Statistica Applicata - Italian Journal of Applied Statistics Vol. 35 (1)
doi.org/10.26398/IJAS.0035-005

welo: AN R PACKAGE FOR WEIGHTED AND STANDARD ELO RATES

Vincenzo Candila[1]
Department of Economics and Statistics, University of Salerno, Fisciano, Italy

Abstract *This paper describes the characteristics of the* `welo` *package, dedicated to calculating the weighted and unweighted (or standard) Elo rates in tennis. The Elo rates are one of the most accurate proxies of the strength of players/teams. In the standard version, the Elo rates are dynamically obtained using the outcome of the two players. In the recent paper of Angelini et al. (2022), the weighted version of the Elo rates (labeled as WElo) has been proposed in order to take into account not only the outcome of the matches but also the scoreline. The present work illustrates the main features of the R package, which allows the user to easily and quickly obtain the WElo and Elo rates, as well as the predicted probabilities of winning.*

Keywords: *Elo rates, Weighted Elo rates, R, Tennis, Betting.*

1. Introduction

The attention of the literature on sport's outcome forecasting has largely increased over the last few years. Many contributions focus on soccer (see, for instance, Angelini and De Angelis, 2017; Koopman and Lit, 2015; Mattera, 2021, among others) and tennis (see Arcagni et al., 2022; Lisi and Zanella, 2017, and references therein). Recently, Kovalchik (2020) has improved the Elo rates for tennis by taking into account, for the first time, the margin of victory. The Elo rates were proposed by the Physics Professor Arpad Elo in 1978 (Elo, 1978) for the rating of chess players. Since then, the Elo rates have been applied in a variety of sports: rugby (Carbone et al., 2016), soccer (Hvattum and Arntzen, 2010; Leitner et al., 2010), American football (Ryall and Bedford, 2010), and tennis (Kovalchik, 2016; Kovalchik and Reid, 2019). Angelini et al. (2022) have further extended the Elo-based models in tennis by weighting the Elo rates according to the number of games or sets won by each player. If the standard Elo rates take only into account the outcome of the match (that is, if a player has won or lost), the recently proposed Weighted Elo (WElo) rates of Angelini et al. (2022) instead are also based

[1]vcandila@unisa.it

on the final scoreline. This additional feature has provided a large benefit in using the WElo rates to calculate the probability of winning, compared to a set of the competing models[2]. The present paper aims at illustrating, in detail, all the tools of the `welo` package, which currently is available on the Comprehensive R Archive Network (CRAN).

There are several R packages on the CRAN and GitHub[3] repositories dealing with the Elo rating systems. But none of the available packages is suitable for calculating the WElo rates as the `welo` package. Moreover, the `welo` package allows the user to directly download tennis data using the http://www.tennis-data.co.uk/ site, which is weekly updated. The `welo` package can also easily plot the WElo and Elo rates, and it is flexible to include specific and user-based weights to some match conditions (for instance, if the match is Grand Slam match or it is played on a given surface). Another feature is the setting of the scale factor (more details will be provided in the next section), which is used to define how much the rate changes after the end of the match. Finally, the `welo` package also calculates the profits and losses deriving from a set of betting strategies. In what follows, we describe the main features of existing R packages dealing with the Elo rates. These information are then synthesized in Table 1.

Package `elo` (Heinzen, 2022) is on CRAN since 2017. It allows the calculation of the Elo rates both for team and non-team sports via its function `elo.run`, which is very flexible because it only requires the indication of the points of the two contendents. However, it does not include the possibility of taking into account the past scoreline to predict future winning probabilities. It neither allows to weight differently specific matches (for instance, the tennis matches played on a particular surface).

Package `EloRating` (Neumann and Kulik, 2020) is devoted to quantify animal dominance hierarchies. However, the main function providing the Elo rates, labeled `fastelo`, could also be used for non-animals data. In particular, it is sufficient to include as inputs in `fastelo` the names of the winners and losers, match by match. On the other side, `EloRating` package can not consider the scoreline of the last matches or specific match conditions. Moreover, it does not allow for a dynamic choice for the scale factor.

Package `EloOptimized` (Feldblum et al., 2021) has the maximum likelihood

[2]The set of competing models used in Angelini et al. (2022) is: the standard Elo model, the Bradly-Terry type model (McHale and Morton, 2011), the logit and probit regressions of Klaassen and Magnus (2003) and Del Corral and Prieto-Rodriguez (2010), respectively.

[3]GitHub hosts freely R packages, many of which are under development before being published on the CRAN.

estimation of the scale factor as the main feature. In particular, such a scale factor is not fixed by the user (even though there is also this possibility), but it is estimated by maximizing the likelihood of the sigmoid probability function as defined by Foerster et al. (2016). In addition, following the same maximum likelihood procedure, `EloOptimized` can also estimate the initial Elo rates.

