Package 'aniDom'

April 3, 2019

Type Package

Title Inferring Dominance Hierarchies and Estimating Uncertainty

Version 0.1.4

Date 2019-04-03

Author Damien R. Farine and Alfredo Sanchez-Tojar

Maintainer Damien R. Farine <dfarine@orn.mpg.de>

Description Provides: (1) Tools to infer dominance hierarchies based on calculating Elo scores, but with custom functions to improve estimates in animals with relatively stable dominance ranks. (2) Tools to plot the shape of the dominance hierarchy and estimate the uncertainty of a given data set.

License GPL-2

Imports rptR

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-03 08:10:07 UTC

R topics documented:

aniDom-package *Inferring Dominance Hierarchies and Estimating Uncertainty*

Description

Provides (1) Tools to infer dominance hierarchies based on calculating Elo scores, but with custom functions to improve estimates in animals with relatively stable dominance ranks. (2) Tools to plot the shape of the dominance hierarchy and estimate the uncertainty of a given data set.

Details

Package:	aniDom
Type:	Package
Version:	0.1.4
Date:	2019-04-03
License:	GPL-2

Author(s)

Written by Damien R. Farine and Alfredo Sanchez-Tojar

Maintainer: Damien R. Farine <dfarine@orn.mpg.de>

References

Sanchez-Tojar, A., Schroeder, J., Farine, D.R. (in prep) Methods for inferring dominance hierarchies and estimating their uncertainty.

Examples

```
# Generate data
data <- generate_interactions(N.inds=10,N.obs=20,a=5,b=3)

# Extract interactions
winners <- data$interactions$Winner
losers <- data$interactions$Loser

# Calculate Elo scores with randomised order
scores <- elo_scores(winners=winners,losers=losers,randomise=TRUE,n.rands=1000)

# Plot ranks
plot_ranks(scores,plot.CIs=TRUE)

# Plot hierachy shape
```

```
plot_hierarchy_shape(identity=1:nrow(scores),rank=1:nrow(scores),
winners=winners,losers=losers,fitted=TRUE)
```

---

elo_scores *Calculate Elo scores (with or without time randomisations)*

Description

Function that takes winners and losers from dominance interactions and returns either their Elo score or their ranks. The function can generate replicated datasets by randomising the order (time) of the interactions, which can be used to generate 95% confidence intervals.

Usage

```
elo_scores(winners, losers, identities = NULL, sigmoid.param = 1/100,
K = 200, init.score = 0, randomise = FALSE, n.rands = 1000,
return.as.ranks = FALSE, return.trajectories = FALSE,
dates = NULL)
```

Arguments

winners	Vector containing the identity of the winners. This can be integers or strings.
losers	Vector containing the identity of the losers. This can be integers or strings. These should be in the same order as the winners (i.e. winners[1] should be the winner and losers[1] should be the loser from the same contest).
identities	Optional vector containing the identity of all individuals. This is useful if not all individuals are represented in the winners and losers.
sigmoid.param	A parameter of the Elo function that determines the steepness of the sigmoid function (i.e how much the scores change for small differences in rank). Smaller values flatten the shape (more linear), whereas larger values create a stronger threshold function (more change for small differences in rank).
K	K is a parameter of the Elo function that determines the speed at which scores change after an encounter (default K=200).
init.score	Parameter of the Elo function that determines the starting score (does not have an effect on relative differences in score).
randomise	Boolean (TRUE/FALSE) describing whether to create replicated datasets by randomising the order of the observed interactions (see details).
n.rands	The number of randomisations to perform (ignored if randomise=FALSE).
return.as.ranks	Boolean (TRUE/FALSE) describing whether to convert scores into ranks before returning the data.
return.trajectories	Boolean (TRUE/FALSE) describing whether to return trajectories (the scores after each interaction T) or only the final scores.
dates	Optional vector containing a timestamp identifier (in any format) for each observation (see details).

Details

This function calculates Elo scores using a sigmoidal function. Because animal groups often have relatively stable hierarchies during a study period, this implementation allows the order of interactions to be randomised (randomise=TRUE) to create K replicated datasets (where K=n.rands). This method can improve the estimate of the ranks for each individual, and allow 95% range of values to be computed. If randomise=FALSE, the scores will be calculated only once and maintain the original ordering of winners and losers for each context.

