# `EloSteepness` - a brief tutorial

(package version 0.5.0)

Christof Neumann

2023-09-20

## Contents



# 1 prelims

`EloSteepness` is a package that allows estimating steepness of dominance hierarchies from interaction networks. These networks can either be static, i.e. we don't know the sequence in which the interactions occurred, or we have temporal information, i.e., we known the sequence in which the interactions occurred/were observed. From such data we estimate Bayesian Elo-ratings, from which the steepness metric can be calculated. The major difference from classic approaches is that we obtain posterior steepness *distributions*, not point estimates. More details on the theoretical background can be found in the accompanying paper (Neumann and Fischer 2023) (preprinted here: Neumann and Fischer (2022)).

Also included in this package is a version of steepness that is based on David's scores, but also with a Bayesian flavor. It turns out that the performance of this approach is somewhat below that of the Elo-based steepness, but still better than the classical algorithms. Also here the result is a posterior distribution. This latter approach will be featured only in passing in this tutorial, but in general, all the functions that work with the Elo-based algorithm will also work in a very similar way with the David's score-based steepness.

More interestingly, the package also contains functions that allow extraction of the raw individual dominance scores (either Bayesian Elo-ratings or David's scores), although this is probably only of secondary interest when focusing on *steepness*.

Another batch of functions in the package relate to method evaluation. These are currently ignored in this document. They are relevant for replicating the simulations and analyses presented in the paper.

# 2 data preparation

Depending on whether we know the sequence of interactions, we have to deal with two different data formats.

If we don't know the sequence, only thing you need to get going is a dominance interaction matrix. This is just a tabulation of dyadic dominance interactions where one individual is the winner and another individual is a loser. By convention, winners are usually represented in the rows and losers in the columns. The important thing is that the matrix needs to be square, i.e. it needs the same number of columns and rows. The `R` functions in the package require that your matrix has column and row names, which should correspond to the individual ids. For the sake of this tutorial, we use example matrices that are included in the `EloRating` package.

If we do know the sequence, we need two vectors, where one represents the winning individual and the other represents the losing individual. Crucially, the orders of the two vectors need to correspond to the order in which the interactions were observed.

Since the evaluation in our study focused on static networks (as those represent the majority of published data sets and also because steepness was conceptualized in a static framework), the largest chunk of this tutorial is devoted to the static approach.

# 3 an example with Bayesian Elo-rating (and unknown sequence of interactions)

## 3.1 steepness

We start with an example of dominance interactions between seven badgers (Hewitt, Macdonald, and Dugdale 2009). This data set has some peculiarities, which will become apparent while we work through the example.

```
library(EloSteepness)
```

```
data("dommats", package = "EloRating")
mat <- dommats$badgers
```

Here it is:

First we apply the workhorse function of the package `elo_steepness_from_matrix()`. In essence, the function only requires you to specify which interaction matrix to use via `mat =`. There is one more important argument though, which determines the number of randomized sequences to be used, `n_rand =`. If this is not explicitly set, the function will determine the value by itself based on the number of interactions in the data set (more details on that are in the paper). Here we have 118 interactions so our rule of thumb suggests to use 20 randomizations. If you want to speed up the calculations I would suggest (at least for the sake of this tutorial) to use a smaller value here though, e.g. `n_rand = 2`, like so:

| | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| **a** | | 6 | 4 | 0 | 4 | 11 | 5 |
| **b** | 0 | | 5 | 0 | 5 | 22 | 7 |
| **c** | 0 | 3 | | 0 | 2 | 7 | 4 |
| **d** | 0 | 0 | 0 | | 0 | 0 | 0 |
| **e** | 0 | 3 | 1 | 0 | | 1 | 2 |
| **f** | 0 | 4 | 1 | 1 | 3 | | 5 |
| **g** | 0 | 1 | 0 | 0 | 1 | 10 | |

Figure 1: An example network of seven badgers.

```
elo_res <- elo_steepness_from_matrix(mat = mat, n_rand = 2, refresh = 0,
                                     cores = 2, iter = 1000, seed = 1)
```

There are some more arguments for `elo_steepness_from_matrix()` that relate to `rstan`'s `sampling()` function. Unless you set them explicitly, they will remain at `sampling()`'s defaults. The most relevant are `refresh =`, which I set to 0 here simply to avoid printing intermediate output into this document. If you want to see some progress updates (`elo_steepness_from_matrix()` takes fairly long to complete...), it's a good idea to just don't set this argument. The other arguments relate to the number of iterations for sampling. `iter =` and `warmup =` determine the number of samples (defaults are 2000 and 1000, respectively). `chains =` sets the number chains (default is 4). And `cores =` determines the number of processor cores to use. Here the default is likely 1, but this depends on how your computer is set up. On my laptop I have 4 cores, so I can use `cores = 4`, which speeds things up substantially.

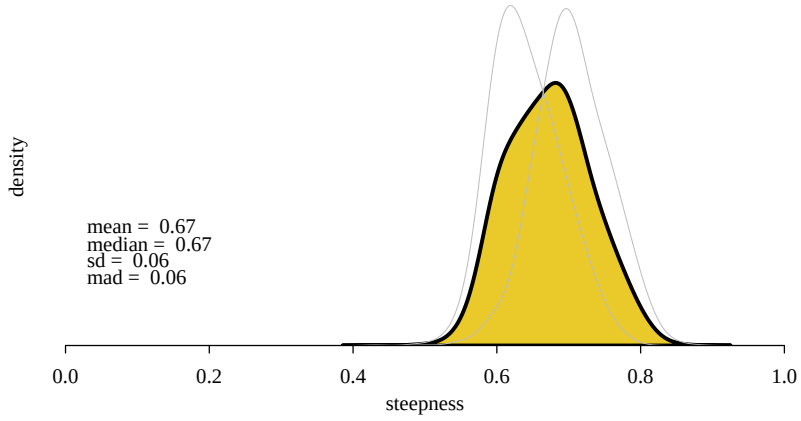
So, depending on your computer's performance that step may take anything between less than a minute and several minutes (or even more...).

