Title  Epistemic Network Analysis

Type  Package

Author  Cody L Marquart [aut, cre] (<https://orcid.org/0000-0002-3387-6792>),
        Zachari Swiecki [aut],
        Wesley Collier [aut],
        Brendan Eagan [aut],
        Roman Woodward [aut],
        David Williamson Shaffer [aut]

Maintainer  Cody L Marquart <cody.marquart@wisc.edu>

Version  0.2.3

Description  ENA (Shaffer, D. W. (2017) Quantitative Ethnography. ISBN: 0578191687) is a method used to identify meaningful and quantifiable patterns in discourse or reasoning. ENA moves beyond the traditional frequency-based assessments by examining the structure of the co-occurrence, or connections in coded data. Moreover, compared to other methodological approaches, ENA has the novelty of (1) modeling whole networks of connections and (2) affording both quantitative and qualitative comparisons between different network models. Shaffer, D.W., Collier, W., & Ruis, A.R. (2016).

LazyData  TRUE

Depends  R (>= 3.0.0)

License  GPL-3 | file LICENSE

LinkingTo  Rcpp, RcppArmadillo

Imports  data.table, Rcpp, R6, methods, foreach, stats, plotly,
         doParallel, parallel, scales, magrittr, concatenate

Suggests  testthat (>= 2.1.0), knitr, rmarkdown, webshot

RoxygenNote  7.1.1

NeedsCompilation  yes

Repository  CRAN

Date/Publication  2021-04-23 15:10:06 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_group</td>
<td>3</td>
</tr>
<tr>
<td>add_network</td>
<td>4</td>
</tr>
<tr>
<td>add_nodes</td>
<td>4</td>
</tr>
<tr>
<td>add_points</td>
<td>5</td>
</tr>
<tr>
<td>add_trajectory</td>
<td>5</td>
</tr>
<tr>
<td>as.ena.co.occurrence</td>
<td>6</td>
</tr>
<tr>
<td>as.ena.matrix</td>
<td>6</td>
</tr>
<tr>
<td>as.ena.metadata</td>
<td>7</td>
</tr>
<tr>
<td>as.matrix.ena.connections</td>
<td>7</td>
</tr>
<tr>
<td>as.matrix.ena.line.weights</td>
<td>8</td>
</tr>
<tr>
<td>as.matrix.ena.matrix</td>
<td>8</td>
</tr>
<tr>
<td>as.matrix.ena.nodes</td>
<td>9</td>
</tr>
<tr>
<td>as.matrix.ena.points</td>
<td>9</td>
</tr>
<tr>
<td>as.matrix.ena.rotation.matrix</td>
<td>10</td>
</tr>
<tr>
<td>as.matrix.row.connections</td>
<td>10</td>
</tr>
<tr>
<td>as_trajectory</td>
<td>11</td>
</tr>
<tr>
<td>clear</td>
<td>11</td>
</tr>
<tr>
<td>connection.matrix</td>
<td>12</td>
</tr>
<tr>
<td>ena</td>
<td>12</td>
</tr>
<tr>
<td>ena.accumulate.data</td>
<td>14</td>
</tr>
<tr>
<td>ena.conversations</td>
<td>16</td>
</tr>
<tr>
<td>ena.correlations</td>
<td>18</td>
</tr>
<tr>
<td>ena.group</td>
<td>18</td>
</tr>
<tr>
<td>ena.make.set</td>
<td>19</td>
</tr>
<tr>
<td>ena.plot</td>
<td>21</td>
</tr>
<tr>
<td>ena.plot.group</td>
<td>23</td>
</tr>
<tr>
<td>ena.plot.network</td>
<td>25</td>
</tr>
<tr>
<td>ena.plot.points</td>
<td>28</td>
</tr>
<tr>
<td>ena.plot.trajectory</td>
<td>30</td>
</tr>
<tr>
<td>ena.plotter</td>
<td>32</td>
</tr>
<tr>
<td>ena.rotate.by.mean</td>
<td>34</td>
</tr>
<tr>
<td>ena.set.creator</td>
<td>34</td>
</tr>
<tr>
<td>ena.svd</td>
<td>36</td>
</tr>
<tr>
<td>ena.writeup</td>
<td>36</td>
</tr>
<tr>
<td>ENAdata</td>
<td>37</td>
</tr>
<tr>
<td>ENAplot</td>
<td>39</td>
</tr>
<tr>
<td>ENARotationSet</td>
<td>41</td>
</tr>
<tr>
<td>ENAset</td>
<td>42</td>
</tr>
<tr>
<td>ena_correlation</td>
<td>44</td>
</tr>
<tr>
<td>find_code_cols</td>
<td>44</td>
</tr>
<tr>
<td>find_dimension_cols</td>
<td>45</td>
</tr>
<tr>
<td>find_meta_cols</td>
<td>45</td>
</tr>
<tr>
<td>fun_cohens.d</td>
<td>46</td>
</tr>
<tr>
<td>fun_skip_sphere_norm</td>
<td>46</td>
</tr>
<tr>
<td>fun_sphere_norm</td>
<td>47</td>
</tr>
<tr>
<td>means_rotate</td>
<td>47</td>
</tr>
</tbody>
</table>
**add_group**

Add a group mean to an ena.plot

### Description

Add a group mean to an ena.plot

### Usage

```r
add_group(x, wh = NULL, ...)
```

### Arguments

- `x` ena.plot object to plot on
- `wh` which points to plot as the trajectory
- `...` additional parameters to pass along

### Value

ena.plot.object
add_network

Add a network to an ENA plot

Description
Add a network to an ENA plot

Usage
add_network(x, wh = NULL, ..., with.mean = F)

Arguments
x
ena.plot object to plot with
wh
network to plot
...
Additional parameters to pass along
with.mean
Logical value, if TRUE plots the mean for the points in the network

Value
ena.plot.object

add_nodes

Title

Description
Title

Usage
add_nodes(x, ...)

Arguments
x
[TBD]
...
[TBD]

Value
TBD
add_points

Plot points on an ena.plot

Description
Plot points on an ena.plot

Usage
add_points(x, wh = NULL, ..., name = "plot", mean = NULL, colors = NULL)

Arguments
x ena.plot to add point on
wh which points to plot
... additional parameters to pass along
name name to give the plot
mean include a mean point for the provided points
colors colors for plotted points

Value
ena.plot.object

add_trajectory

Plot a trajectory on an ena.plot

Description
Plot a trajectory on an ena.plot

Usage
add_trajectory(x, wh = NULL, ..., name = "plot")

Arguments
x ena.plot object to plot on
wh which points to plot as the trajectory
... additional parameters to pass along
name Name, as a character vector, to give the plot

Value
ena.plot.object
### as.ena.co.occurrence

**Description**

Re-class vector as ena.co.occurrence

**Usage**

```r
as.ena.co.occurrence(x)
```

**Arguments**

- `x` Vector to re-class

**Value**

re-classed vector

---

### as.ena.matrix

**Description**

Re-class matrix as ena.matrix

**Usage**

```r
as.ena.matrix(x, new.class = NULL)
```

**Arguments**

- `x` data.frame, data.table, or matrix to extend
- `new.class` Additional class to extend the matrix with, default: NULL

**Value**

Object of same st
### as.ena.metadata

**Re-class matrix as ena.metadata**

**Description**

Re-class matrix as ena.metadata

**Usage**

```r
as.ena.metadata(x)
```

**Arguments**

- `x` : data.frame, data.table, or matrix to extend

**Value**

Object of same st

### as.matrix.ena.connections

**ENA Connections as a matrix**

**Description**

ENA Connections as a matrix

**Usage**

```r
## S3 method for class 'ena.connections'
as.matrix(x, ...)
```

**Arguments**

- `x` : ena.connections object
- `...` : additional arguments to be passed to or from methods

**Value**

If square is FALSE (default), a matrix with all metadata columns removed, otherwise a list with square matrices
as.matrix.ena.line.weights

ENA line weights as matrix

Description
ENA line weights as matrix

Usage
## S3 method for class 'ena.line.weights'
as.matrix(x, ..., square = FALSE)

Arguments
x ena.line.weights data.table to covert to matrix
... additional arguments to be passed to or from methods
square [TBD]

Value
matrix

as.matrix.ena.matrix Matrix without metadata

Description
Matrix without metadata

Usage
## S3 method for class 'ena.matrix'
as.matrix(x, ...)