The date option can be used to return timestamped trajectories. Note that because the function assumes that interactions are ordered temporally, the function simply returns each unique timestamp in the order they are encountered. Thus, the aggregation of observations into dates will be determined by the resolution of the timestamp (e.g. minutes, days, months).

Value

The function returns different values depending on the input parameters.

If randomise=FALSE and return.trajectories=FALSE, then the function returns a vector of size N giving final score for each individual from the original ordering of interactions (where N is the number of individuals in the data or in the identities variable if given). If randomise=TRUE and return.trajectories=FALSE, then the function returns a NxK matrix giving the final scores for each individual (rows) after each randomisation of the orders. If randomise=FALSE and return.trajcetories=TRUE, then the function returns an Nx(T+1) matrix giving the starting score (init.score) in the first column followed by the score after each of the T interactions. If randomise=TRUE and return.trajcetories=TRUE, then the function returns an NxTxK array.

Author(s)

Written by Damien R. Farine & Alfredo Sanchez-Tojar

Maintainer: Damien R. Farine <dfarine@orn.mpg.de>

References

Sanchez-Tojar, A., Schroeder, J., Farine, D.R. (in prep) Methods for inferring dominance hierarchies and estimating their uncertainty.

Examples

```
# Generate some input data
data <- generate_interactions(10,20,5,2)

# Extract winners and losers
winners <- data$interactions$Winner
losers <- data$interactions$Loser

# Calculate Elo scores
scores <- elo_scores(winners,losers)

# Plot ranks
plot_ranks(scores)
```

```
#### return timestamped data


# simulating 10 dominance interactions
# in a group of 5 individuals
output <- generate_interactions(5,
10, a=10, b=5, id.biased=FALSE,
rank.biased=FALSE)

# adding some random dates, this could be, in principle,
# of any format, e.g. (character, numeric, etc).
output$interactions$date <- c("date01", "date01",
"date02","date02","date03","date04",
"date04","date05","date06","date06")

dated.trajectories <- elo_scores(
output$interactions$Winner,
output$interactions$Loser,
dates=output$interactions$date,
identities=c(1:5),
randomise=FALSE,
return.as.ranks=FALSE,
return.trajectories=TRUE)
```

`estimate_uncertainty_by_repeatability` *Calculate the repeatability score of ranks from randomisations of the interaction orders.*

Description

Calculates the Intraclass Correlation Coefficient for each individual after randomising the order of interactions in the given dataset.

Usage

```
estimate_uncertainty_by_repeatability(winners, losers, identities = NULL,
sigmoid.param = 1/100, K = 200, init.score = 0, n.rands = 1000)
```

Arguments

`winners`	Vector containing the identity of the winners. This can be integers or strings.
`losers`	Vector containing the identity of the losers. This can be integers or strings. These should be in the same order as the winners (i.e. winners[1] should be the winner and losers[1] should be the loser from the same contest).
`identities`	Optional vector containing the identity of all individuals. This is useful if not all individuals are represented in the winners and losers.

sigmoid.param	A parameter of the Elo function that determines the steepness of the sigmoid function (i.e how much the scores change for small differences in rank). Smaller values flatten the shape (more linear), whereas larger values create a stronger threshold function (more change for small differences in rank).
K	K is a parameter of the Elo function that determines the speed at which scores change after an encounter (default K=200).
init.score	Parameter of the Elo function that determines the starting score (does not have an effect on relative differences in score).
n.rands	The number of randomisations to perform.

Details

Each ordering of winners and losers will yield slightly different Elo scores. This function takes the Elo scores from n.rands randomisations of the order of interactions. It then computes the repeatability score. This repeatability score can provide some insight into the level of certainty (or robustness) of the input data. Our simulations suggest that a repeatability score above 0.8 suggests a reasonably robust hierarchy, given a large input dataset (can be unreliable for small datasets, i.e. < 10 observations per indiviudal), or for extremely flat hierarchies.

Value

Returns an object of class rpt from the rptR library. This object contains all of the information required to estimate repeatability.

