`EloRating` - a brief tutorial

version 0.43

Christof Neumann & Lars Kulik

November 23, 2014

Contents

1 Preliminary remarks

The `EloRating` package is work in progress. If you have any criticism, suggestions or bugs to report, please let us know.

We describe here the main functionalities of the `EloRating` package.[1] Note that for the sake of this tutorial, we first present an example with the minimal amount of data required: a sequence of decided dominance interactions along with the dates[2] of these interactions. Even though the package is capable of dealing with undecided interactions (in fact the example file contains this information), we decided to omit this aspect for the sake of clarity in the first part (section 3). In addition, this first example is not linked to 'presence' data. In other words, here we assume that all individuals that occur in the data set were present over the entire study period. For the same example utilizing information about presence/absence of individuals and undecided interactions/draws see section 4.

The fictional data set presented here comprises 250 dominance interactions of 10 individuals.

2 Package installation and data preparation

To install the package, just use (given you have a working internet connection):

```
> install.packages("EloRating")
```

We assume that you store your data on dominance interactions in some sort of spreadsheet software. While it is possible to read data directly from Excel files (.xls or .xlsx) or SPSS files (.sav),[3] we suggest that you store your data in simple (tab-separated) text files. For example, from Excel this is possible via File>Save as... and then choosing "tab-delimited text file" as file format.[4]

3 Using `EloRating`

Start by loading the package and reading the raw data.[5]

```
> library(EloRating)
> xdata <- read.table(system.file("ex-sequence.txt", package = 'EloRating'),
                      header=T)
```

[1] Note that one additional package (`zoo`) has to be installed to make our package functional (e.g. by `install.packages("zoo")`)

[2] Dealing with calendar dates in `R` is prone to unexpected behaviour. We decided to stick to a specific format ("YYYY-MM-DD") and the functions assume that dates appear in this format in the objects from which the functions work.

[3] see the R packages `gdata`, `xlsx` and `foreign`

[4] you may also save your file as comma delimited or something similar, but note that you then may need to modify the arguments to `read.table()` or use `read.csv()`

[5] The example files are in the above described tab-delimited text format and can be found in the package directory. If you don't know where that is check `.libPaths()`

Keep in mind that as soon as you use your own data it might be nessary to include absolute paths with the filename.[6] For example:

```
> xdata <- read.table("c:\\temp\\ex-sequence.txt", header = TRUE,
    sep = "\t")
```

3.1 Data checks

We then go on and check whether the data meet the formatting requirements for the remaining functions of the package to work. If there is something appearing not quite right with your data, this function will tell you. "Warnings" can sometimes be ignored (see below), whereas "errors" need to be fixed before the next step. More details on the possible warning and error messages can be found in the help files (`?seqcheck`).

```
> seqcheck(winner=xdata$winner, loser=xdata$loser, Date=xdata$Date)

no presence data supplied
Everything seems to be fine with the interaction sequence...OK
```

3.2 Elo rating calculations

This doesn't give any error message, and so we can go on and calculate the actual Elo ratings and store the results of the calculations in an object we name `res`. Note that in order to ignore possible "warnings" from `seqcheck()` the argument `runcheck=FALSE` has to be set.

```
> res <- elo.seq(winner=xdata$winner, loser=xdata$loser, Date=xdata$Date,
                 runcheck=TRUE)

> summary(res)

Elo ratings from 10 individuals
total (mean/median) number of interactions: 250 (50/49)
range of interactions: 19 - 75
date range: 2000-01-01 - 2000-09-06
startvalue: 1000
uppon arrival treatment: average
k: 100
proportion of draws in the data set: 0
```

3.3 Extract Elo ratings

The most obvious task perhaps is to obtain Elo ratings of specific individuals on a specific date. This can be achieved by running the function `extract.elo()` on the object `res` that was just created. In the output, individuals are ordered by descending Elo ratings.

```
> extract.elo(res, "2000-05-28")

   c    d    a    f    k    s    g    n    w    z
1342 1214 1161 1133 1011 1000  958  844  799  538
```

[6] see also `?setwd`

```
> extract.elo(res, "2000-05-28", IDs=c("s", "a", "c", "k"))

   c    a    k    s
1342 1161 1011 1000
```

3.4 Plotting Elo ratings

`eloplot()` produces quick plots that visualize the development of Elo ratings over time. Note that the example data set contains a rather modest number of interactions and individuals. With larger data sets (both in terms of interactions and individuals), such plots can become messy quickly. Even though it is possible to restrict plotting to date ranges and subsets of individuals, we recommend to create custom plots by directly accessing the `res` object. Specifically, `res$mat` contains raw Elo ratings in a day-by-ID matrix, while the original dates can be found in `res$truedates`.