Package `EloChoice` (Neumann, 2019) calculates the Elo rates through the `elochoice` function. However, the scale factor is fixed and there are no possibilities of setting different weights according to specific match conditions.

Package `comperank` (Chasnovski, 2020a) offers a variety of ranking and rating based on competition methods. Among these methods, the user can obtain the Elo rates via the `elo` function. One of the advantages of the `elo` function is the possibility of having ties. But, on the other side, the `comparank` package requires a specific format of the matches' data, making use of the `as_longcr` function of the `comperes` package (Chasnovski, 2020b). Also for this package, the resulting Elo rates consider a fixed scale factor and do not take into account the match conditions.

There are at least three packages dealing with the Elo rates on GitHub: `elomov`, `mELO` and `bwsTools`. The package `elomov` implements the Elo rates with the margin of victory option, as recently proposed by Kovalchik (2020). At the time of this writing, the whole installation of the `elomov` package via GitHub does not work. However, it is possible to manually install the functions of the package. The package `mELO`, using the `ELO` function, also admits ties, but the resulting Elo rates are based on a fixed scale factor. Finally, the package `bwsTools` allows for the calculation of the Elo rates via `elo` function. The `bwsTools` package does not allow for a time-varying scale factor or for a different rate according to specific match conditions.

Finally, none of the previously cited packages implements betting functions.

The rest of the paper is as follows. Section 2 illustrates how to compute the WElo and Elo rates. Section 3 presents the details of the `welo` package for computing the WElo and Elo rates. Section 4 is devoted to the betting application via the `welo` package. Conclusions follow.

2. Weighted and standard Elo rates

Throughout all the work, we use the same notation of Angelini et al. (2022). Therefore, i and j will indicate two opponents in a tennis match and $E_i(t)$ and $E_j(t)$ their Elo ratings for the match at time t. Then, the probability that player i

Table 1: R packages

Name	Repository	Weighted rates	Scale factor
`welo`	CRAN	Yes	Varying or fixed
`elo`	CRAN	No	Fixed
`EloRating`	CRAN	No	Fixed
`EloOptimized`	CRAN	No	Estimated
`EloChoice`	CRAN	No	Fixed
`comparank`	CRAN	No	Fixed
`elomov`	GitHub	Yes	Fixed
`mELO`	GitHub	No	Fixed
`bwsTools`	GitHub	No	Fixed

wins against player j in match t is:

$$\hat{p}_{i,j}(t) = \frac{1}{1 + 10^{\left(E_j(t) - E_i(t)\right)/400}}. \tag{1}$$

The formula updating the Elo ratings for player i is:

$$E_i(t+1) = E_i(t) + K_i(t)\left[W_i(t) - \hat{p}_{i,j}(t)\right], \tag{2}$$

where $W_i(t)$ represents an indicator function, which is one if player i wins match t and zero otherwise, and $K_i(t)$, as mentioned above, is a scale factor determining how much the Elo rate changes after match t. Such a scale factor is crucial in making effective the differences between the rates across players. It could be fixed to a given value (as many existing packages do). It could be estimated (as the `EloOptimized` package does). Or, as the `welo` package does, it could fixed, time-varying or even time-varying and, jointly, larger for some specific tournaments or surfaces.

The WElo rates, contrary to what happens for the Elo standard rates, allow for the consideration of the scoreline of the matches in the updating formula. More in detail, Eq. (2) incorporates an additional function $f(\cdot)$, depending on the number of games $G_{i,j}(t)$ or number of sets $S_{i,t}(t)$ won by players i and j during match t. When the WElo rates depend on the number of games $G_{i,j}(t)$, the rates (for player i) are defined as:

$$E^*_i(t+1) = E^*_i(t) + K_i(t)\left[W_i(t) - \hat{p}^*_{i,j}(t)\right] f(G_{i,j}(t)), \tag{3}$$

where $\hat{p}^*_{i,j}(t)$ is estimated using Eq. (1) but with $E_i(t)$ and $E_j(t)$ replaced by the corresponding WElo rates, labeled as $E^*_i(t)$ and $E^*_j(t)$, respectively. In Eq. (3), $f(G_{i,j}(t))$ is a function whose values depend on the games played in the previous match. In particular, $f(G_{i,j}(t))$ is defined as:

$$f(G_{i,j}(t)) = \begin{cases} \frac{NG_i(t)}{NG_i(t)+NG_j(t)} & \text{if player } i \text{ has won match } t; \\ \frac{NG_j(t)}{NG_i(t)+NG_j(t)} & \text{if player } i \text{ has lost match } t, \end{cases} \tag{4}$$

where $NG_i(t)$ and $NG_j(t)$ represent the number of games won by player i and player j in match t, respectively.

