Comparing Predictive Performance Of Chess Ratings With The **PlayerRatings** Package

Alec Stephenson

July 24, 2012

Summary

This document presents an example of the use of **PlayerRatings**, using a large dataset to analyse different methods for rating chess players. The analysis shows that, in terms of predictive performance, the Elo system is outperformed by both the Glicko and Stephenson systems. The analysis also suggests that the K factors used in the current FIDE (World Chess Federation) implementation of the Elo system are smaller than what would be needed for optimum predictive performance.

This pdf document is located in the file `doc/ChessRatings.pdf` and the corresponding sweave file is located at `doc/sweave/ChessRatings.Rnw`. The sweave file can be converted to a latex file by calling `Sweave(file)`, where `file` is the filename `system.file("doc/sweave/ChessRatings.Rnw", package = "PlayerRatings")`. This will run all the code in this document and will therefore download about 20 MB of data. The latex file and corresponding pdf figures will then be located in the working directory.

1 Functions and Datasets

The **PlayerRatings** package implements iterative updating systems for rating players (i.e. individuals or teams) in two-player games. These methods are fast and surprisingly accurate. The idea is that given games played in time period t, the ratings can be updated using only the information about the status of the system at the end of time period $t-1$, so that all games before t can be ignored. The ratings can then be used to predict the result of games at time $t+1$. Comparing the game predictions with the actual results gives a method of evaluating the accuracy of the ratings as an estimate of a player's true skill.

The result of a game is considered to be a value in the interval $[0, 1]$. For the chess data, a value of 1 represents a win for white, a value of 0 represents a win for black, and a value of 0.5 represents a draw. The status of the system is typically a small number of features, such as player ratings, player rating (standard) deviations, and the number of games played. The more computationally intensive (and often slightly more accurate)

approaches of using the full gaming history via a time decay weighting function is not considered here.

The functions `elo` and `fide` implement the Elo system (Elo, 1978), the function `glicko` implements the Glicko system (Glickman, 1999), and the function `steph` implements the Stephenson system as detailed in the appendix. There are other functions to aid incorporating additional complexity into the K factor of the Elo system, to predict the result of future games, to produce appropriate plots, and to evaluate predictive performance.

2   Chess Data Creation

The `chess` dataset contains approximately 1.8 million games played over the eleven year period $1999 - 2009$ by 54205 chess players. We create the dataset by downloading comma separated value files from the website Chessmetrics. The data was constructed and made publicly available by Jeff Sonas. We will use the first nine years of data. We take training data from the period $1999 - 2005$, test data from the year 2006 and validation data from the year 2007. The three zip files are about 6MB each and may take some time to download on slow internet connections.

```
> cm <- "http://www.chessmetrics.com/KaggleComp/"
> temp <- tempfile()
> download.file(paste(cm,"primary_training_part1.zip",sep=""),temp)
> chess <- read.csv(unz(temp, "primary_training_part1.csv"))[,2:5]
> download.file(paste(cm,"primary_training_part2.zip",sep=""),temp)
> chess <- rbind(chess,
+   read.csv(unz(temp, "primary_training_part2.csv"))[,2:5])
> download.file(paste(cm,"primary_training_part3.zip",sep=""),temp)
> chess <- rbind(chess,
+   read.csv(unz(temp, "primary_training_part3.csv"))[,2:5])
> names(chess) <- c("Month","White","Black","Score")
> unlink(temp)


> train <- chess[chess$Month < 84.5,]
> trainM <- train$Month
> test <- chess[chess$Month > 84.5 & chess$Month < 96.5,]
> testS <- test$Score
> valid <- chess[chess$Month > 96.5 & chess$Month < 108.5,]
> validS <- valid$Score
```

The dataset `cSt` is created below. It contains FIDE ratings for 14118 chess players at January 1999, before the data in the `chess` dataset were recorded. We will use `cSt` in the remainder of this document to initialize the system. This is not required, however if the information exists it makes sense to use it. It is not an ideal initialization for all methods, but appears to always work better than initializing every player at fixed values. Other players that subsequently enter the system are initialized according to the argument `init` of the modelling functions.

```
> temp <- tempfile()
> download.file(paste(cm,"initial_ratings.csv",sep=""),temp)
> cSt <- read.csv(temp)
> cSt <- data.frame(Player = cSt$Player, Rating = cSt$Rating,
+   Deviation = 200, Games = cSt$NumGames)
> unlink(temp)
```

Finally, we download data that identifies players from their identification numbers in the **chess** dataset. Some names contain non-ASCII characters that may not be read correctly.

```
> temp <- tempfile()
> download.file(paste(cm,"players.csv",sep=""),temp)
> chessPlayers <- read.csv(temp, as.is=TRUE)
> names(chessPlayers) <- c("Player","Name")
> unlink(temp)
```

3   Modelling and Prediction

In this section we will demonstrate the features of the **PlayerRatings** package by comparing the predictive performance of alternative methods applied to the **chess** dataset.

