Package 'compete'

August 29, 2016

Title Analyzing Social Hierarchies

Type Package

Version 0.1

Author c(person(``James P.'', ``Curley'', role = c(``aut'', ``cre''),
email = ``jc3181@columbia.edu''))

Maintainer James P. Curley <jc3181@columbia.edu>

Description Organizing and Analyzing Social Dominance
Hierarchy Data.

Depends R (>= 3.1.0)

License GPL-3

LazyData true

Imports graphics, igraph, sna, stats, utils

URL https://github.com/jalapic/compete

BugReports https://github.com/jalapic/compete

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-17 21:12:20

R topics documented:

---

bonobos *Bonobo sociomatrix*

Description

A win-loss sociomatrix with each row being the number of wins against individual bonobos in each column.

Usage

```
bonobos
```

Format

A data frame with 6 rows and 6 variables:

Dz The number of losses by bonobo Dz against all other bonobos

He The number of losses by bonobo He against all other bonobos

De The number of losses by bonobo De against all other bonobos

Ho The number of losses by bonobo Ho against all other bonobos

Lu The number of losses by bonobo Lu against all other bonobos

Ki The number of losses by bonobo Ki against all other bonobos

Details

Reference: Vervaecke, H., de Vries, H. & van Elsacker, L. 2000. Dominance and its behavioral measures in a captive group of bonobos (Pan paniscus). International Journal of Primatology, 21, 47-68.

`caribou` *Caribou sociomatrix*

Description

A win-loss sociomatrix with each row being the number of wins against individual bonobos in each column.

Usage

```
caribou
```

Format

A data frame with 20 rows and 20 variables:

a The number of losses by caribou a against all other caribou

b The number of losses by caribou b against all other caribou

c The number of losses by caribou c against all other caribou

d The number of losses by caribou d against all other caribou

e The number of losses by caribou e against all other caribou

f The number of losses by caribou f against all other caribou

g The number of losses by caribou g against all other caribou

h The number of losses by caribou h against all other caribou

i The number of losses by caribou i against all other caribou

j The number of losses by caribou j against all other caribou

k The number of losses by caribou k against all other caribou

l The number of losses by caribou l against all other caribou

m The number of losses by caribou m against all other caribou

n The number of losses by caribou n against all other caribou

o The number of losses by caribou o against all other caribou

p The number of losses by caribou p against all other caribou

q The number of losses by caribou q against all other caribou

r The number of losses by caribou r against all other caribou

s The number of losses by caribou s against all other caribou

t The number of losses by caribou t against all other caribou

Details

Reference: Barrette, C. & Vandal, D. 1986. Social rank, dominance, antler size, and access to food in snow-bound wild woodland caribou. Behaviour, 97, 118-146.

dci *Get the directional consistency index (DCI) of a sociomatrix.*

Description

Get the directional consistency index (DCI) of a sociomatrix.

Usage

```
dci(m)
```

Arguments

m	A matrix with individuals ordered identically in rows and columns.

Value

The directional consistency of m.

References

Van Hooff JARAM, Wensing JAB. 1987. Dominance and its behavioural measures in a captive wolf pack. In: Frank HW, editor. Man and Wolf. Dordrecht, Olanda (Netherlands): Junk Publishers pp.219-252.

Further details

The DCI represents the proportion of occurrences of a behavior that occurs across all dyads in a group from the individual within each dyad performing the behavior with a higher frequency (H) to the individual within each dyad performing the behavior with a lower frequency (L). It is calculated by averaging the following formula across all dyads: DCI = (H - L)/(H + L). The DCI ranges from 0 (no directional asymmetry) to 1 (completely unidirectional).

Examples

```
m <- matrix(c(NA,2,30,6,19,122,0,NA,18,
0,19,85,0,1,NA,3,8,84,0,0,0,NA,267,50,0,
0,0,5,NA,10,1,0,4,4,1,NA), ncol=6) #table 2, Vervaecke et al. 2000  - fleeing in bonobos
dci(m)
```

dc_test — *Randomization Test of DC and Skew-Symmetry of a Sociomatrix.*

Description

Randomization Test of DC and Skew-Symmetry of a Sociomatrix.

