Package 'asnipe'

July 22, 2025

Type Package

Title Animal Social Network Inference and Permutations for Ecologists

Version 1.1.17

Date 2023-09-15

Author Damien R. Farine <dfarine@ab.mpg.de>

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

Description Implements several tools that are used in animal social network analysis, as described in Whitehead (2007) Analyzing Animal Societies <University of Chicago Press> and Farine & Whitehead (2015) <doi:10.1111/1365-2656.12418>. In particular, this package provides the tools to infer groups and generate networks from observation data, perform permutation tests on the data, calculate lagged association rates, and performed multiple regression analysis on social network data.

License GPL-2

Encoding UTF-8

Depends R (>= 2.10)

Imports MASS, Matrix

Suggests ape, raster, sna

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-15 06:30:02 UTC

Contents

`asnipe-package` *Animal Social Network inference and Permutation: asnipe*

Description

Provides functions for infering associations, building social networks, performing permutations, and regression testing

Details

Package:	asnipe
Type:	Package
Version:	1.1.17
Date:	2023-09-15
License:	GPL-2

Author(s)

Written by Damien R. Farine

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

gbi *Detections of Individuals Forming Flocks at Bird Feeders*

Description

Dataset consisting of 151 individuals of 5 passerine species in Wytham Woods, UK: 78 blue tits (Cyanistes caeruleus), 7 coal tits (Periparus ater), 51 great tits (Parus major), 11 marsh tits (Poecile palustris), 3 nuthatches (Sitta europaea) and 1 individual of unknown species. Individuals were all fitted with individually-encoded passive integrated transponder (PIT) tags that were logged by radio frequency identification (RFID) antennae fitted to each hole on regular sunflower feeders (we used unhusked sunflower seed). Data were collected from 4 feeders spaced approximately 300m over the course of one day. Feeders logged the presence of individuals at a sub-second resolution, and detections were assigned to flocks using a machine learning algorithm (a Gaussian Mixture Model).

Usage

```
data("group_by_individual")
```

Format

Data are formatted in a group by individual matrix. Each row represents one flock, each column represents one individual.

Source

Farine, D.R., Garroway, C.J., Sheldon, B.C. (2012) Social Network Analysis of mixed-species flocks: exploring the structure and evolution of interspecific social behaviour. Animal Behaviour 84: 1271-1277.

Examples

```
data("group_by_individual")
str(gbi) # see the structure of the data
```

get_associations_points_tw *Calculate Group Membership using Time Window (please read warnings before using this method)*

Description

A time window approach to calculate group co-memberships.

Usage

```
get_associations_points_tw(point_data, time_window = 180, which_days = NULL,
which_locations = NULL)
```

Arguments

point_data	dataframe of four columns: Date Time ID Location. This requirement is strict (see details).
time_window	window around each individual for calculating associations
which_days	subset of Date to include
which_locations	subset of Locations to include

Details

Calculates an ego-centric approach to group membership (see warning). For each detection, a group is created with and all associates within the time window at the same location are included.

Input data must be of the following form: Date is an integer for day (usually starting at 1 on the first day). Time are the number of seconds elapsed from the start (continuous across all dates). ID is a unique character string for each individual. Location is a unique character string for each location.

Value

Returns a list with three objects: 1. group by individual matrix (K rows by N columns) 2. an vector of times for each group 3. a vector of dates for each group 4. a vector of locations for each group

Warning

This method biases associations of dyads occuring in large groups because it creates one row in the association matrix for each detection of an individual. For this reason, this function should not be used (see also Psorakis et al. 2015 Behavioural Ecology & Sociobiology). One way to circumvent this is by including only groups centered around the focal individual when calculating associations. However, none of the functions in this package are implement this way.

Author(s)

Damien R. Farine

Examples

```
data("identified_individuals")

## calculate group_by_individual for first day at one location
group_by_individual <- get_associations_points_tw(identified_individuals, time_window=180,
which_days=1,which_locations="1B")

## split the resulting list
times <- group_by_individual[[2]]
dates <- group_by_individual[[3]]
locations <- group_by_individual[[4]]
group_by_individual <- group_by_individual[[1]]
```

get_group_by_individual *Convert group or individual data into a group by individual matrix*

Description

Converts several different types of data storage into a group by individual matrix for calculating or permuting networks

Usage

```
get_group_by_individual(association_data, identities = NULL,
location = NULL, data_format = c("groups", "individuals"))
```

Arguments

association_data	Can be either a group by individual matrix or a list containing group members in each element
identities	Optional identities for each individual in the dataset
location	Returns these spatial locations for each group
data_format	Format of the input data

Details

This function will calculate an K x N matrix representing K groups and N individuals. If locations are included, these will be returned in the row names.

Value

Returns a K x N matrix, where each K row is an group defined from the input data. Column names of the matrix are given the identity where available. The K row names are given either the time or time_location for each group.