Arguments
x Object to convert to a matrix
... additional arguments to be passed to or from methods

Value
matrix
as.matrix.ena.nodes

### Description
ENA nodes as matrix

### Usage
```r
## S3 method for class 'ena.nodes'
as.matrix(x, ...)
```

### Arguments
- `x`: ena.nodes to convert to matrix
- `...`: additional arguments to be passed to or from methods

### Value
matrix

---

as.matrix.ena.nodes ENA nodes as matrix

---

as.matrix.ena.points ENA points as matrix

---

### Description
ENA points as matrix

### Usage
```r
## S3 method for class 'ena.points'
as.matrix(x, ...)
```

### Arguments
- `x`: ena.points to convert to a matrix
- `...`: additional arguments to be passed to or from methods

### Value
matrix
as.matrix.eno.rotation.matrix

ENA rotations as matrix

Description
ENA rotations as matrix

Usage
## S3 method for class 'eno.rotation.matrix'
as.matrix(x, ...)

Arguments
x eno.rotation.matrix to convert to matrix
...
additional arguments to be passed to or from methods

Value
matrix

as.matrix.row.connections
ENA row connections as matrix

Description
ENA row connections as matrix

Usage
## S3 method for class 'row.connections'
as.matrix(x, ...)

Arguments
x ena.row.connections to convert to a matrix
...
additional arguments to be passed to or from methods

Value
matrix
as_trajectory

Description
Title

Usage
as_trajectory(
x,
by = x$'_function.params'$_conversation[1],
model = c("AccumulatedTrajectory", "SeperateTrajectory"),
...
)

Arguments
x [TBD]
by [TBD]
model [TBD]
... [TBD]

Value
TBD

clear

Description
Title

Usage
clear(x, wh = seq(x$plots))

Arguments
x [TBD]
wh [TBD]

Value
TBD
connection.matrix

Description

Connection counts as square matrix

Usage

connection.matrix(x)

Arguments

x

ena.set or ena.connections (i.e. set$connection.counts)

Value

matrix

ena

Wrapper to generate, and optionally plot, an ENA model

Description

Generates an ENA model by constructing a dimensional reduction of adjacency (co-occurrence) vectors as defined by the supplied conversations, units, and codes.

Usage

ena(
  data,
  codes,
  units,
  conversation,
  metadata = NULL,
  model = c("EndPoint", "AccumulatedTrajectory", "SeparateTrajectory"),
  weight.by = "binary",
  window = c("MovingStanzaWindow", "Conversation"),
  window.size.back = 1,
  include.meta = TRUE,
  groupVar = NULL,
  groups = NULL,
  runTest = FALSE,
  points = FALSE,
  mean = FALSE,
  network = TRUE,
networkMultiplier = 1,
subtractionMultiplier = 1,
unit = NULL,
include.plots = T,
print.plots = F,
...)

Arguments

data     data.frame with containing metadata and coded columns
codes    vector, numeric or character, of columns with codes
units    vector, numeric or character, of columns representing units
conversation    vector, numeric or character, of columns to segment conversations by
metadata    vector, numeric or character, of columns with additional meta information for units
model    character: EndPoint (default), AccumulatedTrajectory, SeparateTrajectory
weight.by  "binary" is default, can supply a function to call (e.g. sum)
window    MovingStanzaWindow (default) or Conversation
window.size.back    Number of lines in the stanza window (default: 1)
include.meta    [TBD]
groupVar    vector, character, of column name containing group identifiers. If column contains at least two unique values, will generate model using a means rotation (a dimensional reduction maximizing the variance between the means of the two groups)
groups    vector, character, of values of groupVar column used for means rotation, plotting, or statistical tests
runTest    logical, TRUE will run a Student’s t-Test and a Wilcoxon test for groups defined by the groups argument
points    logical, TRUE will plot points (default: FALSE)
mean    logical, TRUE will plot the mean position of the groups defined in the groups argument (default: FALSE)
network    logical, TRUE will plot networks (default: TRUE)
networkMultiplier    numeric, scaling factor for non-subtracted networks (default: 1)
subtractionMultiplier    numeric, scaling factor for subtracted networks (default: 1)
unit    vector, character, name of a single unit to plot
include.plots    logical, TRUE will generate plots based on the model (default: TRUE)
print.plots    logical, TRUE will show plots in the Viewer (default: FALSE)
...    Additional parameters passed to set creation and plotting functions
Details

This function generates an ena.set object given a data.frame, units, conversations, and codes. After accumulating the adjacency (co-occurrence) vectors, computes a dimensional reduction (projection), and calculates node positions in the projected ENA space. Returns location of the units in the projected space, as well as locations for node positions, and normalized adjacency (co-occurrence) vectors to construct network graphs. Includes options for returning statistical tests between groups of units, as well as plots of units, groups, and networks.

Value

ena.set object

Examples

data(RS.data)

rs = ena(
  data = RS.data,
  units = c("UserName","Condition","GroupName"),
  conversation = c("Condition","GroupName"),
  codes = c("Data",
    'Technical.Constraints',
    'Performance.Parameters',
    'Client.and.Consultant.Requests',
    'Design.Reasoning',
    'Collaboration'),
  window.size.back = 4,
  print.plots = FALSE,
  groupVar = "Condition",
  groups = c("FirstGame","SecondGame")
)

Description

This function initializes an ENAdata object, processing conversations from coded data to generate adjacency (co-occurrence) vectors

Usage

ena.accumulate.data(
  units = NULL,
  conversation = NULL,
  codes = NULL,
metadata = NULL,
model = c("EndPoint", "AccumulatedTrajectory", "SeparateTrajectory"),
weight.by = "binary",
window = c("MovingStanzaWindow", "Conversation"),
window.size.back = 1,
window.size.forward = 0,
mask = NULL,
include.meta = T,
as.list = T,
...)