Usage

```
dc_test(m, N = 20, ntimes = 10000)
```

Arguments

m	A matrix with individuals ordered identically in rows and columns.
N	The number of behaviors for each dyad
ntimes	Number of simulations

Value

A list containing p-value of each test, observed values and descriptive measures of randomized data.

References

Leiva D et al, 2008, Testing reciprocity in social interactions: A comparison between the directional consistency and skew-symmetry statistics, Behav Res Methods.

Further details

This is a one-sided significance test i.e. that observed values are higher than expected

Examples

```
m <- matrix(c(NA,2,30,6,19,122,0,NA,18,
0,19,85,0,1,NA,3,8,84,0,0,0,NA,267,50,0,
0,0,5,NA,10,1,0,4,4,1,NA), ncol=6)  #table 2, Vervaecke et al. 2000  - fleeing in bonobos
dc_test(m)

mm = matrix(c(0,9,3,6,12,18,13,0,7,17,45,6,7,2,0,5,8,0,11,26,
12,0,8,6,4,3,0,1,0,2,2,5,0,1,0,0),6,6,TRUE)
dc_test(mm,N=16)
```

devries *Calculate the linearity of a dominance hierarchy - De Vries' method*

Description

Calculate the linearity of a dominance hierarchy - De Vries' method

Usage

```
devries(m, Nperms = 10000, history = FALSE, plot = FALSE)
```

Arguments

`m`	A matrix with individuals ordered identically in rows and columns.
`Nperms`	Number of randomizations
`history`	Whether to store results of randomization
`plot`	Whether to plot results of randomization

Value

The modified Landau's h' value of `m`, the associated p-value

References

Han de Vries (1995) An improved test of linearity in dominance hierarchies containing unknown or tied relationships. Animal Behaviour 50 pp. 1375-1389.

Further details

This code is an edited and faster version of code originally written by Dai Shizuka. http://biosci.unl.edu/daizaburo-shizuka Note that plot will only be shown if `history`**=F**

Examples

```
devries(bonobos)
devries(mouse,plot=TRUE)
devries(people,history=TRUE)
```

ds *Get David's Scores of Individuals*

Description

Get David's Scores of Individuals

Usage

```
ds(m, norm = FALSE, type = "D")
```

Arguments

m	A matrix with individuals ordered identically in rows and columns.
norm	whether to normalize scores
type	either `method="D"` for Dij or `method="P"` for Pij.

Value

a vector of David scores in same order as names of `m`.

References

Gammell et al, 2003, David's score: a more appropriate dominance ranking method than Clutton-Brock et al.'s index, Animal Behaviour.

Examples

```
m <- matrix(c(NA,2,30,6,19,122,0,NA,18,
0,19,85,0,1,NA,3,8,84,0,0,0,NA,267,50,0,
0,0,5,NA,10,1,0,4,4,1,NA), ncol=6)  #table 2, Vervaecke et al. 2000  - fleeing in bonobos
ds(m)
ds(m,type="P")
ds(m,norm=TRUE)
```

el *Winner-Loser Edgelist*

Description

A two variable dataframe with columns winner and loser

Usage

```
el
```

Format

A data frame with 2397 rows and 2 variables:

winner The winner of an individual contest

loser The loser of an individual contest

`get_di_matrix` *Transforms a frequency interaction sociomatrix (valued data) into a dichotomized 1/0 matrix*

Description

Transforms a frequency interaction sociomatrix (valued data) into a dichotomized 1/0 matrix

