Sport Ratings

Honours project

Authors:
Joost Hoppenbrouwer
Marysia Winkels

Supervisor:
Christian Schaffner

University of Amsterdam

Contents

1 Introduction

In our current world, sport is one of the biggest businesses. In soccer, for example, both the players and the coaches are earning millions of euros per year, clubs are willing to spend millions at transfer periods and a single win or loss can mean a huge profit or a loss for the club. With these interests, would it not be great if the 'best' team would always win? The problem here is that a single game (in any sport) is influenced by a lot of factors (such as condition of the players involved, or the mistakes made by the referee(s)). This makes predicting the outcomes of sport games almost impossible, but there do nevertheless exist algorithms that attempt to do exactly that.

During this project, it is our goal to determine what the advantages and disadvantages are of these so-called *sport rating systems*. Wikipedia (2013) describes sport rating systems as follows:
"A sports rating system is a system that analyzes the results of sports competitions to provide objective ratings for each team or player. Rankings are then derived by sorting each team's ratings and assigning an ordinal rank to each team starting with the highest rated team earning the #1 rank."

Sport rating systems are mathematical models that provide *ratings* to teams that indicate their strength relative to each other. Ratings are different from *rankings*, as described by Langville and Meyer (2012)[1]:

- "A *ranking* of items is a rank-ordered list of the items. Thus, a ranking vector is a permutation of the integers 1 through n."
- "A *rating* of items assigns a numerical score to each item. A rating list, when sorted, creates a ranking list."

During this project we took a look at tournaments and leagues that aren't of the round-robin[2] type, while we are convinced that round-robin formatted tournaments and competitions are 'fair' given a point-based[3] system. With fair we mean that the final ranking of the teams gives a good reflection on the strength of the teams. This, we think that the #1 ranked team is the team with the highest rating.

Our second goal is to design some graphical representation which can be used

[1]For this project, we will use the definitions for rating and ranking given by Langville and Meyer (2012).

[2]`http://en.wikipedia.org/wiki/Round-robin_tournament`

[3]A point-based system is a system where a team is awarded with an amount of points dependent on a win, loss or draw.

to explain to participants of tournaments why their ratings are they way they are. Several sport rating systems are used today, and during tournaments and leagues, these systems determine the ratings for each team. The problem is that the rating system is not always simple to understand. While experts will understand the ratings, most times players will not. For instance, a player might ask why a team that has lost a game has a higher rating and is ranked above a team that has won all their games.[4] The goal of the graphical representation will be to show a team/player *why* their rating is what it is.

2 Algorithms

We chose to investigate, implement and experiment with five rating systems; least squares, Maximum Likelihood, Keener, Elo and Elo scorebased. The first steps toward using these algorithms to experiment with, is to fully understand the mathematical workings of these rating systems in order to be able to implement them. We chose to implement these algorithms in Matlab[5], a numerical computing environment suitable for matrix manipulations. The rating systems were all built in such a way that they could receive the same matrices as input, to allow the user to use the same representation of the sport data for all algorithms.

In order to understand the mathematical workings of the algorithms, a few sources provided a basis for our mathematical understanding. Massey (1997) for the explanation of the least squares algorithm, and chapters 4 and 5 from Langville and Meyer (2012) for Elo and Keener's method. For our understanding of maximum likelihood, we mostly relied on three blog articles written by user "Doug" on site `www.pro-football-reference.com`; Doug (2006a), Doug (2006b) and Doug (2006c).

2.1 least squares

2.1.1 Mathematical Background

The first rating system that provides ratings for each team in a competition is the *least squares* method. The main assumption that forms the basis of the least square rating system is that the difference in scores between the

[4]The first team most likely had stronger opponents.

[5]`http://www.mathworks.com`

winning team and the losing team is directly proportional to the difference in the ratings of the two teams. Therefore, the least squares rating system attempts to express the margin of victory as a linear function of the strengths of the playing teams.

The least squares method provides us with a vector containing the ratings of each team, by solving the equation $Xr = y$ for r. In this equation, X is a matrix that contains the played games and the wins and losses within that game. For example, a row of X might look like $\begin{bmatrix}1 & 0 & -1 & 0\end{bmatrix}$, which indicates that team one and team three have played a game against each other, which team one has won and team three has lost. The value at the index of the teams that have not played is zero, indicating they were not involved in the game. The value of the teams that have played are is either a one (indicating that they have won) or a minus one (indicating that that team has lost). The matrix X contains a row for each game that has been played in the competition. Vector y has as many rows as matrix X does, as vector y stores information about the score differences in the tournament. This means that a 5×4 matrix X contains information about five games played between four teams. The associated vector y will, as it contains information on the scores of the played games, therefore also have five rows. The vector r in the formula is a vector with as many values as there are teams in the competition. As matrix X and vector y are known after the games have been played, one would ideally like to find values for r in such a way that matrix X multiplied by vector r results in the differences in scores that have been recorded.

The vector of ratings will be a solution to the normal equations $X^T Xr = X^T y$. As the equations are dependent, the last row of the resulting matrix of $X^T X$ can be replaced entirely by a row containing the value one at each index to introduce scale. Additionally, the corresponding value in the vector that resulted from the multiplication of X^T and y should be replaced with a zero. Replacing this information will cause no loss in information about the games that have been played, as the information can be found elsewhere in the $X^T X$ matrix, but does allow the results to be scaled. Vector r is found by solving $X^T Xr = X^T y$.

2.1.2 Implementation

Our implementation of this rating system was pretty straight-forward. Due to the nature of the model, it is possible to find the normal equations without constructing the matrix X. However, as most of the data we have used was

small enough to be within the computational capabilities of our system, we have decided not to alter the system in such a way that the construction of matrix X would not be necessary.

Although this information is not particularly necessary for the least squares algorithm, we have built the system in such a way that it can have a matrix s containing the actual scores scored in the games rather than the score differences.

2.2 Maximum Likelihood

2.2.1 Mathematical Background

Another possible algorithm that can provide ratings for different teams in a competition is called *maximum likelihood*. Using maximum likelihood, one would like to maximize the ratings in such a way that the probability of what actually happened in the competition is the highest of all possible outcomes of the tournament. The probability of the competition's result (the different games ending the way they did) is the cumulative product of the probabilities of the individual games happening the way they did. For example, if in a competition team i beat team j and team k beat team l, the probability that should be maximized is the product of the probability that team i beat team j and the probability that team k beat team l. For this rating system, we are to assume that the probability of team i beating team j can be seen as the outcome of $\frac{r_i}{r_i+r_j}$, meaning the rating of the winning team i divided by the sum of the rating of team i and team j.

If there are n teams that have played in the competition, the function P is a function of n variables. When one attempts to maximize such a function, one way to do so is to set all the partial derivatives with respect to these variables to zero and then to solve that system of n equations. However, as these equations contain products of quotients, the derivatives would be relatively difficult to determine by hand. Instead, in order to reduce the computational complexity, one could work with the natural log instead, as the derivative of P would have the same sign as the derivative of the natural log of P and they would therefore be maximized at the same place. As we have set these n equations to zero, this enables us to determine the individual variables (or team ratings). The easiest way to do this is through iteration. First, set all the team ratings at a certain value, for example one. Using these set values as initial ratings, one can calculate what the new rating would be using this

value. Then, one should continue to calculate the new ratings based on the old ratings, until the numbers associated with the variables stop changing. As these numbers have stopped changing, one has the numbers associated with each team that solves the n equations and these numbers are the ratings for the respective teams.

For example, if one has the equation $P = \frac{i}{i+j} * \frac{j}{j+k} * \frac{k}{i+k} * \frac{i}{i+k}$, this indicates that during the tournament, four games have been played: one in which team i beat team j, one in which team j beat team k, one in which team k beat team i and one of which in which team i beat team k. As it is computationally easier to use natural logs, rather than the products of quotients, the equation becomes:
$\ln(P) = \ln(i) - \ln(i+j) + \ln(j) - \ln(j+k) + \ln(k) - \ln(i+k) + \ln(i) - \ln(i+k)$
or
$\ln(P) = 2\ln(i) + \ln(j) + \ln(k) - (\ln(i+j) + \ln(j+k) + 2\ln(i+c))$.
The partial derivative of this equation with respect to team i is
$\frac{d(\ln(P))}{di} = \frac{2}{i} - \frac{1}{i+j} + \frac{2}{i+k}$.
As we want to set this partial derivative to zero, this is
$0 = \frac{2}{i} - \frac{1}{i+j} + \frac{2}{i+k}$ or $i = \frac{2}{\frac{1}{i+j}+\frac{2}{i+k}}$.

