Rating Australian Rules Football Teams With The **PlayerRatings** Package

Now updated for glicko-2

Alec Stephenson

February 18, 2019

Summary

This vignette presents a short example of the use of **PlayerRatings**, using a small dataset to demonstrate rating Australian football teams and predicting the winner of future games based on those ratings. A second more detailed analysis using a large dataset of chess matches is given in the file `doc/ChessRatings.pdf`.

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 football data, we only use information on wins, draws and losses, so a value of one represents a win for the home team, a value of zero represents a win for the away team, and a value of one half 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 functions `glicko` and `glicko2` implement the Glicko (Glickman, 1999) and Glicko-2 (Glickman, 2001) systems, and the function `steph` implements the Stephenson system as detailed in the appendix of `doc/ChessRatings.pdf`. We only use the `steph` and `glicko2` functions in this vignette.

2 Modelling and Prediction

The `aflodds` dataset includes the results of Australian football games played from 26th March 2009 until 24th June 2012. We use the 2009 and 2010 games for our training data, the 2011 games for our test data and the 2012 data (which represents only the first half of the 2012 season) as our validation data. For the game results we will only use win, loss or draw information, ignoring the size of any victory.

```
> library(PlayerRatings)
> afl <- aflodds[,c(2,3,4,7)]
> train <- afl[afl$Week < 100,]
> test <- afl[afl$Week >= 100 & afl$Week < 150,]
> valid <- afl[afl$Week >= 150,]
> head(train,12)
```

```
   Week            HomeTeam                  AwayTeam Score
1     1     Richmond Tigers             Carlton Blues     0
2     1      Hawthorn Hawks              Geelong Cats     0
3     1 Collingwood Magpies            Adelaide Crows     0
4     1      Brisbane Lions         West Coast Eagles     1
5     1     St Kilda Saints              Sydney Swans     1
6     1    Melbourne Demons North Melbourne Kangaroos     0
7     1 Port Adelaide Power          Essendon Bombers     1
8     1   Fremantle Dockers          Western Bulldogs     0
9     2      Adelaide Crows           St Kilda Saints     0
10    2        Geelong Cats           Richmond Tigers     1
11    2 Collingwood Magpies          Melbourne Demons     1
12    2       Carlton Blues            Brisbane Lions     1
```

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 `steph` to iteratively update the team ratings once every round 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.

```
> sobj <- steph(train[train$Week==1,])
> for(i in 2:80) sobj <- steph(train[train$Week==i,], sobj$ratings)
```

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

```
> sobj <- steph(train, history = TRUE)
> sobj
```

```
Stephenson Ratings For 16 Players Playing 371 Games

                     Player Rating Deviation Games Win Draw Loss Lag
```

```
1         Collingwood Magpies   2405     76.57    51  36    2   13    0
2                Geelong Cats   2350     79.29    50  39    0   11    2
3             St Kilda Saints   2340     80.21    51  39    2   10    0
4           Western Bulldogs    2282     75.47    50  31    0   19    2
5                Sydney Swans   2240     75.43    46  22    0   24    3
6              Hawthorn Hawks   2227     76.18    45  21    1   23    4
7              Adelaide Crows   2205     76.69    46  24    0   22    5
8           Fremantle Dockers   2199     76.75    46  20    0   26    3
9  North Melbourne Kangaroos    2187     77.09    44  18    1   25    5
10               Carlton Blues  2185     75.83    46  24    0   22    4
11        Port Adelaide Power   2160     77.70    44  19    0   25    5
12             Brisbane Lions   2123     76.26    46  21    1   24    5
13           Essendon Bombers   2112     77.53    45  17    1   27    5
14           Melbourne Demons   2107     79.04    44  12    1   31    5
15            Richmond Tigers   2076     79.55    44  11    1   32    5
16          West Coast Eagles   2022     79.09    44  12    0   32    5
```

In either case, the resulting `sobj` object is identical. It gives the current (i.e. the end of 2010) rating for all 16 teams, and also gives a deviation parameter, which is an assessment of the accuracy of the rating. The deviation parameters are similar since all teams play roughly the same number of games. The lag parameter shows the number of weeks since each team has played; the two zero lags are associated with the two teams that played in the grand final of 2010. Unusually, the grand final of 2010 was drawn and was replayed the following week, and therefore no team has a lag value of one.

