Package ‘EloRating’

February 19, 2015

Type Package

Title Animal Dominance Hierarchies by Elo Rating

Version 0.43

Depends zoo

Date 2014-11-23

Author Christof Neumann & Lars Kulik

Maintainer Christof Neumann <christofneumann1@gmail.com>

Description Calculate Elo ratings as means to describe animal dominance hierarchies

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-11-23 23:36:52

R topics documented:

EloRating-package .................................................. 2
adv ................................................................. 3
bonobos ............................................................ 4
creatematrix ......................................................... 4
DS ................................................................. 6
e.single ............................................................ 7
e.lo.seq ............................................................ 8
e.lo.plot .......................................................... 10
extract.elo ......................................................... 11
individuals ......................................................... 12
lastdaypresent .................................................... 13
print ............................................................... 14
print.elo ........................................................... 15
prunk ............................................................... 15
randomelo ......................................................... 16
randomeloextract ............................................... 17
randomsequence ............................................... 18
**EloRating-package**

Description

Calculate Elo ratings as means to describe animal dominance hierarchies

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>EloRating</th>
<th>Type:</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>0.43</td>
<td>Date:</td>
<td>2014-11-23</td>
</tr>
<tr>
<td>License:</td>
<td>GPL (&gt;= 2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Author(s)

Christof Neumann & Lars Kulik
Maintainer: Christof Neumann <christofneumann1@gmail.com>

References


Examples

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

Date  Date of the interaction
Time  Time of the interaction (not mandatory for calculations)
winner  Winner of the interaction
loser  Loser of the interaction
tie  indicates whether the interaction ended undecided, i.e. in a draw/tie

Details

adv is the data set of fictional interactions that was used by Albers and de Vries (2001). adv2 adds to this fictional data to indicate whether an interaction ended undecided (draw). advpres is a fictional presence data set, indicating which individuals were present and absent on which date.

Source


References


Examples

data(adv)
dominance matrix from de Vries et al. 2006

Description
Dominance matrix of seven bonobos

Usage
data(bonobos)

Format
Dominance matrix of seven bonobos

Source

References

Examples
data(bonobos)

Description
create a dominance matrix from the underlying observed sequence

Usage
creatematrix(eloobject, daterange=NULL,
             drawmethod="omit", onlyinteracting=FALSE)
creatematrix

Arguments

- eloobject: output from elo.seq
- daterange: date range to which the matrix should correspond (default from beginning to end of sequence)
- drawmethod: "omit" = undecided interactions (draws/ties) are ignored (default) "0.5" = each undecided is counted half a win for each dyad member "1" = each undecided interaction is counted twice, i.e., as win for both individuals
- onlyinteracting: logical. Indicating whether all individuals that were present (default) 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.

Value

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

Author(s)

- Christof Neumann

Examples

data(adv)
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)

## 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")
**DS**

*calculate David's scores*

---

**Description**

calculate David’s scores

**Usage**

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

**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, otherwise the raw proportion

**Value**

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

**Author(s)**

Christof Neumann

**References**


- more appropriate dominance ranking method than Clutton-Brock et al.’s index. Animal Behaviour,
- 66, 601-605.

- de Vries, H., Stevens, J. M. G. and Vervaecke, H. 2006. Measuring and testing the steepness of

**Examples**

```r
data(adv)
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
mat <- creatematrix(SEQ)
DS(mat)
```
calculate Elo ratings for a single interaction

Description

calculate Elo ratings for a single interaction

Usage

e.single(EL01old, ELO2old, outcome, k = 100)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL01old</td>
<td>Elo rating of the first individual</td>
</tr>
<tr>
<td>ELO2old</td>
<td>Elo rating of the second individual</td>
</tr>
</tbody>
</table>
| outcome  | "1" = first individual wins and second looses  
|          | "2" = second individual wins and first looses  
|          | "0" = interaction ends in a draw/tie (no winner and no looser) |
| k        | k factor |

Value

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

Author(s)