Usage

```
get_di_matrix(m, type = "wl")
```

Arguments

`m`	A matrix with individuals ordered identically in rows and columns.
`type`	Determines the type of dichotomized matrix to be returned. `type=`**"wl"** is the default which returns a win-loss matrix with a '1' representing a consistent winner and a '0' representing a consistent loser for each dyad of the matrix. A consistent winner is defined as being the individual in each dyad that has absolutely more wins than defeats. In the default condition if competitors have the same number of wins each, they both receive a 0. If `type=`**"wlties"** the default dichotomized win-loss matrix will be returned but it will also return 0.5 into cells for tied relationships. If `type=`**"wlties0"** the default dichotomized win-loss matrix will be returned but it will also return 0.5 into cells for tied relationships. Additionally, if two competitors never interacted both cells for that relationship will be returned with a 0. If `type=`**"wlbinom"** every relationship within the win-loss matrix is assessed for whether one competitor significantly wins more competitive interactions than the other competitor. Significance is calculated using a binomial test with probability of p=0.05. A '1' is given to significant winners within a relationship and a '0' is given to significant losers or if neither individual is a winner. If `type=`**"wlbinomties"** The same procedure is done as for `type=`**"wlbinom"**, but if no signficiant winner/loser can be determined then a 0.5 is returned rather than a 0. If `type=`**"pa"** the inputted matrix will be turned into a dichotomized presence-absence matrix, with a '1' indicating that the competitor in a the row of the matrix beat the competitor in the column at least once. A '0' indicates that that competitor never beat the other competitor. If `type=`**"dom"** the inputted matrix will be turned into a dominance score matrix, with a '1' indicating that the competitor in a the row of the matrix dominates the competitor in the column. A '-1' indicates that that competitor in a row is subordinate to the competitor in the column. A '0.5' indicates a tie. A '0' indicates an observational or structural zero.

Value

A dichotomized win/loss or presence/absence matrix.

References

Appleby, M. C. 1983. The probability of linearity in hierarchies. Animal Behaviour, 31, 600-608.

Examples

```
get_di_matrix(bonobos)
get_di_matrix(mouse)
```

`get_wl_df` *Converts results dataframe to win-loss dataframe*

Description

Converts results dataframe to win-loss dataframe

Usage

```
get_wl_df(df, ties = "remove")
```

Arguments

`df`	A results dataframe
`ties`	How to handle ties, default is `ties="remove"` Alternative is `ties="keep"`

Value

A win-loss dataframe

Further details

A results dataframe first 3 variables are id1, id2, result. Results can be "W", "L", or "T" or "1", "0", "0.5". The output will be a win-loss dataframe that will reorganize the first 3 variables into winner, loser and result (=1 for Win or =0.5 for ties).

Examples

```
get_wl_df(randomtourney(8))
```

`get_wl_matrix` *Converts win-loss dataframe to win-loss matrix*

Description

Converts win-loss dataframe to win-loss matrix

Usage

```
get_wl_matrix(df, ties = "remove")
```

Arguments

`df`	A results or win-loss dataframe
`ties`	How to handle ties, default is `ties="remove"` Alternative is `ties="keep"`

Value

A win-loss matrix

Further details

Input dataframes or matrices with only 2 columns are considered to be winners in column 1 and losers in column 2. If input dataframe has three columns, the third column will be the result of the interaction between column 1 subject and column 2 subject. The result can be in the "W/L/T" format or "1/0/0.5" format. See `get_wl_df`: for further info.

Examples

```
get_wl_matrix(randomtourney(8))
get_wl_matrix(randomtourney(15,pties=.15))
get_wl_matrix(randomtourney(15,pties=.15),ties="keep")
get_wl_matrix(el)
```

`isi13` *Compute best ranked matrixed based on new I&SI method*

Description

Compute best ranked matrixed based on new I&SI method

Usage

```
isi13(m, p = c(1, 0, 0, 0), a_max = 50, nTries = 30, p2 = 0.5,
  random = FALSE)
```

Arguments

m	A win-loss matrix
p	A vector of probabilities for each of 4 methods
a_max	Number of tries
nTries	Number of iterations
p2	probability for last method
random	Whether to randomize initial matrix order

Value

A computed ranked matrix best_matrix best_ranking I and SI, and rs - the Spearman correlation between best order and David's Scores.

Further details

Code based on algorithm described by Schmid & de Vries 2013, Finding a dominance order most consistent with a linear hierarchy: An improved algorithm for the I&SI method, Animal Behaviour 86:1097-1105. This first implementation of this algorithm is not very fast. The code is written in R and is fairly slow. It will be replaced by a function written in C++ soon. The number of tries should be very high and/or the function should be run several times to detect the optimal matrix or matrices. It may take several runs to find the matrix with the lowest SI, especially for very large matrices. For small matrices it may be more efficient to use the older algorithm. See `isi98`: for further info. If the algorithm can no longer improve on reducing the I and SI it will return the order found. For some sparse matrices many orders may have an equal I and SI. The best matrix found here will therefore be dependent upon the initial order of individuals in the matrix. By using `random=TRUE` it is possible to randomize the initial order of individuals in the matrix. This can be helpful in identifying other potentially better fits. For solutions with identical I and SI, better fits have a higher value of rs.