The following code uses the `plot` function to plot traces of the ratings across the 2009-2010 period for all 16 teams. We begin the period with no information, and therefore initially the rating changes are large. As the system learns about the teams the rating traces begin to stabilize. Flat lines denote the periods of inactivity that occur for teams not involved in the finals series, which takes place following the regular season.

```
> plot(sobj, npl=16)
> abline(v=c(27,55),lty=2,lwd=2,col="grey")
> text(c(14,42),c(2500,2500),c("2009","2010"),cex=1.5)
```

The `predict` function gives predictions of future matches, expressed as a value in the interval $[0, 1]$. In this vignette we use the argument `thresh` to instead produce binary values representing the predicted winner. This example predicts the results of round one in 2011 and compares the predictions to the actual outcomes. A new team was introduced in 2011; by default the prediction of matches involving new teams (less than 15 games) will be missing. We override this behaviour using the argument `trat`, which sets the parameters of new teams[1] for prediction purposes.

```
> test1 <- test[test$Week==min(test$Week),]
> pred <- predict(sobj, test1, trat = c(1900,300), thresh = 0.5)
> cbind(test1, Predict = pred)
```

[1]The new team did not play in round one and therefore in this particular case the argument makes no difference to the output.

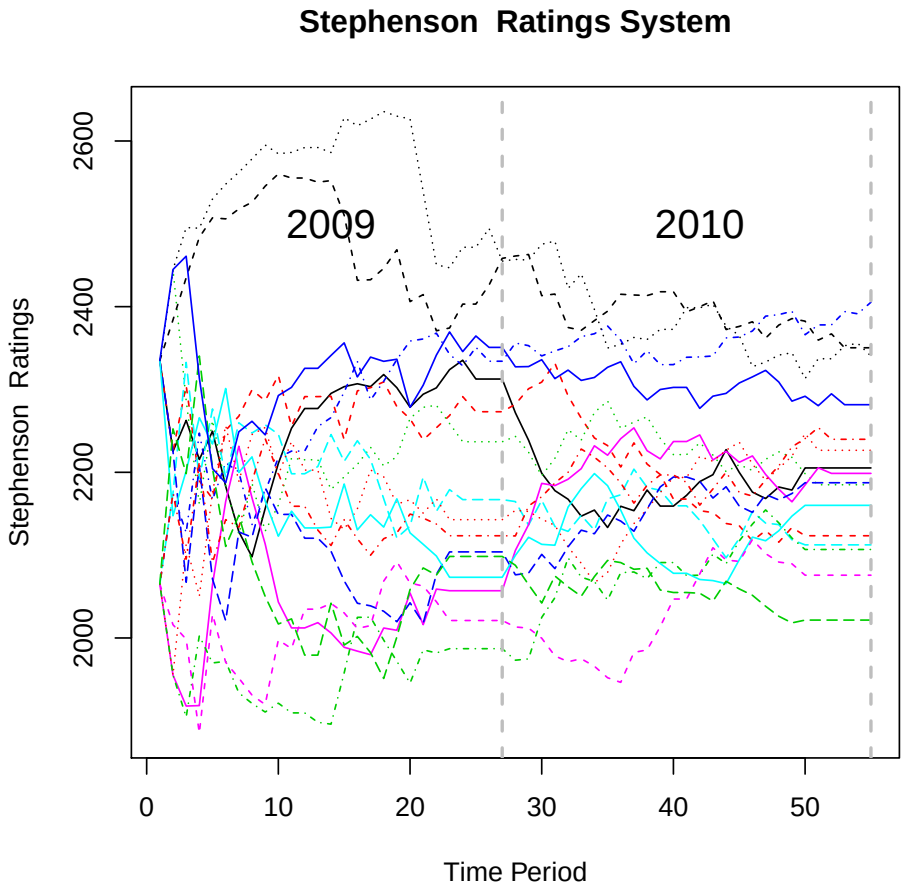

Figure 1: Plots of ratings traces for the 16 teams during 2009-2010, beginning with no information.