Arguments

units A data frame where the columns are the properties by which units will be identified
conversation A data frame where the columns are the properties by which conversations will be identified
codes A data frame where the columns are the codes used to create adjacency (co-occurrence) vectors
metadata (optional) A data frame with additional columns of metadata to be associated with each unit in the data
model A character, choices: EndPoint (or E), AccumulatedTrajectory (or A), or SeparateTrajectory (or S); default: EndPoint. Determines the ENA model to be constructed
weight.by (optional) A function to apply to values after accumulation
window A character, choices are Conversation (or C), MovingStanzaWindow (MSW, MS); default MovingStanzaWindow. Determines how stanzas are constructed, which defines how co-occurrences are modeled
window.size.back A positive integer, Inf, or character (INF or Infinite), default: 1. Determines, for each line in the data frame, the number of previous lines in a conversation to include in the stanza window, which defines how co-occurrences are modeled
window.size.forward (optional) A positive integer, Inf, or character (INF or Infinite), default: 0. Determines, for each line in the data frame, the number of subsequent lines in a conversation to include in the stanza window, which defines how co-occurrences are modeled
mask (optional) A binary matrix of size ncol(codes) x ncol(codes). 0s in the mask matrix row i column j indicates that co-occurrence will not be modeled between code i and code j
include.meta Logical indicating if unit metadata should be attached to the resulting ENAdata object, default is TRUE
as.list R6 objects will be deprecated, but if this is TRUE, the original R6 object will be returned, otherwise a list with class 'ena.set'
... additional parameters addressed in inner function
Details

ENAData objects are created using this function. This accumulation receives separate data frames for units, codes, conversation, and optionally, metadata. It iterates through the data to create an adjacency (co-occurrence) vector corresponding to each unit - or in a trajectory model multiple adjacency (co-occurrence) vectors for each unit.

In the default MovingStanzaWindow model, co-occurrences between codes are calculated for each line k in the data between line k and the window.size.back-1 previous lines and window.size.forward-1 subsequent lines in the same conversation as line k.

In the Conversation model, co-occurrences between codes are calculated across all lines in each conversation. Adjacency (co-occurrence) vectors are constructed for each unit u by summing the co-occurrences for the lines that correspond to u.

Options for how the data is accumulated are endpoint, which produces one adjacency (co-occurrence) vector for each until summing the co-occurrences for all lines, and two trajectory models: AccumulatedTrajectory and SeparateTrajectory. Trajectory models produce an adjacency (co-occurrence) model for each conversation for each unit. In a SeparateTrajectory model, each conversation is modeled as a separate network. In an AccumulatedTrajectory model, the adjacency (co-occurrence) vector for the current conversation includes the co-occurrences from all previous conversations in the data.

Value

ENAData object with data [adjacency (co-occurrence) vectors] accumulated from the provided data frames.

See Also

ENAData, ena.make.set

Description

Find rows of conversations by unit

Usage

ena.conversations(
  set, units, units.by = NULL, codes = NULL, conversation.by = NULL, window = 4, conversation.exclude = c())
### Arguments

- **set**: [TBD]
- **units**: [TBD]
- **units.by**: [TBD]
- **codes**: [TBD]
- **conversation.by**: [TBD]
- **window**: [TBD]
- **conversation.exclude**: [TBD]

### Details

[TBD]

### Value

List containing row indices representing conversations

### Examples

```r
data(RS.data)


accum = ena.accumulate.data(
  units = RS.data[,c("Condition","UserName")],
  conversation = RS.data[,c("Condition","GroupName")],
  metadata = RS.data[,c("CONFIDENCE.Change","CONFIDENCE.Pre", "CONFIDENCE.Post","C.Change")],
  codes = RS.data[,codeNames],
  model = "EndPoint",
  window.size.back = 4
);

set = ena.make.set(
  enadata = accum,
  rotation.by = ena.rotate.by.mean,
  rotation.params = list(accum$meta.data$Condition=="FirstGame", accum$meta.data$Condition=="SecondGame")
);

ena.conversations(set = RS.data,
  units = c("FirstGame.steven z"), units.by=c("Condition","UserName"),
  conversation.by = c("Condition","GroupName"),
codes=codeNames, window = 4)
```
### ena.correlations

**Calculate the correlations**

**Description**

Calculate both Spearman and Pearson correlations for the provided ENAset

**Usage**

```r
ena.correlations(enaset, dims = c(1:2))
```

**Arguments**

- `enaset`: ENAset to run correlations on
- `dims`: The dimensions to calculate the correlations for. Default: c(1,2)

**Value**

Matrix of 2 columns, one for each correlation method, with the corresponding correlations per dimension as the rows.

### ena.group

**Compute summary statistic for groupings of units using given method (typically, mean)**

**Description**

Computes summary statistics for groupings (given as vector) of units in ena data using given method (typically, mean); computes summary statistic for point locations and edge weights for each grouping

**Usage**

```r
ena.group(
  enaset = NULL,
  by = NULL,
  method = mean,
  names = as.vector(unique(by))
)
```
Arguments

enaset  
A vector or ENAset of values to group.

by  
A vector of values the same length as units. Uses rotated points for group positions and normed data to get the group edge weights.

method  
A function that is used on grouped points. Default: mean(). If 'enaset' is an ENAset, enaset$points.rotated will be grouped using 'mean' regardless of 'method' provided.

names  
A vector of names to use for the results. Default: unique(by)

Value

A list containing names, points, and edge weights for each of the unique groups formed by the function.

Examples

data(RS.data)


accum = ena.accumulate.data(
  units = RS.data[,c("UserName","Condition")],
  conversation = RS.data[,c("Condition","GroupName")],
  metadata = RS.data[,c("CONFIDENCE.Change","CONFIDENCE.Pre","CONFIDENCE.Post")],
  codes = RS.data[,codeNames],
  window.size.back = 4
)

set = ena.make.set(
  enadata = accum
)

means = ena.group(set,"Condition")

---

**ena.make.set**  
*Generate ENA Set*

**Description**

Generates an ENA model by constructing a dimensional reduction of adjacency (co-occurrence) vectors in an ENA data object.
**Usage**

```r
ena.make.set(
  enadata,  
  dimensions = 2, 
  norm.by = fun_sphere_norm, 
  rotation.by = ena.svd, 
  rotation.params = NULL, 
  rotation.set = NULL, 
  endpoints.only = TRUE, 
  center.align.to.origin = FALSE, 
  node.position.method = lws.positions.sq, 
  as.list = TRUE,
  ...
)
```

**Arguments**

- **enadata**: ENAdata that will be used to generate an ENA model
- **dimensions**: The number of dimensions to include in the dimensional reduction
- **norm.by**: A function to be used to normalize adjacency (co-occurrence) vectors before computing the dimensional reduction, default: `sphere_norm_c()`
- **rotation.by**: A function to be used to compute the dimensional reduction, default: `ena.svd()`
- **rotation.params**: (optional) A character vector containing additional parameters for the function in `rotation.by`, if needed
- **rotation.set**: A previously-constructed ENARotationSet object to use for the dimensional reduction
- **endpoints.only**: A logical variable which determines whether to only show endpoints for trajectory models
- **center.align.to.origin**: A logical variable when TRUE (default) determines aligns both point center and centroid center to the origin
- **node.position.method**: A function to be used to determine node positions based on the dimensional reduction, default: `lws.position.es()`
- **as.list**: R6 objects will be deprecated, but if this is TRUE, the original R6 object will be returned, otherwise a list with class 'ena.set'
- **...**: additional parameters addressed in inner function

**Details**

This function generates an ENAset object from an ENAdata object. Takes the adjacency (co-occurrence) vectors from enadata, computes a dimensional reduction (projection), and calculates node positions in the projected ENA space. Returns location of the units in the projected space, as well as locations for node positions, and normalized adjacency (co-occurrence) vectors to construct network graphs.
ena.plot