All modelling functions in the package can be used to update player ratings over several time periods, or over individual time periods. For example, the following code uses the Elo system to iteratively update the chess ratings once every month for each of the 84 months in the **train** data. The state of the system is contained in the **ratings** component of the returned object, which can then be passed back into the function for subsequent updates.

```
> library(PlayerRatings)
> robje1 <- elo(train[trainM==1,], cSt)
> for(i in 2:84) robje1 <- elo(train[trainM==i,], robje1$ratings)
```

More simply, we can call the function once to perform the same task.

```
> robje1 <- elo(train, cSt, init=2200, gamma=0, kfac=27)
```

The specified parameters are the defaults. The argument `init` specifies the initial rating for players who are added to the system. The argument `gamma` can account for the advantage of white, however it appears to have little effect for the chess data. The argument `kfac` is the K factor, which by default is equal to 27 for all players. The Elo system is fairly simple, and so several implementations introduce additional complexity by allowing the K factor to depend on aspects of the model such as the player rating or the number of games played by the player. The following give examples of this, where `kfac` is specified using a function that is provided by the package.

```
> robje2 <- elo(train, cSt, kfac=krating, rv=2300, kv=c(32,26))
> robje3 <- elo(train, cSt, kfac=kgames, gv=30, kv=c(32,26))
```

The `robje2` object employs a K factor of 26 for players rated above 2300 and a K factor of 32 otherwise. The `robje3` object employs a K factor of 26 for players who have played more than 30 games and a K factor of 32 otherwise.

The function `fide` also implements the Elo system, but uses exactly the same default parameters and K factors as FIDE, and consequently allows a little more flexibility. It does not implement the initialization system of FIDE, which would require knowledge of the tournaments that correspond to the games. Despite this, it can still be used to gain some insight into the FIDE ratings implementation. In the following, we call `fide` twice: the first with default arguments and the second with the K factor increased by 5 for players who have played 30 or more games. We also call the functions `glicko` and `steph` to implement the Glicko and Stephenson systems respectively.

```
> robjf1 <- fide(train, cSt)
> robjf2 <- fide(train, cSt, kv = c(15,20,30))
> robjg <- glicko(train, cSt, init=c(2200,300), gamma=0, cval=15)
> robjs <- steph(train, cSt, init=c(2200,300), gamma=0, cval=9,
+   hval=9, bval=0, lambda=2)
```

The `steph` implementation was developed by Alec Stephenson in 2012 as a variant of his winning entry in a competition to find the most useful practical chess rating system, organized by Jeff Sonas on Kaggle, a platform for data prediction competitions. The details are given in the appendix as they are not available elsewhere. The `bval` parameter can be used to give a per game bonus to each player; it typically improves prediction accuracy but it also creates ratings inflation, which may not be desirable.

The seven objects we have created are S3 objects of class `"rating"`, with corresponding `print`, `summary`, `predict`, `plot` and `hist` methods. The following code uses the `predict` method in conjunction with the `metrics` function to compare our seven rating methods by evaluating their predictive performance on the 2006 test data. The advantage of white must be accounted for when making predictions. The `predict` function has a white advantage parameter `gamma` which by default is set to the value 30, as this seems to be roughly optimal across all systems.

```
> pre1 <- predict(robje1, test); pre2 <- predict(robje2, test)
> pre3 <- predict(robje3, test); prf1 <- predict(robjf1, test)
> prf2 <- predict(robjf2, test)
> prg <- predict(robjg, test); prs <- predict(robjs, test)
> metrics(testS, cbind(pre1,pre2,pre3,prf1,prf2,prg,prs))
```

```
        bdev    mse     mae
prs   88.668  84.509  86.303
prg   88.781  84.635  86.109
pre2  88.914  84.774  86.639
pre3  88.928  84.786  86.681
prf2  88.996  84.859  87.002
pre1  89.034  84.896  86.811
prf1  89.390  85.282  87.573
```

The `metrics` function implements three metrics, scaled so that random guessing corresponds to the number 100. The first is the binomial deviance, which is the most appropriate metric for chess data. Smaller values on all metrics correspond to more accurate predictions. We see that Stephenson is best, followed by Glicko, then Elo (2), Elo (3), FIDE (2) and Elo (1). The FIDE (1) method is a worst because the parameters were not optimized.

