method evaluation

Christof Neumann

2022-06-22

Contents

This document illustrates the analyses steps from *Extending Bayesian Elo-rating to quantify the steepness of dominance hierarchies* (https://doi.org/10.1101/2022.01.28.478016). Note that the comparative examples are covered in a separate document (vignette *phylogenetic examples*). It was built using `EloSteepness` (version 0.4.0) and `EloSteepness.data` (version 0.9.4).

Since running the entire analysis is very time consuming this document uses a subset of the artificial data set to speed things up for demonstration purposes.

```
library(EloSteepness.data)
library(EloSteepness)
```

1 Obtain steepness values for analysis

This first section describes the steps to obtain all the steepness values for the analyses. The actual analyses of these data are presented in the subsequent sections.

```
# load progress tracker and full set of matrices
#   (use empirical_progress and empirical_matlist for analysing the empirical data)
data("artificial_progress", package = "EloSteepness.data")
data("artificial_matlist", package = "EloSteepness.data")
```

For the sake of this *document*, we don't run the full data set here but rather use a small subset to speed things up. Specifically, we use five small (less than 100 interactions) matrices among the artificial data sets (instead of 1,000 data sets in the main analysis). We even further thin out this demo data set by restricting the number of the reduction steps that are needed for the second analysis further down: Instead of using all (up to) 12 reduction steps per matrix, we only use 3.

```
# select a set of 5 small data sets for this document
sel <- unlist(lapply(artificial_matlist[grepl("step00", names(artificial_matlist))], sum))
sel <- names(sel[sel < 100])[1:5]
# get the all relevant matrices (initial matrices plus their reduced versions)
pat <- paste0(paste0(sel, "|", gsub("step00", "step02", sel), "|",
                     gsub("step00", "step04", sel), "|",
                     gsub("step00", "step06", sel)), collapse = "|")
res <- artificial_progress[grepl(pat, x = artificial_progress$mname), ]
mats <- artificial_matlist[names(artificial_matlist) %in% res$mname]
```

```
length(mats)
#> [1] 20
```

This leaves us with 20 matrices (5 initial data sets + 5 initial data sets * 3 reduction steps).

If you want to run this analysis with larger number of artificial matrices or with the empirical data, use one of the following code chunks:

```
# 9 artificial matrices
# res <- artificial_progress[grepl("matrix000[1-9]_", artificial_progress$mname), ]
# res <- res[grepl("step00|step03|step06|step09|step12", x = res$mname), ]
# mats <- artificial_matlist[names(artificial_matlist) %in% res$mname]
```

```
# if you want to run this with empirical data, you could filter for the elephant data
#   sets by Archie et al (simply because it also adds up to a similar reduction of
#   data to be processed)
# data("empirical_progress", package = "EloSteepness.data")
# data("empirical_matlist", package = "EloSteepness.data")
# res <- empirical_progress[grepl("archie2006", empirical_progress$mname), ]
# mats <- empirical_matlist[names(empirical_matlist) %in% res$mname]
```

Or, if you want to run the entire set of artificial or empirical data sets:

```
# artificial data
# res <- artificial_progress
# mats <- artificial_matlist

# empirical data
# res <- empirical_progress
# mats <- empirical_matlist
```

Now we loop through each combination of matrix and method to obtain the various steepness estimates (but we ignore the implementation using the original code by Goffe et al). Also note that we set a fairly low number of iterations and randomizations (`iter = 1000` and `n_rand = 3`) to speed up the code execution. As a result of the low number of iterations you might see warnings about low effective sample sizes, which should disappear if you increase the number of iterations.

```
# for the evaluation of the different methods we ignore the original implementation
#   of Goffe et al's Elo-rating and use our slightly modified implementation
#   (see figure A4 in manuscript)
res <- droplevels(res[res$method != "elo_bayes_ori", ])
rownames(res) <- NULL
table(res$method)
#>
#>          ds_dij          ds_pij        ds_bayes         elo_rpt elo_bayes_fixed
#>              20              20              20              20              20
#>          upward
#>              20

res$steep_val <- NA # store the generated steepness value
```

```
for (i in seq_len(nrow(res))) {
  meth <- res$method[i]
  mname <- as.character(res$mname[i])
  mat <- mats[[mname]]

  if (meth == "elo_bayes_fixed") {
```