The following plot produces Figure 1.

```
> eloplot(res)
```

Restricting the date range and selecting only a subset of individuals results in Figure 2.

```
> eloplot(res,ids=c("s", "a", "w", "k", "c"),
        from="2000-06-05", to="2000-07-04")
```

Please note, the plotting function will plot a maximum of 20 individuals. Because we meant the plotting function to be an exploratory tool, you can also select `ids="random.20"` if you have more than 20 individuals. Please note also that individuals for which you have observed interactions on only one day in the selected date range (regardless of how many interactions on that day!), such individuals will be omitted from the plot. If you wish to plot such individuals as single points in the graph, you will have to use the approach mentioned above, i.e. use the `res$mat` and `res$truedates` objects. If you need help with that, please get in touch with us.

4 Incorporating presence data and undecided interactions

This section demonstrates how to incorporate presence data and undecided interactions. Please note that the presence data needs to cover *every* day during your data collection, i.e. also those days on which no interactions were observed. We start by reading the additional "presence matrix", followed by reformatting the date column in this object to a date format that `R` is capable of dealing with.

```
> xpres <- read.table(system.file("ex-presence.txt", package = "EloRating"),
     header = T)
> xpres[, 1] <- as.Date(as.character(xpres[, 1]))
```

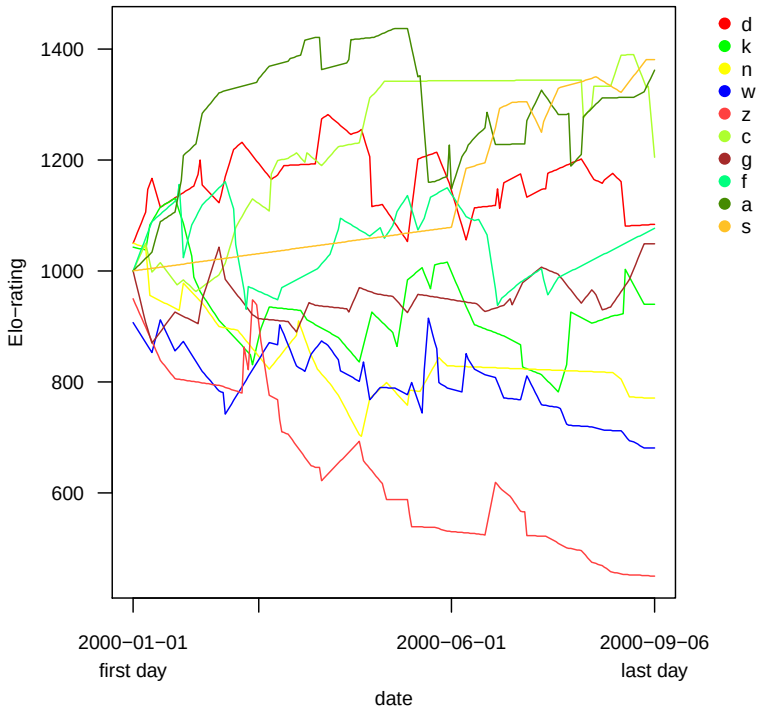

Figure 1: Elo ratings of 10 individuals over the entire study period.

Next, we rerun `seqcheck()` and `elo.seq()` with the additional `presence=` argument as well as incorporating the information about undecided interactions `draw=` into the latter function.

```
> seqcheck(winner=xdata$winner, loser=xdata$loser,
          Date=xdata$Date, presence=xpres)

presence data supplied, see below for details
Everything seems to be fine with the interaction sequence...OK

#####################################

presence data seems to be fine and matches interaction sequence...OK

#####################################

> res2 <- elo.seq(winner = xdata$winner, loser = xdata$loser, Date = xdata$Date,
     presence = xpres, draw = xdata$Draw)
```

Figure 2: Elo ratings of 5 individuals over a month.

