method evaluation

This document illustrates the analysis steps. 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)
```

Obtain steepness with different algorithms

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

# select a set of 9 data sets for this document
# each of these 9 data sets now is represented by up to 12 matrices
#   (initial plus reduced matrices with interactions removed)
# we still remove further data here and use only reduction steps 0 (the initial matrix)
#   step 3, 6, 9 and 12
res <- artificial_progress[grepl("matrix000[2-9]_", 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 (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]

# 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 manuscri
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          upward
#>              32              32              32              32              32              32

res$steep_val <- NA # 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") {
```

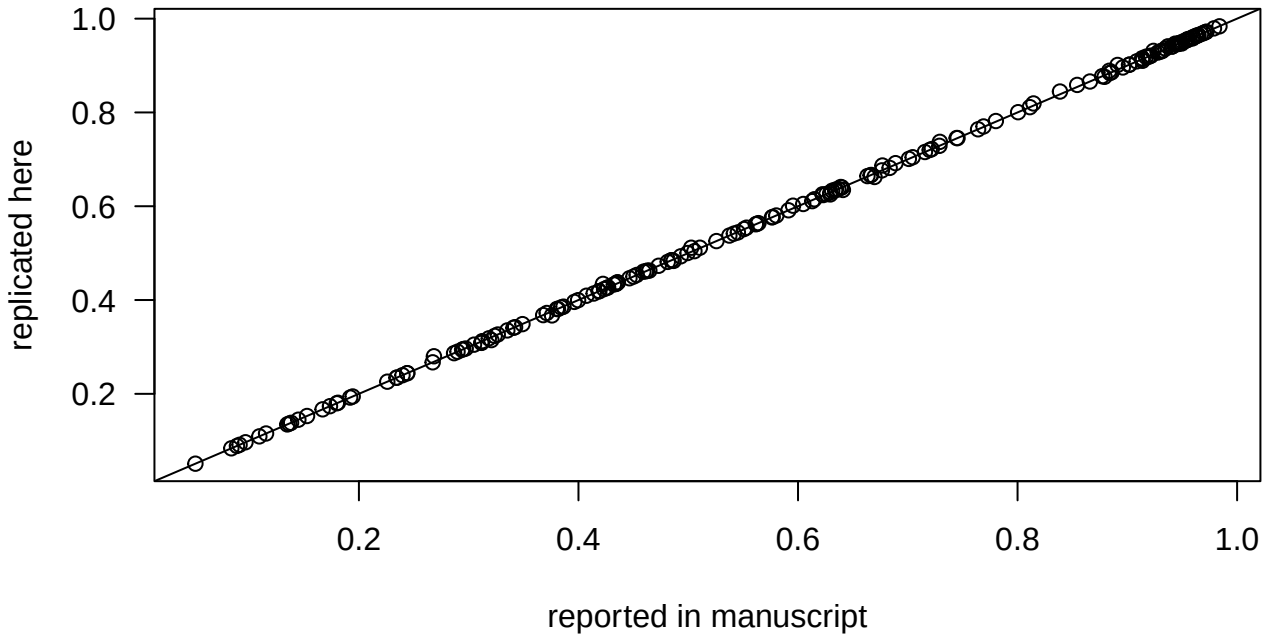
```
    xres <- elo_steepness_from_matrix(mat = mat, refresh = 0, cores = 4)
    res$steep_val[i] <- median(xres$steepness)
    rm(xres)
  }

  if (meth == "ds_bayes") {
    xres <- davids_steepness(mat = mat, refresh = 0, cores = 4)
    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)
  }

  # cat(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.

```
# res <- readRDS("~/ownCloud/replication_pack.rds")
# res <- droplevels(res)
```