Setting the values of the ratings of team i, j and k all to one would mean we are able to calculate a new value of the rating of team i by filling in all the numbers: $i = \frac{2}{\frac{1}{1+1}+\frac{2}{1+1}} = \frac{2}{\frac{1}{2}+1} = \frac{4}{3}$.
One has to do this for the partial derivatives of the equation with respect to j and k as well and update the values of the ratings of all the new ratings have been calculated (in this case, for team j the new value is 1 and for team k, the new value is $\frac{2}{3}$). Imagine a competition with A-Z teams. The general pattern of the formula to determine the rating of a particular team is as follows: Team A $= \frac{\text{Team A's wins}}{\frac{\text{Games against Team B}}{\text{Rating A + Rating B}} + \ldots + \frac{\text{Games against Z}}{\text{Rating A + Rating Z}}}$

However, as maximum likelihood initially does not take the margin of victory into account, the resulting ratings and therefore the probability a team i might defeat a team j might be higher or lower than an expert would intuitively suggest. Additionally, using maximum likelihood in the way previously described means that certain teams can, under certain conditions, get infinite ratings. Especially for sports in which not all teams play against each other, such as in American college football, it might be a good idea to adjust the current use of the maximum likelihood rating system in such a way that the strength of the schedule is taken into account. By treating a win as a 99% win and a 1% loss (suppose that if team i beat team j, one treats this

as that out of a hundred games played, team i won 99 of these games and team j won a single one). This fixes the problem the previously mentioned problem of the maximum likelihood rating system by ensuring that teams can no longer get an infinite rating (if they have never lost). Additionally, one could tune the system by altering how a won game is perceived. The higher the percentage won by the winning game is (the previously mentioned 99%), the more the ratings depends on the wins and losses during the competition. The lower this number is, the more the rating is dependent on the strength of the schedule.

2.2.2 Implementation

However, sadly enough we were not able to get our version of the maximum likelihood rating system working in such a way that it could be used. We encountered difficulties in determining the different partial derivatives needed. We were able to make the algorithm work if the competition had a set number of teams, but unfortunately, this was not enough to work with and we therefore had to renounce this implementation entirely.

2.3 Keener

2.3.1 Mathematical Background

In 1993, James P. Keener developed and proposed a new method for calculating sports ratings, called Keener's method. The method attempt to provide teams with a measure of their *absolute strength*, which is dependent on their *relative strength*. Langville and Meyer (2012) state:

- "The *strength* of a team should be gauged by its interactions with opponents together with the strength of these opponents."
- "The *rating* of each team in a given league should be uniformly proportional to the strength of the team. Strength $s = \lambda\times$ rating r with the same λ for every team."

The strength of a team i relative to the strength of team j is expressed as $s_{ij} = a_{ij}r_j$, where a_{ij} is the measure of score difference. This is what we call the *relative strength*. The *absolute strength* of a team i is the sum of all its relative strengths. The strength vector, containing the absolute strengths of each team, can be calculated as follows:

$$s = Ar = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1m} \\ a_{21} & a_{22} & ... & a_{2m} \\ ... & ... & ... & ... \\ a_{m1} & a_{m2} & ... & a_{mm} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ ... \\ r_m \end{bmatrix}$$

After a tournament has been played, matrix A containing a_{ij} for all teams i and j is known. Both the rating vector r as well as the vector containing the absolute strengths s at that point unknown. However, as $s = Ar$, it is also true that $s = \lambda r$ and therefore that $Ar = \lambda r$. This equation can be solved, according to Langville and Meyer (2012) by using the power method[6] and the Perron-Frobenius theorem. The power method finds the largest eigenvalue of A to be λ and the associated eigenvector to be r.

The Perron-Frobenius theorem ("The Perron-Frobenius Theorem and the Ranking of Football Teams", Keener", 1993) says that if A is nonnegative, meaning all the score differences between two teams are above zero ($a_{ij} > 0$ for all i and j), and irreducible, meaning there is enough competition to get related ratings, there must be some λ and some vector x both greater than zero to solve $Ax = \lambda x$. The λ in this equation is called the *Perron value* and the vector x is called *Perron vector*. In case, the Perron vector will be our rating vector r.

2.3.2 Implementation

As all our implementations should have the same input, the implementation of Keener's method also requires a matrix X and matrix S as input. In order to convert these wins and losses and scores to useful information for the Keener's method, two additional functions had to be implemented; one to create an O matrix, and one to create an S matrix. The O matrix contains the all the scores a team has scored against another team. The amount of points team 1 scored against team 4 is represented by the number in the first row, on the fourth column. The amount of points team 4, in turn, scored against team 1 is represented by the number on the fourth row, in the first column. Matrix S is constructed using matrix O and contains the measure of score differences. Rather than using a percentage, we – like Keener – have used $a_{ij} = \frac{S_{ij}+1}{S_{ij}+S_{ji}+2}$ to calculate this. After O and S have been determined,

[6]See `http://en.wikipedia.org/wiki/Power_iteration`

the implementation makes use of an open-source method to calculate the Perron vector of S which is the returned rating vector r.

2.4 Elo

2.4.1 Mathematical Background

Another algorithm that provides ratings for the different teams in a competition is the Elo rating system. The Elo rating system was created by Hungarian-born physicist and chess master Arpad Elo (1903-1992). Initially, Elo designed the system to rate chess players, but this system soon proved to be applicable to other two team or two player sports as well. It is currently one of the most prominent systems for rating two-player (or teams) games. The advantage of the Elo system is that this system ensures that a player could play somewhat better or somewhat worse than usual without it having major consequences on the player's rating. Additionally, the Elo system rewards a weaker player more for defeating a stronger player than it rewards a stronger player for defeating a weaker opponent.

The initial premise Elo used to design his system was that each chess player's performance could be seen as a normally distributed random variable X whose mean (μ) could gradually change over time. However, the initial premise that a player's performance was a normally distributed variable was reconsidered as more chess data became available and it provided proof that chess performance is, generally, not normally distributed. The United States Chess Federation and Fédération Internationale des Échecs, which had approved of Elo's rating system in 1960 and 1970 respectively, adopted a version of the Elo rating system in which it was assumed that the expected scoring difference between two players is a logistic function of their ratings.

The Elo system, as it is currently used, requires that one starts with an initial set of ratings for each team or player in the competition. As two teams or two players play against each other, their previous ratings are used to determine the new rating for these teams or players. If team i plays team j, the new rating for team i would be determined using the following formula: $r_i(new) = r_i(old) + K(S_{ij} - \mu_{ij})$. In this formula, $r_i(old)$ is the previous rating held by team i, which will be updated to a newer rating $r_i(new)$, and K is a constant. In the initial version of the Elo rating system as devised by Arpad Elo, S was a player's recent performance. In chess, S would be either one, zero or a half to indicate a win, loss or draw respectively. Performance might also be measured by scores from a single match or the sum of scores

acquired during the tournament, but the most used version of the Elo rating system sticks to the idea that S_{ij} is, similarly to chess, either one, zero or a half indicating whether player i has won, lost or played draw against player j. The advantage of this is that the sum of S_{ij} and S_{ji} is always one and the ratings are therefore relative, as the Elo ratings will always sum up to the same constant value, regardless of how often the ratings are updated. The same would be the case if someone were to use the Elo system where S_{ij} indicates the number of points a team has scored against another team, provided that $S_{ij} + S_{ji} = 1$.