To quantify the comparison, we can say that Stephenson gives a

$$\frac{(89.034 - 88.668)}{(100 - 89.034)} = 3.34\% \tag{1}$$

improvement over Elo (with a constant K factor of 27) for this dataset under this metric, whereas Glicko gives a 2.31% improvement over Elo, and Elo gives a 3.36% improvement over the FIDE (1) implementation. We also see that FIDE (2) outperforms FIDE (1) because increasing the K factor by 5 for players who have played 30 or more games gives an increase in predictive performance, with an improvement of 3.71%.

The Elo system has been in existence for more than 50 years. These results suggest that for the chess data, rather than attempting to add complexity to the K factor, a better approach for predictive performance is to use systems such as Glicko or Stephenson, which use a rating deviation value to explicitly model the accuracy of the ratings as an estimate of skill. Players who have not played many games may have very high or very low ratings with large rating deviation values. It therefore makes sense under these systems to only consider a rating official when the player has played some fixed number of games or when the rating deviation decreases below some fixed threshold. We see from the above that Stephenson improves over the Elo implementation of FIDE by 6.80%, and Glicko improves over the Elo implementation of FIDE by 5.74%.

With the exception of Elo (fide), the default parameters of modelling functions have been approximately optimized for predictions on the 2006 test data. We therefore repeat the process again, combining the training and test data to form a larger training dataset for the period $1999 - 2006$, and using the completely untouched 2007 validation data to evaluate performance.

```
> train <- rbind(train, test)
> robje1 <- elo(train, cSt)
> robje2 <- elo(train, cSt, kfac=krating)
> robje3 <- elo(train, cSt, kfac=kgames)
> robjf1 <- fide(train, cSt, history = TRUE)
> robjf2 <- fide(train, cSt, kv = c(15,20,30))
> robjg <- glicko(train, cSt, history = TRUE)
> robjs <- steph(train, cSt, history = TRUE)
> pre1 <- predict(robje1, valid); pre2 <- predict(robje2, valid)
> pre3 <- predict(robje3, valid); prf1 <- predict(robjf1, valid)
> prf2 <- predict(robjf2, valid)
> prg <- predict(robjg, valid); prs <- predict(robjs, valid)
> metrics(validS, cbind(pre1,pre2,pre3,prf1,prf2,prg,prs))

        bdev    mse     mae
prs  89.886  84.763  85.494
```

```
prg  90.021 84.902 85.375
pre3 90.030 84.904 85.715
pre2 90.042 84.921 85.692
prf2 90.046 84.927 85.948
pre1 90.131 85.007 85.837
prf1 90.366 85.265 86.426
```

With this additional data, Stephenson tends to get further ahead of Glicko. Stephenson gives a 2.48% improvement over Elo, while Glicko gives a 1.11% improvement over Elo.

Each object has a `ratings` component containing the current status of the updating algorithm, and by default players are listed in order of rating, from highest to lowest. The top ten players from the FIDE (1), Stephenson and Glicko objects can be shown as follows, selecting from the set of players who have played at least 25 games and have played at least once in 2006. The latter condition removes Garry Kasparov. Note that these are all relative rating systems, and therefore the mean of the overall ratings is dependent on the method of initialization used in any particular application. The number of games played is inaccurate here as they were essentially unknown in the initial `cSt` object. We use the `chessPlayers` dataset to identify the player names.

