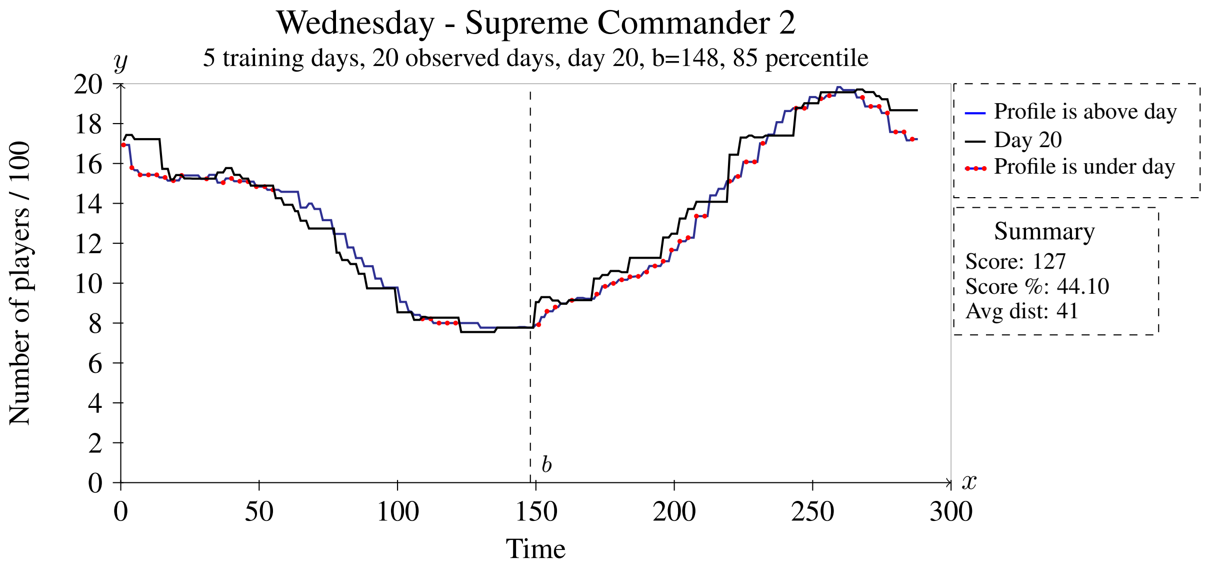


Figure 5.43: Results from the game Supreme Commander 2 of scenario: Wednesday, using *beta* value 148 on day 20

In this next figure 5.43 we see that the *beta* value used to normalize the profile has a slightly lower value than the previous figure. In this case, this means that the profile is unable to remain above the day observed as often as the previous figure.

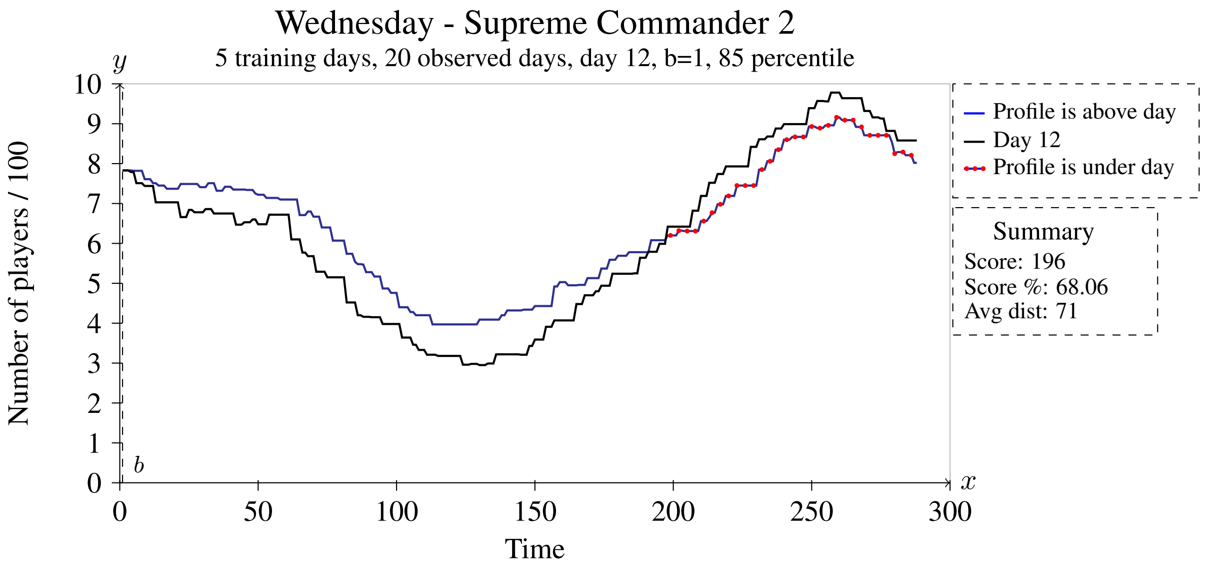


Figure 5.44: Results from the game Supreme Commander 2 of scenario: Wednesday, using *beta* value 1 on day 12

This last figure 5.44 shows an example where the profile is normalized with the beta value of 1. We see that the profile remains above the observed day until the *beta* value of 200. At this point crosses the observed day over the profile the rest of the time. The special thing about this example is that it seems that the algorithm has a *beta* value without "error". This because the profile behaves correctly according to the observed day until the *beta* value of 200.

### Summary

In the figures were compared, it was exciting to see how a small change meant that the profile predicted with a much better score. It could have been interesting to see what kind of results would have been achieved if the normalized profile with a slightly higher value than the one observed on the day.

The latest figure was an exciting discovery to see the profile is simply not able to predict a day where the human behavior increases more than usual when it is normalized to a value with a long distance from the irregularities.

### 5.3.4 Scenario: 60 days

In this section the results from the scenario: 60 days are presented. The algorithm will have a learning period of 60 days and will be tested against the following 21 ongoing days. This scenario will provide knowledge whether feeding the algorithm with large amount of training days will cause better results or not.

The following figure 5.45 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

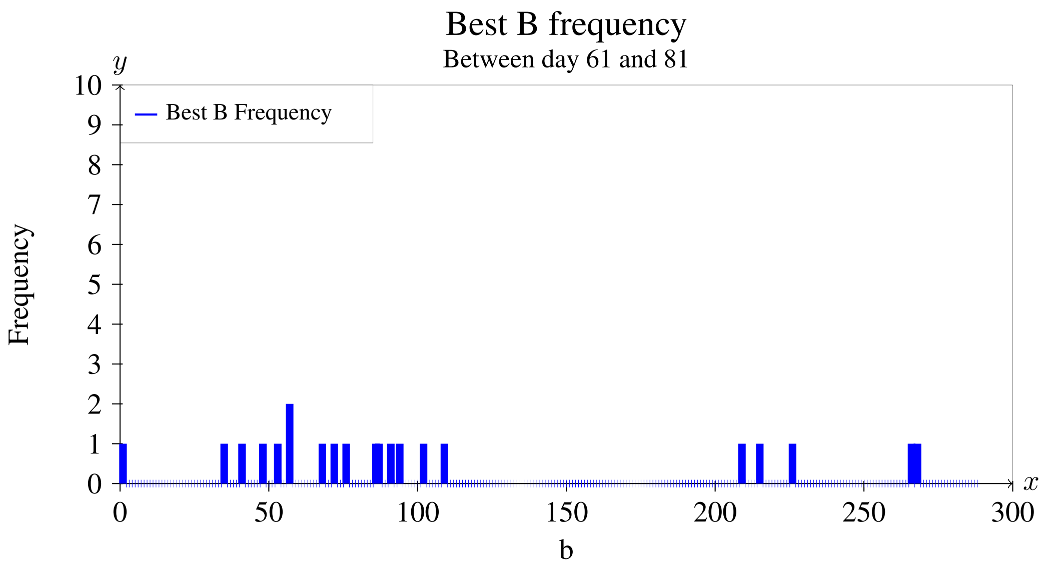


Figure 5.45: Frequency distribution of the best *beta* values from all days in scenario: 60 Days, in game Supreme Commander 2

The frequency distribution in figure 5.45 illustrate that there are more occurrences of the frequencies in the first range of *beta* values. In this area we see that the frequencies are close together. We also notice that there are some frequencies occurred during the period when the number of players tend to be at its highest peak. In only one case has the same *beta* value occurred as the best *beta* value twice. Since we clearly see an area where it is occurring far the most occurrences of frequencies, it may indicate that this area can yield good results for the algorithm.

| Best *beta* values per day | | |
|---|---|---|
| Day | Beta value | Average Distance |
| 1 | 268 | 178 |
| 2 | 48 | 188 |
| 3 | 87 | 967 |
| 4 | 102 | 66 |
| 5 | 91 | 77 |
| 6 | 86 | 125 |
| 7 | 76 | 60 |
| 8 | 57 | 69 |
| 9 | 53 | 65 |
| 10 | 209 | 92 |
| 11 | 215 | 73 |
| 12 | 68 | 95 |
| 13 | 226 | 114 |
| 14 | 72 | 77 |
| 15 | 266 | 104 |
| 16 | 1 | 76 |
| 17 | 94 | 52 |
| 18 | 109 | 60 |
| 19 | 57 | 112 |
| 20 | 41 | 146 |
| 21 | 35 | 77 |

This is a table of best *beta* value per day. These values are from the frequency distribution. Here we see in more detail why the *beta* value was best for the given day because of the average distance information. Here we see that all days except day 3 has a very low average distance at best *beta* value. This may tell us already that a profile based on 60 days provide good predictions.

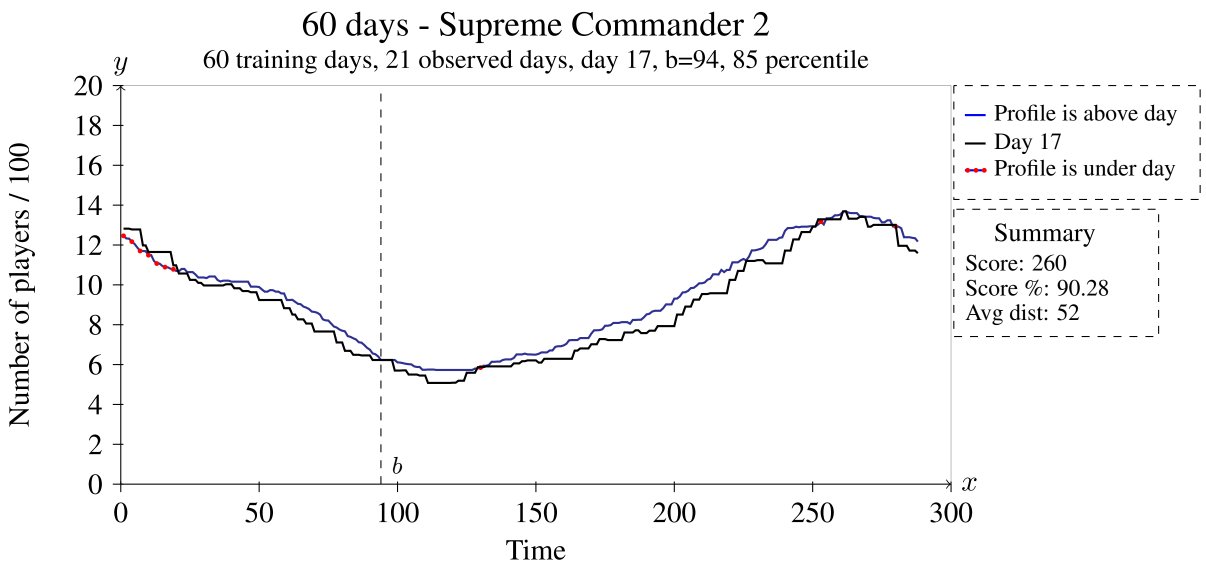


Figure 5.46: Results from the game Supreme Commander 2 of scenario: 60 Days, using *beta* value 94 on day 17

In this first figure 5.46 the best distance average with score over 90% is shown. Here we can see that the algorithm have drawn a curve that correspond very well to the observed day. The *beta* value that is used to normalize the profile to the day, is apart of the group with most occurrences close to each other as explained in the frequency distribution. One can also see that the profile remains slightly below the observed day in the beginning.

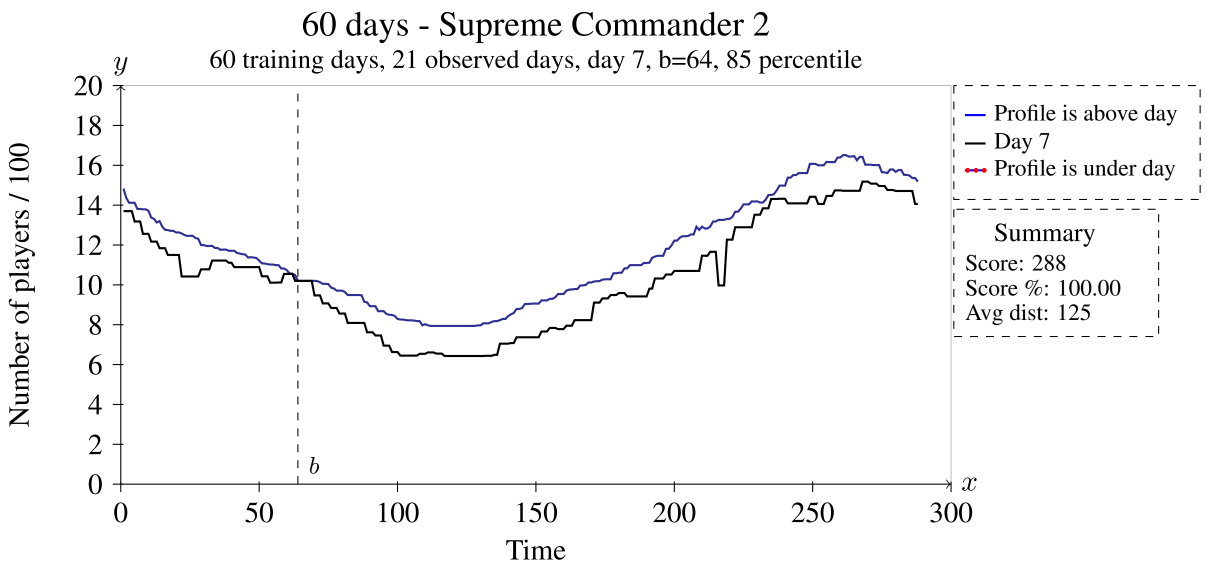


Figure 5.47: Results from the game Supreme Commander 2 of scenario: 60 Days, using *beta* value 64 on day 7

Figure 5.47 is an example where the profile always remain above the and contain close to the observed day with an average distance of 125. This is also shown by observing the graph. If you study the area where the *beta* value is placed. one can see that the observed day has a small increase in players in that time period. This is not a large increase, but it may be that this is the reason that the profile is able to remain above the observed day all the time.

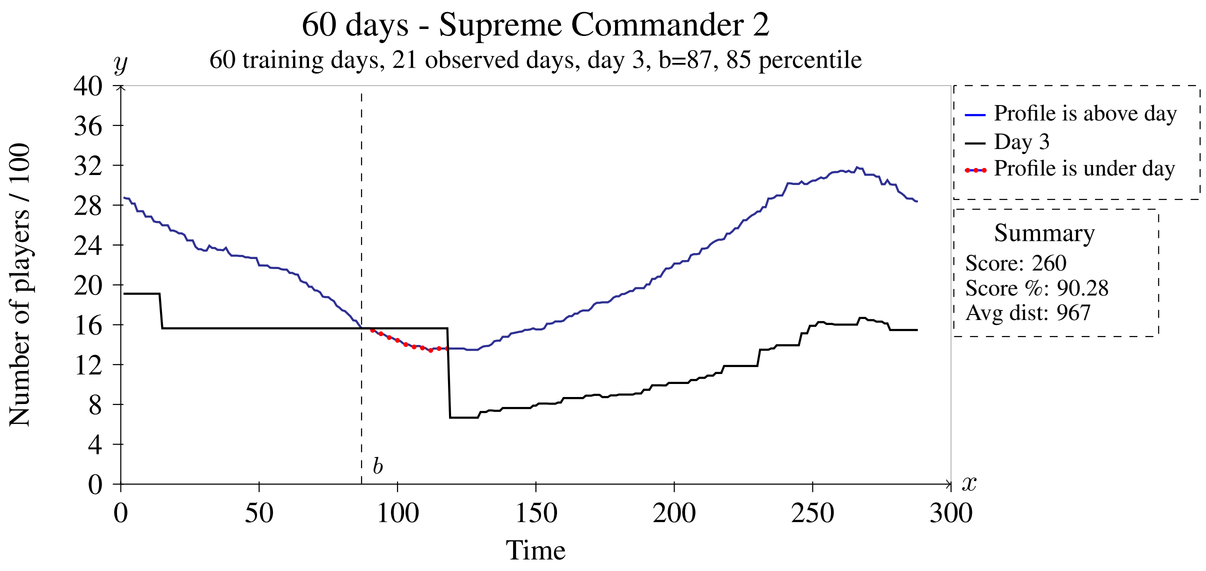


Figure 5.48: Results from the game Supreme Commander 2 of scenario: 60 Days, using *beta* value 87 on day 3

This next figure 5.48 shows another example of how repetitive data mislead the algorithm so that it create a much higher profile.

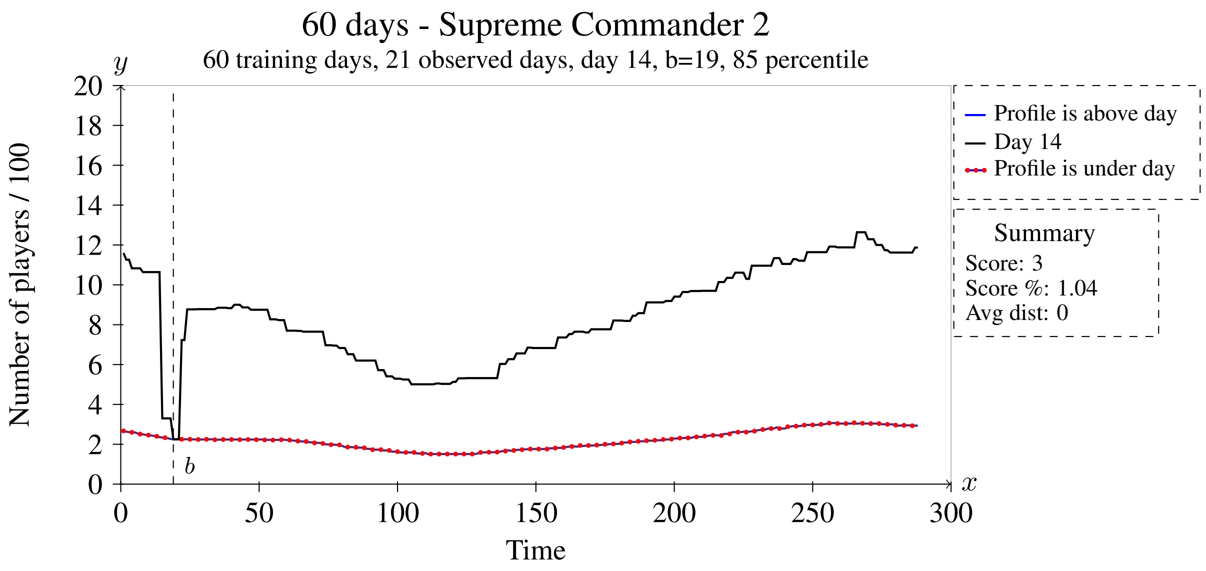


Figure 5.49: Results from the game Supreme Commander 2 of scenario: 60 Days, using *beta* value 19 on day 14

In figure 5.49 the *beta* value is located in area where the number of players is reduced. In this case, this causes the algorithm to predict that the number of players on average to be much lower than it actually is. This means that the profile remains far below the rest of the observed day.