In the above mentioned formula, μ_{ij} is meant to indicate the expected difference in scores between team i and team j if those two teams were playing against each other. It is assumed that the expected score difference between two playing teams is a logistic function of the difference in their ratings. The difference in ratings between teams i and j is d_{ij} and calculated using the old ratings: $d_{ij} = r_i(old) - r_j(old)$. A base-ten logistic function is applied on this difference in ratings between team i and team j, meaning μ_{ij} can be calculated as follows: $\mu_{ij} = L(d_{ij}/\xi) = \frac{1}{1+10^{-d_{ij}/400}}$, where $d_{ij} = r_i(old) - r_j(old)$. The extra parameter, ξ, is the logistic parameter that determines the spread of the ratings. For every ξ rating points team i has a higher rating that team j, the probability that team i beats team j is ten times greater than the probability that team j manages to defeat team i. Adjusting the logistic parameter ξ is a way to fine-tune the system for a particular sport or competition.

Another variable that allows for the fine-tuning of the rating system to fit certain sports or games is the constant value K. The purpose of this constant is to regulate the deviation between old and new ratings. If K is too large, for example, a game that results in a team playing slightly better than expected influences the ratings greatly. If K is too small, however, the Elo rating system will not be able to account for an improvement or deterioration in play and the ratings become too stagnant (and most likely remain similar to the initial ratings the entire time). Therefore, the choice of the value for K is a method for fine-tuning the rating system for a specific sport or even a specific competition. In many sports, such as chess and soccer, the raters allow for a change of the value K over time. Generally speaking, one would allow for a bigger K in the beginning of the competition until a certain number of games have been played, so teams or players have the chance to quickly adjust their initial ratings to ratings more similar to their real performance level, and gradually lower the value of K later on. Additionally, the constant value of K allows us to assign importance to certain games or

tournaments. In soccer, for example, a team is more likely to perform at its best during the world cup and more likely to hold back during friendly, or less important, games. This could be reflected in the algorithm by assigning a larger value for K when it concerns World Cup games or qualifying matches and a lower value for K when the games played are of a more friendly nature.

As previously mentioned, S_{ij} could simply indicate either a win, loss or draw between team i and team j, but it could also incorporate the scores team i scored against team j. It would be unwise to use the raw scores for S_{ij}. If, for example, both team i and team j have a strong offense and weak defense, both S_{ij} and S_{ji} would most likely be very high if one were to take the raw scores for S. If the opposite scenario were true, and both teams' defenses were really good, the scores would be very low in comparison. As the value of S has an effect on the eventual rating, the large scores of the teams in the first scenario would have a disproportionate effect on the ratings when comparing it with the much lower scores of the second scenario. Therefore, it is better to take the total number of points scored during the game into account. Let us, therefore, instead define P_{ij} to be the amount of points team i scores against j. S_{ij} can now be defined as $S_{ij} = \frac{P_{ij}+1}{P_{ij}+P_{ji}+2}$, which ensures that S_{ij} always has a value between zero and one and the sum of S_{ij} and S_{ji} is equal to one. This allows us to interpret S_{ij} as the probability that team i defeats team j.

2.4.2 Implementation

In the implementation of the Elo rating system, we have chosen to have both a version that takes scores into account as well as a version simply based on the wins and losses of a team. As initial ratings, we chose to set the ratings of each team to zero as this ensured that the cumulative ratings and the mean of the ratings would always be zero. Additionally, we determined we wanted to set the value of K at 32 and the value of the logistic parameter ξ at 1000 in both versions of the rating system. The value 32 was chosen, as it was in line with what several internet gaming sites are currently using as a value for K and, through trial-and-error, it seemed to be an appropriate value. The logistic parameter was set at 1000 to provide a good spread in ratings. The final rankings, which are based on the ratings, are not altered or heavily influenced by the logistic parameter.

3 Data

In order to test whether our implemented algorithms worked correctly and to draw conclusions, among other things, about the advantages and disadvantages of each algorithm and compared to each other, we used data from multiple sports and tournaments.

3.1 Windmill Windup

The first set of sport data we used was extracted from `leaguevine.com`, using the API and a python script provided to us by our supervisor. The data contains information about tournaments played during the Ultimate Frisbee tournament *Windmill Windup*. The six individual data files contain information about the team ids, the names of the teams that played during the tournaments, and information about the individual games played during that tournament, such as the wins and losses and scores for each team.

The *Windmill Windup* is an Ultimate Frisbee tournament which takes place over a time period of three days. The tournament features three divisions; open, mixed and women's. The Windmill Windup tournament uses a Swiss Draw format, to ensure that teams are quickly able to play close matches against opponents of similar strength. The teams are ranked after each round of play. To determine the strength of the teams, and therefore the rankings, the Windmill Windup tournament uses the least squares rating system. First, five rounds are played according the Swiss Draw format. After these five rounds, two pools are created; a pool for the teams ranked 1 through 8 and a pool that contains all teams ranked 9 onward. The top ranked teams continue to play in a standard playoff bracket format (quarter finals, semi finals and finals), while the rest of the teams continue with three rounds following the Swiss Draw format.

Ultimate Frisbee itself is a sport in which the two teams play against each other until one of them reaches a score of fifteen (exceptions are made when it takes to long for one of the teams to reach a score of 15).

To summarize:

- **open division (2012)**: 40 teams, 160 games.
- **mixed division (2012)**: 26 teams, 104 games.
- **women's division (2012)**: 14 teams, 56 games.

- **open division (2013)**: 40 teams, 160 games.
- **mixed division (2013)**: 24 teams, 96 games.
- **women's division (2013)**: 16 teams, 64 games.

3.2 National Football League

The second set of data we used is data from the National Football League (NFL), a professional football league in the United States of America. In total, 32 teams participate in the league. Our data stems from the '09/'10 season, in which the first game was played 9^{th} of September 2009 and the last game 7^{th} of February 2010. During this period, 267 games have been played and recorded. During the first seventeen weeks, all teams have played against sixteen other teams and have had one week off. After this seventeen-week period, six teams (four of which are division winners, and two of which have received wild-cards) compete against each other in the NFL playoffs resulting in a single winner. Generally, the score differences between scores in games played in the National Football Leagues are larger than the score differences in Ultimate Frisbee.

We have used this season's data specifically, as this exact season's information was used in Langville and Meyer (2012) to analyse how well different rating systems work in comparison to one another in hindsight and foresight (see section 4.3).

To summarize:

- **NFL '09/'10 season**: 32 teams, 267 games.

3.3 Dummy data

Dummy data was created in order to check whether the implementations of our algorithms worked as they should. The dummy data was designed to look like results of a round-robin tournament, in which all teams play in each round and, eventually, all teams will have played against all other teams. The amount of points a team scored in every game were made to be equal to the column index of the team. In the matrix containing the wins and losses of a tournament, the team which is represented by column 1 will in all games it participated in have a total score of 1 point. The teams represented by the other columns in the matrix will have scored more points in their

games than team 1, and therefore the dummy data represents a tournament in which team 1 will have lost all games. An example of what a tournament represented by our dummy data might look like, where the team names are derived from the column which they are represented by, can be seen in table 1. The advantage of the dummy data, in the first place, was that it was known beforehand what the ranks of each team should be and allowed us to check whether all rating systems did "correctly" predict all ranks (see figure **??**). Team 1, represented by the first column, is the worst playing team and should, according to all algorithms, be the lowest ranking team. The team represented by the last column, in case of the dummy data of table 1 and the matrices in figure 1 team 4, is the best playing team and should therefore be the best ranked team.

Round 1		
Team 1	Team 2	1–2
Team 3	Team 4	3–4
Round 2		
Team 1	Team 3	1–3
Team 2	Team 4	2–4
Round 2		
Team 1	Team 4	1–4
Team 2	Team 3	2–3

Table 1: Example dummy data with four teams.

The dummy data was initially intended to be generated using a Matlab function which requires the amount of teams that compete in the competition as input. Unfortunately, while our implementation of the algorithm that was supposed to generate a round-robin tournament[7] was correct, the algorithm itself turned out to be flawed. Instead, Prolog[8] was used to generate a round-robin tournament in which up to twelve teams could participate (in order for the brute force Prolog script not to overflow).

The output of this function was rewritten to match the data representation of the other data files (see figure 1), using a matrix W[9] to contain information about the wins and losses during the tournament, and a matrix s to contain information about the scores.