Author(s)

Written by Damien R. Farine & Alfredo Sanchez-Tojar

Maintainer: Damien R. Farine <dfarine@orn.mpg.de>

References

Sanchez-Tojar, A., Schroeder, J., Farine, D.R. (in prep) Methods for inferring dominance hierarchies and estimating their uncertainty.

Examples

```
# Generate some input data
data <- generate_interactions(10,20,5,2)

# Extract winners and losers
winners <- data$interactions$Winner
losers <- data$interactions$Loser

# Calculate repeatability
r <- estimate_uncertainty_by_repeatability(winners,losers)
```

`estimate_uncertainty_by_splitting` *Calculate the repeatability score of ranks by splitting the observed data in half.*

Description

Calculates the correlation of the Elo scores for individuals across to exclusive halves of the data.

Usage

```
estimate_uncertainty_by_splitting(winners, losers, identities = NULL,
sigmoid.param = 1/100, K = 200, init.score = 0, randomise = FALSE, n.rands = 1000)
```

Arguments

`winners`	Vector containing the identity of the winners. This can be integers or strings.
`losers`	Vector containing the identity of the losers. This can be integers or strings. These should be in the same order as the winners (i.e. winners[1] should be the winner and losers[1] should be the loser from the same contest).
`identities`	Optional vector containing the identity of all individuals. This is useful if not all individuals are represented in the winners and losers.
`sigmoid.param`	A parameter of the Elo function that determines the steepness of the sigmoid function (i.e how much the scores change for small differences in rank). Smaller values flatten the shape (more linear), whereas larger values create a stronger threshold function (more change for small differences in rank).
`K`	K is a parameter of the Elo function that determines the speed at which scores change after an encounter (default K=200).
`init.score`	Parameter of the Elo function that determines the starting score (does not have an effect on relative differences in score).
`randomise`	Boolean (TRUE/FALSE) describing whether to create replicated datasets by randomising the order of the observed interactions (see details).
`n.rands`	The number of randomisations to perform.

Details

By calculating the correlation of the Elo scores calculated separately for two halves of the data, this function provides some insights into the uncertainty and robustness of the data collected. If randomise=FALSE, the data are split in half and the correlation between the two halves is returned. If randomise=TRUE, then the ordering of the interactions is randomised n.rands times and the correlation is calculated each time in the same way. The function then returns the mean and 95% range of the correlation values. Our simulations suggest that correlations above 0.5 suggests a robust dominance hierarchy (or low uncertainty).

Value

If randomise=FALSE: the Spearman rank correlation coefficient of the first half and second half of the data. If randomise=TRUE: the mean and 95% range of the Spearkman rank correlation coefficients from two halves of the data with the ordering randomised each time.

Author(s)

Written by Damien R. Farine & Alfredo Sanchez-Tojar

Maintainer: Damien R. Farine <dfarine@orn.mpg.de>

References

Sanchez-Tojar, A., Schroeder, J., Farine, D.R. (in prep) Methods for inferring dominance hierarchies and estimating their uncertainty.

Examples

```
# Generate some input data
data <- generate_interactions(10,20,5,2)

# Extract winners and losers
winners <- data$interactions$Winner
losers <- data$interactions$Loser

# Calculate repeatability
r <- estimate_uncertainty_by_splitting(winners,losers,randomise=TRUE)
```

---

generate_interactions *Generate simulated interactions with differently-shaped hierarchies*

Description

Generates simulated winners and losers. The function can generate data for different population sizes, with differently-shaped hierarchies, and with varying biases. The output is the hierarchy, and the winner and loser for each interaction.

Usage

```
generate_interactions(N.inds, N.obs, a, b, id.biased = FALSE, rank.biased = FALSE)
```

Arguments

N.inds	The number of individuals
N.obs	The number of observed interactions (in total).
a	Parameter to control the steepness of the hierarchy (flatter or more sigmoidal) plot_winner_prob.
b	Parameter to control the intercept of the hierachy (moves the sigmoid left or right) plot_winner_prob.
id.biased	Boolean (TRUE/FALSE) describing whether to introduce an individual bias in the observations (some individuals interact more often than others).
rank.biased	Boolean (TRUE/FALSE) describing whether to introduce a rank difference bias in the observations (closely-ranked individuals interact more often).

