Time Series Prediction for Algorithmic Rescaling in the Cloud

Björn Elmers

Master's thesis
2016:E3

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Time series prediction for algorithmic rescaling in the cloud.

Author: Björn Elmers

Supervisor: Kalle Åström

Assistant supervisor: Michael Nordström

Examiner: Magnus Oskarsson

Acknowledgements:
Alex Krasnukhin,
Anders Haglund,
Sofia Henryson

February 4, 2016

ABSTRACT

The main goal of this thesis is to predict the number of players in some instance of DICE's gaming platforms, e.g. Battlefield 3 for PC, 15 minutes in the future. This prediction may be used by the company when buying on-demand cloud servers to tell them how much they need to buy. Several different prediction models are examined and evaluated on historical data from 2011 to 2015. Another focus point is how to detect and handle outliers. The thesis also tries to create a general understanding of player behaviour by using different data separations and statistical methods. From the company's perspective an auto regressive model of order 100 produces the best result. It is shown that with this model it would be beneficial for the company to use cloud servers instead of physical servers if their mean abundance in server capacity with physical servers is greater than 4134 players.

The work was conducted primarily at DICE (EA Digital Illusions CE AB) in Stockholm from 2015-09-07 to 2016-01-29.

SAMMANFATTNING

Huvudsyftet med denna rapport är att förutsäga antalet spelare i någon instans av DICE: s spelplattformar, t.ex. Battlefield 3 för PC, 15 minuter i framtiden. Denna förutsägelse kan användas av företaget vid köp av molnservrar on-demand för att visa dem hur mycket de behöver köpa. Flera olika prognosmodeller undersöks och utvärderas på historisk data från 2011 till 2015. En annan fokuspunkt är hur man upptäcker och hanterar extremvärden. Rapporten försöker också skapa en allmän förståelse av spelarbeteende genom att separera datan på olika sätt och genom olika statistiska metoder. Från företagets perspektiv ger en autoregressiv modell av ordning 100 det bästa resultatet. Det visar sig att med denna modell skulle det vara fördelaktigt för företaget att använda molnservrar istället för fysiska servrar om deras genomsnittliga överflöd i serverkapacitet med fysiska servrar är större än 4134 spelare.

Examensarbetet genomfördes primärt hos DICE (EA Digital Illusions CE AB) i Stockholm från 2015-09-07 till 2016-01-29.

CONTENTS

1

INTRODUCTION

DICE is a Swedish software company that develops computer games. They are responsible for the Battlefield series, which counts as one of the biggest Swedish successes in computer games of all time. The latest release, Battlefield 4, has sold over 10 million copies according to [16]. To make their games run smoothly at all times they have to make sure that their servers always have enough computing power. If the company buys their own servers it means that they must buy enough server capacity to handle the times when there are the most players online. This means that during large parts of the day and the year there is an abundance of computer power. In Table 1, one can see as an example, that only between 07:00 and 13:00 it differs about 80.000 players. Today, more and more services are moving their computing power from physical servers owned by themselves to the cloud. By predicting how much server capacity that is needed at a given time (not too far into the future), DICE will be able to buy and sell as much space in the cloud as they need to ensure a nice game experience for their users and still optimizing their costs. *Optimizing this kind of short predictions for renting on-demand cloud servers is the main focus of this paper.*

Table 1: Peak Simultaneous User's (PSU) at different times during September 1, 2015, calculated over all consoles for Battlefield 3. It uses the maximum value per hour. Notice the big variations.

Hour	Players
06:00	102,404
07:00	93,744
08:00	99,681
09:00	110,194
10:00	122,109
11:00	138,337
12:00	155,360
13:00	171,434

Since it is approximately six times cheaper for the company to use their own physical server than to rent one from a cloud service, a big part of the server capacity could remain on company servers. Because of this the company also needs to know how many servers they should buy for themselves. This motivates player predictions on a larger timescale further into the future since buying physical servers naturally takes much longer time than buying cloud space. One interesting area is prediction on how many players the game will get shortly after the launch. Another interesting area is predictions of the trend a few days or even a month ahead to see if the capacity from the company's own servers is enough. This kind of longer prediction is not the main focus of this thesis but *an analysis is made of how big part of the entire server capacity that should come from the cloud and how much that should come from their physical servers.*

A third area that this paper tries to contribute to, *is to create a general understanding about player behaviour.* The paper presents a few different analyses and tools that hopefully helps with this task.

1.1 RELATED WORK

The subject of cloud computing is discussed in several papers. The number of papers that focuses on the prediction of the workload is limited though. E.g. [1] and [2] are excellent papers that address the full problem of cloud computing in a wide perspective with good solutions on load balancing, architecture and cost-efficiency, but their parts about prediction is more like a sidetrack or discussed as future work. In [3] the subject of prediction is examined with a pattern matching algorithm and is possibly a good solution but its missing a real comparison with other prediction techniques. In [4] an AutoRegressive Moving Average (ARMA) filter is used for the prediction and it gives an acceptable result for their purpose but there is no comparison between different algorithms in this paper either. The most related paper found is probably [5] that actually does a comparison between eight different algorithms and spends some time discussing prediction techniques. It divides players into different groups depending on their play style and does not focus on how many players there are (as in this thesis) but how much they interact with the game.

This paper focuses specifically on DICE games, which prediction techniques that performs best on them and how well they predict. Another thing that is covered by this paper but not by the others, is the parts surrounding the prediction, such as data cleaning and handling outliers.

1.2 CHALLENGES IN CLOUD COMPUTING

There are several challenges in the area of cloud computing. Of the following areas workload prediction is the only one treated by this thesis.

When the company purchases server capacity it is done in the form of so called boxes. These are bought in different sizes depending on what the box will be used for. Some boxes perform better for game servers with a small number of players while some boxes perform better on game servers with many players. The different kind of boxes has room for different number of players and has different costs. How many different boxes that should be bought is an optimization problem of its own and will not be discussed in this paper. To simplify the evaluation part of this work this paper does not take boxes into account at all and instead assumes that the number of players translates directly to server capacity.

Each box may house several game instances and each game instance houses several players. This leads to saturation problems on both the boxes and the game instances. To prevent half empty boxes and game instances some kind of consolidation mechanics is necessary.

Another issue is that cloud servers are rented on an hourly basis. This means that when you decide to start scaling down a system it could take up to an hour before you get to the optimal server capacity. It may or may not be a big deal but at least it brings more complexity to the subject of cost optimization.

The main focus of this paper is workload prediction. What it is does not cover is the translation from PSU (Peak Simultaneous Users) to workload, instead it assumes that the PSU translates directly to workload. In reality the translation ratio differs. For example some game modes may be more intense which could mean that each player need more server capacity. It could be beneficial to analyse the PSU of different game modes separately but that is not covered in this paper.

2

DATA AND EVALUATION

2.1 THE DATASET

The dataset that has been analysed consists of the PSU for 15 minute intervals for Battlefield 3 (BF3), Battlefield 4 and Battlefield Hardline since their respective launches in 2011, 2013 and 2015. Each sample contains the PSU, a timestamp and information about which console and game the sample comes from. All plots, data and results shown in this paper origins from BF3 which is the oldest game of the three. Furthermore, with exception to section 6.2 ”Separating the data per console”, all plots, data and results is also restricted to PC (Personal Computer) usage. A plot of the first half of the analysed dataset can be seen in Figure 1. The quality of the data is quite good but as can be seen in the plot there exists some irregularities, e.g. where the curve spikes down to zero or when it drops a longer period (between Q4-12 and Q4-12), that is either missing values or outliers which is an issue that this paper addresses quite extensively. Another thing to notice is the decreasing trend and the fluctuations which also will be addressed later in this paper.

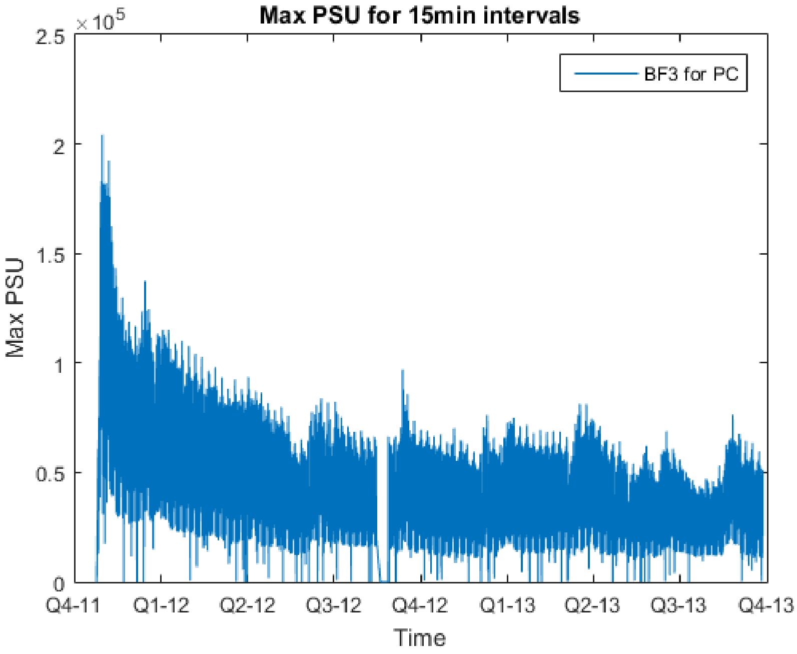

Figure 1: A plot of the analysed dataset for PC and BF3. Notice the general decreasing trend and the fluctuations. Also notice the drops, i.e. when the curve goes to zero. These drops indicates an irregularity, i.e. that either the value is missing or that there has been a technical issue.

Interval of 15 minutes is a suiting interval to work with since a rescaling to the cloud often takes less than 15 minutes [10] and for DICE it will probably land somewhere around 5 minutes. The use of a longer than needed interval at least will not cause a too optimistic result.

2.2 EVALUATION

An important question in this work has been how the predictions from each model should be evaluated. Several values would be interesting to compare and it is not certain that one model will be the best one in all aspects or even on all datasets. What evaluation techniques that are most important also differs from which person that look at it. A mathematician would say one thing while someone from DICE could say something different.

2.2.1 *A mathematician's perspective*

From a science- and mathematical perspective the proper way to evaluate a prediction would be to look at standard statistical evaluation values such as the Root Mean Square Error (RMSE),

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}},$$

the Mean Absolute Percentage Error (MAPE),

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i},$$

or the Median Absolute Percentage Error (MdAPE),

$$MdAPE = \underset{i=1,n}{\text{median}}(|\frac{y_i - \hat{y}_i}{y_i}| \cdot 100),$$

where y_i is the actual value at sample i, $\hat{y}_i$ is the predicted value at sample i and n is the number of samples. Another interesting thing to look at could be the confidence interval of the prediction.

Using MAPE as an evaluation technique was suggested by [11] and it has some nice properties for example if you want to study the relative error and not the absolute difference between forecasted and actual values. An attempt to clean the data from outliers will be done later in the paper but unfortunately it can not be confirmed that all outliers is removed. Therefore the MdAPE was included as an evaluation technique due to the fact that it is more resistant to outliers than MAPE [13]. Another drawback of the MAPE is that it can not handle zero as the true value which will sometimes be the case when evaluating outliers. This can be seen later in the Result section where the MAPE will produce so called inf values which is short for infinity. The MdAPE also has a risk of producing infinity values but that would only occur if at least half of the dataset consisted of zeros which is not the case here.