[7] `http://stackoverflow.com/questions/6648512/scheduling-algorithm-for-a-round-robin-tournament`

[8] `http://www.swi-prolog.org`

[9] Earlier on referred to as X

$$W = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \end{bmatrix}, s = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 1 & 3 \\ 2 & 4 \\ 1 & 4 \\ 2 & 3 \end{bmatrix}$$

Figure 1: The dummy data as input for the Matlab implementations of the algorithms

4 Tools

To get an insight on whether and how the rating systems work, and how they performed compared to each other, we have created and made use of a few tools described in the following subsections.

4.1 Correctness checks & Validation

First of all, it was important to have definite proof that our implementations, based on the mathematical theory, worked as they should. That is, not only should they assign ratings, but they should assign the ratings one would expect the system to assign. For the least squares algorithm, we used Massey (1997) as a theoretical background and used the example that he used within his chapter on least squares, which consisted of four teams playing five games (with no draws), to compare our results to. The ratings that our least squares rating system assigned to the different teams were identical to the ratings Massey provided they should have.

As mentioned in section 3.1, the Windmill Windup tournament makes of the least squares algorithm to determine the ratings for each team. The site from which the data was extracted, `www.leaguevine.com`, also contains information about the ratings of the playing teams after a tournament. Therefore, by comparing the resulting ratings of our least squares algorithm of the divisions with the ratings on the site of the same divisions, we could validate whether our implementation worked correctly. This was the case.

The NFL data was chosen specifically as a data set, because Langville and Meyer (2012), who provided us with a theoretical background of the Elo and

Keener system, also used this data set to provide example ratings as well as hindsight and foresight accuracies for the Elo algorithm and Keener's method. Again, the ratings that our implementations assigned to the different teams were nearly identical to the ratings the book indicated the teams had when using either the Elo or Keener method.

The dummy data turned out to be a functional check for hindsight. The "perfect" tournament that was created should have a hindsight accuracy of 100%, as the best team with the higher rating of the two playing teams does in fact always win the next game. As the function for hindsight was run on the dummy data, the hindsight accuracy that was returned was in fact 100% for all algorithms.

4.2 Visualisation

Visualisation is a powerful tool to show what is happening during a tournament, as a visual representation of the ratings or rankings of the different teams are, generally, easier to understand for an outsider than numbers. An example of this is figure 2, which shows in a manner that is easily interpretable that all four rating systems (least squares, Elo, Elo scorebased, Keener's method) give the exact same rankings to the eight teams, provided with the dummy data input. Additionally, due to the nature of the dummy data, these are the exact results we would expect. Of course, this same graph could also be made of tournament data which was not "perfect", as the dummy data is, which could show that some algorithms rank more similarly than others.

Another graph that gives us insights in the workings of the different rating systems, is shown as applied on the dummy data of eight teams in igure 3. This figure shows the normalized rating so of each team. Already, using only the dummy data, one can see the difference between the algorithms that take the margin of victory into account (least squares, Keener's method, Elo scorebased) and the one that does not (Elo), as the first three assign virtually identical normalized scores, while the Elo curve differs somewhat.

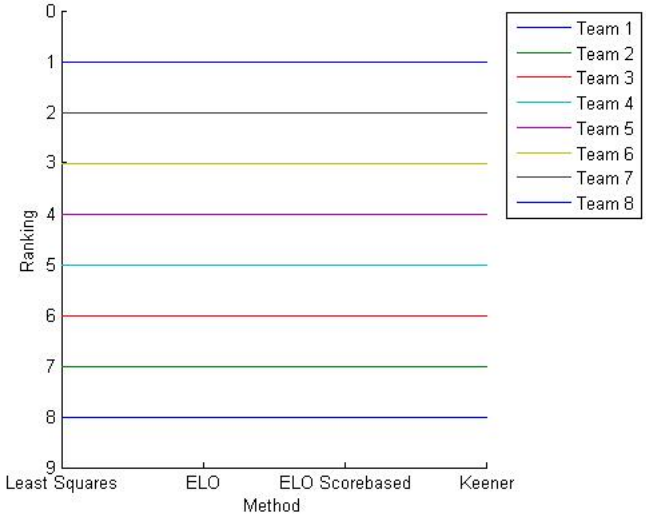

Figure 2: *A graph illustrating the ranks of the different teams according to the different algorithms applied on dummy data with 8 teams*

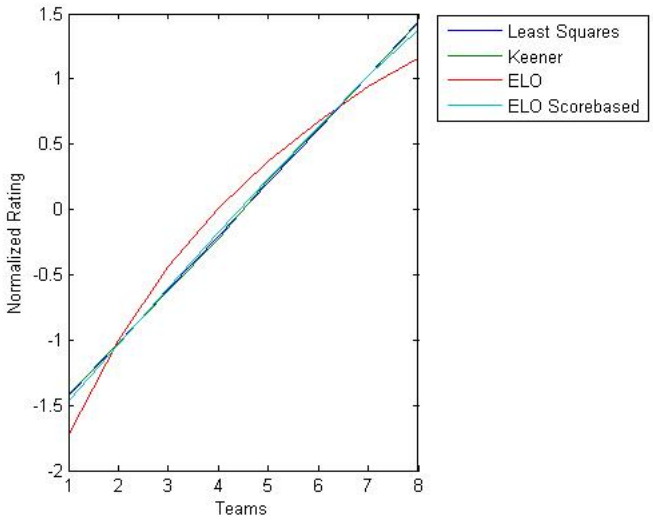

Figure 3: *A graph illustrating the normalized ratings of the different teams using dummy data with 8 teams*

Additionally, we have implemented a visualisation tool that, given a win-loss matrix, the corresponding scores and a method of calculating the ratings, visualizes how the ratings change over time during a tournament. In figure

4, you can see what the ratings are after each game and how they change during the tournament, when the input is the dummy data with the Elo rating system.[10] In figure 5, we have added the information that each round consists of four games as a parameter to the visualisation tool, meaning the graph only shows how the rating changes after each round rather than after each game. For round-robin tournaments, in which multiple teams play against another team at the same time, this might be a more efficient and effective visualisation in the case that teams want to know how each round affected their rating. As shown in figure 6, we have also allowed for the possibility that the user can specify of which teams it wants to view the progress throughout the tournament. In figure 6, the ratings that can be seen are those of team 1, team 2 and team 8. Especially for competitions or tournaments with a large number of participating teams, this extra feature can be a useful tool for the individual teams to understand how their ratings came to be.

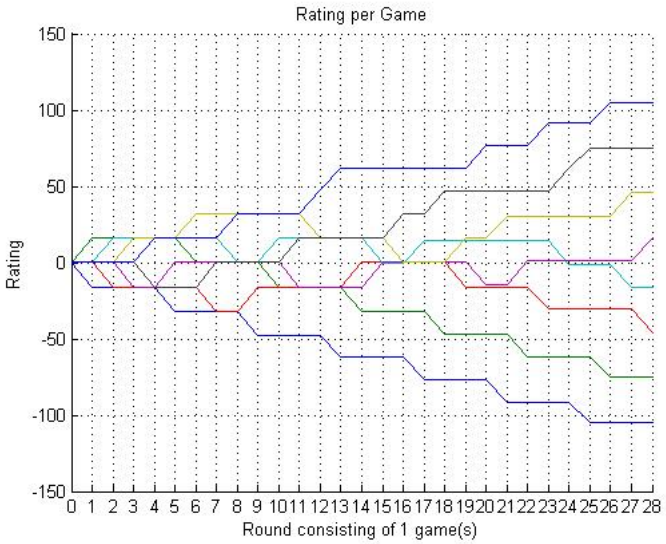

Figure 4: *Rating changes through dummy tournament, using Elo*

[10]For Elo, of course, the score differences are not necessary

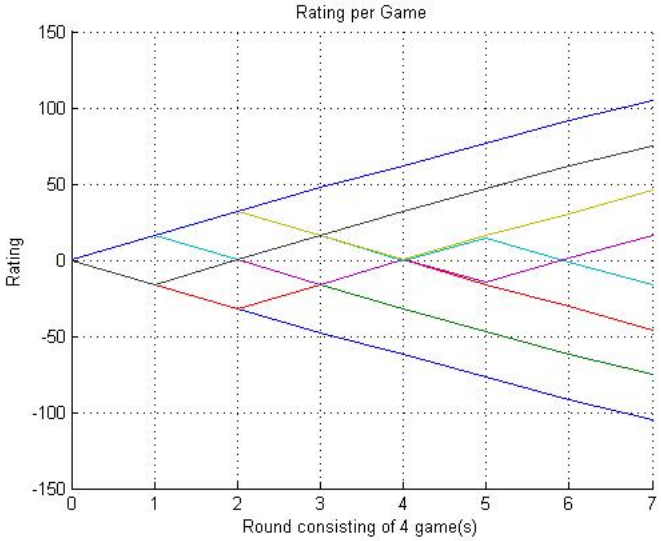

Figure 5: *Rating changes through dummy tournament, using Elo and viewed per round*