Christof Neumann

References


Examples

e.single(EL01old=1200, ELO2old=1000, outcome=1, k=100)  
# same as before  
e.single(EL01old=1000, ELO2old=1200, outcome=2, k=100)  
# an undecided interaction  
e.single(EL01old=1200, ELO2old=1000, outcome=0, k=100)  
# if rating differences are too big, no change occurs  
# if higher-rated individual wins  
e.single(EL01old=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(EL01old=2000, ELO2old=1000, outcome=2, k=100)
elo.seq  

*Calculate Elo ratings*

**Description**

Calculate Elo ratings from a sequence of dominance interactions

**Arguments**

- **winner**: Either a factor or character vector with winners IDs of dominance interactions
- **loser**: Either a factor or character vector with loser IDs of dominance interactions
- **date**: character vector of form "YYYY-MM-DD" with the date of the respective interaction
- **draw**: logical. Did the interaction end 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**: option 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 $k=100$
- **init**: 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, 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" 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**: should several checks regarding data integrity be performed, by default TRUE. See seqcheck
- **progressbar**: should progress bars be displayed, by default progressbar=TRUE
Details

the presence 'matrix' is actually an object of class data.frame containing information about whether 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 info.

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 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 & Lars Kulik

References


Examples

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

Elo rating plotting

Description

Plotting 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**: "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**: by default ("yes") plot interpolated Elo values or plot Elo values without interpolation ("no").
- **from**: "start": the plotted date range will start at the first date of the dataset or provide a custom date ("YYYY-MM-DD").
- **to**: "end": the plotted date range will end at the last date of the dataset or provide a custom date ("YYYY-MM-DD").
- **color**: 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.

Author(s)

Lars Kulik & 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, standardize = FALSE, IDs = NULL, NA.interpolate = FALSE, daterange = 1)

**Arguments**

- **eloobject**: result from `elo.seq`
- **extractdate**: date on which Elo ratings should be obtained, defaults to the last day in the dataset
- **standardize**: should the returned ratings be scaled between 0 and 1. Default is FALSE. See `scale.elo`
- **IDs**: 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...

**Value**

named (IDs) vector of (average) Elo ratings

**Author(s)**

Christof Neumann

**Examples**

```r
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
```
individuals

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)

<table>
<thead>
<tr>
<th>individuals</th>
<th>individuals present in the group</th>
</tr>
</thead>
</table>

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

eoloobject | result from elo.seq
from | from which date onwards should the ID statistics be calculated. By default the first date in the sequence is used

to | 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 | 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 considered another measure of stability on the group level.
**lastdaypresent**

**Value**

numeric or character

**Author(s)**

Christof Neumann

**Examples**

```r
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
```

```r
# 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
```

```r
# 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")
```

---

**lastdaypresent**

*last day an individual was present*

**Description**

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

**Usage**

```r
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 sequence
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 reference date is specified.

Value

Date or NA

Author(s)

Christof Neumann

Examples

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

---

print

prints its argument

Description

prints its argument

Usage

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

Arguments

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

Author(s)

Christof Neumann

Examples

```r
data(adv); data(advpres)
SEQC <- seqcheck(winner=adv$winner, loser=adv$loser, Date=adv$Date, presence=advpres)
print(SEQC)
```
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)

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.
**randomelo**

**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 (not possible...)

**Author(s)**

Christof Neumann

**Examples**

```r
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)
```

**Description**

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

**Usage**