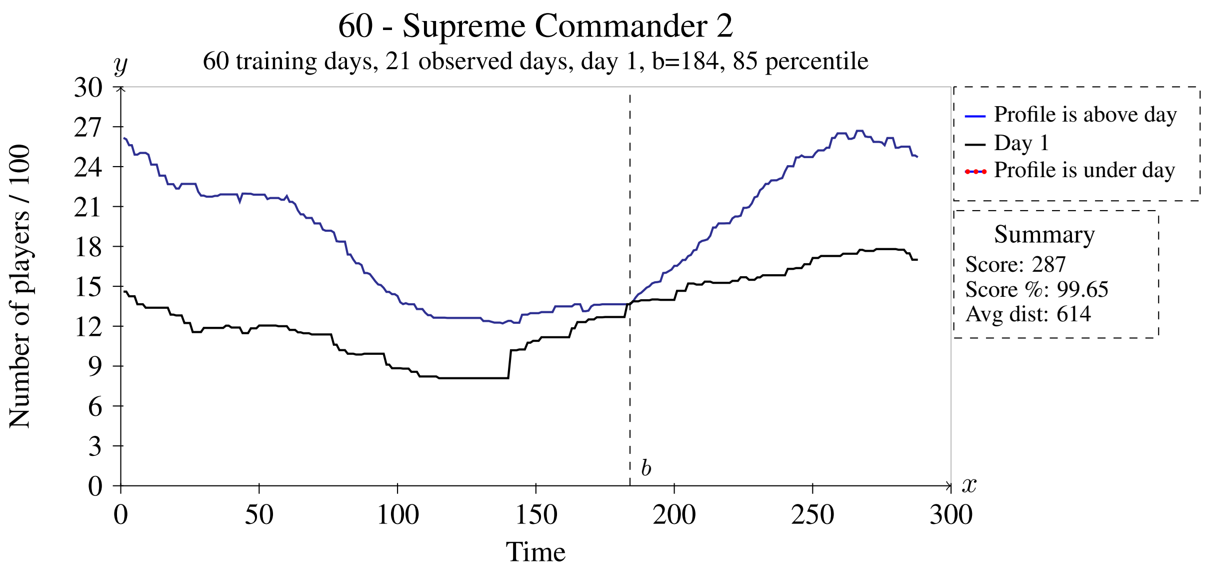


Figure 5.50: Results from the game Supreme Commander 2 of scenario: 60 Days, using *beta* value 184 on day 1

In the frequency distribution we got knowledge of where the best *beta* values for each day are located. The *beta* value used in figure 5.50 is not in that area. The *beta* value is located in the escalation of players coming online. We see here that the profile predicts that the day will have much higher values in the period before and after the *beta* value. When we study the observed day, we see that it has a slightly more slack curve compared with most other days. This may be the reason that the predication have a higher average distance.

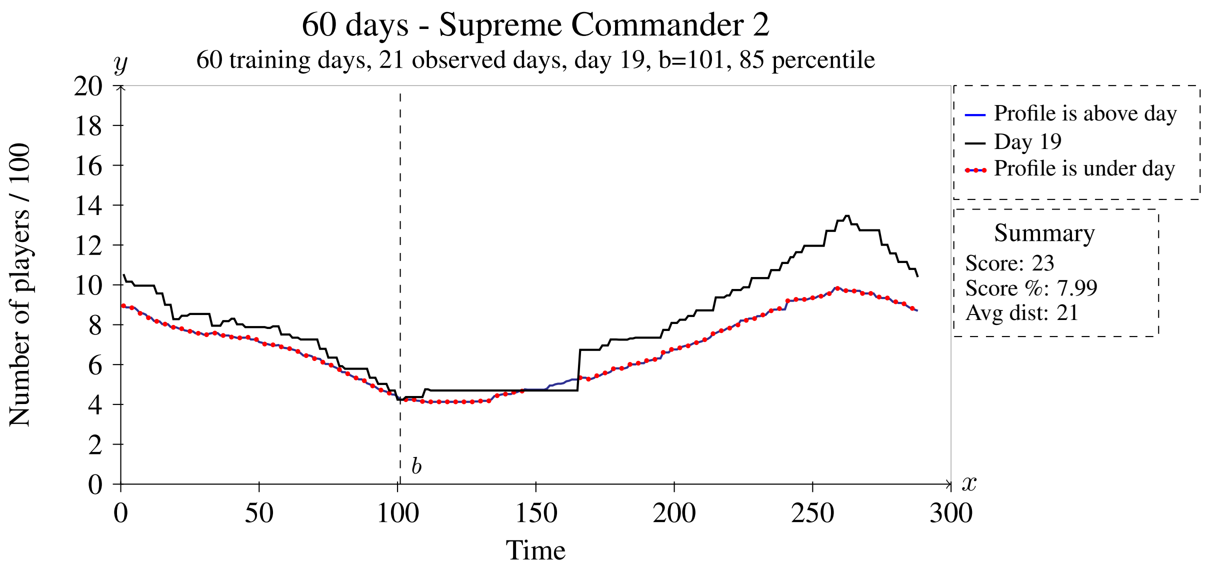


Figure 5.51: Results from the game Supreme Commander 2 of scenario: 60 Days, using *beta* value 101 on day 19

In the last figure we see that the algorithm has not been able to predict very well. We see that the profile remains below the observed day most of the time. The *beta* value that is used in this example is in the area where most occurrences of best *beta* value was found in frequency distribution.

### Summary

It was interesting to see how the algorithm behaved when it was tested on a day with different human behavior. That the day had a more even amount of players made the profile predicted with much higher value on the area where there is normally more players online.

# Chapter 6

# Analysis

In this chapter the findings from the graphical results on the game specific data will be analyzed.

The manual inspection of the test results provided a good understanding on how the algorithm behaves in different situations. The main finding is that the behavior of the algorithm depends entirely on the beta value on the observed day that the profile is normalized to. The following figures will give a more thorough explanation of the main findings.

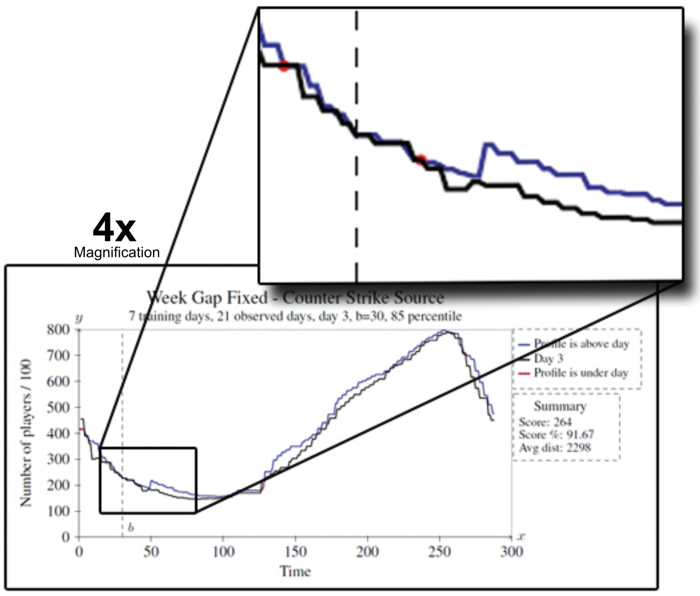


Figure 6.1: Magnification of the area around the normalization point on an accurate prediction.

Figure 6.1 have magnified a small area before and just after the beta value. What we see is that the data flow on the observed day seems very anticipated. This because the number of player value seems to be updated frequently. Because of this, we can assume that the profile has been normalized with a beta value that is updated with an expected value.

A trend that was very frequent, was that in situations where the profile is normalized to an expected *beta* value, the prediction usually tend to be good. These results show that the profile most of the time remain above the observed day and at the same time remained close. This indicates that the best predictability occurred when the algorithm is supplied with a correct flow of data and when the number of players does not have unexpected variation.

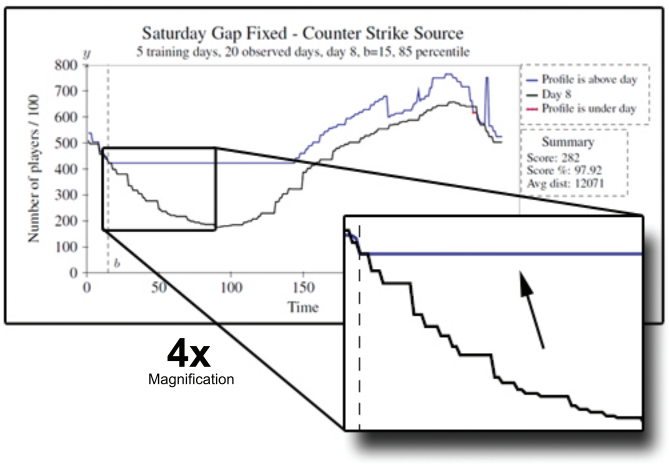


Figure 6.2: Magnification of strange profile behavior around the normalization point.

This next figure 6.2 enlarge an area that clearly show an error in the profile. The profile has been normalized to a beta value that seems anticipated. It would in this case be expected that the profile should be able to predict better than this. Immediately after the beta point, one can see that the profile creates a straight line. This behavior was observed frequently during the manual inspection of one of the scenarios testing Saturdays. A thorough manual check on the Saturdays used in this scenarios show that there was errors in the data. Several of the days had large periods with gaps of data loss. This concludes that a profile which is based on days that contain errors will predict worse. This enhance the importance of a correct data flow to increase the chance for a good predication.

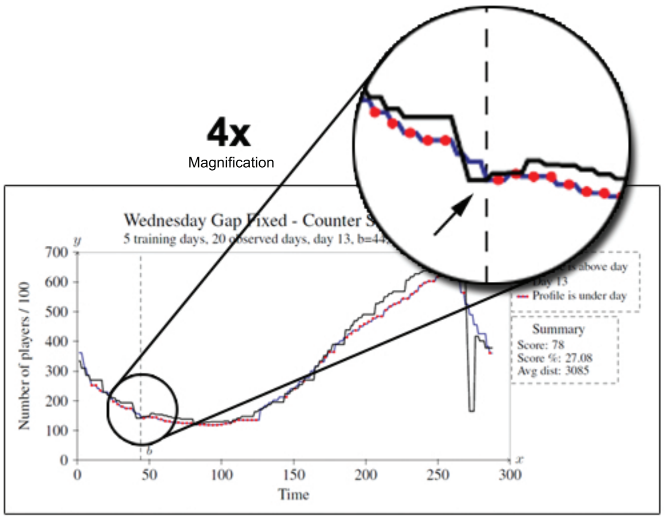


Figure 6.3: Magnification of small player drop at the normalization point.

Figure 6.3 highlights a area where a small drop or decrease in number of players have occurred. The results of normalizing the profile to a beta value located in a small decrease in players, show that the profile is not able to stay over the observed day. This is also a phenomenon that was observed in the manual inspection of the results. There can be several reasons for these drops, regardless, we see that the algorithm is affected by them.

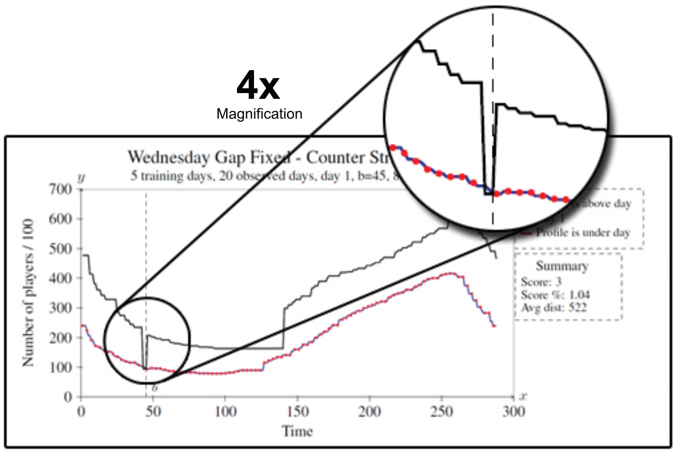


Figure 6.4: Magnification of increased player drop at the normalization point.

Figure 6.4 show an even greater decrease in players. This sudden decrease in number of players is more evident in this example. Here we see that this provoke the profile to drop even more than in the last example.

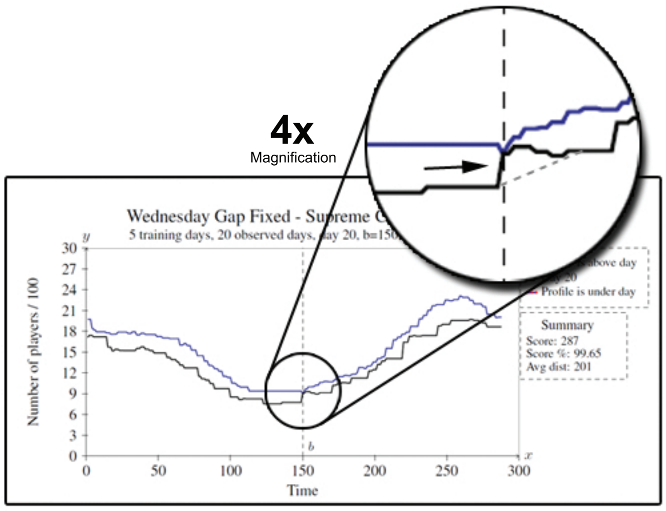


Figure 6.5: Magnification of small increase in players at the normalization point.

In this next figure 6.5 we can see that the opposite occurs. The arrow on the figure locates the small dump of increase in number of players. Since the profile is normalized at this position the profile rises slightly throughout the entire day. This is opposite results than the previous examples with drop in number of players. This strengthens the assumptions that the algorithm predicts best when the data flow is expected. It is unknown why these small drops in players occurs, it might conceivable that it is completely random.

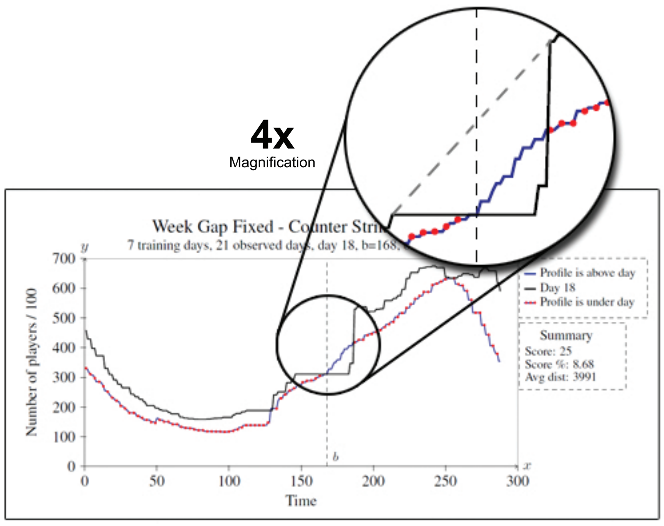


Figure 6.6: Magnification of outdated data at the normalization point.

This last figure 6.6 magnifies a area where data errors in form of outdated data have occurred. The phenomenon where days consists of periods of outdated data was observed very often during the inspection of the results. This is a result of Steam's disability to update the correct number of players with a constant interval.

When the profile is normalized at a time when outdated data have occurred, the prediction usually becomes totally wrong. On the figure we see that a dotted line demonstrates how the behavior might have looked like with a correctly updated number of players. The result of the outdated data makes the profile predict that the day have a lower number in players than it actually has. This usually cause bad predictions.

Throughout the manual inspection of the game specific results it was found that the reasons for accurate or inaccurate predictions was caused on a few set of findings. These findings have been explained in this chapter. Also, it is important to say that the inspection of results was ended when it was believed that the majority in important results was found.

# Chapter 7

# Twitter specific results

This chapter will present a modification to the initial approach, data from the information network Twitter was also tested. This decision was made because the majority of interesting findings in the game data seems to be found, and the curiosity of testing the algorithm on another type of repetitive data had emerged.

Information is shared on Twitter by writing tweets. A tool collects the amount of people that tweet using the keyword “beer“ every 5 minutes, equal intervals as in the retrieval of game data. This way, one will get repetitive data of how many people that tweets using the keyword ”beer“ over a period of time. This script has collected data for a few weeks.

## 7.1 Introduction

The algorithm have been tested on the twitter data in two scenarios. Scenario 1 have a learning period of 3 weeks and will be tested on the following week. Scenario 2 have a learning period of 1 week and will be tested on the following 3 weeks. The scenarios are different from the game data because the collection period is significantly shorter.

## 7.2 Scenario: 3 Weeks

In this section the scenario 3 weeks will be tested, this is the first scenario. The algorithm will have a learning period of 3 weeks and will be tested against the seven continuous days.

The following figure 7.1 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

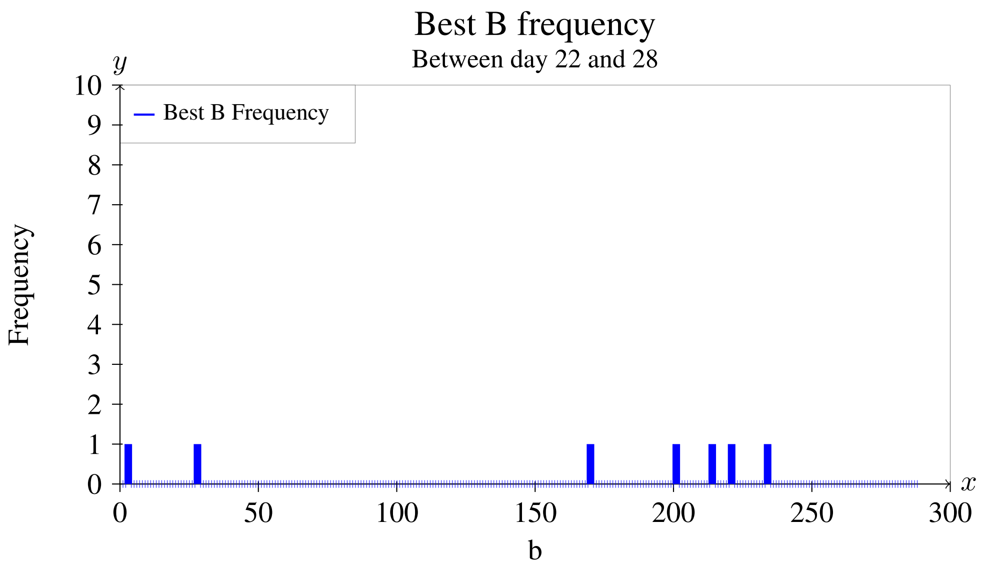


Figure 7.1: Distribution of the best *beta* values in scenario: 3 weeks, on the twitter data

In figure 7.1 we see the distribution of *beta* values with score over 260 and lowest average distance on the 7 observed days. We see that 4 of the frequencies have occurred in the area between *beta* value 170 and 240. This is the area where the peak of most players online are in the game data. Also two occurrences have occurred at *beta* value 3 and 28. This distribution looks similar to most of the distributions in the game scenarios, but in this case it is difficult to tell because the distribution is only on 7 days. The next scenario might provide more information if this is correct.

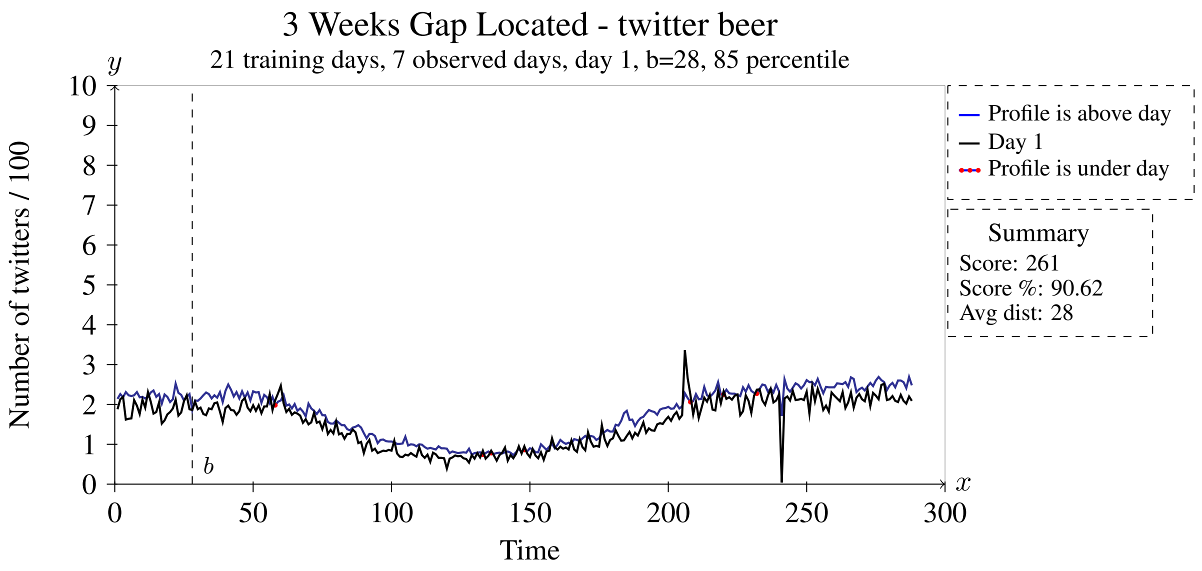


Figure 7.2: Results on the twitter data from scenario: 3 weeks, using *beta* value 28 on day 1

This first figure 7.2 show one of the best results from the distribution. Here the *beta* value 28 provides best results on day 1. With a score of 261, the profile remain above the observed day 90.62% of the time. An average distance of 28 indicates that the profile have remained close to the observed day. By observing the figure one can confirm this by noticing how close the profile remain the observed day. In the area around *beta* value 210, we notice a increase in tweets, the profile was not able to predict this random increase.