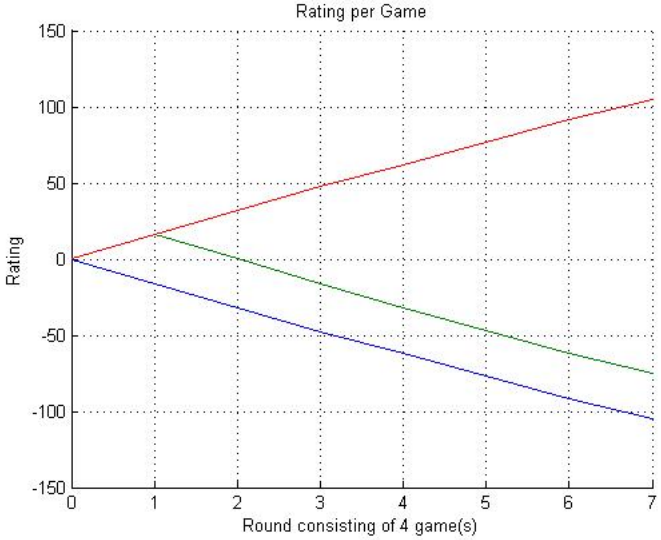

Figure 6: *Rating changes through dummy tournament, using Elo, viewed per round and for team 1, 2, and 8*

4.3 Hindsight and Foresight

One could judge how well a rating system works in two ways; how well a system explains what has happened so far and how well it predicts future games. How well the rating system explains what has happened is called *hindsight*. Hindsight is how well the resulting ratings at the end of the tournament can explain all games that led to those ratings. To determine what the hindsight accuracy is, one should take the ratings at the end of the tournament. For each game that has been played throughout the tournament, these final ratings should be able to correctly say which of the two playing teams has won the game. If the final rating of team i is higher than the final rating of team j, and these two teams played against each other in the first game, the ratings suggest that team i should have won that game. The percentage of games for which the result that the final ratings suggest is equal to the actual wins and losses in those games, is the *hindsight accuracy*.

How well a rating system correctly predicts the outcomes of future games is called *foresight*. A rating system correctly determines a future game if it assigns the ratings to the teams in such a way that the rating of the team that wins the future game is higher than the rating of the team that loses that game. Additionally, if the rating system takes scores into account, it should correctly predict the difference between what the winning team and the losing team scores too. To determine what the foresight accuracy is of a rating system, based on some tournament data, one would have to look at how well the ratings predict each game apart from the first one, as the system has not yet assigned a rating to the teams if the first game has not been played yet. For each game, one takes the last updated ratings and, knowing which teams will play the next game, determine what team the current ratings suggest should win that game. Then, the ratings of each team after the results of that game have been processed are taken and, again, one determines which team should theoretically win the game after that one. This is done until all games, apart from the first played game, have been predicted. The percentage of games in which the prediction of which team should theoretically win is the same as the team that actually wins, is the *foresight accuracy*. Obviously, if a game ends in a draw and, based on the rating system, a draw was predicted, this should result in a higher accuracy.

5 Results and Discussion

5.1 Advantages and Disadvantages in theory

Based on the mathematics underlying the algorithms and the knowledge we have about the data these algorithms can be applied to, we can make some claims that seem plausible which we will then intend to either prove or disprove. For example, one could suspect that, depending on the sport, it might be useful to consider the margins of victory in determining the ratings of the different teams participating in the competition. Margin systems, such as Elo scorebased, least squares and Keener, are expected to perform well on the same type of data, while the Elo rating system, will most likely perform well on a different set of data as it does not take the margin of victory into account. This could already be seen by looking at figure **??**, where the normalized ratings of the margin systems were nearly identical while the Elo ratings differed. The margin systems might, for example, have more trouble with dealing with outliers (extreme results) in scores than a system that simply bases its ratings the wins and losses. Additionally, we would expect both the east quares rating system as well as Keener's method would allow for faster convergence to the the real strengths than either Elo or Elo scorebased will do. This is because, using either least squares of Keener's method, every new game that is introduced can alter the ratings of all the teams participating in the competition rather than solely the teams that have played in the game. Also, we would expect that the ratings determined by using the least squares algorithm would more easily be able to provide us with a prediction of what the score difference will be in any given game. This is because the basic premise of the least-squares algorithm is that the margin of victory is the difference in ratings between two playing teams.

To summarize, what we will be trying to investigate is the following:

- Does the type of sport influence the performance of a rating system?
- What algorithms converge fastest to the "real" strength?
- Can least squares more easily predict future margins of victory between teams, in comparison to Keener and Elo (scorebased)?

5.2 Results

For the different data sets we had at our disposal, the hindsight accuracies of the different algorithms are shown in table 2 and the foresight accuracies are shown in table 3. The first conclusion we can draw is that the algorithms all, on our specific data sets, perform better in hindsight than in foresight. As expected, the hindsight accuracy for all the algorithms on the dummy data set is perfect. The foresight accuracy of the Elo rating system seems to be much lower from both its hindsight accuracies as well as the foresight accuracies of the other rating systems. Interestingly enough, the foresight accuracy of the Elo algorithm on the NFL data set is much less below the average foresight accuracy on the data set than the foresight accuracies of the Elo rating system on the different Windmill Windup data sets, which are all much lower than the average foresight accuracies on those data sets. This might indeed suggest that our initial suspicion that the type of sport influences the performance of a rating system. However, we suspect that the cause of this lower foresight accuracy of Elo might, at least partly, have to do with the fact that when there is not enough information yet to make accurate predictions (mainly in the first few games of a tournament), Elo predicts a lot of draws as a lot of ratings have a value equal to that of another team and are only altered after a team plays again. This is due to the nature of the Elo algorithm, which only takes the wins and losses into account and not the scores. A win is rewarded equally in rating for each playing team during the first round. The ratings do not change until a team has played a next round, and therefore remain equal for a longer period of time than they would using an algorithm in which all games influence all ratings. The Elo algorithm predicts a draw more often than other methods, as a win or loss is reflected in the teams rating equally for each team early in the competition, which decreases the foresight accuracy. That the ratings stay the same value for a longer amount of time is perhaps most obvious when comparing figure 9 to either figure 8, figure 7 or figure 10.