When the WElo rates depend on the number of sets, $f(S_{i,t}(t))$ is obtained as:

$$f(S_{i,j}(t)) = \begin{cases} \frac{NS_i(t)}{NS_i(t)+NS_j(t)} & \text{if player } i \text{ has won match } t; \\ \frac{NS_j(t)}{NS_i(t)+NS_j(t)} & \text{if player } i \text{ has lost match } t, \end{cases} \tag{5}$$

where $NS_i(t)$ and $NS_j(t)$ represent this time the sets won by player i and player j in match t, respectively. Then, $f(S_{i,t}(t))$ replaces $f(G_{i,j}(t))$ in Eq. (3).

3. WElo and Elo rates through the `welo` package

For ease of replicability, the interested user can reproduce all the following codes, once that the `welo` package has been installed from CRAN and loaded, that is:

```
R> install.packages("welo") # only the first time
R> library(welo)
```

The first step for using the `welo` package is the collection of tennis matches. By means of the `tennis_data` function, this step is immediately achieved:

```
R> db<-tennis_data("2021","ATP")
```

By the previous code, the `db` object includes all the matches played in 2021 for the Association of Tennis Professionals (ATP). If we are interested in female matches, then we can replace "ATP" by "WTA", where *WTA* stands for *Women Tennis Association*.

The second step for obtaining the WElo and Elo rates is cleaning the data. This operation is extremely delicate and is performed accurately through the `clean` function:

```
R> db_cleaned<-clean(db)
Number of matches (before cleaning) 2489
Number of matches (after cleaning) 1771
Number of players (before cleaning) 307
Number of players (after cleaning) 121
```

After running the `clean` function, some information automatically appear: the number of matches and players before and after the cleaning. More in detail, the `clean` function executes the following steps:

1. Remove all the uncompleted matches;
2. Remove all the NAs from B365 odds;
3. Remove all the NAs from the variable “ranking”, if any;
4. Remove all the NAs from the variable “games”, if any;
5. Remove all the NAs from the variable “sets”, if any;
6. Remove all the matches where the odds provided by the professional bookmaker Bet365 are equal, if any;
7. Define players i and j and their outcomes (Y_i and Y_j);
8. Remove all the matches of players who played less than the parameter of the `clean` function defined as `MNM`. By default, `MNM` $= 10$, which means that all the players playing less than 10 matches in `db` are excluded;
9. Remove all the matches of players with rank greater than the `MRANK` parameter. By default, `MRANK` $= 500$, which means that all the matches involving players rank above position 500 are excluded;
10. Sort the matches by date.

Changing the optional parameters of the `clean` function will return different cleaned datasets. For instance, if the interest is in the top-100 players playing at least one match, then the code will be:

```
R> db_cleaned_top_100<-clean(db, MNM=1, MRANK=100)
Number of matches (before cleaning) 2489
Number of matches (after cleaning) 1386
Number of players (before cleaning) 307
Number of players (after cleaning) 116
```

Finally, the `clean` function configures the dataset to be ready for the core function of the `welo` package, that is `welofit`. This is done by adding the columns of NG_i, NG_j, NS_i, NS_i, $f(G_{i,j}(t))$ and $f(S_{i,j}(t))$ to the cleaned `db`.

As mentioned above, the most important function of the `welo` package is the `welofit` function, which is very flexible and has several options. By default, it calculates the WElo and Elo rates with the following code:

```
R> res<-welofit(db_clean)
        Brier  Log-Loss
WElo  0.2274     0.6451
Elo   0.2325     0.6581
```

As for the `clean` function, also the `welofit` function automatically synthesizes some information in the console after the execution. In this case, the user can quickly verify if the WElo performs better or worse than the standard Elo rates, according to the Brier (Brier, 1950) and Log-Loss (used by Kovalchik, 2016, among others) loss functions. These two loss functions map the distance between the predicted probability and the actual outcome of all the matches. The smaller the loss function is, the better that model is. By default, the WElo and Elo rates are calculated using the time-varying scale factor reported in Kovalchik (2016), that is:

$$K_i(t) = \frac{250}{(N_i(t)+5)^{0.4}}, \tag{6}$$

where $N_i(t)$ represents the number of matches of player i at time t. This configuration increases the variation of the Elo and WElo ratings if player i has played few matches and vice versa.

Finally, the default setting of the `welo` function considers the scores of the games (see Eq. (4)) for the WElo rates, the starting points fixed to 1500, while the standard errors are not estimated.