2.2.2 *The company's perspective*

From the company's- and an economic perspective the goal is to get a prediction that minimizes the necessary server capacity but ensures that it does not get to low. If the company has too low server capacity it costs them a lot in terms of player dissatisfaction and in the long term lower sell numbers. Instead of trying to calculate how big this cost is (which is really difficult), a requirement was put on the prediction to not produce too low result more than 10 samples a year. In percentage that equals

$$\frac{10}{4 * 24 * 365} \approx 0.9997 = 99.97\%.$$

This means that by using a lower bound confidence interval with a confidence level at 0.9997 only 10 samples per year is expected to fall below the interval.

The number in the confidence interval will then correspond to the minimum amount of extra capacity above the predicted value the company will have to buy to fulfill the requirement. If the mean of the residuals from each model was zero, the above mentioned confidence interval would also correspond to how much mean abundance the company would need to fulfill the requirement. Although the residuals from the models will not always be centered around zero and therefore the confidence interval is not enough. The cost function is created by also using the mean of the residuals and is described below.

The cost function *By adding the mean residual to the lower bound confidence interval it leads to a resulting value of how much mean abundance the company would need while still meeting the requirement specified above.*

3

PREPROCESSING

Preprocessing the data is an important part of forecasting. How the data is cleaned, transformed and handled in general before using it in a model could make a great impact on the result.

One goal of the preprocessing is to separate different behaviours from each other and allow them to be treated differently. By doing this the datasets decreases in complexity and becomes easier to analyse. Another goal is to remove the effects of outliers.

In this work the data has been separated in a few different ways explained in the sections below. An additional separation (dividing per day in week) is discussed in the section about analysis, although this separation was not implemented in the main preprocessing step but as a sidetrack.

3.1 TRIMMING THE DATASET

Since the dataset used included data from the very start of each game, the beginning of each game's dataset looked very different from the rest of the dataset. To remove the effects of the special player behaviour when the game is all new the first 1000 samples was removed. This is illustrated in Figure 2. In the same plot it can also be noticed some smaller irregularities, e.g. between 11/02 and 11/03, that could not be seen earlier in Figure 1. Yet another thing to notice is the difference between regular weekdays, like Wednesday and Thursday (11/04 and 11/05) and a weekend, like Friday and Saturday (11/06 and 11/07). These differences will be examined closer later in this paper.

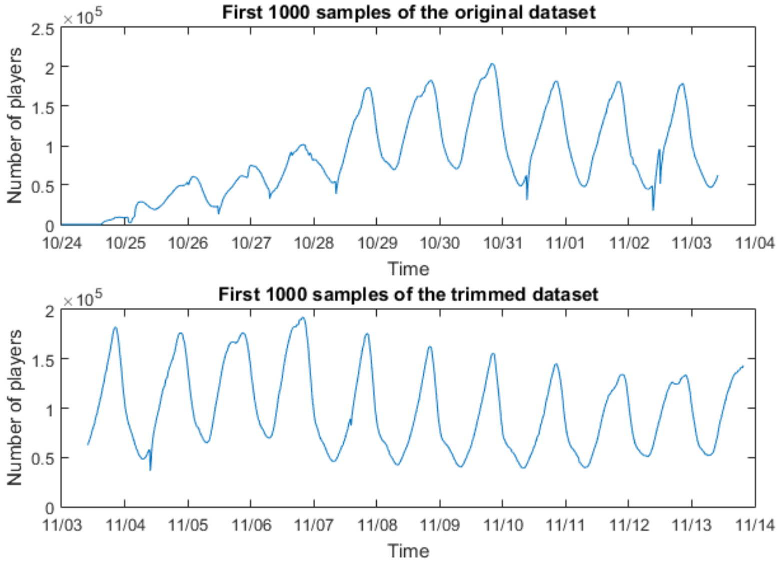

Figure 2: The first 1000 samples of the original dataset for BF3 and PC together with the first 1000 samples for the trimmed version. Note that the curve looks different during the first four or five days.

3.2 PLATFORM AND GAME SEPARATION

Next preprocessing step was to divide all data into different sets for each game and for each console. It could be argued that it would be interesting to perform an analysis over different consoles since the company could want to buy common cloud space over multiple platforms but the chosen way to do this was to first analyse each console by itself and then combine the results. An evaluation of whether this was a good choice or not is presented in the result section.

That the data should be separated into different sets based on the game is clear though, at least from the company's view. This is because each game has its own budget and buying common cloud space for several games is therefore not an option.

3.3 TRAINING- AND TEST SET SEPARATION

When using a model for prediction of new data, it is inappropriate to evaluate the model on the data that has been used in estimation of the model's parameters [17]. Because of this the data was separated in a training set and a test set. The training set was used when analysing the dataset and estimating parameters in the model while the test set only was used for evaluation. The first half of the dataset was used as the training set and the second half of the dataset was used as the test set.

3.4 MISSING VALUES AND OUTLIERS

In the used data there exists three categories of values. Good values, outliers and missing values.

The majority of the data consists of the so called good values. The good values are the values that are not influenced by server issues or other technical problems.These values should mostly depend on player behaviour.

The second category is the outliers. The outliers are the values that are influenced by server issues or other technical problems. In sections with such values, the player behaviour is shadowed by a decreased server capacity. Although in the very end of sections like this, when the server issues have disappeared but the PSU still has not reached its normal level, the player behaviour comes into play again but it looks different than with good values. Therefore it should still be analysed separately.

The third category is the missing values. This category is simply the places in the timeseries where values are missing. Since nothing is known about these values there is no point in trying to analyse them.

The phenomenon with outliers and missing values is a common problem when working with data from real life and it has to be handled properly to prevent that they disturb the forecasting procedures [6]. Several approaches like *mean substitution*, *pairwise deletion* and *casewise deletion* have been proposed or examined by e.g. [6] and [7]. Another way is to try to interpolate the sections holding outliers and with a good interpolation technique this could maybe be a preferable solution, at least if the sections with outliers or missing values are short. In this work though, all missing values were ignored in the evaluation part and the evaluation of outliers and good values was done separately. The reason for not trying to interpolate the missing values or handling them with any other smart technique, was that the available dataset was big enough and the loss of these values would probably not be noticeable. For the analysed dataset only 1915 values was marked as missing which corresponds to 1.4%. The reason for still evaluating the outliers was that the end of these sections could still be interesting as explained above.

The first approach to handle the outliers and missing values was by in some sense ignoring them in the prediction part. The prediction at the samples with outliers or missing values was set to the true value instead of the predicted. This naturally made the resulting predictions more accurate. Since all models increased in accuracy in the same way this did not matter due to the fact that what is interesting in this case is to compare models, not how well a specific model predicts.

The second approach (which was used with all models) was to move the responsibility of handling outliers and missing values from the models to the evaluation part. This approach was chosen to be the final one since it allowed evaluation of both the outliers and the good values. Some models were good at predicting sections with outliers and bad at predicting sections with good values and this kind of evaluation was not possible with the first approach.

3.5 DEPENDENCIES ON OUTLIERS OR MISSING VALUES

One detail that made the problem with missing and outliers a little bit more complex was that before you flag a value as a good value, first you have to make sure that it does not depend on an outlier or a missing value in any of the models. For instance the Last Week model uses values from one week back and if any of those values is not a good value then the prediction from that model would be influenced by an outlier or a missing value. To come around this issue a window was created described as

$$window = [t - k \cdot week - 1, t - k \cdot week, t - k \cdot week + 1, t - i],$$

where $week = 24 \cdot 4 \cdot 7$ (the number of samples in a week), t is the current sample, $1 \leq k \leq 5$ (to cover the model in section 5.5) and $1 \leq i \leq 25$ (to cover the AutoRegressive model of order 25). This window covered almost all models dependencies with the exception of the AutoRegressive model of order 100 (AR100). The AR100 model uses values 100 samples back which would have made the window too big and the remaining dataset too small. Note that the possible impact of not covering the AR100 model completely will be that some outliers could remain in the evaluation step. That would mean a worse result for the model. The same window was used for all models to be able to compare their results in a fair way.

3.6 FINDING OUTLIERS

To be able to handle missing values or outliers they have to be identified. To identify missing values is trivial but to find outliers is a little bit harder. If the servers are completely down then of course it is easy, then outliers will be zero. The issue is to find when there are server problems that just prevent some players to play. The first step in solving this problem was to manually set out flags on all values that looked like outliers. This was needed to be able to evaluate the algorithmic solution for finding the outliers. The manual flagging was done by scrolling through parts of the dataset while looking for

```python
for i in range(weekly_period, len(psumax)):
    if math.isnan(psumax[i]) or psumax[i] == "nan":
        flags[i] = float("NaN")
    elif psumax[i] == 0.0:
        flags[i] = "True"
    elif abs((psumax[i] - psumax[i - weekly_period]) / psumax[i]) >
        threshold:
        flags[i] = "True"
    else:
        flags[i] = "False"
```

Figure 3: First algorithm for finding outliers in the dataset written in Python. After the optimization described below, the threshold was set to 1.0. Although, since this algorithm was not the final one that threshold value was only used in early testing.

irregularities. Most of the large player drops was found but some of the smaller ones were probably missed.

The first approach with an algorithmic solution was to look at the percentage difference between the value at hand and the value one week back in the timeseries. If the difference got bigger than a certain value then it got flagged as a outlier.

The motivation for this solution was that the datasets had such strong weekly period. The question remained what value to use as a threshold. To analyse this, different values was used and then the resulting flagged values was compared to the manual flags. The comparison was categorized into four different classes; true Positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) [14]. True positives are the ones that are flagged correctly as a outlier, true negatives are the ones that are flagged correctly as a good value, false positives are the ones that are flagged incorrectly as an outlier and false negatives are the ones flagged incorrectly as a good value.

With these four classes the true positive rate (TPR or sensitivity) and the false positive rate (FPR or fall-out) could be calculated as

$$TPR = \frac{TP}{TP + FN}$$

and

$$FPR = \frac{FP}{FP + TN}.$$

The result was presented in a Receiver Operating Characteristic (ROC) plot seen in Figure 4. A ROC plot shows the false positive rate on the x-axis and the true positive rate on the y-axis and hopefully gives a hint of how to set your threshold value, [15]. In many ROC-plots the curve starts in orig inwhich neither the red or the blue curve does in Figure 4. The reason for this is that many of the bad values are zero and with the used algorithms all zero values will be found regardless of the threshold value.