Author(s)

Damien R. Farine

Examples

```
## define group memberships (these would be read from a file)
individuals <- data.frame(ID=c("C695905","H300253","H300253",
"H300283","H839876","F464557","H300296","H300253",
"F464557","H300296","C695905","H300283","H839876"),
GROUP=c(1,1,2,2,2,3,3,4,5,5,6,6,6))

## create a time column
individuals <- cbind(individuals,
```

```
DAY=c(1,1,1,1,1,2,2,2,3,3,3,3,3))

gbi <- get_group_by_individual(individuals,
data_format="individuals")

## define group memberships (these would be read from a file)
groups <- list(G1=c("C695905","H300253"),
G2=c("H300253","H300283","H839876"),
G3=c("F464557","H300296"),
G4=c("H300253"),
G5=c("F464557","H300296"),
G6=c("C695905","H300283","H839876"))

## create a time variable
days <- c(1,1,2,2,3,3)

gbi <- get_group_by_individual(groups,
data_format="groups")
```

`get_network` *Calculating Weighted Network*

Description

Calculate a network from a group by individual matrix. This function allows various levels of subsetting.

Usage

```
get_network(association_data, data_format = "GBI",
association_index = "SRI", identities = NULL,
which_identities = NULL, times = NULL, occurrences = NULL,
locations = NULL, which_locations = NULL, start_time = NULL,
end_time = NULL, classes = NULL, which_classes = NULL,
enter_time = NULL, exit_time = NULL)
```

Arguments

`association_data`
: a K x N matrix of K groups (observations, gathering events, etc.) and N individuals (all individuals that are present in at least one group) OR a K x N x N array of sampling periods.

`data_format`
: `"GBI"` expect a group by individual matrix, `"SP"` Expect a sampling periods array

`association_index`
: `"SRI"` Simple ratio index, `"HWI"` Half-weight index (more to come)

identities	N vector of identifiers for each individual (column) in the group by individual matrix
which_identities	vector of identities to include in the network (subset of identities)
times	K vector of times defining the middle of each group/event
occurrences	N x S matrix with the occurrence of each individual in each sampling period (see details) containing only 0s and 1s
locations	K vector of locations defining the location of each group/event
which_locations	vector of locations to include in the network (subset of locations)
start_time	element describing the starting time for inclusion in the network (useful for temporal analysis)
end_time	element describing the ending time for inclusion in the network (useful for temporal analysis)
classes	N vector of types or class of each individual (column) in the group by individual matrix (for subsetting)
which_classes	vector of class(es)/type(s) to include in the network (subset of classes)
enter_time	N vector of times when each individual entered the population
exit_time	N vector of times when each individual departed the population

Details

Provides the ability to generate networks from one group by individual matrix and subsetting within the function. This is particularly useful for generating several networks with different characteristics from the same group by individual matrix (for example networks from a given location or set of locations, or of a particular sex).

Including occurrence data is recommended when using sampling periods (not required for GBI data). If an individual is only observed alone in a sampling period, then it will not be included in the sampling period matrices (as these record only associations or interactions, not presence). Thus, a matrix containing N (for number of individuals) rows and S (for number of sampling periods) is required. See the get_sampling_periods function for help generating this matrix.

In some situations it is useful to calculate the network based only on the period in which each dyad overlapped within the population. In such cases, the entry_time and/or the exit_time variables can be given. These must given in the same format as the times variable, and all need to be in a format capable of doing time or date comparisons using > and < operators. The easiest is YYYYMMDD, whereas MMDDYYYY or DDMMYYYY will not work properly.

Value

N x N matrix of association weights for each dyad.

Author(s)

Damien R. Farine

References

Whitehead (2008) *Analyzing Animal Societies*

Examples

```
data("group_by_individual")
data("times")

# subset GBI (to reduce run time of the example)
gbi <- gbi[,1:80]

## define to 2 x N x N network to hold two association matrices
networks <- array(0, c(2, ncol(gbi), ncol(gbi)))

## calculate network for first half of the time
networks[1,,] <- get_network(gbi, data_format="GBI",
association_index="SRI", times=times, start_time=0,
end_time=max(times)/2)
networks[2,,] <- get_network(gbi, data_format="GBI",
association_index="SRI", times=times,
start_time=max(times)/2, end_time=max(times))

## test if one predicts the other via a mantel test (must be loaded externally)
library(ape)
mantel.test(networks[1,,],networks[2,,])

## convert to igraph network and calculate degree of the first network
## Not run:
library(igraph)
net <- graph.adjacency(networks[1,,], mode="undirected", diag=FALSE, weighted=TRUE)
deg_weighted <- graph.strength(net)
detach(package:igraph)


## alternatively package SNA can use matrix stacks directly
library(sna)
deg_weighted <- degree(networks,gmode="graph", g=c(1,2), ignore.eval=FALSE)
detach(package:sna)

## End(Not run)
```

---

get_sampling_periods *Convert group or individual data into sampling periods*

Description

Converts several different types of data storage into sampling periods for calculating or permuting networks

Usage

```
get_sampling_periods(association_data, association_times, sampling_period,
identities = NULL, location = NULL, within_locations = FALSE,
data_format = c("gbi", "groups", "individuals"), return="SP")
```

Arguments

association_data	Can be either a group by individual matrix, a list containing group members in each element, or a two-column data frame with individual ID in the first column and group ID in the second column
association_times	Because sampling periods are inferred over time, each group must contain some time data (can be in any format, such as seconds, days, etc.). One time must be provided for each row of the association data.
sampling_period	The number of time periods over which data are combined (for example 10 days, 3600 seconds)
identities	Optional identities for each individual in the dataset
location	If spatial disaggregation need to be maintained, samping periods can be calculated per time per location
within_locations	Flag whether to include location information
data_format	Format of the input data
return	By default ("SP") returns the sampling periods. Anything else will return the occurrence data (see get_network function)

Details

This function will calculate an association matrix for each sampling period. If locations are included, these will be treated independently.