Let us now focus on the resulting object of the `welo` function, which, in this case, has been called `res`. This object is a ‘welo’ object, which is a list containing the following components:

```
R> class(res)
[1] "welo"
R> names(res)
[1] "results" "matches" "period" "loss" "highest_welo"
[6] "highest_elo" "dataset"
```

The previous components are:

1. results: The data.frame including a variety of variables, among which there are the estimated WElo and Elo rates, before and after the match t, for players i and j, the probability of winning the match for player i (labeled as `WElo_pi_hat` and `Elo_pi_hat`, for the probabilities obtained from the WElo and Elo models, respectively).

2. matches: The number of matches analyzed.

3. period: The sample period considered.

4. loss: The Brier score and log-loss averages.

5. highest_welo: The player with the highest WElo rate and the correspondent date.

6. highest_elo: The player with the highest Elo rate and the correspondent date.

7. dataset: The dataset used for the estimation of the WElo and Elo rates.

The `welo` function allows for a variety of options. Firstly, the WELo rates can be calculated using the sets instead of the games. This is can be easily achieved through:

```
R> res_s<-welofit(db_clean,W="SETS")
       Brier  Log-Loss
WElo  0.2301    0.6521
Elo   0.2325    0.6581
```

Unsurprisingly, the (smaller) information content included in the sets, with respect to the games, worsens the WElo performance.

Moreover, the user can change the starting values of the WElo and Elo rates setting the parameter `SP` to another option. For instance, if the user wants the starting values equal to 1000 (instead of 1500, which is the default value), it is sufficient to run the following code:

```
R> res_1000<-welofit(db_clean,SP=1000)
       Brier  Log-Loss
WElo  0.2274    0.6451
Elo   0.2325    0.6581
```

Unexpectedly, it can be noted that the performances of the WElo and Elo models, when the starting values are set to 1000, are the same as the case with the starting points equal to 1500.

Another interesting feature of the `welo` function is flexibility of the scale factor $K_i(t)$ (and $K_j(t)$). By default, the scale factor is time-varying, according to Eq. (6). But such a parameter could be easily changed to be constant. For instance, if the user wants a constant scale factor of 100, the code will be:

```
R> res_K_100<-welofit(db_clean,K=100)
       Brier  Log-Loss
WElo  0.2274    0.6450
Elo   0.2340    0.6619
```

In this case, the better performance of the WElo model appears even more evident. Another possibility is to set K such that more weight is given to specific tournaments or match surfaces. Currently, four options are available: “Grand_Slam”, “Surface_Hard”, “Surface_Clay” and “Surface_Grass”. Each of the previous options increases the time-varying scale factor in (6) by 1.1 if the match is a Grand Slam match, is played on hard, clay, or grass, respectively. For instance, if the user wants to calculate the WElo and Elo rates giving more emphasis on the Grand Slam matches, then the code will be:

```
R> res_gs<-welofit(db_clean,K="Grand_Slam")
       Brier  Log-Loss
WElo  0.2272    0.6447
Elo   0.2325    0.6584
```

Another peculiar feature of the `welofit` function is the calculation of the standard errors for the WElo and Elo rates, according to the procedure suggested by Angelini et al. (2022). The code will be:

```
R> res_ci<-welofit(db_clean,CI=TRUE)
       Brier  Log-Loss
WElo  0.2274    0.6451
Elo   0.2325    0.6581
```

The resulting Brier and Log-Loss averages are exactly the same of `res`. This is because the setting parameters are unchanged. But, this time, the “results” component of `res_ci` includes also the lower (labeled with the suffix “_lb”) and upper (labeled with the suffix “_ub”) bootstrap confidence intervals. The confidence intervals are obtained according to the procedure illustrated in Angelini et al. 2022 (see their Section 2.1). By default, the bootstrap confidence intervals are obtained

Table 2: Grand Slam 2021 finals, WElo rates and standard errors

Grand Slam	Players	WElo	$\hat{p}_{i,j}(t)$	LB	UB
Australian	i) **Djokovic N.**	1704.187	0.512	1655.157	1750.961
Open	j) Medvedev D.	1696.004		1649.230	1745.034
Roland	i) **Djokovic N.**	1847.900	0.487	1816.258	1881.268
Garros	j) Tsitsipas S.	1857.125		1829.618	1883.210
Wimbledon	i) **Djokovic N.**	1894.685	0.632	1856.548	1916.848
	j) Berrettini M.	1800.394		1778.231	1838.531
US Open	i) Djokovic N.	1943.422	0.649	1906.715	1963.316
	j) **Medvedev D.**	1837.013		1818.781	1870.651

Note: Winning player is in **Bold**.

using a significance level `alpha` $= 0.05$ and a number of bootstrap replicates `B` $= 1000$. The WElo rates calculated before each Grand Slam 2021 final, together with the bootstrap standard errors and the probability that player i wins over player j (that is, $\hat{p}_{i,j}(t)$) are reported in Table 2.

