`EloSteepness` - a brief tutorial

(package version 0.3.0)

Contents

prelims **1**

installing `EloSteepness` **1**

a final example 2

Bayesian David's scores **4**

prelims

`EloSteepness` is a package that allows estimating steepness of dominance hierarchies from interaction networks. It does so by estimating 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 preprint/paper.

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 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.

installing `EloSteepness`

The first requirement is that you use a fairly recent version of `R`, i.e. at least `R 3.5.0`, no way around that. To find which version you have, do this:

```
R.version$version.string
#> [1] "R version 4.0.4 (2021-02-15)"
```

If this returns at least 3.5.0 (or as in my case "`R version 4.0.4 (2021-02-15)`") all is good. Otherwise you need to update `R`, which might be a good idea anyway.

In order to get the package up an running you need a working installation of `rstan`. This in turn requires `stan` to be installed but this is taken care of during the setup of the `rstan` package. The easiest way of doing all this is to install the `brms` package.[1] If you already have `brms` (or `rstan`) then you are probably good to go. If not, then execute the following command and if asked for whether you want to install packages *from source* select 'no' (unless you know what you are doing of course).

[1] `brms` is not actually required for `EloSteepness` to work, but it handles the installation of `rstan` and friends very conveniently.

```
install.packages("brms")
```

The only other thing you need are two more packages, `EloRating` and `aniDom`, which are easy to install:

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

With this done, you can install `EloSteepness`. For the moment, this works only via a local file that can be downloaded here https://owncloud.dpz.eu/index.php/s/N8VW9a9QIVIuIqZ. In the near future I will make the package publicly available via GitHub and hopefully also via CRAN.

Which file to choose from the three in the folder depends on your OS and your level of adventurousness. Download the one you need (don't unpack it!), and remember the path you saved it to...

If you are on Windows, go for the `.zip` file and run (and don't forget to change the path):

```
install.packages("C:/where/i/saved/my/file/EloSteepness_0.2.0.zip",
                 repos = NULL, type = "win.binary")
```

If you are on MacOS, go for the `.tgz` file and run (and don't forget to change the path):

```
install.packages("~/where/i/saved/my/file/EloSteepness_0.2.0.tgz",
                 repos = NULL, type = "mac.binary")
```

And finally, if you are on Linux and/or feel adventurous take the `.tar.gz` file:

```
install.packages("~/Documents/EloSteepness_0.2.0.tar.gz",
                 repos = NULL, type = "source")
```

And for good measure at this point it might be a good idea to restart your computer, or at least restart R (or RStudio).

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.

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.

