# Package 'EloRating'

July 21, 2025

**Type** Package

**Title** Animal Dominance Hierarchies by Elo Rating

**Version** 0.46.18

**Date** 2024-07-15

**Maintainer** Christof Neumann <christofneumann1@gmail.com>

**Description** Provides functions to quantify animal dominance hierarchies. The major focus is on Elo rating and its ability to deal with temporal dynamics in dominance interaction sequences. For static data, David's score and de Vries' I&SI are also implemented. In addition, the package provides functions to assess transitivity, linearity and stability of dominance networks. See Neumann et al (2011) <doi:10.1016/j.anbehav.2011.07.016> for an introduction.

**License** GPL (>= 2)

**LazyData** true

**RoxygenNote** 7.3.2

**Depends** zoo, sna, network, R (>= 3.2.0)

**Suggests** testthat (>= 2.1.0), knitr, aniDom, rmarkdown

**LinkingTo** Rcpp, RcppArmadillo

**Imports** Rcpp, Rdpack

**NeedsCompilation** yes

**VignetteBuilder** knitr

**Encoding** UTF-8

**RdMacros** Rdpack

**URL** https://github.com/gobbios/EloRating

**BugReports** https://github.com/gobbios/EloRating/issues

**Author** Christof Neumann [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0236-1219>),
Lars Kulik [aut]

**Repository** CRAN

**Date/Publication** 2024-07-15 16:40:02 UTC

# Contents

---

`EloRating-package` *Animal Dominance Hierarchies by Elo Rating*

---

## Description

Provides functions to quantify animal dominance hierarchies. The major focus is on Elo rating and its ability to deal with temporal dynamics in dominance interaction sequences. For static data, David's score and de Vries' I&SI are also implemented. In addition, the package provides functions to assess transitivity, linearity and stability of dominance networks. See Neumann et al (2011) <doi:10.1016/j.anbehav.2011.07.016> for an introduction.

## Author(s)

NA

Maintainer: Christof Neumann <christofneumann1@gmail.com>

## References

Elo, A. E. 1978. The Rating of Chess Players, Past and Present. New York: Arco.

Albers, P. C. H. & de Vries, H. 2001. Elo-rating as a tool in the sequential estimation of dominance strengths. Animal Behaviour, 61, 489-495 (doi:10.1006/anbe.2000.1571).

Neumann, C., Duboscq, J., Dubuc, C., Ginting, A., Irwan, A. M., Agil, M., Widdig, A. & Engelhardt, A. 2011. Assessing dominance hierarchies: validation and advantages of progressive evaluation with Elo-rating. Animal Behaviour, 82, 911-921 (doi:10.1016/j.anbehav.2011.07.016).

## Examples

```
data(adv)
SEQ <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date)
summary(SEQ)
```

## .diffmat *difference matrix*

### Description

difference matrix

### Usage

```
.diffmat(mat)
```

### Arguments

| | |
|---|---|
| mat | square interaction matrix with winner in rows and losers in columns, for example the output from creatematrix |

### Details

helper function for ISI

### Value

a matrix with ranking differences assuming that the matrix reflects the order. This information is contained in the upper triangle of the returned matrix.

### Author(s)

Christof Neumann

### Examples

```
data(bonobos)
EloRating:::.diffmat(bonobos)
```

## .elo.seq_old *calculate Elo ratings*

### Description

calculate Elo ratings from a sequence of dominance interactions

## Usage

```
.elo.seq_old(
  winner,
  loser,
  Date,
  draw = NULL,
  presence = NULL,
  startvalue = 1000,
  k = 100,
  normprob = TRUE,
  init = "average",
  iterate = 0,
  runcheck = TRUE,
  progressbar = FALSE
)
```

## Arguments

| | |
|---|---|
| winner | either a factor or character vector with winner IDs of dyadic dominance interactions |
| loser | either a factor or character vector with loser IDs of dyadic dominance interactions |
| Date | character vector of form "YYYY-MM-DD" with the date of the respective interaction |
| draw | logical, which interactions ended undecided (i.e. drawn or tied)? By default all FALSE, i.e. no undecided interactions occurred. Note that in this case, winner/loser values can be interchanged |
| presence | optional data.frame, to supply data about presence and absence of individuals for part of the time the data collection covered. see details |
| startvalue | the value of Elo ratings of the two individuals that are involved in the first interaction of the overall sequence prior to this interaction. By default set to 1000. See also init |
| k | factor *k* that determines the maximum change in ratings. By default k=100 |
| normprob | logical (by default TRUE). Should a normal curve be assumed for calculating the winning/losing probablities, or a logistic curve. See winprob for details |
| init | character, what Elo rating does an individual have prior to its first interaction. Three options are available: average: individuals always start with the value specified in startvalue. Given stable composition of the group, this also reflects the average Elo rating on each day in that group,<br>bottom: subjects entering at the current lowest Elo value if the lowest value getting lower its getting lower for all subjects which had this lowest values before, it is reflecting that in some species new subjects entering a group at the bottom level "bottom entry"<br>bottom_low: same as bottom but additionally the start values getting after the first interaction lower for all non-interacting subjects and, reflecting that we have at start no knowledge about the subjects this option offers for "bottom entry" |

| | |
|---|---|
| | species the possibility to consider that in a way that those subjects which are not interacting getting lower from start on |
| iterate | not yet implemented |
| runcheck | logical, should several checks regarding data integrety be performed, by default TRUE. See seqcheck |
| progressbar | logical, should progress bars be displayed, by default progressbar=TRUE |

## Details

the presence 'matrix' is actually an object of class data.frame containing information about wether an individual was present on a given day or not. The first column represents the dates, running at least from the date of the earliest interaction until at least the date of the last interaction with one line per day (regardless of whether there were actually interactions observed on each day). Further, each individual is represented as a column in which "1" indicates an individual was present on the row-date and a "0" indicates the individuals absence on this date. NAs are not allowed. See advpres for an example.

## Value

An object of class elo, which is list with 10 items that serves as basis to extract relevant information:

| | |
|---|---|
| mat | a date by ID-matrix with raw Elo ratings |
| lmat | a date by ID-matrix with raw Elo ratings |
| cmat | a date by ID-matrix with raw Elo ratings |
| pmat | a date by ID-matrix with with presence data |
| nmat | a date by ID-matrix containing the number of interactions a given ID was involved in on a given day |
| logtable | details on each single interaction |
| stability | a data.frame containing information about stability (see stab_elo) |
| truedates | vector of class Date covering the ranges of dates in the dataset |
| misc | various |
| allids | a (sorted) character vector with all IDs that occur in the dataset |

## Author(s)

Christof Neumann and Lars Kulik

## References

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

Albers PCH, de Vries H (2001). "Elo-rating as a tool in the sequential estimation of dominance strengths." *Animal Behaviour*, **61**, 489-495. doi:10.1006/anbe.2000.1571.

Neumann C, Duboscq J, Dubuc C, Ginting A, Irwan AM, Agil M, Widdig A, Engelhardt A (2011). "Assessing dominance hierarchies: validation and advantages of progressive evaluation with elo-rating." *Animal Behaviour*, **82**, 911-921. doi:10.1016/j.anbehav.2011.07.016.

## Examples

```
data(adv)
SEQ <- EloRating:::.elo.seq_old(winner=adv$winner, loser=adv$loser, Date=adv$Date)
summary(SEQ)
```

---

`.incon` *number of inconsistencies*

---

## Description

calculate number of inconsistencies

## Usage

```
.incon(mat)
```

## Arguments

| | |
|---|---|
| `mat` | square interaction matrix with winner in rows and losers in columns, for example the output from `creatematrix` |

## Value

integer, the number of inconsistencies in the matrix

## Author(s)

Christof Neumann

## References

de Vries H (1998). "Finding a dominance order most consistent with a linear hierarchy: a new procedure and review." *Animal Behaviour*, **55**, 827-843. doi:10.1006/anbe.1997.0708.

## Examples

```
data(bonobos)
EloRating:::.incon(bonobos)
```

`.sincon` *strength of inconsistencies*

## Description

calculate strength of inconsistencies

## Usage

```
.sincon(mat)
```

## Arguments

| | |
|---|---|
| `mat` | square interaction matrix with winner in rows and losers in columns, for example the output from `creatematrix` |

## Details

helper function for `ISI`

## Value

integer, the summed strength of inconsistencies in the matrix

## Author(s)

Christof Neumann

## References

de Vries H (1998). "Finding a dominance order most consistent with a linear hierarchy: a new procedure and review." *Animal Behaviour*, **55**, 827-843. doi:10.1006/anbe.1997.0708.

## Examples

```
data(bonobos)
EloRating:::.sincon(bonobos)
```

## adv *Dominance sequence from Albers and de Vries (2001)*

### Description

Dominance sequence from Albers and de Vries (2001)

### Usage

```
data(adv)
```

### Format

Fictional example of an interaction sequence, with 33 interactions between 7 individuals.

### References

Albers PCH, de Vries H (2001). "Elo-rating as a tool in the sequential estimation of dominance strengths." *Animal Behaviour*, **61**, 489-495. doi:10.1006/anbe.2000.1571.

### Examples

```
data(adv)
```

## adv2 *Dominance sequence from Albers and de Vries (2001)*

### Description

Dominance sequence from Albers and de Vries (2001) with added information about interaction type and whether interaction ended in a draw

### Usage

```
data(adv2)
```

### Format

Fictional example of an interaction sequence, with 33 interactions between 7 individuals.

### References

Albers PCH, de Vries H (2001). "Elo-rating as a tool in the sequential estimation of dominance strengths." *Animal Behaviour*, **61**, 489-495. doi:10.1006/anbe.2000.1571.

### Examples

```
data(adv2)
```

## advpres *Fictional presence data for Albers and de Vries (2001)*

### Description

Fictional presence data for Albers and de Vries (2001)

### Usage

```
data(advpres)
```

### Format

Fictional example of an interaction sequence, with 33 interactions between 7 individuals.

### References

Albers PCH, de Vries H (2001). "Elo-rating as a tool in the sequential estimation of dominance strengths." *Animal Behaviour*, **61**, 489-495. doi:10.1006/anbe.2000.1571.

### Examples

```
data(advpres)
```

## baboons1 *Baboon dominance sequences*

### Description

Baboon dominance sequences

### Usage

```
baboons1
```

### Format

Data sets of 5 groups of baboons, with date, winner and loser columns

### Details

The exact dates of the interactions were not given in the actual online data sets, so I set them to roughly match the data collection period presented in the actual paper (1996 - 2011)

## References

Franz M, McLean E, Tung J, Altmann J, Alberts SC (2015). "Self-organizing dominance hierarchies in a wild primate population." *Proceedings of the Royal Society B: Biological Sciences*, **282**, 20151512. doi:10.1098/rspb.2015.1512.