	Week	HomeTeam	AwayTeam	Score	Predict
372	105	Carlton Blues	Richmond Tigers	1.0	1
373	105	Geelong Cats	St Kilda Saints	1.0	1
374	105	Collingwood Magpies	Port Adelaide Power	1.0	1
375	105	Adelaide Crows	Hawthorn Hawks	1.0	1
376	105	Brisbane Lions	Fremantle Dockers	0.0	0
377	105	Essendon Bombers	Western Bulldogs	1.0	0
378	105	Melbourne Demons	Sydney Swans	0.5	0
379	105	West Coast Eagles	North Melbourne Kangaroos	1.0	0

We now combine the above code snippets in order to predict all games in the test set. We first run the system on the training data, and then loop through each round of the test set.

```
> sobj <- steph(train, init = c(2200,300), cval = 8,
+   hval = 8, lambda = 5)
> pred <- NULL
> for(i in unique(test$Week)) {
+   testi <- test[test$Week == i,]
+   predi <- predict(sobj, testi, trat = c(1900,300), gamma = 30,
```

```
+     thresh = 0.5)
+   pred <- c(pred, predi)
+   sobj <- steph(testi, sobj$ratings, init = c(2200,300), cval = 8,
+     hval = 8, lambda = 5)
+ }
> table(Result=test$Score, Predictions=pred)
```

```
      Predictions
Result  0  1
   0   53 31
   0.5  1  2
   1   21 88
```

We now make a couple of adjustments to the above. Firstly, we better account for new teams entering the system. In Australian football, the two new teams introduced in 2011 and 2012 were largely made up of younger players and were expected to me much weaker. To account for this, we create our own starting object `st0` to initialize the system, allowing the `init` argument to apply to the new teams only, and hence allowing us to account for this expected weakness.

Secondly, we focus on the `gamma` argument to `predict`, which accounts for the home team advantage. In Australian football teams are often from the same location or share the same ground, in which case the home advantage is likely to be zero. We can account for this, with a little work, by passing a vector to gamma. We first define a helper function which returns a logical vector to indicate whether the away team is travelling.

```
> trav <- function(dat) {
+   teams <- sort(unique(afl$HomeTeam))
+   locs <- c("Ade","Bri","Mel","Mel","Mel","Per","Gel","Bri","Syd",
+     "Mel","Mel","Mel","Ade","Mel","Mel","Syd","Per","Mel")
+   (locs[factor(dat$HomeTeam,levels=teams)]
+     != locs[factor(dat$AwayTeam,levels=teams)])
+ }
```

In the code below, we multiply our original `gamma` value by `trav(testi)` in order to specify a zero home advantage when the away team does not travel.

```
> st0 <- data.frame(Player=sort(unique(train$HomeTeam)), Rating=2200,
+   Deviation=300, stringsAsFactors=FALSE)
> sobj <- steph(train, st0, init = c(1900,300), cval = 8,
+   hval = 8, lambda = 5)
> pred <- NULL
> for(i in unique(test$Week)) {
+   testi <- test[test$Week == i,]
+   predi <- predict(sobj, testi, trat = c(1900,300),
+     gamma = 30*trav(testi), thresh = 0.5)
+   pred <- c(pred, predi)
```

```
+   sobj <- steph(testi, sobj$ratings, init = c(1900,300), cval = 8,
+     hval = 8, lambda = 5)
+ }
> rp <- table(Result=test$Score, Predictions=pred)
> rp

      Predictions
Result  0  1
   0   58 26
   0.5  1  2
   1   21 88

> round(100*(rp[1,2]+rp[nrow(rp),1])/sum(rp), 2)

[1] 23.98
```