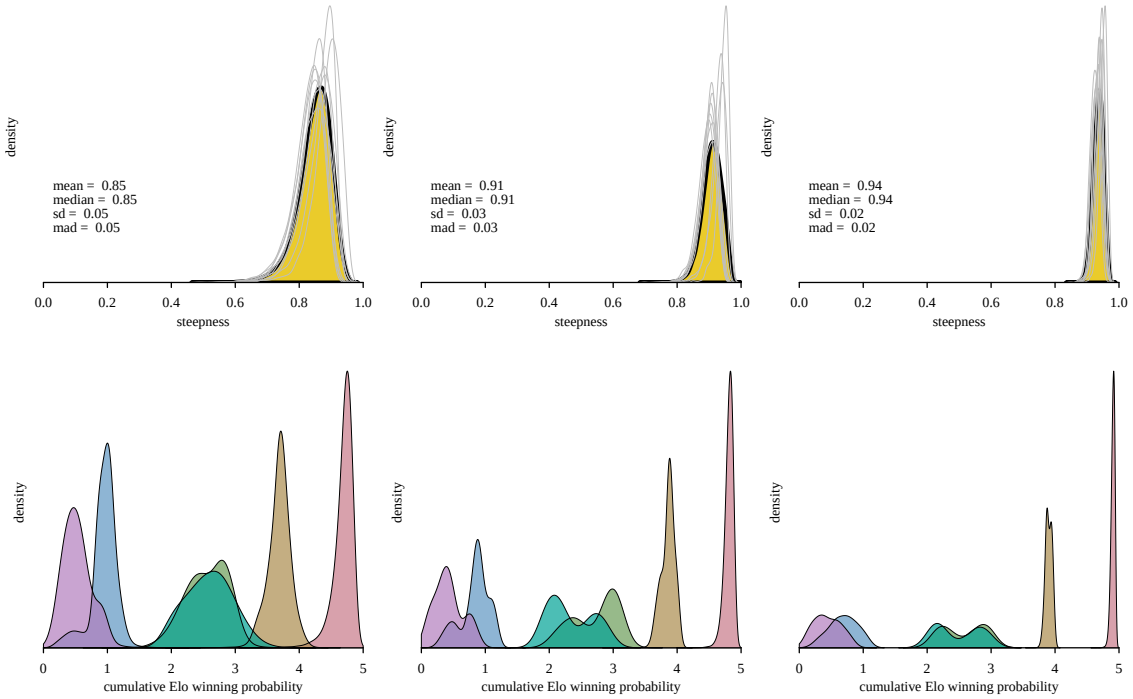
```
set.seed(123)
# generate matrices
m1 <- simple_steep_gen(n_ind = 6, n_int = 40, steep = 0.9)$matrix
m2 <- m1 * 2
m5 <- m1 * 5
# calculate steepness
r1 <- elo_steepness_from_matrix(mat = m1, n_rand = 10, cores = 4)
r2 <- elo_steepness_from_matrix(mat = m2, n_rand = 10, cores = 4)
r5 <- elo_steepness_from_matrix(mat = m5, n_rand = 10, cores = 4)
```

```
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)
```



Bayesian David's scores

Using David's scores instead of Elo-rating-based cumulative 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[2].

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

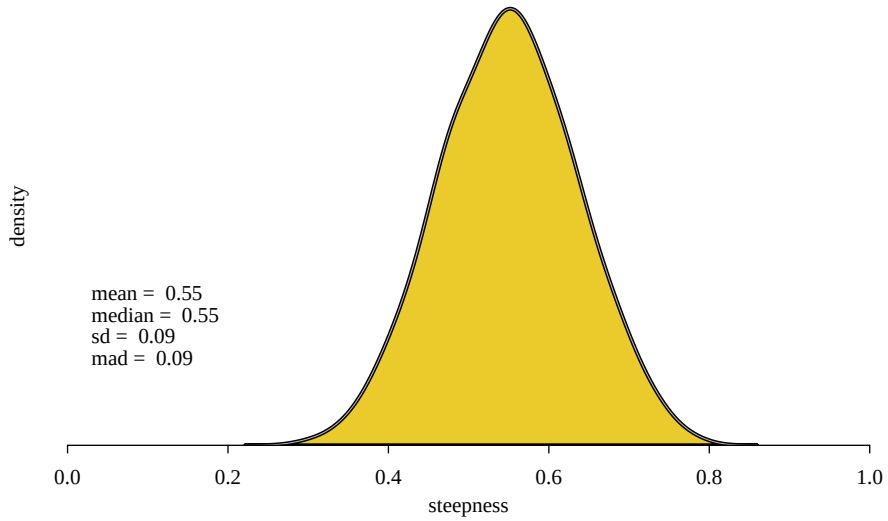
Given the results of the method comparison in the paper, I would recommend not to interpret the steepness value (based on Bayesian David's scores) of this exercise as meaningful given that there are still a substantial amount of unknown relationships in this data set.