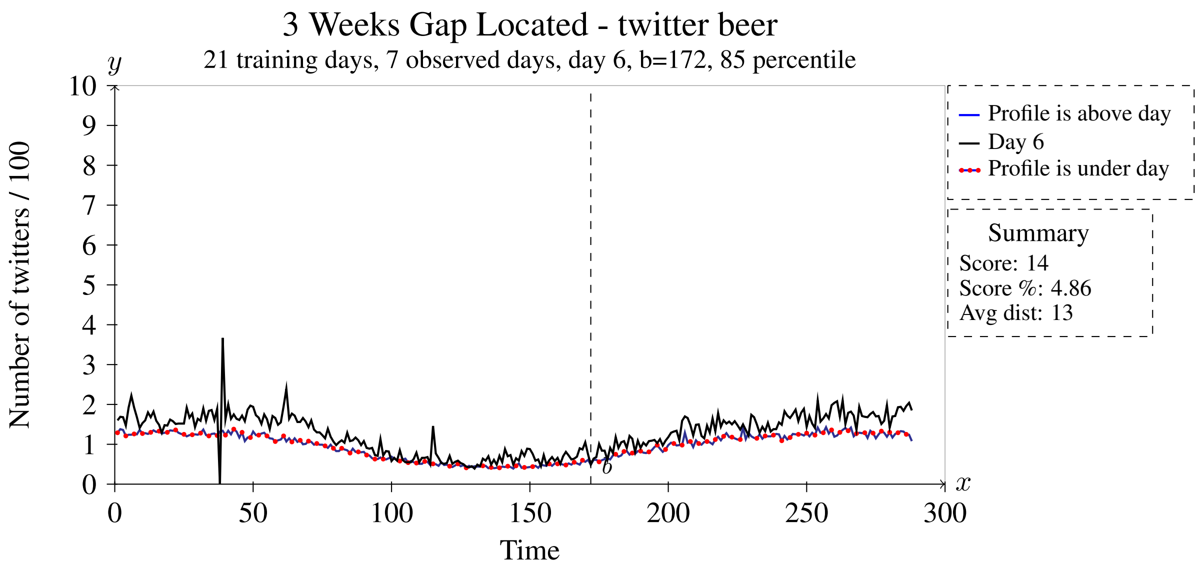


Figure 7.3: Results on the twitter data from scenario: 3 weeks, using *beta* value 172 on day 6

This next figure 7.3 show one of the results with lowest score. A score of 14 tell us that the profile was only over the observed day 4.86% of the time. The average distance of 13 is misleading because the profile does not calculate the distance when the profile is under the observed day. By studying the figure we notice that the profile does remain rather close throughout the day. An important notification is that it seems that the profile is normalized to a beta value that is located in a small decrease in tweets.

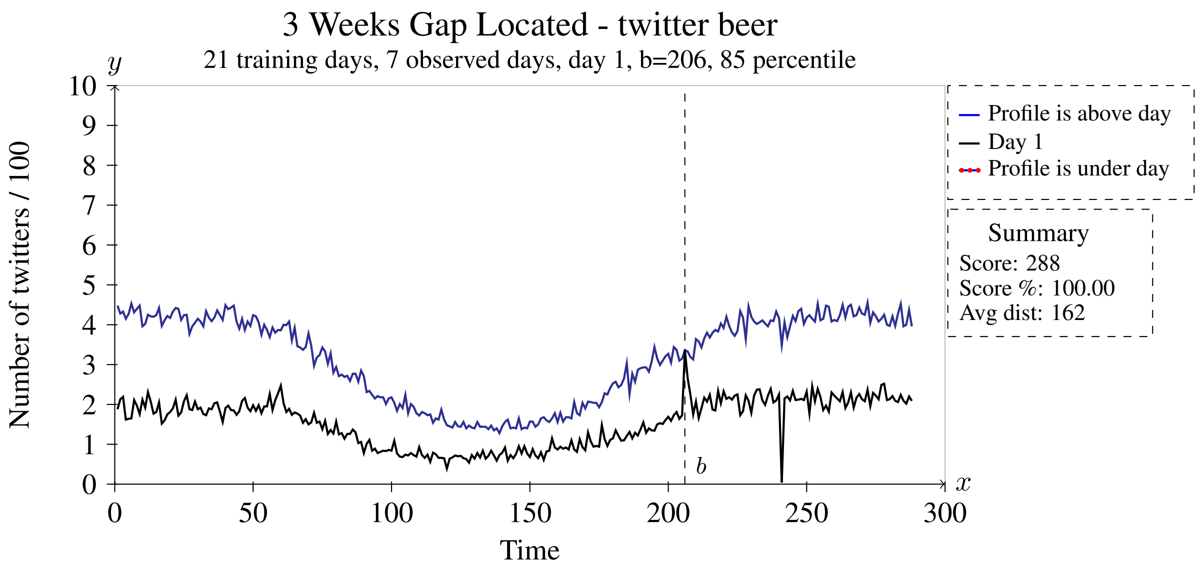


Figure 7.4: Results on the twitter data from scenario: 3 weeks, using *beta* value 206 on day 1

This next figure 7.4 show one of the results with maximum score. We see that the algorithm was able to predict above the observed day 100% of the time. The average distance of 162 indicate that the profile has not been able to remain as close to the observed day as the previous results have shown. By observing the figure we see that the profile have greater distance from the observed day than the other results in this section. We also notice that the *beta* value the profile is normalized on is located at a unusual increase in tweets in that area. This seems to be the reason for the high score and increase in average distance.

## 7.3 Scenario: 1 Week

In this section the scenario 1 week will be tested. The algorithm will have a learning period of 1 week and will be tested against the following 3 weeks. This scenario distinguishes itself from the previous scenario because it will find if there is a difference in using different length in learning period on this data.

The following figure 7.5 illustrates the frequency distribution of the best *beta* for every observed days. The best *beta* is a score over 260 with the lowest average distance.

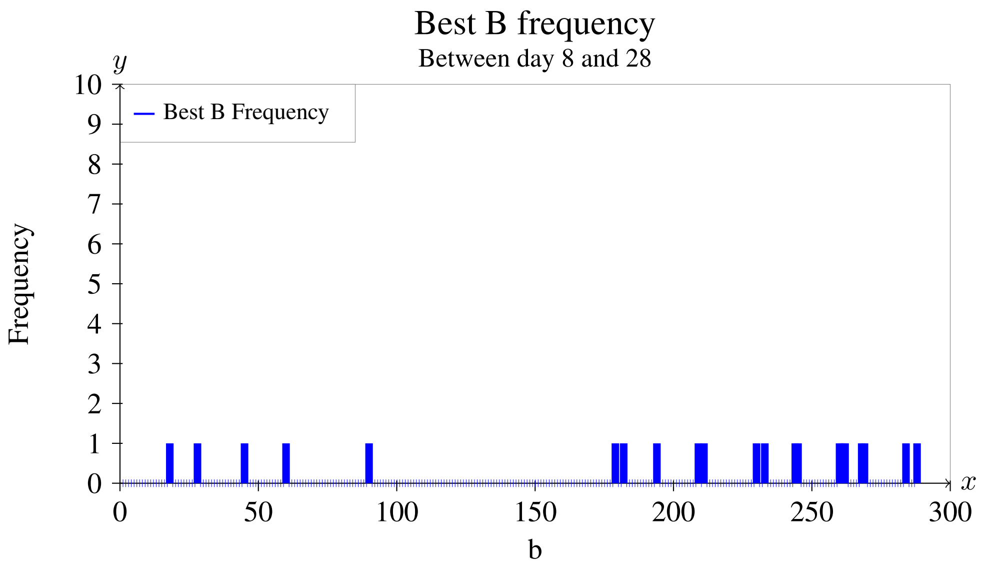


Figure 7.5: Distribution of the best *beta* values in scenario: 1 week, on the twitter data

This distribution is very similar to the distribution in the previous scenario. The apparent similarity is in the period between *beta* value 90 and 170. This area have no best *beta* values. It appears that this area seems to provoke bad predictions using the twitter data as in the game data. We also notice that none of the frequencies in the distribution have occurred more than once. But some of the frequencies in the area between *beta* value 170 and 288 seems to be very close to each other.

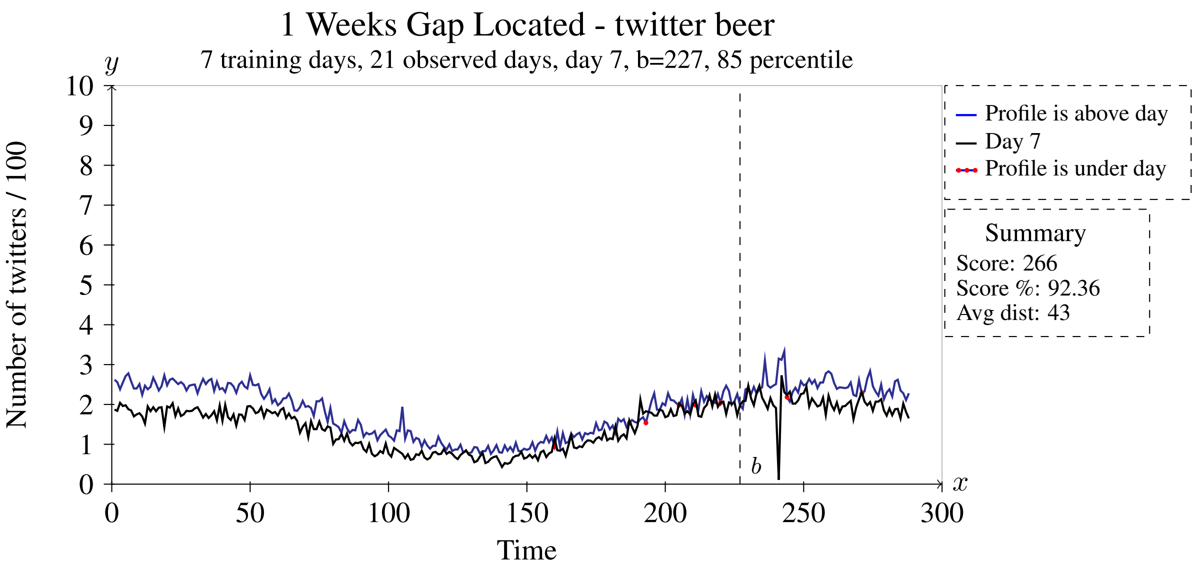


Figure 7.6: Results on the twitter data from scenario: 1 week, using *beta* value 227 on day 7

This first figure 7.6 show one of the best predictions with a score over 260 and lowest possible average distance. We can see in the figure that the *beta* value 227 is in the area where most of the best *beta* values where found in the frequency distribution. This example have a score of 266 that indicates that the profile have remained above the observed day 92.36% of the time. An average distance of 43 indicate that the profile have remained close to the observed day, which can be verified by observing the figure.

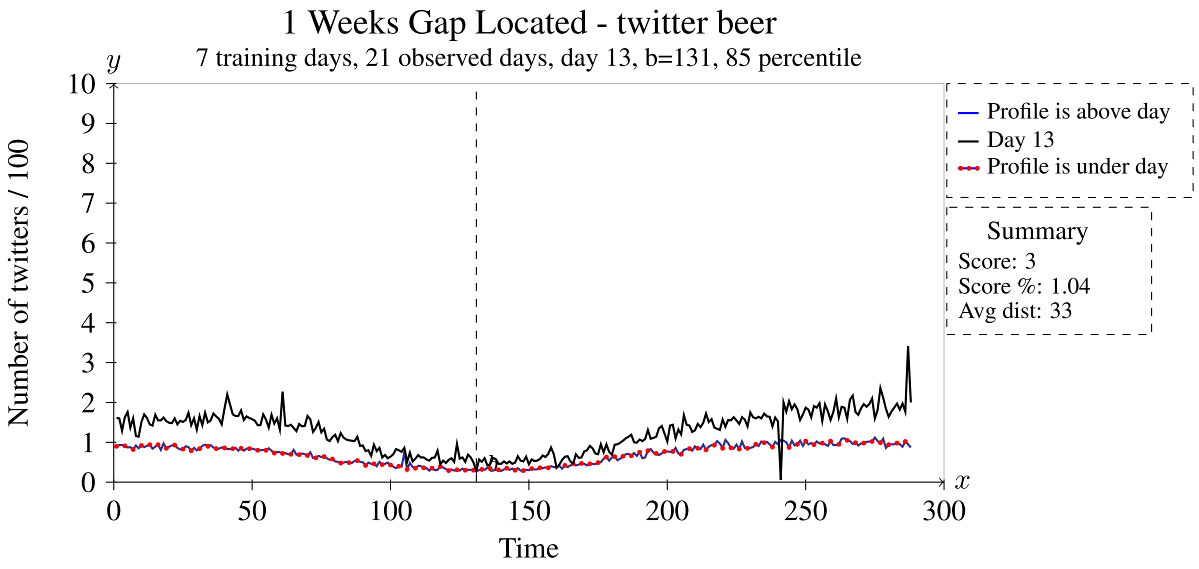


Figure 7.7: Results on the twitter data from scenario: 1 week, using *beta* value 131 on day 13

In this next figure 7.7 we see that the profile remain under the observed day almost all the time. A score of only 3 indicate that the profile have only occurred above the observed day 3 times. The average distance of 33 is misleading because the distance is not calculated under the observed day, but we can observe that the profile have remained under the observed day with some distance almost all the time. It might seem that the *beta* value used to normalize the profile is in a small decrease in tweets, but it is hard to tell. We also notice that the profile have less noise in this figure.

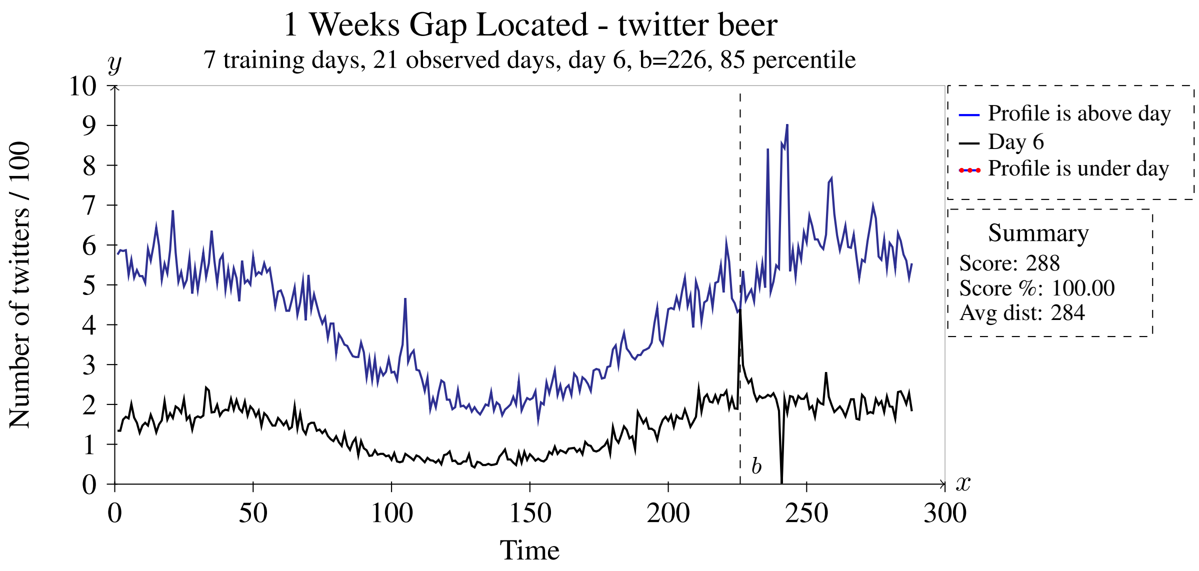


Figure 7.8: Results on the twitter data from scenario: 1 week, using *beta* value 226 on day 6

In figure 7.8 we see similar results to one of the cases in the previous scenario. Here the profile is normalized to a *beta* value with a unusual increase in tweets. The same outcome have occurred in this case, the profile remain high above the observed day all the time, resulting in a score of 288. The average distance is 284 which also indicates that the profile have greater distance than shown in the other examples. We also notice that the profile have more noise than usual in this case.

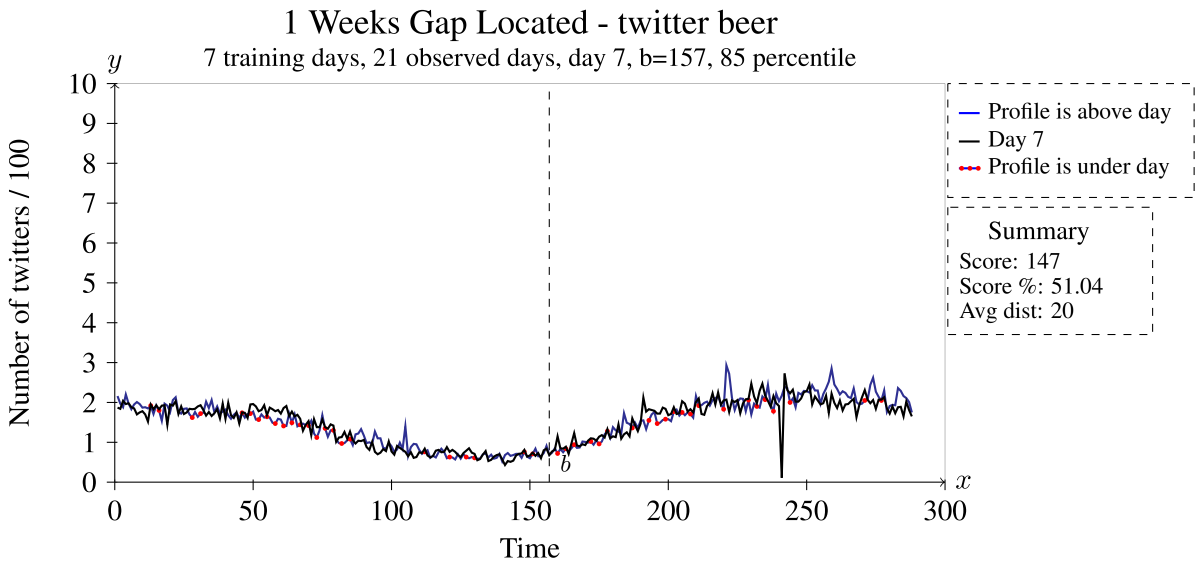


Figure 7.9: Results on the twitter data from scenario: 1 week, using *beta* value 157 on day 7

In this last figure 7.9 we see that the profile remain very close to the observed day, it is actually hard to tell by observing the figure if the profile is above or under the profile most of the time. The algorithm have managed to predict a almost prefect curve similar to the observed day. Some overhead in this case might result in a very good prediction and score. This case have a score of 147, this indicates that the profile have remained above the observed day 51.04% of the time.

## 7.4 Twitter Results Analysis

The first thing that come to mind when observing these results from the twitter data is the difference in data noise from the game data. It is easy to see that the twitter data has a more correct flow in data. We also notice that the time series have similar human behavior as the game data. Another important finding in the manual inspection of the graphical results is that it was really hard to find results where bad prediction was made. It is an absolute majority in good prediction where the profile have remained close and over the observed days. The curve the algorithm creates is very rarely different from the observed day. It was also very hard to find any differences between the results in the two scenarios, the only noticeable difference was in the profile curve noise. The noise in the scenario with a profile made by using 3 weeks had less noise than the profile made of 1 week.

The intuition in the game data was that the algorithm is very influenced by sudden changes in the data. This is more clearly demonstrated in some of the figures in these scenarios. Figure 7.10 and figure 7.11 illustrate this issue. We can notice that when the *beta* value is located in a small drop of tweets, the profile seem to decrease throughout the day and therefore remain lower than desired. If the profile is normalized on a *beta* value on a small increase of tweets, we notice that the profile usually remain over the observed day.