Once this is done, we can obtain a numeric summary, which at this stage you probably only want to skim over. It is really just a summary of the things we will look at in more detail below and will give you an idea about whether anything went totally wrong.

```
summary(elo_res)
#> steepness based on Bayesian Elo-ratings
#> (using fixed standard deviation of start ratings)
#> ----------------------------------------
#> data supplied in matrix format:
#> number of randomized sequences: 2
#> ----------------------------------------
#> total number of posterior samples generated: 4000
#> number of samples with issues: 0
#> number of parameters with issues: 0
#> ----------------------------------------
#> matrix with 118 interactions between 7 individuals
#> 16.9 interactions per individual
#> 5.6 interactions per dyad
#> proportion of unknown relationships: 0.238
#> k value was estimated with median = 0.30 (89% CI: 0.03-0.87)
#> ----------------------------------------
#> mean steepness is 0.67 (SD=0.06)
#> median steepness is 0.67 (MAD=0.06)
#> 89% credible interval is between 0.58 and 0.77
```

If this worked, the first thing I'd do is to look at the posterior distribution of the steepness values.

```
plot_steepness(elo_res)
```




The yellow area gives a visual summary of the posterior combined across all randomizations. The finer gray lines (3 in my case, corresponding to 3 randomized sequences) represent the posteriors for each individual randomization. By default, the plot also provides some summary statistics, which in this case shows that mean and median of the posterior correspond to very similar values. These summary statistics can be turned off by setting `print_numbers = FALSE`. I'd also note that the posterior is fairly wide, indicating that steepness in this example is unlikely to be greater than 0.9 or smaller than 0.5 (as seen by nearly no probability mass falling outside the interval between 0.5 and 0.9).

To get some 'hard numbers' on the uncertainty, we can use yet another function:

```
p <- steepness_precis(elo_res, quantiles = c(0.055, 0.25, 0.75, 0.945))
round(p, 2)
#>   mean   sd median  mad q055 q250 q750 q945 mean_cv median_cv
#> 1 0.67 0.06   0.67 0.06 0.58 0.63 0.71 0.77    0.07      0.07
```

Here the same numbers are returned as in the plot (mean, median, standard deviation and median absolute deviation). In addition, it provides quantiles of the posterior and the ones I specified above are the default ones (so if you leave the argument not-specified you'll get the same output). These quantiles show that only 5.5% of posterior samples provided steepness values below 0.58 and 5.5 % above 0.74 (the 89% credible interval). This by and large confirms the visual assessment. If you want to report interquartile-range, this is also included via the 25% and 75% quantiles (0.62 and 0.69).

The final two numbers represent measures of variation (coefficient of variation) across the randomizations, and are calculated only for the mean and median. I'm not sure yet how to interpret those (other than 'smaller is better'), but they are here to be explicit about the fact that we use and combine different interaction sequences to obtain our steepness distribution, and that each interaction sequence will result in slightly different posteriors.

And if you want to create your own plots or get your own numeric summaries, you can extract all steepness values (i.e. the samples which form the basis for the plots and summaries above) by accessing:

```
elo_res$steepness
```

This is a matrix with one column for each randomization (i.e. `n_rand`) and one row for each sample. So you could for example draw a histogram if you feel this is what you want (I don't show the actual histogram here though).

```
hist(elo_res$steepness, xlim = c(0, 1))
```

And if you want to dig into the actual `rstan` object, this is available via:

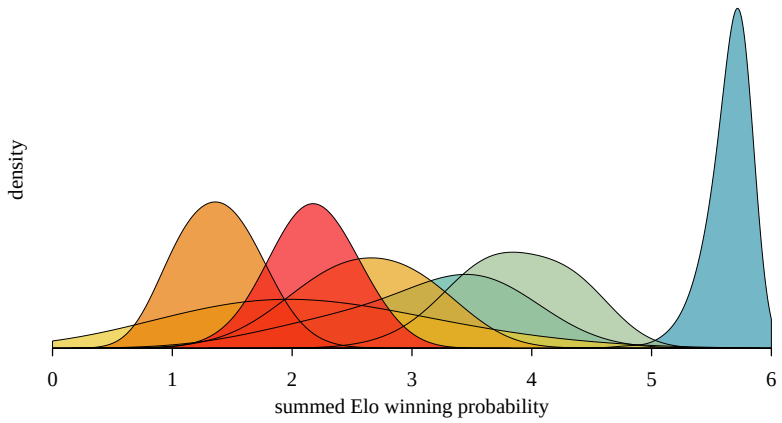
```
elo_res$stanfit
```

## 3.2 individual scores ('ranks')

As far as steepness is concerned, this is it (more or less). But then, what is actually also really interesting is to look at the underlying individual scores on which the steepness calculation is based, i.e. the dominance 'ranks' of individuals. In the case of Elo-rating, these are summed winning probabilities (see paper for more details).