Value

Returns a K x N x N stack of matrices, where each N x N slice is an association matrix. Row names and Column names of these slices are given the identity where available. The K slice names are given either the time or time_location for each sampling period. Alternatively (return != "SP") the function returns the occurrence of each individual in each sampling period, with individuals as rows and sampling periods as columns.

Author(s)

Damien R. Farine

Examples

```
## define group memberships (these would be read from a file)
individuals <- data.frame(ID=c("C695905","H300253","H300253",
"H300283","H839876","F464557","H300296","H300253",
"F464557","H300296","C695905","H300283","H839876"),
GROUP=c(1,1,2,2,2,3,3,4,5,5,6,6,6))

## create a time column
individuals <- cbind(individuals,
DAY=c(1,1,1,1,1,2,2,2,3,3,3,3,3))

SPs <- get_sampling_periods(individuals[,c(1,2)],
individuals[,3],1,data_format="individuals")
occurs <- get_sampling_periods(individuals[,c(1,2)],
individuals[,3],1,data_format="individuals", return="occ")

## define group memberships (these would be read from a file)
groups <- list(G1=c("C695905","H300253"),
G2=c("H300253","H300283","H839876"),
G3=c("F464557","H300296"),
G4=c("H300253"),
G5=c("F464557","H300296"),
G6=c("C695905","H300283","H839876"))

## create a time variable
days <- c(1,1,2,2,3,3)

SPs <- get_sampling_periods(groups,
days,1,data_format="groups")
occurs <- get_sampling_periods(groups,
days,1,data_format="groups", return="occ")
```

`gmmevents` *Infer gathering events*

Description

Infer gathering events (groups or flocks) from a temporal datastream of observations, such as PIT tag data.

Usage

```
gmmevents(time, identity, location, global_ids=NULL, verbose=TRUE, splitGroups=TRUE)
```

Arguments

`time`	The timestamp for the observation. Must be a real number (i.e. not a date or time format). See details below.

identity	The identify of the individual in each observation (can be a number or a string, e.g. PIT tag code).
location	The location of the observation (can be a number or a string).
global_ids	A vector of all the IDs in the study, used if consistency needs to be maintained across datasets.
verbose	Whether to print out progress and information.
splitGroups	Whether or not to split overlappling groups (see details).

Details

The gmmevents function has 3 primary inputs: time, identity, and location:

The time must be a number representing a real valued time stamp. This can be number of seconds since the start of the day, number of seconds since the start of the study, hour of the day, Julian date, etc. The time stamps should represent a meaningful scale given the group membership definition - for example if an edge is the propensity to observe an individual at the same location on the same day, then time stamps should be the day value. In the example below, the time stamps are in seconds, because flocks of birds visit feeders over a matter of minutes and the group definition is being in the same flock (and these occur over seconds to minutes). The input must be numeric whole numbers.

The identity is the unique identifier for each individual. This should be consistent across all of the data sets. In the example here, PIT tags are given, but in broader analyses, we would convert these to ring (band) numbers because individuals can have different PIT tag numbers in the course of the study but never change ring numbers. The function will accept any string or numeric input.

The location is where the observation took place. This should reflect meaningful observation locations for the study. The function will accept any string or numeric inputs.

If the analysis is being conducted as part of a broader analysis in the same populations, it can be useful to get the results in a consistent form each time. In that case, the global_ids variable can be used to maintain consistency each time an analysis is run, regardless of which individuals were identified in the current input data. That is, the group by individual (gbi) matrix will include a column for every individual provided in global_ids.

Further notes on usage:

The gmm_events functions requires a few careful considerations. First, the amount of memory used is the square of the amount of data - so having many observations in a given location can run out of memory. With 16gb of RAM, generally up to 10,000 observations per location (per day - see next point) seems to be a safe limit.

The input data provided for each location should take into account any artificial gaps in the observation stream. For example, if there are gaps in data collection at a given location, then the location information provided into the gmm_events function should be split into two 'locations' to represent each continuous set of observations. For example, in the PIT tag data set provided there are 8 days of sampling. Providing gmm_events with only the location data from the original data will cause the gap between days to override any gaps between groups (or flocks) within a given day. To overcome this, instead of providing the gmm_events function with just the location, it is important to provide a location by day variable. This variable is then returned in the metadata and the information extracted out again (using strsplit - see example below).

Finally, I have included a new variable called splitGroups. The original function (in both Matlab and R) would return the occasional group that overlapped other groups. This occured when a

small group was extracted from the data, and then the remaining observations were formed into a larger group that spanned the smaller group from the same location. For example, say detections of individuals are made in the same location at 2,8,10,11,12,14,20 seconds, and the first group extracted contains 10,11,12 then the remaining data look like an evenly-spread group (2,8,14,20). Setting splitGroups=TRUE identifies such incidences and would split the data into three groups (2,8), (10,11,12), and (14,20).

Value

Returns a list containing three items:

The first item is the group by individual matrix (gbi), which is a matrix where each row is a gathering event - or group - and each column is an individual. Cells in the matrix have a value of 1 if the individual was observed in that gathering event, and 0 if not.