Elo Ratings (Jan 2007):

```
> re <- robjf1$ratings
> re <- re[re$Lag <= 11 & re$Games >= 25,-c(4:6,8:10)]
> PlayerN <- chessPlayers$Name[re$Player]
> row.names(re) <- 1:nrow(re)
> head(cbind(PlayerN, round(re,0)), 10)
```

	PlayerN	Player	Rating	Games	Lag
1	Anand, Viswanathan	21308	2761	341	2
2	Topalov, Veselin	2115	2756	348	2
3	Kramnik, Vladimir	41314	2736	300	2
4	Morozevich, Alexander	16709	2723	370	0
5	Leko, Peter	28823	2722	373	1
6	Ponomariov, Ruslan	6961	2716	350	1
7	Aronian, Levon	27271	2709	489	1
8	Ivanchuk, Vassily	9394	2707	553	0
9	Adams, Michael	7772	2705	499	4
10	Mamedyarov, Shakhriyar	36438	2705	547	1

Glicko Ratings (Jan 2007):

```
> rg <- robjg$ratings
> rg <- rg[rg$Lag <= 11 & rg$Games >= 25,-(5:7)]
> PlayerN <- chessPlayers$Name[rg$Player]
> row.names(rg) <- 1:nrow(rg)
> head(cbind(PlayerN, round(rg,0)), 10)
```

```
             PlayerN Player Rating Deviation Games Lag
1        Anand, Viswanathan  21308   2803        61   341   2
2          Topalov, Veselin   2115   2796        50   348   2
3         Kramnik, Vladimir  41314   2786        55   300   2
4     Morozevich, Alexander  16709   2778        48   370   0
5        Ponomariov, Ruslan   6961   2773        54   350   1
6               Leko, Peter  28823   2769        56   373   1
7              Polgar, Judit 23254   2760        66   287   2
8             Aronian, Levon 27271   2760        49   489   1
9    Mamedyarov, Shakhriyar  36438   2758        46   547   1
10         Radjabov, Teimour 36248   2756        50   466   2
```

Stephenson Ratings (Jan 2007):

```
> rs <- robjs$ratings
> rs <- rs[rs$Lag <= 11 & rs$Games >= 25,-(5:7)]
> PlayerN <- chessPlayers$Name[rs$Player]
> row.names(rs) <- 1:nrow(rs)
> top <- head(cbind(PlayerN, round(rs,0)), 10); top
```

```
             PlayerN Player Rating Deviation Games Lag
1        Anand, Viswanathan  21308   2759        65   341   2
2         Kramnik, Vladimir  41314   2757        61   300   2
3          Topalov, Veselin   2115   2756        59   348   2
4     Morozevich, Alexander  16709   2755        60   370   0
5        Ponomariov, Ruslan   6961   2751        60   350   1
6    Mamedyarov, Shakhriyar  36438   2750        59   547   1
7               Leko, Peter  28823   2741        61   373   1
8             Aronian, Levon 27271   2737        60   489   1
9          Radjabov, Teimour 36248   2731        61   466   2
10             Polgar, Judit 23254   2728        65   287   2
```

The ranking of both Glicko and Stephenson methods are similar, but in Stephenson the absolute ratings are lower. This is a direct consequence of the parameter `lambda`, which draws player's ratings towards their opponents and therefore prevents spread at both the high and low ends. Figure 1 shows this feature of the system, comparing Elo (the FIDE implementation), Glicko and Stephenson. Notice that the Stephenson denisity is more peaked than Glicko, so it acts more like Elo in the upper tail. When `lambda` is zero, the Glicko and Stephenson densities (not shown) are virtually identical. So `lambda` narrows the spread.

For comparison purposes, Table 1 shows FIDE ratings for the top fifteen players from January 2007 as archived on their website. The top ten players under both Glicko and Stephenson all appear in the top fifteen FIDE ratings table. Note that our implementation of FIDE Elo will not be the same as the actual FIDE ratings because of the different initialization procedures.

	Name	Rating
1	Topalov, Veselin	2783
2	Anand, Viswanathan	2779
3	Kramnik, Vladimir	2766
4	Mamedyarov, Shakhriyar	2754
5	Ivanchuk, Vassily	2750
6	Leko, Peter	2749
7	Aronian, Levon	2744
8	Morozevich, Alexander	2741
9	Adams, Michael	2735
10	Gelfand, Boris	2733
11	Radjabov, Teimour	2729
12	Svidler, Peter	2728
13	Polgar, Judit	2727
14	Ponomariov, Ruslan	2723
15	Navara, David	2719