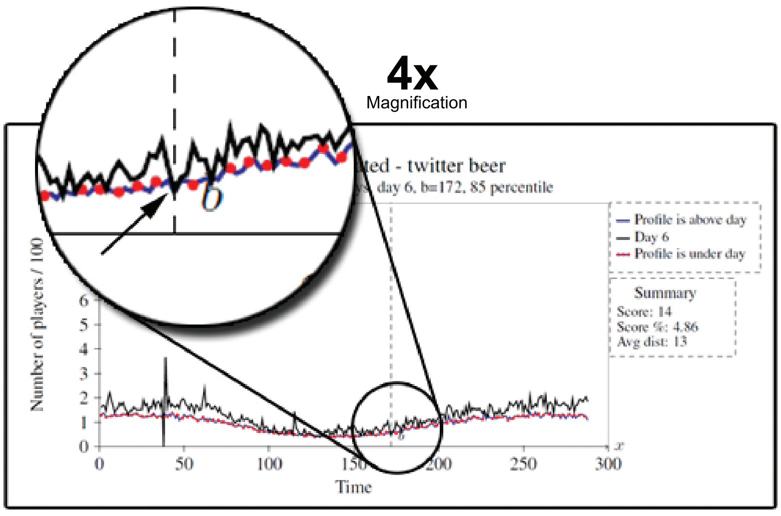


Figure 7.10: Magnification of a small drop in tweets at the normalization point.

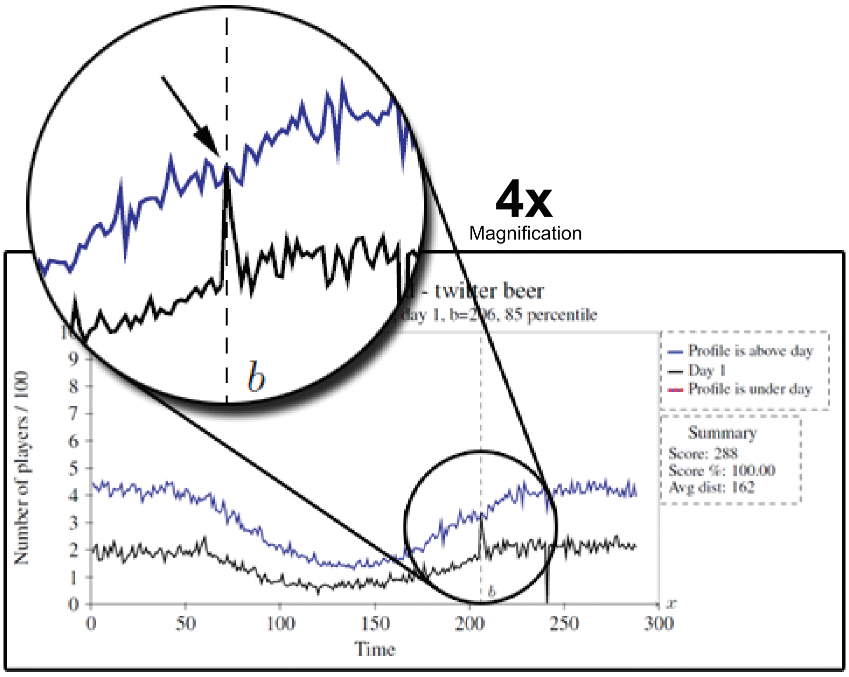
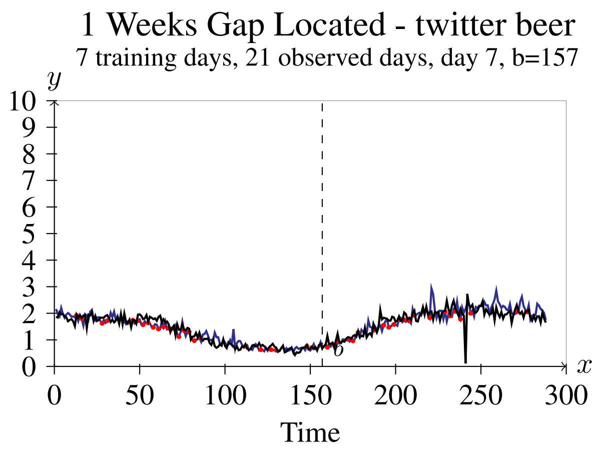


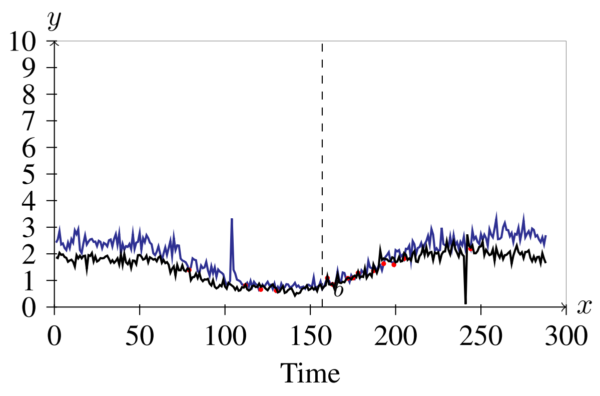
Figure 7.11: Magnification of an increase in tweets at the normalization point.

Since the predictability of twitter tweets are much more unpredictable than current online players in the game data, we notice that the algorithm have difficulties in picking up these unpredictable changes. This tells us that the way the algorithm is now, it has best accuracy in predicting data that is more predictable, such as the game data.

In one of the examples (figure 7.9) in scenario: 1 week, we noticed that the profile remained very close to the observed day and also draw a curve almost identical to the day curve. Therefor testing was performed with 95 percentiles as well. These results showed that the profile was able to increase the score and remain above the observed day for a prolonged period. This show that a higher percentile can provide better score. Both figures are shown under to illustrate the differences.



(a) 85 Percentiles



(b) 95 Percentiles

Figure 7.12: Results from 85% percentile and 95% percentile on twitter data from scenario: 1 week, using *beta* value 157 on day 7.

Figure 7.12 illustrate the results from day 7 using *beta* value 157 on 85 and 95 percentiles. The summery box is removed from this figure so the summery will be explained here. Using 85 percentiles the profile score 147 remaining above the observed day 51.04% of the time. While using 95 percentiles the profile have been lifted resulting in a score of 253 remaining above the observed day 87.85% of the time. This means that the change in percentile "lifted" the profile resulting in 36.81% better score.

# Chapter 8

# Discussion

This chapter will discuss various aspects of this project. From result specific discussions, as well as implications of the approach and design choices made. The discussion will lead up to the conclusion, which will provide a final evaluation of the work performed in this project. Future research in this field will be suggested along the way in this chapter.

## 8.1 The challenge of large-scale data analysis

Naturally, the usability is important when working with large amounts of data. The first obstacle that emerged was that the raw data had a impractical format. All the data were placed in a giant file with a somewhat difficult structure for the desired use. The big file size made it difficult for some file management software to run fluently. Working in this format would have cause some frustration and would require additional processing time. The data had to be transformed into a structure that is more readily available and faster to process.

The actual work of finding the appropriate format was important, the desired criteria was that the data would be easy to handle and fast. Sufficient knowledge of database proved itself important in this situation. A tool was developed to transform the data into a format comprehensible for the database. The data was then added to the database and was ready to be used.

The benefits of choosing a database are many, most important is that they support the criteria required in this situation. Databases are both fast and easy to handle. For continued or related work it is recommended to take advantage of the benefits that a database provides, especially when you work with large amounts of data.

## 8.2 Working with real vs. synthetic data

There are both negative and positive aspects of working with real data. Since the data in this project was real, a pre-analysis was conducted to determine the condition of the data. It soon became known that there were errors located in the data set. There were several different types of errors and these had to be handled accordingly. One of the errors were gaps of data loss, these gaps have different lengths and could therefore not be solved in the same way. A tool was therefore developed to address these data gaps.

The tool sealed the gaps where the gap had a short distance by using a linear mathematical method. This would not work particularly well in areas where gaps were of a longer length. Since these areas had significantly greater gaps another solution is needed. There was simply no time to seal these large gaps, but an idea of how this could be done was investigated. In periods where the day have large gaps of data loss transplantation could be the solution. For example, a Monday have a large part of the day with data loss, data from an earlier Mondays could be used to seal the data gap.

It was expected that these gaps of data loss was located in the data set. But it was unexpected that another data error of a completely different type of nature had emerged. This problem was about the flow of data that was extracted from the Steam application. The tool that extracts data from Steam, collects data every 5 minutes. Steam update its current online player status more often than that, but sometimes the current player values stays outdated over a long period of time. Why this happens is uncertain because this is on Steam's side, but it is conceivable that the feature that keeps track of how many people that play the different types of games, for some reason stop working so that the current values therefore stay outdated over a period of time.

This problem was unanticipated and created a difficult situation. This because the algorithm is dependent on correct flow of data for optimal performance, when data of outdated nature either become a part of the profile or is a part of the observed days, bad predictions occur.

What was learned from this? The positive effect of testing the algorithm on real data is that one is able to find errors that can occur in real-life situations. This required to perform additional preparations that would make it possible to overcome more of these errors if the algorithm is tested in the future. These errors would definitely not been found with synthetic data.

As a system administrator one does not always get the luxury of working with "good" data. It is a superior advantage to remain at a realistic point so that one acquires experience that will be necessary in the future. It is recommended that future work on this algorithm continues to be tested on real data.

The design of the algorithm was therefore very delicate. The algorithm should have the ability to adapt and use any kind of repetitive data. Finding or customizing “perfect“ data that would work optimally on the algorithm was never an option. This decision made the algorithm more applicable and optimized for work on any type of repetitive data.

## 8.3 Designing and developing a new predictive algorithm

The interest and desire to learn is absolutely a necessity when acquiring new type of knowledge. Many of the the challenges that occurred in this project was particularly difficult to overcome due to lack of knowledge. It was therefore especially rewarding to implement parts of the project that required new skills. Everyone have different kind of knowledge, and the authors area of knowledge is of a more technical nature. It was therefore a natural part of this project to acquire the needed type of knowledge to conduct a quality investigation on this field of study. The accomplishment of developing a predictive algorithm was therefore a personal achievement.

As mentioned in the background chapter, the idea behind the algorithm was in cooperation with Kyrre Begnum and Hugo Lewi Hammer. This cooperation enable the author to learn the work-flow in creating the predictive algorithm, this was the fundamental basis that made the development of the predictive algorithm possible. The process of developing the algorithm was obviously somewhat advanced, but it is with confident I can say that the actual hard work was the delicate process of creating a framework that would handle the large amount of data and rate the actual performance of the algorithm.

Since the design of the algorithm was somewhat clarified early in the project, the most challenging task in the development of the algorithm was to make it work as intended. Therefore, during the development a lot of debugging was conducted throughout the project period. It was absolutely critical to ensure that the algorithm performed exactly as intended. The size of the tools developed in this project are large and complex, this makes them very vulnerable. Small errors can have disastrous consequences for the results.

## 8.4 How to rate predictability?

There are already existing methods today that can rate predictability in data, for example, the correlation test was mentioned in the background chapter. However, this project created a simple scoring method to determine the predictability. In statistics, predictability is the degree to which a correct prediction can be made either qualitatively or quantitatively. In this project, a qualitative prediction is determined when the algorithm is able to predict equal to or above the observed day. A rating method for these specific criteria was not found in any of the researched methods. Therefore a decision was made to develop a scoring method to investigate the predictability, and leave the already existing measurement methods as a suggestion for measuring the data in the future.

The scoring method gave an simple indication of how well the algorithm performed and may resemble as some kind of benchmarking results. Although the scoring method gave more general understanding of the results, there were still some questions unanswered about how the algorithm actually performed.

A simple score was not enough, the need to visually *see* how the algorithm performed became a more natural solution to this problem. This was unexpected, suddenly there was a need to create huge amount of graphical results. There are many ways to create graphical results and there are plenty of software that do this today. But in this case, huge amount of graphical results had to be made and it would therefore be an impractical process at best to create these graphs manually by using a plotting tool. It was therefore important to integrate a tool that could create large amounts of graphical results in an organized fashion to the test framework.

A tool to create graphical results was developed by combining Perl programming with Tikz programming. Tikz makes it possible to generate graphics directly into TeX. Tikz works almost the same way as an arbitrary graphical program, the biggest difference is that one have to learn the Tikz programming language. Learning this new language and develop this tool is very time consuming, but the necessity and curiosity to learn Tikz resulted in a completely new plotting tool.

The design of the project's graphic results is therefor customized especially for this project. By developing the tool in Perl, the process of plotting became automated. This way, the graphical results are automatically generated each time the algorithm is tested against the data. This method made it possible to create graphical results of the algorithms behavior on all the various days in all the scenarios.

This tool can be used in many situations. The Plot tool is developed so that it only requires 2 arrays that represent the data that is to be plotted. The size of the plot can be defined, the rest is scaled dynamically based on the size of the data in the array. The tool can be found in the Appendix and is free for non-commercial use. Future work that would require multi creation of graphical results are recommended to use this tool.

The results were now in a completely different form than originally anticipated. This opened for a huge and interesting analysis. The amount of results became much greater than foreseen. Graphical results were made for each beta value on every observed day for all the scenarios. Altogether, approximately 144,000 graphical results, and that is disregarding the results of different use of percentiles.

The desire and interest of analyzing the graphical results, reduced the number of scenarios that initially were to be analyzed. The process to analyze each selected scenario was still an amount that required many hours of investigation. Each scenario had approximately 6000 graphical results, which were examined to find anomalies, reasons and interesting findings using data mining techniques. The plotting tool creates the results in a pdf format, displaying one plot on each page. This allowed for easy access to the results and made it possible to create short film clips on how the algorithm transforms throughout a day.

This process formed most of the result chapter. The advantages of performing a manual inspection of all results is that one get a complete overview of how the algorithm behaves. This provides a genuine understanding of the algorithms ability to predict and was essential to investigate the predictability in the game data. The drawback is that this process takes huge amount of time, but it was definitely worth it. This process provoked interesting findings that would not have been found without a manual inspection.

Explicit or implicit, record similarity is a fundamental aspect of most data mining algorithms [1]. For traditional tabular data, the similarity often measured by value similarity or equality. For more complex data, for example time series of resources used by a system, such simple similarity measures do not perform very well.

For example, assume we have three time series A,B and C, where B is constantly 5 values above A, whereas C is randomly 2 values belove or above A. Such simple similarity measurements would rate C as far more similar to A than B, whereas a human expert would rate A and B as very similar because they have the same shape. In this project the idea was not only to predict similar to the observed day but also above or equal. So in the example above, remaining with a constant value above the observed day is good results while random values above or belove is looked upon as ”bad” results.

In retrospect after manually inspect the huge amount of graphical results a question have come to mind. What is good results? Many graphical results was found with a low score but remained very close to the observed day. Using the algorithm as a tool to dynamically adjust the resources on game serves, some of these “bad“ results would be able to provide enough resources although it does not remain above the observed day all the time. Shouldn't these results also be looked upon as good? It might be a topic for future work to modify the scoring system to score based on the ability to predict the resources needed instead of scoring based on a higher value than the observed day.

Also, the score system developed in this project only calculate the average distance when values are above the observed day. Because of this the average distance will not be calculated in periods where the profile value is under the observed day. In retrospect we see that this might give a wrong impression of the prediction based on the score value. For example, if a prediction is given a score of about 150 and an average distance of 1000, the average distance is only calculated on about 50% of the observed day. It might be that the real average distance is less or more based on the behavior of the profile in the period where the profile remain below the observed day. This makes it difficult to compare results with differences in score.

For example, lets say that results A have a score of 260 and average distance of 1000, while results B score 288 and have a average distance of 1500. Results B have remained above the observed day 100% of the time while results A have remained above 90% of the time. The remaining 10% of results A is under the observed day so the average distance is not calculated on these distances. It might be that the remaining 10% is far belove the observed day and should therefor have a higher average distance. In this case, a numbers like score and average distance does not provide enough information to say whether the prediction was "successful" or not.

In retrospect one might say that the scoring method and the calculation of the average distance could have been developed differently, it might be that also calculating the average distance when the profile is under the observed day, will provide a better numeric understanding of how well the algorithm performed. These proposals of changes will be passed on as future work.

It is with full awareness that the scoring method calculate the score based on all the 288 points on the observed day, although the profile is normalized with a beta far into the observed day. This means that the score is also decided by time of day that have already passed. For example, if the profile is normalized to beta value 150, the scoring method also provide score to points in time that have occurred before beta value 150. This is done to keep the analysis as straight forward as possible.

A real-life situation would not score back in time. Because testing was done on one day at a time it would be a poor basis for comparison on how the different beta values would predict if the scoring method would only considered future data. It could be a suggestion for future work to base the results only on future data, this would be more realistic. Another suggestion would be to calculate a score from the normalized beta point to X hours in the future. This way one could get a better picture over how well and how far the algorithm is able to predict.

## 8.5 Predictability in game resource data

The main focus in this project have been on developing the predictive algorithm, creating a test framework and producing results to determine its ability to predict. It might seem that the topic of finding the predictability in game resource data has somewhat faded in the shadow of the algorithm. This is not the case. The key in determine the predictability in the game data, depend on the results from the algorithm. The focus on the algorithm was this projects way to find the predictability in game data. Naturally one can question why an already existing predictive algorithm was not used to find answers faster and have more time to analyze the results. However, the foundation of this project was build on an idea of a new type of predictive algorithm and the curiosity of its ability to predict. Also, developing an algorithm for this type of data somewhat force you to sit down and learn about it. As a learning experience this has been very valuable because there is no better way to learn about data than making an algorithm trying to predict it. Therefore, this project answers to the predictability in game data is based on results from this newly developed predictive algorithm.

Testing the game data on the algorithm has shown a various specter of results. Both to be considered as good and bad predictions. First, it is fairly impossible to conduct a perfect prediction using the algorithm on repetitive data. A perfect prediction would only occur if the whole learning period would be identical to the observed day. So, how predictable is the game resource data? The results from the testing have provided answers based on various factors. In this project a successful prediction have been decided by these criteria.

1. The profile must be equal to or above the observed day more than or equal to 90% of the time.
2. A manual inspection of the graphical result must show that the distance between the profile and the observed day is somewhat constant and evenly similar throughout the day.

In contrary, a failed prediction is determined when the above criteria is not met. With this taken into consideration, the majority of the results has shown a successful prediction among all of the games that where tested. This indicates that one can say that game resource data have proven to be predictable in many cases.

Although the majority in results are categorized as successful, the remaining results is still of a large extent. These results failed for several various reasons:

1. Outdated data
   (a) If the profile is based on large amount of outdated data, the chance to predict decreases.
   (b) If the observed day have periods of outdated data, the chance to predict will decrease if the profile is normalized to a beta value that is located slightly into this period.
2. Gaps of data loss
   (a) If the profile is normalized to a beta value where data loss have occurred, the prediction will be completely wrong.
3. Unexpected change in the data
   (a) When the profile is normalized to a beta value that is located when a unexpected decrease in number of players suddenly occur, it usually results in a bad prediction. These drops can occur because of:
      i. unpredictable random human behavior
      ii. game updates that require server restart.
      iii. hardware failure.

This indicate that although the game data can be looked upon as predicable, the variables above can drastically influence the predictability in the game data.

Some modifications in the algorithm would eliminate the adverse reasons that induce unwanted results. The graphical results reflect that the profile indeed remain close to the observed day in most of the cases, but it would be optimal in this project's point of view to always remain above or equal as well. The problem with prediction based on previous data is that it is not guaranteed, but it is however possible to make measures to increase this chance drastically.

Profile padding is a modification that could fix some issues. Padding would increase the value of point beta. This would normalized the profile to a higher value and would therefore “lift“ the profile. This would increase the chance of getting a better score, but also increase the chance of a greater average distance. The size of the profile padding will therefor be of great importance. Excessive padding would maintain the profile high above the observed day, but would go against the initial plan to maintain close to the observed day. Profile padding would probably eliminate some of the bad performance results when beta values are located on drops in players the observed day.

## 8.6 Manually Inspection of Results

In this project the results has been provided in numeric and graphical form. Based on a manual inspection of all the results, only a small amount of graphical results has chosen to be displayed in this thesis. This approach require the reader to thrust the project to perform a quality and thorough examination. If the goal was to find a accurate indication of how predicable the game data is, which probably would be needed in a comparative study, a more mathematical correct approach would be suitable. However, this project is about investigating the predictability in data by developing and implementing an predictive algorithm. It was therefore a natural decision to investigate by manually inspecting the performance of the algorithm, and at the same time determine the predictability based on these observations.