We can explore those visually and numerically. First, we define a set of seven colors, so that individuals will appear with the same colors across the different plots. And then use `plot_scores()` to visualize the posteriors.

```
my_colors <- hcl.colors(n = 7, palette = "zissou1", alpha = 0.7)
plot_scores(elo_res, color = my_colors)
```




This shows the posterior distributions of individual summed winning probabilities. Numerically, we can get a summary via:

```
my_scores <- scores(elo_res, quantiles = c(0.055, 0.945))
```

```
my_scores
```

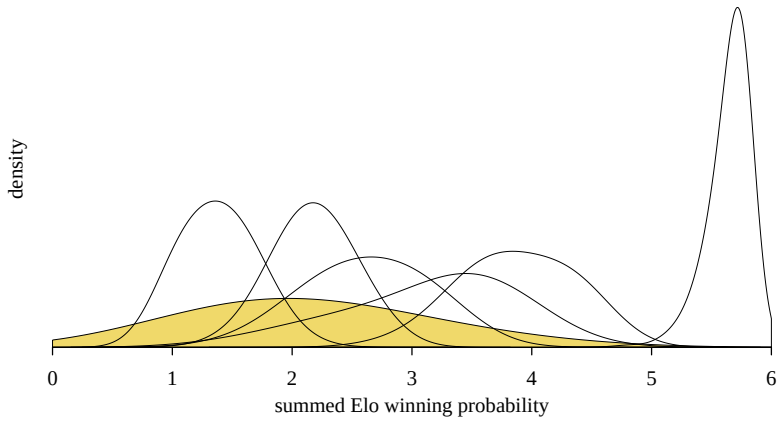
| id | mean | sd | median | mad | q055 | q945 | mean_cv |
|---|---|---|---|---|---|---|---|
| a | 5.65 | 0.17 | 5.69 | 0.13 | 5.32 | 5.83 | 0.01 |
| b | 3.14 | 0.66 | 3.33 | 0.59 | 1.88 | 3.93 | 0.20 |
| c | 3.88 | 0.44 | 3.86 | 0.52 | 3.17 | 4.52 | 0.11 |
| d | 2.15 | 0.93 | 2.06 | 0.95 | 0.78 | 3.80 | 0.01 |
| e | 2.62 | 0.48 | 2.63 | 0.53 | 1.84 | 3.35 | 0.15 |
| f | 1.38 | 0.30 | 1.38 | 0.33 | 0.93 | 1.87 | 0.19 |
| g | 2.18 | 0.32 | 2.18 | 0.31 | 1.67 | 2.69 | 0.00 |

These numbers reflect the same summaries of the individual posteriors as for the steepness measure above, i.e. central tendencies (mean and median), spread (SD, MAD, quantiles) and variation across the different randomized sequences.

There are two things that I wanted to point out, which are nicely illustrated in this particular example.

The first thing is how data density *can* inform (un)certainty of individual scores. There is one individual *d*, which only has one observed interaction (a loss against *f*). If we focus on *d* we see can several things.

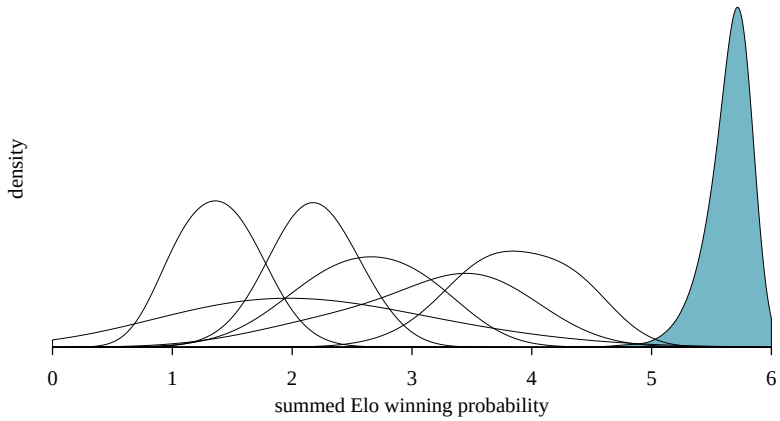
```
plot_scores(elo_res, color = my_colors, subset_ids = "d")
```




For one, its posterior is pretty wide, ranging from almost 0 to about 5, with its 'peak' at about 2 (median is 1.91 to be precise). And this makes sense, because we have very little data for *d* (just one observation), which leads to the wide posterior. Also, this one observation was a loss, which shifts the peak to below the average value (a naive view would suggest that an individual for which we have no knowledge is average, which is in fact the prior in the `Stan` model). The average is 3 in this case ($(n-1)/2$), which you can verify by `mean(my_scores$mean)`. The fact that *d*'s peak is shifted fairly far below 3 is probably due to the loss of *d* being against *f*, which is a fairly low-rated individual itself (because it lost most of its interactions).

In contrast to *d*, individual *a*'s posterior is the narrowest of all individuals.