Table 1: FIDE ratings for the top fifteen chess players, January 2007.

```
> hist(robjs, density=TRUE, lwd=3, ylim=c(0,0.004), xlim=c(1800,2800),
+   main = "Rating System Comparison")
> hist(robjg, density=TRUE, lwd=3, lty=2, col=2, add=TRUE)
> hist(robjf1, density=TRUE, lwd=3, lty=3, col=3, add=TRUE)
> legend(2400,0.003, c("Stephenson","Glicko","Elo"), lty=1:3,
+   col=1:3, lwd=3, cex=1.1)
```

The role of `cval` in Glicko is to increase the rating deviations over time. In Stephenson this role is shared by `cval` and `hval`, and so `cval` should typically be lower in Stephenson than the corresponding parameter in Glicko. The `hval` parameter appears to make little or no difference to the overall density of the ratings, but typically improves predictive performance.

4 Producing Plots

The are two plotting methods for visualizing `"rating"` objects. The S3 method function `hist` will plot a histogram or density estimate of the player ratings. It can also plot other features of the current status, selectable by the argument `which`. If the full history of ratings for each time period is retained in the object, then `hist` can produce a series of histograms. The following produces (not shown) 96 histograms, one for each month, prompting the user between each display. By default, players are only depicted on histograms if they have played 15 games or more.

```
> hist(robjs, history=TRUE, xlim = c(1900,2900))
```

The S3 method function `plot` can only be used if the full history of ratings has been retained. It plots line traces across time of estimated ratings or other features for a
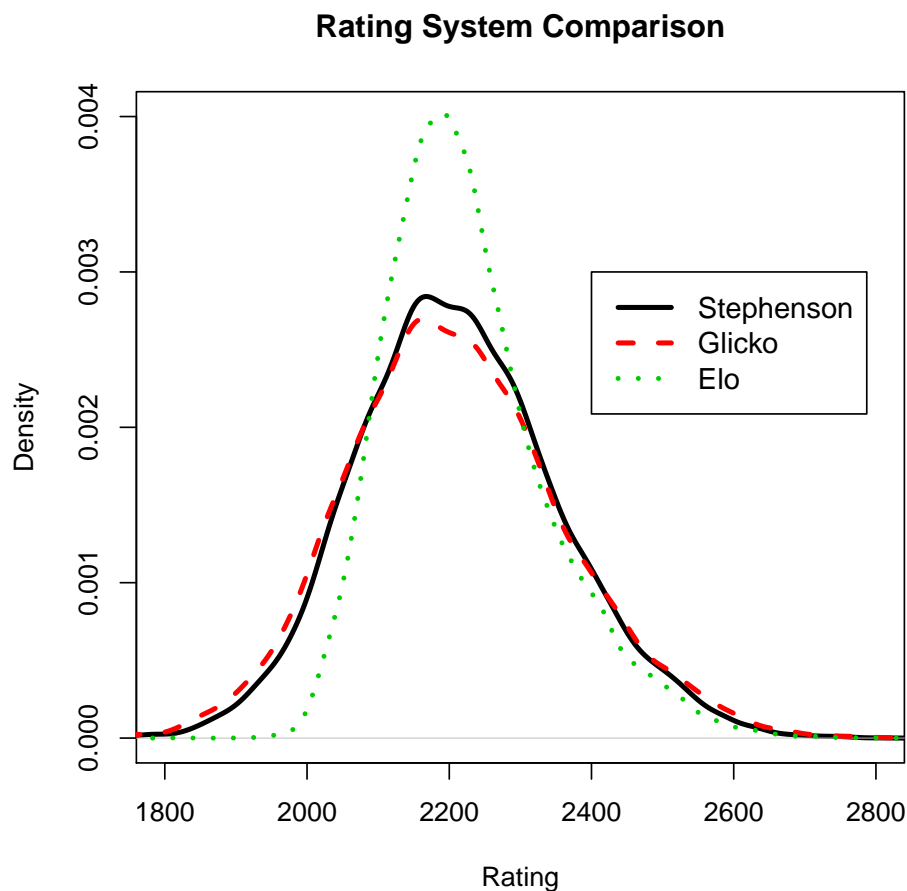
Rating System Comparison

Figure 1: A comparison of ratings distributions.