Franz M, McLean E, Tung J, Altmann J, Alberts SC (2015). "Data from: Self-organizing dominance hierarchies in a wild primate population." *Dryad*. doi:10.5061/dryad.d0g0d.

## Examples

```
data(baboons1)
```

---

# bonobos *Dominance matrix from de Vries et al. 2006*

---

## Description

Dominance matrix of seven bonobos

## Usage

```
data(bonobos)
```

## Format

Integer matrix, with column and row names. Winners in rows and losers in columns.

## References

de Vries H, Stevens JMG, Vervaecke H (2006). "Measuring and testing the steepness of dominance hierarchies." *Animal Behaviour*, **71**, 585-592. doi:10.1016/j.anbehav.2005.05.015.

## Examples

```
data(bonobos)
```

CBI *Clutton-Brock et al 1979 index (CBI)*

## Description

Clutton-Brock et al 1979 index (CBI)

## Usage

```
CBI(mat)
```

## Arguments

| | |
|---|---|
| mat | matrix |

## Details

The results of this function diverge from published examples in some cases. While the function produces identical scores as the results in Gammell et al. (2003) and de Vries and Appleby (2000) there are some slight deviations for the example in Whitehead (2008). The final example from Bang et al. (2010) is fairly off, but that seems to be because these authors might have applied different definitions: Bang et al. (2010) talk about 'who dominates' while (Clutton-Brock et al. 1979) consider 'who won interactions', which are two very different conceptualizations, and which might explain the discrepancies.

## Value

a named numeric vector with the indices for each individual

## Author(s)

Christof Neumann

## References

Clutton-Brock TH, Albon SD, Gibson RM, Guinness FE (1979). "The logical stag: adaptive aspects of fighting in red deer (*Cervus elaphus* L.)." *Animal Behaviour*, **27**, 211-225. doi:10.1016/0003-3472(79)901416.

Bang A, Deshpande SA, Sumana A, Gadagkar R (2010). "Choosing an appropriate index to construct dominance hierarchies in animal societies: a comparison of three indices." *Animal Behaviour*, **79**, 631-636. doi:10.1016/j.anbehav.2009.12.009.

Gammell MP, de Vries H, Jennings DJ, Carlin CM, Hayden TJ (2003). "David's score: a more appropriate dominance ranking method than Clutton-Brock et al.'s index." *Animal Behaviour*, **66**, 601-605. doi:10.1006/anbe.2003.2226.

de Vries H, Appleby MC (2000). "Finding an appropriate order for a hierarchy: a comparison of the I&SI and the BBS methods." *Animal Behaviour*, **59**, 239-245. doi:10.1006/anbe.1999.1299.

Whitehead H (2008). *Analyzing animal societies: quantitative methods for vertebrate social analysis*. University of Chicago Press, Chicago.

## Examples

```
# example from Gammell et al 2003 (table 1)
m <- matrix(0, nrow = 5, ncol = 5)
m[upper.tri(m)] <- 100
m[1, 5] <- 99
m[5, 1] <- 1
colnames(m) <- rownames(m) <- c("r", "s", "t", "u", "v")
m
CBI(m)

# example from Whitehead 2008 (table 5.8, 5.9)
m <- c(0, 2, 0, 5, 2, 2, 1, 0, 2, 0,
       0, 0, 2, 2, 1, 0, 3, 2, 1, 1,
       0, 1, 0, 1, 1, 3, 1, 1, 4, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 7, 1, 4, 2, 3,
       0, 0, 0, 0, 0, 0, 2, 3, 6, 10,
       0, 1, 1, 0, 2, 0, 0, 0, 0, 2,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
mat <- matrix(m, nrow = 10, byrow = TRUE)
colnames(mat) <- rownames(mat) <- c("x907", "x915", "x912", "x910", "x917",
                                    "x898", "x897", "x911", "x904", "x902")
round(CBI(mat), 2)
# results in book:
# 33, 2.75, 3.08, 0.91, 0.86, 0.82, 0.92, 0.53, 0.23, 0.03

simple_dom(mat2seq(mat)$winner, mat2seq(mat)$loser)

# example from Bang et al 2010 (table 1)
m <- c(0, 1, 0, 2,
       1, 0, 4, 0,
       2, 2, 0, 3,
       3, 0, 1, 0)
m <- matrix(m, ncol = 4, byrow = TRUE)
m <- t(m)
colnames(m) <- rownames(m) <- letters[1:4]
CBI(m)
# results in paper:
# 1.43, 1, 0.7, 1

# and from de Vries and Appleby (2000, table 4)
m <- c(0, 1, 1, 4, 0, 3, 6,
       0, 0, 1, 4, 0, 0, 0,
       0, 0, 0, 1, 1, 3, 14,
       0, 0, 0, 0, 2, 2, 1,
       0, 0, 0, 0, 0, 17, 2,
       0, 0, 0, 0, 0, 0, 12,
       0, 0, 0, 0, 0, 0, 0)
m <- matrix(m, ncol = 7, byrow = TRUE)
colnames(m) <- rownames(m) <- letters[1:7]
```

```
CBI(m)
simple_dom(mat2seq(m)$winner, mat2seq(m)$loser)
```

---

`coresidence` *coresidence summary*

---

## Description

coresidence summary

## Usage

```
coresidence(eloobject)
```

## Arguments

| | |
|---|---|
| `eloobject` | result from `elo.seq` |

## Details

This function provides a summary of the presence of individuals (and dyads) during the data sequence. This will be only informative if there was actually presence information supplied to `elo.seq`.

## Value

a list with three items:

`$global` **(a numeric vector)** `n_int` total number of interactions

- `n_dyads` total number of dyads
- `prop_nocores` proportion of dyads that were never co-resident
- `mean_cores_prop` mean proportion over individuals of proportions of all other IDs the focal was co-resident with at some point

`$dyads` **(a data.frame)** `id1`, `id2` the IDs

- `n_int` number of interactions for dyad
- `cores_dur` the duration of co-residence
- `none_dur` the duration for neither ID being present (both are absent)
- `one_dur` the duration of time when one ID was present but not the other

`$individuals` **(a data.frame)** `id` the ID

- `n_int` number of interactions
- `presdays` days of presence
- `cores_n_ind` co-resident with these individuals at some point
- `cores_prop` proportion of individuals with which ID was co-resident
- `stints` number of continuous bouts of presence

## See Also

presence_summary

## Examples

```
set.seed(123)
IA <- randomsequence(nID = 10, avgIA = 20, presence = c(0.7, 0.8))
SEQ <- elo.seq(winner = IA$seqdat$winner, loser = IA$seqdat$loser, Date = IA$seqdat$Date,
               presence = IA$pres, runcheck = FALSE, progressbar = FALSE)
coresidence(SEQ)
```

---

# correctly_predicted *correctly predicted outcomes*

---

## Description

correctly predicted outcomes

## Usage

```
correctly_predicted(xdata, ...)

## Default S3 method:
correctly_predicted(xdata, ...)

## S3 method for class 'elo'
correctly_predicted(xdata, exclude_draws = TRUE, daterange = NULL, ...)

## S3 method for class 'fastelo'
correctly_predicted(xdata, ...)

## S3 method for class 'list'
correctly_predicted(xdata, ...)

## S3 method for class 'matrix'
correctly_predicted(xdata, ...)
```

## Arguments

| | |
|---|---|
| xdata | result from elo.seq, fastelo, a square interaction matrix or a list with two items where the first is a character vector with ID names (which represents the rank order to be checked) and the second is an interaction matrix (which needs to be square and which has column and row names) |
| ... | additional arguments depending on the class of object you supplied |
| exclude_draws | logical, should draws be excluded from the calculation, by default TRUE. If they are included, such interactions will be scored as incorrectly predicted. |
| daterange | character or Date of length two, which allows to restrict the time range to be considered for elo objects |

## Details

If you provide results from elo.seq or fastelo, this function first extracts the number of interactions for which a winning expectation can be expressed, i.e. for all interactions for which the winning probability for either individual is different from 0.5. If the winning probability for both IDs is 0.5 then either outcome is equally likely and hence it cannot be verified whether the winning probability 'worked correctly'.

If you provide an interaction matrix, the order of columns in which it is supplied is taken as the order to be checked, i.e. this just calculates the proportion of interactions that are in upper triangle of the matrix.

If you provide a list with a rank order and an interaction matrix, the matrix will be 'reshuffled' according to the rank order and then all entries above the diagonal will be divided by the total number of interactions.

Note that there is one potential issue for the list-based method (rank order and interaction matrix supplied), which is that it can't accomodate tied ranks.

## Value

a list with two items where the first item is the proportion of correctly predicted outcomes and the second item is the total number of interactions for which the winning probability is not 0.5 (in the case of elo or fastelo) or the total number of interactions (in case of matrix or list)

## Methods (by class)

- correctly_predicted(default): default method for logical vector
- correctly_predicted(elo): for usage with results of elo.seq
- correctly_predicted(fastelo): for usage with results of fastelo
- correctly_predicted(list): for usage with a list of order and interaction matrix
- correctly_predicted(matrix): for usage with an interaction matrix

## Author(s)

Christof Neumann

## Examples

```
data(adv)
res <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date)
correctly_predicted(res)
correctly_predicted(res, daterange = c("2010-01-10", "2010-01-20"))
# only one interaction considered because for the first no expection was
# expressed (same starting values for both contestants)
correctly_predicted(res, daterange = c("2010-01-01", "2010-01-02"))

data("devries98")
correctly_predicted(list(colnames(devries98), devries98))
# is the same as
correctly_predicted(devries98)
# reversed order
```

```
correctly_predicted(list(rev(colnames(devries98)), devries98))

mat <- matrix(ncol = 10, nrow = 10, 0)
colnames(mat) <- rownames(mat) <- letters[1:10]
mat[upper.tri(mat)] <- 101
mat[lower.tri(mat)] <- 100
# correct order
order1 <- colnames(mat)
correctly_predicted(list(order1, mat))
# not very good

# the worst possible order for that matrix:
order2 <- rev(order1)
correctly_predicted(list(order2, mat))
# not much worse than order 1...

mat <- matrix(ncol = 10, nrow = 10, 0)
colnames(mat) <- rownames(mat) <- letters[1:10]
mat[upper.tri(mat)] <- 1
mat[1, 2] <- 100
# correct ranking
order1 <- letters[1:10]
correctly_predicted(xdata = list(order1, mat))
# almost correct order
order2 <- c("b", "a", letters[3:10])
correctly_predicted(xdata = list(order2, mat))
```

---

`creatematrix` *create a dominance matrix*

---

## Description

create a dominance matrix from the underlying observed sequence

## Usage

```
creatematrix(
  eloobject,
  daterange = NULL,
  drawmethod = "omit",
  onlyinteracting = FALSE,
  winners,
  losers,
  draw = NULL
)
```

## Arguments

| | |
|---|---|
| eloobject | output from elo.seq |
| daterange | character of length 2, date range to which the matrix should correspond (default from beginning to end of sequence) |
| drawmethod | character with the following options:<br>"omit" = undecided interactions (draws/ties) are ignored (default)<br>"0.5" = each undecided is counted half a win for each dyad member<br>"1" = each undecided interaction is counted twice, i.e. as win for both individuals |
| onlyinteracting | logical, indicating whether all individuals that were present (default, TRUE) are shown in the matrix, or only those that were involved in an interaction in the specified date period. If no presence data was supplied to elo.seq, it is assumed that all individuals were present at all times |
| winners | vector of winners (see details) |
| losers | vector of losers (see details) |
| draw | logical vector (currently not doing anything) |

## Details

The function works with either the output of elo.seq, or with two vectors of winners and losers. If you use winner and loser vectors, their arguments need to be named, and also the remaining arguments (daterange= and onlyinteracting=) are ignored. The function does not yet allow to include draws if you supply winner/loser vectors. If you go via the elo.seq-route, the function can handle draws (via the drawmethod= argument).