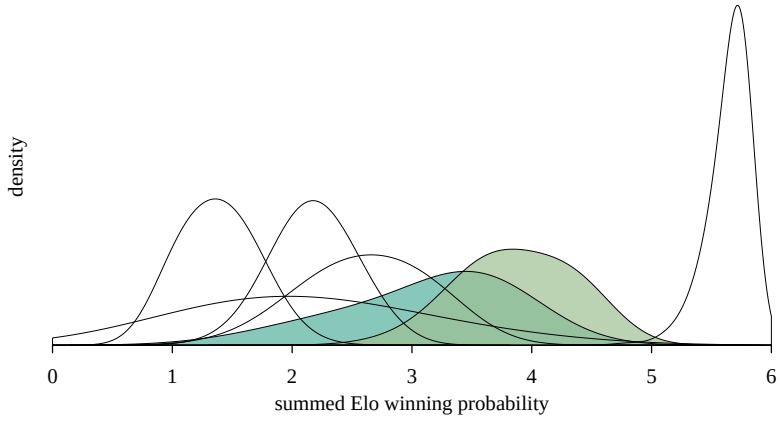
```
plot_scores(elo_res, color = my_colors, subset_ids = "a")
```




$a$ does, however, not have the largest number of observations as one might suspect. But, $a$ never lost an interaction but won all of them, so it seems fair to say that we should expect a high rating for it (or rather summed winning probability) with fairly little uncertainty.

The second thing I wish to point in the example are individuals $b$ and $c$.

```
plot_scores(elo_res, color = my_colors, subset_ids = c("b", "c"))
```




Their posteriors overlap extensively, although $b$ has a slightly higher mean/median.[1] The interesting thing here is that these two are close in rank/rating/score whatever way you look at it. And their *dyadic* rela-

[1] In fact, the medians of these two individuals are so close that in some runs (e.g. if using a different number of randomizations or a different seed), chances are that $c$ will have a slightly higher median.

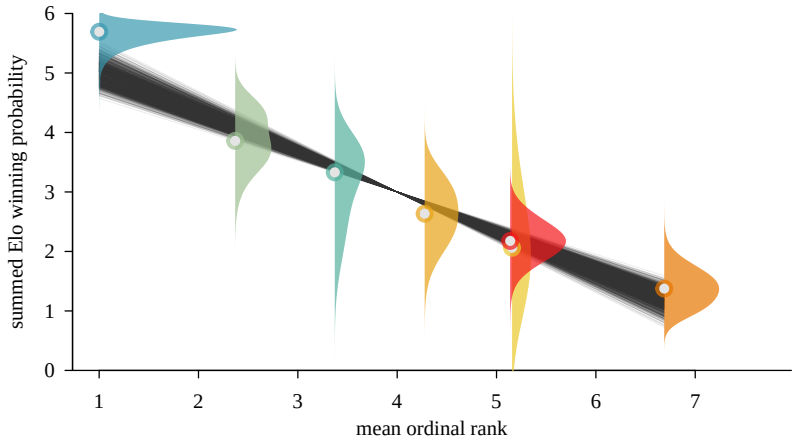
tionship is far from clear-cut (out of 8 interactions, $b$ won 5 and $c$ won 3), so it makes sense that these two overlap with a slight 'advantage' for $b$.

Let me just repeat here that this is a specific example. But in my view it nicely illustrates how the idea of considering individual ratings as posterior *distributions* makes a lot of characteristics of dominance relationships explicit that we otherwise could only intuit.

### 3.3 combining steepness and individual scores

The last way of visualizing steepness is to combine steepness with the individual scores. Here is my attempt at doing this:

```
plot_steepness_regression(elo_res, color = my_colors, width_fac = 0.5)
```




This puts together everything. It provides the individual posteriors alongside the mean steepness slope (and a random sample of 1,000 steepness values from the steepness posterior). Remember, steepness at its core is the slope of regressing scores/ratings on their ordinal ranks. The one thing that this figure illustrates which is hard to visualize differently, is that in each of the posterior draws the ordinal ranks are calculated afresh. And here we plot the mean rank across all samples for each individual along the horizontal rank axis. Coming back to individuals $b$ and $c$, whose posteriors overlap substantially: there are a lot of posterior samples in which $b$ has rank 2, but there is also a substantial amount of samples in which $c$ is ranked second. This is the reason that these two individuals come close to each other along the horizontal axis.

And you may have noticed the `width_fac=` argument. This is just a visual adjustment of how 'high' the highest individual posterior extends (along the horizontal axis). So if you use this on your own data, the value needs to be adjusted (by some trial and error, I suppose).

To tie this all together: If you want to report steepness for your own data set, I think the most important information is the posterior of the steepness itself (`plot_steepness()` and `steepness_precis()`). How you actually report this depends a bit on context I suppose. For a scientific paper, I would recommend a combination of point estimate (median of the posterior), some credible intervals (for example 89% and 50%) and visual display (perhaps in the supplement). This should give a reader all the relevant information.

### 3.4 more on data density

To illustrate some more of the underlying intuitiveness of this system let's run a little experiment/simulation. The core idea is here is to show how data density informs our assessment of uncertainty. And by *data density* I mean the number of interactions per individual (or per dyad). With more data, uncertainties should become smaller (narrower posterior distributions and credible intervals). With less data, uncertainties should become larger (wider posteriors and credible intervals).

We will take the same badger data set, but pretend we have observed only for half the time and obtained only half the data (i.e. half the interactions). We simulate this by dividing the observations in each cell by 2 (and we round .5 randomly up or down).[2] The important thing is that by doing so, we don't change the *network density*, i.e. the proportion of unknown relationships remains constant across the three networks (5 unobserved dyads out of 21).

```
data("dommats", package = "EloRating")
mat <- dommats$badgers

set.seed(123)
mat1 <- mat / 2
und <- mat1 - floor(mat1) != 0
mat1[und] <- round(mat1[und] + runif(sum(und), -0.1, 0.1))
mat1["f", "d"] <- 1 # just make sure that 'd' keeps its one loss to 'f'
```

And then we also pretend that we observed for twice the amount of time by just doubling all values in `mat`.

```
mat2 <- mat * 2
```

Now we just need redo the calculations for those two 'new' data sets. Note that I set `cores = 4` as my PC has 4 cores and also use a smaller number of randomized sequences, to speed up things.