selected set of players. By default, active players are selected, and therefore these players may be more likely to improve than the general population. Figures 2 and 3 are plotted as follows. The first uses a default selection of the most active players in January 2001, whereas the second selects the 'current' (i.e. at the end of the year 2006) top ten players as identified previously.

```
> tv <- seq(2001, 2007, 1/12)[-73]
> plot(robjs, t0=25, lwd=2, tv=tv, xlab="Year")
> plot(robjs, players = top$Player, t0=25, lwd=2, tv=tv, xlab="Year")
> legend(2004, 2630, chessPlayers$Name[top$Player], lty=1:5,
+    col=1:6, lwd=3, cex=0.9)
```

The function `plot` can also analyse ratings inflation by setting the `inflation` argument to `TRUE`. The mean rating of the top `np` players at any given time point is then plotted. The example below shows the progression in the mean rating for the top 100 players, comparing the FIDE implementation of Elo with Glicko and Stephenson. System initialization was performed in 1999 using FIDE ratings for all systems, and we therefore plot from 2001 to ensure that the systems have had time to stabilize. There does not appear to be any evidence of ratings inflation for the top 100 players in this time period under Elo and Stephenson, but there is some suggestion of ratings inflation for Glicko.

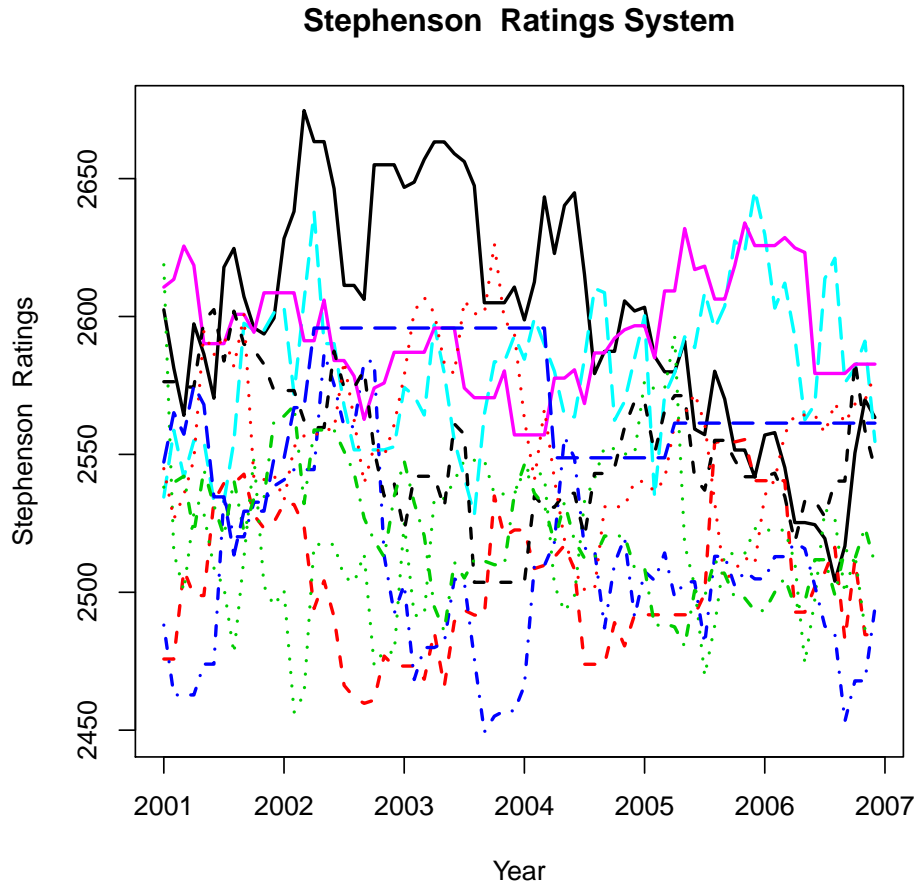## Stephenson Ratings System

Figure 2: Ratings over time for 10 active players.

```
> tv <- seq(2001,2007,1/12)[-73]
> plot(robjs, t0=25, lwd=2, tv=tv, xlab="Year", ylim = c(2630,2690),
+   inflation=TRUE, np = 100)
> plot(robjg, t0=25, lwd=2, tv=tv, lty=2, col=2, inflation=TRUE,
+   add=TRUE, np = 100)
> plot(robjf1, t0=25, lwd=2, tv=tv, lty=3, col=3, inflation=TRUE,
+   add=TRUE, np = 100)
> legend(2001,2690, c("Stephenson","Glicko","Elo"), lty=1:3,
+   col=1:3, lwd=3, cex=1)
```

Appendix: Stephenson System

Suppose that at the beginning of the ith month a player has a rating r and a variance v. After the ith month, these values need to be updated.