One of the most interesting features of the `welo` package is the possibility of plotting the WElo and Elo rates in nice graphs. The plot can be obtained by the `welo_plot` function, whose only input required is the (character) vector of players. Being in a ggplot2 environment, the user can complete the plot by adding font size details via the `ggplot2::theme()` option. Suppose that the user wants the plot of the WELo rates for the following players: Nadal, Djokovic, Berrettini, and Sinner. Moreover, suppose that the user considers the rates from the `res` object previously obtained. Then, the code will be:

```
R> require(ggplot2)
R> players<-c("Nadal R.","Djokovic N.",
"Berrettini M.","Sinner J.")
R> welo_plot(res, players)+
ggplot2::theme(text = element_text(size = 20))
```

The output of the previous lines is in Figure 1(a). Figure 1(a) has some interesting peculiarities. First, at the end of 2021, Djokovic was largely the player with the highest WElo rate. Second, there is evidence of periods where some players did not play. These periods are highlighted in the plot with a horizontal line. For instance, during the second half of the 2021 season, Nadal played only two matches (in August, at the Washington Citi Open) after the defeat at the Roland Garros in June. This is the reason why Nadal's orange line is horizontal from mid-June to the end of 2021. By default, the WElo rates are considered. Changing the optional parameter `rates` of `welo_plot` from "WElo" to "Elo", the standard Elo rates depicted in Figure 1(b) are obtained. It is worth noting that the patterns of the WElo and Elo rates are very similar, even though the former are always smaller than the latter.

```
R> welo_plot(res,players,rates="Elo")+
ggplot2::theme(text = element_text(size = 20))
```

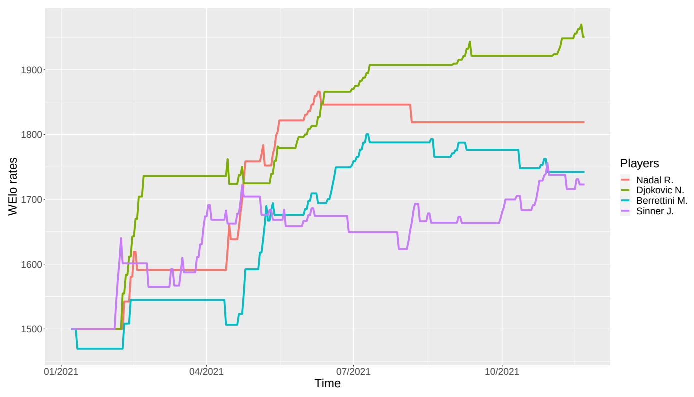
The `welo` package also provides a function to plot players' official (ATP or WTA) rank. The following code will plot (in Figure 1(c)) the official ranks of the four players already used in Figures 1(a) and 1(b):

```
R> rank_plot(res,players)+ ggplot2::theme(text =
element_text(size = 20))
```

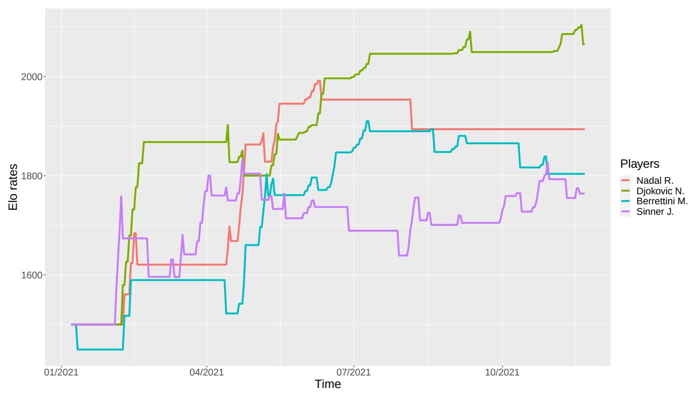
Some considerations arise looking at Figure 1. First, a necessary burn-in period is required to make the WElo and Elo rates reliable. This problem could be easily solved by enlarging the sample period. Second, even though at the end of sample, the WElo and Elo rates have the same order of the official ATP rank, the best player during Spring 2021 is Rafael Nadal for the WElo and Elo rates (in place of the official number one of the ATP rank, Novak Djokovic). Third, at the end of the 2021 season, mainly for the WElo rates, the young Italian player Jannik Sinner is pretty close to the other Italian tennis top player, Matteo Berrettini. This closeness between the two Italian players is not overall captured by the official ATP rank.