Generate a plot of an ENA set

Description

Generates an a plot from a given ENA set object

Usage

ena.plot(
  enaset,
  title = "ENA Plot",
  dimension.labels = c("", "")
)
font.size = 10,  
font.color = "#000000",  
font.family = c("Arial", "Courier New", "Times New Roman"),  
scale.to = "network",  
...  
)

Arguments

enaset The ENAset that will be used to generate a plot

title A character used for the title of the plot, default: ENA Plot
dimension.labels A character vector containing labels for the axes, default: c(X, Y)
font.size An integer determining the font size for graph labels, default: 10
font.color A character determining the color of label font, default: black
font.family A character determining the font type, choices: Arial, Courier New, Times New Roman, default: Arial
scale.to "network" (default), "points", or a list with x and y ranges. Network and points both scale to the c(-max, max) of the corresponding data.frame
...
additional parameters addressed in inner function

Details

This function defines the axes and other features of a plot for displaying an ENAset; generates an ENAplot object that can used to plot points, network graphs, and other information from an ENAset

Value

ENAplot used for plotting an ENAset

See Also

ena.make.set, ena.plot.points

Examples

data(RS.data)


accum = ena.accumulate.data(  
units = RS.data[,c("UserName", "Condition")],  
conversation = RS.data[,c("Condition", "GroupName")],  
metadata = RS.data[,c("CONFIDENCE. Change", "CONFIDENCE. Pre", "CONFIDENCE. Post")],  
codes = RS.data[,codeNames],  
window.size.back = 4  
)
set = ena.make.set(
    enadata = accum
)

plot = ena.plot(set)

group1.points = set$points.rotated[set$enadata$units$Condition == "FirstGame",]
plot = ena.plot.points(plot, points = group1.points);
print(plot);

ena.plot.group  Plot of ENA set groups

Description
Plot a point based on a summary statistic computed from a given method (typically, mean) for a set of points in a projected ENA space

Usage
ena.plot.group(
    enaplot,
    points = NULL,
    method = "mean",
    labels = NULL,
    colors = default.colors[1],
    shape = c("square", "triangle-up", "diamond", "circle"),
    confidence.interval = c("none", "crosshairs", "box"),
    outlier.interval = c("none", "crosshairs", "box"),
    label.offset = "bottom right",
    label.font.size = NULL,
    label.font.color = NULL,
    label.font.family = NULL,
    show.legend = T,
    legend.name = NULL,
    ...
)

Arguments
ena.plot  ENAplot object to use for plotting
points  A matrix or data.frame where columns contain coordinates of points in a projected ENA space
method  A function for computing a summary statistic for each column of points
labels  A character which will be the label for the group’s point
colors  A character, determines color of the group’s point, default: enaplot$color
shape A character, determines shape of the group’s point, choices: square, triangle, diamond, circle, default: square

confidence.interval A character that determines how the confidence interval is displayed, choices: none, box, crosshair, default: none

outlier.interval A character that determines how outlier interval is displayed, choices: none, box, crosshair, default: none

data(RS.data)
codeNames = c(
  'Data', 'Technical.Constraints', 'Performance.Parameters',

accum = ena.accumulate.data(
  units = RS.data[,c("UserName","Condition")],
  conversation = RS.data[,c("Condition","GroupName")],
  metadata = RS.data[,c("CONFIDENCE.Change","CONFIDENCE.Pre","CONFIDENCE.Post")],
  codes = RS.data[,codeNames],
  window.size.back = 4
)

label.offset label: top left (default), top center, top right, middle left, middle center, middle right, bottom left, bottom center, bottom right

data(RS.data)
codeNames = c(
  'Data', 'Technical.Constraints', 'Performance.Parameters',

accum = ena.accumulate.data(
  units = RS.data[,c("UserName","Condition")],
  conversation = RS.data[,c("Condition","GroupName")],
  metadata = RS.data[,c("CONFIDENCE.Change","CONFIDENCE.Pre","CONFIDENCE.Post")],
  codes = RS.data[,codeNames],
  window.size.back = 4
)

label.font.size An integer which determines the font size for label, default: enaplot$font.size

label.font.color A character which determines the color of label, default: enaplot$font.color

label.font.family A character which determines font type, choices: Arial, Courier New, Times New Roman, default: enaplot$font.family

show.legend Logical indicating whether to show the point labels in the in legend

legend.name Character indicating the name to show above the plot legend

Details

Plots a point based on a summary statistic for a group (typically, mean)

Value

The ENApplot provided to the function, with its plot updated to include the new group point.

See Also

ena.plot, ena.plot.points

Examples
set = ena.make.set(
    enadata = accum,
    rotation.by = ena.rotate.by.mean,
    rotation.params = list(
        accum$meta.data$Condition=="FirstGame",
        accum$meta.data$Condition=="SecondGame"
    )
)

plot = ena.plot(set)

unitNames = set$enadata$units

### Plot Condition 1 Group Mean
plot = ena.plot.group(plot, as.matrix(set$points$Condition$FirstGame), labels = "FirstGame",
                       colors = "red", confidence.interval = "box")

### plot Condition 2 Group Mean
plot = ena.plot.group(plot, as.matrix(set$points$Condition$SecondGame), labels = "SecondGame",
                       colors = "blue", confidence.interval = "box")

print(plot);

---

**ena.plot.network**  
*Plot an ENA network*

**Description**

Plot an ENA network: nodes and edges

**Usage**


```r
ena.plot.network(
    enaplot = NULL,
    network = NULL,
    node.positions = enaplot$enaset$rotation$nodes,
    adjacency.key = NULL,
    colors = c(pos = enaplot$palette[1], enaplot$palette[2]),
    edge_type = "line",
    show.all.nodes = T,
    threshold = c(0),
    thin.lines.in.front = T,
    thickness = c(min(abs(network)), max(abs(network))),
    opacity = thickness,
    saturation = thickness,
    scale.range = c(ifelse(min(network) == 0, 0, 0.1), 1),
    node.size = c(3, 10),
    labels = NULL,
```
Arguments

enaplot ENApplot object to use for plotting
network dataframe or matrix containing the edge weights for the network graph; typically comes from ENAset$line.weights
node.positions matrix containing the positions of the nodes. Defaults to enaplot$ENAset$node.positions
adjacency.key matrix containing the adjacency key for looking up the names and positions
colors A String or vector of colors for positive and negative line weights. E.g. red or c(pos= red, neg = blue), default: c(pos= red, neg = blue)
edge_type A String representing the type of line to draw, either "line", "dash", or "dot"
show.all.nodes A Logical variable, default: true
threshold A vector of numeric min/max values, default: c(0,Inf) plotting. Edge weights below the min value will not be displayed; edge weights above the max value will be shown at the max value.
thin.lines.in.front A logical, default: true
thickness A vector of numeric min/max values for thickness, default: c(min(abs(network)), max(abs(network))
opacity A vector of numeric min/max values for opacity, default: thickness
saturation A vector of numeric min/max values for saturation, default: thickness
scale.range A vector of numeric min/max to scale from, default: c(0.1,1) or if min(network) is 0, c(0,1)
node.size A lower and upper bound used for scaling the size of the nodes, default c(0, 20)
labels A character vector of node labels, default: code names
label.offset A character vector of representing the positional offset relative to the respective node. Defaults to "middle right" for all nodes. If a single values is provided, it is used for all positions, else the length of the
label.font.size An integer which determines the font size for graph labels, default: enaplot$font.size
label.font.color A character which determines the color of label font, default: enaplot$font.color
label.font.family A character which determines font type, choices: Arial, Courier New, Times New Roman, default: enaplot$font.family
legend.name  A character name used in the plot legend. Not included in legend when NULL (Default), if legend.include.edges is TRUE will always be "Nodes"

legend.include.edges  Logical value indicating if the edge names should be included in the plot legend. Forces legend.name to be "Nodes"

scale.weights  Logical indicating to scale the supplied network

Details

lots a network graph, including nodes (taken from codes in the ENAplot) and the edges (provided in network)

Value

The ENAplot provided to the function, with its plot updated to include the nodes and provided connecting lines.