## Value

square matrix with dominance interactions (winner in rows, loser in columns)

## Author(s)

Christof Neumann

## Examples

```
data(adv)
# from winner/loser sequence directly
creatematrix(winners=adv$winner, losers=adv$loser)
# via an eloobject
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
# create dyadic matrix over the entire period of data collection
creatematrix(SEQ)
# limit to a subset of interactions
creatematrix(SEQ, daterange=c("2010-01-25", "2010-02-01"))
# limit to a subset of interactions and show only those IDs that were
# involved in at least one interaction
creatematrix(SEQ, daterange=c("2010-01-25", "2010-02-01"),
```

```
              onlyinteracting=TRUE)

# interactions restricted to single date
creatematrix(SEQ, daterange = c("2010-01-25", "2010-01-25"))

## dealing with undecided interactions
data(adv2)
SEQ <- elo.seq(winner=adv2$winner, loser=adv2$loser, Date=adv2$Date,
               draw=adv2$tie)
# omit ties/draws
creatematrix(SEQ)
# omit ties/draws
creatematrix(SEQ, drawmethod="0.5")
# omit ties/draws
creatematrix(SEQ, drawmethod="1")
```

---

`createstartvalues` *calculate start values from prior knowledge*

---

## Description

calculate start values from prior knowledge

## Usage

```
createstartvalues(
  ranks = NULL,
  rankclasses = NULL,
  shape = 0.3,
  startvalue = 1000,
  k = 100
)
```

## Arguments

| | |
|---|---|
| `ranks` | named vector, contains the ordinal ranks of all individuals for which such prior knowledge exists, names of the vector refer to the individual codes as they occur in the interaction sequence supplied to `elo.seq` |
| `rankclasses` | list with four items, each representing a rank class in descending order, if a given rank class is empty supply it as `NULL`, see details and examples |
| `shape` | numeric, between 0 and 1, by default `shape=0.3`. This value determines the 'steepness' of the initial values. Steepest is at `shape=0` and shallowest is at `shape=1`. See examples. |
| `startvalue` | numeric, the rating value with which an individual starts into the rating process. By default `startvalue=1000` |
| `k` | numeric, the *k* factor that determines the maximum change in ratings. By default `k=100` |

## Details

only one of `ranks` or `rankclasses` can be supplied.

if you wish to supply rank classes you need to supply four categories and it is assumed that the first list item is the highest class. If you have less than four rank classes, you still need to supply a list with four items and set those that you wish to ignore to `NULL`, see examples.

## Value

list with three items:

| | |
|---|---|
| `res` | a named numeric vector with the startvalues to be supplied to `elo.seq` |
| `k` | *k* factor used |
| `startvalue` | start value used |

## Author(s)

Christof Neumann

## References

Newton-Fisher NE (2017). "Modeling social dominance: Elo-ratings, prior history, and the intensity of aggression." *International Journal of Primatology*, **38**, 427-447. doi:10.1007/s10764017-99522.

## Examples

```
# assuming a group with 7 individuals
# with four rank classes
myrankclasses <- list(alpha = "a", high=c("b", "c"), mid=c("d", "e"), low=c("f", "g"))
createstartvalues(rankclasses = myrankclasses)
# with two rank classes
myrankclasses2 <- list(class1 = NULL, high=c("a", "b", "c"), class3=NULL, low=c("d", "e", "f", "g"))
createstartvalues(rankclasses = myrankclasses2)

# with ordinal ranks
myranks <- 1:7; names(myranks) <- letters[1:7]
createstartvalues(ranks = myranks)
```

---

# `DCindex` *Directional Consistency Index*

---

## Description

calculate Directional Consistency Index

## Usage

```
DCindex(interactionmatrix)
```

## Arguments

| | |
|---|---|
| interactionmatrix | square interaction matrix with winner in rows and losers in columns, for example the output from creatematrix |

## Value

numeric value, the DCI

## Author(s)

Christof Neumann

## References

van Hooff JARAM, Wensing JAB (1987). "Dominance and its behavioural measures in a captive wolf pack." In Frank H (ed.), *Man and Wolf*, 219-252. Junk, Dordrecht.

## Examples

```
data(adv)
SEQ <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date)
mat <- creatematrix(SEQ)
DCindex(mat)

# or directly from a matrix
data(bonobos)
DCindex(bonobos)
```

---

# devries98 *Dominance matrix from de Vries (1998)*

---

## Description

Fictional dominance matrix from de Vries (1998) from 10 individuals.

## Usage

```
data(devries98)
```

## Format

Named integer matrix.

## References

de Vries H (1998). "Finding a dominance order most consistent with a linear hierarchy: a new procedure and review." *Animal Behaviour*, **55**, 827-843. doi:10.1006/anbe.1997.0708.

## Examples

```
data(devries98)
```

---

`dommats` *Example dominance matrices*

---

## Description

Example dominance matrices

## Usage

```
dommats
```

## Format

A named list with dominance matrices:

- `badgers`: 7 badgers (Hewitt et al 2009, Fig. A1 PO2004)
- `squirrels`: 8 squirrels (Farentinos 1972, Table 1C)
- `elephants`: 7 elephants (Archie et al 2006, Fig. 2, JA)

## References

Farentinos RC (1972). "Social dominance and mating activity in the tassel-eared squirrel (*Sciurus aberti ferreus*)." *Animal Behaviour*, **20**, 316-326. doi:10.1016/S00033472(72)800538.

Archie EA, Morrison TA, Foley CAH, Moss CJ, Alberts SC (2006). "Dominance rank relationships among wild female African elephants, *Loxodonta africana*." *Animal Behaviour*, **71**, 117-127. doi:10.1016/j.anbehav.2005.03.023.

Hewitt SE, Macdonald DW, Dugdale HL (2009). "Context-dependent linear dominance hierarchies in social groups of European badgers, *Meles meles*." *Animal Behaviour*, **77**, 161-169. doi:10.1016/j.anbehav.2008.09.022.

## Examples

```
data(dommats)
```

DS *David's score*

## Description

calculate David's scores from an interaction matrix

## Usage

```
DS(interactionmatrix, prop = c("Dij", "Pij"))
```

## Arguments

| | |
|---|---|
| `interactionmatrix` | square interaction matrix with winner in rows and losers in columns, for example the output from `creatematrix` |
| `prop` | the type of dyadic win proportion to be use. By default corrected for number of interactions in a dyad (`prop="Dij"`), otherwise the raw proportion (`prop="Pij"`) |

## Value

a data.frame with columns ID, DS (David's scores) and normDS (normalized David's scores)

## Author(s)

Christof Neumann

## References

David HA (1987). "Ranking from unbalanced paired-comparison data." *Biometrika*, **74**, 432-436. doi:10.1093/biomet/74.2.432.

Gammell MP, de Vries H, Jennings DJ, Carlin CM, Hayden TJ (2003). "David's score: a more appropriate dominance ranking method than Clutton-Brock et al.'s index." *Animal Behaviour*, **66**, 601-605. doi:10.1006/anbe.2003.2226.

de Vries H, Stevens JMG, Vervaecke H (2006). "Measuring and testing the steepness of dominance hierarchies." *Animal Behaviour*, **71**, 585-592. doi:10.1016/j.anbehav.2005.05.015.

## Examples

```
data(bonobos)
DS(bonobos)
DS(bonobos, prop = "Pij")
```

## dyadic_dom *dyadic dominance relations*

### Description

dyadic dominance relations

### Usage

```
dyadic_dom(winner, loser, Date = NULL, daterange = NULL)
```

### Arguments

| | |
|---|---|
| winner | character or factor with winner |
| loser | character or factor with winner |
| Date | not yet implemented |
| daterange | not yet implemented |

### Value

a data.frame with one row per dyad

### Examples

```
xdata <- randomsequence(nID = 5, avgIA = 10, reversals = 0.1)$seqdat
dyadic_dom(xdata$winner, xdata$loser)
```

## dyadic_reversals *changes in dyadic relationships*

### Description

compare dyadic relationships before and after a certain date

### Usage

```
dyadic_reversals(eloobject, cutpoint = NULL, daterange = NULL)
```

### Arguments

| | |
|---|---|
| eloobject | result from elo.seq |
| cutpoint | character or Date, the date at which to split into pre and post (default is NULL, where the data is split in halves). The actual date here will be included in the 'pre' period. |
| daterange | character or Date of length 2, the date range to be considered (default is NULL where the entire date range in the data is used) |

## Value

a data.frame with one line per dyad:

**id1,id2** the dyad

**pre_n,post_n** the number of interactions for that dyad pre and post cutpoint date

**pre,post** which of the two was dominant (`1` = id1, `2` = id2, `0` = tied relationship, `NA` = unknown relationship, i.e. 0 interactions)

## Examples

```
data(adv)
eloobject <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date)
# split at halfway point ("2010-01-17")
# one reversal: a-f
dyadic_reversals(eloobject)
# shift split date so that both interactions for a/f occur in the post period,
# which makes it a tie in post and unknown in pre
dyadic_reversals(eloobject, cutpoint = "2010-01-10")
```

---

# e.single *Elo ratings for a single interaction*

---

## Description

calculate/update Elo ratings for a single dyadic interaction

## Usage

```
e.single(ELO1old, ELO2old, outcome, k = 100, normprob = TRUE)
```

## Arguments

| | |
|---|---|
| `ELO1old`, `ELO2old` | numeric, Elo rating of the first and second individual |
| `outcome` | `1` = first individual wins and second looses<br>`2` = second individual wins and first looses<br>`0` = interaction ends in a draw/tie (no winner and no looser) |
| `k` | numeric, *k* factor, by default `k = 100` |
| `normprob` | logical (by default `TRUE`). Should a normal curve be assumed for calculating the winning/losing probablities, or a logistic curve. See `winprob` for details |

## Value

integer vector of length 2 with updated ratings of first and second individual after the interaction

## Author(s)

Christof Neumann

## References

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

Albers PCH, de Vries H (2001). "Elo-rating as a tool in the sequential estimation of dominance strengths." *Animal Behaviour*, **61**, 489-495. doi:10.1006/anbe.2000.1571.