## 8.7 Comparative analysis is needed

It would make sense that the next step would be to compare this algorithm with existing algorithms. This would have been a natural part of the project with an extended project period. With the time constraints to this project, a comparative analysis would most likely have removed the focus of data mining on the results and direct all of the attention to the differences between the algorithms.

The deep understanding and knowledge of the algorithm's behavior would most likely not be found. We can also envisage that the plotting tool would not have been developed. The interest and curiosity to analyze the graphical and numeric results, searching for patterns and statistical reasoning to the algorithms behavior, caused a natural conversion on this project to focus on data mining.

When it comes of further investigation of the algorithm, a suggested approach would be to implement other algorithms on the already developed test framework created in this project. This would provide equal data-flow to the algorithms and provide results both in graphical and numeric form. Similar results would also increase the chance to see differences in the behavior of the algorithms. In retrospect, it is conceivable that the approach used in this project opened for a easy way to compare algorithms in the future.

## 8.8 Type specific profile for human/service interaction?

The algorithm was tested with a varying number of learning days. A large learning period turned out to perform slightly better than a profile with less learning days, but it was surprising how well a small number of learning days performed. We saw for example the scenario where the algorithm had a learning period of 60 days had similar results as scenarios with a learning period of only 5 days. Why is this?

A very interesting phenomenon was observed by studying the graphical results. It seems that the curve the human behavior creates is somewhat different for the various games. The structure of the curve seems to be very similar in each game, but not necessarily numerically equal. This suggests that the key to better predictability perhaps lies in finding the “perfect“ curve to a specific type of game. Could this mean that one can transfer data from one type of game to another game of the same type?

If this is the case, it would mean that repetitive data could be predicted without using previous data knowledge. It would therefore be enough to predict the repetitive data by categorizing it to a certain specific type of profile. For example, football games would have one specific profile type, and strategy games would have another specific profile type. Future work could investigating this by using the predictive algorithm to create type specific profiles and test it on other data categorized as similar data.

Different human behavior in different types of games indicate that there also is a difference in managing server resources in different games. Lets use an example. Lets say that as a system administrator you want to dynamically allocate resources based on the current need. This because you want to provide enough resources to keep the players happy, and at the same time save the environment for unnecessary use of power. So you want to implement a algorithm that turn on and off servers based on the current need.

By implementing a traditional reactive algorithm the amount of servers running will be decided on the actual amount of players that is already connected to the servers. If a change in number of players occur, the reactive algorithm decide whether it is a need to start a new or stop a running sever.

This method works to a certain degree, but if a game have a human behavior that suddenly require the algorithm to turn on many servers, the server might not boot fast enough, resulting in a period of time where players either get bad gaming experience or is unable to play that game in this period. To prevent this, the algorithm in this project have been suggested, but also a type specific profile could work simultaneously with the reactive algorithm overriding the reactive algorithm if it predicts that the above scenario might occur.

## 8.9 Anomaly detection?

After seeing how the unexpected errors like gap of data loss and outdated data affects the profile a question have come to mind, can the algorithm be used for anomaly detection? The results has shown that the score and the average distance in cases where the profile have predicted completely wrong because of normalization on abnormal data, is very different from results normalized on normal data.

Could it be possible to use this information to locate the abnormalities based on the great differences between the profile and the observed day? If this is possible, the algorithm would be able to use already gathered information to detect abnormal behavior in the data and act accordingly. This could actually make the algorithm able to detect errors like outdated data. This would not only significantly improve the algorithms ability to predict but also make the algorithm able to automatically detect errors. This topic is certainly interesting and will clearly be a suggestion for further work.

## 8.10 Tools developed

Many different tools have been programmed during the project period. Some of these tools have become a natural part of each other to serve a mutual task while others are individual that makes simple but necessary tasks. Most of these tools have been standing in the shadow of this report, but have served as a absolutely necessary assignment to automate processes that would take very long to do manually. These tools are listed belove and found in the Appendix:

| Name | Description |
|---|---|
| TestFramework.pl | The script including the algorithm, the score method and the data processing. This tool can be use to perform the algorithm. |
| ConvertToBaseFormat.pl | A script that converts data from the data fetching script to a format understandable by a database |
| AddToBase.pl | A script that adds data to the database |
| Tikzplot.pl | A script that create graphically results based on arrays with data. |
| GetScore.pl | A script that calculate a score based on the differences between two arrays |
| GetDay.pl | A script that provide data from specific day combinations from the database |
| Normalizer.pl | A script used to normalize the profile to the observed day |
| GapAnalysis.pl | A script that create graphical results over gaps in data. |
| CrateLatexTable.pl | A script that creates a table over the best beta values from repetitive data. |
| ProfilePlot.pl | A script to plot a profile. |

## 8.11 Algorithm applicability and availability

So, how far away is a real-life implementation? In the introduction it was emphasized that it is very important with enough resources available when it comes to online gaming experience. Implementing the ability to dynamically allocate sufficient resources in real-time using the developed predictive algorithm has not been done because of the time constraints. However, the road to enable this feature is not long.

Looking back at the game data, the number of players is looked upon as a resource demanding variable. By simply satisfying this variable one can determine how much resources that is needed. For example, if a servers capacity is constant, it is easy to use the algorithm to determine how many servers that is needed to satisfy the resource demands. This way the algorithm can turn on and off servers in advance to always provide enough resources for the players. The implementing of this feature would be an interesting topic for future work. Although the results from the testing showed that is was possible to predict the game data, it would be even more interesting to study the algorithm in a real-life situation.

This study has opened for a new approach to predict the resource usage on game data. The predictive algorithm has proven to be successful and have provided many interesting topics for further development and testing. The algorithm is designed so that it will work regardless of platform. This makes the algorithm very easy to implement. The algorithm is available for everyone, all the tools and scripts created for this project is open source and free for non-commercial use.

The algorithm requires no calculations prepared to work. There are only a few parameters of data flow that is necessary. There are also no requirements on type of service, online gaming was merely a chosen data type for this study, any other service with repetitive data can be used.

## 8.12 Testing on Twitter data

Well into the testing of the game data the curiosity of testing another type of repetitive data emerged. The test results from the game data became monotonous. The reasons for good and bad predictions proved to be repetitive, it seemed that the majority of interesting findings was found.

The idea was to see how the algorithm performed on data with more noise. A new data fetching script was therefore developed to gather information about tweets on the online information network Twitter.

This script gathered information about how many people that tweets with specific keywords every 5 minutes, same interval as the game data. The keyword "beer" was used in this project. After a few weeks, repetitive data on how many people on twitter that have tweeted using the word beer was collected in the same way as the game data. The testing scenarios was not similar as the game data because the data was only collected over a few weeks.

The results from the twitter testing highlighted some of the similar findings from the game data, no new findings was made. The twitter data had much more reliable flow of data, this prevented bad predictions caused by errors. A more reliable flow of data showed that most of the predictions was similar to the observed day. Since the twitter data had more noise than the game data, the analysis highlighted what happens when the profile is normalized to low and high values in the noise. When normalized to a low value, the profile usually remain under the observed day, vice versa with a high value. In the majority of tests, the profile have a similar curve as the observed day.

# Chapter 9

# Conclusion

The goal of this project has was to *investigate the predictability of game server resource usage by developing and implementing a predictive algorithm.*

During the project time period following main tasks have been completed:

- Data has been prepared for testing.
- A test framework has been developed.
- A new predictive algorithm has been developed.
- The predictive algorithm has been implemented by testing it on different kinds of repetitive data.
- Results has been produced in graphical and numeric form.
- Results has been analyzed.

The algorithm has shown the ability to predict as initially intended and can therefore be used to investigate the predictability in game server resource data. If the question is whether the data is predictive, the answer from this project is that using simple statistical methods an algorithm can be developed which to a greater extent is successful in its prediction. in conclusion, the findings support the initial assumptions that the data indeed is predictable and opens for clear applications for resource management.

# Bibliography

[1] *Principles of data mining and knowledge discovery*. 1999.

[2] G. Armitage. An experimental estimation of latency sensitivity in multiplayer Quake 3. In *Networks, 2003. ICON2003. The 11th IEEE International Conference on*, pages 137–141. IEEE, 2003.

[3] K. Begnum and M. Burgess. Improving anomaly detection event analysis using the eventrank algorithm. *Inter-Domain Management*, pages 145–155, 2007.

[4] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The effects of loss and latency on user performance in unreal tournament 2003®. In *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pages 144–151. ACM, 2004.

[5] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing sla violations. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, pages 119–128. IEEE, 2007.

[6] M. Bray. Review of computer energy consumption and potential savings. *Dragon Systems Software Limited (DssW), December*, 2006.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):1–58, 2009.

[8] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, and R.P. Doyle. Managing energy and server resources in hosting centers. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 103–116. ACM, 2001.

[9] K.T. Chen, P. Huang, G.S. Wang, C.Y. Huang, and C.L. Lei. On the sensitivity of online game playing time to network QoS. In *IEEE Infocom, April*. Citeseer, 2006.

[10] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. How sensitive are online gamers to network quality? *Commun. ACM*, 49:34–38, November 2006.

[11] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 303–314. ACM, 2005.

[12] Christopher Clifton. Encyclopedia britannica: Definition of data mining, 2010.

[13] W.G. Cochran. Some methods for strengthening the common $\chi$ 2 tests. *Biometrics*, 10(4):417–451, 1954.

[14] Ian Whalley James E, Jeffrey O. Kephart, and Hanson Malgorzata Steinder. Intelligent workload consolidation with power and performance tradeoffs.

[15] E. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. *Power-Aware Computer Systems*, pages 179–197, 2003.

[16] W. Feng, D. Brandt, and D. Saha. A long-term study of a popular mmorpg. In *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 19–24. ACM, 2007.

[17] W.C. Feng. Making a case for efficient supercomputing. *Queue*, 1(7):54–64, 2003.

[18] G.B. Folland and A. Sitaram. The uncertainty principle: a mathematical survey. *Journal of Fourier Analysis and Applications*, 3(3):207–238, 1997.

[19] Gartner. Gartner estimates ict industry accounts for 2 percent of global co2 emissions, 2007. http://www.gartner.com/it/page.jsp?id=503867.

[20] S. Geisser. *Predictive inference: An introduction*. Chapman & Hall/CRC, 1993.

[21] Seymour Geisser. *Predictive Inference: An Introduction*. 1993.

[22] J.E. Hanson, I. Whalley, M. Steinder, and J.O. Kephart. Multi-aspect hardware management in enterprise server consolidation. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 543–550. IEEE, 2010.

[23] R.R. Harmon and N. Auseklis. Sustainable it services: Assessing the impact of green computing practices. In *Management of Engineering Technology, 2009. PICMET 2009. Portland International Conference on*, pages 1707 – 1717, aug. 2009.

[24] T. Henderson and S. Bhatti. Networked games: a QoS-sensitive application for QoS-insensitive users? In *Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS: What have we learned, why do we care?*, pages 141–147. ACM, 2003.

[25] N. Kandasamy, S. Abdelwahed, and J.P. Hayes. Self-optimization in computer systems via on-line control: Application to power management. 2004.

[26] K. Kant and J. Alexander. Proactive vs. reactive idle power control. *Proc. of DTTC*, 2008.

[27] J.O. Kephart and D.M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[28] B. Khargharia, S. Hariri, and M.S. Yousif. Autonomic power and performance management for computing systems. *Cluster Computing*, 11(2):167–181, 2008.

[29] D.G. Kleinbaum, M. Klein, and E.R. Pryor. *Logistic regression: a self-learning text*. Springer Verlag, 2010.

[30] Steve Kleynhans. the green pc environment, 2007.

[31] H.P. Kriegel, P. Kroger, and A. Zimek. Outlier detection techniques. In *Tutorial at the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Citeseer, 2009.

[32] P. Kurp. Green computing. *Communications of the ACM*, 51(10):11–13, 2008.

[33] D. Kushner. Engineering everquest: online gaming demands heavyweight data centers. *Spectrum, IEEE*, 42(7):34 – 39, 2005.

[34] Liang Liu, Hao Wang, Xue Liu, Xing Jin, Wen Bo He, Qing Bo Wang, and Ying Chen. Greencloud: a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, ICAC-INDST '09, pages 29–38, New York, NY, USA, 2009. ACM.

[35] J.R. Lorch and A.J. Smith. Reducing processor power consumption by improving processor time management in a single-user operating system. In *Proceedings of the 2nd annual international conference on Mobile computing and networking*, pages 143–154. ACM, 1996.

[36] K. Matthias. Towards autonomic management in system administration. 2008.

[37] Simon Mingay. IT Vendors, Service Providers and Users Can Lighten IT's Environmental Footprint. 2007.

[38] R. Mohanty, V. Ravi, and M. Patra. Software Reliability Prediction Using Group Method of Data Handling. *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, pages 344–351, 2009.

[39] G.E. Moore et al. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[40] J. Nichols and M. Claypool. The effects of latency on online madden NFL football. In *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, pages 146–151. ACM, 2004.

[41] University of Illinois Department of Atmospheric Sciences. Other forecasting methods: climatology, analogue and numerical weather prediction. http://ww2010.atmos.uiuc.edu/%28Gh%29/guides/mtr/fcst/mth/oth.rxml.

[42] M. Oliveira and T. Henderson. What online gamers really think of the Internet? In *Proceedings of the 2nd workshop on Network and system support for games*, pages 185–193. ACM, 2003.

[43] Robert A. Donnelly Ph.D. *The Complete Idiot's Guide to Statistics, 2nd Edition*. 2007.

[44] J.C. Platt and C.J.C. Burges. Probability estimate for k-nearest neighbor classification, 2009.

[45] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No power struggles: coordinated multi-level power management for the data center. *SIGARCH Comput. Archit. News*, 36:48–59, March 2008.

[46] I. Rish. An empirical study of the naive Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46, 2001.

[47] J.A. Roberson, G.K. Homan, A. Mahajan, B. Nordman, C.A. Webber, R.E. Brown, M. McWhinney, and J.G. Koomey. Energy use and power levels in new monitors and personal computers. *Lawrence Berkeley National Laboratory, Berkeley, CA*, 2002.

[48] J.L. Rodgers and W.A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[49] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu. Power-aware qos management in web servers. In *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, pages 63–72. IEEE, 2003.

[50] D. Sheppard. Beginnerâs introduction to perl, 2003.

[51] Sqlite. Sqlite webpage. http://www.sqlite.org/.

[52] Steam. About steam. http://store.steampowered.com/about/.

[53] J.M. Steele. Eclipse prediction in mesopotamia. *Astronomy*, 55:1–73, 1969.

[54] Till Tantau. Tikz. http://sourceforge.net/projects/pgf/.

[55] Twitter. About twitter. http://twitter.com/about.

[56] Unknown Wiley. Data-mining concepts. http://media.wiley.com/product_data/excerpt/24/04712285/0471228524-1.pdf.

[57] T. Yasui, Y. Ishibashi, and T. Ikedo. Influences of network latency and packet loss on consistency in networked racing games. In *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–8. ACM, 2005.

[58] S. Zander and G. Armitage. Empirically measuring the QoS sensitivity of interactive online game players. In *ATNACâ04: Australian Telecommunications Networks and Applications Conference*, 2004.