See Also

ena.plot, ena.plot.points

Examples

data(RS.data)


accum = ena.accumulate.data(
  units = RS.data[,c("UserName","Condition")],
  conversation = RS.data[,c("Condition","GroupName")],
  metadata = RS.data[,c("CONFIDENCE.Change","CONFIDENCE.Pre","CONFIDENCE.Post")],
  codes = RS.data[,codeNames],
  window.size.back = 4
)

set = ena.make.set(
  enadata = accum,
  rotation.by = ena.rotate.by.mean,
  rotation.params = list(
    accum$meta.data$Condition=="FirstGame",
    accum$meta.data$Condition=="SecondGame"
  )
)

plot = ena.plot(set)

### Subset rotated points and plot Condition 1 Group Mean
as.matrix(set$points$Condition$FirstGame)
first.game.points = as.matrix(set$points$Condition$FirstGame)
plot = ena.plot.group(plot, first.game.points, labels = "FirstGame",
                      colors = "red", confidence.interval = "box")

### Subset rotated points and plot Condition 2 Group Mean
second.game.points = as.matrix(set$points$Condition$SecondGame)
plot = ena.plot.group(plot, second.game.points, labels = "SecondGame",
                      colors = "blue", confidence.interval = "box")

### get mean network plots
first.game.lineweights = as.matrix(set$line.weights$Condition$FirstGame)
first.game.mean = colMeans(first.game.lineweights)

second.game.lineweights = as.matrix(set$line.weights$Condition$SecondGame)
second.game.mean = colMeans(second.game.lineweights)

subtracted.network = first.game.mean - second.game.mean
plot = ena.plot.network(plot, network = subtracted.network)
print(plot)
`ena.plot.points`

```r
...
})
```

**Arguments**

- `enaplot` **ENAplot** object to use for plotting
- `points` A dataframe of matrix where the first two column are X and Y coordinates of points to plot in a projected ENA space defined in ENAplot
- `point.size` A data.frame or matrix where the first two column are X and Y coordinates of points to plot in a projected ENA space defined in ENAplot
- `labels` A character vector of point labels, length nrow(points); default: NULL
- `label.offset` character: top left (default), top center, top right, middle left, middle center, middle right, bottom left, bottom center, bottom right
- `label.group` A string used to group the labels in the legend. Items plotted with the same label.group will show/hide together when clicked within the legend.
- `label.font.size` An integer which determines the font size for point labels, default: enaplot$font.size
- `label.font.color` A character which determines the color of label font, default: enaplot$font.color
- `label.font.family` A character which determines label font type, choices: Arial, Courier New, Times New Roman, default: enaplot$font.family
- `shape` A character which determines the shape of point markers, choices: square, triangle, diamond, circle, default: circle
- `colors` A character vector of the point marker colors; if one given it is used for all, otherwise must be same length as points; default: black
- `confidence.interval.values` A matrix/dataframe where columns are CI x and y values for each point
- `confidence.interval` A character determining markings to use for confidence intervals, choices: none, box, crosshair, default: none
- `outlier.interval.values` A matrix/dataframe where columns are OI x and y values for each point
- `outlier.interval` A character determining markings to use for outlier interval, choices: none, box, crosshair, default: none
- `show.legend` Logical indicating whether to show the point labels in the in legend
- `legend.name` Character indicating the name to show above the plot legend
- `texts` [TBD]

**Value**

- **ENAplot** The ENAplot provided to the function, with its plot updated to include the new points.
See Also

`ena.plot`, `ENAplot`, `ena.plot.group`

Examples

```r
data(RS.data)


accum = ena.accumulate.data(
  units = RS.data[,c("UserName","Condition")],
  conversation = RS.data[,c("Condition","GroupName")],
  metadata = RS.data[,c("CONFIDENCE.Change","CONFIDENCE.Pre","CONFIDENCE.Post")],
  codes = RS.data[,codeNames],
  window.size.back = 4
)

set = ena.make.set(
  enadata = accum,
  rotation.by = ena.rotate.by.mean,
  rotation.params = list(
    accum$meta.data$Condition=="FirstGame",
    accum$meta.data$Condition=="SecondGame"
  )
)

plot = ena.plot(set)

group1.points = set$points[set$meta.data$Condition == "FirstGame",]
group2.points = set$points[set$meta.data$Condition == "SecondGame",]
plot = ena.plot.points(plot, points = group1.points);
plot = ena.plot.points(plot, points = group2.points);
print(plot);
```

Description

Function used to plot trajectories

Usage

```r
ena.plot.trajectory(
  enaplot,
  points,
  by = NULL,
)```
labels = NULL,
labels.show = c("Always", "Hover", "Both"),
names = NULL,
label.offset = NULL,
label.font.size = enaplot$get("font.size"),
label.font.color = enaplot$get("font.color"),
label.font.family = c("Arial", "Courier New", "Times New Roman"),
shape = c("circle", "square", "triangle-up", "diamond"),
colors = NULL,
default.hidden = F
)

**Arguments**

- **enaplot**: ENAplot object to use for plotting
- **points**: dataframe of matrix - first two column are X and Y coordinates, each row is a point in a trajectory
- **by**: vector used to subset points into individual trajectories, length nrow(points)
- **labels**: character vector - point labels, length nrow(points)
- **labels.show**: A character choice: Always, Hover, Both. Default: Both
- **names**: character vector - labels for each trajectory of points, length length(unique(by))
- **label.offset**: A numeric vector of an x and y value to offset labels from the coordinates of the points
- **label.font.size**: An integer which determines the font size for labels, default: enaplot$font.size
- **label.font.color**: A character which determines the color of label font, default: enaplot$font.color
- **label.font.family**: A character which determines font type, choices: Arial, Courier New, Times New Roman, default: enaplot$font.family
- **shape**: A character which determines the shape of markers, choices: square, triangle, diamond, circle, default: circle
- **colors**: A character vector, that determines marker color, default NULL results in alternating random colors. If single color is supplied, it will be used for all trajectories, otherwise the length of the supplied color vector should be equal to the length of the supplied names (i.e a color for each trajectory being plotted)
- **default.hidden**: A logical indicating if the trajectories should start hidden (click on the legend to show them) Default: FALSE

**Value**

The ENAplot provided to the function, with its plot updated to include the trajectories

**See Also**

ENAplot
Examples