## Examples

```
e.single(ELO1old = 1200, ELO2old = 1000, outcome = 1, k = 100)
# same as before
e.single(ELO1old = 1000, ELO2old = 1200, outcome = 2, k = 100)
# an undecided interaction
e.single(ELO1old = 1200, ELO2old = 1000, outcome = 0, k = 100)
# if rating differences are too big, no change occurs
# if higher-rated individual wins
e.single(ELO1old = 2000, ELO2old = 1000, outcome = 1, k = 100)
# same as before but lower-rated individual wins and
# therefore wins maximum number of points possible (i.e. k)
e.single(ELO1old = 2000, ELO2old = 1000, outcome = 2, k = 100)
```

---

# elo.seq *calculate Elo ratings*

---

## Description

calculate Elo ratings from a sequence of dominance interactions

## Usage

```
elo.seq(winner, loser, Date, draw = NULL, presence = NULL, startvalue = 1000,
               k = 100, normprob = TRUE, init = "average", intensity = NULL,
               iterate = 0, runcheck = TRUE, progressbar = FALSE)
fastelo(WINNER, LOSER, ALLIDS, KVALS, STARTVALUES, NORMPROB = TRUE, ROUND = TRUE)
```

## Arguments

| | |
|---|---|
| winner | either a factor or character vector with winner IDs of dyadic dominance interactions |
| loser | either a factor or character vector with loser IDs of dyadic dominance interactions |
| Date | character vector of form "YYYY-MM-DD" with the date of the respective interaction |
| draw | logical, which interactions ended undecided (i.e. drawn or tied)? By default all FALSE, i.e. no undecided interactions occurred. Note that in this case, winner/loser values can be interchanged |
| presence | optional data.frame, to supply data about presence and absence of individuals for part of the time the data collection covered. see details |

| | |
|---|---|
| startvalue | the value of Elo ratings of the two individuals that are involved in the first interaction of the overall sequence prior to this interaction. By default set to 1000. See also init |
| k | factor *k* that determines the maximum change in ratings. By default k=100 |
| normprob | logical (by default TRUE). Should a normal curve be assumed for calculating the winning/losing probabilities, or a logistic curve. See winprob for details |
| init | character, what Elo rating does an individual have prior to its first interaction. Three options are available: average: individuals always start with the value specified in startvalue. Given stable composition of the group, this also reflects the average Elo rating on each day in that group,<br>bottom: subjects entering at the current lowest Elo value if the lowest value getting lower its getting lower for all subjects which had this lowest values before, it is reflecting that in some species new subjects entering a group at the bottom level "bottom entry"<br>bottom_low: same as bottom but additionally the start values getting after the first interaction lower for all non-interacting subjects and, reflecting that we have at start no knowledge about the subjects this option offers for "bottom entry" species the possibility to consider that in a way that those subjects which are not interacting getting lower from start on |
| intensity | a character vector or factor describing intensity of interaction, to be matched with custom k values if specified |
| iterate | not yet implemented |
| runcheck | logical, should several checks regarding data integrity be performed, by default TRUE. See seqcheck |
| progressbar | logical, should progress bars be displayed, by default progressbar=TRUE |
| WINNER | same as winner for use in fastelo() |
| LOSER | same as loser for use in fastelo() |
| ALLIDS | character vector, contains all the indivuals IDS |
| KVALS | numeric vector of the same length WINNER, i.e. one k value for each interaction |
| STARTVALUES | numeric vector of the same length as ALLIDS, i.e. one start value for each individual |
| NORMPROB | logical, by default TRUE: same as normprob for use in fastelo() |
| ROUND | logical, by default TRUE: should ratings be rounded to integers. For use in fastelo() |

## Details

The presence 'matrix' is actually an object of class data.frame containing information about wether an individual was present on a given day or not. The first column represents the dates, running at least from the date of the earliest interaction until at least the date of the last interaction with one line per day (regardless of whether there were actually interactions observed on each day). Further, each individual is represented as a column in which "1" indicates an individual was present on the row-date and a "0" indicates the individuals absence on this date. NAs are not allowed. See advpres for an example.

The function `fastelo()` is a stripped-down version of `elo.seq()`, which performs only the most basic calculations while ignoring anything that is date and presence related. Neither does it perform data checks. In other words, it just calculates ratings based on the sequence. It's most useful in simulations, for example when estimating optimal k parameters. Its main advantage is its speed, which is substantially faster than `elo.seq()`. Note that currently there is no support for tied interactions. The main difference to note is that both, start values and k values have to be supplied as vectors with one value for each individual and interaction respectively.

## Value

An object of class `elo`, which is list with 10 items that serves as basis to extract relevant information:

| | |
|---|---|
| `mat` | a date by ID-`matrix` with raw Elo ratings |
| `lmat` | a date by ID-`matrix` with raw Elo ratings |
| `cmat` | a date by ID-`matrix` with raw Elo ratings |
| `pmat` | a date by ID-`matrix` with with presence data |
| `nmat` | a date by ID-`matrix` containing the number of interactions a given ID was involved in on a given day |
| `logtable` | details on each single interaction |
| `stability` | a `data.frame` containing information about stability (see `stab_elo`) |
| `truedates` | vector of class `Date` covering the ranges of dates in the dataset |
| `misc` | various |
| `allids` | a (sorted) character vector with all IDs that occur in the dataset |

`fastelo()` returns a list with ten items:

| | |
|---|---|
| `$ratings` | numeric vector of the final ratings in the same order as `ALLIDS` |
| `$winprobs` | numeric vector with winning probabilities in the same order as the interactions were supplied |
| `$rtype` | character of length 1, as a marker that the result comes from `fastelo()` |
| `$startvalues` | numeric vector with start values |
| `$kvalues` | numeric vector with k values |
| `$winner` | character vector with winners |
| `$loser` | character vector with losers |
| `$allids` | character vector with all IDs that occur in the sequence |
| `$normprob` | logical, was normal probability used for winning expectations |
| `$round` | logical, was rounding to integers used during the calculation of ratings |

## Author(s)

Christof Neumann and Lars Kulik

## References

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

Albers PCH, de Vries H (2001). "Elo-rating as a tool in the sequential estimation of dominance strengths." *Animal Behaviour*, **61**, 489-495. doi:10.1006/anbe.2000.1571.

Neumann C, Duboscq J, Dubuc C, Ginting A, Irwan AM, Agil M, Widdig A, Engelhardt A (2011). "Assessing dominance hierarchies: validation and advantages of progressive evaluation with elo-rating." *Animal Behaviour*, **82**, 911-921. doi:10.1016/j.anbehav.2011.07.016.

Newton-Fisher NE (2017). "Modeling social dominance: Elo-ratings, prior history, and the intensity of aggression." *International Journal of Primatology*, **38**, 427-447. doi:10.1007/s10764017-99522.

## Examples

```
data(adv)
res <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date)
summary(res)

# with custom k
data(adv2)
table(adv2$intensity)

myks <- list(displace = 20, fight = 200)
res <- elo.seq(winner = adv2$winner, loser = adv2$loser, Date = adv2$Date,
               k = myks, intensity = adv2$intensity)
extract_elo(res)
summary(res)

# with custom start values
# if we know prior ranks:
myranks <- 1:7
names(myranks) <- letters[1:7]
mypriors <- createstartvalues(myranks, shape = 0.3)
res <- elo.seq(winner = adv2$winner, loser = adv2$loser, Date = adv2$Date,
               k = myks, intensity = adv2$intensity, startvalue = mypriors$res)
extract_elo(res)

# compare elo.seq and fastelo
xdata <- randomsequence(10, 500)
allids <- colnames(xdata$pres)[2:ncol(xdata$pres)]
winner <- xdata$seqdat$winner
loser <- xdata$seqdat$loser
Date <- xdata$seqdat$Date
k <- rep(100, length(winner))
svals <- rep(1000, length(allids))

res1 <- fastelo(WINNER = winner, LOSER = loser, ALLIDS = allids, KVALS = k,
                STARTVALUES = svals, NORMPROB = TRUE)$ratings
names(res1) <- allids
res1 <- sort(res1, decreasing = TRUE)
res2 <- extract_elo(elo.seq(winner = winner, loser = loser, Date = Date,
```

```
                          startvalue = 1000, k = 100, normprob = TRUE,
                          runcheck = FALSE))
res1
res2
```

---

`eloplot` *Elo rating plots*

---

## Description

plot Elo ratings for all or selected individuals over a specified time period

## Usage

```
eloplot(
  eloobject,
  ids = "all",
  interpolate = "yes",
  from = "start",
  to = "end",
  color = TRUE
)
```

## Arguments

| | |
|---|---|
| `eloobject` | elo object, output of `elo.seq` function |
| `ids` | character, `"all"` will plot trajectories for all individuals within the dataset. `"first.20"` will plot the 20 first individuals. `"random.20"` will plot 20 randomly chosen individuals from the dataset. Alternatively, provide a list of individual IDs. |
| `interpolate` | character, by default (`"yes"`) plot interpolated Elo values or plot Elo values without interpolation (`"no"`) |
| `from` | character, either `"start"`, i.e. the plotted date range will start at the first date of the dataset, or provide a custom date ("YYYY-MM-DD") |
| `to` | character, either `"end"`, i.e. the plotted date range will end at the last date of the dataset, or provide a custom date ("YYYY-MM-DD") |
| `color` | logical, the plot is either colored (`TRUE`) or in black and white with symbols |

## Details

For a visual inspection of an Elo object it is useful to plot the calculated trajectories. We recommend not to plot trajectories for more than 20 individuals at once.

Note also, if plots for IDs are requested that had observations on only one day, these IDs are excluded from plotting and a corresponding warning message is produced.

## Value

a plot

### Author(s)

Lars Kulik and Christof Neumann

### Examples

```
data(adv)
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
eloplot(SEQ, ids="all", interpolate="yes", from="start", to="end",
        color=TRUE)
```

---

## extract_elo *extract Elo ratings from elo object*

---

### Description

extract Elo ratings from elo object

### Usage

```
extract_elo(
  eloobject,
  extractdate = eloobject$misc["maxDate"],
  standardize = FALSE,
  IDs = NULL,
  NA.interpolate = FALSE,
  daterange = 1
)
```

### Arguments

| | |
|---|---|
| eloobject | result from elo.seq |
| extractdate | character, date on which Elo ratings should be obtained, defaults to the last day in the data set |
| standardize | logical, should the returned ratings be scaled between 0 and 1. Default is FALSE. See scale_elo |
| IDs | character, specify IDs for which ratings are returned. By default, returns all that were present on the date or at least on one day of the date range |
| NA.interpolate | if FALSE (default), the last known rating is returned, which might not be from the specified date itself (but older). If TRUE, ratings on days without observations are linearly interpolated between days with known ratings (i.e. dates with observed interactions) |
| daterange | if averaged ratings are desired, supply here the number of days from extractdate - 1. By default (daterange = 1), the ratings of the single extractdate are returned. daterange = 2 produces average ratings from extractdate and the day after, and so on... |

## Details

`extractdate` can be also a vector of dates. In this case, the `IDs` argument has to be either a vector of length 1 (i.e. a single individual) or a vector of the same length as `extractdate`. In the first case, the ratings for the same individual are returned on the dates specified in `extractdate`. In the second case, dates and IDs are matched, i.e. the rating of the individual on that date is returned in the same order as the dates/IDs vectors.

