Package ‘EloChoice’

July 21, 2025

Type Package

Title Preference Rating for Visual Stimuli Based on Elo Ratings

Version 0.29.4

Date 2019-07-03

Author Christof Neumann

Maintainer Christof Neumann <christofneumann1@gmail.com>

Description Allows calculating global scores for characteristics of visual stimuli as assessed by human raters. Stimuli are presented as sequence of pairwise comparisons ('contests'), during each of which a rater expresses preference for one stimulus over the other (forced choice). The algorithm for calculating global scores is based on Elo rating, which updates individual scores after each single pairwise contest. Elo rating is widely used to rank chess players according to their performance. Its core feature is that dyadic contests with expected outcomes lead to smaller changes of participants' scores than outcomes that were unexpected. As such, Elo rating is an efficient tool to rate individual stimuli when a large number of such stimuli are paired against each other in the context of experiments where the goal is to rank stimuli according to some characteristic of interest. Clark et al (2018) <doi:10.1371/journal.pone.0190393> provide details.

License GPL (>= 3)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, psychotools, Rdpack

Suggests xtable, knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 6.1.1

RdMacros Rdpack

URL https://github.com/gobbios/EloChoice

BugReports https://github.com/gobbios/EloChoice/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-07-04 08:40:02 UTC

Contents

`elochoice` *Elo-ratings for pairwise comparisons of visual stimuli*

Description

Elo-ratings for pairwise comparisons of visual stimuli

Usage

```
elochoice(winner, loser, kval = 100, startvalue = 0, runs = 1, normprob = FALSE)
eloint(winner, loser, allids, kval, startvalues, runs)
elointnorm(winner, loser, allids, kval, startvalues, runs)
```

Arguments

`winner` character, vector with the IDs of the winning (preferred) and losing (not preferred) stimuli

`loser` character, vector with the IDs of the winning (preferred) and losing (not preferred) stimuli

`kval` numeric, k-value, which determines the maximum number of points a stimulus' rating can change after a single rating event, by default 100

`startvalue` numeric, start value around which ratings are centered, by default 0

`runs` numeric, number of randomizations

`normprob` logical, by default `FALSE`, which indicates that a logistic approach is taken for calculating winning probabilities (see Elo 1978). Alternatively (`TRUE`), such that winning probabilities are calculated from a normal distribution

`startvalues` numeric, start value around which ratings are centered, by default 0

`allids` internal, character of all stimulus IDs in the data set

Details

`elochoice()` is the workhorse function of the package, which wraps up all the calculations for obtaining Elo-ratings and the information for the reliability index

`eloint()` and `elointnorm()` are internal functions (which `elochoice()` makes use of) that do most of the calculations, but are usually not directly addressed by the user.

Value

an object of class `elochoice`, i.e. a list with the following items

`ratmat` numeric matrix with final ratings for each stimulus, one row per randomization

`decmat` logical matrix showing for each randomization (row) and each single rating event (column) whether or not there was an expectation for that trial, i.e. whether the two stimuli's ratings differed before the rating

`upsmat` logical matrix showing for each randomization (row) and each single rating event (column) whether or not the outcome of a trial was in the direction of the expectation, i.e. whether or not the higher rated stimulus won

`wgtmat` numeric matrix showing for each randomization (row) and each single rating event (column) the absolute difference in ratings before the rating event

`misc` various information

`ov` data set overview, i.e. in how many trials was a stimulus involved and how many trials did each stimulus win and lose

`ias` character matrix, with the original sequence of rating events

Author(s)

Christof Neumann

References

Elo AE (1978). *The rating of chess players, past and present*. Arco, New York.

Clark AP, Howard KL, Woods AT, Penton-Voak IS, Neumann C (2018). “Why rate when you could compare? Using the ’EloChoice’ package to assess pairwise comparisons of perceived physical strength.” *PloS one*, **13**(1), e0190393. doi: 10.1371/journal.pone.0190393.