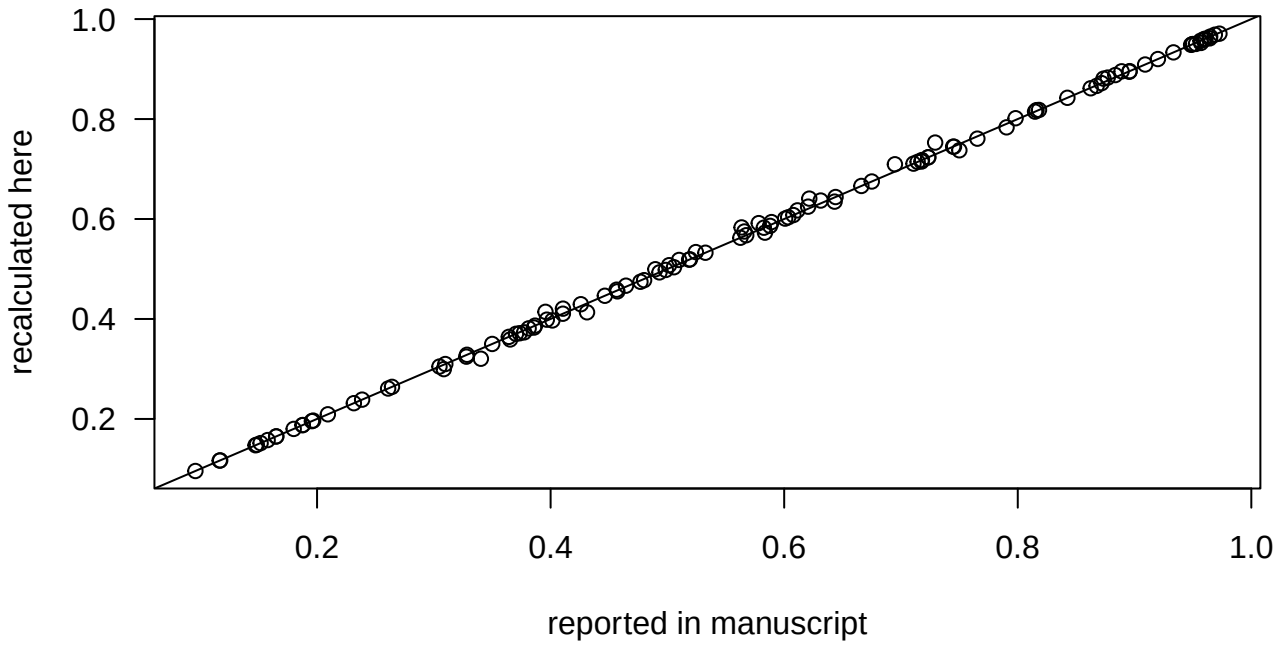
```
    # sped up code:
    xres <- elo_steepness_from_matrix(mat = mat, refresh = 0, cores = 2, iter = 1000, n_rand = 3)
    # original analysis:
    # xres <- elo_steepness_from_matrix(mat = mat, refresh = 0, cores = 4)
    res$steep_val[i] <- median(xres$steepness)
    rm(xres)
  }

  if (meth == "ds_bayes") {
    xres <- davids_steepness(mat = mat, refresh = 0)
    res$steep_val[i] <- median(xres$steepness)
    rm(xres)
  }

  if (meth == "elo_rpt") {
    res$steep_val[i] <- repeatability_steepness(mat)$steepness
  }

  if (meth == "ds_dij") {
    res$steep_val[i] <- steepness(mat, Dij = TRUE)[1]
  }

  if (meth == "ds_pij") {
    res$steep_val[i] <- steepness(mat, Dij = FALSE)[1]
  }

  if (meth == "upward") {
    res$steep_val[i] <- upward_steepness(mat)
  }

  if (interactive()) cat(nrow(res) - i, "\r")
  rm(meth, mname, mat)
}
```

Now we add the numbers that we reported in the manuscript. This just serves as an initial sanity check to see whether the results we produced here match.

```
# the steepness value reported in the results of the manuscript
reported <- EloSteepness.data::removal_global
res$reported <- NA

for (i in seq_len(nrow(res))) {
  res$reported[i] <- reported[reported$mat_name == res$mname[i] &
                                reported$method == res$meth[i], "steep_val"]
}
```