```r
randomelo(interactionmatrix, runs=2000)
```

**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 matrix

**Value**

list of length 2. The first element contains a matrix with the final ratings 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
randomeloextract

Examples

data(adv)
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
mat <- creatematrix(SEQ)
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
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)
SEQ <- elo.seq(winner=adv$winner, loser=adv$loser, Date=adv$Date)
mat <- creatematrix(SEQ)
res <- randomelo(mat, 10)
randomeloextract(res, "a", "samp")
randomeloextract(res, "a", "obj")
randomeloextract(res, "a", "avg")
randomsequence  
*create a random dominance sequence*

**Description**
create a random dominance sequence for testing or simulations

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

**Arguments**
- **nID**  
  number of IDs, must be less than 2601.
- **avgIA**  
  average number of interactions and individual is involved in.
- **startdate**  
  a start date
- **alphabet**  
  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**  
  this proportion of interactions ends in reversed outcomes, i.e. the individual that comes first in the alphabet looses. Default = 0.1.
- **ties**  
  this proportion of interactions ends undecided.
- **presence**  
  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)
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)
stab.elo(SEQ)
## scale.elo

**standardize Elo ratings**

### Description

standardize Elo ratings between 0 and 1

### Usage

```r
## S3 method for class 'elo'
scale(x, center = TRUE, scale = TRUE)
```

### Arguments

- `x`: a vector of Elo ratings.
- `center`: ignored.
- `scale`: ignored.

### Value

returns a 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

```r
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  factor or character vector of winner IDs
loser   factor or character vector of loser IDs
Date    character vector of form "YYYY-MM-DD" with the date of the respective interaction
draw    logical (of length(winner)). Did did the interaction end undecided (i.e. drewed or tied)? By default all FALSE, i.e. no undecided interactions occurred
presence data.frame with presence data, see elo.seq

Details

calendar 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 occurred 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 occurring in the interaction data

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 NAs 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. elo.plot will give a warning in such cases, too.

Value
returns info about possible errors, or states that data are fine for running with elo.seq

Author(s)
Christof Neumann

Examples

```r
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
faultypres <- advpres[-1, ]
faultypres[5,2:8] <- 0
# seqcheck(winner=adv$winner, loser=adv$loser, Date=adv$Date,
#         presence=faultypres)
```
stab. elo

**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**: from which date onwards should $S$ be calculated. By default the first date in the sequence is used.
- **to**: until which date should $S$ be calculated. By default the last date in the sequence is used.
- **weight**: 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**


summary.elo

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)

---

summary.elo  summarize elo object

Description

overview of 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.

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`: the ID(s) of the individual(s)
- `from`: from which date onwards should the trajectory be calculated. By default the first date in the sequence is used.
- `to`: 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**

```r
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"))
```
**Description**

calculate expected probability of winning given known strength of both opponents

**Usage**

`winprob(elo1, elo2)`

**Arguments**

- `elo1` : Elo rating from individual for which the winning probability should be calculated
- `elo2` : Elo rating of the opponent

**Value**

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

**Author(s)**

Christof Neumann

**References**


**Examples**

- `winprob(1200, 1000)`
- `winprob(1000, 1200)`
- `winprob(1000, 1000)`
Index

*Topic `textasciitildedekwd1`
- e.single, 7
- extract.elo, 11
- print, 14
- print.elo, 15
- randomsequence, 18
- scale.elo, 19
- seqcheck, 19
- traj.elo, 24

*Topic `textasciitildedekwd2`
- e.single, 7
- extract.elo, 11
- print, 14
- print.elo, 15
- randomsequence, 18
- scale.elo, 19
- seqcheck, 19
- traj.elo, 24

*Topic `datasets`
- adv, 3
- bonobos, 4

*Topic `package`
- EloRating-package, 2

- adv, 3
- adv2 (adv), 3
- advpres, 9
- advpres (adv), 3

- bonobos, 4

- creatematrix, 4, 6, 15, 16

- DS, 6

- e.single, 7
- elo.seq, 5, 8, 10–13, 15, 19–24
- eloplot, 10, 21
- EloRating (EloRating-package), 2
- EloRating-package, 2
- extract.elo, 11

individuals, 12
lastdaypresent, 13
print, 14
print.elo, 15
prunk, 15
randomelo, 16, 17
randomeloextract, 17
randomsequence, 18
scale.elo, 11, 19
seqcheck, 8, 14, 19
stab.elo, 9, 22
summary.elo, 23
traj.elo, 24
winprob, 25