What can be seen from the graphs representing the ratings of the different teams throughout the tournament is that the ratings seem to converge to a certain rating and remain that way much faster for the least squares algorithm and Keener's method. Especially when comparing the graph representing ratings according to the Elo algorithm throughout the tournament of the dummy data (figure 9) and the graph of the same data using the least squares algorithm (figure 7, it is apparent that the ranks that the teams eventually get can be determined using much less information and much earlier in the tournament using the least squares algorithm than the Elo algorithm. The

same seems the case for Keener's method (figure 8) on the data set, where most of the ranks apart from the middle two seem to be decided much earlier on than is the case with Elo's algorithm. Shuffling the order of the games for this dummy data, as seen in figure 11 lets the Keener algorithm converge as fast as the least squares algorithm. NFLThis indicates that the least squares rating system and Keener's method converge much faster to the ratings they will have at the end of the tournament, or to a rating very similar to that, than the Elo algorithm. This claim is supported by the other data, such as the NFL season data (see figure 12, figure 13 and figure 14).

	least squares	**Keener**	**Elo**	**Elo scorebased**
NFL '09/'10	72.66	72.28	75.28	71.16
WW open '12	83.13	78.13	79.378	70.00
WW mixed '12	76.92	78.85	77.88	74.04
WW women '12	87.50	85.71	85.71	75.00
WW open '13	84.38	75.63	77.50	68.13
WW mixed '13	78.13	79.17	80.21	72.92
WW women '13	87.50	79.69	81.25	76.56
Dummy 4 teams	100.00	100.00	100.00	100.00
Dummy 6 teams	100.00	100.00	100.00	100.00
Dummy 8 teams	100.00	100.00	100.00	100.00

Table 2: Hindsight accuracy of rating systems for available data sets

	least squares	**Keener**	**Elo**	**Elo scorebased**
NFL '09/'10	57.30	61.44	55.06	59.93
WW open '12	57.50	54.38	25.63	51.25
WW mixed '12	53.85	51.92	31.73	48.08
WW women '12	58.93	60.71	44.64	55.36
WW open '13	51.88	54.38	35.63	47.50
WW mixed '13	50.00	48.96	43.75	43.75
WW women '13	65.63	54.69	39.06	50.00
Dummy 4 teams	50.00	50.00	33.33	50.00
Dummy 6 teams	73.33	66.67	60.00	60.00
Dummy 8 teams	64.29	71.43	42.86	71.43

Table 3: Foresight accuracy of rating systems for available data sets

5.3 Conclusions & Future Research

We were able to confirm that the algorithms behave differently and might be more effective on some data, however, unfortunately, we have not been able to determine what specific types of data the algorithms are more effective on in comparison to the other algorithms. It might be interesting to research what rating systems are better rating providers for specific sports. We do know that least squares is very effective in general, but does not allow the user to tune the system in any way. For Keener's method, however, the implementation can be altered to fit a specific sport or data set by skewing, applying LaPlace's rule of succession and normalization. For Elo, both the regular algorithm and the scorebased function, the logistic parameter and constant K can be altered to suit a specific sport. Even though least squares seems, based on our results, to be a very good rating system, it might be interesting to investigate whether one of the algorithms can be improved in such a way that it performs much better on a specific data set, or even a specific goal (improving hindsight or foresight accuracy, predicting ratings, predicting rankings).

In order to do this, one would have to have access to more data sets than we currently had available. As we only had the Windmill Windup and NFL data sets at our disposal, this could not provide us with enough information to draw such general conclusions from. During the course of this project, we have managed to arrange a meeting between ourselves and *Infostrada*, a company with a large database of sport information. A collaboration between the two parties could be beneficial to both. As shuffling the games in a tournament (meaning the exact same games were played, only the order in which they were played was different) had an effect on how fast the ratings determined by Keener's method converged to the ratings the teams had at the end of the tournament, it might also be interesting to look at what influence tournament formats have on the performance of the rating systems.

Another possibility is to take a look at how Elo performs in foresight in comparison to the other algorithms when the resulting ratings of the first few games are not taken into account. And as we have not been able to provide definite proof whether least squares will more easily predict the margin of victory between two playing teams based on their ratings than the other algorithms, as might still be worth investigating as well. One would have to devise a system to predict the margin of victory between two teams based on the other algorithms, and compare its performance to that of the least squares algorithm.

Lastly, we have not been able to complete our second goal which was to design some graphical representation that could provide the participants in a tournament with some insight as to why their ratings are as they are. This is still something that could be done, especially after seeing how effective simple visualisation tools as described in section 4.2 were to draw conclusions from.

References

Doug (2006a). *Another rating system: maximum likelihood*. URL: http://www.pro-football-reference.com/blog/?p=171.

— (2006b). *Maximum likelihood, part II*. URL: http://www.pro-football-reference.com/blog/?p=206.

— (2006c). *Maximum likelihood with home field and margin of victory*. URL: http://www.pro-football-reference.com/blog/?p=210.

James P. Keener“ (1993). “The Perron-Frobenius Theorem and the Ranking of Football Teams”. In: *SIAM Review* 35.1, pp. 80–93.

Amy N. Langville and Carl D. Meyer (2012). *Who's #1?: The Science of Rating and Ranking*. Princeton, NJ: Princeton University Press.

Kenneth Massey (1997). *Statistical Models Applied to the Rating of Sports Teams*.

Wikipedia (2013). *Sports Rating System*. URL: http://en.wikipedia.org/wiki/Sports_rating_system.