This figure now contains 120 pairs of values (if you followed the initial reduction of data for this demonstration). And these 120 pairs are distributed across the six methods. Ideally, all pairs of values should be identical.

```
plot(res$reported, res$steep_val, las = 1,
     xlab = "reported in manuscript", ylab = "recalculated here")
abline(0, 1)
```



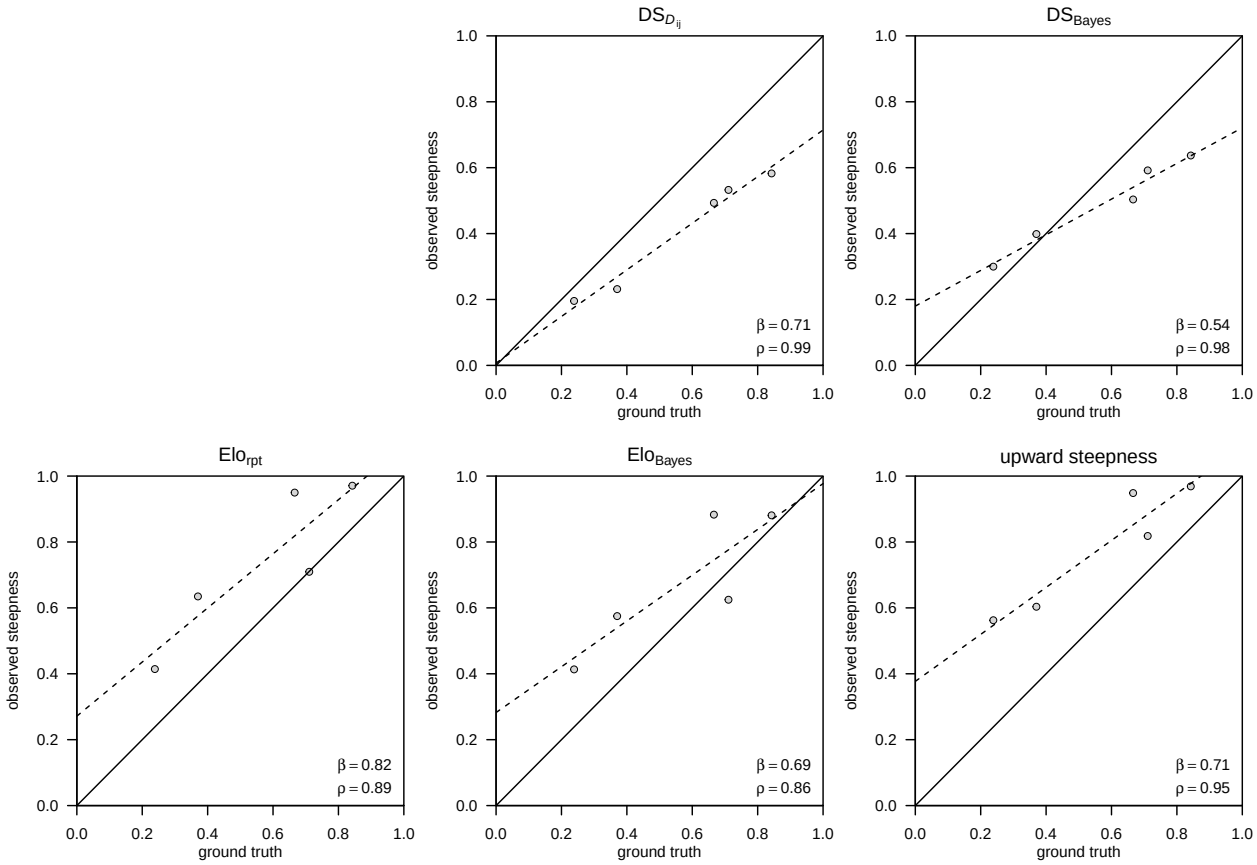
2 Recovering ground truth

To replicate figures 4 through 6 we need to obtain ground truth. As described in the manuscript the approach we took here differed between the empirical and artificial networks. For this demonstration we use the approach where we assume ground truth to be the DS-based (P_{ij}) steepness because we can apply this to both, artificial and empirical data. But first, we need to wrangle the data a bit.

```
pdata <- res
# take only the initial matrices
pdata$step <-  regmatches(pdata$mname, regexpr("step[0-1][0-9]$", pdata$mname))
pdata <- pdata[pdata$step == "step00", ]
# add unknown relationships
pdata$prunk <- unlist(lapply(mats[as.character(pdata$mname)],
                             function(x) EloRating::prunk(x)[1]))
# add reference steepness (i.e. ground truth)
pdata$ref_steep <- unlist(lapply(mats[as.character(pdata$mname)],
                                 function(x) EloRating::steepness(x, Dij = FALSE)[1]))
# indicator for whether the data are empirical or not
#   (set to TRUE if you used empirical data)
pdata$empirical <- FALSE
```