The second item is a matrix containing three columns: the start time, end time, and location of each detected event. The number of rows in this events matrix matches the number of rows in the gbi file, and the rows correspond to one another (thus, row 3 of the gbi has the start and end times, and location, of row 3 of the events matrix).

The third item has the same structure as the group by individual matrix, but instead of being binary (0 or 1), it contains the number of observations of each individual in each event.

Author(s)

Ioannis Psorakis (original code)
Julian Evans (R implementation)
Damien R. Farine (current implementation)

References

Psorakis, I., Roberts, S. J., Rezek, I., & Sheldon, B. C. (2012). Inferring social network structure in ecological systems from spatio-temporal data streams. Journal of the Royal Society Interface, 9(76), 3055-3066. doi:10.1098/Rsif.2012.0223
Psorakis, I., Voelkl, B., Garroway, C. J., Radersma, R., Aplin, L. M., Crates, R. A., Culina, A., Farine, D. R., Firth, J.A., Hinde, C.A., Kidd, L.R., Milligan, N.D., Roberts, S.J., Verhelst, B., Sheldon, B. C. (2015). Inferring social structure from temporal data. Behavioral Ecology and Sociobiology, 69(5), 857-866. doi:10.1007/s00265-015-1906-0

Examples

```
library(asnipe)
data("identified_individuals")

# Create unique locations in time
identified_individuals$Loc_date <-
paste(identified_individuals$Location,
identified_individuals$Date,sep="_")

# Provide global identity list (including individuals
```

```
# not found in these data, but that need to be included).
# Not including this will generate gbi with only the
# individuals provided in the data set (in this case 151
# individuals)
global_ids <- levels(identified_individuals$ID)

# Generate GMM data
gmm_data <- gmmevents(time=identified_individuals$Time,
identity=identified_individuals$ID,
location=identified_individuals$Loc_date,
global_ids=global_ids)

# Extract output
gbi <- gmm_data$gbi
events <- gmm_data$metadata
observations_per_event <- gmm_data$B

# Can also subset gbi to only individuals observed
# in the dataset to give same answer as if
# global_ids had not been provided
gbi <- gbi[,which(colSums(gbi)>0)]

# Split up location and date data
tmp <- strsplit(events$Location,"_")
tmp <- do.call("rbind",tmp)
events$Location <- tmp[,1]
events$Date <- tmp[,2]
```

`identified_individuals` *Raw Observation Data of Individual Birds Feeding at Flocks*

Description

Contains the raw observation data, of which the first day was used to form the group by individual file. IDs correspond to TAG in the "individuals" data (note that some tags are error codes, which have not been removed, and thus do not occur in the individuals data).

Usage

```
data("identified_individuals")
```

Format

Data frame containing 4 columns: Date - The observation day (1 to 8, where days 1-2 are the first weekend, 3-4 the second weekend, etc..) Time - The time in seconds since the very first observation ID - The PIT tag code of the individual Location - The location where the detection was made (1B, 1C, 1D, 1E)

Source

Farine, D.R., Garroway, C.J., Sheldon, B.C. (2012) Social Network Analysis of mixed-species flocks: exploring the structure and evolution of interspecific social behaviour. Animal Behaviour 84: 1271-1277.

Examples

```
data("identified_individuals")
head(identified_individuals)
table(identified_individuals$Location)
```

---

inds *Data on the Individual Birds Contained in the Group by Individual data*

Description

Information about the PIT tag number, ring number, species, and sex (where available) for each individual in the group by individual data. Each row represents one column in the group by individual file, and the order is maintained.

Usage

```
data("individuals")
```

Format

Data frame containing: TAG - A 10 character hexadecimal code unique to each individual RING.NUMBER - A 7 character unique ring (or band) number for each individual SPECIES - Each species, where BLUTI=blue tit, COATI=coal tit, GRETI=great tit, MARTI=marsh tit, and NUTHA=nuthatch

Source

Farine, D.R., Garroway, C.J., Sheldon, B.C. (2012) Social Network Analysis of mixed-species flocks: exploring the structure and evolution of interspecific social behaviour. Animal Behaviour 84: 1271-1277.

Examples

```
data("individuals")
data("group_by_individual")
colnames(gbi) <- inds$RING.NUMBER
```

LAR *Mean Lagged Association Rate*

Description

Calculate lagged association rate g(tau) from Whitehead (2008)

Usage

```
LAR(group_by_individual, times, timejump, min_time = NULL, max_time = NULL,
identities = NULL, which_identities = NULL, locations = NULL,
which_locations = NULL, start_time = NULL, end_time = NULL, classes = NULL,
which_classes = NULL)
```

Arguments

group_by_individual	a K x N matrix of K groups (observations, gathering events, etc.) and N individuals (all individuals that are present in at least one group)
times	K vector of times defining the middle of each group/event
timejump	step length for tau
min_time	minimum/starting value of tau
max_time	maximum/ending value of tau
identities	N vector of identifiers for each individual (column) in the group by individual matrix
which_identities	vector of identities to include in the network (subset of identities)
locations	K vector of locations defining the location of each group/event
which_locations	vector of locations to include in the network (subset of locations)
start_time	element describing the starting time for inclusion in the network (useful for temporal analysis)
end_time	element describing the ending time for inclusion in the network (useful for temporal analysis)
classes	N vector of types or class of each individual (column) in the group by individual matrix (for subsetting)
which_classes	vector of class(es)/type(s) to include in the network (subset of classes)

Details

Calculate the lagged association rate for given timesteps.