```
# 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"]
}
```

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



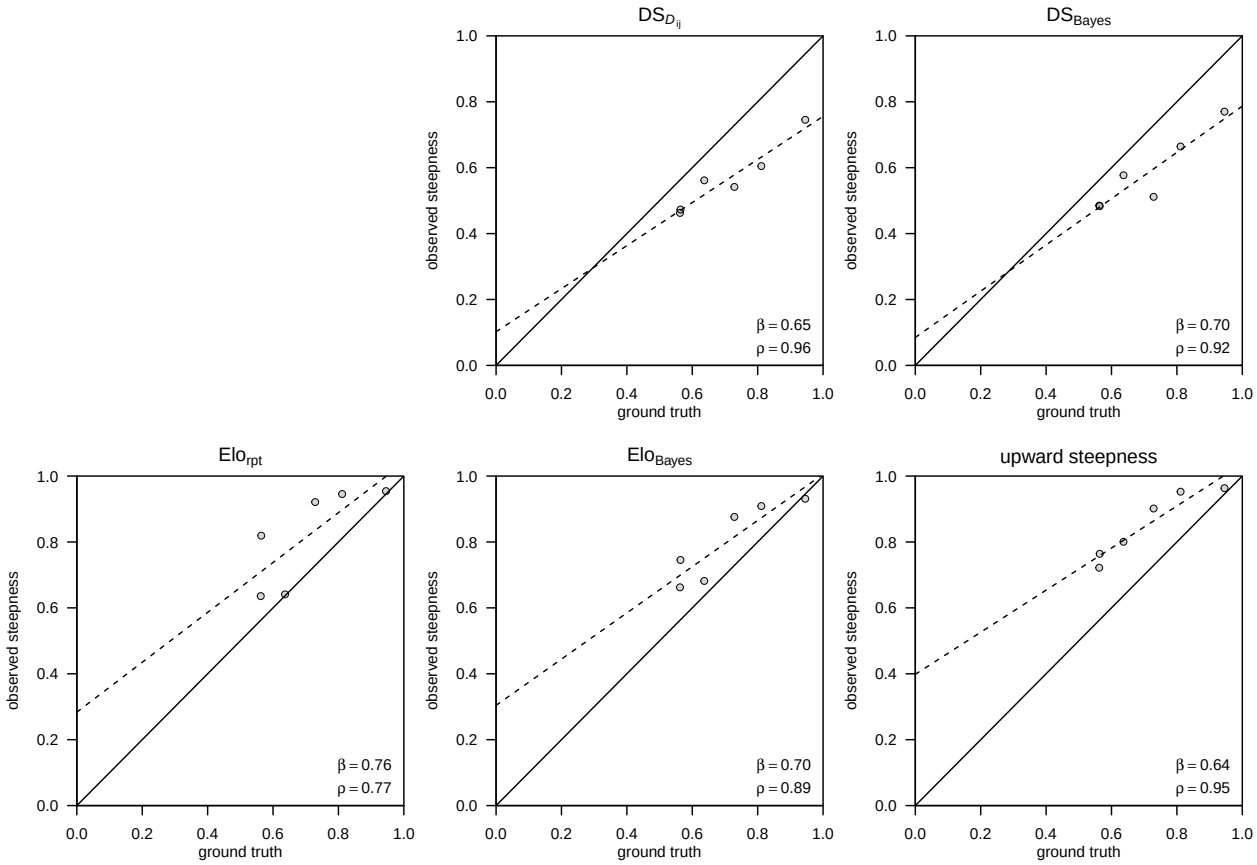
Recovering ground truth

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
```