Examples

```
data(physical)
set.seed(123)
res <- elochoice(winner = physical$Winner, loser = physical$Loser, runs = 100)
summary(res)
ratings(res, show = NULL, drawplot = TRUE)
```

makepairwise *transform preference data*

Description

transform preference data into paircomp format (paircomp)

Usage

```
makepairwise(winner, loser, rater)
```

Arguments

winner	character, vector with the IDs of the winning (preferred) stimuli
loser	character, vector with the IDs of the losing (not preferred) stimuli
rater	character, vector of rater identity

Value

object of class paircomp

Author(s)

Christof Neumann

See Also

psychotools

Examples

```
w <- c("B", "A", "E", "E", "D", "D", "A", "D", "E", "B", "A", "E", "D", "C", "A")
l <- c("C", "C", "C", "D", "B", "C", "E", "A", "B", "D", "E", "B", "E", "D", "C")
raters <- rep(letters[1:3], 5)
makepairwise(w, l, raters)
```

`physical` *Physical strength of males*

Description

Physical strength of males

Usage

```
data(physical)
```

Format

4592 pairwise comparisons (contests) between 82 stimuli (average of 112 appearances per stimulus). 56 raters came to the lab and made 82 judgements each. They were asked to choose which image of a pair of stimulus images depicted the physically stronger looking male.

`Date` Date of the rating

`Winner` Winner of the interaction

`Loser` Loser of the interaction

`raterID` A numeric indicator of rater identity

Source

Andrew Clark

References

Andrew Clark

Examples

```
data(physical)
```

`randompairs` *generate random data of pairwise preference ratings*

Description

generate random data of pairwise preference ratings

Usage

```
randompairs(nstim = 10, nint = 100, reverse = 0.1, skew = FALSE)
```

Arguments

`nstim` numeric, number of stimuli, must be less than 2,602

`nint` numeric, number of paired ratings to be created

`reverse` numeric, proportion of ratings that go against the default preference, see below for details

`skew` logical, by default `FALSE`, see below for details

Details

The default preference for a given pair is given by their alphanumerical order. E.g. *A* is preferred over *M*, and *kf* over *kz*. The `reverse=` argument specifies the proportion of ratings that go against this default order.

The number of appearances of a given stimulus in the data set is by default determined by uniform sampling of individual stimuli, i.e. all stimuli will roughly appear equally often in a data set. If a somewhat more realistic (i.e. unbalanced) distribution is desired, the argument `skew=TRUE` will achieve sampling based on a negative binomial distribution.

Value

`data.frame` with winner and loser column. An additional column (`index`) serves as an index for the sequence in which the trials occurred.

Author(s)

Christof Neumann

Examples

```
# a relatively balanced data set
xdata <- randompairs(20, 500, skew=FALSE)
table(c(as.character(xdata$winner), as.character(xdata$loser)))
range(table(c(as.character(xdata$winner), as.character(xdata$loser))))

# and a less balanced data set
xdata <- randompairs(20, 500, skew=TRUE)
table(c(as.character(xdata$winner), as.character(xdata$loser)))
range(table(c(as.character(xdata$winner), as.character(xdata$loser))))
```

`raterprog` *reliability with progressive rater inclusion*

Description

reliability with progressive rater inclusion

Usage

```
raterprog(winner, loser, raterID, runs=100, ratershuffle=1, progbar=TRUE, kval=100,
startvalue=0, normprob=FALSE)

raterprogplot(xdata)
```

Arguments

winner	character, vector with the IDs of the winning (preferred) stimuli
loser	character, vector with the IDs of the losing (not preferred) stimuli
raterID	a vector (numeric, character, factor) with rater IDs
runs	numeric, number of randomizations
ratershuffle	numeric, number of times rater order is reshuffled/randomized
progbar	logical, should a progress bar be displayed
kval	numeric, k-value, which determines the maximum number of points a stimulus' rating can change after a single rating event, by default 100
startvalue	numeric, start value around which ratings are centered, by default 0
normprob	logical, by default FALSE, which indicates a logistic approach is taken for calculating winning probabilities (see Elo 1978). Alternatively (TRUE), winning probabilities are calculated from a normal distribution
xdata	results from raterprog

Details

raterprog() calculates reliability, increasing the number of raters to be included in the rating process in a step-wise fashion. In the first (and by default only one) run, the first rater is the one that appears first in the data set, and in subsequent steps raters are added by the order in which they occur. If ratershuffle= is set to values larger than 1, the order in which raters are included is randomized.

raterprogplot() plots the matrix resulting from raterprog(). If ratershuffle= is larger than 1, the average reliability index is plotted alongside quartiles and results from the original rater inclusion sequence.