Value

Returns a matrix with Log(time) in the first column and the lagged association rate in the second

Author(s)

Damien R. Farine

References

Whitehead (2008) *Analyzing Animal Societies* section 5.5.1

Examples

```
data("group_by_individual")
data("times")
data("individuals")

## calculate lagged association rate for great tits
lagged_rates <- LAR(gbi,times,3600, classes=inds$SPECIES, which_classes="GRETI")

## plot the results
plot(lagged_rates, type='l', axes=FALSE, xlab="Time (hours)", ylab="LAR", ylim=c(0,1))
axis(2)
axis(1, at=lagged_rates[,1], labels=c(1:nrow(lagged_rates)))
```

LRA *Dyadic Lagged Association Rate*

Description

Calculate lagged association rate g(tau) from Whitehead (2008) for each dyad individually

Usage

```
LRA(group_by_individual, times, timejump, output_style = 1, min_time = NULL,
max_time = NULL, identities = NULL, which_identities = NULL, locations = NULL,
which_locations = NULL, start_time = NULL, end_time = NULL, classes = NULL,
which_classes = NULL, association_rate = TRUE)
```

Arguments

group_by_individual	a K x N matrix of K groups (observations, gathering events, etc.) and N individuals (all individuals that are present in at least one group)
times	K vector of times defining the middle of each group/event
timejump	step length for tau
output_style	either 1 or 2, see details
min_time	minimum/starting value of tau
max_time	maximum/ending value of tau

identities	N vector of identifiers for each individual (column) in the group by individual matrix
which_identities	vector of identities to include in the network (subset of identities)
locations	K vector of locations defining the location of each group/event
which_locations	vector of locations to include in the network (subset of locations)
start_time	element describing the starting time for inclusion in the network (useful for temporal analysis)
end_time	element describing the ending time for inclusion in the network (useful for temporal analysis)
classes	N vector of types or class of each individual (column) in the group by individual matrix (for subsetting)
which_classes	vector of class(es)/type(s) to include in the network (subset of classes)
association_rate	calculate lagged rate of association (see details)

Details

Calculates the dyadic lagged association rate. The lagged rate of association incorporates the number of observations of each individuals as a simple ratio index within each time period, leading to a better estimation of the assocation rate for data where many observations of individuals can be made within a single time period.

Value

If `output_style == 1` then a stack of matrices is returned that is `N x N x tau`. If `output_style == 2` then a dataframe is returned containing the focal ID, associate, `tau`, and lagged association rate.

Author(s)

Damien R. Farine

References

Expanded from Whitehead (2008)

Examples

```
data("group_by_individual")
data("times")
data("individuals")

## calculate lagged association rate
lagged_rates <- LRA(gbi,times,3600, classes=inds$SPECIES, which_classes="GRETI", output_style=2)

## do something (run a model, plot a surface, etc..)
```

`mrqap.custom.null` *MRQAP function with custom permutation networks*

Description

Calculate MRQAP with random networks provides (i.e. generated by a custom model of user's choice)

Usage

```
mrqap.custom.null(formula, random.y, intercept = TRUE, directed = "undirected",
diagonal = FALSE, test.statistic = "t-value",
tol = 1e-07)
```

Arguments

`formula`	input formula (e.g. y ~ x1 + x2), where y and each x are NxN matrices
`random.y`	a k x N x N matrix containing a set of random networks generated by some permutation method
`intercept`	calculate intercept (TRUE or FALSE value)
`directed`	whether the network is directed or undirected (enter either "directed" or "undirected")
`diagonal`	whether to include self-loop values (TRUE or FALSE)
`test.statistic`	what to calculate P-value, either t-statistic ("t-value") or regression coefficient ("beta")
`tol`	tolerance value for the qr function

Details

Calculate the regression coefficient for each input matrix using MRQAP but where the random networks are provided. This is in contrast to mrqap.dsp which has a built-in node permutation (which I have shown has higher rates of type II errors - see Farine & Whitehead 2015 and Farine in prep.). This method can easily be interfaced with the network_permutation method. Note however that this method tests whether y is related to x1 and x2 together because the different fixed effects are not permuted independently (as suggested by Dekker et al 2007). Whilst the potential to avoid type II errors may warrant this approach, further theoretical testing is needed to confirm this approach is appropriate.

Value

Returns a mrqap.dsp object containing the regression coefficient and P-values for each indendent matrix (x) and associated statistics

Author(s)

Damien R. Farine

References

Dekker, D., Krackhard, D., Snijders, T.A.B (2007) Sensitivity of MRQAP tests to collinearity and autocorellation conditions. *Psychometrika* 72(4): 563-581. Farine, D. R., & Whitehead, H. (2015) Constructing, conducting, and interpreting animal social network analysis. Journal of Animal Ecology, 84(5), 1144-1163. Farine, D. R. (in prep) Why and how to use null models in animal social network analysis.