Step 1: Increase the variance of each player using $v = v + ct$ where c is a value to be decided and $t > 0$ is the number of periods since the player last competed.

Step 2: Let (r^*, v^*) be the player's rating and variance at the beginning of the $(i + 1)$th month. Then, with $q = \ln(10)/400$, the updating formulas are given as follows, where
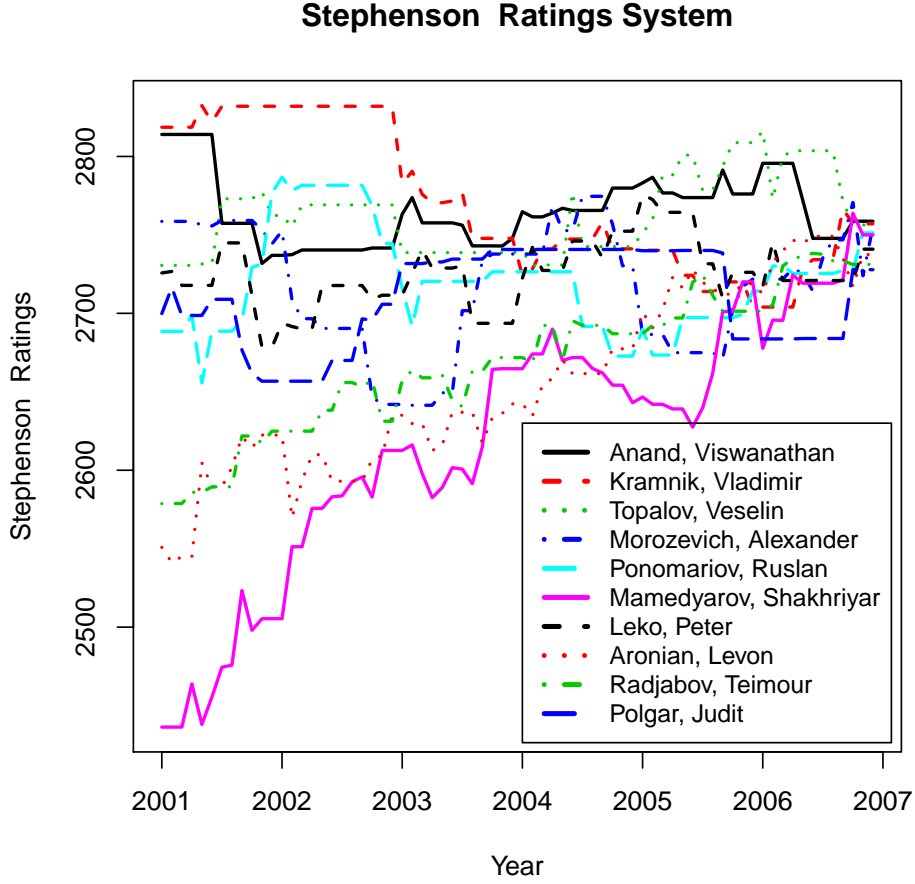
10

Stephenson Ratings System

Figure 3: Ratings over time for the 'current' (Jan 2007) top 10 players.

(r_j, v_j) for $j = 1, \ldots, m$ are the ratings and variances at the beginning of month i of the player's opponents in the $m > 0$ games that the player plays in that month, and where s_j are the scores in those games. Let $\bar{r} = (\sum_j r_j)/m$ and let w_j be a colour indicator with $w_j = 1$ if the the player is white, $w_j = -1$ if the player is black, and $w_j = 0$ if this is unknown.