Appendix I

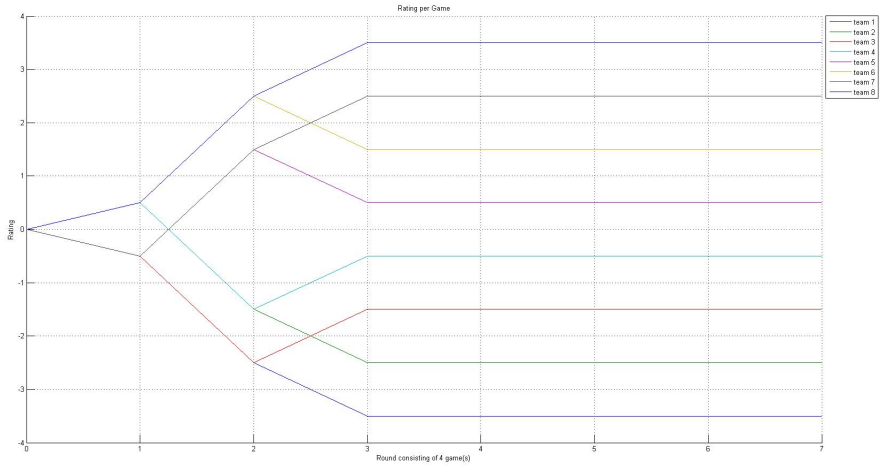

Figure 7: *least squares rating system on the dummy data with 8 teams*

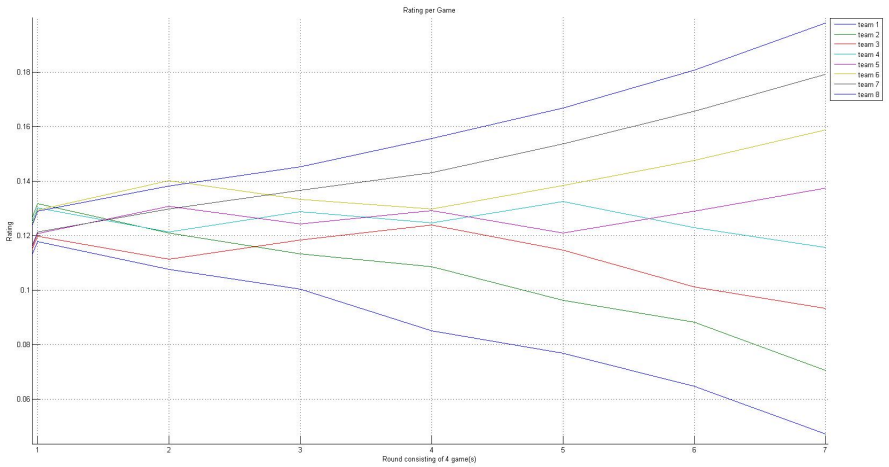

Figure 8: *Keener rating system on the dummy data with 8 teams*

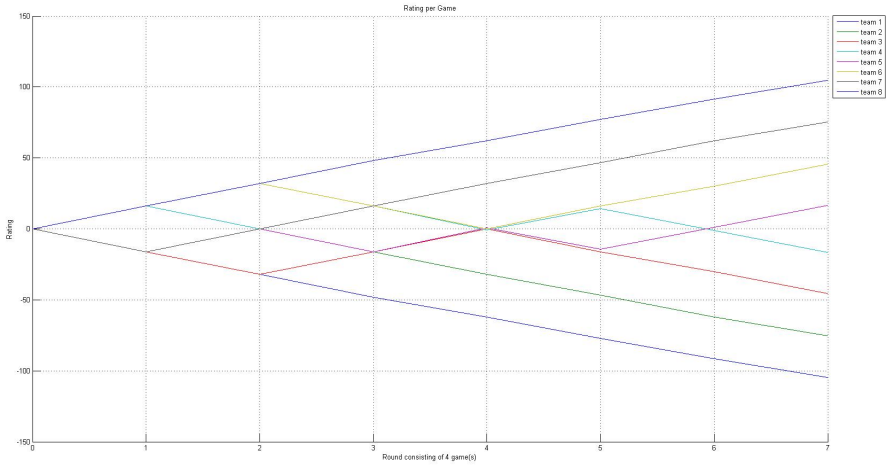

Figure 9: *Elo rating system on the dummy data with 8 teams*

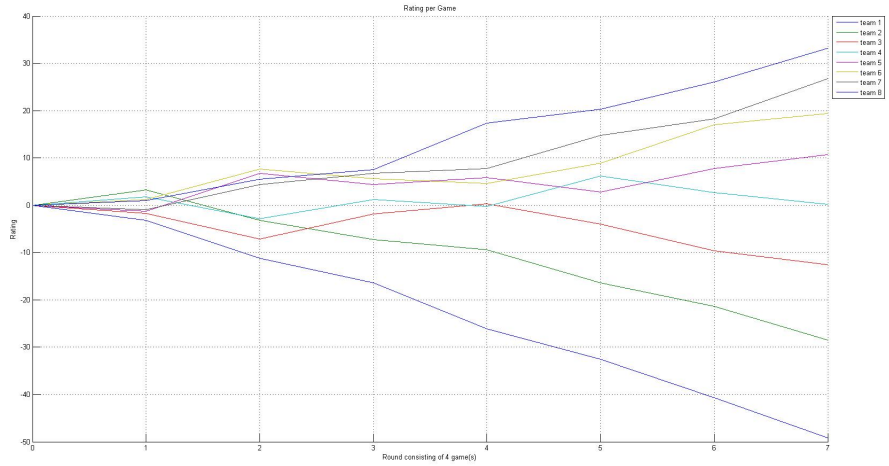

Figure 10: *Elo scorebased rating system on the dummy data with 8 teams*

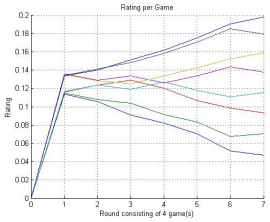

Figure 11: *Keener rating systems on shuffled dummy data with 8 teams*

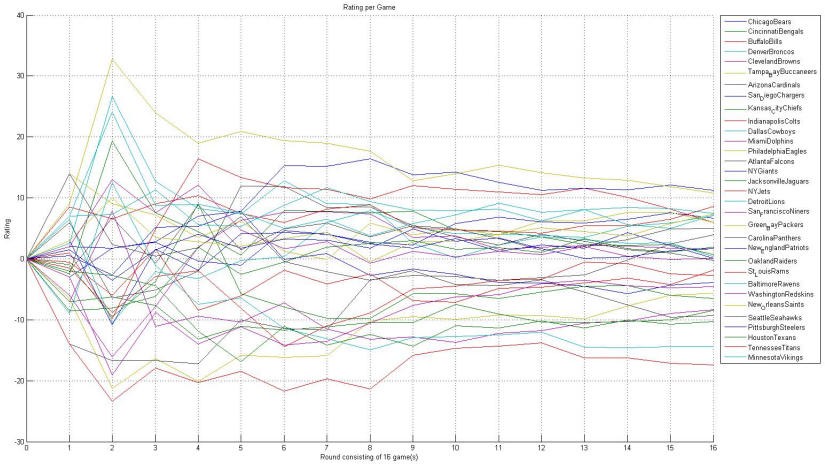

Figure 12: *least squares rating system on the NFL data*

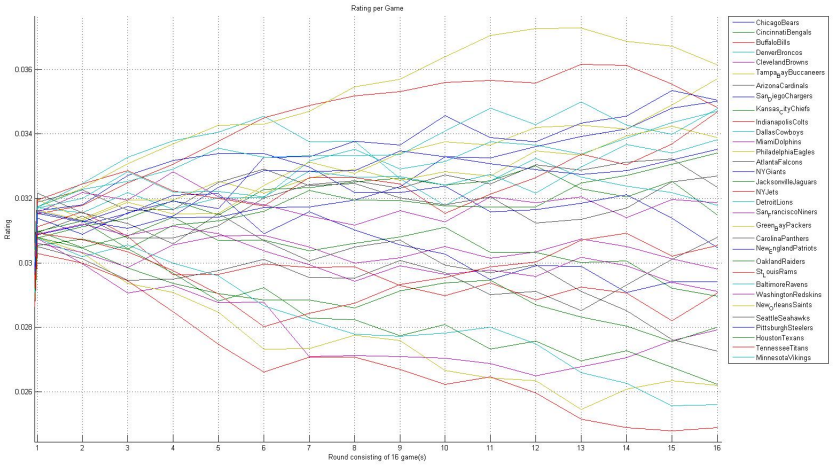

Figure 13: *Keener rating system on the NFL data*

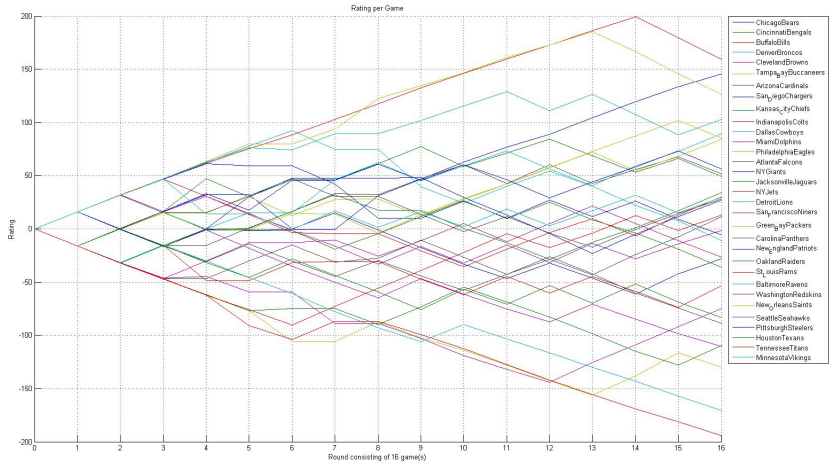

Figure 14: *Elo rating system on the NFL data*

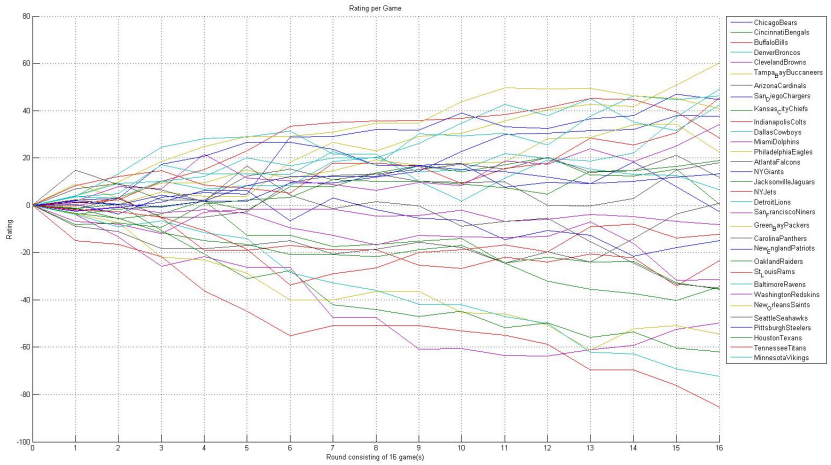

Figure 15: *Elo scorebased rating system on the NFL data*

For all the other figures, graphs and plots see `https://www.dropbox.com/sh/6ib67xgr3z4zywp/FluVGxu4D2`

Appendix II

Rating systems

LEAST SQUARES

```
function [ r ] = least_squares(W, s)
% Cast from int to double if necessary
W = double(W);
s = double(s);
% If s contains scores, calculate score differences.
if(size(s, 2) == 2)
    s = abs(s(:,1) - s(:,2));
end
%compute
X = W' * W;
y = W' * s;
% Scaling
y(size(y,1)) = 0;
X(size(X,1),:) = ones(1,size(X,1));
% Calculate ratings (~ to supress output)
[r, ~] = lsqr(X,y,[],100);
end
```