The mis-classification percentage as given above (which counts draws as correctly classified) may be overly optimistic since we roughly chose our parameters to be optimal over the test data[2]. We therefore combine our training and test datasets to predict results on the validation data using the same parameters. In other words, we use the 2009-2011 results to predict the results in the first half of the 2012 season.

```
> st0 <- data.frame(Player=sort(unique(train$HomeTeam)), Rating=2200,
+   Deviation=300, stringsAsFactors=FALSE)
> sobj <- steph(rbind(train,test), st0, init = c(1900,300), cval = 8,
+   hval = 8, lambda = 5)
> pred <- NULL
> for(i in unique(valid$Week)) {
+   testi <- valid[valid$Week == i,]
+   predi <- predict(sobj, testi, trat = c(1900,300),
+     gamma = 30*trav(testi), thresh = 0.5)
+   pred <- c(pred, predi)
+   sobj <- steph(testi, sobj$ratings, init = c(1900,300), cval = 8,
+     hval = 8, lambda = 5)
+ }
> rp <- table(Result=valid$Score, Predictions=pred)
> rp

      Predictions
Result  0  1
     0 32 14
     1 16 46

> round(100*(rp[1,2]+rp[nrow(rp),1])/sum(rp), 2)
```

[2]The football dataset is much smaller and contains far less information than the chess dataset, and therefore different parameter combinations often yield similar predictions.

```
[1] 27.78
```

```
> sobj
```

```
Stephenson Ratings For 18 Players Playing 675 Games
```

	Player	Rating	Deviation	Games	Win	Draw	Loss	Lag
1	Collingwood Magpies	2284	66.74	88	68	2	18	0
2	Hawthorn Hawks	2241	65.47	82	48	1	33	1
3	West Coast Eagles	2230	66.20	81	39	0	42	0
4	Sydney Swans	2225	65.87	82	44	1	37	0
5	Geelong Cats	2223	66.90	87	68	0	19	0
6	Adelaide Crows	2199	66.78	80	40	0	40	0
7	Essendon Bombers	2197	66.91	80	37	2	41	0
8	St Kilda Saints	2175	65.86	86	57	3	26	1
9	Richmond Tigers	2171	66.71	78	25	2	51	1
10	Carlton Blues	2161	65.76	82	45	1	36	1
11	Fremantle Dockers	2160	66.36	80	35	0	45	0
12	North Melbourne Kangaroos	2157	67.09	78	34	1	43	0
13	Brisbane Lions	2132	67.35	80	30	1	49	0
14	Western Bulldogs	2131	66.66	84	45	0	39	0
15	Port Adelaide Power	2108	66.67	78	26	0	52	1
16	Melbourne Demons	2100	67.71	78	22	2	54	0
17	Gold Coast Suns	1962	79.88	34	3	0	31	1
18	Greater Western Sydney	1889	129.66	12	1	0	11	0

The code takes less than one-tenth of one second on my machine. We correctly predict 72.2% of the game results in the first half of 2012. We show above the current ratings as of 24th June 2012. We see that the two new teams (the lowest rated) have larger deviation values because they have played less games.

We finish by showing plots of the rating traces for the 16 established teams from mid-2010 to mid-2012. The rating trace plots require the full history of the process to be retained, which requires re-running the updates with the argument `history` set to `TRUE`. The current top eight teams are plotted first, with the remainder plotted second.

```
> sobj <- steph(rbind(train,test,valid), st0, init = c(1900,300), cval = 8,
+   hval = 8, lambda = 5, history = TRUE)
> p1 <- sobj$ratings[1:8,1]; p2 <- sobj$ratings[9:16,1]

> plot(sobj, t0 = 40, players = p1, ylim = c(2050,2350),lwd = 2)
> abline(v=c(55,83),lty=2,lwd=2,col="grey")
> legend(70,2160,p1,lty=1:5,col=1:6,lwd=3,cex=0.8)
> text(c(47,70,90),rep(2320,3),c("2010","2011","2012"),cex=1.5)

> plot(sobj, t0 = 40, players = p2, ylim = c(2050,2350),lwd = 2)
> abline(v=c(55,83),lty=2,lwd=2,col="grey")
> legend(68,2350,p2,lty=1:5,col=1:6,lwd=3,cex=0.8)
> text(c(47,70,90),rep(2070,3),c("2010","2011","2012"),cex=1.5)
```