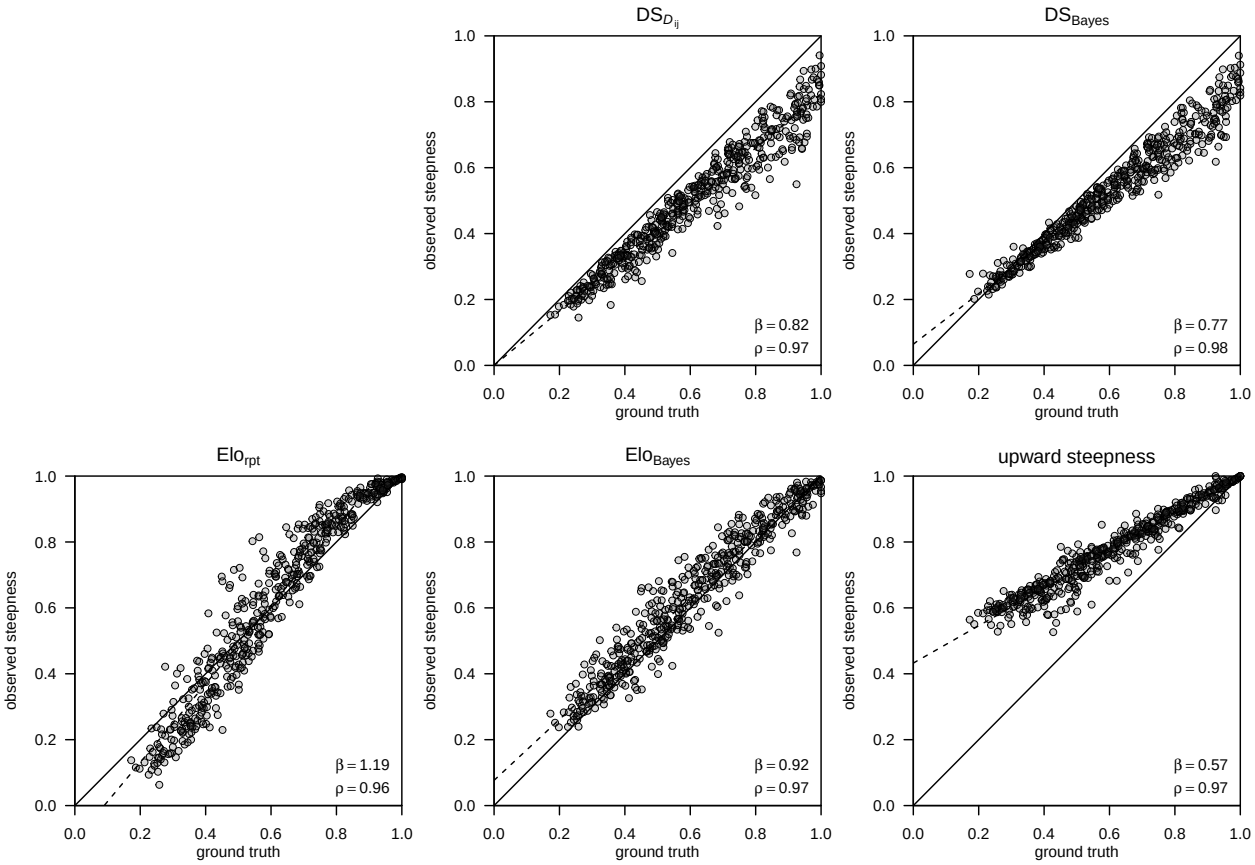
In the manuscript, we used only matrices with not more than 5% of unknown relationships (because of the problems DS steepness has with unknown relationships). Here (for demonstration purposes), we are a much more liberal (using 50%, `prunkcut = 0.5`). The reason is that the small data sets we used here are (partly) fairly empty.

```
benchmark_plot(pdata = pdata, empirical = FALSE, prunkcut = 0.5, benchmark = "pij")
```



