EloRating - a brief tutorial

Christof Neumann & Lars Kulik

March 20, 2014

1 Preliminary remarks

This tutorial describes 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

Up to now, the `EloRating` package is not yet available from any online repository and therefore needs to be installed from a local file. This is possible with the function `install.packages()`, for example:

```
> install.packages("EloRating_0.31.tar.gz", type = "source")
```

As soon as the package is accepted on CRAN, the following command should be sufficient to install EloRating (given you have a working internet connection):

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

---

[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.

If you know how to read data in R you may skip remainder of this section and proceed directly to the next section. The same applies if you handle and process your data in R anyway. Otherwise, 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 necessary data[5].

```
> library(EloRating)
> xdata <- read.table("ex-sequence.txt",
+                      header=T, sep="\t")
```

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 usually 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 them in an object we name `res`. Note that in order to ignore possible "warnings" from `seqcheck()` the argument `runcheck=FALSE` has to be set.

---

[3]see the R packages `gdata` 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 can be found in the package directory in the folder "vignettes"

[6]see also `?setwd`

```
> 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

> 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.
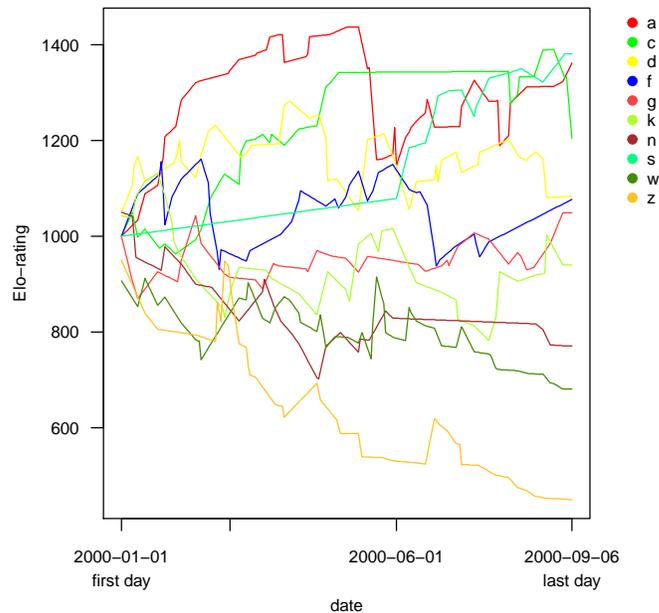
```
> eloplot(res)
```

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

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")
```

4 Incorporating presence data and undecided interactions

This section demonstrates how to incorporate presence data and undecided interactions. 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("ex-presence.txt", header = T, sep = "\t")
> xpres[, 1] <- as.Date(as.character(xpres[, 1]))
```
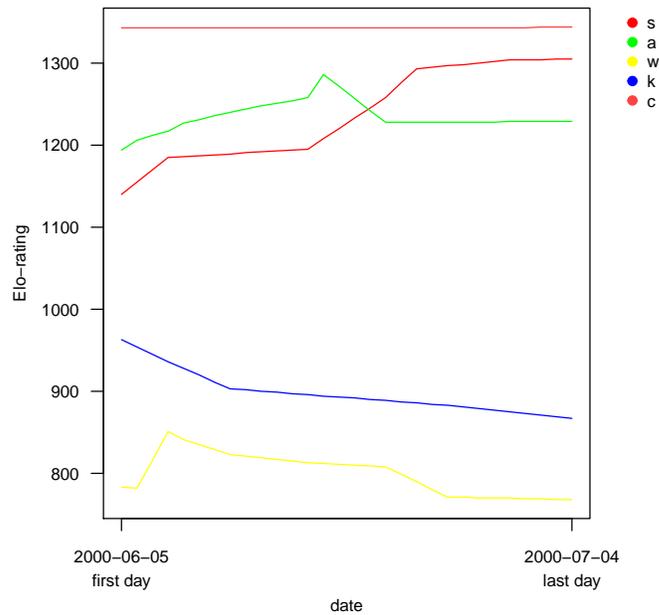
Figure 2: Elo ratings of 5 individuals over a month.

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

######################################
```

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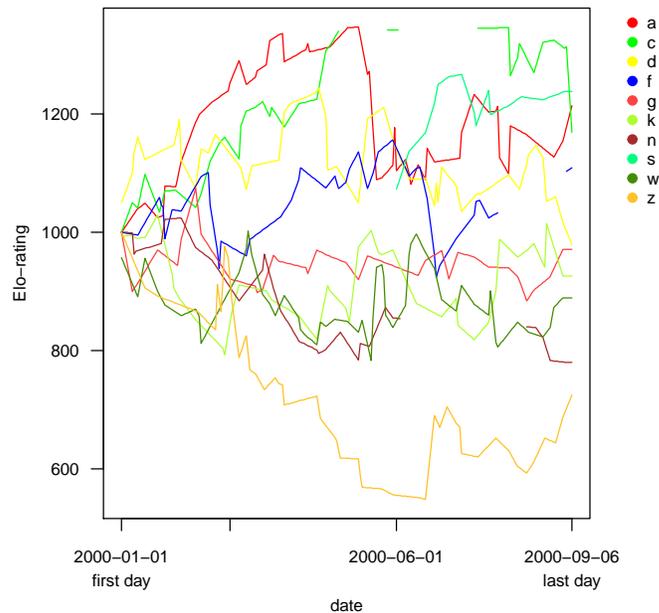
Figure 3: Elo ratings of 10 individuals over the entire study period. Note that several individuals were absent during parts of the date range and are therefore omitted from the plot (e.g. "c" and "f"). Compare to Figure 1

```
> 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.
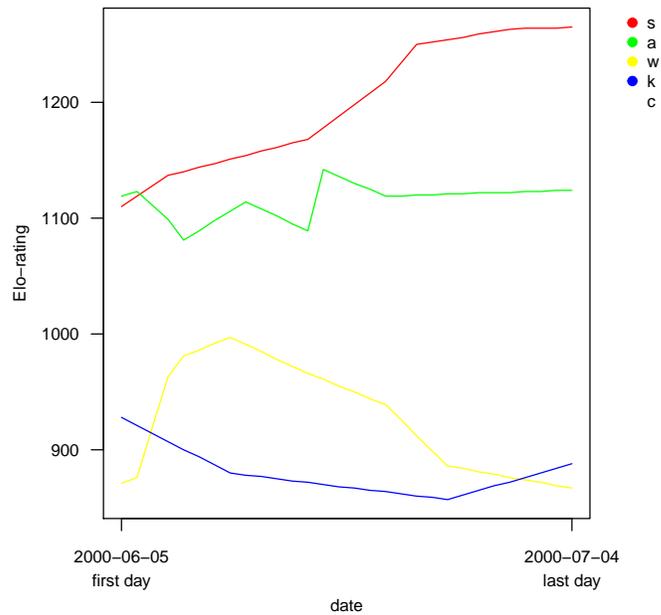
```
> eloplot(res2)
```

6

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

```
> 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   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).

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

```
[1] 0.9674
```

5.2 `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 or David's score). If undecided interactions 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 1 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, "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, "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.frames (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 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 (20/19)
range of interactions: 15 - 26
date range: 2000-01-01 - 2000-04-09
startvalue: 1000
uppon arrival treatment: average
k: 100
proportion of draws in the data set: 0
```