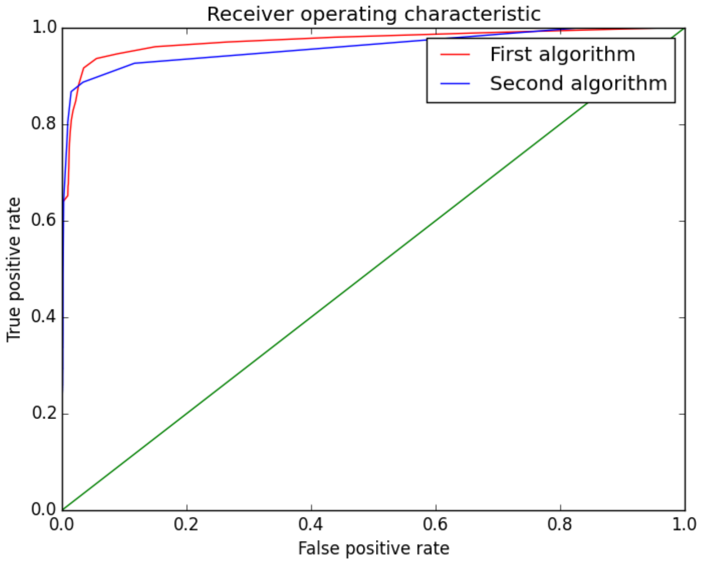

Figure 4: Plot with receiver operating characteristic. Algorithm in Figure 3 was used for the red line and algorithm in Figure 5 was used for the blue line. The green line is just a line from (0,0) to (1,1). Notice that the blue and red line starts a bit over 0.6. This is because of zeros in the dataset.

With an optimal algorithm the blue curve would have intersected the point $(0.0, 1.0)$ which would have meant that all outliers were found and that all found values were outliers. Now when that is not the case one has to decide how important FPR is compared to TPR. Since the dataset at hand is as big as it is and the main goal is to analyse the good values, the TPR is much more important than the FPR.

The second approach with an algorithmic solution was to look at the percentage difference between the value at hand and the previous value instead of the value one week back as in the first approach. As seen in the implementation in Figure 5 another difference from the first algorithm was introduced. The algorithm remembers the PSU value of the last sample that was flagged as good. If the previous sample was marked as an outlier then the PSU from the memory is used instead.

```
for i in range(weekly_period, len(psumax)):
    if math.isnan(psumax[i]) or psumax[i] == "nan":
        flags[i] = float('NaN')
    elif psumax[i] == 0.0:
        flags[i] = "True"
    elif flags[i - 1] == "True":
        if (psumax[i] - last_good_value) / psumax[i] < threshold:
            flags[i] = "True"
        else:
            flags[i] = "False"
            last_good_value = psumax[i]
    else:
        if (psumax[i] - psumax[i - 1]) / psumax[i] < threshold:
            flags[i] = "True"
        else:
            flags[i] = "False"
            last_good_value = psumax[i]
```

Figure 5: Second algorithm for finding outliers in the dataset written in Python. If the last value was an outlier then the difference is calculated from the last good value instead.

As can be seen by comparing the red and the blue lines in Figure 4, the second algorithm did not improve much. But it opened up an opportunity to introduce a second threshold value for when the last value was flagged as an outlier. A third algorithm was therefore implemented as seen in Figure 6. A few curves with different threshold values was plotted in Figure 7. Since a high TPR value is most important, the red curve with $second\_threshold = 0.2$ was chosen as the best one and was used as the final algorithm in the preprocessing step. When comparing the actual numbers on TPR and FPR, for $TPR > 0.99$ the first algorithm got $FPR \approx 0.69$ with $threshold = 0.05$, while the final algorithm got $FPR \approx 0.06$ with $threshold = -0.2$.

```
for i in range(1, len(psumax)):
    if math.isnan(psumax[i]) or psumax[i] == "nan":
        flags[i] = float('NaN')
    elif psumax[i] == 0.0:
        flags[i] = "True"
    elif flags[i - 1] == "True":
        if (psumax[i] - last_good_value) / psumax[i] < (threshold +
            second_threshold):
            flags[i] = "True"
        else:
            flags[i] = "False"
            last_good_value = psumax[i]
    else:
        if (psumax[i] - psumax[i - 1]) / psumax[i] < threshold:
            flags[i] = "True"
        else:
            flags[i] = "False"
            last_good_value = psumax[i]
```

Figure 6: Third and final algorithm for finding outliers in the dataset written in Python. Almost identical to 5 but has a second threshold value.

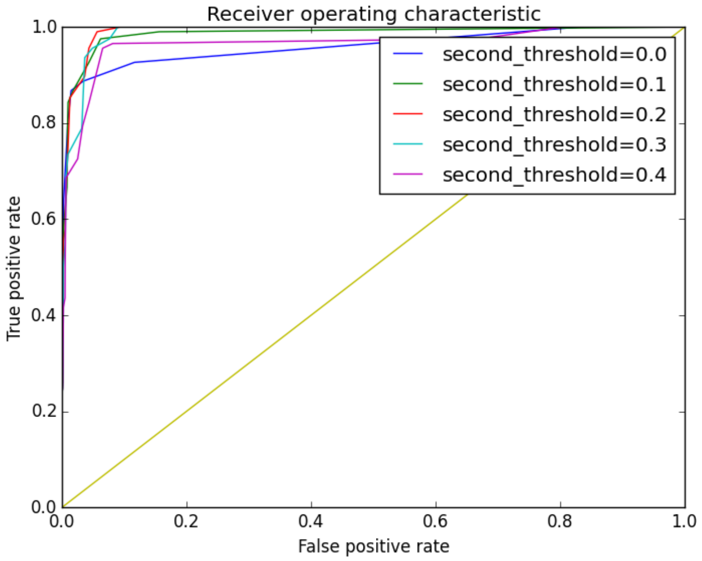

Figure 7: Plot with receiver operating characteristics with different values on the second threshold. Algorithm in Figure 6 is used. The straight yellow line is just a line from (0,0) to (1,1). The red line gets a very high TPR early which makes it the best option.

4

ANALYZING THE DATA

The goal of this chapter is to create a general understanding of the player behaviour. With a solid understanding about the player behaviour it later becomes easier to come up with ideas for models and to understand why certain models behave the way they do. To make the patterns in the dataset clearer the first thing done was to center the data.

4.1 SEASONALITY

By thinking of how people play computer games it is easy to guess that there will be some kind of both daily and weekly seasonality. A guess is that people generally play more during evenings than during the middle of the day and that people play more during weekends than during the rest of the week. By just looking at the data it became clear that the first guess was correct, the daily oscillations are very easy to spot. To confirm the suspicions about a weekly seasonality a MA (Moving Average) filter was put on the data described as

$$\text{season} = \frac{y * ones}{N},$$

where N is the length of the window and *ones* is a list with length N consisting only of ones. A window with length $4 * 24 = 96$ samples was used to remove the daily oscillations. To see if there existed any other clear seasonality a second ma-filter was put on the data with length $4 * 24 * 7 = 672$ to remove the weekly oscillations. The resulting plot with the centered data and the two filtered versions can be seen in Figure 8. The blue line clearly shows a weekly seasonality which suggests that our assumption was correct. The red line does not show any clear season though. The red line could be viewed as some kind of trend. Most of the ups and downs in the red line could be explained by different marketing campaigns, offers and expansion packs.

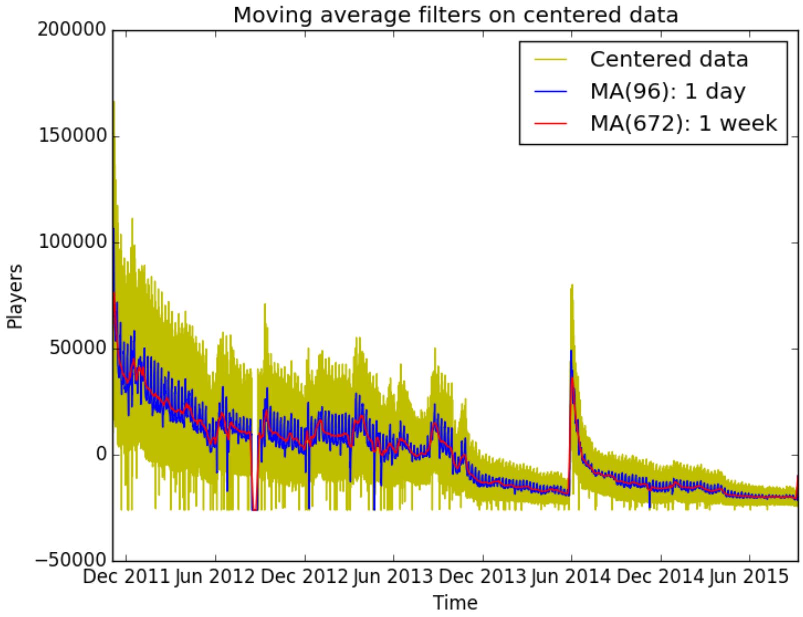

Figure 8: The plot shows the centered original dataset and two sets that has been filtered with a moving average filter with window lengths 96 and 672. Notice the remaining seasonality in the blue line which corresponds to the weekly behaviour and the red line's lack of seasonality.

4.2 DAILY PATTERN

To get a better understanding of how the player behaviour differs over a day, each day was plotted separately in the top plot in Figure 9. There is a clear daily pattern with a dip at around sample number 25 and a peak around sample number 80. The next thing that was examined was if this daily pattern could become clearer by looking at each day in the week separately which can be seen in the bottom plot in Figure 9. It may be hard to separate the colors of each day, but by looking closely it can be seen that Fridays and Saturdays has a later decrease than the rest, Saturdays and Sundays has an earlier increase than the rest and Mondays to Thursdays has very similar patterns. As expected people stop playing later on Fridays and Saturdays and they start playing earlier on Saturdays and Sundays and a little bit earlier on Fridays. The different pattern between each weekday could motivate a different analysis for each weekday.

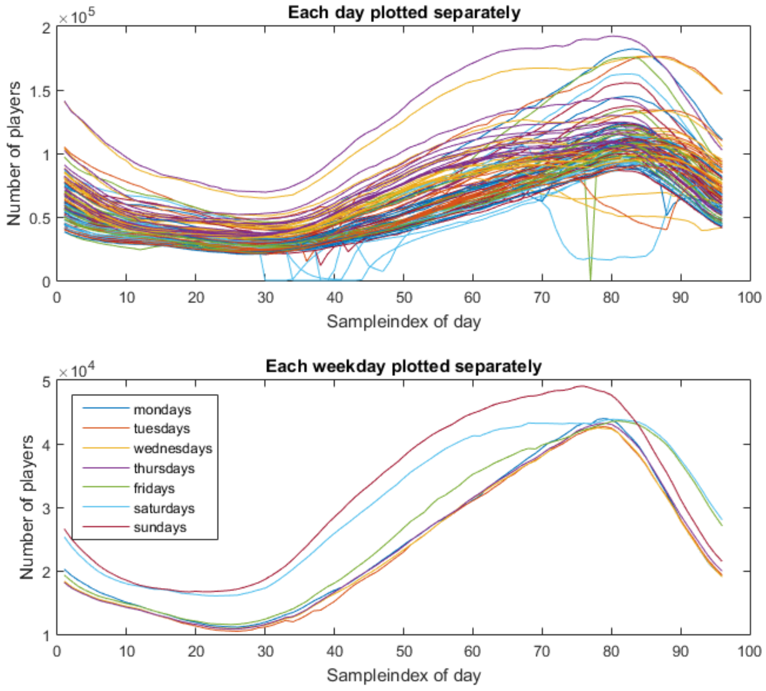

Figure 9: In the top plot each day is plotted by itself. In the bottom plot a mean of each weekday is plotted by itself. Notice the difference between regular weekdays and weekends.