Examples

```
library(asnipe)
data("individuals")
data("group_by_individual")

# Generate network
network <- get_network(gbi)

# Create a species similarity matrix
species <- array(0,dim(network))

# Create a sex similarity matrix
sex <- array(0,dim(network))

# Fill each matrix with 1 (same) or 0 (different)
for (i in 1:nrow(network)) {
species[i,-i] <- as.numeric(inds$SPECIES[i] == inds$SPECIES[-i])
sex[i,-i] <- as.numeric(inds$SEX[i] == inds$SEX[-i])
}

# Perform network randomisation
# Note randomisations are limited to 10 to reduce runtime
networks_rand <- network_permutation(gbi, association_matrix=network, permutations=10)

# Run mrqap.custom.null
# Note randomisations are limited to 10 to reduce runtime
reg <- mrqap.custom.null(network ~ species + sex, random.y=networks_rand)

# Look at results
reg
```

`mrqap.dsp` *MRQAP with Double-Semi-Partialing (DSP)*

Description

Calculate MRQAP with Double-Semi-Partialing (DSP) from Dekker et al (2007)

Usage

```
mrqap.dsp(formula, intercept = TRUE, directed = "undirected",
diagonal = FALSE, test.statistic = "t-value",
tol = 1e-07, randomisations = 1000)
```

Arguments

formula	input formula (e.g. y ~ x1 + x2), where y and each x are NxN matrices
intercept	calculate intercept (TRUE or FALSE value)
directed	whether the network is directed or undirected (enter either "directed" or "undirected")
diagonal	whether to include self-loop values (TRUE or FALSE)
test.statistic	what to calculate P-value, either t-statistic ("t-value") or regression coefficient ("beta")
tol	tolerance value for the qr function
randomisations	number of randomisations to perform for calculating P-value.

Details

Calculate the regression coefficient for each input matrix using the DSP method in Dekker et al (2007). This method randomises the residuals from the regression on each independent variable (fixed effect) in order to calculate the P value. This is the same as testing whether y is related to x1 on y while controlling for x2. This differs from regular mrqap, where the dependent (y) value is randomised, testing for whether y is related to x1 and x2 together.

Value

Returns a mrqap.dsp object containing the regression coefficient and P-values for each indendent matrix (x) and associated statistics

Author(s)

Damien R. Farine

References

Dekker, D., Krackhard, D., Snijders, T.A.B (2007) Sensitivity of MRQAP tests to collinearity and autocorellation conditions. *Psychometrika* 72(4): 563-581.

Examples

```
library(asnipe)
data("individuals")
data("group_by_individual")

# Generate network
network <- get_network(gbi)
```

```
# Create a species similarity matrix
species <- array(0,dim(network))

# Create a sex similarity matrix
sex <- array(0,dim(network))

# Fill each matrix with 1 (same) or 0 (different)
for (i in 1:nrow(network)) {
species[i,-i] <- as.numeric(inds$SPECIES[i] == inds$SPECIES[-i])
sex[i,-i] <- as.numeric(inds$SEX[i] == inds$SEX[-i])
}

# Run mrqap.dsp
# Note randomisations are limited to 10 to reduce runtime
reg <- mrqap.dsp(network ~ species + sex, randomisations=10)

# Look at results
reg
```

`network_permutation` *Perform Permutation*

Description

Performs permutations on the data and calculates network for each step

Usage

```
network_permutation(association_data, data_format = "GBI", permutations = 1000,
returns=1, association_index = "SRI", association_matrix = NULL,
identities = NULL, which_identities = NULL, times = NULL, occurrences = NULL,
locations = NULL,  which_locations = NULL, start_time = NULL,
end_time = NULL, classes = NULL, which_classes = NULL,
days = NULL, within_day = FALSE, within_location = FALSE, within_class = FALSE,
enter_time = NULL, exit_time = NULL, symmetric=TRUE, trialSwap=TRUE)
```

Arguments

`association_data`	a K x N matrix of K groups (observations, gathering events, etc.) and N individuals (all individuals that are present in at least one group) OR a K x N x N array of sampling periods.
`data_format`	`"GBI"` expect a group by individual matrix, `"SP"` Expect a sampling periods array
`permutations`	number of permutations (default = 1000)
`returns`	number of swaps to perform between each association matrix that is returned (default = 1)

`association_index`
: `"SRI"` Simple ratio index, `"HWI"` Half-weight index (more to come)

`association_matrix`
: provide a starting association matrix (see details)

`identities`
: N vector of identifiers for each individual (column) in the group by individual matrix

`which_identities`
: vector of identities to include in the network (subset of identities)

`times`
: K vector of times defining the middle of each group/event

`occurrences`
: N x S matrix with the occurrence of each individual in each sampling period (see details) containing only 0s and 1s

`locations`
: K vector of locations defining the location of each group/event

`which_locations`
: vector of locations to include in the network (subset of locations)

`start_time`
: element describing the starting time for inclusion in the network (useful for temporal analysis)

`end_time`
: element describing the ending time for inclusion in the network (useful for temporal analysis)

`classes`
: N vector of types or class of each individual (column) in the group by individual matrix (for subsetting)

`which_classes`
: vector of class(es)/type(s) to include in the network (subset of classes)

`days`
: K vector of day stamp for each event (can be integer or string representing any period of time)

`within_day`
: if `TRUE` then permutations will be done within the time periods

`within_location`
: if `TRUE` then permutations will be done within the given locations

`within_class`
: if `TRUE` then permutations will be done within the given classes

`enter_time`
: N vector of times when each individual entered the population

`exit_time`
: N vector of times when each individual departed the population

`symmetric`
: Boolean to ensure that permutations maintain symmetry within sampling periods if using `data_format="SP"`

`trialSwap`
: Boolean to include trial swaps (if true, then every attempted permutation is returned)

Details

Performs permutations on the group by individual matrix as given by Whitehead (2008). In order to save computing, only the recently swapped individuals are recalculated, hence why the association matrix of the original data can be provided or is recalculated.