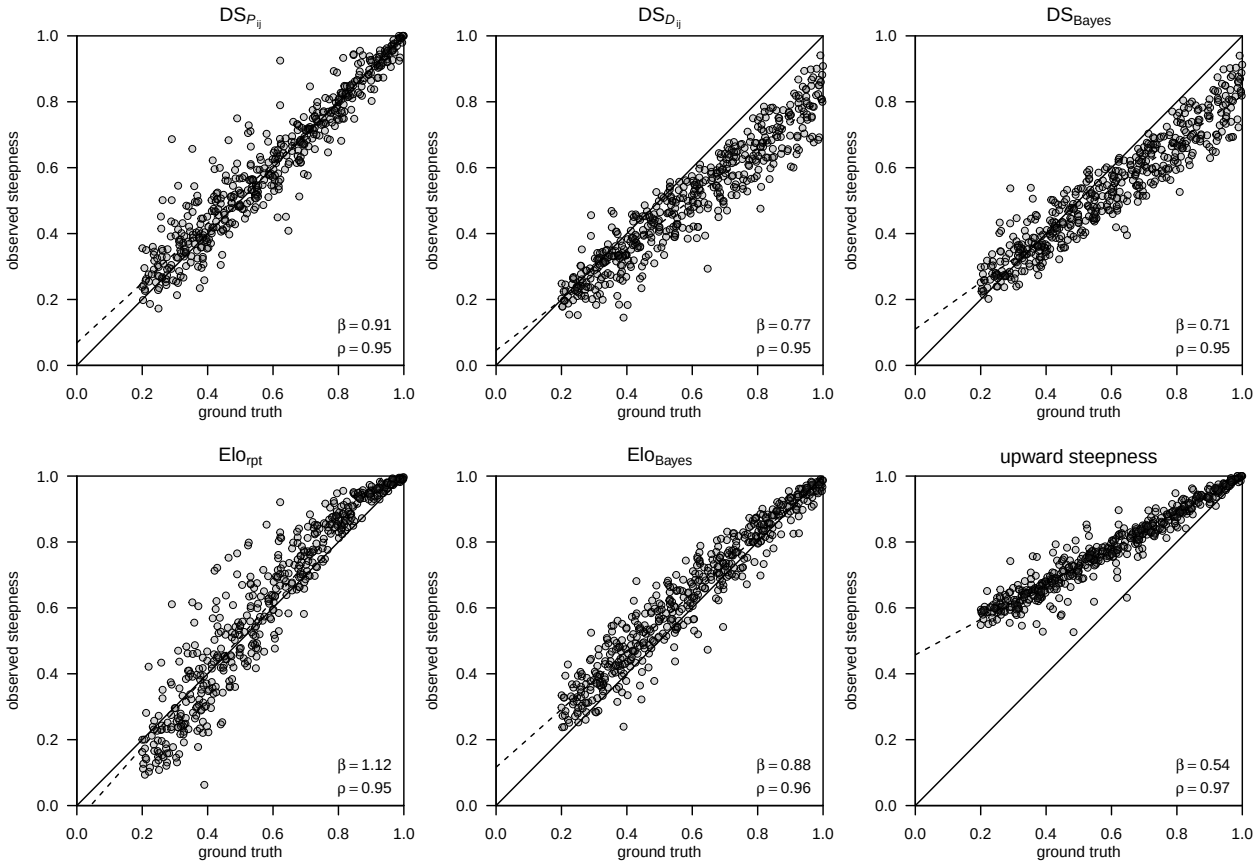
If you wanted to replicate the figures as they appear in the manuscript, use the *full data set* (which is included in the package).

```
data("removal_global")
benchmark_plot(pdata = removal_global, empirical = FALSE, prunkcut = 0.05, benchmark = "pij")
```



Or with the artificial data where we *know* ground truth:

```
data("removal_global")
benchmark_plot(pdata = removal_global, empirical = FALSE, prunkcut = 0.05, benchmark = "input")
```