4.3 AUTO CORRELATION FUNCTIONS

To further study the seasonalities in the dataset, the data was visualized with plots of the Auto Correlation Function (ACF) and of the Partial Auto Correlation Function (PACF) to see if the data was stationary or included any seasonalities. By looking at the ACF and PACF it is possible to comment on what kind of model to use. When talking about classical statistical models like an AutoRegressive Integrated Moving Average (ARIMA) it is also possible to comment on what order to use on the different terms in the model which will be done later in the paper. In Figure 10 the unmodified data has been used to see if any differentiation is necessary.

The pattern in the ACF plot is typical for a data that is both strongly seasonal and nonstationary. It does not have a fast decrease and it shows a clear seasonal period. This means that some kind of differencing will be necessary. It could be a nonseasonal difference (Figure 11), a seasonal difference (Figure 12) or both (Figure 13). Which alternative that is best will be determined by looking at the ACF's and the PACF's of all three possibilities. All of these plots together strongly indicates that both a nonseasonal and a seasonal difference should be included [19].

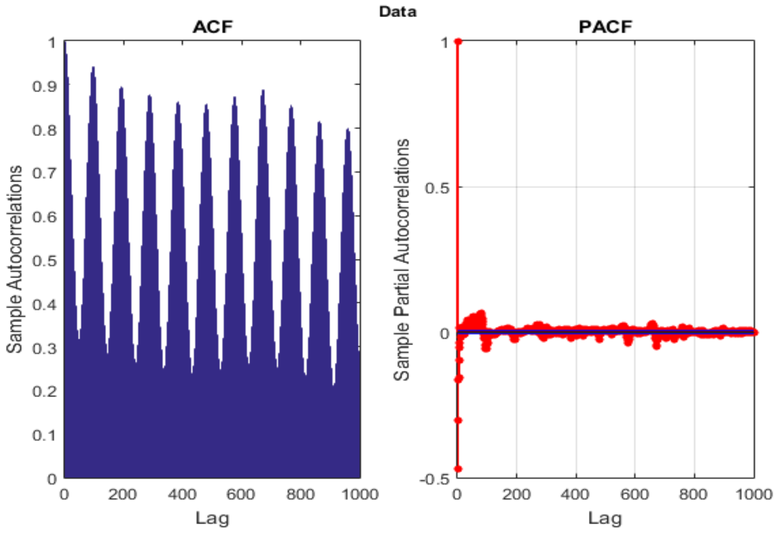

Figure 10: An ACF and PACF plot of the unmodified data. That means that no differentiation has been done. The season in the ACF indicates that some differentiation is necessary to induce stationarity.

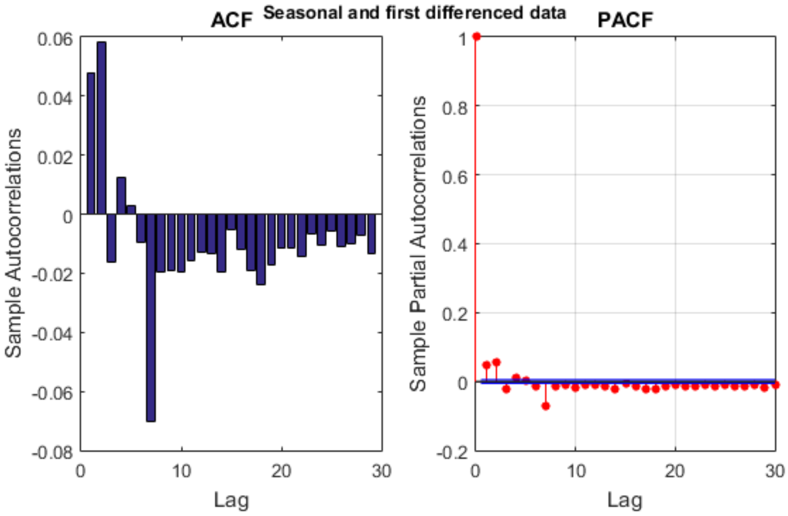

Figure 14: This plot is the same as Figure 13 with the difference that it is zoomed in. Notice the positive spike at lag 1 in the PACF and the cutoff after the spike.

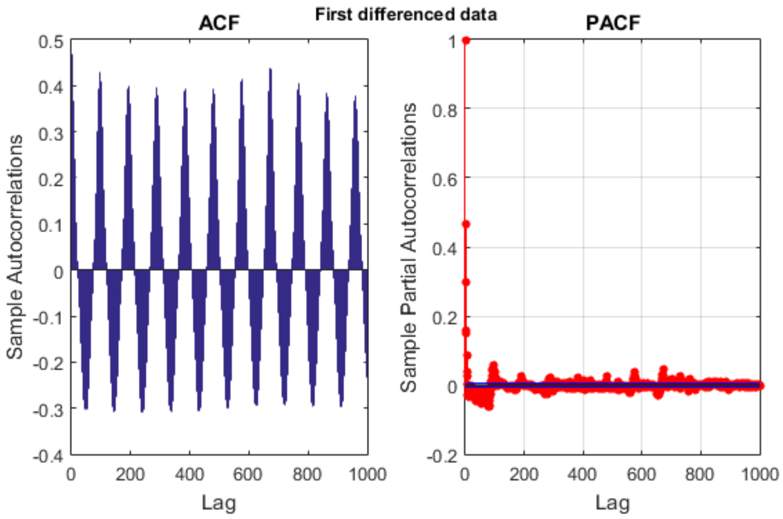

Figure 11: In these plots the data has been modified with a first difference. That means that $y_t = Y_t - Y_{t-1}$ where y_t us the differentiated data at time t and Y_t is the original data at time t. It seems that much of the seasonality and the stationarity is still there. It has decreased by half though which is a sign that a first differencing should be included. That the seasonality still is there indicates that a seasonal difference also should be included.

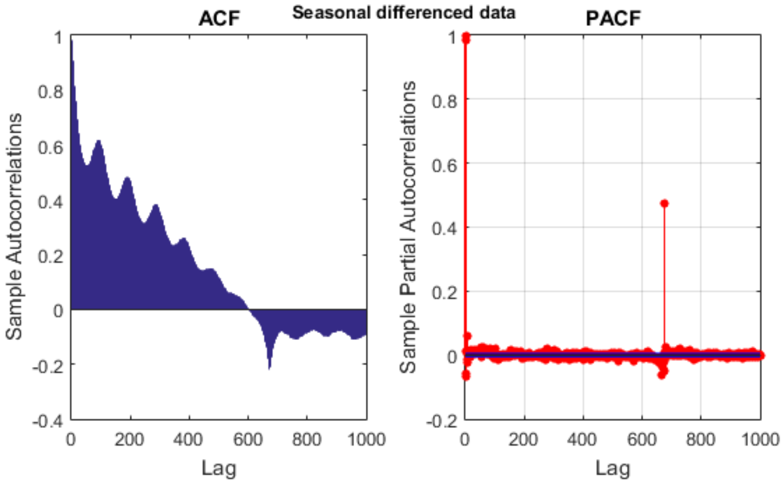

Figure 12: In these plots the data has been modified with a seasonal difference with season 672 which corresponds to a week. That means that $y_t = Y_t - Y_{t-672}$ where y_t is the differentiated data at time t and Y_t is the original data at time t. The ACF show a strong positive auto correlation which could signal for another differentiation.

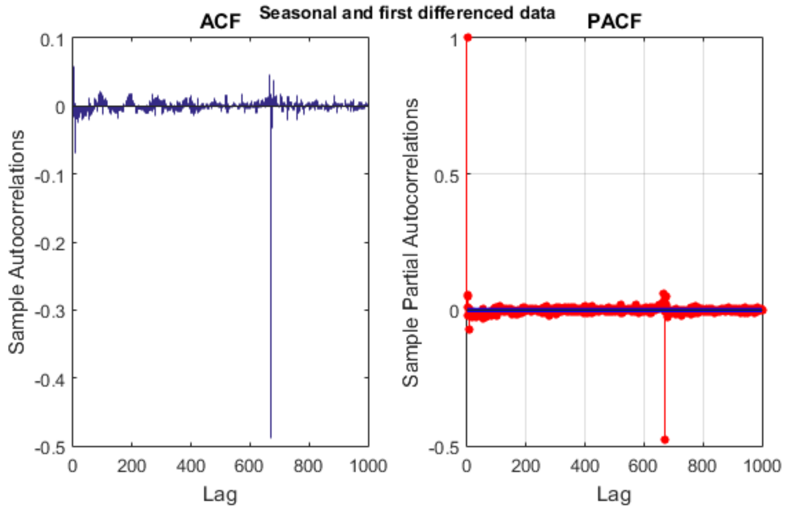

Figure 13: In these plots the data has been modified with both a first difference and a seasonal difference with season 672. The stationarity seems to be gone and most of the seasonality as well except for a very small oscillation. It also shows a clear negative spike at lag 672.

4.4 baseline with physical servers

As briefly mentioned in the introduction, it is about six times cheaper for the company to use their own physical servers than to rent cloud servers. The benefit with cloud servers is that it is a lot faster to set up a new cloud server than to buy a new physical server which means that it has a great potential of decreasing the abundance. This means that to optimize the server cost, some of the capacity should probably be bought as physical servers and some should be bought as cloud servers. How long time it takes to buy and setup a new physical servers differs a lot and depend on several factors but in the following analysis it is assumed that it takes 30 days.

In Figure 15 it is examined where to put the baseline for a cost optimized system. To find each curve in the top plot, i.e. the curve for each window, a cost function was created described with pseudo code in algorithm 1.

Algorithm 1 Baseline cost function

```
for Baseline levels between 0.01-1.0 do
    baseline = level $\cdot$ max(values)
    cloud values = values $>$ baseline
    physical values = values $\leq$ baseline
    cloud cost $= 6 \cdot \sum$ cloud values
    physical cost = baseline $\cdot$ length(physical values)
    total cost = cloud cost + physical cost
end for
```

After the cost of each window had been found, the mean of all these curves was calculated which can be seen in the bottom plot in Figure 15. The minimum of this curve is marked with a red circle at 67% which is the result of the analysis and the recommended level for the baseline assuming that it takes 30 days to rescale with physical servers and that cloud servers are 6 times as expensive.

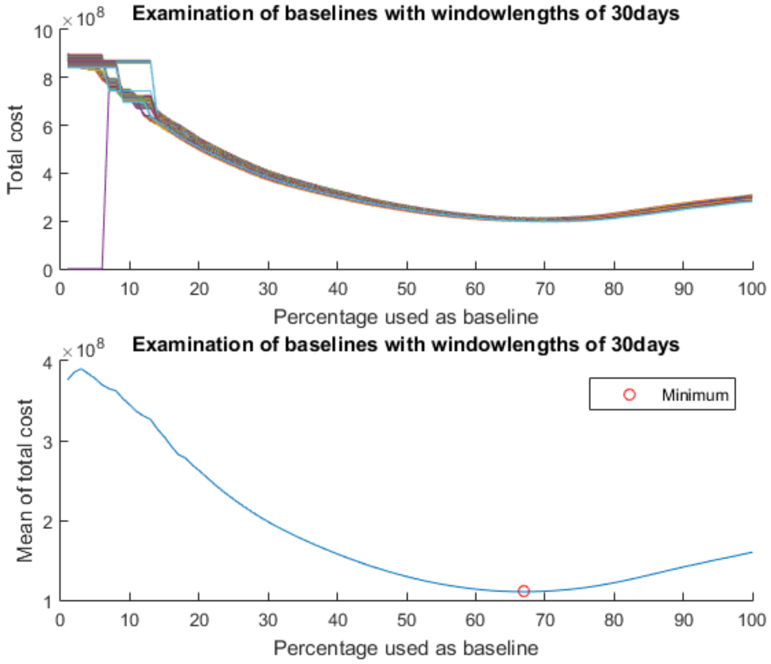

Figure 15: Examination of different percentages on the baseline. In the top plot the cost of different percentages are plotted separately for 2000 different windows. In the bottom plot the mean of all possible windows of with length of 30 days from the dataset is plotted. Notice the red circle which is the minimum of the curve.