$$
v^* = \left(\frac{1}{v + hm} + d \right)^{-1}
$$

$$
r^* = r + qv^* \sum_{j=1}^{m} k_j(s_j - e_j + b) + \lambda(\bar{r} - r)
$$

where

$$
k_j = \frac{1}{\sqrt{1 + 3q^2 v_j/\pi^2}}
$$

$$
e_j = \frac{1}{1 + 10^{-k_j(r - r_j + \gamma w_j)/400}}
$$

$$
d = q^2 \sum_{j=1}^{m} k_j^2 e_j(1 - e_j)
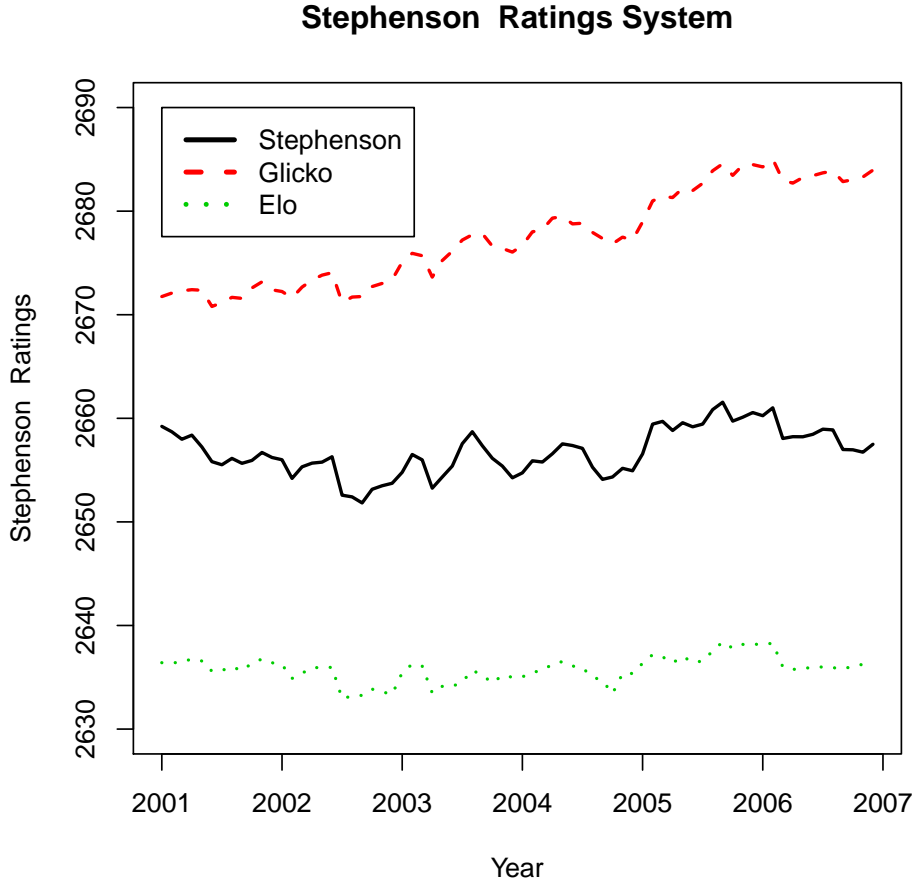$$

11

Stephenson Ratings System

Figure 4: Average ratings over time for top 100 players in any given time period.

Prediction

If player a playing white with current rating vector (r_a, v_a) has a game against player b playing black with current rating vector (r_b, v_b), and γ is a white advantage parameter, then the predicted score is given by

$$e_{ab} = \frac{1}{1 + 10^{-k_{ab}(r_a - r_b + \gamma)/400}},$$

where

$$k_{ab} = \frac{1}{\sqrt{1 + 3q^2(v_a + v_b)/\pi^2}}.$$

Note that the γ used in prediction is not necessarily the same as the γ used for constructing the ratings. For chess data, accounting for the advantage of white is important in prediction but appears to be of little importance for ratings construction.

R Function

In the R function `steph`, the argument `gamma` is γ, `cval` is $\sqrt{c}$, `hval` is $\sqrt{h}$, `bval` is $100b$ and `lambda` is 100λ. We use the same terminology as Glicko, so the player rating deviations are the standard deviations of the ratings given by $\sqrt{v}$. In Step 1 above we

12

impose a ceiling of 350 on the deviations. This is not necessary but is done to ensure that Stephenson contains Glicko as a special case, so that `steph` reproduces `glicko` upon setting $h = b = \lambda = 0$. The R function `predict` has an argument `gamma` so that different γ values can be used for constructing the ratings and for obtaining predictions.

Bibliography

Elo, A. (1978) *The Rating of Chessplayers, Past and Present.* Arco. ISBN 0-668-04721-6

Glickman, M. E. (1999) Parameter estimation in large dynamic paired comparison experiments. *Applied Statistics*, **48**, 377–394.