4. Betting with the `welo` package

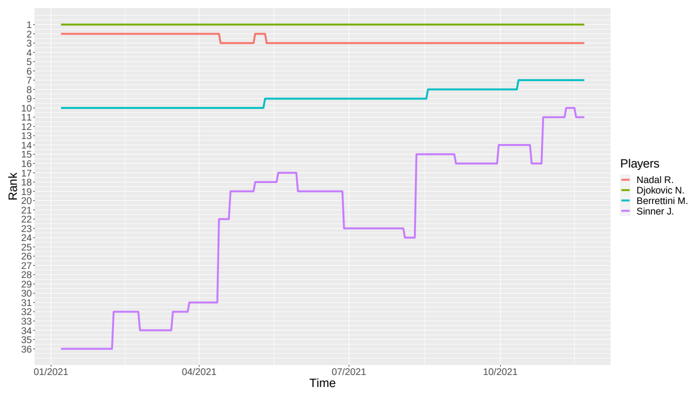
In the sports literature, one of the main aims of a forecasting model is to verify its performance from an economic point of view. This can be easily achieved through the `welo` package, thanks to the `betting` function. Such a function represents a novelty in the context of R packages dealing with Elo rates because none of the existing packages has similar functions at the time of this writing. The `betting` function requires four inputs to work: `x`, `r`, `q` and `model`. The first input `x` is a 'welo' object from the `welofit` function. The second and third inputs `r` and `q` are two thresholds that identify the matches on which place an amount of $1. More in detail, as suggested by Angelini et al. (2022), the bets are placed on the matches



(a) WElo rates



(b) Elo rates



(c) ATP Rank

Figure 1: Plots of the `welo` package

satisfying the following conditions:

$$\frac{\hat{P}_{i,j}(t)}{q_{i,j}(t)} > r \quad \text{and} \quad q_{i,j}(t) > q, \tag{7}$$

where $\hat{P}_{i,j}(t)$ are the two probabilities as resulting from the Elo and WElo models for the match between i and j at time t, that is, $\hat{P}_{i,j}(t) = \left\{\hat{p}_{i,j}(t), \hat{p}^{*}_{i,j}(t)\right\}$, and $q_{i,j}(t)$ is the inverse of the published odds for the same match, also named implied probability. For coverage reasons, the `welo` package considers the implied probabilities $q_{i,j}(t)$ offered by the professional bookmaker Bet365. The user can decide the model (WElo or Elo) originating the probabilities of winning via the fourth input of the `betting` function, that is setting `model` = "WELO" or `model` = "ELO".

In line with McHale and Morton (2011) and Dixon and Coles (1997), the threshold `r` is used to discriminate among matches on which place a bets or not. For instance, if `r` = 1, only matches whose predicted probabilities are greater than the implied probabilities are worthy of a bet. When `r` increases, fewer matches will be selected. The `betting` function allows also for the inclusion of a set of values for `r`. As concerns the threshold `q`, such a value is needed to exclude heavy underdogs. For instance, when `q` = 0.30, then all the matches whose Bet365 implied probabilities are smaller than 0.30 will be excluded. Bearing this in mind, a general configuration of the `betting` function could be:

```
R> res_bet_welo<-betting(res,r=seq(1,1.3,0.05),
q=0.3,model="WELO")
         r # Bets      ROI(%)       LCI       UCI
[1,] 1.00   1002 10.045908 3.4753003 16.22180
[2,] 1.05    823  8.883354 1.4252170 16.11634
[3,] 1.10    655 10.175573 1.8184030 18.80244
[4,] 1.15    533 10.553471 1.4241092 20.09728
[5,] 1.20    438 11.808219 1.1060541 22.28150
[6,] 1.25    338 13.239645 0.6448331 25.11509
[7,] 1.30    265 16.584906 1.6015294 31.78393
```

The predicted probabilities included in the 'welo' object labeled `res` are considered in the previous command. The resulting output of the `betting` function is a matrix that includes five columns. The first column reports the values of the threshold `r`. Hence, among the 1771 matches of the full dataset, the betting rule suggests of betting on 1002 matches, when `r` = 1 and `q` = 0.3. The second column includes the number of bets (for each correspondent threshold `r`). As mentioned above, the higher the threshold is, the smaller the number of matches to bet on is. The third column reports the Returns-on-Investment (ROI), in percentage. The last two columns show the lower (LCI) and upper (UCI) bootstrap confidence intervals, computed using the default number of bootstrap replicates (`R` $= 2000$) and the default significance level (`alpha` $= 0.1$). The user can easily change those two settings. In what follows, there is the code and the output for the Elo probabilities:

```
R> res_bet_elo<-betting(res,r=seq(1,1.3,0.05),
q=0.3,model="ELO")
         r # Bets    ROI(%)        LCI      UCI
[1,] 1.00    1096  6.914234  1.4332144 12.67208
[2,] 1.05     893  7.767077  1.0466995 14.56551
[3,] 1.10     713  8.830295  1.2234941 16.32349
[4,] 1.15     607  9.059308  0.2738958 17.10543
[5,] 1.20     506 11.373518  2.0207622 21.47579
[6,] 1.25     409  9.312958 -1.1881413 20.27323
[7,] 1.30     329 11.556231 -0.7741607 23.21550
```