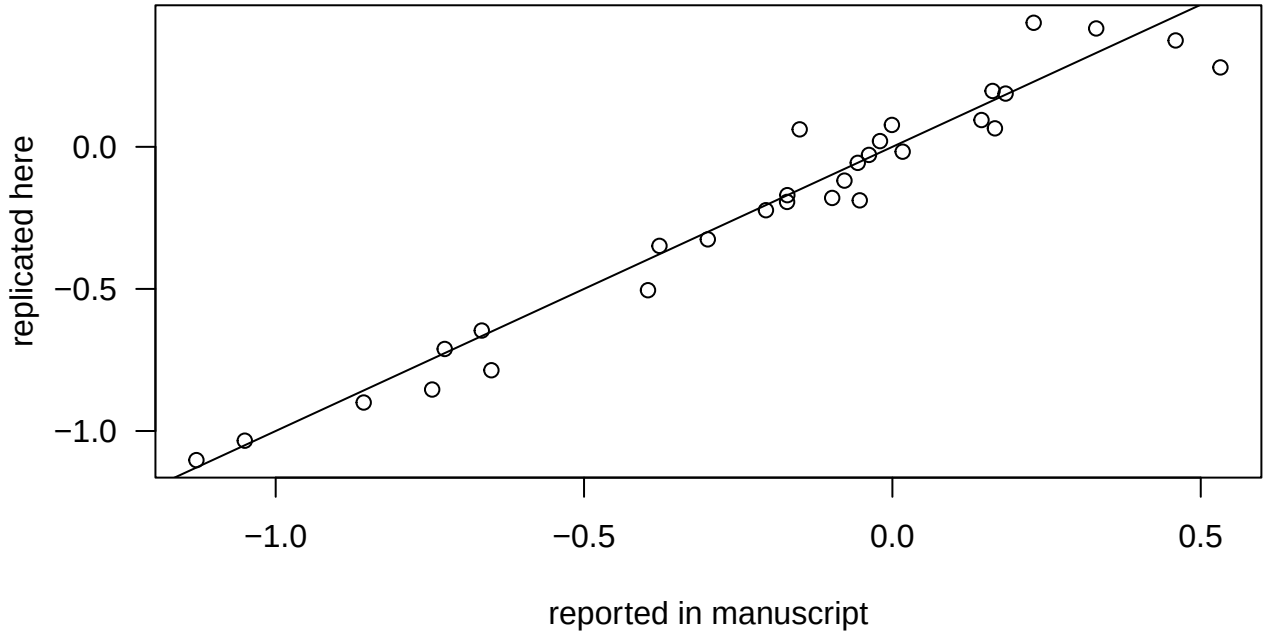
3 Handling sparse data sets/removal experiment

Here we keep going with the small set of results we created above. Now we need to calculate per method and data set the slope of the regression between steepness and unknown relationships.

```
pdata <- res
# data set name
pdata$dataset <- unlist(lapply(strsplit(as.character(pdata$mname), "_@_"), function(x)x[1]))
# add unknown relationships
pdata$prunk <- unlist(lapply(mats[as.character(pdata$mname)],
                             function(x) EloRating::prunk(x)[1]))

# calculate the slopes
xres <- expand.grid(dataset = unique(pdata$dataset), method = unique(pdata$method))
xres$slope_prunk <- NA

for (i in seq_len(nrow(xres))) {
  aux <- pdata[pdata$dataset == xres$dataset[i] & pdata$method == xres$method[i], ]
  xres$slope_prunk[i] <- coef(lm(steep_val ~ prunk, data = aux))[2]
}
```