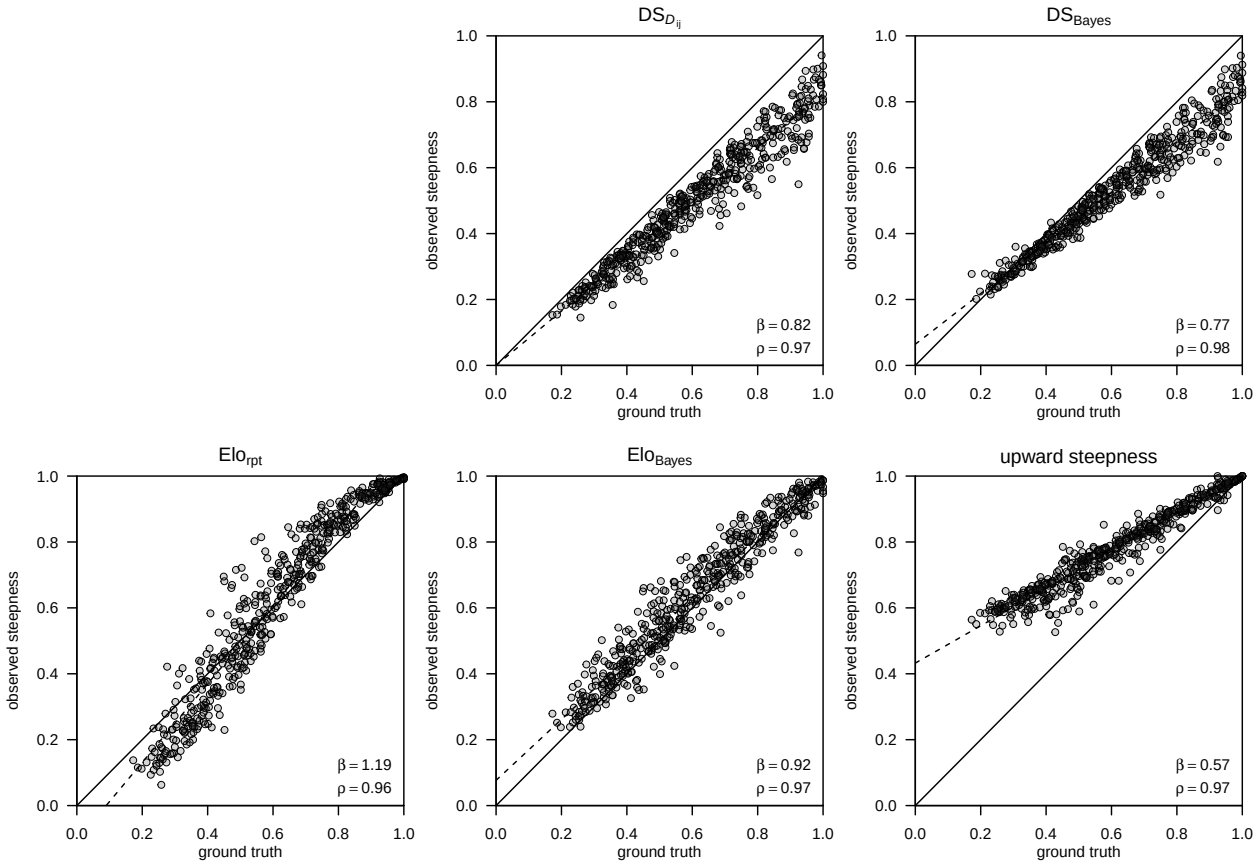
In the manuscript, we used only matrices with not more than 5% of unknown relationships (because of the problems DS has with unknown relationships). Here (for demonstration purposes), we are a bit more liberal (using 20%, `prunkcut = 0.2`).

```
benchmark_plot(pdata = pdata, empirical = FALSE, prunkcut = 0.2, benchmark = "pij")
```



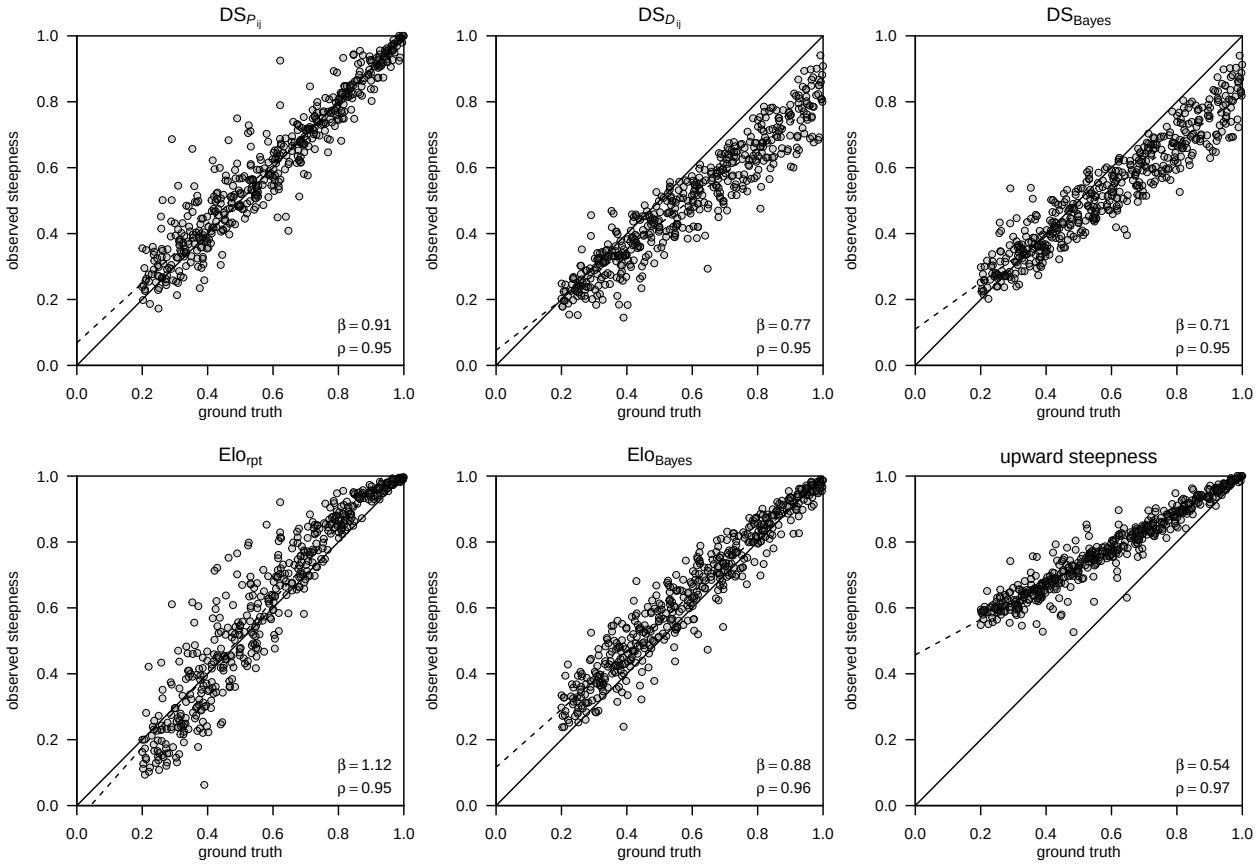
If you wanted to replicate the figures as they appear in the manuscript, use the full *data set*.

```
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")
```