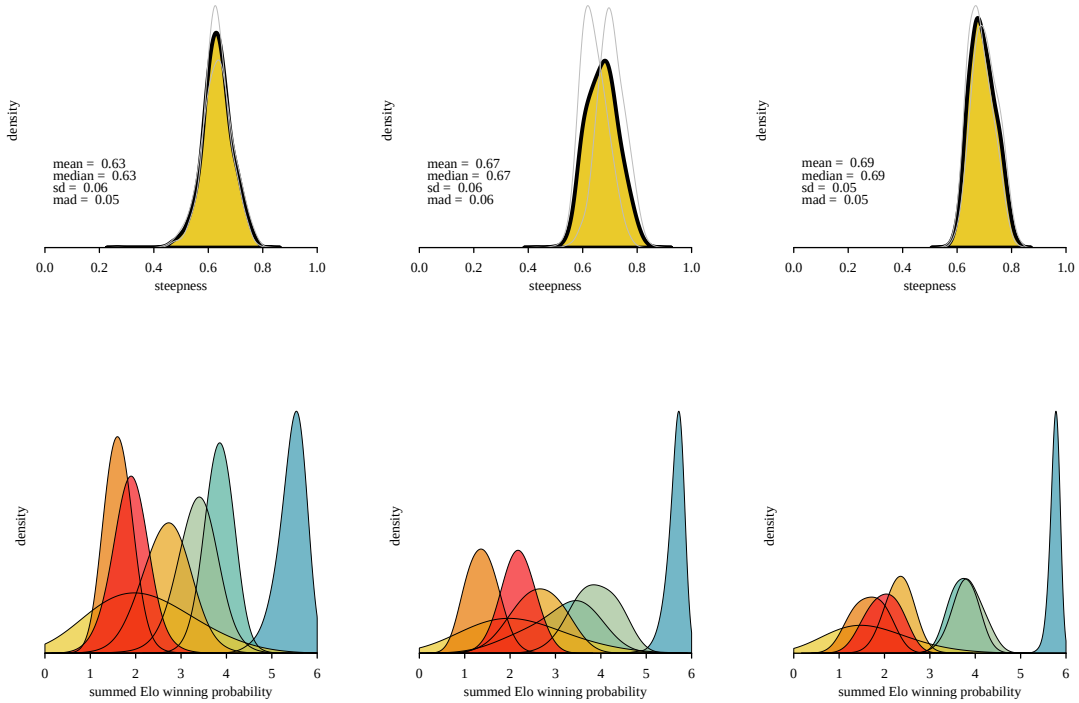
```
elo_res_half <- elo_steepness_from_matrix(mat = mat1, n_rand = 2, refresh = 0,
                                          cores = 2, iter = 1000, seed = 2)
elo_res_doubled <- elo_steepness_from_matrix(mat = mat2, n_rand = 2, refresh = 0,
                                             cores = 2, iter = 1000, seed = 3)
```

Then we visualize the posteriors of steepness and the individual scores.

```
plot_steepness(elo_res_half)
plot_steepness(elo_res)
plot_steepness(elo_res_doubled)

plot_scores(elo_res_half, color = my_colors)
plot_scores(elo_res, color = my_colors)
plot_scores(elo_res_doubled, color = my_colors)
```

[2] we just have to make sure that in the new matrix $d$ keeps its one interaction, because otherwise $d$ would be dropped (as would any individual without any interactions observed), but we want to have equal group sizes




In this plot, the original data set is in the center column, with the reduced data on the left and the doubled data on the right. The thing to note here is that the credible intervals become narrower from left to right, i.e. from less to more data. Admittedly though, this is not extremely obvious in this example. But you can verify it by looking at the output of `steepness_precis()` for each of the steepness objects (`elo_res_half`, `elo_res` and `elo_res_doubled`).[3]

As far as the individual scores are concerned, the same general pattern of credible intervals becoming narrower with more data emerges. Again, this is not overly obvious, other than for the individual with the highest score (blue on the right). What is interesting here is that the overlap between adjacent individuals does not seem to become smaller, i.e. there are 'clusters' of individuals, where even more data doesn't help to separate their scores.

## 3.5 a final example

Finally, we go for another example. This one actually uses simulated data. The major point of this example is to illustrate the idea of more data generally equaling less uncertainty (i.e. narrower distributions). The second point of this example is that posteriors do not necessarily look symmetric and can have indeed fairly odd shapes[4].

We also repeat our experiment about 'increasing the amount of data'. But this time we just multiply, so as not to have to bother with rounding up or down.

```
set.seed(123)
```

[3] It might be a good idea to actually increase the number of randomizations, to make this more obvious.

[4] Although at least some of these odd shapes, particularly the bimodal distributions, are likely to be artefacts of using only 2 randomizations in this example.