Interestingly, it can be noted that the ROI(%) of the WElo probabilities are higher than the corresponding Elo probabilities, independently of the threshold `r` adopted. Moreover, all the ROI(%) of the WElo model are statistically significant, while the same does not happen for the ROI(%) of the Elo model.

Finally, the `betting` function has some optional parameters which could be set: `bets`, `R`, `alpha`, `start_oos`, and `end_oos`. The parameter `bets` identifies the type of bet used. By default, it is “Best_odds”, which means that the bets are placed using the best odds available among all the bookmakers. Alternative choices for `bets` are: “Avg_odds” and “B365_odds”. “Avg_odds” are the average odds among all the odds published by the professional bookmakers for the match under consideration and “B365_odds” are the Bet365 odds. The parameter `R` represents the number of bootstrap replicates to calculate the confidence intervals of the ROI(%). Its default value is 2000. The parameter `alpha` is the significance level for the boostrap confidence intervals. By default, `alpha` $= 0.1$. Eventually, the user could also bet on a specific time period. This is can be easily done setting the parameters `start_oos` and `end_oos`, which have to be formatted as “YYYY”. For instance, if the user is interested in the time interval from 2021 to 2022, then he/she has to format `start_oos` = “2021” and `end_oos` = “2022”.[4]

[4]The time interval from 2021 to 2022 would require a larger dataset including also data for 2022.

For comparison purposes, the welo package includes also another betting function, labeled random_betting. Such a function is useful when the user wants to evaluate if randomly betting on players i and j is a winning strategy with respect to the decision on the basis of the WElo and Elo probabilities. To make a fair comparison, random_betting shares almost all the inputs with the function betting: this means that the user can set the two functions similarly to select the same matches. As said before, in the case of random_betting, the players i and j are randomly selected. To reduce the impact of this randomness, the random_betting function repeats the random selection B times, which is the only (optional) parameter of the random_betting function not included in the betting function. By default, B = 10000. The resulting matrix reports the overall mean of the ROI (in percentage) across the B values for every threshold r used. The code will be:

```
R> res_rand_bet<-random_betting(res,r=seq(1,1.3,0.05),
q=0.3,model="WELO")
        r  # Bets    ROI(%)
[1,]  1.00   1002  2.589963
[2,]  1.05    823  2.907354
[3,]  1.10    655  3.340112
[4,]  1.15    533  2.776383
[5,]  1.20    438  3.015227
[6,]  1.25    338  3.906786
[7,]  1.30    265  4.292838
R> res_rand_bet<-random_betting(res,r=seq(1,1.3,0.05),
q=0.3,model="ELO")
        r  # Bets    ROI(%)
[1,]  1.00   1096  2.328365
[2,]  1.05    893  2.419626
[3,]  1.10    713  2.052510
[4,]  1.15    607  2.458521
[5,]  1.20    506  3.137259
[6,]  1.25    409  2.724166
[7,]  1.30    329  3.513878
```

From the last two R outputs, it can be noted that the random selection of players on which place a bet, even if repeated B times, does not yield larger ROI(%) with respect to the previous two ROI(%) obtained from the WElo and Elo rates.

5. Conclusions

The present contribution aimed at explaining the details of the `welo` package, an R package for the calculation of the standard (that is, unweighted) and weighted Elo (WElo) rates for tennis. The `welo` package has some interesting features: (i) the direct download of data for male and female professional tennis matches (via the `tennis_data` function); (ii) the cleaning of the tennis data (through the `clean` function); (iii) the calculation of standard and WElo rates by the core function of the package labeled `welofit`, with the possibility of weighting differently some tournaments and surfaces and having constant or time-varying scale factor; (iv) the plot of the resulting Elo and WElo rates with the `welo_plot` function; (v) the economic evaluation of the Returns-on-Investment (ROI) obtained from the predicted probabilities of the Elo and WElo rates, according to the betting rule of Angelini et al. (2022) and references therein; (vi) the comparison of the previous ROI with the ROI obtained from the random betting strategy.

The current paper can serve as a guide for practitioners and R users for the first time dealing with the calculation of the Elo and WElo rates. Further extensions of the `welo` package could enlarge the sports under consideration, like basket, volley and so forth.