Extracting Elo ratings takes advantage of the presence data by either omitting absent IDs from the output or returning them as `NA`. The differences in ratings stem from incorporating undecided interactions.

```
> extract.elo(res2, "2000-05-28")

   c    d    f    a    k    g    n    w    z
1340 1211 1136 1092  962  960  873  860  566

> # note that "s" is absent and omitted
> extract.elo(res2, "2000-05-28", IDs=c("s", "a", "c", "k"))

   c    a    k    s
1340 1092  962   NA

> # note that "s" is absent and returned as NA
```

Likewise, `eloplot()` omits absent IDs from the resulting plots.

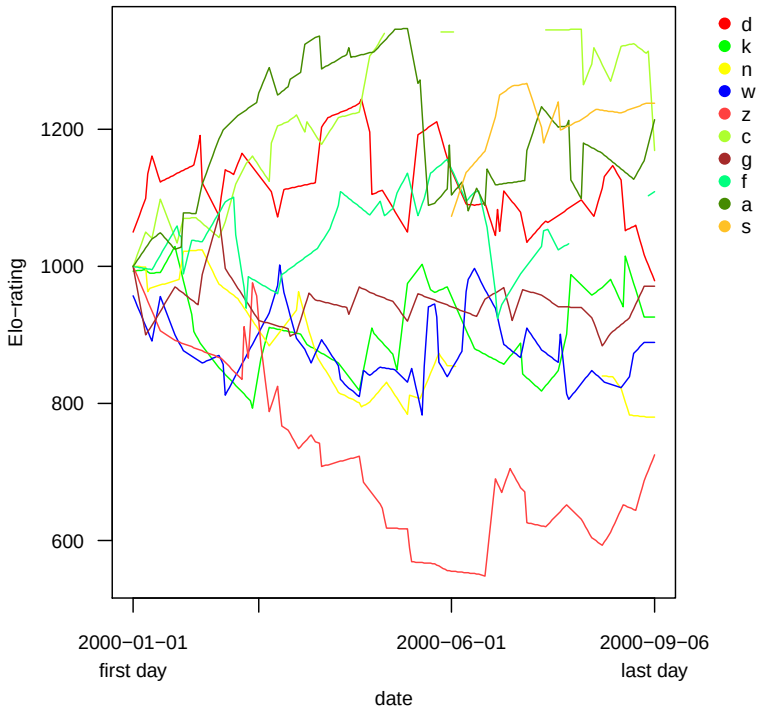

Figure 3: Elo ratings of 10 individuals over the entire study period. Note that several individuals were absent during parts of the date range and are therefore appear with gaps in the plot (e.g. "c" and "f"). Compare to Figure 1

```
> eloplot(res2)
```

```
> eloplot(res2, ids=c("s", "a", "w", "k", "c"),
          from="2000-06-05", to="2000-07-04")
```

5 Further functions

In addition to calculate, extract and display/plot Elo ratings, our package also provides some more functions that may be useful in some contexts.

5.1 hierarchy stability with `stab.elo()`

`stab.elo()` can be used to calculate an index of hierarchy stability (S, see Neumann et al. (2011) and McDonald and Shizuka (2013)). Please note that in contrast to the original