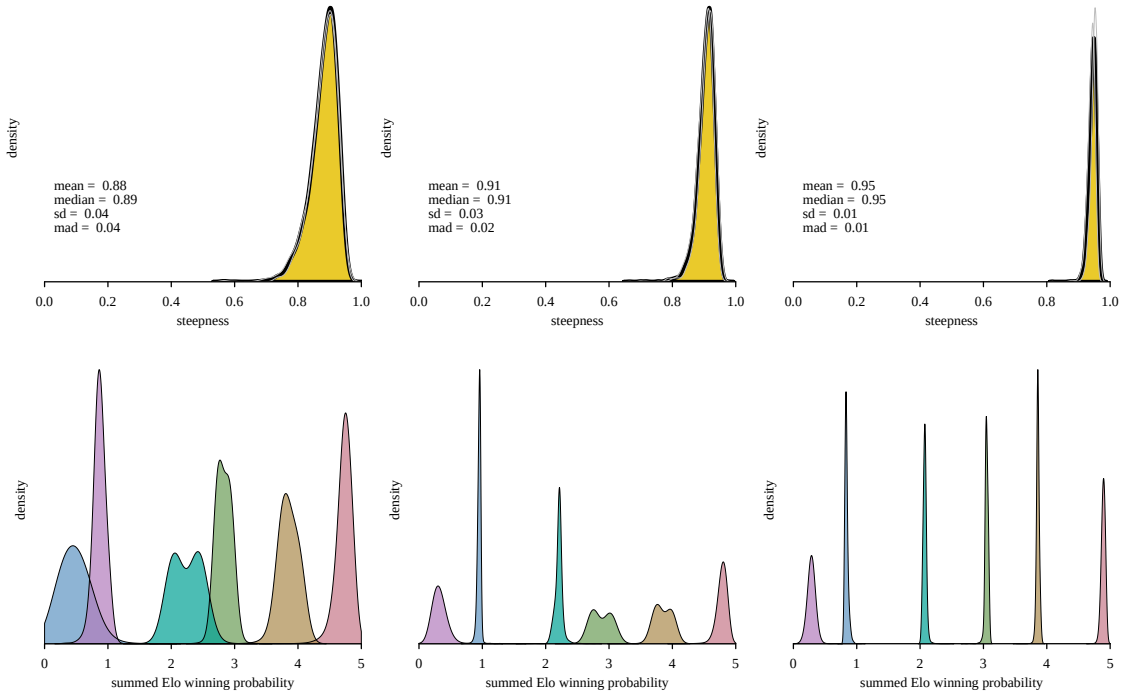
```r
# generate matrices
m1 <- simple_steep_gen(n_ind = 6, n_int = 40, steep = 0.9)$matrix
m2 <- m1 * 2
m5 <- m1 * 5

# calculate steepness
r1 <- elo_steepness_from_matrix(mat = m1, n_rand = 2, cores = 2, iter = 1000,
                                seed = 1, refresh = 0)
r2 <- elo_steepness_from_matrix(mat = m2, n_rand = 2, cores = 2, iter = 1000,
                                seed = 2, refresh = 0)
r5 <- elo_steepness_from_matrix(mat = m5, n_rand = 2, cores = 2, iter = 1000,
                                seed = 3, refresh = 0)
```

```r
mycols <- hcl.colors(6, palette = "Dark 2", alpha = 0.7)

plot_steepness(r1)
plot_steepness(r2)
plot_steepness(r5)

plot_scores(r1, color = mycols)
plot_scores(r2, color = mycols)
plot_scores(r5, color = mycols)
```



# 4 a brief example with Bayesian Elo-rating (and known sequence of interactions)

If we actually know the sequence of interactions, we can do away with randomized sequences as a means to deal with this specific source of uncertainty.

To demonstrate, we use a data set on male baboons (Franz et al. 2015b, 2015a).

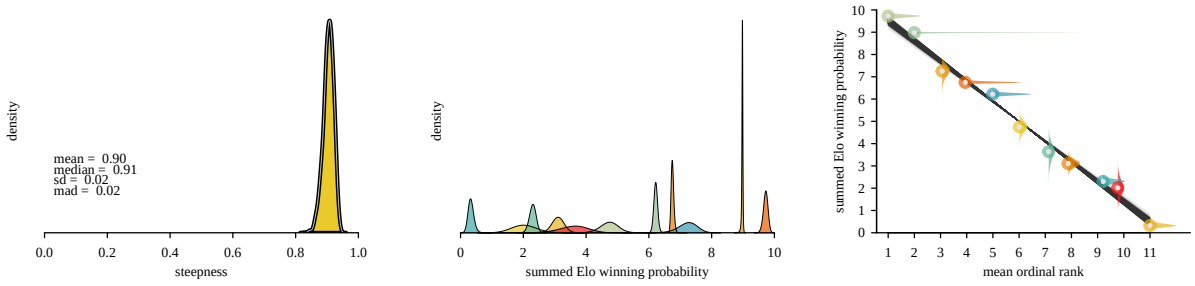
```
data("baboons1", package = "EloRating")
```

To make this data set a bit more manageable (the full data set contains 4118 interactions between 61 individuals), we reduce the original sequence to 200 interactions, which occur among 11 males.