## Value

named (IDs) vector of (average) Elo ratings, or an unnamed vector of ratings (if length of `extracte` is larger than 1)

## Author(s)

Christof Neumann

## Examples

```
data(adv)
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
extract_elo(SEQ, "2010-01-30")
extract_elo(SEQ, "2010-01-30", standardize=TRUE)

# same ratings (regardless of NA.interpolate),
# since "g" was observed on both days
extract_elo(SEQ, "2010-01-29", IDs="g")
extract_elo(SEQ, "2010-01-29", IDs="g", NA.interpolate=TRUE)

extract_elo(SEQ, "2010-01-31", IDs="g")
extract_elo(SEQ, "2010-01-31", IDs="g", NA.interpolate=TRUE)

# different ratings (depending on NA.interpolate),
# since "g" was not observed that day
extract_elo(SEQ, "2010-01-30", IDs="g")
extract_elo(SEQ, "2010-01-30", IDs="g", NA.interpolate=TRUE)

extract_elo(SEQ, "2010-01-10", daterange=5)
extract_elo(SEQ, "2010-01-10", daterange=5, NA.interpolate=TRUE)

# and for multiple dates and a single IDs
dates <- sample(adv$Date, size = 10, replace = TRUE)
ids <- "b"
extract_elo(eloobject = SEQ, extractdate = dates, standardize = FALSE, IDs = ids)

# and for multiple dates and IDs
dates <- sample(adv$Date, size = 10, replace = TRUE)
ids <- sample(colnames(advpres)[2:8], size = 10, replace = TRUE)
extract_elo(eloobject = SEQ, extractdate = dates, standardize = FALSE, IDs = ids)
```

`h.index` *linearity indices*

## Description

linearity indices

## Usage

```
h.index(interactionmatrix, loops = 1000)
```

## Arguments

| | |
|---|---|
| `interactionmatrix` | square interaction matrix with winner in rows and losers in columns, for example the output from `creatematrix` |
| `loops` | numeric, the number of randomizations to perform (by default: 1000) |

## Details

Note that the expected value of *h* can also be calculated as 3/(N+1).

## Value

a data.frame with with values for the number of individuals in the matrix (N), linearity indices (h, h' and expected h), p-value, number of randomizations, and number of unknown and tied relationships.

## Author(s)

Christof Neumann

## References

Appleby MC (1983). "The probability of linearity in hierarchies." *Animal Behaviour*, **31**, 600-608. doi:10.1016/S00033472(83)800840.

de Vries H (1995). "An improved test of linearity in dominance hierarchies containing unknown or tied relationships." *Animal Behaviour*, **50**, 1375-1389. doi:10.1016/00033472(95)800530.

## Examples

```
data(bonobos)
h.index(bonobos)
```

heatmapplot *heatmap*

## Description

heatmap

## Usage

```
heatmapplot(
  formula,
  data,
  xbreaks = NULL,
  ybreaks = NULL,
  addvals = FALSE,
  addN = FALSE,
  digits = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| formula | formula for plot |
| data | data set for plot (typically a data frame) |
| xbreaks | numeric, the breakpoints for the horizontal axis |
| ybreaks | numeric, the breakpoints for the vertical axis |
| addvals | add the response values to the plot |
| addN | add the sample size to the plot |
| digits | numeric: if response variable is plotted, round to this many digits (default is 1) |
| ... | other parameters passed on to plot() or text() |

## Value

a plot

## Examples

```
xdata <- expand.grid(a = seq(0, 1, 0.1), b = seq(10, 20, 1))
xdata$resp <- rnorm(nrow(xdata))
heatmapplot(resp ~ a + b, data = xdata)

set.seed(123)
xdata <- expand.grid(k = seq(8, 200, length.out = 31), shape = seq(0, 1, length.out = 31))
idata <- randomsequence(10, 50, reversals = 0.3)
allids <- colnames(idata$pres)[2:ncol(idata$pres)]
winner <- as.character(idata$seqdat$winner)
```

```
loser <- as.character(idata$seqdat$loser)

myranks <- 1:length(allids)
names(myranks) <- allids

for(i in 1:nrow(xdata)) {
  kv <- rep(xdata$k[i], length(winner))
  sv <- createstartvalues(ranks = myranks, shape = xdata$shape[i])$res
 res <- fastelo(WINNER = winner, LOSER = loser, ALLIDS = allids, KVALS = kv, STARTVALUES = sv,
                ROUND = FALSE)
  xdata$ll[i] <- likelo(res)
}

heatmapplot(ll ~ k + shape, data = xdata)
```

---

`incontable` *number and strength of inconsistencies*

---

## Description

calculate number and strength of inconsistencies

## Usage

```
incontable(mat)
```

## Arguments

| | |
|---|---|
| `mat` | square interaction matrix with winner in rows and losers in columns, for example the output from `creatematrix` |

## Value

data frame with inconsistencies and their strength

## Author(s)

Christof Neumann

## References

de Vries H (1998). "Finding a dominance order most consistent with a linear hierarchy: a new procedure and review." *Animal Behaviour*, **55**, 827-843. doi:10.1006/anbe.1997.0708.

## Examples

```
data(bonobos)
incontable(bonobos)
```

`individuals` *individuals present in the group*

## Description

returns IDs, number or IDs, or CV of number of present individuals

## Usage

```
individuals(
  eloobject,
  from = eloobject$misc["maxDate"],
  to = NULL,
  outp = c("N", "IDs", "CV")
)
```

## Arguments

| | |
|---|---|
| `eloobject` | result from `elo.seq` |
| `from` | character, from which date onwards should the ID statistics be calculated. By default the first date in the sequence is used |
| `to` | character, until which date should the ID statistics be calculated. By default `NULL`, i.e. the returned information refers to only the date specified by `from` |
| `outp` | character, one of three options to determine which kind of information is returned: (1) `"N"`: the (average) number of individuals present, (2) `"IDs"`: the actual IDs, and (3): `"CV"`: coefficient of number of individuals present |

## Details

if `to=NULL`, either the IDs (`outp="IDs"`) or the number of individuals (`outp="N"`) present on this day is returned. `outp="CV"` is not defined in such a case (returns `NA`).

if a `to` date is set (i.e. different from `NULL`), either the IDs of all individuals that were present on at least one day of the date range (`outp="IDs"`) is returned or the average number of individuals present during this time (`outp="N"`). If `outp="CV"`, the coefficient of variation of the number of individuals present is returned, which might be considererd another measure of stability on the group level.

## Value

numeric or character

## Author(s)

Christof Neumann

## Examples

```
data(adv)
SEQ <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date)
individuals(SEQ, outp = "N")
individuals(SEQ, outp = "IDs")
individuals(SEQ, outp = "CV") # not defined

# consider additional presence information
data(advpres)
SEQ <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date,
               presence = advpres)
individuals(SEQ, outp = "N")
individuals(SEQ, outp = "IDs")
individuals(SEQ, outp = "CV") # not defined

# across a date range
individuals(SEQ, from = "2010-01-01", to = "2010-01-31", outp = "N")
individuals(SEQ, from = "2010-01-01", to = "2010-01-31", outp = "IDs")
individuals(SEQ, from = "2010-01-01", to = "2010-01-31", outp = "CV")
```

---

# ISI *de Vries' I&SI ranking*

---

## Description

de Vries' I&SI ranking

## Usage

```
ISI(mat, runs = 5000, printmessages = TRUE)
```

## Arguments

| | |
|---|---|
| `mat` | square interaction matrix with winner in rows and losers in columns, for example the output from `creatematrix` |
| `runs` | numeric, number of iterations, by default `5000` |
| `printmessages` | logical, should the number of I and SI be printed (as well as a message if there is more than one solution). By default `TRUE`. |

## Details

The number of interations is set substantially higher than what was suggested in the de Vries' 1998 paper, because my algorithm here is less efficient.

The I&SI algorithm (c.f. de Vries 1998) does not necessarily result in a unique order (see example below). If such a case occurs, all (equally good) solutions are returned as a list.

The function checks whether a `table` is supplied instead of a `matrix` and converts from table to matrix if possible (trying to keep the column and row names if supplied in the table).

If the matrix does not have column-names, unique column- and row-names are assigned.

## Value

a list with the best possible matrix (or matrices if there is more than one best solution)

## Author(s)

Christof Neumann

## References

de Vries H (1998). "Finding a dominance order most consistent with a linear hierarchy: a new procedure and review." *Animal Behaviour*, **55**, 827-843. doi:10.1006/anbe.1997.0708.

## See Also

ISIranks

## Examples

```
data(devries98)
h.index(devries98)
ISI(devries98)

##
data(adv)
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
mat <- creatematrix(SEQ)
res <- ISI(mat)
# note that this matrix is not sufficiently linear to justify such ordering
h.index(mat)
```

---

# ISIranks *ISI ranks*

---

## Description

ISI ranks

## Usage

```
ISIranks(x, sortbyID = TRUE)
```

## Arguments

| | |
|---|---|
| x | a list of matrices, with the same column names, typically the output of ISI |
| sortbyID | logical, should the output be sorted by ID (default is TRUE). If FALSE, output is sorted by (average rank) |

### Details

if there is more than one solution resulting from ISI, average (mean) ranks will be calculated. If there is only one solution, the average rank will be the same as the rank from the (one) ISI ranking

### Value

a data.frame with at least three columns: IDs, their average rank and the rankings of all rankings that satisfy ISI's minimum criteria

### Examples

```
# no unique solution
data(adv)
mat <- creatematrix(winners = adv$winner, losers = adv$loser)
set.seed(123)
res <- ISI(mat)
ISIranks(res)
ISIranks(res, sortbyID = FALSE)

# only one (and unique) solution
data(bonobos)
set.seed(123)
res <- ISI(bonobos)
ISIranks(res)
ISIranks(res, sortbyID = FALSE)
```

---

## lastdaypresent *last day an individual was present*

---

### Description

last day an individual was present with respect to a reference date

### Usage