```r
data(RS.data)


accum = ena.accumulate.data(
  units = RS.data[,c("UserName","Condition")],
  conversation = RS.data[,c("GroupName","ActivityNumber")],
  metadata = RS.data[,c("CONFIDENCE.Change","CONFIDENCE.Pre","CONFIDENCE.Post","C.Change")],
  codes = RS.data[,codeNames],
  window.size.back = 4,
  model = "A"
);

set = ena.make.set(accum);

### get mean network plots
first.game.lineweights = as.matrix(set$line.weights$Condition$FirstGame)
first.game.mean = colMeans(first.game.lineweights)

second.game.lineweights = as.matrix(set$line.weights$Condition$SecondGame)
second.game.mean = colMeans(second.game.lineweights)

subtracted.network = first.game.mean - second.game.mean

# Plot dimension 1 against ActivityNumber metadata
dim.by.activity = cbind(
  as.matrix(set$points)[,1],
  set$trajectories$ActivityNumber * .8/14 -.4 #scale down to dimension 1
)

plot = ena.plot(set)
plot = ena.plot.network(plot, network = subtracted.network, legend.name="Network")
plot = ena.plot.trajectory(
  plot,
  points = dim.by.activity,
  names = unique(set$model$unit.label),
  by = set$trajectories$ENA_UNIT
);
print(plot)
```

### ena.plotter

Wrapper to generate plots of units, groups, and networks

#### Description

Plots individual units, all units, groups of units, networks, and network subtractions
ena.plotter

Usage

ena.plotter(
  set,
  groupVar = NULL,
  groups = NULL,
  points = FALSE,
  mean = FALSE,
  network = TRUE,
  networkMultiplier = 1,
  subtractionMultiplier = 1,
  unit = NULL,
  print.plots = F,
  ...
)

Arguments

set an ena.set object

groupVar vector, character, of column name containing group identifiers.
groups vector, character, of values of groupVar column you wish to plot. Maximum of two groups allowed.

points logical, TRUE will plot points (default: FALSE)

mean logical, TRUE will plot the mean position of the groups defined in the groups argument (default: FALSE)

network logical, TRUE will plot networks (default: TRUE)

networkMultiplier numeric, scaling factor for non-subtracted networks (default: 1)

subtractionMultiplier numeric, scaling factor for subtracted networks (default: 1)

unit vector, character, name of a single unit to plot

print.plots logical, TRUE will show plots in the Viewer (default: FALSE)

... Additional parameters passed to set creation and plotting functions

Details

This function includes options to plots individual units, all units, groups of units, networks, and network subtractions, given an ena.set objects. Plots are stored on the supplied ena.set object.

Value

ena.set object
ena.set.creator

Description
Generates an ENA model by constructing a dimensional reduction of adjacency (co-occurrence) vectors as defined by the supplied conversations, units, and codes.

Usage
ena.set.creator(
  data,
  codes,
  units,
  conversation,
  metadata = NULL,
  model = c("EndPoint", "AccumulatedTrajectory", "SeparateTrajectory"),
  weight.by = "binary",
  window = c("MovingStanzaWindow", "Conversation"),
  window.size.back = 1,
  include.meta = TRUE,
)

ena.rotate.by.mean
ENA Rotate by mean

Description
Computes a dimensional reduction from a matrix of points such that the first dimension of the projected space passes through the means of two groups in the original space. Subsequent dimensions of the projected space are computed using ena.svd.

Usage
ena.rotate.by.mean(enaset, groups)

Arguments

enaset
An ENAset

groups
A list containing two logical vectors of length nrow(ENA.set$ena.data$units), where each vector defines whether a unit is in one of the two groups whose means are used to determine the dimensional reduction

Value

ENARotationSet

ena.set.creator
Wrapper to generate an ENA model
groupVar = NULL,
groups = NULL,
runTest = FALSE,
...
)

**Arguments**

data data.frame with containing metadata and coded columns
codes vector, numeric or character, of columns with codes
units vector, numeric or character, of columns representing units
conversation vector, numeric or character, of columns to segment conversations by
metadata vector, numeric or character, of columns with additional meta information for units
model character: EndPoint (default), AccumulatedTrajectory, SeparateTrajectory
weight.by "binary" is default, can supply a function to call (e.g. sum)
window MovingStanzaWindow (default) or Conversation
window.size.back Number of lines in the stanza window (default: 1)
include.meta [TBD]
groupVar vector, character, of column name containing group identifiers. If column contains at least two unique values, will generate model using a means rotation (a dimensional reduction maximizing the variance between the means of the two groups)
groups vector, character, of values of groupVar column used for means rotation or statistical tests
runTest logical, TRUE will run a Student’s t-Test and a Wilcoxon test for groups defined by the groups argument
...
Additional parameters passed to model generation

**Details**

This function generates an ena.set object given a data.frame, units, conversations, and codes. After accumulating the adjacency (co-occurrence) vectors, computes a dimensional reduction (projection), and calculates node positions in the projected ENA space. Returns location of the units in the projected space, as well as locations for node positions, and normalized adjacency (co-occurrence) vectors to construct network graphs. Includes options for returning statistical tests between groups of units.

**Value**

ena.set object
ena.svd  \hspace{1cm} ENA SVD

Description

ENA method computing a dimensional reduction of points in an ENA set using SVD

Usage

ena.svd(enaset, ...)

Arguments

enaset  \hspace{1cm} An ENAset
...

Unused, necessary for ena.make.set

ena.writeup  \hspace{1cm} Calculate the correlations

Description

Calculate both Spearman and Pearson correlations for the provided ENAset

Usage

ena.writeup(
enaset,
  tool = "rENA",
tool.version = as.character(packageVersion(tool)),
  comparison = NULL,
  comparison.groups = NULL,
  sig.dig = 2,
  output_dir = getwd(),
  type = c("file", "stream"),
  theory = T,
  methods = T,
  params = NULL,
  output_file = NULL
)
**Arguments**

- **enaset**: ENAset to view methods of
- **tool**: `c("rENA","webENA")`
- **tool.version**: `as.character(packageVersion(tool))`
- **comparison**: character string representing the comparison used, `c(NULL, "parametric", "non-parametric")`. Default NULL
- **comparison.groups**: Groups that were used for the comparison
- **sig.dig**: Integer for the number of digits to round to
- **output_dir**: Where to save the output file
- **type**: `c("file","stream")` File will save to a file in `output_dir`, Stream returns the contents directly
- **theory**: Logical indicating whether to include theory in the writeup
- **methods**: Logical indicating whether to include methods in the writeup
- **params**: additional parameters for `rmarkdown::render`
- **output_file**: character

**Value**

String representing the methods used to generate the model