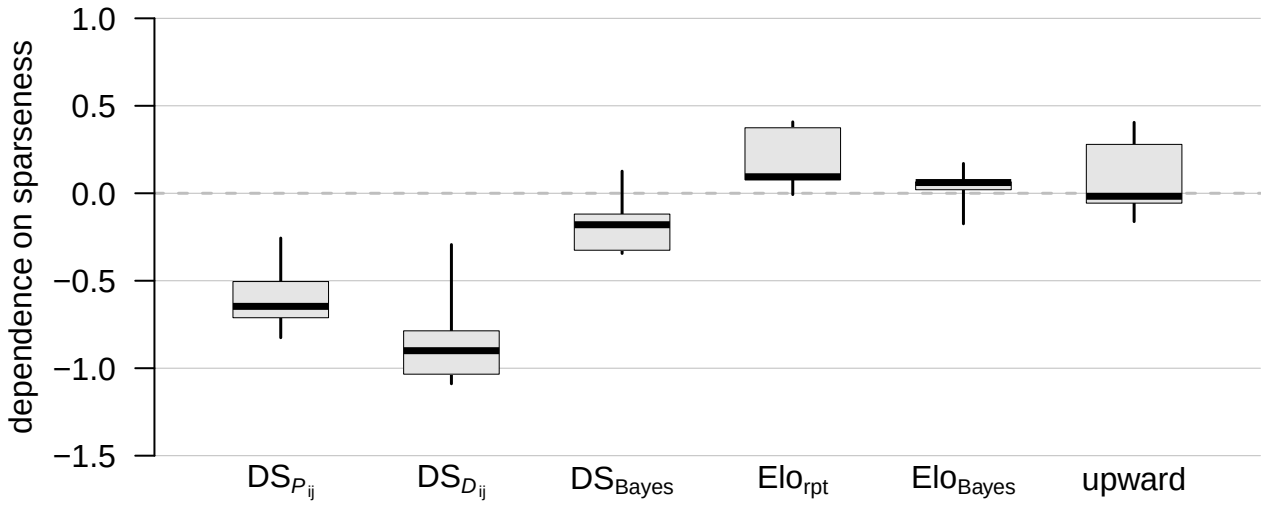
Now run again a small sanity check to see whether these 'manually' obtained slopes match the ones we report. As above, ideally these paired values (one generated here in this document, one from our complete analysis) should be identical. The reason this sanity check is a bit noisier compared to the one above is that the slopes generated here are based on four values (i.e., four reduction steps) per matrix and method (rather than up to 13 for the complete analysis).

```r
data("removal_slopes")
xres$slope_prunk_reported <- NA
for (i in seq_len(nrow(xres))) {
  xres$slope_prunk_reported[i] <-
    removal_slopes$slope_prunk[removal_slopes$data_set == as.character(xres$dataset[i]) &
                               removal_slopes$method == as.character(xres$method[i])]
}

plot(xres$slope_prunk_reported, xres$slope_prunk, las = 1,
     xlab = "reported in manuscript", ylab = "replicated here")
abline(0, 1)
```



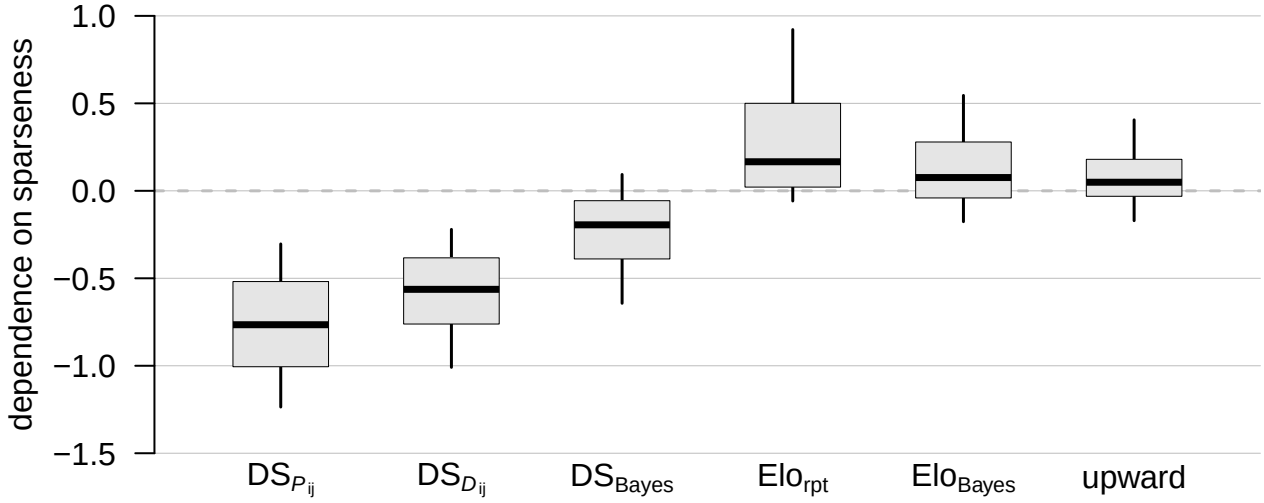
To generate the actual results plot use the following code:

```r
# indicator for whether the data are empirical or not
#   (set to TRUE if you used empirical data)
xres$empirical <- FALSE

slope_box(pdata = xres, empirical = FALSE)
```



And again, if you want to replicate the full figures 7 and 8, just use the *full data set*.

```
data("removal_slopes")
slope_box(pdata = removal_slopes, empirical = FALSE)
```




```
slope_box(pdata = removal_slopes, empirical = TRUE)
```