References

Angelini, G., Candila, V., and De Angelis, L. (2022). Weighted Elo rating for tennis match predictions. In *European Journal of Operational Research*, 297 (1): 120–132.

Angelini, G. and De Angelis, L. (2017). PARX model for football match predictions. In *Journal of Forecasting*, 36 (7): 795–807.

Arcagni, A., Candila, V., and Grassi, R. (2022). A new model for predicting the winner in tennis based on the eigenvector centrality. In *Annals of Operations Research*, 1–18.

Brier, G.W. (1950). Verification of forecasts expressed in terms of probability. In *Monthly weather review*, 78 (1): 1–3.

Carbone, J., Corke, T., and Moisiadis, F. (2016). The rugby league prediction model: Using an Elo-based approach to predict the outcome of National Rugby League (NRL) matches. In *International Educational Scientific Research Journal*, 2 (5): 26–30.

Chasnovski, E. (2020a). *comperank: Ranking Methods for Competition Results*. URL `https://CRAN.R-project.org/package=comperank`. R package version 0.1.1.

Chasnovski, E. (2020b). *comperes: Manage Competition Results*. URL `https://CRAN.R-project.org/package=comperes`. R package version 0.2.5.

Del Corral, J. and Prieto-Rodriguez, J. (2010). Are differences in ranks good predictors for Grand Slam tennis matches? In *International Journal of Forecasting*, 26 (3): 551–563.

Dixon, M.J. and Coles, S.G. (1997). Modelling association football scores and inefficiencies in the football betting market. In *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 46 (2): 265–280.

Elo, A.E. (1978). *The rating of chessplayers, past and present*. Arco Pub.

Feldblum, J., Foerster, S., and Franz, M. (2021). *EloOptimized: Optimized Elo Rating Method for Obtaining Dominance Ranks*. URL `https://CRAN.R-project.org/package=EloOptimized`. R package version 0.3.1.

Foerster, S., Franz, M., Murray, C.M., Gilby, I.C., Feldblum, J.T., Walker, K.K., and Pusey, A.E. (2016). Chimpanzee females queue but males compete for social status. In *Scientific reports*, 6 (1): 1–11.

Heinzen, E. (2022). *elo: Ranking Teams by Elo Rating and Comparable Methods*. URL `https://CRAN.R-project.org/package=elo`. R package version 3.0.1.

Hvattum, L.M. and Arntzen, H. (2010). Using ELO ratings for match result prediction in association football. In *International Journal of Forecasting*, 26 (3): 460–470.

Klaassen, F.J. and Magnus, J.R. (2003). Forecasting the winner of a tennis match. In *European Journal of Operational Research*, 148 (2): 257–267.

Koopman, S.J. and Lit, R. (2015). A dynamic bivariate poisson model for analysing and forecasting match results in the English Premier League. In *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 178 (1): 167–186.

Kovalchik, S.A. (2016). Searching for the GOAT of tennis win prediction. In *Journal of Quantitative Analysis in Sports*, 12 (3): 127–138.

Kovalchik, S. (2020). Extension of the Elo rating system to margin of victory. In *International Journal of Forecasting*, 36: 1329–1341.

Kovalchik, S. and Reid, M. (2019). A calibration method with dynamic updates for within-match forecasting of wins in tennis. In *International Journal of Forecasting*, 35 (2): 756–766.

Leitner, C., Zeileis, A., and Hornik, K. (2010). Forecasting sports tournaments by ratings of (prob) abilities: A comparison for the EURO 2008. In *International Journal of Forecasting*, 26 (3): 471–481.

Lisi, F. and Zanella, G. (2017). Tennis betting: can statistics beat bookmakers? In *Electronic Journal of Applied Statistical Analysis*, 10 (3): 790–808.

Mattera, R. (2021). Forecasting binary outcomes in soccer. In *Annals of Operations Research*, 1–20.

McHale, I. and Morton, A. (2011). A Bradley-Terry type model for forecasting tennis match results. In *International Journal of Forecasting*, 27 (2): 619–630.

Neumann, C. (2019). *EloChoice: Preference Rating for Visual Stimuli Based on Elo Ratings*. URL `https://CRAN.R-project.org/package=EloChoice`. R package version 0.29.4.

Neumann, C. and Kulik, L. (2020). *EloRating: Animal Dominance Hierarchies by Elo Rating*. URL `https://CRAN.R-project.org/package=EloRating`. R package version 0.46.11.

Ryall, R. and Bedford, A. (2010). An optimized ratings-based model for forecasting Australian Rules football. In *International Journal of Forecasting*, 26 (3): 511–517.