Examples

```
isi13(people,nTries=10)
```

---

isi98 *Compute best ranked matrixed based on original I&SI method*

Description

Compute best ranked matrixed based on original I&SI method

Usage

```
isi98(m, nTries = 100, random = FALSE)
```

Arguments

`m`	A win-loss matrix
`nTries`	Number of tries to find best order
`random`	Whether to randomize initial matrix order

Value

A matrix with best ranking of I and SI plus the correlation (rs) between found ranking and David's Scores

Further details

Code based on algorithm described by de Vries, H. 1998. Finding a dominance order most consistent with a linear hierarchy: a new procedure and review. Animal Behaviour, 55, 827-843. The code is written in R and is fairly slow. It will be replaced by a function written in C++ soon. The number of iterations should be very high and/or the function should be run several times to detect the optimal matrix or matrices. It may take several runs to find a matrix with the lowest SI, especially for very large matrices. The function will stop once it finds a matrix with an I or SI that it can no longer improve upon. The order of this matrix will be dependent upon the input name order of the original matrix. To find further solutions, try using `random==TRUE` to shuffle the name order of the initial matrix. For solutions with identical I and SI, better fits have a higher value of rs. See `isi13`: for further info.

Examples

```
isi98(mouse,nTries=50)
isi98(people, random=TRUE)
```

`mouse` *Mouse sociomatrix*

Description

A win-loss sociomatrix with each row being the number of wins against individual mice in each column.

Usage

```
mouse
```

Format

A data frame with 12 rows and 12 variables:

A The number of losses by mouse A against all other mice

B The number of losses by mouse B against all other mice

C The number of losses by mouse C against all other mice

D The number of losses by mouse D against all other mice

E The number of losses by mouse E against all other mice

F The number of losses by mouse F against all other mice

G The number of losses by mouse G against all other mice

H The number of losses by mouse H against all other mice

I The number of losses by mouse I against all other mice

J The number of losses by mouse J against all other mice

K The number of losses by mouse K against all other mice

L The number of losses by mouse L against all other mice

`org_matrix` *Organize rows and columns of a matrix*

Description

Organize rows and columns of a matrix

Usage

```
org_matrix(m, method = "alpha")
```

Arguments

`m`	A matrix with individuals ordered identically in rows and columns.
`method`	The method to be used to reorganize the matrix. `method="alpha"` is the default and will organize rows/columns based on alphanumeric order of rownames/colnames. `method="wins"` will return a matrix ordered in descending order of summed rows (i.e. total competitive interactions won). If rows have tied number of total wins, they will be returned in the order of the inputted matrix. `method="ds"` Will return a matrix ordered by David's Score.

Value

The same matrix `m` with reordered rows/columns

Examples

```
org_matrix(bonobos)
org_matrix(mouse, method="wins")
org_matrix(people, method="ds")
```

`people` *Human sociomatrix*

Description

A win-loss sociomatrix with each row being the number of wins against individual humans in each column.

Usage

```
people
```

Format

A data frame with 6 rows and 6 variables:

Adam The number of losses by human Adam against all other humans

Bryan The number of losses by human Bryan against all other humans

Chris The number of losses by human Chris against all other humans

Derek The number of losses by human Derek against all other humans

Eddie The number of losses by human Eddie against all other humans

Frank The number of losses by human Frank against all other humans

---

`phi` *Get the phi skew-symmetry of a sociomatrix.*

Description

Get the phi skew-symmetry of a sociomatrix.

Usage

```
phi(m)
```

Arguments

`m`	A matrix with individuals ordered identically in rows and columns.

Value

The phi skew-symmetry index of `m`.

References

Leiva D et al, 2008, Testing reciprocity in social interactions: A comparison between the directional consistency and skew-symmetry statistics, Behav Res Methods.