The key difference is that we don't supply an interaction matrix, but two vectors with ids of winners and losers, respectively. To be explicit, the function that calculates ratings, summed winning probabilities and steepness is named `elo_steepness_from_sequence()`. Apart from supplying two vectors instead of a matrix, the function works the same as `elo_steepness_from_matrix()`, except of course the absence of the `n_rand =` argument. The remaining functions also work the same way (I won't show the results of `summary(babseq)`, `steepness_precis(babseq)` and `scores(babseq)` here, though).

```
s <- baboons1[1:200, ]
babseq <- elo_steepness_from_sequence(winner = s$Winner,
                                      loser = s$Loser,
                                      refresh = 0,
                                      cores = 2,
                                      seed = 1,
                                      iter = 1000)
```

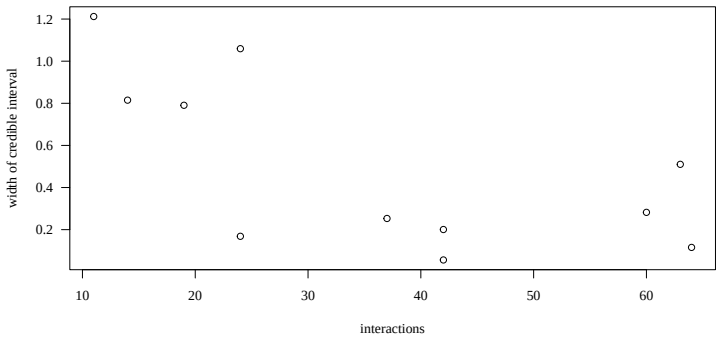
```
plot_steepness(babseq)
plot_scores(babseq)
plot_steepness_regression(babseq, width_fac = 0.2)
```




One other interesting thing to look at here, which again hopefully proves correct some of the intuitions I have, is the idea that also on an individual level, more data should translate into lower uncertainties. For this, we extract the number of interactions per individual and plot them against the width of the posterior. I'd expect a negative relationship, i.e., more interactions lead to narrower posteriors of the summed winning probabilities.

```r
# extract number of interactions
ints <- table(c(s$Winner, s$Loser))
ints <- ints[order(names(ints))]
# get the scores for all individuals
the_scores <- scores(babseq)
the_scores <- the_scores[order(the_scores$id), ]

plot(as.numeric(ints), the_scores$q955 - the_scores$q045,
     xlab = "interactions", ylab = "width of credible interval", las = 1)
```




So that seems to work out reasonably well, albeit perhaps not as perfectly as one would have hoped.

# 5 steepness based on Bayesian David's scores

Using David's scores instead of Elo-rating-based summed winning probabilities works pretty much in the same way. The one key difference is that there is no data sequence to be randomized. Therefore, we don't need to set randomizations because the matrix is just what it is[5].

The workhorse function is `davids_steepness()`, which only requires the `mat=` argument to be specified. We use again the badgers data set to illustrate.

Given the results of the method comparison in the paper, I would recommend not to interpret the steepness value (based on Bayesian David's scores) of this exercise as meaningful given that there are still a substantial amount of unknown relationships in this data set.

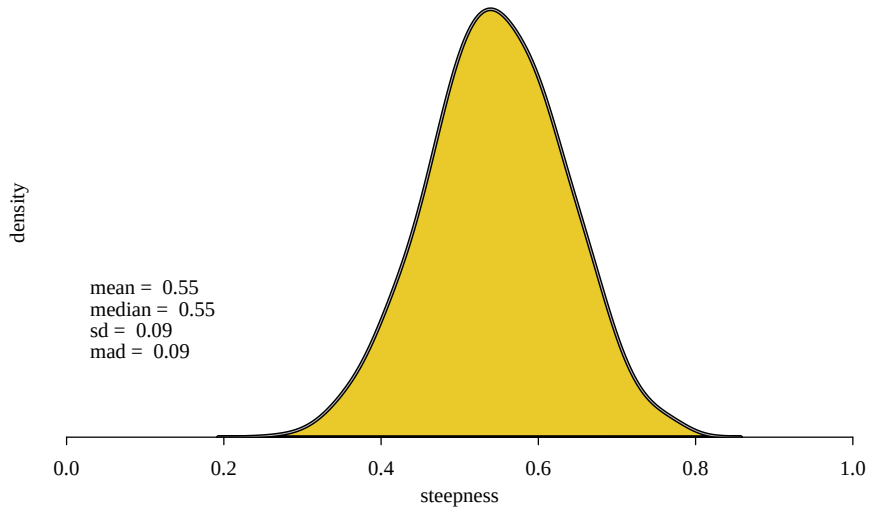
```r
data("dommats", package = "EloRating")
mat <- dommats$badgers
```

```r
david_res <- davids_steepness(mat, refresh = 0, seed = 1, iter = 1000, cores = 2)
```

Since there are no randomized sequences involved, the steepness posterior does only show the results of the single set of samples along with some descriptives of the posterior.

[5] as opposed to Elo-rating, which requires translating the matrix into sequences of interactions for which the actual order of interactions is not known and hence randomized multiple times

```
plot_steepness(david_res)
```