Note that the function currently only calculates the weighted version of the reliability index.

Value

a numeric matrix. Rows correspond to number of raters in the data set, while columns reflect the number of times the rater order is reshuffled.

Author(s)

Christof Neumann after suggestion by TF

References

Clark AP, Howard KL, Woods AT, Penton-Voak IS, Neumann C (2018). "Why rate when you could compare? Using the 'EloChoice' package to assess pairwise comparisons of perceived physical strength." *PloS one*, **13**(1), e0190393. doi: 10.1371/journal.pone.0190393.

Examples

```
data("physical")
# limit to 12 raters
physical <- physical[physical$raterID < 14, ]

x <- raterprog(physical$Winner, physical$Loser, physical$raterID, ratershuffle = 1)
raterprogplot(x)

# with multiple orders in which raters are added
x <- raterprog(physical$Winner, physical$Loser, physical$raterID, ratershuffle = 10)
raterprogplot(x)
```

ratings *indiviual stimulus ratings*

Description

get stimulus ratings and/or a summary plot

Usage

```
ratings(x, show = "mean", drawplot = TRUE)
```

Arguments

x	an object of class "elochoice", usually the result of a call to elochoice
show	character, what values should be returned, see below
drawplot	logical, should a plot drawn

Details

If show="original", show="mean" or show="var", a numeric vector is returned which contains either the ratings obtained from the initial/original sequence, the average ratings across all randomizations, or the total variance.

If show="range" or show="all", a matrix is returned that contains either the range of ratings across all randomizations, or all ratings of all randomizations.

If you simply want to create the plot without any rating output being generated, use show=NULL.

If drawplot=TRUE, a plot is created that depicts the values of the ratings obtained from the initial sequence (red), the mean ratings across all randomizations (black) and the range of ratings across all randomizations.

Value

numeric vector or matrix, and/or a plot

Author(s)

Christof Neumann

Examples

```
xdata <- randompairs(nstim = 10, nint = 100)
x <- elochoice(xdata$winner, xdata$loser, runs = 10)

# ratings from the initial sequence
ratings(x, "original", drawplot = FALSE)

# range of ratings across all randomizations
ratings(x, "range", drawplot = FALSE)

# and producing plot
ratings(x, NULL, drawplot = TRUE)
```

`reliability` *calculate reliability-index*

Description

calculate reliability-index of Elo-ratings

Usage

```
reliability(x)
```

Arguments

`x`	elochoice-object, the result of `elochoice`

Value

a data.frame with as many rows as randomizations were run in the original call to `elochoice()`. The first column represents the unweighted and the second the weighted reliability index (*R* and *R'*), which is followed by the total number of trials that contributed to the calculation of the index. Note that this number cannot reach the total number of trials in the data set because at least for the very first trial we did not have an expectation for the outcome of that trial (and such trials do not contribute to the calculation of the reliability index).

Author(s)

Christof Neumann

References

Clark AP, Howard KL, Woods AT, Penton-Voak IS, Neumann C (2018). “Why rate when you could compare? Using the ’EloChoice’ package to assess pairwise comparisons of perceived physical strength.” *PloS one*, **13**(1), e0190393. doi: 10.1371/journal.pone.0190393.