Details

This function is useful for generating input data with a known hierarchy. The shape of the hierarchy can be controlled using two parameters, though is by default a sigmoidal shape. Higher values of a typically create a greater probability of a dominant winning (turn the function into more of a threshold). Higher values of b typically decrease the probability of a dominant winning when ranks are very similar. The plot_winner_prob function allows visualisation of the hierarchy function (see examples below).

Value

Returns a list with two elements: hierarchy: A dataframe containing three columns, the ID of the individual, its Rank, and its Probability of interacting (varies if id.biased=TRUE). interactions: A dataframe containing two columns, the Winner and the Loser for each interaction. Each row represents one interaction.

Author(s)

Written by Damien R. Farine & Alfredo Sanchez-Tojar

Maintainer: Damien R. Farine <dfarine@orn.mpg.de>

References

Sanchez-Tojar, A., Schroeder, J., Farine, D.R. (in prep) Methods for inferring dominance hierarchies and estimating their uncertainty.

Examples

```
par(mfrow=c(2,2))

# Set population size
N <- 20

# Set shape parameters
a = 15
```

```
b = 3

# See what this looks like
plot_winner_prob(1:N,a,b)

# Generate some input data
data <- generate_interactions(N,400,a,b)

# See what the hierarchy looks like from the output data
winners <- data$interactions$Winner
losers <- data$interactions$Loser
identities <- data$hierarchy$ID
ranks <- data$hierarchy$Rank
plot_hierarchy_shape(identities,ranks,winners,losers,fitted=TRUE)

# Set new shape parameters
a = 3
b = 3

# See what this looks like
plot_winner_prob(1:N,a,b)

# Generate some input data
data <- generate_interactions(N,400,a,b)

# See what the hierarchy looks like from the output data
winners <- data$interactions$Winner
losers <- data$interactions$Loser
identities <- data$hierarchy$ID
ranks <- data$hierarchy$Rank
plot_hierarchy_shape(identities,ranks,winners,losers,fitted=TRUE)
```

`plot_hierarchy_shape` *Plots the shape of a dominance hierarchy from empirical data*

Description

This function takes a set of winners and losers from observed interactions and plots the probability of the dominant individual in an interaction winning given the difference in rank to the subordinate in the same interaction.

Usage

```
plot_hierarchy_shape(identity, rank, winners, losers, fitted = FALSE)
```

Arguments

`identity`	A vector containing the identities of all individuals in the data.
`rank`	A vector giving the ranks for each individual (in the same order as the identities).

winners	A vector giving the identity of the winner for each interaction.
losers	A vector giving the identity of the loser for each interaction in the same order as the winners.
fitted	A Boolean (TRUE/FALSE) describing whether to add a fitted line to the plot

Details

This function is useful for examining how the probability of winning is shaped by the difference in rank. The shape of this graph provides information about the shape of the dominance hierarchy.

Value

This function will return the data for x (difference in rank) and y (probability of dominant winning) coordinates of the plot as a data frame.

Author(s)

Written by Damien R. Farine & Alfredo Sanchez-Tojar

Maintainer: Damien R. Farine <dfarine@orn.mpg.de>

References

Sanchez-Tojar, A., Schroeder, J., Farine, D.R. (in prep) Methods for inferring dominance hierarchies and estimating their uncertainty.

Examples

```
par(mfrow=c(1,2))

# Set population size
N <- 20

# Set shape parameters
a = 15
b = 3

# See what this looks like
plot_winner_prob(1:N,a,b)

# Generate some input data
data <- generate_interactions(N,400,a,b)

# See what the hierarchy looks like from the output data
winners <- data$interactions$Winner
losers <- data$interactions$Loser
identities <- data$hierarchy$ID
ranks <- data$hierarchy$Rank
shape <- plot_hierarchy_shape(identities,ranks,winners,losers,fitted=TRUE)

# Data is contained in shape
```

shape

---

plot_ranks *Plot the ranking of individuals*

Description

Function to plot the ranking of individuals in different ways.