5

MODELS

This chapter consists of a short explanation of each model and their forecasting equations. For the models that have parameters with unknown values an explanation of the estimation step is also included.

5.1 STATIC MODEL

The equation for the static model is

$$\hat{y}(t) = \max(y_{training\_set}).$$

The purpose of this model was to give a result against which other models may be compared [9]. What the model does is simply to find the highest peak in the training set and then set the prediction to that value at all future points. This can be looked at as a model of the system without prediction. By comparing other models to this one, a discussion can be made about how beneficial it is to put a prediction model into use.

5.2 LAST VALUE MODEL

The equation for the last value model is

$$\hat{y}(t) = y(t-1).$$

What is does is to look at the last value in the time series and put the next value to the same. It is a very naive and simple model but in this case, where it only is necessary to predict the next value, its prediction is not that bad. In statistical terms this is basically an AR(1) model with its parameter $a_1 = 1$

5.3 LINEAR EXTRAPOLATION MODEL

The equation for the linear extrapolation model is

$$\hat{y}(t) = y(t-1) + (y(t-1) - y(t-2)) = 2 * y(t-1) - y(t-2).$$

What it does is assuming that the difference between the last two sample will not differ too much from the difference between the last and the next sample. The model simply adds the difference between the last two sample to the last sample.

5.4 LAST WEEK MODEL

The equation for the last week model is

$$\hat{y}(t) = \frac{\sum_{k=1}^{n} y(t - k \cdot week)}{n}.$$

The model is based on the observation that the weekly period is strong. What it does is to just look at the value exactly one week back and uses that value for the prediction. To dampen the effect of not found outliers instead of only using values from one week back, the idea of using the mean of values n weeks back was tested. In plot 16 the RMSE, mean shortage and mean abundance is shown for predictions from the model using values $1 - 10$ weeks back. The lowest RMSE occurs with either one or six weeks in memory. Since they are almost the same and since using only one week memory creates a simpler model, one week memory, $n = 1$, was chosen. The strange drop at six weeks is not examined any further for the simple reason that the Last week model will prove to be a pretty useless model.

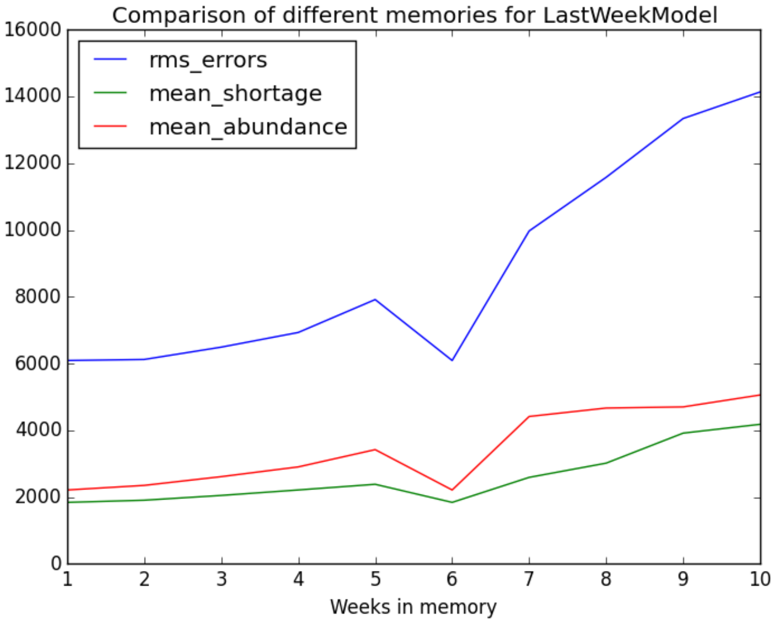

Figure 16: The mean shortage, mean abundance and RMSE of the predictions vs the number of weeks used for last week model. Note that these values will not be the same as in the results due to the fact that in this plot only values from the training set are used. Note the drop at six weeks that is one approximately the same level as one week.

5.5 LAST WEEK MODEL WITH LINEAR EXTRAPOLATION

The equation for the last week model with linear extrapolation is

$$\hat{y}(t) = y(t-1) + \frac{\sum_{k=1}^{n} y(t-k \cdot week) - y(t-k \cdot week - 1)}{n},$$

where n is the number of weeks to take the mean from. What the model does is to add the difference from the samples one week ago to the last sample. To dampen the effect of undetected outliers, instead of only using values from one week back, the model takes the mean difference from n weeks back. To determine what n to use, the model was executed for a few different n:s and then the mean shortage, mean abundance and RMSE was plotted against the number of weeks used in Figure 17. It can be seen that an improvement occurs when looking up to three weeks back but not further which motivated to use $n = 3$.

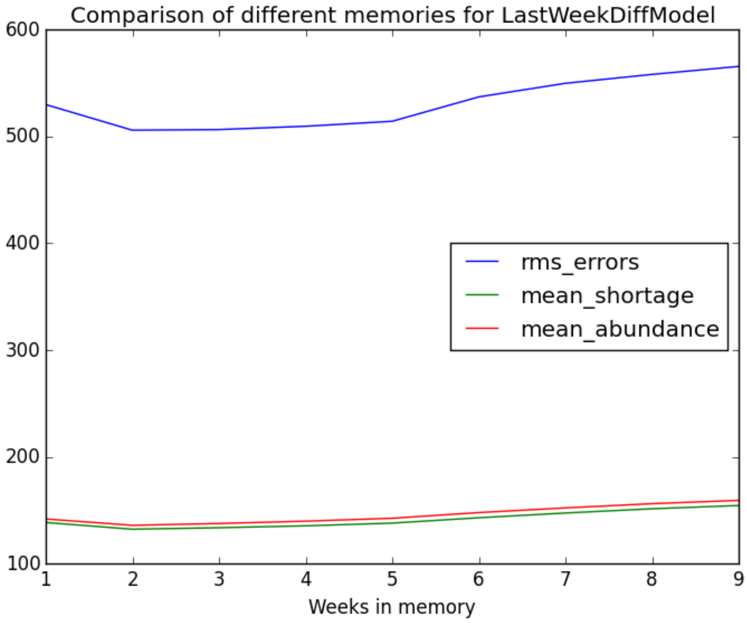

Figure 17: The mean shortage, mean abundance and RMSE of the predictions vs the number of weeks used for last week model with linear extrapolation. The best value seems to be with two or three weeks. Note that these values will not be the same as in the results due to the fact that in this plot only values from the training set are used.

5.6 AR MODEL

The equation for the AR (Auto Regressive) model is

$$y(t) = \mu - a_1 \cdot y(t-1) - a_2 \cdot y(t-2) - \cdots - a_p \cdot y(t-p) + \varepsilon(t),$$

where a_1 to a_p has to be estimated, μ is a constant and $\varepsilon(t)$ is the residual at sample t. This is a standard model in literature when working with periodic time series [17]. To decide what order the model should have, i.e. how long its memory should be, the RMSE, the mean shortage and the mean abundance from the prediction was plotted against the order of the model in Figure 18. It seems that the RMSE keeps decreasing until around order 100 so a model of order 100 could be interesting to evaluate further. In the bottom plot the mean absolute error is plotted separate for residuals < 0 and residuals > 0 or in other words the shortage and the abundance. Since it seems to be a dip in the shortage at order 25 a model of that order could be interesting as well. Since the plots comes from data of the training set only, the values in the evaluation part will not be the same as these.

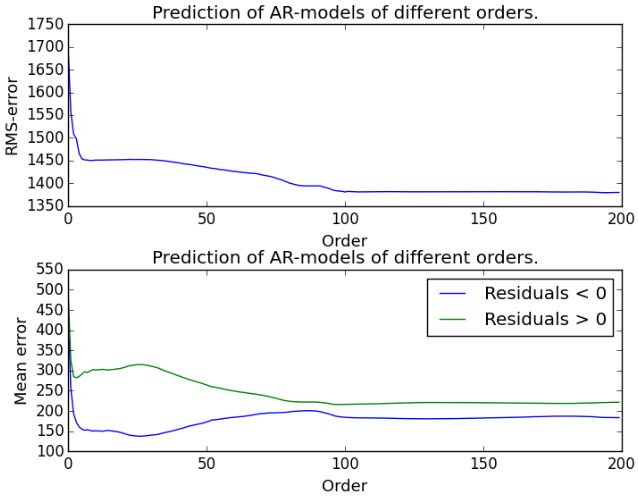

Figure 18: The mean RMSE, the shortage and the abundance of the predictions vs the order of the AR-model. The RMSE decreases until around order 100 and the shortage is the smallest at around order 25.

The parameters for the AR models was estimated with the fit function found in the Statsmodels package in Python. The first parameter in the AR100 model was estimated to $a_1 = 1.10$ while the next 99 parameters was in the range of $0.01 - 0.00001$. Note that the parameters only was estimated once and not before each prediction step.

5.7 AUTOREGRESSIVE INTEGRATED MOVING AVERAGE MODEL

The equation for the AutoRegressive Integrated Moving Average (ARIMA) model is

$$\begin{aligned} y(t) = \mu + \phi_1 y(t-1) + \phi_2 y(t-2) + \cdots + \phi_p y(t-p) \\ -\theta_1 \varepsilon(t-1) - \theta_2 \varepsilon(t-2) - \cdots - \theta_q \varepsilon(t-q), \end{aligned}$$

where ϕ_1 to ϕ_p and θ_1 to θ_q has to be estimated, $\varepsilon(t)$ is the residual at sample t and μ is a constant. The ARIMA model is also a standard model when it comes to periodic time series. It works in the same way as the AR model but adds a smoothing factor, the moving average (MA) part, and a differencing factor, the integration (I) part. A regular $ARIMA(p,d,q)$ process consists of three parts, p, d and q where p is the order of the AR term, d is the number of times to differentiate and q is the order of the MA term. A version of this process is the seasonal ARIMA process (SARIMA) explained in [12] which is suitable in this case since we know that the data has seasonality. It is denoted $SARIMA(p,d,q)x(P,D,Q)s$ where P is the seasonal AR term, D is the seasonal integrated term, Q is the seasonal MA term and s is the season of the time series. To determine the order of all six terms the ACF and PACF plots in section 4.3 will be used. A very easy to follow guide of how this identification works can be read in [19]. The general equation for the seasonal autoregressive integrated moving average model is

$$\Phi_P(B^s)\phi(B)\Delta_s^D\Delta^d y_t = \Theta_Q(B^s)\theta(B)\varepsilon_t$$

where B is the backshift operator and Δ^d and Δ_s^D are ordinary and seasonal difference components. The ordinary and seasonal autoregressive components are represented as $\phi(B)$ and $\Phi_P(B^s)$ and the ordinary and seasonal moving average components are represented as $\theta(B)$ and $\Theta_Q(B^s)$.