---

**ENAd ata**

**ENAd ata R6class**

**Description**

ENAd ata R6class

ENAd ata R6class

**Public fields**

- **raw**: A data frame constructed from the unit, convo, code, and metadata parameters of `ena.accumulate.data`
- **adjacency.vectors**: A data frame of adjacency (co-occurrence) vectors by row
- **accumulated.adjacency.vectors**: A data frame of adjacency (co-occurrence) vectors accumulated per unit
- **model**: The type of ENA model: EndPoint, Accumulated Trajectory, or Separate Trajectory
- **units**: A data frame of columns that were combined to make the unique units. Includes column for trajectory selections. (unique)
- **unit.names**: A vector of unique unit values
- **metadata**: A data frame of unique metadata for each unit
trajectories A list: units - data frame, for a given row tells which trajectory it's a part; step - data frame, where along the trajectory a row sits

adjacency.matrix TBD
adjacency.vectors.raw TBD
codes A vector of code names

function.call The string representation of function called and parameters provided
function.params A list of all parameters sent to function call Construct ENAdata

Methods

Public methods:

• ENAdata$new()
• ENAdata$process()
• ENAdata$get()
• ENAdata$add.metadata()
• ENAdata$clone()

Method new():

Usage:
ENAdata$new(
  file,
  units = NULL,
  units.used = NULL,
  units.by = NULL,
  conversations.by = NULL,
  codes = NULL,
  model = NULL,
  weight.by = "binary",
  window.size.back = 1,
  window.size.forward = 0,
  mask = NULL,
  include.meta = T,
  ...
)

Arguments:
file TBD
units TBD
units.used TBD
units.by TBD
conversations.by TBD
codes TBD
model TBD
weight.by TBD
window.size.back TBD
window.size.forward TBD
mask TBD
include.meta TBD
... TBD

Returns: Process accumulation

Method process():
Usage:
ENAdata$process()

Returns: ENAdata Get property from object

Method get():
Usage:
ENAdata$get(x = "data")

Arguments:
x character key to retrieve from object

Returns: value from object at x Add metadata

Method add.metadata():
Usage:
ENAdata$add.metadata(merge = F)

Arguments:
merge logical (default: FALSE)

Returns: data.frame

Method clone(): The objects of this class are cloneable with this method.
Usage:
ENAdata$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.
Public fields

enaset - The ENAset object from which the ENAplot was constructed
plot - The plotly object used for data visualization
axes - TBD
point - TBD
palette - TBD
plotted - TBD Create ENAplot

Methods

Public methods:

• ENAplot$new()
• ENAplot$print()
• ENAplot$get()
• ENAplot$clone()

Method new():

Usage:
ENAplot$new(
enaset = NULL,
title = "ENA Plot",
dimension.labels = c("", ""),
font.size = 14,
font.color = "#000000",
font.family = "Arial",
scale.to = "network",
...)

Arguments:
enaset TBD
title TBD
dimension.labels TBD
font.size TBD
font.color TBD
font.family TBD
scale.to TBD
... TBD

Returns: ENAplot Print ENA plot

Method print():

Usage:
ENAplot$print()

Returns: Get property from object
**Method get():**

*Usage:*
ENAPlot$get(x)

*Arguments:*
- x: character key to retrieve from object

*Returns:*
value from object at x

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*
ENAPlot$clone(deep = FALSE)

*Arguments:*
- deep: Whether to make a deep clone.

---

**Description**

ENARotationSet R6class
ENARotationSet R6class

**Public fields**

- rotation: TBD
- node.positions: TBD
- codes: TBD
- eigenvalues: TBD

Create ENARotationSet

**Methods**

**Public methods:**

- ENARotationSet$new()
- ENARotationSet$clone()

**Method new():**

*Usage:*
ENARotationSet$new(rotation, codes, node.positions, eigenvalues = NULL)

*Arguments:*
- rotation: TBD
- codes: TBD
- node.positions: TBD
- eigenvalues: TBD
Returns: ENARotationsSet

Method clone(): The objects of this class are cloneable with this method.

Usage:
ENARotationSet$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

---

Description

ENAset R6class
ENAset R6class

Public fields

enadata An ENAdata object originally used to create the set
points.raw A data frame containing accumulated adjacency (co-occurrence) vectors per unit
points.normed.centered A data frame of centered normed accumulated adjacency (co-occurrence) vectors for each unit
points.rotated A data frame of point positions for number of dimensions specified in ena.make.set (i.e., the centered, normed, and rotated data)
line.weights A data frame of connections strengths per unit (Data frame of normed accumulated adjacency (co-occurrence) vectors for each unit)
node.positions - A data frame of positions for each code
codes - A vector of code names
rotation.set - An ENARotationSet object
variance - A vector of variance accounted for by each dimension specified
centroids - A matrix of the calculated centroid positions
function.call - The string representation of function called
function.params - A list of all parameters sent to function call
rotation_dists TBD
points.rotated.scaled TBD
points.rotated.non.zero TBD
line.weights.unrotated TBD
line.weights.non.zero TBD
correlations A data frame of spearman and pearson correlations for each dimension specified
center.align.to.origin - align point and centroid centers to origin
Methods

Public methods:

• ENAset$new()
• ENAset$process()
• ENAset$get()
• ENAset$clone()

Method new():

Usage:
ENAset$new(
enadata,  
dimensions = 2,  
norm.by = fun_sphere_norm,  
rotation.by = ena.svd.R6,  
rotation.params = NULL,  
rotation.set = NULL,  
node.position.method = lws.positions.sq.R6,  
endpoints.only = TRUE,  
center.align.to.origin = FALSE,  
...
)

Arguments:
enadata TBD  
dimensions TBD  
norm.by TBD  
rotation.by TBD  
rotation.params TBD  
rotation.set TBD  
node.position.method TBD  
endpoints.only TBD  
... TBD

Returns: ENAset Process ENAset

Method process():

Usage:
ENAset$process()

Returns: ENASet Get property from object

Method get():

Usage:
ENAset$get(x = "enadata")

Arguments:

x character key to retrieve from object
Method clone(): The objects of this class are cloneable with this method.

Usage:
ENAset$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

find_code_cols

Description
Find code columns

Usage
find_code_cols(x)

Arguments
x data.table (or frame) to search for columns of class ena.co.occurrence

Value
logical vector

ena_correlation

Calculate the correlations

Description
Calculate both Pearson correlations for the provided points and centroids

Usage
ena_correlation(points, centroids)

Arguments
points TBD
centroids TBD
find_dimension_cols  

**Description**

Find dimension columns

**Usage**

`find_dimension_cols(x)`

**Arguments**

- `x` data.table (or frame) to search for columns of class ena.dimension

**Value**

logical vector

---

find_meta_cols  

**Description**

Find metadata columns

**Usage**

`find_meta_cols(x)`

**Arguments**

- `x` data.table (or frame) to search for columns of class ena.metadata

**Value**

logical vector
### fun_cohens.d  
*Cohen's d*

**Description**
Calculate Conhen's d

**Usage**