Handling sparse data sets/removal experiment

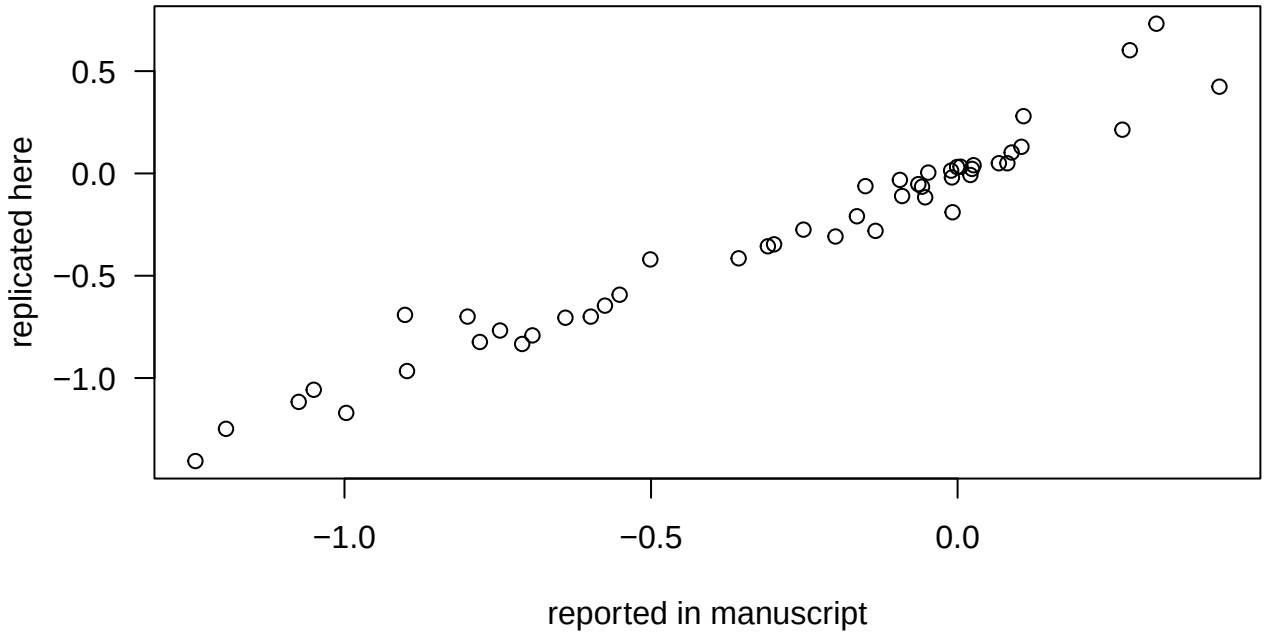
Here we keep going with the small set of results we created above. Now we need to calculate per method and data set the slope of the regression between steepness and unknown relationships.

```
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]
}
```