In section 4.3, about ACF and PACF, it is established that both a nonseasonal differentiation and a seasonal differentiation with season 672 is necessary to make the process stationary. This corresponds to a $SARIMA(0,1,0)x(0,1,0)_{672}$ where the first parenthesis is the nonseasonal part and the second parenthesis is the seasonal part. As mentioned before there is one negative spike in Figure 13 at lag 672. This indicates that a seasonal MA term should be added. There is also a very small positive spike at around lag 672 which could indicate a seasonal AR term but this is ignored since it is so small and that it would most likely lead to overfitting. In the PACF plot in figure 13 there is a spike at lag 1. According to [19] this indicates a small underdifferencing and that one or several AR terms should be added. The lag beyond which the PACF cuts off, i.e. lag two, is the indicated number of AR terms. This gives our model its final form $SARIMA(2,1,0)x(0,1,1)_{672}$. Its equation is

$$\phi(B)\Delta_{672}\Delta y_t = \Theta_1(B^{672})\varepsilon_t$$

which also can be written as

$$(1-\phi_1 B-\phi_2 B^2)(1-B^{672})(1-B)y_t = (1+\Theta B^{672})\varepsilon_t$$

or when in its forecasting form like

$$\begin{aligned}\hat{y}_t =& y_{t-1}(1+\phi_1) - y_{t-2}(\phi_1-\phi_2) - y_{t-3}\phi_2 + y_{t-672} - \\ & y_{t-673}(1+\phi_1) - y_{t-674}(-\phi_1+\phi_2) + y_{t-675}\phi_2 + \\ & \varepsilon_t + \Theta\varepsilon_{t-672}.\end{aligned}$$

The coefficients ϕ_1, ϕ_2 and Θ was estimated, by using Matlab's estimate function on the model and the training set, to

$$\begin{aligned}\phi_1 &= 0.0766545,\\ \phi_2 &= 0.0856706,\\ \Theta &= -0.867538.\end{aligned}$$

5.8 DECOMPOSITION MODEL

The decomposition model thinks of the data as three different components. A seasonal component which is the daily and weekly change in the data, a trend component which is a long-term increase of decrease of the data and an error component which is everything that is not explained by the other components [18]. Sometimes a fourth component is also considered, the cyclic component which is when the data increases and decreases but not over a fixed period. In our model though, the cyclic pattern is included in the trend component. The decomposition of the dataset is shown in Figure 19.

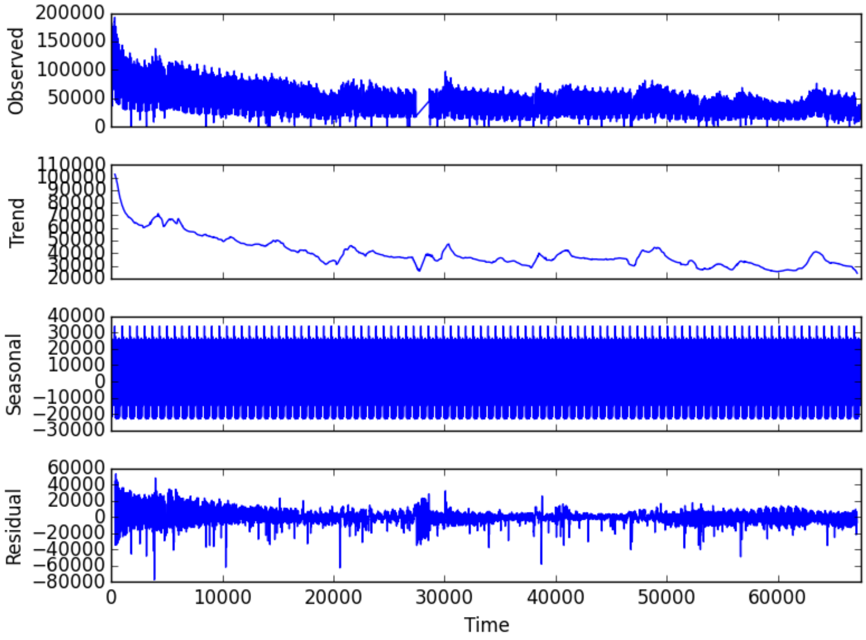

Figure 19: Decomposition of the dataset done with the Python package Pandas.

Two different decomposition models can often be found in literature, the additive model

$$y_t = S_t + T_t + E_t,$$

and the multiplicative model

$$y_t = S_t \cdot T_t \cdot E_t,$$

where Y_t is the signal, S_t is the seasonal component, T_t is the trend component and E_t is the error component. The additive model is suitable if the magnitude of the seasonal fluctuations does not vary with the level of the data [18].

In Figure 20 it can be seen that when the mean of the data increases the magnitude of the fluctuation does the same. That means that an additive model is not suitable and a multiplicative model was chosen instead.

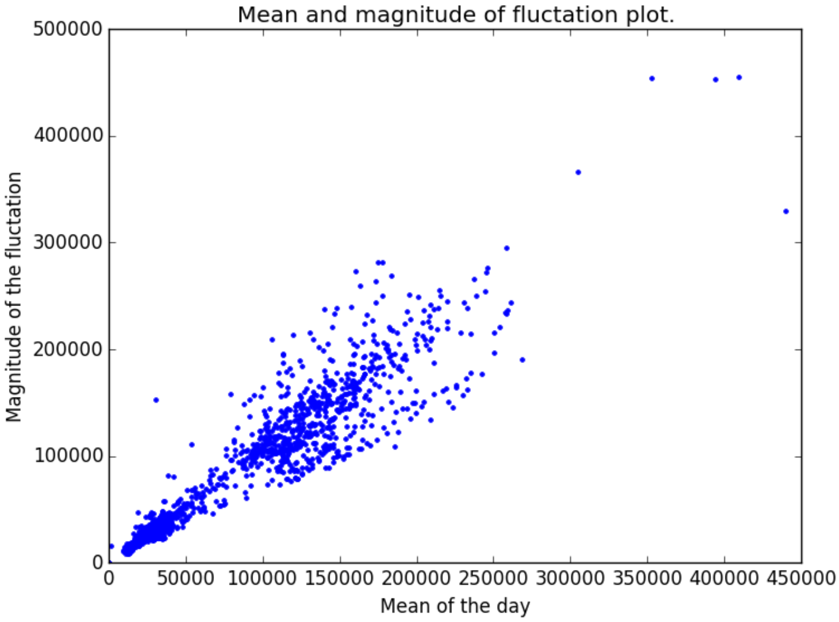

Figure 20: The dataset is divided in daily periods and then the mean of each day is plotted against the magnitude of that day's fluctuation.

6

RESULTS

This chapter will start with the main results which is an evaluation on which model that performs best and how well it performs. After that follows three shorter parts. The first one evaluates whether or not it was a good choice to separate the data per console. The second one evaluates how big impact the outliers and missing values would have on the result if they were not handled. The last part consists of a short examination of how well some of the models would perform on further look-ahead prediction.

6.1 MAIN RESULTS

The models will first be evaluated from a mathematician's perspective. According to section 2.2.1 RMSE, MAPE and MdAPE will be used as performance metrics. The results with these metrics from all the models is shown in Table 2. The model "Last week extrapol opt", which is short for the last week model with linear extrapolation with memory length $n = 3$, produces the best values in MAPE and MdAPE while "Last value extrapol" is best when it comes to RMSE.

From the company's perspective the cost function in section 2.2.2 is the most interesting value and therefore the mean residual and the lower bound confidence interval with confidence level 99.97% is presented in Table 3. The model AR25 has the best confidence interval but its residuals are not centered around zero which instead gives the AR100 model the best cost function value at 689 players. The "Last value extrapol" model has the second best cost function value at 735 which is not far behind. As can be seen in Figure 21 the prediction from the AR100 model is very close to the true values.

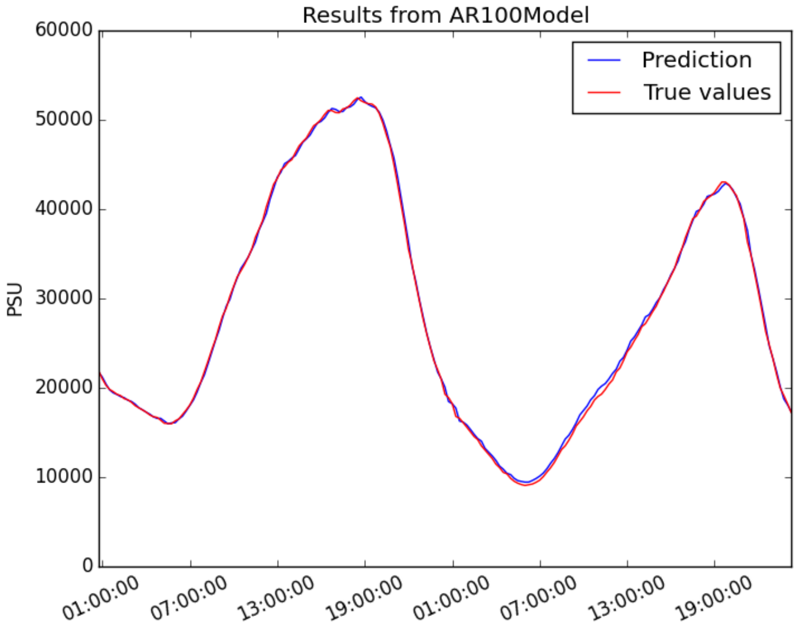

Figure 21: Predictions from the AR100 model together with the true values. Note that it is almost impossible to distinguish the two lines.

The lower bound confidence interval (-624) of the AR100 model tells us that there is a 99.97% probability that the next predicted value will not be further below the true value than 624 players. From the company's perspective this means that if they make sure to have server capacity for 624 players more than the predicted value then its expected that 99.97% of the time, the server capacity will be sufficient. This corresponds to that it is expected there will be 10 samples each year where the server capacity is too low. The word expected means that it is 50% probability that the server capacity will be sufficient more often than 99.97% of the time and 50% probability that the server capacity will be sufficient less than 99.97% of the time.

So by having a mean abundance in the server capacity of 689 players the company could expect shortages only 10 samples per year. Since cloud capacity is approximately $6x$ more expensive than physical capacity this means that only if the company could scale their physical servers with such precision that their mean abundance would get less than $6 \cdot 689 = 4134$ players, it would be beneficial to stay at physical servers. *If their mean abundance with physical servers is greater than* 4134 *they should switch to a system with cloud servers.*

One way to visualize the results and the spread of the residuals better is by using a boxplot which can be seen in Figure 22. The whiskers corresponds to percentiles [1, 99]. The red line in the middle is the mean and the box in the middle contains 50% of the residuals. To see things clearer all values outside the percentiles [1, 99] is ignored in Figure 23. It can be seen that most errors are somewhere in the range of −500 to 600.