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

```
david_res <- davids_steepness(mat, refresh = 0)
```

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.

```
plot_steepness(david_res)
```



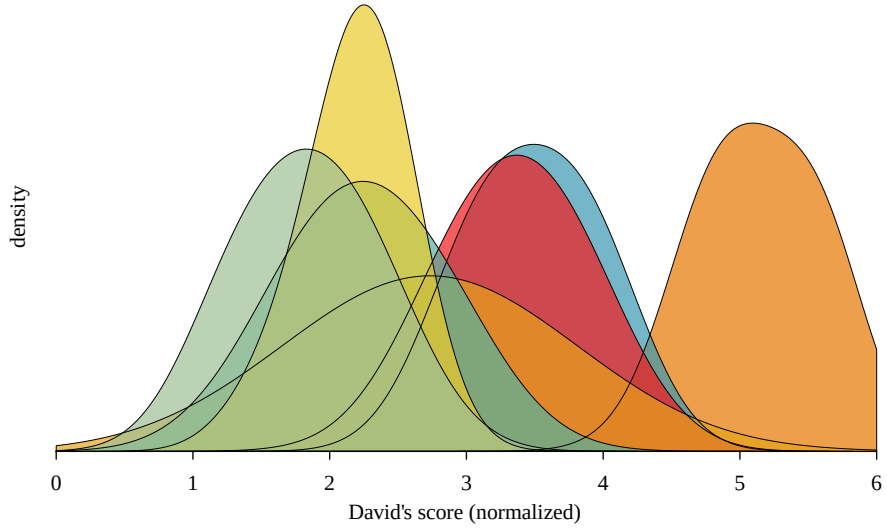

```
summary(david_res)
#> steepness based on Bayesian David's scores
#> ------------------------------------------
#> total number of posterior samples generated: 4000
#> number of samples with issues: 0
#> matrix with 118 interactions between 7 individuals
#> 16.9 interactons per individual
#> proportion 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 0.40 and 0.70
```

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

[2] 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

```
round(steepness_precis(david_res), 2)
#>   mean   sd median  mad q045 q250 q750 q955
#> 1 0.55 0.09   0.55 0.09  0.4 0.49 0.61  0.7
```

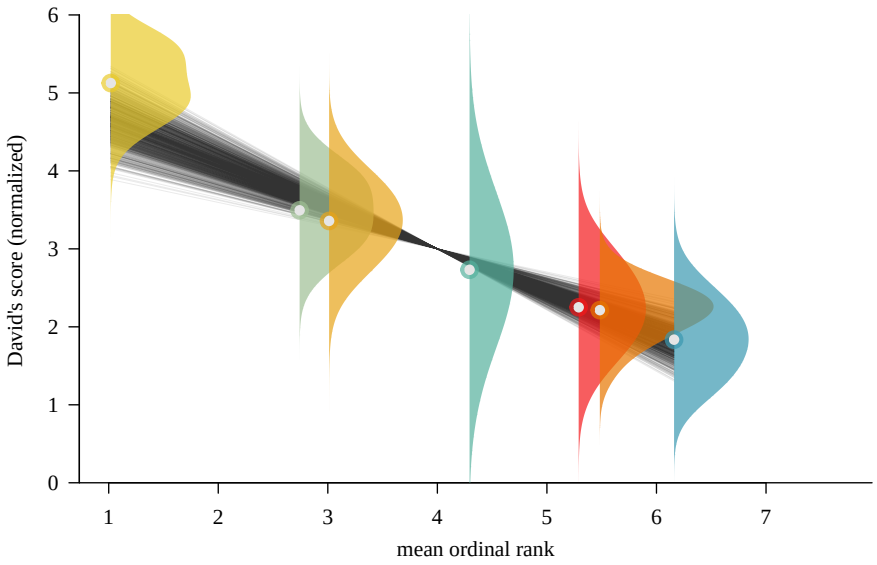
```
plot_scores(david_res)
```




```
scores(david_res)
```

id	mean	sd	median	mad	q045	q955
a	5.142	0.420	5.126	0.500	4.458	5.793
b	3.490	0.460	3.494	0.539	2.739	4.233
c	3.355	0.503	3.358	0.537	2.504	4.183
d	2.733	0.866	2.732	0.885	1.255	4.209
e	2.258	0.551	2.251	0.584	1.343	3.171
f	2.184	0.357	2.214	0.341	1.497	2.733
g	1.837	0.481	1.836	0.538	1.060	2.645

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


David's score (normalized)
0
1
2
3
4
5
6
1
2
3
4
5
6
7
mean ordinal rank