```r
fun_cohens.d(x, y)
```

**Arguments**

- `x` [TBD]
- `y` [TBD]

**Details**

Cohen's d calculation

[TBD]

**Value**

numeric Cohen's d calculation

---

### fun_skip_sphere_norm  
*Non sphere norm*

**Description**
TBD

**Usage**

```r
fun_skip_sphere_norm(dfM)
```

**Arguments**

- `dfM`    Dataframe

**Details**

Non sphere norm
fun_sphere_norm

Description

TBD

Usage

fun_sphere_norm(dfM)

Arguments

dfM  Dataframe

Details

Sphere norm

---

means_rotate

Description

Title

Usage

means_rotate(x, on = NULL)

Arguments

x  [TBD]

on  [TBD]

Value

TBD
merge_columns_c  

**Description**

TBD

**Usage**

```
merge_columns_c(df, cols, sep = ".")
```

**Arguments**

- **df**: Dataframe
- **cols**: Vector
- **sep**: Character separator

**Details**

Merge data frame columns

---

methods_report  

**Description**

Methods report for rmarkdown

**Usage**

```
methods_report(
  toc = FALSE,
  toc_depth = 3,
  fig_width = 5,
  fig_height = 4,
  keep_md = FALSE,
  md_extensions = NULL,
  pandoc_args = NULL
)
```
**namesToAdjacencyKey**

**Names to Adjacency Key**

**Arguments**

- **toc** [TBD]
- **toc_depth** [TBD]
- **fig.width** [TBD]
- **fig.height** [TBD]
- **keep_md** [TBD]
- **md_extensions** [TBD]
- **pandoc_args** [TBD]

**Description**

Convert a vector of strings, representing names of a square matrix, to an adjacency

**Usage**

```
namesToAdjacencyKey(vector, upper_triangle = TRUE)
```

**Arguments**

- **vector** Vector representing the names of a square matrix
- **upper_triangle** Not Implemented

**Details**

Returns a matrix of 2 rows by choose(length(vector), 2) columns

---

**plot.ena.set**

**Plot an ena.set object**

**Description**

Plot an ena.set object

**Usage**

```
## S3 method for class 'ena.set'
plot(x, y, ...)
```
prepare_trajectory_data

Title

Description
 Title
Usage

```r
prepare_trajectory_data(
  x = NULL,
  by = x$'_function.params'$conversation[1],
  rotation_matrix = x$rotation.matrix,
  points = NULL,
  units = points,
  units_by = x$'_function.params'$units,
  steps = NULL
)
```

Arguments

- `x` [TBD]
- `by` [TBD]
- `rotation_matrix` [TBD]
- `points` [TBD]
- `units` [TBD]
- `units_by` [TBD]
- `steps` [TBD]

Value

TBD

---

print.enal.set | Title
---

Description

Title

Usage

```r
## S3 method for class 'ena.set'
print(x, ..., plot = FALSE, set = TRUE)
```

Arguments

- `x` [TBD]
- `...` [TBD]
- `plot` [TBD]
- `set` [TBD]
remove_meta_data

Value
TBD

<table>
<thead>
<tr>
<th>project_in</th>
<th>Title</th>
</tr>
</thead>
</table>

Description
Title

Usage
project_in(x, by = NULL, ...)

Arguments
- x [TBD]
- by [TBD]
- ... [TBD]

Value
TBD

remove_meta_data

Remove meta columns from data.table

Description
Remove meta columns from data.table

Usage
remove_meta_data(x)

Arguments
- x [TBD]

Value
data.table with the columns of class ena.meta.data removed
Description

rENA is used to create and visualize network models of discourse and other phenomena from coded data using Epistemic Network Analysis (ENA). A more complete description of the methods will be provided with the next release. See also XXXXX

RS.data

Coded Rescushell Chat Data

Description

A dataset containing sample chat data from the Rescushell Virtual Internship

Usage

RS.data

Format

An object of class 'data.frame' with 3824 rows and 20 columns.

scale.ena.set

Title

Description

Title

Usage

## S3 method for class 'ena.set'
scale(x, center = TRUE, scale = TRUE)

Arguments

x [TBD]
center Ignored.
scale [TBD]

Value

TBD
show

Title

Description
Title

Usage
show(x, ...)

Arguments
x [TBD]
...
[TBD]

Value
TBD

vector_to_ut

vector to upper triangle

Description
TBD

Usage
vector_to_ut(v)

Arguments
v [TBD]

Details
Upper Triangle from Vector
with_means

<table>
<thead>
<tr>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Title</td>
</tr>
<tr>
<td>Usage</td>
</tr>
<tr>
<td>with_means(x)</td>
</tr>
<tr>
<td>Arguments</td>
</tr>
<tr>
<td>x [TBD]</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>TBD</td>
</tr>
</tbody>
</table>

with_trajectory

<table>
<thead>
<tr>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Title</td>
</tr>
<tr>
<td>Usage</td>
</tr>
<tr>
<td>with_trajectory(</td>
</tr>
<tr>
<td>x,</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>by = x$_function.params_conversation[1],</td>
</tr>
<tr>
<td>add_jitter = TRUE,</td>
</tr>
<tr>
<td>frame = 1100,</td>
</tr>
<tr>
<td>transition = 1000,</td>
</tr>
<tr>
<td>easing = &quot;circle-in-out&quot;</td>
</tr>
<tr>
<td>)</td>
</tr>
<tr>
<td>Arguments</td>
</tr>
<tr>
<td>x [TBD]</td>
</tr>
<tr>
<td>... [TBD]</td>
</tr>
<tr>
<td>by [TBD]</td>
</tr>
<tr>
<td>add_jitter [TBD]</td>
</tr>
<tr>
<td>frame [TBD]</td>
</tr>
<tr>
<td>transition [TBD]</td>
</tr>
<tr>
<td>easing [TBD]</td>
</tr>
</tbody>
</table>
### $\$.ena.metadata

**Description**

Extract metadata easily

**Usage**

```r
## S3 method for class 'ena.metadata'
x$i
```

**Arguments**

- `x` [TBD]
- `i` [TBD]

**Value**

TBD

### $\$.ena.points

**Description**

Extract points easily

**Usage**

```r
## S3 method for class 'ena.points'
x$i
```

**Arguments**

- `x` [TBD]
- `i` [TBD]

**Value**

TBD
$.line.weights  

Extract line.weights easily

Description

Extract line.weights easily

Usage

```r
## S3 method for class 'line.weights'
x$i
```

Arguments

- **x** [TBD]
- **i** [TBD]

Value

TBD
Index

* datasets
  - RS.data, 53
  - $.ena.metadata, 56
  - $.ena.points, 56
  - $.line.weights, 57

add_group, 3
add_network, 4
add_nodes, 4
add_points, 5
add_trajectory, 5
as.ena.co.occurrence, 6
as.ena.matrix, 6
as.ena.metadata, 7
as.matrix.ena.connections, 7
as.matrix.ena.line.weights, 8
as.matrix.ena.matrix, 8
as.matrix.ena.nodes, 9
as.matrix.ena.points, 9
as.matrix.ena.rotation.matrix, 10
as.matrix.row.connections, 10
as_trajectory, 11

clear, 11
connection.matrix, 12

ena, 12
ena.accumulate.data, 14, 21
ena.conversations, 16
ena.correlations, 18
ena.group, 18
ena.make.set, 16, 19, 22
ena.plot, 21, 24, 27, 30, 31
ena.plot.group, 23, 30
ena.plot.network, 25
ena.plot.points, 22, 27, 28
ena.plot.trajectory, 30
ena.plotter, 32
ena.rotate.by.mean, 34
ena.set.creator, 34
ena.svd, 36
ena.writeup, 36
ena.correlation, 44
ENAdata, 16, 20, 37, 42
ENAplot, 22–24, 26, 27, 29–31, 39
ENARotationSet, 34, 41, 42
ENAs, 19, 21, 22, 34, 36, 40, 42
find_code_cols, 44
find_dimension_cols, 45
find_meta_cols, 45
fun_cohens.d, 46
fun_skip_sphere_norm, 46
fun_sphere_norm, 47
means_rotate, 47
merge_columns_c, 48
methods_report, 48
namesToAdjacencyKey, 49
plot.ena.set, 49
prepare_trajectory_data, 50
print.ena.set, 51
project_in, 52
remove_meta_data, 52
rENA, 53
RS.data, 53
scale.ena.set, 53
show, 54
vector_to_ut, 54
with_means, 55
with_trajectory, 55