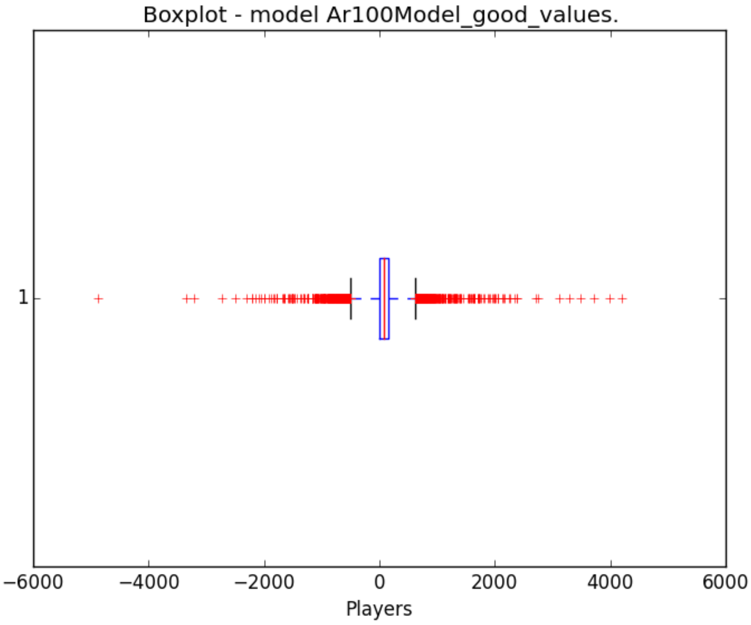

Figure 22: Boxplot of the residuals from model Ar100 with good values. Whiskers are at percentiles [1, 99]. Each red cross marks a value.

Table 2: Evaluation of models on good values with data from BF3 and PC.

Model	RMSE	MAPE	MdAPE
Static	151000	2110	1630
Last value	498	3.11	2.69
Last value extrapol	**214**	1.42	1.06
Last week	7280	10.0	5.4
Last week extrapol	7280	10.0	5.49
Last week extrapol opt	269	**1.26**	**0.934**
AR1	497	3.12	2.70
AR25	463	5.81	4.55
AR100	217	1.58	1.10
SARIMA	269	1.37	1.02
Decomposition	11124	109	83.4

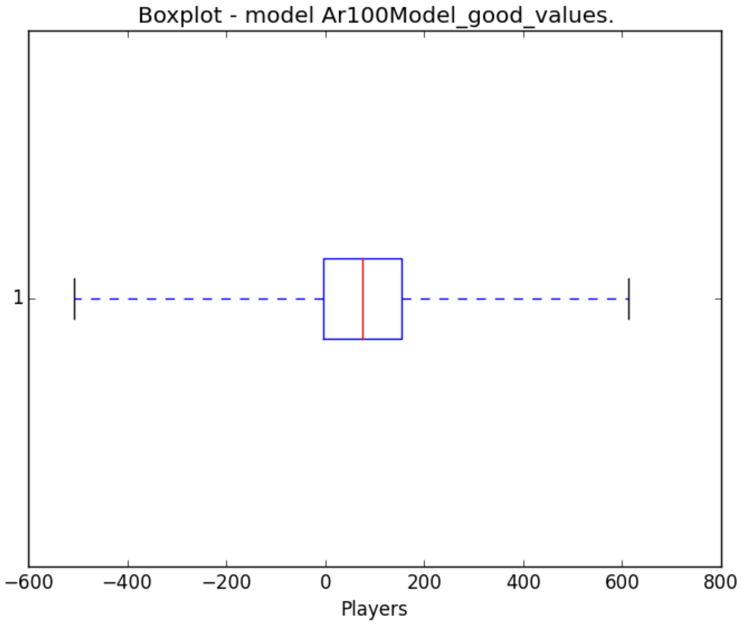

Figure 23: Boxplot of the residuals from model Ar100 with good values. Whiskers are at percentiles [1, 99].

Table 3: Evaluation of models on good values with data from BF3 and PC.

Model	Lower bound conf interval	Mean residual	Cost function
Static	(-112000)	150000	370000
Last value	(-1690)	14.9	1710
Last value extrapol	(-733)	2.59	735
Last week	(-25000)	-24.6	25000
Last week extrapol	(-1000)	-24.6	1000
Last week extrapol opt	(-922)	2.72	924
AR1	(-1700)	9.95	1707
AR25	**(−458)**	391	849
AR100	(-624)	74	**689**
SARIMA	(-924)	0.177	924
Decomposition	(-34300)	2735	37000

In the evaluation of the outliers in Table 4, the model "Last value extrapol" again has the lowest RMSE (958) while the AR100 model has the lowest MdAPE (0.860). Since the outliers sometimes has zero as the true value, the MAPE gets infinity on all models.

In Table 5 it can be seen that the model "Last value extrapol" is best from the company's perspective with a cost function value of 3290.

A visualization of how well the model "Last value extrapol" performs on outliers can be seen in Figure 24. The outliers are the extra jagged parts with the biggest one between the 8th and 9th of July.

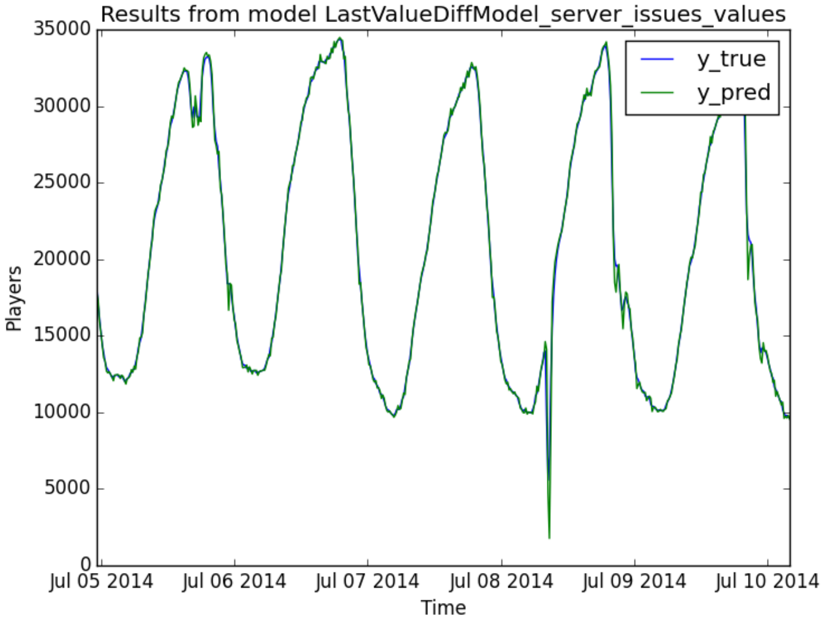

Figure 24: Prediction and signal from model Last Value with Linear Extrapolation showing a section with many outliers.

Table 4: Evaluation of models on outliers with data from BF3 and PC.

Model	RMSE	MAPE	MdAPE
Static	146000	inf	995
Last value	1100	inf	2.40
Last value extrapol	**958**	inf	0.939
Last week	4410	inf	6.48
Last week extrapol	1170	inf	0.971
Last week extrapol opt	1100	inf	1.04
AR1	1100	inf	2.41
AR25	1080	inf	2.14
AR100	990	inf	**0.860**
SARIMA	963	inf	0.903
Decomposition	11300	inf	31.8

Table 5: Evaluation of models on outliers with data from BF3 and PC.

Model	Lower bound conf interval	Mean residual	Cost function
Static	(115097)	146000	31000
Last value	(-3750)	13.5	3760
Last value extrapol	(-3290)	-3.24	**3290**
Last week	(-12000)	1800	13800
Last week extrapol	(-4000)	10.4	4010
Last week extrapol opt	(-3770)	10.5	3780
AR1	(-3760)	6.91	3760
AR25	$(-\mathbf{3230})$	310	3540
AR100	(-3310)	73.5	3390
SARIMA	(-3280)	24.4	3310
Decomposition	(-31000)	4570	35600

6.2 separating the data per console

The idea of separating the data per console and then combine the predictions instead of using the data from all consoles at once was tested. A comparison was made for the model with the best cost function, the AR100 model. The model was first used directly on the combined players for BF3 from all consoles which produced a cost function value of 1170. Then the model was used on the different consoles separately and then added together. This produced a cost function value of 1280 which is worse than the previous one. If a prediction would be made on a game over all consoles it seems that it should not first be separated per console.

6.3 impact of outliers

In Table 6 all values have been used and it can be seen that it is the same models as for the good values that have the best results. The cost function value for the AR100 model for the good values was 689 while in this evaluation it is 1620 which is more than double. On most of the models, with exception of the worst ones, the cost function value from this evaluation was approximately $500 - 1000$ higher than from the evaluation without outliers.

Table 6: Evaluation of models on all values with data from BF3 and PC.

Model	Lower bound conf interval	Mean residual	Cost function
Static	(114000)	150000	35800
Last value	(-2240)	0.145	2240
Last value extrapol	(-1670)	3.84	1680
Last week	(-22600)	187	22800
Last week extrapol	(-2290)	-0.106	2290
Last week extrapol opt	(-1700)	0.0373	1700
AR1	(-2240)	- 4.84	2240
AR25	**(−1360)**	377	1740
AR100	(-1540)	74	**1620**
SARIMA	(-1720)	-0.0908	1720
Decomposition	(-28800)	3230	32000

6.4 further look-ahead predictions

It would have been interesting to compare all models with further look-ahead prediction but there was not enough time to do a full out comparison. A one day look-ahead prediction was made with three of the models; The AR100 model, the Last Value with linear extrapolation model and the SARIMA model. The reason for choosing the first two were that they produced the best result with a next sample prediction. The SARIMA was chosen because it had the most statistical theory behind it. The predictions together with the true values can be seen in Figure 25, 26 and 27. The results from all models are of course a lot worse than with the 15 minutes look-ahead predictions but at least the models have captured the general shape of the data. The SARIMA is clearly the winner in this category which can be seen only by looking at the figures. Even its cost function value, in Table 7, at 7170 players, which is approximately 8 times worse than what the same model produced when only predicting the next sample, is a lot better than the other two.

Table 7: Evaluation of models on all values with data from BF3 and PC with one day ahead prediction.

Model	Lower bound conf interval	Mean residual	Cost function
Last value extrapol	(-10500)	-63.7	10400
AR100	(-18700)	15400	34000
SARIMA	(-7010)	165	**7170**

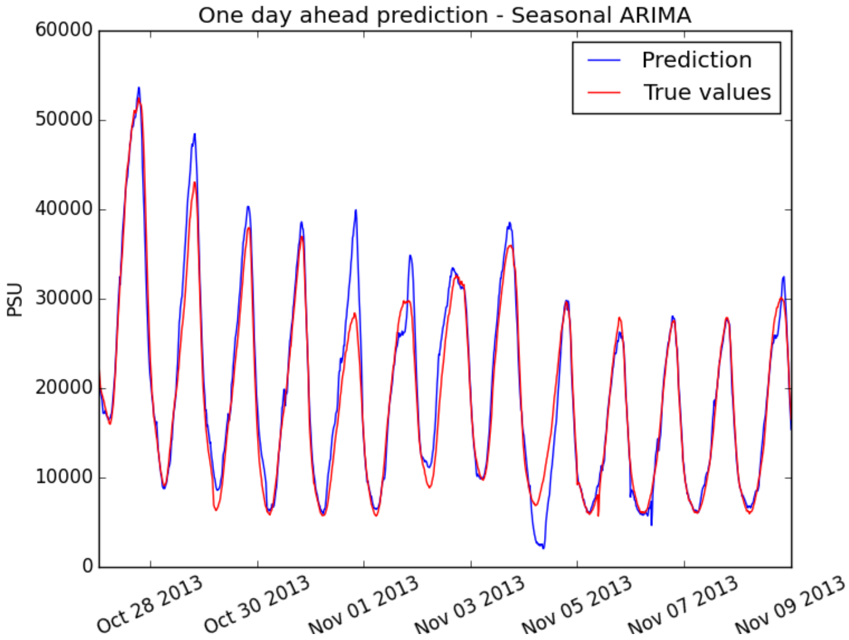

Figure 25: Prediction and signal from model SARIMA with one day look-ahead predictions. Note that the prediction is much worse than with 15 minutes look-ahead prediction.