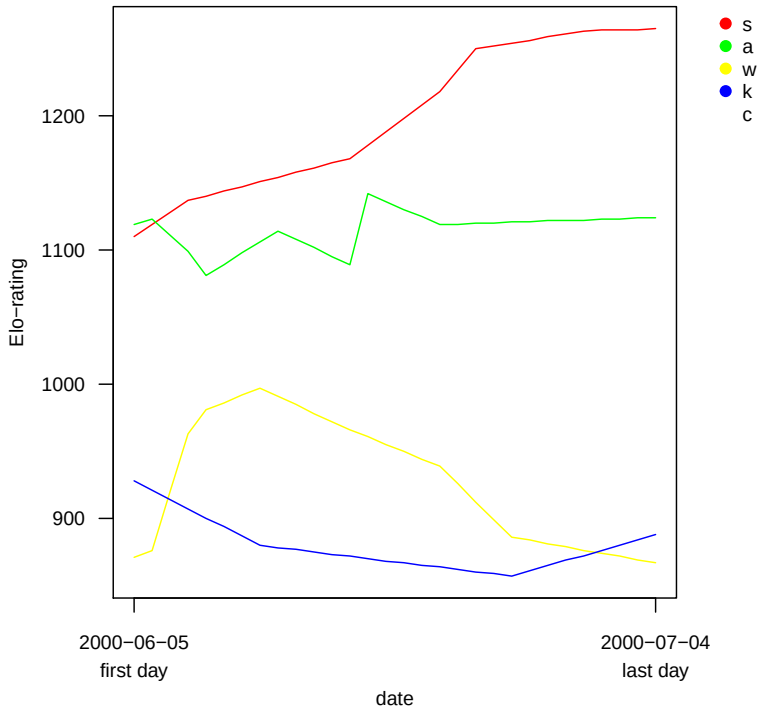

Figure 4: Elo ratings of 5 individuals over a month. Note that individual "c" is not displayed in the plot, since it has not been present during the date range supplied to `eloplot()`. Compare to Figure 2

publication, S now is limited to a range between 0 and 1, where 1 indicates a stable hierarchy in which no rank changes occured.

```
> stab.elo(res2, from="2000-05-05", to="2000-06-05")

[1] 0.9674
```

5.2 individual rating trajectories with `traj.elo()`

`traj.elo()` provides information about Elo rating trajectories over time.

```
> traj.elo(res2, ID=c("f", "n"),
          from="2000-05-05", to="2000-06-05")

  ID   fromDate     toDate    slope Nobs
1  f 2000-05-05 2000-06-05 1.696998    6
2  n 2000-05-05 2000-06-05 3.904463    5
```

5.3 individuals()

`individuals()` provides information about which/how many individuals were present on specific dates. When applied over a date *range*, the average number of individuals can be returned as can the coefficient of variation of the number of individuals present on each date. Note that this function has little relevance when the calculation of Elo ratings (see above) is *not* supplemented by presence data.

```
> individuals(res2, from="2000-05-05", to="2000-05-05", outp="N")

[1] 8

> individuals(res2, from="2000-05-05", to="2000-06-05", outp="N")

[1] 8.3125

> individuals(res2, from="2000-05-05", to="2000-06-05", outp="CV")

[1] 0.07125283

> individuals(res2, from="2000-05-05", to="2000-06-05", outp="IDs")

 [1] "d" "k" "n" "w" "z" "c" "g" "f" "a" "s"
```

5.4 winprob()

`winprob()` simply returns the expected probablity of an individual winning given its own Elo rating and that of its opponent.

```
> winprob(1000,1200)

[1] 0.2397501

> winprob(1200,1000)

[1] 0.7602499

> winprob(1200,1200)

[1] 0.5
```

5.5 creatematrix()