ELO

```
function [ r ] = elo( W )
% get amount of games played and amount of teams playing.
nog = size(W,1);
not = size(W,2);

% initialise variables
r_old = zeros(not,1);
r_new = zeros(not,1);
mu = zeros(1,not);
S = zeros(1,2);

% set parameters: logistic parameter P and K.
P = 1000;
K = 32;

% loop through each game
for c = 1:nog,
    game = W(c,:); % get the individual game
    playing = find(game ~= 0); % get array of indices of playing
        teams
    i = playing(1); % define i (index of first playing team)
    j = playing(2); % define j (index of last playing team)
```

```
22
23      % get S
24      if(game(i) > game(j))
25          S(i) = 1;
26          S(j) = 0;
27      elseif(game(i) < game(j))
28          S(i) = 0;
29          S(j) = 1;
30      elseif(game(i) == game(j))
31          S(i) = 0.5;
32          S(j) = 0.5;
33      end
34
35      %calculate mu for i and j
36      mu(i) = 1 / (1 + 10^(-1*((r_old(i) - r_old(j)))/P));
37      mu(j) = 1 / (1 + 10^(-1*((r_old(j) - r_old(i)))/P));
38
39      %get new ratings
40      r_new(i) = r_old(i) + (K * (S(i) - mu(i)));
41      r_new(j) = r_old(j) + (K * (S(j) - mu(j)));
42
43      %reset mu and r_old
44      mu = zeros(1,not);
45      r_old = r_new;
46  end
47
48  %return ratings.
49  r = r_new;
50  end
```

Elo scorebased

```
1   function [ r ] = elo_scorebased( W, s )
2   nog = size(W,1);
3   not = size(W,2);
4
5   % get scores, amount of games played and amount of teams playing
        .
6   O = createO(W,s);
7   S = createS(O);
8
9   % initialise variables
10  r_old = zeros(not,1);
11  r_new = zeros(not,1);
12  mu = zeros(1,not);
13
14  % set parameters logistic parameter P and K.
15  P = 1000;
16  K = 32;
```

```
17
18  % loop through each game
19  for c = 1:nog,
20      game = W(c,:); % get the individual game
21      playing = find(game ~= 0); % get array of indices of playing
            teams
22      i = playing(1); % define i (index of first playing team)
23      j = playing(2); % define j (index of last playing team)
24
25      %calculate mu for i and j
26      mu(i) = 1 / (1 + 10^(-1*((r_old(i) - r_old(j)))/P));
27      mu(j) = 1 / (1 + 10^(-1*((r_old(j) - r_old(i)))/P));
28
29      %get new ratings
30      r_new(i) = r_old(i) + (K * (S(i,j) - mu(i)));
31      r_new(j) = r_old(j) + (K * (S(j,i) - mu(j)));
32
33      %reset mu and r_old
34      mu = zeros(1,not);
35      r_old = r_new;
36  end
37
38  %return ratings.
39  r = r_new;
40  end
```

Keener

```
1   function [ r ] = keener(W, s)
2   not = size(W,2);
3
4   %   Every point is a win, while it gives less chance on 0 wins
5   O = createO(W,s);
6   S = createS(O);
7
8   % Skewing
9   for i = 1:not,
10      for j = 1:not,
11          S(i,j) = 0.5 + ((sign(S(i,j)-0.5)*sqrt(abs(2*S(i,j)-1)))
                /2);
12      end
13  end
14
15
16  % SOLVE R
17  [~, r] = perron(S);
18
19  end
```

Tools

Visualisation

```
function [] = visual(W, s, method, teams, round_length,
    team_names)
% if 3 arguments, display all teams with round length one
if nargin == 3,
    teams = 'all';
    round_length = 1;
end

% if round_length not set, set at 1
if nargin == 4,
    round_length = 1;
end

% get datamatrix for arguments
data = datamatrix(W, s, method, round_length);
nor = size(data,2);
not = size(data,1);

% display graph
xlabel(['Round consisting of ' num2str(round_length) ' game(s)'
    ])
ylabel('Rating')
title('Rating per Game')
xlim([1,nor])
set(gca, 'XTick', 1:nor);

if(strcmp(teams, 'all') == 1)
    for j = 1:not
    hold all
    plot(data(j,:))
    end
else
    for i = 1:size(teams,2)
        hold all
        plot(data(teams(i),:))
    end
end

x = 0:1:nor-1;
set(gca, 'XTickLabel', x);
if nargin == 6,
    l = team_names(1);
    for i = 2:length(team_names)
        l = [l; team_names(i)];
    end
```

```
44      legend(l,'Location','NorthEastOutside')
45  end
46  grid on
47
48  end
```

Comparison

```
1   function [] = comparison( W,s )
2   % This function compares al the implemented rating methods by
        displaying a
3   % plot with al the final rankings and displaying all the
        hindsight and
4   % foresight percentages.
5   not = size(W,2); %number of teams
6
7   % get hindsight scores for all methods
8   hindsight_ls = hindsight(W, s, 'ls');
9   hindsight_keener = hindsight(W, s, 'keener');
10  hindsight_elo = hindsight(W, s, 'elo');
11  hindsight_eloscores = hindsight(W, s, 'eloscores');
12
13  % display hindsight scores for all methods
14  display(hindsight_ls)
15  display(hindsight_keener)
16  display(hindsight_elo)
17  display(hindsight_eloscores)
18
19  % get all foresight scores
20  foresight_ls = foresight(W, s, 'ls');
21  foresight_keener = foresight(W, s, 'keener');
22  foresight_elo = foresight(W, s, 'elo');
23  foresight_eloscores = foresight(W, s, 'eloscores');
24
25  % display all foresight scores
26  display(foresight_ls)
27  display(foresight_keener)
28  display(foresight_elo)
29  display(foresight_eloscores)
30
31  ratings = zeros(not,4);
32  ratings(:,1) = least_squares(W,s);
33  ratings(:,4) = keener(W,s);
34  ratings(:,2) = elo(W);
35  ratings(:,3) = elo_scorebased(W,s);
36
37  % Normalization
38  ratings = bsxfun(@minus,ratings,mean(ratings));
39  ratings = bsxfun(@rdivide,ratings,std(ratings));
```

```
40
41 figure(1)
42 x = 1:not;
43 plot(x, ratings(:,1), x, ratings(:,2), x, ratings(:,3), x,
       ratings(:,4))
44 xlim([1, not]);
45 xlabel('Teams')
46 ylabel('Normalized Rating')
47 set(gca,'XTick',1:not)
48 leg = legend('least squares', 'Keener', 'ELO', 'ELO scorebased')
       ;
49 set(leg,'Location','NorthEastOut')
50
51 rankings_ls = sort(ratings(:,1), 'descend');
52 rankings_keener = sort(ratings(:,2), 'descend');
53 rankings_elo = sort(ratings(:,3), 'descend');
54 rankings_eloscores = sort(ratings(:,4), 'descend');
55 rankings = zeros(not,4);
56
57 for i = 1:not
58     index_ls = find(ratings(:,1) == rankings_ls(i));
59     index_keener = find(ratings(:,2) == rankings_keener(i));
60     index_elo = find(ratings(:,3) == rankings_elo(i));
61     index_eloscores = find(ratings(:,4) == rankings_eloscores(i)
           );
62     rankings(i,1) = index_ls;
63     rankings(i,2) = index_keener;
64     rankings(i,3) = index_elo;
65     rankings(i,4) = index_eloscores;
66 end
67
68 display(rankings)
69
70 figure(2)
71 x = 1:not;
72 for j = 1:not
73     hold all
74     [indices, ~] = find(rankings == j);
75     name = sprintf('Team %d', j);
76     plot(1:4, indices, 'DisplayName', name);
77 end
78 xlabel('Method')
79 ylabel('Ranking')
80 set(gca,'XTick',1:4)
81 ylim([0,not+1]);
82 set(gca,'YTick',0:not+1)
83 set(gca,'XTickLabel',{'least squares';'ELO';'ELO scorebased';'
       Keener'})
84 set(gca,'YDir','reverse');
```

```
85 leg = legend('-DynamicLegend');
86 set(leg,'Location','NorthEastOut')
87
88 end
```

For all code see https://www.dropbox.com/sh/275kj5pgzoov4pd/QIVtRm0I3E