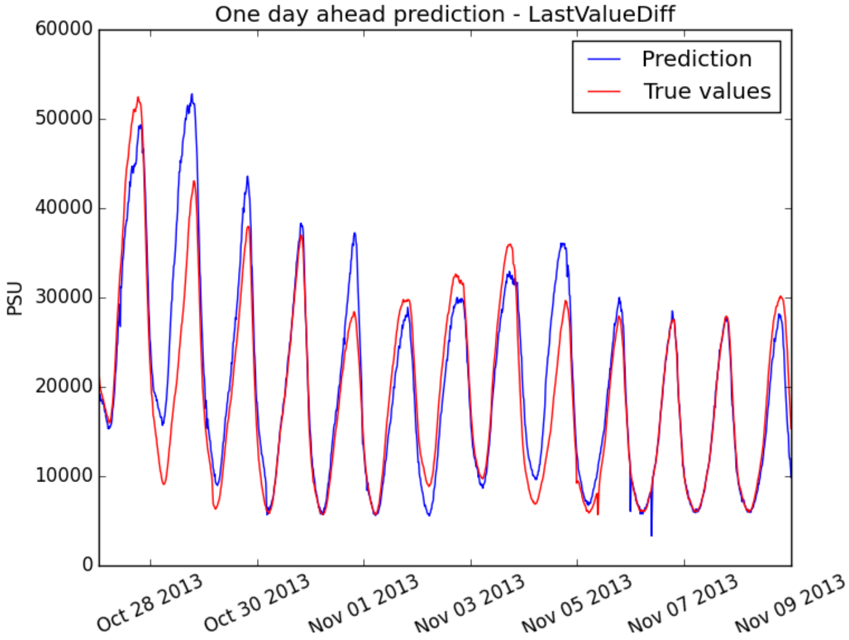

Figure 27: Prediction and signal from model AR100 with one day look-ahead predictions. Note that the prediction is much worse than with 15 minutes look-ahead prediction.

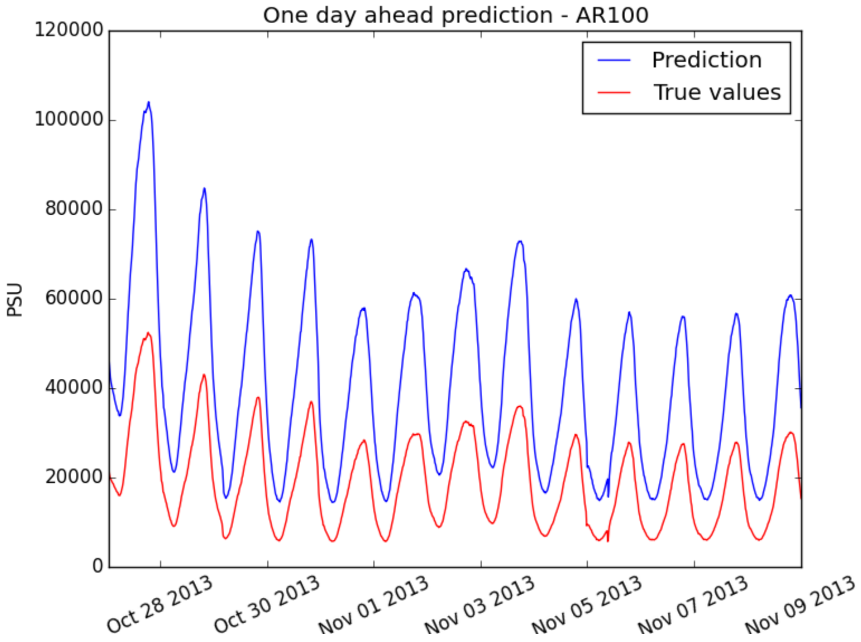

Figure 26: Prediction and signal from model AR100 with one day look-ahead predictions. Note that the prediction is much worse than with 15 minutes look-ahead prediction and also much worse than the prediction from the SARIMA in figure 25.

7

DISCUSSION AND FUTURE WORK

Now that the best model for good values and the best model for outliers has been found, these models could be combined. One model could be used as long as it only depends on good values and as soon as it starts depending on values with server issues then the second model could take over. The problem is that it is not known how good it actually would perform since the current evaluation techniques is done on all outliers and not only on values that comes right after a server issue. It does not yet exists any method for separating outliers into values with server issues and values that follow on server issues. If data about when it has been server issues would become available, then such a method would be easy to implement and a combination of a model for good values and one for outliers would make sense.

Even though the algorithm for detecting outliers almost found all manually flagged values, it probably missed several since the manually flagging was not perfect. By looking at the worst predictions from the models several probable outliers was found and if these were removed from the evaluation the results would probably be even better. By using the models in a real system all flags that comes from server issues would be accessible and the algorithm for detecting outliers could probably be improved a lot. With a new algorithm the evaluation of the predictions should be done anew since it could produce another winner among the models.

Another thing that might improve the results for some of the models with an estimation step, is to perform the estimation step at each sample. In this case where samples from more than a year should be predicted at once, an estimation step before each prediction would have been too time consuming.

As mentioned earlier samples with 15 minute intervals has been used in this work while for DICE it would probably only take around 5 minutes to rescale their systems. By shorten the interval to 5 minutes all results would most likely improve a lot.

Another subject for future work is to implement a system that is able to use the models on a continuous data flow and that updates the evaluation values for the model after each prediction. Then the confidence interval will always be up-to-date and able to tell how much extra cloud servers that has to be bought.

Instead of using methods from time series analysis it could be interesting to try machine learning tools instead. This is particularly relevant because both Google and Microsoft released their machine learning tools as open source during the autumn.

8

CONCLUSION

An auto regressive model of order 100 was the model that performed the best and could be a suitable choice. It should be confirmed on the dataset at hand if the order 100 still is the best choice. The model "Last value extrapol" almost performed as good as the winner and in regards to outliers it got the best results. It could also be a suitable choice if the company values to have a really easy to understand model. The results from other datasets (not presented in this thesis) were similar with either an AR model or the "Last value extrapol" model as the winner.

When the prediction horizon is as short as 15 minutes it seems that it is sufficient with really simple statistical models. If one were to predict further ahead than the next sample, it appears like more advanced models becomes necessary.

To be as cost optimized as possible the company should try to use physical servers for 67% of the max capacity assuming that it takes them 30 days to rescale the physical servers and that cloud servers are 6 times as expensive as physical servers. Then of course the task of predicting what the max capacity will be remains, but that is outside the scope of this thesis.

It was also stated that if the company has a mean abundance higher than 4134 players using a system with only physical servers, it will be beneficial to switch to a pure cloud server based system. By instead of using only cloud servers, they put up a baseline with physical servers at 67% of the predicted maximum players, the costs will drop even more.

BIBLIOGRAPHY

[1] Nae, Vlad, Radu Prodan, and Thomas Fahringer. *"Cost-efficient hosting and load balancing of massively multiplayer online games."* Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on. IEEE, 2010.

[2] Buyya, Rajkumar, Rajiv Ranjan, and Rodrigo N. Calheiros. *"Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services."* Algorithms and architectures for parallel processing. Springer Berlin Heidelberg, 2010. 13-31.

[3] Caron, Eddy, Frederic Desprez, and Adrian Muresan. *"Forecasting for grid and cloud computing on-demand resources based on pattern matching."* Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010.

[4] Nilabja Roy, Abhishek Dubey and Aniruddha Gokhale. *"Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting."* 2011 IEEE 4th International Conference on Cloud Computing.

[5] Vlad Nae, Alexandru Iosup, Radu Prodan. *"Dynamic Resource Provisioning in Massively Multiplayer Online Games."* Transactions on Parallel and Distributed Systems, 2008.

[6] Acock, A. C. (2005), *"Working with missing values."* Journal of Marriage and Family, 67: 1012–1028. doi: 10.1111/j.1741-3737.2005.00191.x

[7] Burke, Shaun. *"Missing values, outliers, robust statistics & non-parametric methods."* LC-GC Europe Online Supplement, Statistics & Data Analysis 2.0 (2001): 19-24.

[8] Jerome T. Connor, R. Douglas Martin, Member, IEEE, and L. E. Atlas, Member IEEE. *"Recurrent Neural Networks and Robust Time Series Prediction."* IEEE Transactions on neural networks, vol. 5, no. 2, march 1994.

[9] J Kupferman, J Silverman, P Jara, J Browne. *"Scaling Into The Cloud."* Technical report, University of California, Santa Barbara; CS270 - Advanced Operating Systems, 2009. URL http://cs.ucsb.edu/ jkupferman/docs/ScalingIntoTheClouds.pdf.

[10] Akindele A. Bankole and Samuel A. Ajila. *Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment.* The 7th IEEE International Symposium on Service-Oriented System Engineering (IEEESOSE 2013), San Francisco Bay, USA March 25 – 28, 2013.

[11] Ajila SA, Bankole AA. *Cloud client prediction models using machine learning techniques.* 37th IEEE Annual Computer Software and Applications Conference (COMPSAC), IEEE Computer Society, 2013; 134–142.

[12] Xinghua Chang, Meng Gao, Yan Wang and Xiyong Hou. *Seasonal autoregressive integrated moving average model for precipitation time series.* Journal of Mathematics and Statistics 8 (4): 500-505, 2012.

[13] M.V. Shcherbakov, A. Brebels, N.L. Shcherbakova, A.P. Tyukov, T.A. Janovsky, V.A. Kamaev (2013) *A Survey of Forecast Error Measures.* World Appl. Sci. J. 24 (Information Technologies in Modern Industry, Education & Society), 171–176.

[14] Precision and Recall. (2016, January 12). In *Wikipedia, The Free Encyclopedia.* Retrieved 10:53, january 12, 2016, from `https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=698378999`.

[15] Receiver operating characteristic. (2016, January 12). In *Wikipedia, The Free Encyclopedia.* Retrieved 09:56, January 12, 2016, from `https://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=698253308`.

[16] Battlefield 4. (2016, January 12). In *VGChartz.* Retrieved 10:15, january 12, 2016 from `http://www.vgchartz.com/gamedb/?name=battlefield+4`.

[17] Lennart Olbjer, Ulla Holst, Jan Holst. *Tidsserieanalys.* Lunds Universitet och Lunds Tekniska Högskola 1994.

[18] Time series decomposition. (2016, January). In *Texts online, open-access textbooks.* Retrieved january 2016. `https://www.otexts.org/fpp/6`

[19] ARIMA models for time series forecasting. (2016, january). In *Robert Nau, Fuqua School of Business Duke University.* Retrieved january 2016. `http://people.duke.edu/ rnau/411home.htm`

Master's Theses in Mathematical Sciences 2016:E3
ISSN 1404-6342

LUTFMA-3287-2016

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/