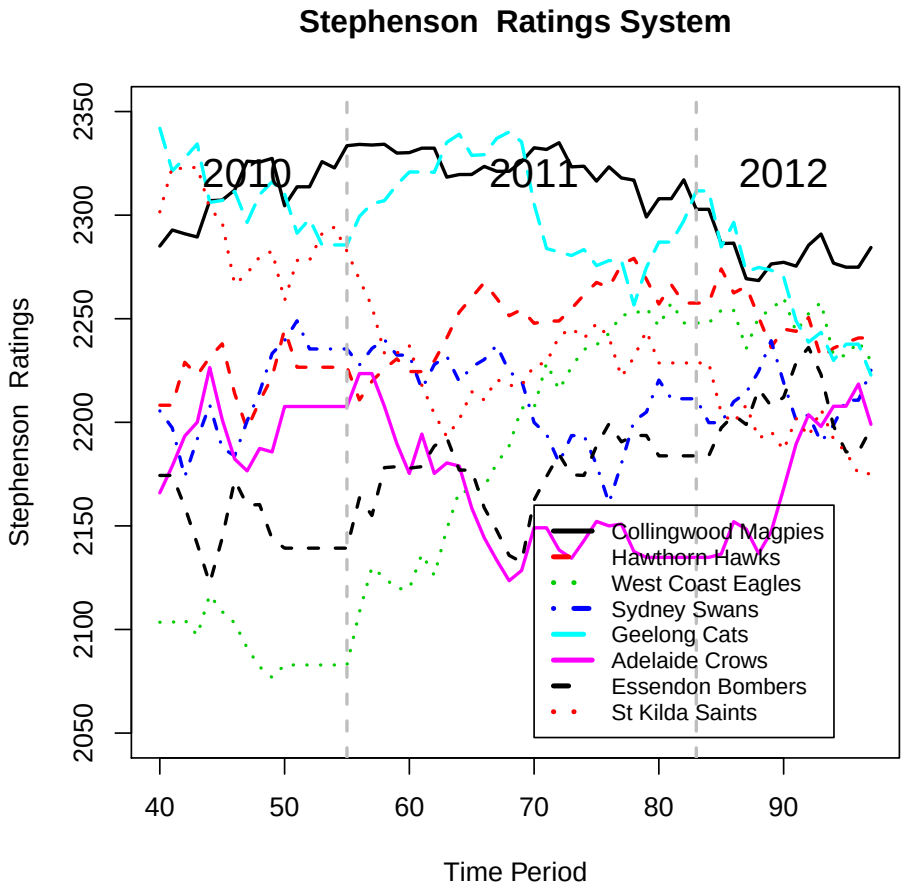

Figure 2: Plots of ratings traces for eight football teams from mid-2010 to mid-2012.

3 Glicko-2 Ratings

In the Glicko-2 rating system each team has a volatility parameter in addition to a deviation parameter. The calculation of the volatility requires a single parameter function optimization for each team within each time period, and will therefore be slower than Glicko or Stephenson.

```
> library(PlayerRatings)
> afl <- aflodds[,c(2,3,4,7)]
> train <- afl[afl$Week < 100,]
> test <- afl[afl$Week >= 100 & afl$Week < 150,]
> valid <- afl[afl$Week >= 150,]
> sobj <- glicko2(train, history = TRUE)
> print(sobj, cols=1:4)

Glicko-2 Ratings For 16 Players Playing 371 Games

                    Player Rating Deviation Volatility
1        Collingwood Magpies   2533    106.08     0.1484
2              Geelong Cats   2425    107.68     0.1501
3          St Kilda Saints   2412    108.86     0.1523
```