Further details

Phi is the skew-symmetry index (0 means completely symmetric, 0.5 means completely not symmetric)

Examples

```
m <- matrix(c(NA,2,30,6,19,122,0,NA,18,
0,19,85,0,1,NA,3,8,84,0,0,0,NA,267,50,0,
0,0,5,NA,10,1,0,4,4,1,NA), ncol=6)  #table 2, Vervaecke et al. 2000  - fleeing in bonobos
phi(m)
```

---

randomtourney   *Generates a randomized tournament with random outcomes*

Description

Generates a randomized tournament with random outcomes

Usage

```
randomtourney(n, matchups = 2, pties = 0, ints = 100, type = "char")
```

Arguments

`n`	Number of individuals in tournament
`matchups`	Number of times individuals compete in tournament. Can be a numeric input, or, if `matchups`**="random"** interactions are random
`pties`	Probability of each individual matchup ending in a tie. Default is 0, i.e. no ties. Needs to be a number between 0 and 1.
`ints`	The number of interactions in the tournament if matchups are set to random.
`type`	Whether to return results as W/L characters or 1/2 numbers. `type`**="char"** is the default, `type`**="nums"** returns 1/2 numbers referring to winner as id1 or id2

Value

A competition results dataframe

Further details

Specify number of individuals to compete in a tournament and the number of times they compete with each other. Winners and losers are determined at random. The resulting dataframe will have variables: `id1`, `id2`, `result`. Result refers to the outcome from `id1`'s perspective, i.e. a "W" refers to `id1` beating `id2`, and a "L" refers to `id2` beating `id1`. Individuals are referred to by a random assignment of two conjoined letters.

Examples

```
randomtourney(20,2) #20 individuals interact twice with each other
randomtourney(5,6) #5 individuals interact six times with each other
randomtourney(8) #8 individuals interact twice with each other
```

---

rshps — *Get Relaiontship Descriptives of Sociomatrix*

Description

Get Relaiontship Descriptives of Sociomatrix

Usage

```
rshps(m)
```

Arguments

m	A matrix with individuals ordered identically in rows and columns.

Value

a list of total, unknown, tied, two-way and one-way relationships

Examples

```
rshps(people)
```

---

sparseness — *Calculate the sparseness of relationships in a sociomatrix*

Description

Calculate the sparseness of relationships in a sociomatrix

Usage

```
sparseness(m)
```

Arguments

m	A matrix with individuals ordered identically in rows and columns.

Value

The sparseness of `m`

Further details

The sparseness of a matrix is the proportion of null dyads

Examples

```
sparseness(mouse)
sparseness(caribou)
```

`ttri` *Get Triangle Transitivity of Sociomatrix*

Description

Get Triangle Transitivity of Sociomatrix

Usage

```
ttri(m)
```

Arguments

`m`	A frequency binary win-loss matrix

Value

Pt and t.tri

Further details

Algorithm described in D. Shizuka and D. B. McDonald, 2012, A social network perspective on transitivity and linearity in dominance hierarchies. Animal Behaviour. DOI:10.1016/j.anbehav.2012.01.011 Code adapted from original code by Dai Shikua - see http://www.shizukalab.com/toolkits/sna/triangle-transitivity

Examples

```
ttri(mouse)
```

ttri_test *Signficance testing of Triangle Transitivity of Sociomatrix*

Description

Signficance testing of Triangle Transitivity of Sociomatrix

Usage

```
ttri_test(m, Nperms = 1000)
```

Arguments

m	A frequency or binary win-loss matrix
Nperms	Number of randomizations in signficance test

Value

Pt, triangle transitivity and pvalue

Further details

Algorithm described in D. Shizuka and D. B. McDonald, 2012, A social network perspective on transitivity and linearity in dominance hierarchies. Animal Behaviour. DOI:10.1016/j.anbehav.2012.01.011 Code adapted from original code by Dai Shikua - see http://www.shizukalab.com/toolkits/sna/triangle-transitivity

Examples

```
ttri_test(mouse)
```

unknowns *Calculate the number of unknown relationships in a sociomatrix*

Description

Calculate the number of unknown relationships in a sociomatrix

Usage

```
unknowns(m)
```

Arguments

m	A matrix with individuals ordered identically in rows and columns.

Value

The number of unknown relationships in `m`

Further details

An unknown relationship is defined as one whereby M(i,j)==M(j,i)==0 . The zeros in each cell of dyads that have this property may be referred to as structural zeros.

Examples

```
unknowns(mouse)
```

Index