This implementation allows permutations (swaps) to be restricted to within any of three classes. Though each class is labelled, the function is flexible. Hence, days can represent any time period (months, hours, etc.).

Swaps are implemented in a hybrid between the trial swaps proposed by Miklos and Podani (2004) and full swaps (a swap every permutation). Every permutation, a candidate edge is selected (as opposed to a dyad which could or could not have an edge, as proposed by Miklos and Podani). Then a second possible dyad is selected, from all dyads. If the selected portions of the data satisfy the baseline rules (e.g. the checkerboard pattern), then either the selection is attempted again `trialSwap = FALSE` or not `trialSwap = TRUE`. This should be set to TRUE, but the option for FALSE is provided for legacy analyses (full swap).

See get_network function for additional details on each field.

Value

Returns a p x N x N stack of matrices with the dyadic association rates of each pair of individuals after each swap or after a number of swaps, where p = ceiling(permutations/returns)

Author(s)

Damien R. Farine

References

Whitehead (2008) *Analyzing Animal Societies*

Examples

```
### USING TIMES, ETC.

data("group_by_individual")
data("times")

# subset GBI (to reduce run time of the example)
gbi <- gbi[,1:80]

## define to 2 x N x N network to hold two association matrices
networks <- array(0, c(2, ncol(gbi), ncol(gbi)))

## calculate network for first half of the time
networks[1,,] <- get_network(gbi, data_format="GBI",
association_index="SRI", times=times, start_time=0,
end_time=max(times)/2)
networks[2,,] <- get_network(gbi, data_format="GBI",
association_index="SRI", times=times,
start_time=max(times)/2, end_time=max(times))

## calculate the weighted degree
library(sna)
deg_weighted <- degree(networks,gmode="graph", g=c(1,2), ignore.eval=FALSE)

## perform the permutations constricting within hour of observation
## note permutations are limited to 10 to reduce runtime
network1_perm <- network_permutation(gbi, data_format="GBI",
association_matrix=networks[1,,], times=times, start_time=0,
```

```
end_time=max(times)/2, days=floor(times/3600), within_day=TRUE,
permutations=10)
network2_perm <- network_permutation(gbi, data_format="GBI",
association_matrix=networks[2,,], times=times,
start_time=max(times)/2, end_time=max(times), days=floor(times/3600), within_day=TRUE,
permutations=10)

## calculate the weighted degree for each permutation
deg_weighted_perm1 <- degree(network1_perm,gmode="graph", g=c(1:10), ignore.eval=FALSE)
deg_weighted_perm2 <- degree(network2_perm,gmode="graph", g=c(1:10), ignore.eval=FALSE)
detach(package:sna)

## plot the distribution of permutations with the original data overlaid
par(mfrow=c(1,2))
hist(colMeans(deg_weighted_perm1),breaks=100,
main=paste("P = ",
sum(mean(deg_weighted[,1]) < colMeans(deg_weighted_perm1))/ncol(deg_weighted_perm1)),
xlab="Weighted degree", ylab="Probability")
abline(v=mean(deg_weighted[,1]), col='red')
hist(colMeans(deg_weighted_perm2),breaks=100,
main=paste("P = ",
sum(mean(deg_weighted[,2]) < colMeans(deg_weighted_perm2))/ncol(deg_weighted_perm2)),
xlab="Weighted degree", ylab="Probability")
abline(v=mean(deg_weighted[,2]), col='red')



#### DOUBLE PERMUTATION EXAMPLE (see Farine & Carter 2021)

## Load data
data("group_by_individual")
data("times")

# subset GBI (to reduce run time of the example)
gbi <- gbi[,1:40]

# Specify metric
metric <- "DEGREE"

# calculate observed network
network <- get_network(gbi, data_format="GBI",
association_index="SRI", times=times)

# Calculate observed metric (degree)
degrees <- rowSums(network)

# Do randomisation (as above, permutations should be >=1000)
networks.perm <- network_permutation(gbi, data_format="GBI",
association_matrix=network, times=times, permutations=10)

# Now calculate the same metric on all the random networks
degrees.rand <- apply(networks.perm,1,function(x) { rowSums(x)})
```

```
# Now substract each individual's median from the observed
degree.controlled <- degrees - apply(degrees.rand,1,median)

#### Now use degree.controlled for any later test. For example, to related against a trait:

# Make a trait
trait <- rnorm(length(degree.controlled))

# get the coefficient of this:
coef <- summary(lm(degree.controlled~trait))$coefficients[2,3]

# Compare this to a node permutation
# (here just randomising the trait values)
# note this should be done >= 1000 times
n.node.perm <- 10
coefs.random <- rep(NA, n.node.perm)

for (i in 1:n.node.perm) {
trait.random <- sample(trait)
coefs.random[i] <- summary(lm(degree.controlled~trait.random))$coefficients[2,3]
}

# calculate P value (note this is only one sided)
P <- sum(coef <= coefs.random)/n.node.perm
```