`creatematrix()` returns a square matrix which can be used with other, matrix-based algorithms to calculate dominance scores or ranks (e.g. I&SI (de Vries 1998) or David's score (David 1987, Gammell et al. 2003, de Vries et al. 2006, see section 5.8)). If undecided interactions (ties/draws) are present in the data, the user can decide on how to treat them (either 0.5 or 1 for both individuals, or they are omitted (default)). Individuals that were absent during the specified date range are excluded from the matrix by default. In addition, the matrix can be restricted to individuals that had interactions (i.e. *observed* interactions) in the date range.

```
> creatematrix(res2)
```

```
  a c d f g k n s  w  z
a 0 5 5 2 9 4 2 1 10  6
c 0 0 4 7 3 4 1 1  5  2
d 2 0 0 2 5 5 4 0  8 10
f 0 2 0 0 2 6 4 0  6  5
g 0 0 0 0 0 4 3 0  6  2
k 1 0 3 0 0 0 2 0  2  6
n 0 0 0 0 2 0 0 0  2  3
s 3 0 2 1 3 0 0 0  2  2
w 2 0 0 0 0 1 1 0  0 11
z 0 0 0 2 1 1 0 0  0  0
```

```
> sum(creatematrix(res2))
```

```
[1] 200
```

```
> creatematrix(res2, drawmethod="0.5")
```

```
    a   c   d   f    g   k   n   s    w    z
a 0.0 6.0 6.0 2.5 10.0 4.0 2.0 1.0 13.0  6.5
c 1.0 0.0 4.0 7.5  3.0 4.0 1.0 1.5  6.0  2.5
d 3.0 0.0 0.0 3.5  5.0 5.5 4.0 0.5  8.0 12.0
f 0.5 2.5 1.5 0.0  2.5 6.5 5.0 0.5  6.5  5.0
g 1.0 0.0 0.0 0.5  0.0 4.0 3.0 0.0  8.0  2.5
k 1.0 0.0 3.5 0.5  0.0 0.0 2.5 0.0  3.0  7.0
n 0.0 0.0 0.0 1.0  2.0 0.5 0.0 0.0  2.5  4.0
s 3.0 0.5 2.5 1.5  3.0 0.0 0.0 0.0  2.5  2.0
w 5.0 1.0 0.0 0.5  2.0 2.0 1.5 0.5  0.0 12.0
z 0.5 0.5 2.0 2.0  1.5 2.0 1.0 0.0  1.0  0.0
```

```
> sum(creatematrix(res2, drawmethod="0.5"))
```

```
[1] 250
```

```
> # "c" and "n" are omitted
> creatematrix(res2, c("2000-06-10", "2000-06-16"))
```

```
  a d f g k s w z
a 0 0 0 1 0 0 0 0
d 0 0 0 0 0 0 0 0
f 0 0 0 0 1 0 0 1
g 0 0 0 0 0 0 0 0
k 0 0 0 0 0 0 0 0
s 0 0 0 0 0 0 0 0
w 0 0 0 0 0 0 0 0
z 0 0 0 0 0 0 0 0
```

```
> creatematrix(res2, c("2000-06-10", "2000-06-16"),
              onlyinteracting=TRUE)
```

```
  a f g k z
a 0 0 1 0 0
f 0 0 0 1 1
g 0 0 0 0 0
k 0 0 0 0 0
z 0 0 0 0 0
```

5.6 randomsequence()

Finally, `randomsequence()` creates random data sets, which can be used for simulations for example. It returns a list with two `data.frame`s (named "seqdat" and "pres" for the actual sequence and presence data, respectively). By default, it creates a sequence of 100 interactions between 10 individuals. All IDs are present the entire time and there are no undecided interactions. Also by default, IDs are simply single letters and in order to produce realistic data, IDs that appear earlier in alphabetic order are more likely to win any given interaction (`alphabet=TRUE`). The proportion of reversals (against that order) is by default set to `reversals=0.1`.

```
> rdata <- randomsequence()

> xres <- elo.seq(rdata$seqdat$winner, rdata$seqdat$loser,
                  rdata$seqdat$Date, presence=rdata$pres)

> summary(xres)

Elo ratings from 10 individuals
total (mean/median) number of interactions: 100 (20/21)
range of interactions: 13 - 24
date range: 2000-01-01 - 2000-04-09
startvalue: 1000
uppon arrival treatment: average
k: 100
proportion of draws in the data set: 0
```

5.7 David's scores with DS()

With this function you can calculate David's scores (David 1987, Gammell et al. 2003, de Vries et al. 2006). Note that this function only works on square matrices (see above [section 5.5] for how to create a matrix from a sequence).

```
> data(bonobos)
> DS(bonobos)

  ID         DS    normDS
1 He  14.011236 5.0016051
2 Dz  11.635634 4.6622334
3 Ho   6.606742 3.9438202
4 De  -1.285714 2.8163265
5 Ko  -5.988764 2.1444623
6 Re  -8.729133 1.7529810
7 Ki -16.250000 0.6785714
```