# Appendix A

# The Test Framework script

Listing A.1: TestFramework.pl

```
#!/usr/bin/perl
use Statistics::Descriptive;
use Getopt::Std;
use POSIX qw(ceil);

my $opt_string = 'vdhA:D:T:O:t:m:ewn';
getopts("$opt_string",\my %opt ) or usage() and exit 1;

$A = $opt{'A'}; ## Percentile
$T = $opt{'T'}; ## number of training days for the profile.
$D = $opt{'D'}; ## totdays 228
$NumberOfGames = 0; # -1
$date = getDate();
$VERBOSE = 1;
$OFFSET = $opt{'O'};
my $TOPIC = $opt{'t'};
my $FOLDER = $opt{'m'};
my $ERROR_CORRECT = $opt{'e'};
my $WEEKEND = $opt{'w'};
my $WORKDAY = $opt{'n'};
my @WEEKEND_CONVERT_TABLE = ();
my $DATA_SKEW;

# @games = ("Counter_Strike","Counter_Strike__Source",
#      "Call_of_Duty__Modern_Warfare_2___Multiplayer","
    Football_Manager_2010",
#      "Team_Fortress_2","Left_4_Dead_2","Garry_s_Mod","
    Empire__Total_War",
#      "Condition_Zero","Battlefield__Bad_Company_2","Borderlands",
#      "Call_of_Duty__Modern_Warfare_2","Day_of_Defeat__Source","
    Football_Manager_2009",
#      "Napoleon__Total_War","Left_4_Dead","Mount_Blade__Warband",
#      "Warhammer_40_000__Dawn_of_War_II___Chaos_Rising","
    Killing_Floor",
#      "Half_Life_2__Deathmatch","Half_Life_2","
    Warhammer_40_000__Dawn_of_War_II",
#      "Dragon_Age__Origins","Day_of_Defeat","Portal","
    Supreme_Commander_2",
#      "Just_Cause_2","Sid_Meier_s_Civilization_IV__Beyond_the_Sword
    ","Torchlight",
#      "Grand_Theft_Auto_IV","Metro_2033");
@games = ("Supreme_Commander_2");

%GAMECACHE = ();
%SCORE_HASH = ();
@EMPTY = ();
$FOLDER = "${FOLDER}_$date";
mkdir($FOLDER);

open(OUT,">$FOLDER/Descriptive.tex") or die "Error:␣$!\n";
open(LOG,">$FOLDER/${TOPIC}_A${A}_T${T}_D${D}_O${OFFSET}.dat") or
    die "Error:␣$!\n";

# exit 0;

print OUT "\\section\{Statistic␣score␣using␣$A␣percentile:\}\n";
print OUT "\\begin\{tabular\}\{p{4cm}|c|c|c|c|c|c|c|c|\}␣%␣for␣
    vertical␣lines,␣insert␣|,␣like␣|c|ccc|\n";
print OUT "Game␣&␣b␣&␣Mean␣&␣Median␣&␣Distance␣&␣Min␣&␣Max␣&␣
    Distance␣\\\\\n";
print OUT "\\hline\n";

for ( $g = 0; $g <= $NumberOfGames; $g++){
    $game = $games[$g];
    $game =~ s/_/ /g;
    %GAMECACHE = ();
    print "game␣nr␣$g:␣$game\n";
    $bestb = 0;
    $bestmean = 0;
    $bestmedian = 0;
    $bestMeanMediandistance = 0;
    $bestmin = 0;
    $bestmax = 0;
    $bestMinMaxdistance = 0;
    $input_holes = 0;
    @BEST_SCORE_FREQUENCY = ();

    for ( $b = 1; $b <= 288; $b++){
        my @newarray = ();
        for(my $i = 1; $i < $T; $i++){
            @day = getDay($i,$games[$g],$OFFSET);
#           print "b: $b T:$i -> " . $day[( $b + 1 )] . "\n";
            if ( $day[$b] ){

                push(@newarray, $day[$b]);
            } else {
                $input_holes++;
            }
        }
#       print "ser dagene\n";
        if ( @newarray ){
#           print "start newarray stats\n";
            $stat = Statistics::Descriptive::Full->new();
            $stat->add_data(@newarray);

## N er funnet basert paa $b for alle dager i $T
            $N = $stat->percentile($A);
#           print "stop newarray stats\n";
        } else {
            print "error:␣empty␣day␣for␣b␣=␣$b␣($input_holes)\n";
            exit;
        }
#       exit 0;
## Faktor $f som skal brukes paa arrays for hver dag

## Gaar igjennom alle dager i T.
        @NORMALISERTE_DAGER = ();
#       print "Normaliser: start\n";
        for ( $i = 1; $i <= $T; $i++ ){
            my @d = getDay($i,$games[$g],$OFFSET);
            if ( $d[$b] ){
                $f = $N / $d[$b];
#               print "faktor: $f\n";
## Normaliserer verdiene t = 1 til t = 288 ved hjelp av N ved aa
    bruke $f faktoren.
                my @normalisertdag = normaliser(\@d, $f);
#               print "norm: $normalisertdag[$i]\n";

## Skriver hver normalisert dag til individuelle filer
#   debugPrintArrayTilFil(\@normalisertdag,"normalisert_dag_$i.
    dat");


## lager array av arrays av alle T normaliserte dager
                $NORMALISERTE_DAGER[$i] = [ @normalisertdag ];
            }
        }
#       print "Normaliser: stop\n";

# vi lager oss en array som til slutt kommer til aa inneholde
    profilen
        my @min_profil = ();

# gaar igjennom alle tidspunkt
#       print "lag profil: start\n";
        my $start = time;
        for ( $i = 1; $i <= 288; $i++ ){

            # vi henter alle verdiene for det tidspunktet inni en
                temporaer array.
            my @temparray = ();
#           print "\tlager array\n";
```

```
                for ( $d = 1; $d <= $T; $d++ ){
#                   print "NORMALISERTE_DAGER[$d][$i] ->
    $NORMALISERTE_DAGER[$d][$i]\n";
                    if ( $NORMALISERTE_DAGER[$d][$i] ){
#                       $temparray[$d] = $NORMALISERTE_DAGER[$d][$i];
                        push(@temparray,$NORMALISERTE_DAGER[$d][$i]);
                    }
#           print "temp: $temparray[$d]\n";
                }

                if ( @temparray ){
                    if ( $i == $b ){
                        $min_profil[$i] = $temparray[0];
                    } elsif ( unevenArray(@temparray)) {
#                       print "start temparray stats\n";
                        my $stat2 = Statistics::Descriptive::Full->new
                            ();

                        $stat2->add_data(@temparray);

                        # regn ut $A persentilen
                        $min_profil[$i] = $stat2->percentile($A);
#                       print "stop temparray stats\n";
                    } else {
                        $min_profil[$i] = $temparray[0];
                    }
                }
#               exit 0 if $i == 2;

#               print "normalisering, ferdig\n";

            }
#           exit ;


# skriv ut profilen, saa vi kan sjekke om det ser fornuftig ut:
# debugPrintArrayTilFil(\@min_profil,"ferdig_profil.dat");

# vi har ogsaa en array hvor vi sammler inn alle scorene
            my @SCORES = ();
            my @SCORE_FREQUENCY = ();
#           print "scorer: start\n";
            for ( $i = $T + 1; $i <= ( $D + $T ); $i++){

                # hent neste dag:
                my @dag = getDay($i,$games[$g],$OFFSET);

                if ( not $dag[$b] ){
                    print "AARGH!␣har␣hull␣ved␣beta␣$b\n";
                    next;
                }

                # normaliser profilen til denne dagen:
                # finn faktor ved punkt $b, som typisk er 1:
```

```
                $fs  = $dag[$b] / $min_profil[$b];

                # lag en normalisert versjon av profilen, som er
                    tilpasset dagen:
                @normalisertprofil = normaliser(\@min_profil, $fs);

                # score dagen i forhold til den normaliserte profilen
                my ( $score, $avg_distance ) = getScore(\
                    @normalisertprofil,\@dag);



                # print "Score for dag $i : $score\n";
                # lagre scoren i en array:
                push(@SCORES, $score);
                $SCORE_FREQUENCY[$score]++;

                $SCORE_HASH{$game}{$b}{$i}{"score"} = $score;
                $SCORE_HASH{$game}{$b}{$i}{"avg_distance"} =
                    $avg_distance;

                print LOG "$game,$b,$i,$score,$avg_distance\n";

                    plot2legends(\@normalisertprofil,\@dag,"GAME:␣
                        $game␣BETA:␣$b␣A:␣$A␣DAY:␣". ($i - $T) ."␣(␣of
                        ␣$D␣days)",$score,$avg_distance); # if ($b ==
                         150);

            }
#           print "scorer: slutt\n";
#           debugPrintArrayTilFil(\@SCORES,"SCORE.dat");

# statistikk paa SCORES:
#           exit 0;
#           print "start Sstat stats\n";
            $Sstat = Statistics::Descriptive::Full->new();
            $Sstat->add_data(@SCORES);
            $Smin = $Sstat->min();
            $Smax = $Sstat->max();
            $Smean = $Sstat->mean();
            $Smedian = $Sstat->median();
#           print "stop Sstat stats\n";

            if ($Smean > $bestmean){
#               print "Check: Smean is greater than bestmean\n";
                $bestb = $b;
                $bestmean = $Smean;
                $bestmedian = $Smedian;
                $bestMeanMediandistance = $Smedian - $Smean;
                $bestmin = $Smin;
                $bestmax = $Smax;
                $bestMinMaxdistance = $Smax - $Smin;
                @BEST_SCORE_FREQUENCY = @SCORE_FREQUENCY;
#               $bestavg_distance = $avg_distance;
```

```
            }

        }

        plot1legends(\@BEST_SCORE_FREQUENCY,"Score frequency for best 
            b in game $game");

        ## create stat table

        print OUT "\\hline\n";
        printf OUT "$game & ";
        printf OUT ("%.0f & ", $bestb);
        printf OUT ("%.2f & ", $bestmean);
        printf OUT ("%.0f & ", $bestmedian);
        printf OUT ("%.2f & ", $bestMeanMediandistance);
        printf OUT ("%.0f & ", $bestmin);
        printf OUT ("%.0f & ", $bestmax);
        printf OUT ("%.0f \\\\\n", $bestMinMaxdistance);
}

print OUT "\\end\{tabular\}\n";
close(OUT);
close(UT);
close(LOG);
# system("pdflatex Descriptive_rapport.tex");

## Sub Rutiner:

sub debugPrintArrayTilFil{
        @arr = @{@_[0]};
        $fil = $_[1];
        open (FIL, ">$fil") or die "can't open: $!";
        foreach ( @arr )
        {
            print FIL "$_\n";
        }
        close FIL;
}

sub getDay{
        my $day = $_[0];
        my $spill = $_[1];
        my @day;

        my $offset = $_[2];

        my $sqlday = $day;

        if ( $WEEKEND ){
#           print "requesting day $sqlday\n";

            $sqlday = convertToWeekend($sqlday);

#           print "Using day: $sqlday\n";
```

```
    }

    if ( $WORKDAY ){
#        print "requesting day $sqlday\n";

        $sqlday = convertToWorkday($sqlday);

#        print "Using day: $sqlday\n";
    }

#     exit;


    if ( $offset ){
        $sqlday = $day * 7 + $offset;

#        print "OFFSET is $OFFSET, adjusting $day to $sqlday\n";

    }
#     verbose("getDay: $spill ( $day )\n");
    my @finalday;
    if ( not $GAMECACHE{$spill}{$sqlday}){
        $value = qx(sqlite3 gamebasev2.sqlite "select␣date,count␣
            from␣games␣where␣day␣=␣'$sqlday'␣AND␣game␣=␣'$spill'")
            ;
        my @day = split("\n",$value);
        my $first_date;
        my $forrige_dato;
        foreach my $dday (@day){
            my ($dato,$count) = split /\|/,$dday;
            $first_date = $dato unless $first_date;
            my $diff = $dato - $forrige_dato;
            my $newindex = int( ($dato - $first_date) / 300 + 0.5)
                + 1;
#            print "$newindex: $dato, $count\n";
            if ( $forrige_dato and $diff > 400 ){

                $newdiff = $diff - 300;
                $hull_space = int(( $newdiff / 300 ) + 0.5);
                my $forrige_index = $newindex - $hull_space - 1;
#                print "$newindex: hull between $forrige_dato and
    $dato ( $diff ) ( space: $hull_space ) ( Forrige verdi [
    $forrige_index]: " . $finalday[$forrige_index] . ")\n";

                if ( $hull_space <= 3 ){
                    for ( my $l = 0; $l < $hull_space; $l++){
                        my $index = $newindex - $hull_space + $l;

                        my $value = $finalday[$forrige_index] + (
                            $l + 1) * (( $count - $finalday[
                            $forrige_index])/($hull_space));
#                        print "inserting finalday[$index] = $value
    \n";
                        if ( $ERROR_CORRECT ){
                            $finalday[$index] = $value;
                        }

                    }
                }


            }
##          my $newindex = int( ($dato - $first_date) / 300);
            $finalday[$newindex] = $count;
            $forrige_dato = $dato;
            # foreach my $item ( @finalday ){
            #    print "value: $item\n";
            # }
        }
        $GAMECACHE{$spill}{$sqlday} = \@finalday;
    } else {
#       verbose(" ( From cache )\n");

        @finalday = @{$GAMECACHE{$spill}{$sqlday}};
#       verbose("item 0: $day[1] -> @day\n");
    }
    return @finalday;
}

sub getScore{
    my @normarr = @{@_[0]};
    my @dayarr = @{@_[1]};
    my $distance = 0;
    my $score = 0;
    for ( $s = 1; $s <= 288; $s++ ){

        if($normarr[$s] >= $dayarr[$s]){
            $score++;
            $distance += $normarr[$s] - $dayarr[$s];
        }
    }
    my $avg_distance = -1;
    if ( $score > 0 ){
    $avg_distance = int($distance / $score);
    }
    return ($score,$avg_distance);
}

sub normaliser {
    @array = @{@_[0]};
    $f1 = $_[1];
    @tempnormarray = ();
    for ( $j = 1; $j <= 288; $j++ ){

        $tempnormarray[$j] = int($array[$j] * $f1);
    }
    return @tempnormarray;
}
```

```

sub getDate {
    my ($sec, $min, $hour, $mday, $mon, $year) = localtime(time);
    $year = $year + 1900;
    $mon++;
    if ($mon < 10) {$mon = "0" . $mon;}
    if ($mday < 10) {$mday = "0" . $mday;}
    if ($hour < 10) {$hour = "0" . $hour;}
    if ($min < 10) {$min = "0" . $min;}
    if ($sec < 10) {$sec = "0" . $sec;}
    $date = "$mday"."."."$mon"."."."$year"."-"."$hour".":"."$min".
        ":"."$sec";
    return ($date);
}

sub verbose {
    print $_[0] if $VERBOSE;
}

# my @numbers = (10.22,20.33,22.3,11.3,12.4,8.3,10.4);

# for qw/25 50 75 90 95 99/;

sub percentile {
    my ($p,$aref) = @_;
    my $percentile = int(($p * $#{$aref}/100) + 0.5);
    return (sort @$aref)[$percentile];
}

sub plot2legends {

    my @profil = @{@_[0]};
    my @dag = @{@_[1]};
    my $caption = $_[2];
    my $Pscore = $_[3];
    my $Pavg_distance = $_[4];
    # Stats

    my $pmax = getMax(@profil);
    my $dmax = getMax(@dag);

    if ( $pmax > $dmax ){
#        print "ProfilMAX: $pmax  -  DagMAX: $dmax\n";
#        print "Profil har stoerst verdi!!!!!!!!!!!\n";
        $min = getMin(@profil);
        $max = getMax(@profil);
    }else{
#        print "ProfilMAX: $pmax  -  DagMAX: $dmax\n";
#        print "!!!!!!!!!! Dag har stoerst verdi\n";
        $min = getMin(@dag);
        $max = getMax(@dag);
    }

    if ($max < 7000 ){
        $roundup_max = roundup(($max / 100),10);
}else{
    $roundup_max = roundup(($max / 100),100);
}
    $proc = 100;

# tiks
    if ($roundup_max == 10){ $tikz = 1; }
    if ($roundup_max == 20){ $tikz = 2; }
    if ($roundup_max == 30){ $tikz = 3; }
    if ($roundup_max == 40){ $tikz = 4; }
    if ($roundup_max == 50){ $tikz = 5; }
    if ($roundup_max == 60){ $tikz = 6; }
    if ($roundup_max == 70){ $tikz = 7; }
    if ($roundup_max == 80){ $tikz = 8; }
    if ($roundup_max == 90){ $tikz = 9; }
    if ($roundup_max == 100){ $tikz = 10; }
    if ($roundup_max == 200){ $tikz = 20; }
    if ($roundup_max == 300){ $tikz = 50; }
    if ($roundup_max == 400){ $tikz = 50; }
    if ($roundup_max == 500){ $tikz = 50; }
    if ($roundup_max == 600){ $tikz = 100; }
    if ($roundup_max == 700){ $tikz = 100; }
    if ($roundup_max == 800){ $tikz = 100; }
    if ($roundup_max == 900){ $tikz = 100; }
    if ($roundup_max == 1000){ $tikz = 100; }
    if ($roundup_max == 1100){ $tikz = 110; }
    if ($roundup_max == 1200){ $tikz = 120; }
    if ($roundup_max == 1300){ $tikz = 130; }
    if ($roundup_max == 1400){ $tikz = 140; }
    if ($roundup_max == 1500){ $tikz = 150; }
    if ($roundup_max == 1600){ $tikz = 160; }
    if ($roundup_max == 1700){ $tikz = 170; }
    if ($roundup_max == 1800){ $tikz = 180; }
    if ($roundup_max == 1900){ $tikz = 190; }
    if ($roundup_max == 2000){ $tikz = 200; }
    if ($roundup_max == 2100){ $tikz = 210; }
    if ($roundup_max == 2200){ $tikz = 220; }
    if ($roundup_max == 2300){ $tikz = 230; }
    if ($roundup_max == 2400){ $tikz = 240; }
    if ($roundup_max == 2500){ $tikz = 250; }
    if ($roundup_max == 2600){ $tikz = 260; }
    if ($roundup_max == 2700){ $tikz = 270; }
    if ($roundup_max == 2800){ $tikz = 280; }
    if ($roundup_max == 2900){ $tikz = 290; }
    if ($roundup_max == 3000){ $tikz = 300; } # else{ $tikz = 300
        }
```

```
# Changeable variables
# Plot size in pt
    $ysize = 150;
    $xsize = 300;

# X data size
        $xdatasize = $#profil;

# Lable names
    $ylable = "Number␣of␣players␣/␣$proc";
    $xlable = "Time";

# Topic
    $linje1 = "$TOPIC␣-␣$game";
    $linje2 = "$T␣training␣days,␣$D␣observed␣days,␣day␣". ($i - $T
        ) . ",␣b=$b,␣$A␣percentile";

# Number of lables points
    $ylablespoints = 7;
    $xlablepoints = 10;

# Figure name
    $enlegend = "Profile␣is␣above␣day";
    $tolegend = "Day␣". ($i - $T) . "";
    $trelegend = "Profile␣is␣under␣day";

# Summery
    $score1 = "Score:␣$Pscore";
    $scorepercent = ($Pscore / 288) * 100;
    $score2 =
    $score3 = "Avg␣dist:␣$Pavg_distance";
#####################

# y
    $foreachmax = roundup(($roundup_max / $ylablespoints),100);
    $tikzyvalue = ($ysize / $roundup_max );
    $maxpluss = (($max + $foreachmax) - 1) / 100;



# x
    $xforeachmax = ($xdatasize / $xlablepoints);
    $tikzxvalue = ($xsize / $xdatasize);
    $xmaxpluss = (($xdatasize + $xforeachmax) - 1) / 100;
# legend
    $legendplaced = ($max / 2 / 100);

# Captio
#   $caption = "The profile tested against Monday \\# 6 of 20
    using 85 percentiles in game Counter Strike";


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

```
    open(UT,">>$FOLDER/awesome.tex") or die;
    print UT "\\begin\{figure\}\n";
    print UT "\\centering\n";
    print UT "\\begin\{tikzpicture\}\[x=$tikzxvalue" . "pt,y=$tikzyvalue" . "pt\]\n";
    print UT "\\def\\xmin\{0\}\n";
    print UT "\\def\\xmax\{300\}\n";
    print UT "\\def\\ymin\{0\}\n";
    print UT "\\def\\ymax\{" . $roundup_max . "\}\n";
    print UT "\\draw\[dashed,color=black, font=\\footnotesize\]  \($b,\\ymin\) node\[above=0.25cm, right=0.01cm\] \{\$b\$\} -- \($b,\\ymax\)\;\n";
    print UT "\\draw\[dashed,color=black,fill=white\] \(301,". ($roundup_max / 1.45) ."\) rectangle \(375,". ($roundup_max / 2.7) ."\)\;\n";
    printf UT ("\\draw\[style=help lines, ystep=%.0f, xstep=300\] \(\\xmin,\\ymin\) grid \(\\xmax,\\ymax\)\;\n",$roundup_max);
    print UT "\\draw\[dashed,color=black,fill=white\] \(301,". ($roundup_max / 1) ."\) rectangle \(390,". ($roundup_max / 1.4) ."\)\;\n";

    print UT "\\draw\[->\] \(\\xmin,\\ymin\) -- coordinate \(x axis_mid\) \(\\xmax,\\ymin\) node\[right\] \{\$x\$\}\;\n";
    print UT "\\draw\[->\] \(\\xmin,\\ymin\) -- coordinate \(y axis_mid\) \(\\xmin,\\ymax\) node\[above\] \{\$y\$\}\;\n";

    print UT "\\node at \(150, \\ymax\) \[above=0.40cm, scale=1.2\] \{ $linje1 }\;\n";
    print UT "\\node at \(150, \\ymax\) \[above=0.01cm, scale=1.1, font=\\footnotesize\] \{ $linje2 }\;\n";
    print UT "\\foreach \\x in \{0,50,...,300\} \\draw \(\\x,1pt\) -- \(\\x,-3pt\) node\[anchor=north\] \{\\x\} \;\n";
    printf UT ("\\foreach \\y in \{0,%.0f,..., $roundup_max\} \\draw \(1pt,\\y\) -- \(-3pt,\\y\) node\[anchor=east\] \{\\y\}\;\n", $tikz);

    print UT "\\draw\[color=Blue,thick\] plot\[mark=,mark size=1pt\] coordinates\{";



    for ( $k = 1; $k <= 288; $k++){
        $profilen = $profil[$k] / $proc;
        if($profil[$k]){

            print UT "\($k,$profilen\)";
        }else{
            print UT "\($k,0\)";
        }
    }

    print UT "\}\;\n";



    @tmp;


    for ( $k = 1; $k <= 288; $k+=3){
        if($profil[$k] < $dag[$k]){
            $profilen = $profil[$k] / $proc;
            print UT "\\draw\[color=red,thick\]␣plot\[mark=*,mark␣
                size=0.5pt\]␣coordinates\{\($k,$profilen\)\}\;\n";
        }
    }

    print UT "\\draw\[color=black,thick\]␣plot\[mark=,mark␣size=1
        pt\]␣coordinates\{";
    for ( $k = 1; $k <= 288; $k++){
        $dagen = $dag[$k] / $proc;
        if($dag[$k]){

        print UT "\($k,$dagen\)";
        }else{
            print UT "\($k,0\)";
        }
    }

    print UT "\}\;\n";

# Score

    print UT "\\begin\{scope\}\[shift=\{\(312,". ($roundup_max /
        1.60) ."\)\}\]\n";
    print UT "\\draw\[color=blue,thick,scale=1.5\]␣␣node\[right,
        black,scale=0.9\]\{Summary\}\;\n";
    print UT "\\end\{scope\}\n";

    print UT "\\begin\{scope\}\[shift=\{\(302,". ($roundup_max /
        1.80) ."\)\}\]\n";
    print UT "\\draw\[color=blue,thick,scale=1.5\]␣␣node\[right,
        black,scale=0.8\]\{$score1\}\;\n";
    print UT "\\end\{scope\}\n";

    print UT "\\begin\{scope\}\[shift=\{\(302,". ($roundup_max /
        2.03) ."\)\}\]\n";
    printf UT ("\\draw\[color=blue,thick,scale=1.5\]␣␣node\[right,
        black,scale=0.8\]\{Score␣\\%:␣%.2f\}\;\n", $scorepercent);
    print UT "\\end\{scope\}\n";

    print UT "\\begin\{scope\}\[shift=\{\(302,". ($roundup_max /
        2.35) ."\)\}\]\n";
    print UT "\\draw\[color=blue,thick,scale=1.5\]␣␣node\[right,
        black,scale=0.8\]\{$score3\}\;\n";
    print UT "\\end\{scope\}\n";

```