Examples

```
# create data set and calculate ratings (with five randomizations)
xdata <- randompairs(12, 500)
x <- elochoice(xdata$winner, xdata$loser, runs=5)
# extract the reliability values
(u <- reliability(x))
# calculate average reliability index
mean(u$upset)
# and in its weighted form
mean(u$upset.wgt)
```

`singlechoice` *update stimulus ratings after one rating event*

Description

update stimulus ratings after one rating event

Usage

```
singlechoice(val1, val2, k)
```

Arguments

`val1` rating of the preferred stimulus *before* the rating event

`val2` rating of the unpreferred stimulus *before* the rating event

`k` value of *k*-constant, which determines the maximum change of ratings after a single rating event

Value

vector with two values: updated ratings *after* the rating event for preferred and unpreferred stimulus

Author(s)

Christof Neumann

References

Elo AE (1978). *The rating of chess players, past and present*. Arco, New York.

See Also

EloRating

Examples

```
# little change because rating difference is large (positive), i.e. expectation is clear
singlechoice(1200, 500, 100)
# no change because rating difference is very large (positive), i.e. expectation is clear
singlechoice(1500, 500, 100)
# large change because rating difference is small (negative), i.e. expectation is clearly violated
singlechoice(500, 1500, 100)
```

---

summary.elochoice *summarize elochoice object*

Description

summarize elochoice object

Usage

```
## S3 method for class 'elochoice'
summary(object, ...)
```

Arguments

object	an object of class "elochoice", usually the result of a call to elochoice
...	further arguments passed to or from other methods. Nothing relevant in this case.

Author(s)

Christof Neumann

Examples

```
xdata <- randompairs(nstim=10, nint=500)
x <- elochoice(xdata$winner, xdata$loser, runs=5)
summary(x)
```

triplets *calculate ratings from sequence of rating events, allowing for more than two stimuli*

Description

calculate ratings from sequence of rating events, allowing for more than two stimuli

Usage

```
triplets(xdata, winner, runs = 2, startvalue = 0, k = 100,
  progressbar = TRUE, mode = "avg")
```

Arguments

`xdata`	data.frame or matrix with stimulus IDs, each row representing one trial, needs to contain at least two columns
`winner`	numeric vector of the same length as `nrow(xdata)`, indicating which column in `xdata` is the winner/preferred stimulus
`runs`	numeric, the number of times the data set should be randomized
`startvalue`	numeric, initial value of ratings, by default `0`
`k`	numeric, value of *k*-constant
`progressbar`	logical, by default `TRUE`. Should a progress bar be displayed
`mode`	character, either `"avg"` (default) or `"seq"`, see Details section

Details

The `mode="avg"` option considers the losers of the trial as one individual/stimulus, whose rating is averaged. This reflects one rating step for each trial (as for `elochoice()`).

The `mode="seq"` option runs a sequence of interactions *within* a trial, i.e. one rating step for each of the loosing stimuli. E.g. if you have three stimuli, that would be two rating steps. With four stimuli, we would have three steps, etc.

Because of the larger number of rating events with `mode="seq"`, the range of Elo-ratings will be larger as compared to `mode="avg"`. The average values will be the same for both though (start value). See examples...

Also note that this is an experimental function that has not yet been tested thoroughly! In addition, this function calculates winning probabilities in a slightly different way as compared to `elochoice`, i.e. based on normal probabilities (see `elochoice`).

Value

a matrix with ratings

Author(s)

Christof Neumann

Examples

```
data(physical)
y <- round(triplets(physical[, 2:3], winner = rep(1,nrow(physical)), runs = 1))
x <- ratings(elochoice(physical$Winner, physical$Loser, runs = 1), show = "all", drawplot = FALSE)
x <- x[order(names(x))]
plot(x, y)

xdata <- as.matrix(t(sapply(1:500, function(x)sample(letters[1:8], 3))))
xdata <- t(apply(xdata, 1, sort))
winner <- sample(1:3, nrow(xdata), TRUE, prob = c(4, 0.8, 0.1))

x <- triplets(xdata, winner, runs=20, mode="avg")
y <- triplets(xdata, winner, runs=20, mode="seq")

# note different ranges along the axes
plot(colMeans(x), colMeans(y))
range(colMeans(x))
range(colMeans(y))
```

Index