```
lastdaypresent(x, ID = "all", refdate = NULL)
```

### Arguments

| | |
|---|---|
| x | output from elo.seq |
| ID | character, if "all", all individuals are returned, otherwise only for the desired ID |
| refdate | character or Date (YYYY-MM-DD), up to which date the presence data should be considered, by default the last date of the sequene |

### Details

the function can result in NA for two reasons. 1) the ID is not found in the presence data, which is accompanied by a warning and 2) the ID was not yet present if a referene date is specified

## Value

Date or NA

## Author(s)

Christof Neumann

## Examples

```
data(adv)
data(advpres)
SEQ <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date,
               presence = advpres)
lastdaypresent(SEQ, ID = "all", refdate = "2010-01-02")
lastdaypresent(SEQ, ID = "f", refdate = "2010-02-02")
```

---

# likelo *(log) likelihood of Elo-rating model*

---

## Description

(log) likelihood of Elo-rating model

## Usage

```
likelo(eloobject, burnin = 0, ll = TRUE, daterange = NULL)
```

## Arguments

| | |
|---|---|
| eloobject | output from elo.seq (or from fastelo) |
| burnin | numeric, the number of interactions to be excluded from the calculation of the (log) likelihood. This parameter is ignored if a date range is supplied. By default burnin = 0, i.e. all interactions are considered. |
| ll | logical, should the log likelihood be returned rather than the likelihood, by default TRUE |
| daterange | character or Date of length 2, gives the date range for which likelihood should be calculated. By default, the entire date range of all interactions is considered. |

## Details

This function returns the (log) likelihood of a dominance interaction sequence. The likelihood is the product of all winning probabilities (for each interaction).

## Value

numeric of length 1, the (log) likelihood

## References

Franz M, McLean E, Tung J, Altmann J, Alberts SC (2015). "Self-organizing dominance hierarchies in a wild primate population." *Proceedings of the Royal Society B: Biological Sciences*, **282**, 20151512. doi:10.1098/rspb.2015.1512.

McMahan CA, Morris MD (1984). "Application of maximum likelihood paired comparison ranking to estimation of a linear dominance hierarchy in animal societies." *Animal Behaviour*, **32**, 374-378. doi:10.1016/S00033472(84)802717.

## Examples

```
data(adv)
res <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date, k = 200)
likelo(res)
res <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date, k = 100)
likelo(res)
ks <- seq(100, 400, by = 20)
liks <- numeric(length(ks))
for(i in 1:length(liks)) {
  liks[i] <- likelo(elo.seq(winner = adv$winner, loser = adv$loser,
                    Date = adv$Date, k = ks[i]))
}
plot(ks, liks, type = "l")

# discard early interactions via 'burnin'
likelo(res)
# the same as above:
likelo(res, burnin = 0)
# discard the first 10 interactions:
likelo(res, burnin = 10)
# discard all but the last interaction:
likelo(res, burnin = 32)
# which is the same as the log of the last winning probability:
log(winprob(res$logtable$Apre[33], res$logtable$Bpre[33]))
```

---

# mat2seq *matrix to sequence conversion*

---

## Description

matrix to sequence conversion

## Usage

```
mat2seq(mat)
```

## Arguments

| | |
|---|---|
| mat | square interaction matrix with winner in rows and losers in columns (can have column/row names or not) |

## Value

a data.frame with a winner and a loser column

## Examples

```
mat <- matrix(c(0,1,1,0,0,1,0,0,0), ncol=3, byrow = TRUE)
rownames(mat) <- colnames(mat) <- LETTERS[1:3]
mat2seq(mat)

mat <- matrix(c(0,1,1,0,0,1,3,0,0), ncol=3, byrow = TRUE)
rownames(mat) <- colnames(mat) <- LETTERS[1:3]
mat2seq(mat)

# without column names
mat <- matrix(c(0,1,1,0,0,1,0,0,0), ncol=3, byrow = TRUE)
mat2seq(mat)
```

---

# optimizek *optimize the k parameter*

---

## Description

optimize the k parameter

## Usage

```
optimizek(
  eloobject,
  krange = c(2, 400),
  optimode = "loop",
  resolution = 100,
  itype = NULL,
  daterange = NULL,
  burnin = 0,
  doplot = FALSE,
  progbar = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| eloobject | output from elo.seq or from fastelo |
| krange | either a vector of length 2, giving the range of k values to be tested, or a named list with vectors of length 2, in which each list item is named according to different interaction types (see the intensity= argument in elo.seq) |
| optimode | character, either "loop" or "optimize". See details. |

| | |
|---|---|
| resolution | numeric, the number of steps between the range of k values to be tested. Currently only a single value can be supplied here and in case krange is a list this value will be applied to all items in this list |
| itype | character or factor containing the different interaction types, which is only relevant if krange is a list. The content of itype and the names of krange have to match! |
| daterange | character or Date of length 2, provides a date range for optimization. Only relevant in case eloobject is the result of elo.seq() |
| burnin | numeric, the number of interactions to be excluded from the calculation of the (log) likelihood. This parameter is ignored if a date range is supplied. By default burnin = 0, i.e. all interactions are considered. |
| doplot | logical, should a plot be returned. Works only if optimode = "loop", and only if there are maximally two different interaction types |
| progbar | logical, should a progress bar be displayed, not yet implemented |
| ... | additional arguments for the plot and text functions, e.g. for setting cex or lwd |

## Details

this function attempts to find the objectively best k parameter. This is done by a maximum likelihood approach in which the likelihood is represented by the individual winning probabilities. In a perfect situation, in each interaction the winner would have a winning probability of 1, whereas in the worst case, in each interaction the winner would have a winning probability of 0.

There are two major approaches to find the best k. One does it 'by hand', i.e. by means of a loop trying many different k values (specified by resolution), recalculating the ratings (and associated winning probabilities) and return the likelihood for each k value. The second approach uses the optimize function, but this is not yet implemented.

One thing to note is that you can use interaction-level k values, i.e. if you have interactions of different types (e.g. fights vs. displacements) you can try to find the optimal k for each interaction type. This is achieved in the ("loop" approach by trying different *combinations* of k values. Because of the combinatorial nature of this approach, the number of individual sequences to be fitted increases sharply with higher resolutions: if you have two different interaction types and use a resolution of 5, the function will need to run 25 (= 5 * 5) iterations. If you use a more reasonable resolution of 100 the number of iterations will be already 10000. Also note that in that case the actual plotting of the results might take a lot of time in such cases. Just try with low values first to see whether it works as expected and the potentially increase the resolution.

## Value

a list with two items: (1) $best, a data frame with one line, in which the maximal log likelihood is returned alongside the one or several corresponding k values, and (2) $complete, a data frame with all the values tested and their log likelihoods

## References

Franz M, McLean E, Tung J, Altmann J, Alberts SC (2015). "Self-organizing dominance hierarchies in a wild primate population." *Proceedings of the Royal Society B: Biological Sciences*, **282**, 20151512. doi:10.1098/rspb.2015.1512.

McMahan CA, Morris MD (1984). "Application of maximum likelihood paired comparison ranking to estimation of a linear dominance hierarchy in animal societies." *Animal Behaviour*, **32**, 374-378. doi:10.1016/S00033472(84)802717.

## Examples

```
data(adv2)
res <- elo.seq(winner = adv2$winner, loser = adv2$loser, Date = adv2$Date)
optimizek(eloobject = res, krange = c(50, 400), resolution = 200, doplot = TRUE)$best

# with a burnin value set:
optimizek(eloobject = res, krange = c(50, 400), resolution = 200, burnin = 15, doplot = TRUE)$best

# using different interaction intensities
myks <- list(displace = 20, fight = 200)
res <- elo.seq(winner = adv2$winner, loser = adv2$loser, Date = adv2$Date,
               k = myks, intensity = adv2$intensity)
optimizek(eloobject = res, optimode = "loop",
          krange = list(fight = c(50, 600), displace = c(20, 200)),
          resolution = 100, itype = adv2$intensity, main = 'bla')$best
```

---

# optistart *optimize start values*

---

## Description

experimental function to test different sets of randomly selected start values

## Usage

```
optistart(
  eloobject,
  burnin = 0,
  spread = 200,
  runs = 2000,
  doplot = FALSE,
  initialcohort = TRUE
)
```

## Arguments

| | |
|---|---|
| `eloobject` | output from `elo.seq` |
| `burnin` | numeric, the number of interactions to be excluded from the calculation of the (log) likelihood. This parameter is ignored if a date range is supplied. By default `burnin = 0`, i.e. all interactions are considered. |
| `spread` | numeric, the standard deviation of the ratings to be tested (by default 200) |
| `runs` | numeric, number of inital ratings to be tested (by default 2000) |
| `doplot` | logical, should the distribution of log likelihoods be plotted |
| `initialcohort` | logical, not yet implemented |

### Details

if the plot is produced, the red line indicates the log-likelihood when all individuals are assigned the same starting value

the item `$best` reflects the optimal start values found

### Value

a list with multiple items:

### Author(s)

Christof Neumann

### Examples

```
set.seed(123)
xdata <- randomsequence(8, 100)$seqdat
res1 <- elo.seq(xdata$winner, xdata$loser, xdata$Date)
ores <- optistart(res1)
res2 <- elo.seq(xdata$winner, xdata$loser, xdata$Date, startvalue = ores$best)
eloplot(res1)
eloplot(res2)
```

---

## presence_summary *Summarize presence data*

---

### Description

Summarize presence data

### Usage

```
presence_summary(presence, from = NULL, to = NULL)
```

### Arguments

| | |
|---|---|
| `presence` | a data.frame with one date column (needs to be named "Date") and columns for each individual with 0/1 indicating absence/presence of that individual on that date |
| `from` | character indicating the beginning of the period to be considered (by default the first date in the Date column) |
| `to` | character indicating the end of the period to be considered (by default the last date in the Date column) |

## Details

If an individual left and/or joined multiple times, this will be indicated by the `stint` column.

The `init` column marks those individuals that were present on the beginning of the period considered.

## Value

a data.frame with entries for each individual indicating the first and last dates of their stays.