```
##

    print UT "\\begin\{scope\}\[shift=\{\(305,". ($roundup_max / 1.08) ."\)\}\]\n";
    print UT "\\draw\[color=blue,thick,scale=1.5\]␣\(0,0\)␣--␣\(5,0\)␣--␣\(5,0\)␣node\[right,black,scale=0.8\]\{$enlegend\}\;\n";
    print UT "\\end\{scope\}\n";
    print UT "\\begin\{scope\}\[shift=\{\(305,". ($roundup_max / 1.17) ."\)\}\]\n";
    print UT "\\draw\[mark=*,mark␣size=0.3pt,color=black,thick,scale=1.5\]␣\(0,0\)␣--␣\(5,0\)␣--␣\(5,0\)␣node\[right,black,scale=0.8\]\{$tolegend\}\;\n";
    print UT "\\end\{scope\}\n";
    print UT "\\begin\{scope\}\[shift=\{\(305,". ($roundup_max / 1.27) ."\)\}\]\n";
    print UT "\\draw\[color=blue,thick,scale=1.5\]␣\(0,0\)␣--␣\(5,0\)␣--␣\(5,0\)␣node\[right,black,scale=0.8\]\{$trelegend\}\;\n";
    print UT "\\end\{scope\}\n";
    print UT "\\draw\[color=red,thick\]␣plot\[mark=*,mark␣size=0.5pt\]␣coordinates\{\(306,". ($roundup_max / 1.27) ."\)\}\;\n";
    print UT "\\draw\[color=red,thick\]␣plot\[mark=*,mark␣size=0.5pt\]␣coordinates\{\(309,". ($roundup_max / 1.27) ."\)\}\;\n";
    print UT "\\draw\[color=red,thick\]␣plot\[mark=*,mark␣size=0.5pt\]␣coordinates\{\(312,". ($roundup_max / 1.27) ."\)\}\;\n";


    print UT "\\node\[below=0.60cm\]␣at␣\(x␣axis␣mid\)␣\{$xlable\}\;\n";
    print UT "\\node\[rotate=90,␣above=1cm\]␣at␣\(y␣axis␣mid\)␣\{$ylable\}\;\n";
    print UT "\\end\{tikzpicture\}\n";
    print UT "\\caption\{$caption\}\n";
    print UT "\\end\{figure\}\n";
    print UT "\\clearpage\n";
    $count++;
    close(UT);

#    print "max: $max\n";
#    print "rountup: $roundup_max\n";


}

sub plot1legends {

    my @profil = @{@_[0]};
    my $caption = $_[1];
    # Stats
```

```
        $min = getMin(@profil);
        $max = getMax(@profil);

# tiks
    $roundup_max = roundup($max,10);

    if ($roundup_max == 10){ $tikz = 1; }
    if ($roundup_max == 20){ $tikz = 2; }
    if ($roundup_max == 30){ $tikz = 3; }
    if ($roundup_max == 40){ $tikz = 4; }
    if ($roundup_max == 50){ $tikz = 5; }
    if ($roundup_max == 60){ $tikz = 6; }
    if ($roundup_max == 70){ $tikz = 7; }
    if ($roundup_max == 80){ $tikz = 8; }
    if ($roundup_max == 90){ $tikz = 9; }
    if ($roundup_max == 100){ $tikz = 10; }


# Changeable variables
# Plot size in pt
    $ysize = 150;
    $xsize = 300;

# X data size
    $xdatasize = $#profil;

# Lable names
    $ylable = "Frequency";
    $xlable = "Score";

# Topic
    $linje1 = "$game␣score␣frequency␣on␣senario␣$TOPIC";
    $linje2 = "Frequency␣distribution␣on␣the␣best␣b␣of␣the␣$D␣
        observed␣days";

# Figure name
    $enlegend = "Score␣frequency";

#####################

# y
    $tikzyvalue = ($ysize / $roundup_max );
# x
    $tikzxvalue = ($xsize / $xdatasize);
# legend

# Captio
#     $caption = "The profile tested against Monday \\# 6 of 20
    using 85 percentiles in game Counter Strike";


######################

    open(UT1,">>$FOLDER/scorefreq.tex")or die;
    print UT1 "\\begin\{figure\}\n";
    print UT1 "\\centering\n";
    print UT1 "\\begin\{tikzpicture\}\[x=$tikzxvalue" . "pt,y=
        $tikzyvalue" . "pt\]\n";
    print UT1 "\\def\\xmin\{0\}\n";
    print UT1 "\\def\\xmax\{300\}\n";
    print UT1 "\\def\\ymin\{0\}\n";
    print UT1 "\\def\\ymax\{" . $roundup_max . "\}\n";
    printf UT1 ("\\draw\[style=help␣lines,␣ystep=%.0f,␣xstep=300\]
        ␣\(\\xmin,\\ymin\)␣grid␣\(\\xmax,\\ymax\)\;\n",$tikz);
    print UT1 "\\draw\[dashed,color=black,fill=white\]␣\(301,". (
        $roundup_max / 1) ."\)␣rectangle␣\(390,". ($roundup_max /
        1.18) ."\)\;\n";
    print UT1 "\\draw\[->\]␣\(\\xmin,\\ymin\)␣--␣coordinate␣\(x␣
        axis␣mid\)␣\(\\xmax,\\ymin\)␣node\[right\]␣\{\$x\$\}\;\n";
    print UT1 "\\draw\[->\]␣\(\\xmin,\\ymin\)␣--␣coordinate␣\(y␣
        axis␣mid\)␣\(\\xmin,\\ymax\)␣node\[above\]␣\{\$y\$\}\;\n";
    print UT1 "\\node␣at␣\(150,␣\\ymax\)␣\[above=0.40cm,␣scale
        =1.2\]␣\{ $linje1 }\;\n";
    print UT1 "\\node␣at␣\(150,␣\\ymax\)␣\[above=0.01cm,␣scale
        =1.1,␣font=\\footnotesize\]␣\{ $linje2 }\;\n";
    print UT1 "\\foreach␣\\x␣in␣\{0,20,...,300\}␣\\draw␣\(\\x,1pt
        \)␣--␣\(\\x,-3pt\)␣node\[anchor=north\]␣\{\\x\}␣\;\n";
    printf UT1 ("\\foreach␣\\y␣in␣\{0,%.0f,..., $roundup_max\}␣\\
        draw␣\(1pt,\\y\)␣--␣\(-3pt,\\y\)␣node\[anchor=east\]␣\{\\y
        \}\;\n", $tikz);

    # print UT1 "\\draw\[color=Blue, thick\] plot\[mark=,mark size
        =1pt\] coordinates\{";
    print UT1 "\\draw[ycomb,color=blue,line␣width=0.1cm]␣plot␣
        coordinates\{";
    for ( $k = 1; $k <= 288; $k++){
        if($profil[$k]){
            print UT1 "\($k,$profil[$k]\)";
        }else{
            print UT1 "\($k,0\)";
        }
    }

    print UT1 "\}\;\n";

##

    print UT1 "\\begin\{scope\}\[shift=\{\(305,". ($roundup_max /
        1.08) ."\)\}\]\n";
    print UT1 "\\draw\[color=blue,thick,scale=1.5\]␣\(0,0\)␣--␣
        \(5,0\)␣--␣\(5,0\)␣node\[right,black,scale=0.8\]\{
        $enlegend\}\;\n";
    print UT1 "\\end\{scope\}\n";
    print UT1 "\\node\[below=0.60cm\]␣at␣\(x␣axis␣mid\)␣\{ $xlable
        \}\;\n";
    print UT1 "\\node\[rotate=90,␣above=1cm\]␣at␣\(y␣axis␣mid\)␣\{
        $ylable\}\;\n";
    print UT1 "\\end\{tikzpicture\}\n";
    print UT1 "\\caption\{$caption\}\n";
    print UT1 "\\end\{figure\}\n";
    print UT1 "\\clearpage\n";
    close(UT1);

#     print "max: $max\n";
 #    print "rountup: $roundup_max\n";



}

sub unevenArray {

    my $num;
    foreach my $item (@_){
        if ( $num and $num ne $item ){
#               print "uneven array: $num / $item\n";
                return 1;
        }
        $num = $item;
    }

    return 0;
}

sub getMin {

    my $min;
    foreach my $item (@_){
        if ( ( $min and $min > $item ) or not $min ){
            $min = $item;
        }
    }

    return $min;
}

sub getMax {

    my $max;
    foreach my $item (@_){
        if ( ( $max and $max < $item ) or not $max ){
            $max = $item;
        }
    }

    return $max;
}






foreach $number ( @numbers ){

        print "$number␣->␣" . roundup($number,100) . "\n";

    }

sub roundup
  {
          my $num = shift;
          my $roundto = shift || 1;

          return int(ceil($num/$roundto))*$roundto;
      }

sub max
  {
          return $_[0] > $_[1] ? $_[0] : $_[1];
      }

sub rightscale
  {
          my $num = shift;
      return 10 ** max(int(log(abs($num))/log(10))-1,1);
  }

    sub isweek {
        $sqlite = `sqlite3 gamebasev2.sqlite "select␣wday␣from␣
            games␣where␣day␣=␣$_[0]␣limit␣1"`;
        chomp $sqlite;
        print "$sqlite\n";
        if ($sqlite < 6 && $sqlite > 0){
                return 1;
        }
        return 0;
    }

    sub weekday {
        $sqlite = `sqlite3 gamebasev2.sqlite "select␣wday␣from␣
            games␣where␣day␣=␣$_[0]␣limit␣1"`;
        chomp $sqlite;
        return $sqlite;

    }

sub convertToWeekend {

    my $day = $_[0];
    my $daycounter = 0;
    my $token = 1;
    my $weekday = weekday(1);;
#       return $WEEKEND_CONVERT_TABLE[$day] if $WEEKEND_CONVERT_TABLE
    [$day] = $token;

    while ( 1 ){
#           print "d:$day, t:$token, dc:$daycounter\n";
```

```
        if ( $weekday == 6 or $weekday == 0){

            $daycounter++;
            $WEEKEND_CONVERT_TABLE[$day] = $token;
            return $token if $daycounter == $day;

        }
        $token++;
        $weekday++;
        $weekday = 0 if $weekday == 7;

    }


}

sub convertToWorkday {

    my $day = $_[0];
    my $daycounter = 0;
    my $token = 1;
    my $weekday = weekday(1);;
#     return $WEEKEND_CONVERT_TABLE[$day] if $WEEKEND_CONVERT_TABLE
    [$day] = $token;

    while ( 1 ){
#         print "d:$day, t:$token, dc:$daycounter\n";
        if ( $weekday > 0 and $weekday < 6){

            $daycounter++;
            $WEEKEND_CONVERT_TABLE[$day] = $token;
            return $token if $daycounter == $day;

        }
        $token++;
        $weekday++;
        $weekday = 0 if $weekday == 7;

    }


}
```

# Appendix B

# Other Scripts

Listing B.1: DayFreq.pl

```
#!/usr/bin/perl
use POSIX qw(ceil);
use Getopt::Std;

my $opt_string = 'vdhf:';
getopts("$opt_string",\my %opt ) or usage() and exit 1;

@alleb;

my $F = $opt{'f'};

# for ($i = 11; $i <= 30; $i++){
open(FIL,"$F")or die;
$newavg_dist = 999999;
@bestb = (); # bruker denne til aa holde orden paa hvilken dag jeg
     skal maale mot.
@best_dist = (); # bruker denne til aa holde orden paa beste
     distanse, relativt til dagen.
$start_day;
$stop_day = 0;

while($line = <FIL>){

    if($line =~ /([a-z A-Z 0-9]+),(\d+),(\d+),(\d+),(\d+)/g){
            $game = $1;
            $b = $2;
            $day = $3;
            $score = $4;
            $avg_dist = $5;
            if ($score >= 260 ){
                    if ($avg_dist < $best_dist[$day] or not
                        $best_dist[$day] ){
                        #print "b: $b is better than $bestb[$day]
                            with $avg_dist > $best_dist[$day],
                            score: $score\n";
                            $best_dist[$day] = $avg_dist;
                            $bestb[$day] = $b;
                            #$newday = $day;
                            #$newscore = $score;
                                }
                }
            $start_day = $day unless $start_day;
            $stop_day = $day if $day > $stop_day;
                }
    }

# saa maa jeg gaa igjennom @bestb og telle hvor ofte de
    forskjellige B'ene dukket opp
print "start_day:␣$start_day,␣stop_day:␣$stop_day\n";

@b_frequency = ();

for( my $i = $start_day; $i <= $stop_day; $i++ ){
        $b_frequency[$bestb[$i]]++;
        print "best␣b␣for␣day␣$i:␣$bestb[$i]␣with␣avg_distance:␣
            $best_dist[$i]\n";
    }

# naa kan jeg skrive den ut:
# denne skal plottes

# $alleb[$newb]++;
# print "dag: $i - BestB: $newb - Frequency: $alleb[$newb]\n";
# close(FIL);
# }


# Stats
$min = getMin(@b_frequency);
$max = getMax(@b_frequency);


if ($max < 7000 ){
      $roundup_max = roundup(($max / 100),10);
}else{
      $roundup_max = roundup(($max / 100),100);
}
$proc = 100;



# tiks
if ($roundup_max == 10){ $tikz = 1; }
if ($roundup_max == 20){ $tikz = 2; }
if ($roundup_max == 30){ $tikz = 3; }
if ($roundup_max == 40){ $tikz = 4; }
if ($roundup_max == 50){ $tikz = 5; }
if ($roundup_max == 60){ $tikz = 6; }
if ($roundup_max == 70){ $tikz = 7; }
if ($roundup_max == 80){ $tikz = 8; }
if ($roundup_max == 90){ $tikz = 9; }
if ($roundup_max == 100){ $tikz = 10; }
if ($roundup_max == 200){ $tikz = 20; }
if ($roundup_max == 300){ $tikz = 50; }
if ($roundup_max == 400){ $tikz = 50; }
if ($roundup_max == 500){ $tikz = 50; }
if ($roundup_max == 600){ $tikz = 100; }
if ($roundup_max == 700){ $tikz = 100; }
if ($roundup_max == 800){ $tikz = 100; }
if ($roundup_max == 900){ $tikz = 100; }
if ($roundup_max == 1000){ $tikz = 100; }



# Changeable variables
# Plot size in pt
$ysize = 150;
$xsize = 300;

# X data size
$xdatasize = $#b_frequency;

# Lable names
$ylable = "Frequency";
$xlable = "b";

# Topic
$linje1 = "Best␣B␣frequency";
$linje2 = "Between␣day␣$start_day␣and␣$stop_day\n";

# Number of lables points
# $ylablespoints = 7;
# $xlablepoints = 10;

# Figure name
# $figurename = "";
$enlegend = "Best␣B␣Frequency";
# $tolegend = "Day 6";

# Summery
# $score1 = "Score: 284";
# $score2 = "Score \\%: 98";
# $score3 = "Avg dist: 8340";
######################

# y
# $foreachmax = roundup(($roundup_max / $ylablespoints),100);
$tikzyvalue = ($ysize / $roundup_max );
# $maxpluss = (($max + $foreachmax) - 1) / 100;

# x
# $xforeachmax = ($xdatasize / $xlablepoints);
$tikzxvalue = ($xsize / $xdatasize);
```

```
# $xmaxpluss = (( $xdatasize + $xforeachmax) - 1) / 100;
# legend
# $legendplaced = ($max / 2 / 100);

# Captio
# $caption = "The profile tested against Monday \\# 6 of 20 using 85 percentiles in game Counter Strike";


######################

open(UT,">awesome2.tex")or die;
print UT "\\documentclass\[11pt,english,a4paper\]\{report\}\n";
print UT "\\usepackage\[usenames,dvipsnames,pdftex\]\{xcolor\}\n";
print UT "\\usepackage\{tikz,ifthen\}\n";
print UT "\\usepackage\[utf8\]\{inputenc\}\n";
print UT "\\usepackage\[acadian\]\{babel\}\n";
print UT "\\renewcommand\{\\familydefault\}\{\\sfdefault\}\n";
print UT "\\begin\{document\}\n";
print UT "\\begin\{figure\}\n";
print UT "\\centering\n";
print UT "\\begin\{tikzpicture\}\[x=$tikzxvalue" . "pt,y=$tikzyvalue" . "pt\]\n";
print UT "\\def\\xmin\{0\}\n";
print UT "\\def\\xmax\{300\}\n";
print UT "\\def\\ymin\{0\}\n";
print UT "\\def\\ymax\{" . $roundup_max . "\}\n";
printf UT ("\\draw\[style=help␣lines,␣ystep=%.0f,␣xstep=300\]␣\(\\xmin,\\ymin\)␣grid␣\(\\xmax,\\ymax\)\;\n",$roundup_max);
print UT "\\draw\[style=help␣lines,fill=white\]␣\(0,". ($roundup_max / 1) ."\)␣rectangle␣\(85,". ($roundup_max / 1.17) ."\)\;\n";

print UT "\\draw\[->\]␣\(\\xmin,\\ymin\)␣--␣coordinate␣\(x␣axis␣mid\)␣\(\\xmax,\\ymin\)␣node\[right\]␣\{\$x\$\}\;\n";
print UT "\\draw\[->\]␣\(\\xmin,\\ymin\)␣--␣coordinate␣\(y␣axis␣mid\)␣\(\\xmin,\\ymax\)␣node\[above\]␣\{\$y\$\}\;\n";

print UT "\\node␣at␣\(150,␣\\ymax\)␣\[above=0.40cm,␣scale=1.2\]␣\{$linje1}\;\n";
print UT "\\node␣at␣\(150,␣\\ymax\)␣\[above=0.01cm,␣scale=1.1,␣font=\\footnotesize\]␣\{$linje2}\;\n";
print UT "\\foreach␣\\x␣in␣\{0,50,...,300\}␣\\draw␣\(\\x,1pt\)␣--␣\(\\x,-3pt\)␣node\[anchor=north\]␣\{\\x\}␣\;\n";
printf UT ("\\foreach␣\\y␣in␣\{0,%.0f,..., $roundup_max\}␣\\draw␣\(1pt,\\y\)␣--␣\(-3pt,\\y\)␣node\[anchor=east\]␣\{\\y\}\;\n",$tikz);

# print UT "\\draw\[color=Blue,thick\] plot\[mark=,mark size=1pt\] coordinates\{";
print UT "\\draw[ycomb,color=blue,line␣width=0.1cm]␣plot␣coordinates\{";

for ( my $i = 1; $i <= 288; $i++ ){
    if($b_frequency[$i]){
        print UT "\($i,$b_frequency[$i]\)";
        print "$i,$b_frequency[$i]\n";
    }else{
        print UT "\($i,0\)";
#       print "\($i,0\)";
    }
}
print UT "\}\;\n";

print UT "\\begin\{scope\}\[shift=\{\(5,". ($roundup_max / 1.08) .
    "\)\}\]\n";
print UT "\\draw\[color=blue,thick,scale=1.5\]␣\(0,0\)␣—␣\(5,0\)␣
    —␣\(5,0\)␣node\[right,black,scale=0.8\]\{$enlegend\}\;\n";
print UT "\\end\{scope\}\n";
print UT "\\node\[below=0.60cm\]␣at␣\(x␣axis␣mid\)␣\{$xlable\}\;\n
    ";
print UT "\\node\[rotate=90,␣above=1cm\]␣at␣\(y␣axis␣mid\)␣\{
    $ylable\}\;\n";
print UT "\\end\{tikzpicture\}\n";
print UT "\\caption\{$caption\}\n";
print UT "\\end\{figure\}\n";
print UT "\\end\{document\}\n";
close(UT);
system("pdflatex␣awesome2.tex");


print "max:␣$max\n";

print "roundup:␣$roundup_max\n";

print "tikz:␣$tikz\n";

print "Proc:␣$proc\n";


sub getMin {
    my $min;
    foreach my $item (@_){
        if ( ( $min and $min > $item ) or not $min ){
            $min = $item;
        }
    }
    return $min;
}

sub getMax {
    my $max;
    foreach my $item (@_){
        if ( ( $max and $max < $item ) or not $max ){
            $max = $item;
        }
    }
    return $max;
```

```
}
foreach $number ( @numbers ){

     print "$number␣->␣" . roundup($number,100) . "\n";

}

sub roundup
{
     my $num = shift;
     my $roundto = shift || 1;

     return int(ceil($num/$roundto))*$roundto;
}

sub max
{
     return $_[0] > $_[1] ? $_[0] : $_[1];
}

sub rightscale
{
     my $num = shift;
     return 10 ** max(int(log(abs($num))/log(10))-1,1);
}
```

Listing B.2: GapAnalysis.pl

```
#!/usr/bin/perl
use Statistics::Descriptive;