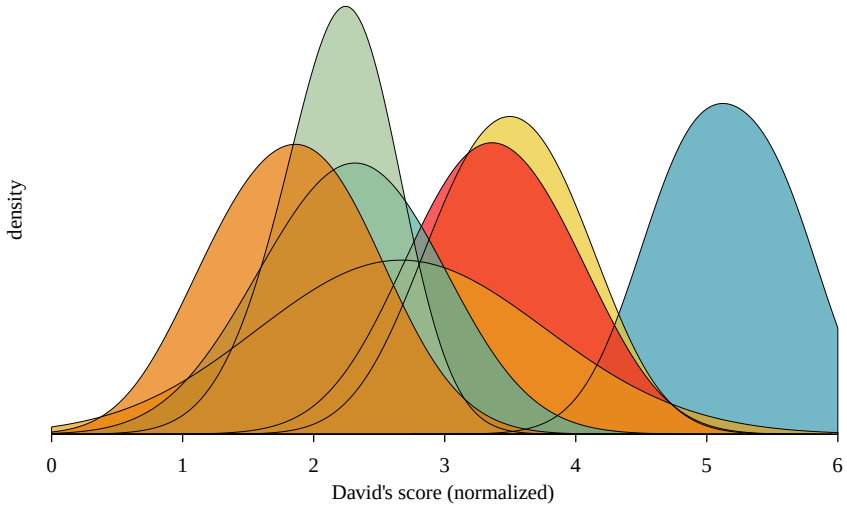



```
summary(david_res)
#> steepness based on Bayesian David's scores
#> -----------------------------------------
#> total number of posterior samples generated: 2000
#> number of samples with issues: 0
#> number of parameters with issues: 0
#> ---------------------------------------
#> matrix with 118 interactions between 7 individuals
#> 16.9 interactions per individual
#> 5.6 interactions per dyad
#> proportion of unknown relationships: 0.238
#> -----------------------------------------
#> mean steepness is 0.55 (SD=0.09)
#> median steepness is 0.55 (MAD=0.09)
#> 89% credible interval is between 0.41 and 0.68
```

The other functions that are available for Elo-rating based steepness are also available for the David's score-based steepness:

```
round(steepness_precis(david_res), 2)
#>   mean   sd median  mad q055 q250 q750 q945
#> 1 0.55 0.09   0.55 0.09 0.41 0.49 0.61 0.68
```

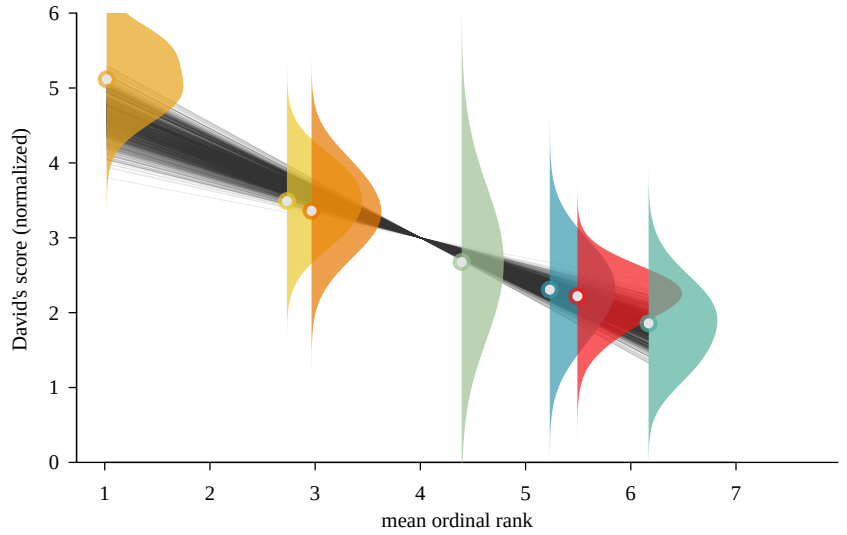
```
plot_scores(david_res)
```




```
scores(david_res)
```

| id | mean | sd | median | mad | q045 | q955 |
|---|---|---|---|---|---|---|
| a | 5.14 | 0.42 | 5.12 | 0.49 | 4.45 | 5.78 |
| b | 3.49 | 0.45 | 3.49 | 0.51 | 2.75 | 4.22 |
| c | 3.37 | 0.50 | 3.36 | 0.54 | 2.53 | 4.19 |
| d | 2.69 | 0.85 | 2.68 | 0.88 | 1.30 | 4.17 |
| e | 2.28 | 0.54 | 2.31 | 0.57 | 1.35 | 3.18 |
| f | 2.18 | 0.36 | 2.22 | 0.34 | 1.53 | 2.74 |
| g | 1.85 | 0.49 | 1.86 | 0.55 | 1.04 | 2.68 |

```
plot_steepness_regression(david_res, width_fac = 1)
```




This document took 0 minutes to compile.

# References


Franz, Mathias, Emily McLean, Jenny Tung, Jeanne Altmann, and Susan C Alberts. 2015a. "Data from: Self-Organizing Dominance Hierarchies in a Wild Primate Population." *Dryad*. https://doi.org/10.5061/dryad.d0g0d.

———. 2015b. "Self-Organizing Dominance Hierarchies in a Wild Primate Population." *Proceedings of the Royal Society B: Biological Sciences* 282: 20151512. https://doi.org/10.1098/rspb.2015.1512.

Hewitt, Stacey E, David W Macdonald, and Hannah L Dugdale. 2009. "Context-Dependent Linear Dominance Hierarchies in Social Groups of European Badgers, *Meles meles*." *Animal Behaviour*, 161–69. https://doi.org/10.1016/j.anbehav.2008.09.022.

Neumann, Christof, and Julia Fischer. 2022. "Extending Bayesian Elo-Rating to Quantify Dominance Hierarchy Steepness." *bioRxiv*, 2022.01.28.478016. https://doi.org/10.1101/2022.01.28.478016.

———. 2023. "Extending Bayesian Elo-Rating to Quantify Dominance Hierarchy Steepness." *Methods in Ecology and Evolution* 14: 669–82. https://doi.org/10.1111/2041-210X.14021.