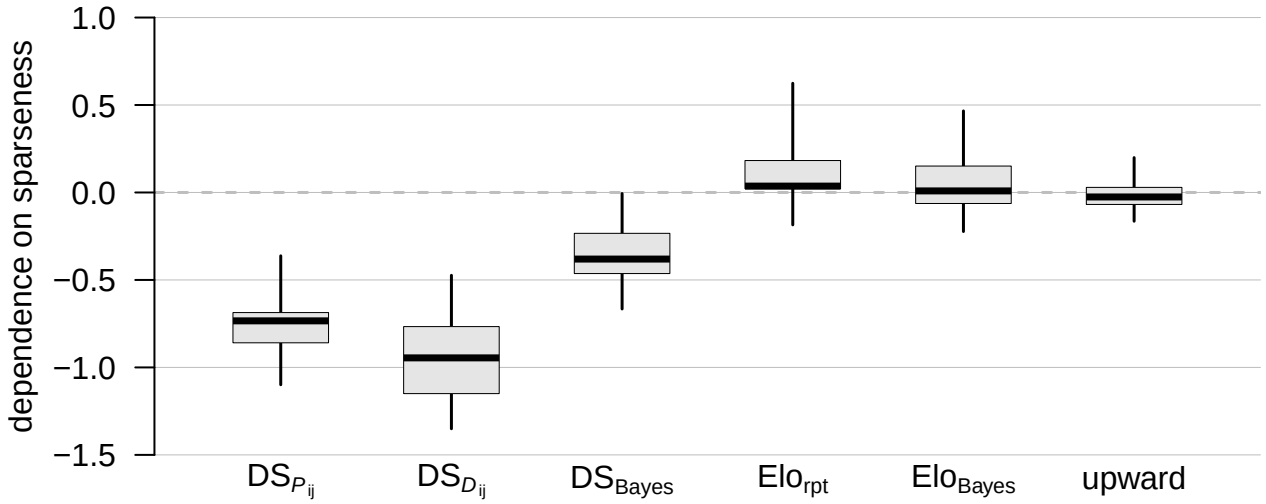
Now run again a small sanity check to see whether these 'manually' obtained slopes match the ones we report.

```
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$data_set == as.character(xr
                                                             removal_slopes$method == as.character(xr
}

plot(xres$slope_prunk_reported, xres$slope_prunk, las = 1, xlab = "reported in manuscript", ylab = "rep
```

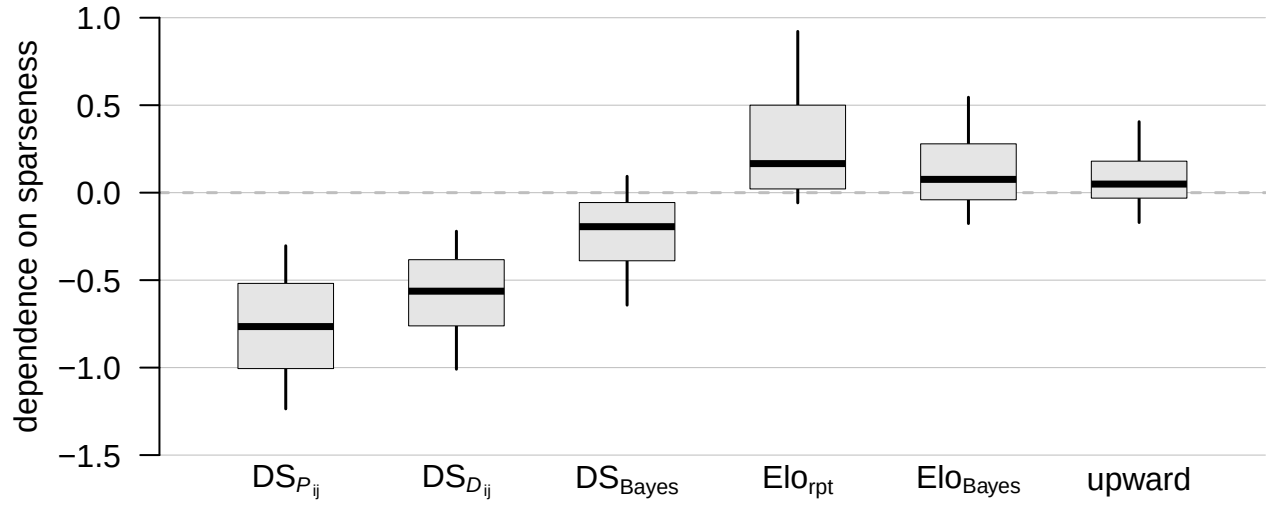



```
# indicator for whether the data are empirical or not
#   (set to TRUE if you used empirical data)
xres$empirical <- FALSE

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)
```