---

network_swap *Perform one (or more) random swap*

Description

Performs one (or more) random swap on the data and re-calculates network, returning both the new network and the data stream

Usage

```
network_swap(association_data, data_format = "GBI", swaps = 1,
association_index = "SRI", association_matrix = NULL,
identities = NULL, which_identities = NULL, times = NULL,
occurrences = NULL, locations = NULL, which_locations = NULL,
start_time = NULL, end_time = NULL, classes = NULL,
which_classes = NULL, days = NULL, within_day = FALSE,
within_location = FALSE, within_class = FALSE, symmetric=TRUE,
trialSwap=TRUE)
```

Arguments

association_data	a K x N matrix of K groups (observations, gathering events, etc.) and N individuals (all individuals that are present in at least one group) OR a K x N x N array of sampling periods.
data_format	"GBI" expect a group by individual matrix, "SP" Expect a sampling periods array
swaps	number of swaps (default = 1000)
association_index	"SRI" Simple ratio index, "HWI" Half-weight index (more to come)
association_matrix	provide a starting association matrix (see details)
identities	N vector of identifiers for each individual (column) in the group by individual matrix
which_identities	vector of identities to include in the network (subset of identities)
times	K vector of times defining the middle of each group/event
occurrences	N x S matrix with the occurrence of each individual in each sampling period (see details) containing only 0s and 1s
locations	K vector of locations defining the location of each group/event
which_locations	vector of locations to include in the network (subset of locations)
start_time	element describing the starting time for inclusion in the network (useful for temporal analysis)
end_time	element describing the ending time for inclusion in the network (useful for temporal analysis)
classes	N vector of types or class of each individual (column) in the group by individual matrix (for subsetting)
which_classes	vector of class(es)/type(s) to include in the network (subset of classes)
days	K vector of day stamp for each event (can be integer or string representing any period of time)
within_day	if TRUE then permutations will be done within the time periods
within_location	if TRUE then permutations will be done within the given locations
within_class	if TRUE then permutations will be done within the given classes
symmetric	Boolean to ensure that permutations maintain symmetry within sampling periods if using data_format="SP"
trialSwap	Boolean to include trial swaps (if TRUE, then every attempted permutation is returned)

Details

Performs one or more permutation swaps on the group by individual matrix as given by Whitehead (2008). In order to save on memory use, this function computers the number of swaps and returns the association matrix and the data stream resulting from these, thus not needing to create a large stack of networks to store each permutation. This can then be implemented in a loop as shown in the example below. Note that this method is quite a bit slower than the network_permutation function.

This implementation allows permutations (swaps) to be restricted to within any of three classes. Though each class is labelled, the function is flexible. Hence, days can represent any time period (months, hours, etc.). However, unlike the network_permutation, the subsetting of the data must be done outside of this function (for reasons that might be obvious) - see the example below.

Trial swaps are implemented following Miklos and Podani (2004). Every permutation, a candidate swap is selected. If the selected portions of the data satisfy the baseline rules (e.g. the checkerboard pattern), then either the selection is attempted again `trialSwap = FALSE` or not `trialSwap = TRUE`. This should be set to TRUE, but the option for FALSE is provided for legacy analyses.

Value

Returns a list containing an `N x N` matrix with the dyadic association rates of each pair of individuals after performing the swaps, and the `N x N` data stream post-swap, as two list elements.

Author(s)

Damien R. Farine

References

Whitehead (2008) *Analyzing Animal Societies*

Examples

```
# load data
data("group_by_individual")
data("times")

# subset GBI (to reduce run time of the example)
gbi <- gbi[,1:80]

# calculate network for data based on morning associations
network <- get_network(gbi, association_index="SRI",
times=times, start_time=0, end_time=max(times)/2)

# perform 100 permutations and calculate the coefficient
# of variance after each permutation.
# note that the subsetting is done outside of the function
library(raster)
cvs <- rep(NA,100)
network_perm = list(network,gbi[which(times <= max(times)/2),])
hours <- floor(times/3600)[which(times <= max(times)/2)]
for (i in 1:100) {
network_perm <- network_swap(network_perm[[2]], swaps=1,
```

```
association_matrix=network_perm[[1]], days=hours,
within_day=TRUE)
cvs[i] <- cv(network_perm[[1]])
}

# plot the results with the original network as a red dot
plot(cvs,pch=20,cex=0.5)
points(0,cv(network),cex=1,pch=20,col="red")
```

`print.mrqap.dsp` *Print function for mrqap.dsp*

Description

Print function for MRQAP DSP function

Usage

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

Arguments

`x`	an mrqap.dsp object (from function in asnipe package)
`...`	Further arguments passed to or from print methods

Details

Print formatted results.

Value

Prints formatted results

Author(s)

Damien R. Farine

References

see mrqap.dsp function

`times` *Observation Time for each Flock Contained in the Group by Individual data*

Description

The start time for each flock in the group by individual data since the first flock observed.

Usage

```
data("times")
```

Format

An integer array with the time (in seconds since the first flock of the day).

Source

Farine, D.R., Garroway, C.J., Sheldon, B.C. (2012) Social Network Analysis of mixed-species flocks: exploring the structure and evolution of interspecific social behaviour. Animal Behaviour 84: 1271-1277.

Examples

```
data("times")
data("group_by_individual")
rownames(gbi) <- times
```

Index