Usage

```
plot_ranks(ranks, plot.CIs = FALSE, ordered.by.rank = TRUE,
identities = NULL, plot.identities = TRUE, colors = NULL)
```

Arguments

ranks	Either a vector containing the score or rank of each individual, or an NxK matrix containing the results of K randomisations of the data.
plot.CIs	Boolean (TRUE/FALSE): if providing an NxK matrix, then setting plot.CIs to TRUE will plot the 95% range of the scores or ranks given for each individual.
ordered.by.rank	Boolean (TRUE/FALSE) describing whether to order individuals by rank or not.
identities	A vector containing the identity (name) of each individual to be plotted along the X axis.
plot.identities	Boolean (TRUE/FALSE) describing whether to plot the identity of each individual along the X axis.
colors	A vector containing the colour for each individual (default="black"). This is useful for example to colour individuals by sex.

Details

A simple function that plots individuals' ranks, with options to colour individuals or order them. Here the y axis is reverse, so that rank=1 occurs at the top.

Value

Generates a plot. No data is returned.

Author(s)

Written by Damien R. Farine & Alfredo Sanchez-Tojar

Maintainer: Damien R. Farine <dfarine@orn.mpg.de>

References

Sanchez-Tojar, A., Schroeder, J., Farine, D.R. (in prep) Methods for inferring dominance hierarchies and estimating their uncertainty.

Examples

```
# Set population size
N <- 10

# Set shape parameters
a = 15
b = 3

# Generate data
data <- generate_interactions(N,100,a,b)

# Extract data (and turn IDs into letters for this example)
winners <- letters[data$interactions$Winner]
losers <- letters[data$interactions$Loser]
identities <- letters[data$hierarchy$ID]

# Calculate Elo scores
scores <- elo_scores(winners,losers,identities=identities,randomise=TRUE)

# Plot results
plot_ranks(scores, plot.CIs=TRUE,identities=TRUE,colors=rainbow(N))
```

---

plot_trajectories *Plot the Elo trajectories of individuals*

Description

Plot the trajectories of Elo scores after each interaction

Usage

```
plot_trajectories(trajectories, colors = NULL)
```

Arguments

trajectories	The output of elo_scores with the return.trajectories option set to TRUE: an Nx(T+1) matrix (where T is the number of interactions).
colors	An optional vector of colours for each line (default = "black").

Details

Plots one set of trajectories. If the randomise option in elo_scores is set to TRUE, then the resulting matrices should be passed one at a time.

Value

Generates a plot. No data is returned.

Author(s)

Written by Damien R. Farine & Alfredo Sanchez-Tojar

Maintainer: Damien R. Farine <dfarine@orn.mpg.de>

References

Sanchez-Tojar, A., Schroeder, J., Farine, D.R. (in prep) Methods for inferring dominance hierarchies and estimating their uncertainty.

Examples

```
# Set population size
N <- 10

# Set shape parameters
a = 15
b = 3

# Generate data
data <- generate_interactions(N,100,a,b)

# Extract data (and turn IDs into letters for this example)
winners <- letters[data$interactions$Winner]
losers <- letters[data$interactions$Loser]
identities <- letters[data$hierarchy$ID]

# Calculate Elo scores
scores <- elo_scores(winners,losers,identities=identities,
randomise=FALSE,return.trajectories=TRUE)

# Plot results
plot_trajectories(scores, colors=rainbow(N))
```

plot_winner_prob *Plot the shape of the hierarchy given some input parameters*

Description

A simple function that provides visualisations of the shape of the hierarchy given parameters a and b in the generate_interactions function

Usage

```
plot_winner_prob(diff.rank, a, b)
```

Arguments

diff.rank	A vector containing the x values of the plot (i.e. differences in rank).
a	Parameter a (see generate_interactions).
b	Parameter b (see generate_interactions).

Details

A simple plotting function to visualise the shapes of curves in the generate_interactions function.

Value

Generates a plot. No data is returned.

Author(s)

Written by Damien R. Farine & Alfredo Sanchez-Tojar

Maintainer: Damien R. Farine <dfarine@orn.mpg.de>

References

Sanchez-Tojar, A., Schroeder, J., Farine, D.R. (in prep) Methods for inferring dominance hierarchies and estimating their uncertainty.

Examples

```
# Set population size
N <- 10

# Set shape parameters
a = 15
b = 3

# Plot the shape
```

```
plot_winner_prob(1:10,a,b)
```

Index