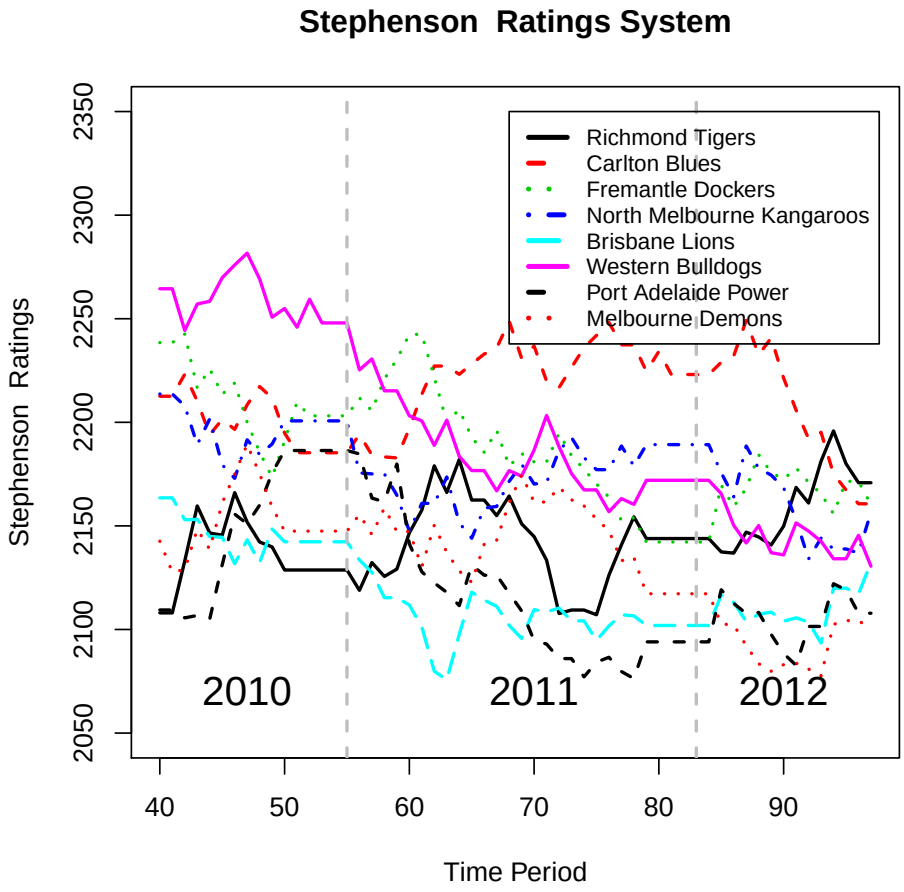

Figure 3: Plots of ratings traces for eight football teams during mid-2010 to mid-2012.

```
4            Western Bulldogs   2316    101.40     0.1487
5                Sydney Swans   2267    100.45     0.1496
6              Hawthorn Hawks   2244    102.56     0.1517
7              Adelaide Crows   2205    105.54     0.1512
8           Fremantle Dockers   2200    102.75     0.1519
9  North Melbourne Kangaroos   2187    103.50     0.1505
10              Carlton Blues   2163    102.15     0.1510
11       Port Adelaide Power   2157    105.76     0.1516
12          Melbourne Demons   2067    103.27     0.1505
13            Brisbane Lions   2062    102.89     0.1507
14          Essendon Bombers   2044    106.54     0.1538
15           Richmond Tigers   2022    106.86     0.1502
16         West Coast Eagles   1907    109.06     0.1508
```

The traces of the ratings for the Glicko-2 system are given below. Glicko-2 is primarily designed for situations where a player (or team) plays several games in any single time period. This is not the case here, and therefore the volatilities show little movement. This can be seen from plotting the volatility traces using `plot(sobj, npl=16, which = "Volatility")`.

```
> plot(sobj, npl=16)
```

```
> abline(v=c(27,55),lty=2,lwd=2,col="grey")
> text(c(14,42),c(2500,2500),c("2009","2010"),cex=1.5)
```