## Examples

```
data(advpres)
presence_summary(advpres)

presence_summary(advpres, from = "2010-01-27", to = "2010-02-02")
```

---

# print.elo *prints its argument*

---

## Description

prints its argument

## Usage

```
## S3 method for class 'elo'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | result from `elo.seq` |
| `...` | further arguments passed to or from other methods |

## Author(s)

Christof Neumann

## Examples

```
data(adv)
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
print(SEQ)
```

## print.seqchecknopres *prints its argument*

### Description

prints its argument

### Usage

```
## S3 method for class 'seqchecknopres'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | result from seqcheck |
| ... | further arguments passed to or from other methods |

### Author(s)

Christof Neumann

### Examples

```
data(adv)
print(seqcheck(winner = adv$winner, loser = adv$loser, Date = adv$Date))
```

## print.sequencecheck *prints its argument*

### Description

prints its argument

### Usage

```
## S3 method for class 'sequencecheck'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | result from seqcheck |
| ... | further arguments passed to or from other methods (ignored here) |

### Author(s)

Christof Neumann

## Examples

```
data(adv)
data(advpres)
print(seqcheck(winner = adv$winner, loser = adv$loser, Date = adv$Date,
                presence = advpres))
```

---

prunk *unknown relationships*

---

## Description

unknown relationships

## Usage

```
prunk(eloobject, daterange = NULL)
```

## Arguments

| | |
|---|---|
| eloobject | output from elo.seq or a matrix, e.g. from creatematrix |
| daterange | date range to be considered (character or Date of length 2), by default considers the entire date range of the sequence. In case the function works on a matrix this is ignored. |

## Value

numeric, proportion of unknown relationships (and total N) when considering all possible dyads, and the same after accounting for co-residency. For matrices, considering co-residency is ignored.

## Author(s)

Christof Neumann

## Examples

```
data(adv); data(advpres)
x <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date, presence = advpres)
prunk(x, c("2010-01-01", "2010-01-15"))
mat <- creatematrix(x, c("2010-01-01", "2010-01-15"))
prunk(mat)
```

randomelo *calculate Elo ratings from an interaction matrix*

## Description

calculate Elo ratings from an interaction matrix based on randomly generated sequences

## Usage

```
randomelo(
  interactionmatrix,
  runs = 2000,
  normprob = TRUE,
  k = 100,
  progressbar = FALSE
)
```

## Arguments

| | |
|---|---|
| interactionmatrix | square interaction matrix with winner in rows and losers in columns, for example the output from `creatematrix` |
| runs | number of randomly generated sequences based on the interactions in the `interactionmatrix` |
| normprob | logical (by default `TRUE`). Should a normal curve be assumed for calculating the winning/losing probablities, or a logistic curve. See `winprob` for details |
| k | numeric, factor *k* that determines the maximum change in ratings. By default `k=100` |
| progressbar | logical, should progress bars be displayed, by default `progressbar=TRUE` |

## Value

list of length 2. The first element contains a matrix with the final rating of each individual from each random sequence. IDs are in the columns, each run is represented as one row. The second element of the list contains the original interaction matrix.

## Author(s)

Christof Neumann

## Examples

```
data(adv)
elores <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date)
mat <- creatematrix(elores)
res <- randomelo(mat, 10)
data.frame(ID = colnames(res[[1]]), avg = round(colMeans(res[[1]]), 1))
```

randomeloextract *extract ratings from random sequences based on an interaction matrix*

## Description

extract ratings from random sequences based on an interaction matrix

## Usage

```
randomeloextract(x, ID, mode = c("obj", "samp", "avg"))
```

## Arguments

| | |
|---|---|
| x | output from randomelo |
| ID | character, ID |
| mode | character, one of three: 1) "obj": a random value from all the sequences; 2) "samp": a random value sampled from a normal distribution with mean and sd of all randomized values; 3) "avg": the average value from all the runs |

## Value

numeric

## Author(s)

Christof Neumann

## Examples

```
data(adv)
elores <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date)
mat <- creatematrix(elores)
res <- randomelo(mat, runs = 10)
randomeloextract(res, "a", "samp")
randomeloextract(res, "a", "obj")
randomeloextract(res, "a", "avg")
```

`randomsequence` *random dominance interaction sequence*

## Description

create a random dominance sequence for testing or simulations

## Usage

```
randomsequence(
  nID = 10,
  avgIA = 20,
  startdate = as.Date("2000-01-01"),
  alphabet = TRUE,
  reversals = 0.1,
  ties = NULL,
  presence = NULL
)
```

## Arguments

| | |
|---|---|
| `nID` | integer, number of IDs, must be less than 2601 |
| `avgIA` | numeric, average number of interactions an individual is involved in |
| `startdate` | character, a start date, by default `"2000-01-01"` |
| `alphabet` | logical, should the individual within an interaction that comes first in alphabetical order be the winner? By default `TRUE`, which gives some orderliness in the hierarchy |
| `reversals` | numeric, proportion of interactions that ends in reversed outcomes, i.e. the initial winner (if `alphabet=TRUE`) the `first` according to `alphanumeric` order) is changed into the loser. By default `0.1` |
| `ties` | numeric, proportion of interactions that ends undecided |
| `presence` | numeric vector of length 2. The first value indicates what proportion of individuals are absent for some time. The second value indicates the proportion of time (days) these individuals are absent |

## Value

an object of class `randomsequence`, which is a list with the following items:

| | |
|---|---|
| `seqdat` | an interaction sequence |
| `pres` | a presence matrix, actually a `data.frame` |

## Author(s)

Christof Neumann

## Examples

```
IA <- randomsequence()
SEQ <- elo.seq(winner = IA$seqdat$winner, loser = IA$seqdat$loser, Date = IA$seqdat$Date,
               runcheck = FALSE, progressbar = FALSE)
stab_elo(SEQ)
#
IA <- randomsequence(presence = c(0.5, 0.5))
SEQ <- elo.seq(winner = IA$seqdat$winner, loser = IA$seqdat$loser, Date = IA$seqdat$Date,
               presence = IA$pres, runcheck = FALSE, progressbar = FALSE)
stab_elo(SEQ)
```

---

# scale_elo *standardize Elo ratings*

---

## Description

standardize Elo ratings between 0 and 1

## Usage

```
scale_elo(x)
```

## Arguments

| | |
|---|---|
| x | numeric, a vector of Elo ratings |

## Value

a numeric vector of Elo ratings, which are scaled between 0 and 1, with the highest rating that is supplied becoming 1, the lowest becoming 0, and all others being proportionally scaled in between

## Author(s)

Christof Neumann

## Examples

```
data(adv)
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
extract_elo(SEQ, "2010-01-30")
extract_elo(SEQ, "2010-01-30", standardize=TRUE)

# same as
scale_elo(extract_elo(SEQ, "2010-01-30"))
```

seqcheck *runs raw data diagnostics for Elo rating*

## Description

runs some diagnostics on the data supplied to elo.seq, to check whether elo.seq will run without errors

## Usage

```
seqcheck(winner, loser, Date, draw = NULL, presence = NULL)
```

## Arguments

| | |
|---|---|
| winner | either a factor or character vector with winner IDs of dyadic dominance interactions |
| loser | either a factor or character vector with loser IDs of dyadic dominance interactions |
| Date | character vector of form "YYYY-MM-DD" with the date of the respective interaction |
| draw | logical, which interactions ended undecided (i.e. drawn or tied)? By default all FALSE, i.e. no undecided interactions occurred. Note that in this case, winner/loser values can be interchanged |
| presence | optional data.frame, to supply data about presence and absence of individuals for part of the time the data collection covered. see details |

## Details

calender dates (for the sequence as well as in the first column of presence, if supplied) need to be in "YYYY-MM-DD" format!

seqcheck will return two types of messages: warnings and errors. Errors will result in the data NOT working when supplied to elo.seq, and need to be fixed. Warning message do not necessarily lead to failure of executing elo.seq. Note that by default seqcheck is part of elo.seq. If any error or warning is produced by seqcheck, these data will not work in elo.seq. Some warning (but not error) messages can be ignored (see below) and if the runcheck argument in elo.seq is set to FALSE Elo-ratings will be calculated properly in such cases.

The actual checks (and corresponding messages) that are performed are described in more detail here:

Most likely (i.e. in our experience), problems are caused by mismatches between the interaction data and the corresponding presence data.

Errors:
Presence starts AFTER data: indicates that during interactions at the beginning of the sequence, no corresponding information was found in the presence data. Solution: augment presence data, or remove interactions until the date on which presence data starts

`Presence stops BEFORE data`: refers to the corresponding problem towards the end of interaction and presence data

`During the following interactions, IDs were absent...`: indicates that according to the presence data, IDs were absent (i.e. "0"), but interactions with them occured on the very date(s) according to the interaction data

`The following IDs occur in the data sequence but NOT...`: there is/are no columns corresponding to the listed IDs in the presence data

`There appear to be gaps in your presence (days missing?)...`: check whether your presence data includes a line for *each date* starting from the date of the first interaction through to the date of the last interaction

Warnings:

`Presence continues beyond data`: indicates that presence and interaction data do not end on the same date.

`Presence starts earlier than data`: indicates that presence and interaction data do not start on the same date.

`The following IDs occur in the presence data but NOT...`: there are more ID columns in the presence data than IDs occuring in the interaction data

`Date column is not ordered`: The dates are not supplied in ascending order. `elo.seq` will still work but the results won't be reliable because the interactions were not in the correct sequence.

Other warnings/errors can result from inconsistencies in either the presence or sequence data, or be of a more general nature:

Errors:

`No 'Date' column found`: in the presence data, no column exists with the name/header "Date". Please rename (or add) the necessary column named "Date" to your presence data.

`At least one presence entry is not 1 or 0`: presence data must come in binary form, i.e. an ID was either present ("1") or absent ("0") on a given date. No `NA`s or other values are allowed.

`Your data vectors do not match in length`: at least one of the three mandatory arguments (winner, loser, Date) differs from one other in length. Consider handling your data in a data.frame, which avoids this error.

Warnings:

`IDs occur in the data with inconsistent capitalization`: because `R` is case-sensitive, "A" and "a" are considered different individuals. If such labelling of IDs is on purpose, ignore the warning and set `runcheck=FALSE` when calling `elo.seq()`

`There is (are) X case(s) in which loser ID equals winner ID`: winner and loser represent the same ID

`The following individuals were observed only on one day`: while not per se a problem for the calculation of Elo ratings, individuals that were observed only on one day (irrespective of the number of interactions on that day) cannot be plotted. `eloplot` will give a warning in such cases, too.

## Value

returns textual information about possible issues with the supplied data set, or states that data are fine for running with `elo.seq`

### Author(s)

Christof Neumann

### Examples

```
data(adv)
seqcheck(winner = adv$winner, loser = adv$loser, Date = adv$Date)
data(advpres)
seqcheck(winner = adv$winner, loser = adv$loser, Date = adv$Date,
         presence = advpres)

# create faulty presence data
# remove one line from presence data
faultypres <- advpres[-1, ]
# make all individuals absent on one day
faultypres[5, 2:8] <- 0
# run check
seqcheck(winner = adv$winner, loser = adv$loser, Date = adv$Date,
         presence = faultypres)

# fix first error
faultypres <- rbind(faultypres[1, ], faultypres)
faultypres$Date[1] <- "2010-01-01"

# run check again
seqcheck(winner = adv$winner, loser = adv$loser, Date = adv$Date,
         presence = faultypres)

# fix presence on date for interaction number 6
faultypres[6, 2:8] <- 1

# run check again
seqcheck(winner = adv$winner, loser = adv$loser, Date = adv$Date,
         presence = faultypres)
# all good now
```

---

## simple_dom *simple dominance indices*

---

### Description

simple dominance indices

### Usage