# $spill = $ARGV[0];

# @games = ("Counter_Strike","Counter_Strike__Source",
#      "Call_of_Duty__Modern_Warfare_2___Multiplayer","
     Football_Manager_2010",
#      "Team_Fortress_2","Left_4_Dead_2","Garry_s_Mod","
     Empire__Total_War",
#      "Condition_Zero","Battlefield__Bad_Company_2","Borderlands",
#      "Call_of_Duty__Modern_Warfare_2","Day_of_Defeat__Source","
     Football_Manager_2009",
#      "Napoleon__Total_War","Left_4_Dead","Mount_Blade__Warband",
#      "Warhammer_40_000__Dawn_of_War_II___Chaos_Rising","
     Killing_Floor",
#      "Half_Life_2__Deathmatch","Half_Life_2","
     Warhammer_40_000__Dawn_of_War_II",
#      "Dragon_Age__Origins","Day_of_Defeat","Portal","
     Supreme_Commander_2",
#      "Just_Cause_2","Sid_Meier_s_Civilization_IV__Beyond_the_Sword
     ","Torchlight",
#      "Grand_Theft_Auto_IV","Metro_2033");
@games = ("Counter_Strike");

# $NumberOfGames = 0; # -1
# for ( $g = 0; $g <= $NumberOfGames; $g++){


open(UT,">awesome.tex")or die;
print UT "\\documentclass\[11pt,english,a4paper\]\{report\}\n";
print UT "\\usepackage\[usenames,dvipsnames,pdftex\]\{xcolor\}\n";
print UT "\\usepackage\[usenames,dvipsnames,pdftex\]\{xcolor\}\n";
print UT "\\usepackage\{tikz,ifthen\}\n";
print UT "\\usepackage\[utf8\]\{inputenc\}\n";
print UT "\\usepackage\[acadian\]\{babel\}\n";
print UT "\\renewcommand\{\\familydefault\}\{\\sfdefault\}\n";
print UT "\\begin\{document\}\n";

$arraysize = scalar @games;

$count = scalar @games;
foreach $game (@games){
    $percentDone = (( $count / $arraysize) * 100) - 100;
    printf ("%.0f", $percentDone);
    $gamename = $game;
    $gamename =~ s/_/ /g;
    print "%␣$gamename␣starting␣.␣.␣.␣";
    $value = qx(sqlite3 gamebasev2.sqlite "select␣date,count␣from␣
        games␣where␣game␣=␣'$game'");
@days = split("\n",$value);

my @finalday;

my $forrige_dato;
my $first_date;

my @hull_count;
foreach $day (@days){
#     print "$day\n";


    my ($dato,$count) = split /\|/,$day;

    $first_date = $dato unless $first_date;

#     print "dato: $dato, count: $count\n";
    my $diff = $dato - $forrige_dato;
    if ( $forrige_dato and $diff > 400 ){

        $newdiff = $diff - 300;

        $hull_space = $newdiff / 300;
        my $sdiff = sprintf("%d",$hull_space);
        my $final_space = int($hull_space + 0.5);
#         print "hull between $forrige_dato and $dato  ( $diff ) (
    sdiff: $sdiff) ( space: $hull_space ) -> $final_space\n";
        $hull_count[$final_space]++;
    }
    my $newindex = int( ($dato - $first_date) / 300);
    $finalday[$newindex] = $count;
#     print "$newindex -> $count\n";

    $forrige_dato = $dato;
}
shift(@hull_count);
 for ( $i = 0; $i <= $#hull_count; $i++ ){
    if (not $hull_count[$i]){
        $hull_count[$i] = 0;
    }
  }

# Changeable variables
# Plot size in pt
$ysize = 100;
$xsize = 200;

# X data size
$xdatasize = $#hull_count;

# Lable names
$ylable = "Frequency";
$xlable = "Distance␣Space";

# Number of lables points
$ylablespoints = 3;
$xlablepoints = 5;

# Figure name
$figurename = "Distance␣Space␣Frequency";

#####################
# Stats
$stat = Statistics::Descriptive::Full->new();
$stat->add_data(@hull_count);
$min = $stat->min();
$max = $stat->max();
# y
$foreachmax = ($max / $ylablespoints);
$tikzyvalue = ($ysize / $max);
$maxpluss = ($max + $foreachmax) - 1;

# x
$xforeachmax = ($xdatasize / $xlablepoints);
$tikzxvalue = ($xsize / $xdatasize);
$xmaxpluss = ($xdatasize + $xforeachmax) - 1;
# legend
$legendplaced = ($max / 2 );

print UT "\\begin\{figure\}\n";
print UT "\\centering\n";
print UT "\\begin\{tikzpicture\}\[x=$tikzxvalue" . "pt,y=
    $tikzyvalue" . "pt\]\n";
print UT "\\def\\xmin\{-10\}\n";
print UT "\\def\\xmax\{$xdatasize\}\n";
print UT "\\def\\ymin\{-1\}\n";
print UT "\\def\\ymax\{$max\}\n";
# printf UT ("\\draw\[style=help lines, ystep=%.0f, xstep=300\]
    \(\\xmin,\\ymin\) grid \(\\xmax,\\ymax\)\;\n", $foreachmax);
print UT "\\draw\[->\]␣\(\\xmin,\\ymin\)␣--␣coordinate␣\(x␣axis␣
    mid\)␣\(\\xmax,\\ymin\)␣node\[right\]␣\{\$x\$\}\;\n";
print UT "\\draw\[->\]␣\(\\xmin,\\ymin\)␣--␣coordinate␣\(y␣axis␣
    mid\)␣\(\\xmin,\\ymax\)␣node\[above\]␣\{\$y\$\}\;\n";
printf UT ("\\foreach␣\\x␣in␣\{-10,%.0f,...,$xmaxpluss\}\n",
    $xforeachmax);
# print UT "\\foreach \\x in \{0,50,...,300\}\n";
print UT "\\node␣at␣\(\\x,␣\\ymin\)␣\[below\]␣\{\\x\}\;\n";
printf UT ("\\foreach␣\\y␣in␣\{0,%.0f,...,$maxpluss\}\n",
    $foreachmax);
print UT "\\node␣at␣\(\\xmin,\\y\)␣\[left\]␣\{\\y\}\;\n";
# Line
print UT "\\draw\[color=blue,thick\]␣plot\[smooth,mark=,mark␣size
    =1pt\]␣coordinates\{";
# Bars
# print UT "\\draw[ycomb,color=gray,line width=0.1cm] plot
    coordinates\{";

for ( $k = 0; $k <= $#hull_count; $k++){

    if ( $hull_count[$k] ){
        # skriver ut verdien dersom den finnes
        print UT "\($k,$hull_count[$k]\)";
    } else {
        # skriver bare ut 0 dersom den er tom
            print UT "\($k,0\)";
        }
}
print UT "\}\;\n";
print UT "\\begin\{scope\}\[shift=\{\(305,$legendplaced\)\}\]\n";
print UT "\\draw\[color=blue,thick\]␣\(0,0\)␣--␣\(5,0\)␣--␣\(5,0\)
    ␣node\[right,blue\]\{$figurename\}\;\n";
print UT "\\end\{scope\}\n";
print UT "\\node\[below=0.5cm\]␣at␣\(x␣axis␣mid\)␣\{$xlable\}\;\n"
    ;
print UT "\\node\[rotate=90,␣above=1cm\]␣at␣\(y␣axis␣mid\)␣\{
    $ylable\}\;\n";
print UT "\\end\{tikzpicture\}\n";
print UT "\\caption\{$gamename\}\n";
print UT "\\end\{figure\}\n";
print UT "\\clearpage\n";
print "finished.\n";
    $count++;
}
print UT "\\end\{document\}\n";
close(UT);
system("pdflatex␣awesome.tex");
```

Listing B.3: GetScore.pl

```
#!/usr/bin/perl
use Statistics::Descriptive;

$t = 85;
$b = 1;
$T = 147;

for($yday = 137; $yday < $T; $yday++){
    $value = qx(sqlite3 gamebase.sqlite "select␣count␣from␣games␣
        where␣wday␣=␣'$yday'␣AND␣game␣=␣'$ARGV[0]'");
    @day = split("\n",$value);
    push(@newarray, $day[$b])
}

$stat = Statistics::Descriptive::Full->new();
$stat->add_data(@newarray);

$N = $stat->percentile($t);
# print "Percentile at $t % $N\n";

$f = $N / $day[$b];
@normalisertdag;
for ( $i = 1; $i <= 288; $i++ ){

$normalisertdag[$i] = $day[$i] * $f;

}


$value2 = qx(sqlite3 gamebase.sqlite "select␣count␣from␣games␣
    where␣wday␣=␣'$ARGV[1]'␣AND␣game␣=␣'$ARGV[0]'");
@day2 = split("\n",$value2);

$score = 0;

for ( $j = 1; $j <= 288; $j++ ){

    if($normalisertdag[$j] > $day2[$j]){
        $score++;
    }


}
print "Score:␣$score\n";
```

```

# foreach (@normalisertdag) {
#             print "$_\n";
#          }
```

Listing B.4: ConvertToBaseFormat.pl

```
#!/usr/bin/perl
open(FILE,"./steam.dat");
open(FILE2,">steam2.dat");

# $count = 0;

while( my $line = <FILE> ){
    if ( $line =~ /(^\d+)/){
        $date = $1;
        ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
            localtime($1);

    }
    if ( $line =~ /(\w+).value (\d+)/){
#       $count++;
        $day = ($yday - 136);
        $game = $1;
        $gamecount = $2;
        print FILE2 "$date".",". "$day".",". "$wday".",". "$game".","
            ."$gamecount\n";

    }

}
close(FILE);
```

Listing B.5: AddToBase.pl

```
#!/usr/bin/perl
use DBI;
my $dbargs = {AutoCommit => 0, PrintError => 1};
my $dbh = DBI->connect("dbi:SQLite:dbname=gamebasev2.sqlite","",""
    ,$dbargs);
if ( not $dbh ){
    die "no␣database:␣$!\n";
    }
open(FILE,"./steam2.dat");
while( my $line = <FILE> ){
    if ( $line =~ /(.*?),(.*?),(.*?),(.*?),(.*?)\s/ ){
            print "insert␣into␣games␣values(NULL,'$1','$2','$3','
                $4','$5')\n";
            $dbh->do("insert␣into␣games␣values(NULL,'$1','$2','$3
                ','$4','$5')");
            if ($dbh->err()) { die "$DBI::errstr\n"; }
            }
    }
```

```
close(FILE);
$dbh->commit();
$dbh->disconnect();
```

Listing B.6: TikzPlot.pl

```
#!/usr/bin/perl
use POSIX qw(ceil);

@RandomNumbers;
@RandomNumbers2;

# Genarate random numbers we want to plot.
$range = 50;
for ( $i = 1; $i <= 300; $i++){
     $random_number = int(rand($range) + 30);
     push(@RandomNumbers, $random_number);

}
$range2 = 30;
for ( $i = 1; $i <= 300; $i++){
     $random_number2 = int(rand($range2));
     push(@RandomNumbers2, $random_number2);
}
# Stats
$max = getMax(@RandomNumbers);
$xdatasize = $#RandomNumbers;
$roundup_max = roundup(($max / 100),100);

# Plot size in pt

$ysize = 150;
$xsize = 300;

# y
$tikzyvalue = ($ysize / $roundup_max);

# x
$tikzxvalue = ($xsize / $xdatasize);


# Topic
$linje1 = "Tikz␣Plot";
$linje2 = "I␣cant␣hear␣you␣over␣the␣sound␣of␣how␣awesome␣this␣plot
     ␣is!";

# Lable names
$ylabel = "y␣axis";
$xlabel = "x␣axis";

# Legend name
$enlegend = "Random␣numbers";

# Captio
```

```
$caption = "Random␣numbers";


######################

open(UT,">tikzplot.tex")or die;
print UT "\\documentclass\[11pt,english,a4paper\]\{report\}\n";
print UT "\\usepackage\[usenames,dvipsnames,pdftex\]\{xcolor\}\n";
print UT "\\usepackage\{tikz,ifthen\}\n";
print UT "\\usepackage\[utf8\]\{inputenc\}\n";
print UT "\\usepackage\[acadian\]\{babel\}\n";
print UT "\\renewcommand\{\\familydefault\}\{\\sfdefault\}\n";
print UT "\\begin\{document\}\n";
print UT "\\begin\{figure\}\n";
print UT "\\centering\n";
# $tikzxvalue og $tikzyvalue bestemmer lengden 1 x og 1 y har i pt
    .
print UT "\\begin\{tikzpicture\}\[x=$tikzxvalue" . "pt,y=
    $tikzyvalue" . "pt\]\n";
# Her defineres hva ymax, ymin, xmax og xmin er.
print UT "\\def\\xmin\{0\}\n";
print UT "\\def\\xmax\{300\}\n";
print UT "\\def\\ymin\{0\}\n";
print UT "\\def\\ymax\{" . $roundup_max . "\}\n";
# Her tegnes linjen rundt plotten
printf UT ("\\draw\[style=help␣lines,␣ystep=%.0f,␣xstep=300\]␣\(\\
    xmin,\\ymin\)␣grid␣\(\\xmax,\\ymax\)\;\n",$roundup_max);
# Her tegnes y og x aksene.
print UT "\\draw\[->\]␣\(\\xmin,\\ymin\)␣--␣coordinate␣\(x␣axis␣
    mid\)␣\(\\xmax,\\ymin\)␣node\[right\]␣\{\$x\$\}\;\n";
print UT "\\draw\[->\]␣\(\\xmin,\\ymin\)␣--␣coordinate␣\(y␣axis␣
    mid\)␣\(\\xmin,\\ymax\)␣node\[above\]␣\{\$y\$\}\;\n";
# Her tegnes overskriftene inn.
print UT "\\node␣at␣\(150,␣\\ymax\)␣\[above=0.40cm,␣scale=1.2\]␣\{
    $linje1 }\;\n";
print UT "\\node␣at␣\(150,␣\\ymax\)␣\[above=0.01cm,␣scale=1.1,␣
    font=\\footnotesize\]␣\{ $linje2 }\;\n";
# Her tegnes alle tiks inn paa y og x aksen.
print UT "\\foreach␣\\x␣in␣\{0,50,...,300\}␣\\draw␣\(\\x,1pt\)␣--␣
    \(\\x,-3pt\)␣node\[anchor=north\]␣\{\\x\}␣\;\n";
print UT "\\foreach␣\\y␣in␣\{0,10,...,$roundup_max\}␣\\draw␣\(1pt
    ,\\y\)␣--␣\(-3pt,\\y\)␣node\[anchor=east\]␣\{\\y\}\;\n";
# Her tegnes kordinatene til plottet inn. (tregner ikke
    kordinater, kan ogsaa bruke filer med data.)
# print UT "\\draw\[smooth,color=Blue,thick\] plot\[mark=,mark
    size=1pt\] coordinates\{";


# for ( $i = 1; $i <=300; $i+=10){
#     print UT "\($i,$RandomNumbers[$i]\) ";
# }
# print UT "\}\;\n";

```

```
print UT "\\draw\[color=Blue,thick,smooth\]␣plot\[mark=,mark␣size=1pt\]␣coordinates\{";
for ( $k = 1; $k <= 288; $k+=50){
        print UT "\($k,$RandomNumbers[$k]\)";
}
print UT "\}\;\n";

for ( $k = 1; $k <= 288; $k+=50){
    if($RandomNumbers[$k] < $RandomNumbers2[$k]){
        print UT "\\draw\[color=red,thick\]␣plot\[mark=*,mark␣size=0.5pt\]␣coordinates\{\($k,$RandomNumbers[$k]\)\}\;\n";
    }
}

print UT "\\draw\[color=black,thick,smooth\]␣plot\[mark=,mark␣size=1pt\]␣coordinates\{";
for ( $k = 1; $k <= 288; $k+=50){
    print UT "\($k,$RandomNumbers2[$k]\)";
}

print UT "\}\;\n";


# Her tegnes legend inn
print UT "\\begin\{scope\}\[shift=\{\(305,". ($roundup_max / 2) ."\)\}\]\n";
print UT "\\draw\[color=blue,thick,scale=1.5\]␣\(0,0\)␣--␣\(5,0\)␣--␣\(5,0\)␣node\[right,black,scale=0.8\]\{$enlegend\}\;\n";
print UT "\\end\{scope\}\n";
# Her tegnes navn paa y og x akse.
print UT "\\node\[below=0.60cm\]␣at␣\(x␣axis␣mid\)␣\{$xlabel\}\;\n";
print UT "\\node\[rotate=90,␣above=1cm\]␣at␣\(y␣axis␣mid\)␣\{$ylabel\}\;\n";
print UT "\\end\{tikzpicture\}\n";
print UT "\\caption\{$caption\}\n";
print UT "\\end\{figure\}\n";
print UT "\\end\{document\}\n";
close(UT);
system("pdflatex␣tikzplot.tex");

sub getMax {
    my $max;
    foreach my $item (@_){
        if ( ( $max and $max < $item ) or not $max ){
            $max = $item;
        }
    }
    return $max;
}
foreach $number ( @numbers ){

```

```perl
    print "$number␣->␣" . roundup($number,100) . "\n";

}

sub roundup
{
    my $num = shift;
    my $roundto = shift || 1;

    return int(ceil($num/$roundto))*$roundto;
}
```

# Appendix C

# Tabel of best beta value on all games

| Game | best b | Mean | Median | Distance | Min | Max | Distance |
|---|---|---|---|---|---|---|---|
| Counter Strike | 109 | 279.00 | 283 | 4.00 | 247 | 288 | 41 |
| Counter Strike Source | 97 | 287.10 | 288 | 0.90 | 281 | 288 | 7 |
| Call of Duty Modern Warfare 2 Multiplayer | 109 | 286.05 | 288 | 1.95 | 255 | 288 | 33 |
| Football Manager 2010 | 42 | 267.14 | 273 | 5.86 | 208 | 288 | 80 |
| Team Fortress 2 | 127 | 285.95 | 288 | 2.05 | 274 | 288 | 14 |
| Left 4 Dead 2 | 129 | 287.67 | 288 | 0.33 | 285 | 288 | 3 |
| Garry s Mod | 93 | 284.43 | 287 | 2.57 | 255 | 288 | 33 |
| Empire Total War | 96 | 273.86 | 283 | 9.14 | 217 | 288 | 71 |
| Condition Zero | 109 | 286.14 | 288 | 1.86 | 276 | 288 | 12 |
| Battlefield Bad Company 2 | 128 | 279.14 | 288 | 8.86 | 188 | 288 | 100 |
| Borderlands | 127 | 285.10 | 288 | 2.90 | 246 | 288 | 42 |
| Call of Duty Modern Warfare 2 | 109 | 283.24 | 287 | 3.76 | 258 | 288 | 30 |
| Day of Defeat Source | 118 | 285.62 | 287 | 1.38 | 270 | 288 | 18 |
| Football Manager 2009 | 42 | 263.57 | 279 | 15.43 | 210 | 288 | 78 |
| Napoleon Total War | 109 | 277.57 | 285 | 7.43 | 220 | 288 | 68 |
| Left 4 Dead | 129 | 287.71 | 288 | 0.29 | 286 | 288 | 2 |
| Mount Blade Warband | 139 | 273.90 | 283 | 9.10 | 167 | 288 | 121 |
| Warhammer 40 000 Dawn of War II Chaos Rising | 77 | 281.90 | 285 | 3.10 | 247 | 288 | 41 |
| Killing Floor | 96 | 281.38 | 286 | 4.62 | 233 | 288 | 55 |
| Half Life 2 Deathmatch | 130 | 280.24 | 288 | 7.76 | 238 | 288 | 50 |
| Half Life 2 | 83 | 283.38 | 288 | 4.62 | 245 | 288 | 43 |

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| Warhammer 40 000 Dawn of War II | 83 | 285.76 | 288 | 2.24 | 262 | 288 | 26 |
| Dragon Age Origins | 129 | 275.29 | 287 | 11.71 | 210 | 288 | 78 |
| Day of Defeat | 107 | 284.81 | 288 | 3.19 | 249 | 288 | 39 |
| Portal | 89 | 281.38 | 285 | 3.62 | 243 | 288 | 45 |
| Supreme Commander 2 | 129 | 282.05 | 286 | 3.95 | 257 | 288 | 31 |
| Just Cause 2 | 85 | 281.33 | 286 | 4.67 | 251 | 288 | 37 |
| Sid Meier s Civilization IV Beyond the Sword | 56 | 287.86 | 288 | 0.14 | 287 | 288 | 1 |
| Torchlight | 28 | 272.38 | 279 | 6.62 | 197 | 288 | 91 |
| Grand Theft Auto IV | 31 | 278.48 | 282 | 3.52 | 240 | 288 | 48 |
| Metro 2033 | 69 | 285.43 | 288 | 2.57 | 273 | 288 | 15 |