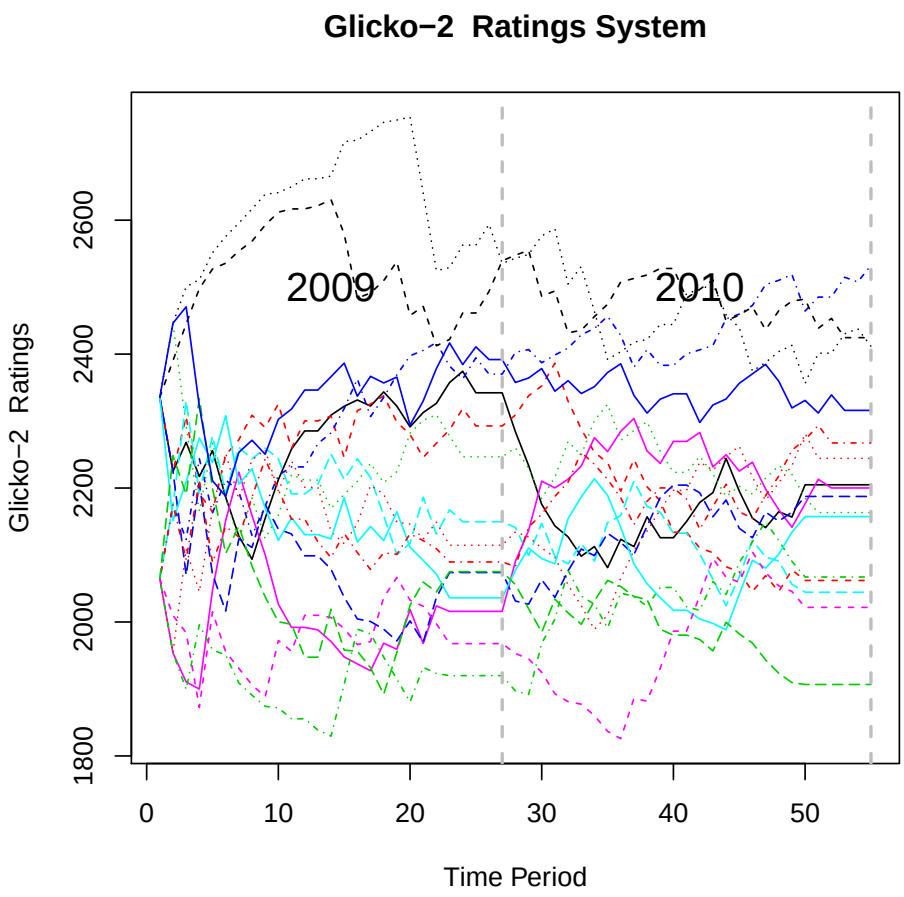

Figure 4: Plots of Glicko-2 ratings traces for the 16 teams during 2009-2010, beginning with no information.

The code in the previous section can be replicated for Glicko-2, with only minor alterations. The volatility parameter must be included in the status object and in the `init` vector. The Glicko-2 system parameter is called `tau`; smaller values of `tau` restrict the movement of the volatilities. If `tau` is zero or negative, then the volatilities are never updated. The code below provides an example.

```
> st0 <- data.frame(Player=sort(unique(train$HomeTeam)), Rating=2200,
+   Deviation=300, Volatility=0.15, stringsAsFactors=FALSE)
> sobj <- glicko2(train, st0, init = c(1900,300,0.15), tau = 1.2)
> pred <- NULL
> for(i in unique(test$Week)) {
+   testi <- test[test$Week == i,]
+   predi <- predict(sobj, testi, trat = c(1900,300),
+     gamma = 30*trav(testi), thresh = 0.5)
+   pred <- c(pred, predi)
+   sobj <- glicko2(testi, sobj$ratings, init = c(1900,300,0.15),
+     tau = 1.2)
```

```
+ }
> rp <- table(Result=test$Score, Predictions=pred)
> rp

      Predictions
Result  0  1
   0   62 22
   0.5  1  2
   1   27 82

> round(100*(rp[1,2]+rp[nrow(rp),1])/sum(rp), 2)

[1] 25
```

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.

Glickman, M.E. (2001) Dynamic paired comparison models with stochastic variances. *Journal of Applied Statistics*, **28**, 673-689.