```
simple_dom(winner, loser, Date = NULL, daterange = NULL)
```

### Arguments

| | |
|---|---|
| winner | character or factor with winner |
| loser | character or factor with winner |
| Date | not yet implemented |
| daterange | not yet implemented |

### Details

The indices that are calculated are the following

winprop the proportion of all interactions won

domover the proportion of individuals dominated (regardless of whether any interactions may have occured, i.e. the number of individuals dominated is divided by N - 1 for all individuals)

domover_rel the proportion of individuals dominated with which the focal interacted

### Value

a data.frame with one row per individual and several 'simple' dominance indices

### Examples

```
xdata <- randomsequence(nID = 10, avgIA = 20, reversals = 0.2)$seqdat
simple_dom(xdata$winner, xdata$loser)
```

---

## stab_elo *stability index* S

---

### Description

calculates the *S* index as metric for the overall stability of a hierarchy during a specified time period

### Usage

```
stab_elo(
  eloobject,
  from = min(eloobject$stability$date),
  to = max(eloobject$stability$date),
  weight = TRUE
)
```

## Arguments

| | |
|---|---|
| eloobject | an object of class "elo", usually the result of a call to elo.seq |
| from | character, from which date onwards should *S* be calculated. By default the first date in the sequence is used |
| to | character, until which date should *S* be calculated. By default the last date in the sequence is used |
| weight | logical, should single rank changes be weighted by the Elo rating of the highest-rated individual involved in a rank change? Default is TRUE |

## Details

*S* ranges between 0 and 1, where 0 indicates an unstable hierarchy, in which the ordering reverses every other day, and 1, in which the ordering is stable and no rank changes occur.

In contrast to the originally proposed *S*, this version is now standardized between 0 and 1, and additionally, the interpretation is reversed, i.e. 1 refers to stable situations, whereas values closer to 0 indicate more instable hierarchies

## Value

returns the *S* index

## Author(s)

Christof Neumann

## References

Neumann C, Duboscq J, Dubuc C, Ginting A, Irwan AM, Agil M, Widdig A, Engelhardt A (2011). "Assessing dominance hierarchies: validation and advantages of progressive evaluation with elo-rating." *Animal Behaviour*, **82**, 911-921. doi:10.1016/j.anbehav.2011.07.016.

McDonald DB, Shizuka D (2013). "Comparative transitive and temporal orderliness in dominance networks." *Behavioral Ecology*, **24**, 511-520. doi:10.1093/beheco/ars192.

## Examples

```
data(adv)
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
stab_elo(SEQ)
stab_elo(SEQ, weight=FALSE)
stab_elo(SEQ, from="2010-01-20", to="2010-01-30")
stab_elo(SEQ, from="2010-01-20", to="2010-01-30", weight=FALSE)
```

steepness *hierarchy steepness based on David's scores*

## Description

hierarchy steepness based on David's scores

## Usage

```
steepness(mat, nrand = 0, Dij = TRUE, returnfig = FALSE)
```

## Arguments

| | |
|---|---|
| mat | square dominance matrix |
| nrand | numeric, the number of runs for the randomization test |
| Dij | logical, by default TRUE i.e. corrected for number of interactions in a dyad, otherwise simple proportion of wins/losses |
| returnfig | logical, should a figure be produced that shows the distribution of expected steepness |

## Value

a named vector, with the observed steepness, the expected steepness, p-value and the number of randomizations used

## Author(s)

Christof Neumann

## References

de Vries H, Stevens JMG, Vervaecke H (2006). "Measuring and testing the steepness of dominance hierarchies." *Animal Behaviour*, **71**, 585-592. doi:10.1016/j.anbehav.2005.05.015.

## Examples

```
data(bonobos)
steepness(bonobos) # no randomization test

# with randomization test
steepness(bonobos, nrand = 100)
```

## summary.elo *summarize elo object*

### Description

summarize elo object

### Usage

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

### Arguments

| | |
|---|---|
| object | an object of class "elo", usually the result of a call to elo.seq |
| ... | further arguments passed to or from other methods (ignored) |

### Author(s)

Christof Neumann

### Examples

```
IA <- randomsequence()
SEQ <- elo.seq(winner=IA$seqdat$winner, loser=IA$seqdat$loser,
               Date=IA$seqdat$Date, draw=IA$seqdat$Draw,
               presence=IA$pres)
summary(SEQ)
```

## traj_elo *calculate dominance trajectory*

### Description

calculate individual Elo rating trajectory over time

### Usage

```
traj_elo(
  eloobject,
  ID,
  from = min(eloobject$stability$date),
  to = max(eloobject$stability$date)
)
```

### Arguments

| | |
|---|---|
| eloobject | result from elo.seq |
| ID | character, the ID(s) of the individual(s) |
| from | character, from which date onwards should the trajectory be calculated. By default the first date in the sequence is used |
| to | character, until which date should the trajectory be calculated. By default the last date in the sequence is used |

### Value

A data.frame with as many lines as specified in ID, columns for ID, date range, the actual slope (trajectory), and the number of observed interactions within the date range

### Author(s)

Christof Neumann

### Examples

```
data(adv)
SEQ <- elo.seq(winner = adv$winner, loser = adv$loser, Date = adv$Date)
traj_elo(SEQ, "a")

traj_elo(SEQ, "a", from = "2010-01-20", to = "2010-01-30")

# no slope available if ID was not observed interacting
# inside the date range
traj_elo(SEQ, "a", from = "2010-01-17", to = "2010-01-18")

# no slope available if ID was only observed interacting
# once within the date range
traj_elo(SEQ, "a", from = "2010-01-17", to = "2010-01-19")

# for several individuals
traj_elo(SEQ, c("a", "b", "c"))
```

---

## transitivity *triangle transitivity*

---

### Description

triangle transitivity

### Usage

```
transitivity(m, runs = 2000, returnfig = FALSE)
```

### Arguments

| | |
|---|---|
| m | square dominance matrix |
| runs | numeric, the number of runs for the randomization test |
| returnfig | logical, should a figure be produced that shows the distribution of expectation |

### Value

a named vector of length four

### References

Shizuka D, McDonald DB (2012). "A social network perspective on measurements of dominance hierarchies." *Animal Behaviour*, **83**, 925-934. doi:10.1016/j.anbehav.2012.01.011.

https://shizukalab.com/r/triangle-transitivity-in-dominance-hierarchies-directed-graphs/

### Examples

```
data(bonobos)
transitivity(bonobos)
```

---

## winprob *expected winning probability*

---

### Description

calculate expected probability of winning given known strengths of two opponents

### Usage

```
winprob(elo1, elo2, normprob = TRUE, fac = NULL)
```

### Arguments

| | |
|---|---|
| elo1 | Elo rating from individual for which the winning probability should be calculated |
| elo2 | Elo rating of the opponent |
| normprob | logical (by default TRUE). Should a normal curve be assumed for calculating the winning/losing probablities, or not (see details). |
| fac | numeric (by default NULL). A scaling factor (see details) |

## Details

Elo (1978) proposed three ways of calculating winning probabilities (section 8.73), one of which (the 'linear' approach) is ignored here because it "lacks the sophistication and flexibility to express the limitation on D [rating difference] and the deflation controls required for integrity of the ratings". Between the two remaining approaches (normal and logistic), Elo favored initially the normal over the logistic function, though he writes that the logistic function "better reflects large deviations in an extended series". Because of Elo's initial preference, the default approach taken by the package's functions is the normal one, though it can be changed to the logistic one if desired.

In the meantime, several studies have used an addtional approach to calculate winning probabilities, which is based on an exponential distribution. This can be invoked by setting `normprob = FALSE` and `fac` to some number. The value I have seen used is 0.01 (Franz et al. 2015). Sánchez-Tójar et al. (2018) refer to it as `sigmoid.param` in their `aniDom` package. Goffe et al. (2018) also use this approach but their scaling factor is 1 (referred to as `diff_f`) because their ratings are on a completely different scale.

Finally, this function is for demonstration only, i.e. it is not used anywhere in the package (other than in vignettes). As such, the functions in the package (most importantly `e.single`) only allow the two primary options for the calculation of winning probabilities (for now).

## Value

numeric, expected chance of first individual to win an interacation with the second individual

## Author(s)

Christof Neumann

## References

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

Franz M, McLean E, Tung J, Altmann J, Alberts SC (2015). "Self-organizing dominance hierarchies in a wild primate population." *Proceedings of the Royal Society B: Biological Sciences*, **282**, 20151512. doi:10.1098/rspb.2015.1512.

Sánchez-Tójar A, Schroeder J, Farine DR (2018). "A practical guide for inferring reliable dominance hierarchies and estimating their uncertainty." *Journal of Animal Ecology*, **87**, 594-608. doi:10.1111/13652656.12776.

Goffe AS, Fischer J, Sennhenn-Reulen H (2018). "Bayesian inference and simulation approaches improve the assessment of Elo-ratings in the analysis of social behaviour." *Methods in Ecology and Evolution*, **9**, 2131-2144. doi:10.1111/2041210X.13072.

## Examples

```
winprob(1200,1000)
winprob(1000,1200)
winprob(1000,1000)
winprob(1200,1000, normprob = FALSE)
winprob(1000,1200, normprob = FALSE)
winprob(1000,1000, normprob = FALSE)
winprob(1200,1000, normprob = FALSE, fac = 0.01)
```

```
winprob(1000,1200, normprob = FALSE, fac = 0.01)
winprob(1000,1000, normprob = FALSE, fac = 0.01)

# compare different algorithms visually
w <- rep(0, 1001) # winner rating: constant
l <- w - 0:1000 # loser rating: varying

elonorm <- numeric(length(w))
eloexpo <- numeric(length(w))
eloopti <- numeric(length(w))
eloopti2 <- numeric(length(w))

for(i in 1:length(w)) {
  elonorm[i] <- winprob(w[i], l[i], normprob = TRUE)
  eloexpo[i] <- winprob(w[i], l[i], normprob = FALSE)
  eloopti[i] <- winprob(w[i], l[i], normprob = FALSE, fac = 0.01)
  eloopti2[i] <- winprob(w[i], l[i], normprob = FALSE, fac = 0.005)
}

plot(0, 0, type = "n", las = 1, yaxs = "i",
     xlim = c(0, 1000), ylim = c(0.5, 1),
     xlab = "rating difference",
     ylab = "winning probability")
points(abs(l), elonorm, "l", col = "#4B0055", lwd = 3)
points(abs(l), eloexpo, "l", col = "#007094", lwd = 3)
points(abs(l), eloopti, "l", col = "#00BE7D", lwd = 2)
points(abs(l), eloopti2, "l", col = "#FDE333", lwd = 2)

legend("bottomright",
     legend = c("normal", "logistic", "exponential (fac = 0.01)", "exponential (fac = 0.005)"),
       col = c("#4B0055", "#007094", "#00BE7D", "#FDE333"),
       lwd = 2,
       cex = 0.9)
```

# Index