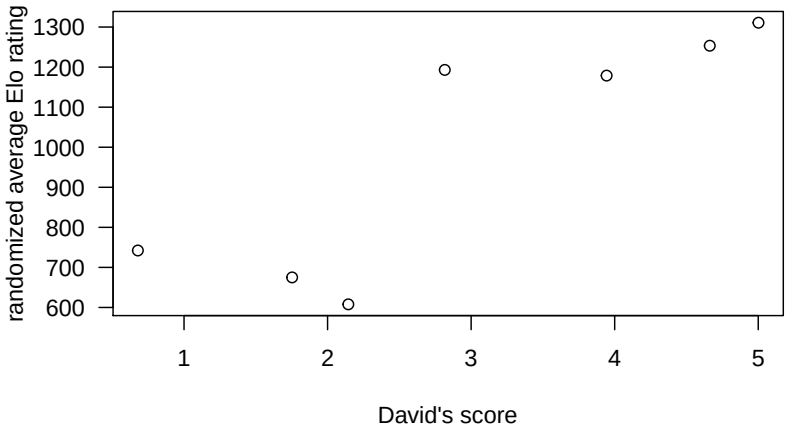

Figure 5: David's scores and average randomized Elo ratings from seven bonobos (data taken from de Vries et al. 2006)

5.8 `randomelo()`

This is an experimental function to generate a set of random sequences based on an interaction matrix. Based on the randomly generated sequences, Elo ratings are calculated.

```
> data(bonobos)
> xdata <- randomelo(bonobos, 100)

> res <- data.frame(ID=colnames(xdata[[1]]), avg=round(colMeans(xdata[[1]]),1))
```

Now, compare that to David's scores (figure 5).[7]

```
> ds <- DS(bonobos)
> ds <- ds[order(ds$ID), ]

> plot(ds$normDS, res$avg, xlab="David's score",
      ylab="randomized average Elo rating", las=1)
```

5.9 proportion of unknown relationships with `prunk()`

This function lets you determine how large the proportion of dyads in your data set is for which no interactions have been observed. You can use this function on both the results of `elo.seq()` or an interaction matrix. If used on an `eloobject`, you will see as a result the unknown relationships for all dyads that were found in the date range, and additionally

[7] Note, the plot you will get will differ because the generation of Elo ratings is based on *random* sequences

restricted to those dyads that were actually co-resident at some point during the date range. In the example, this results in the identical output since all dyads were co-resident at some point. Of course, the accuracy of the second part of the output depends on presence data being supplied. Note, for matrices, we cannot control for coresidency, so the second part of the output is omitted if `prunk()` is used with a matrix.

```
> data(adv); data(advpres)
> x <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date,
              presence=advpres)

> prunk(x, c("2010-01-01", "2010-01-15"))

     pu.all   dyads.all    pu.cores dyads.cores
      0.524      21.000       0.524      21.000

> mat <- creatematrix(x, c("2010-01-01", "2010-01-15"))
> prunk(mat)

   pu.all dyads.all
    0.524    21.000
```

References

David, H. A. (1987), 'Ranking from unbalanced paired–comparison data', *Biometrika* **74**(2), 432–436.

de Vries, H. (1998), 'Finding a dominance order most consistent with a linear hierarchy: a new procedure and review', *Animal Behaviour* **55**(4), 827–843.

de Vries, H., Stevens, J. M. G. & Vervaecke, H. (2006), 'Measuring and testing the steepness of dominance hierarchies', *Animal Behaviour* **71**(3), 585–592.

Gammell, M. P., de Vries, H., Jennings, D. J., Carlin, C. M. & Hayden, T. J. (2003), 'David's score: a more appropriate dominance ranking method than clutton-brock et al.'s index', *Animal Behaviour* **66**(3), 601–605.

McDonald, D. B. & Shizuka, D. (2013), 'Comparative transitive and temporal orderliness in dominance networks', *Behavioral Ecology* **24**(2), 511–520.

Neumann, C., Duboscq, J., Dubuc, C., Ginting, A., Irwan, A. M., Agil, M., Widdig, A. & Engelhardt, A. (2011), 'Assessing dominance hierarchies: validation and advantages of progressive evaluation with elo-rating', *Animal Behaviour* **82**(4), 911–921.