Package ‘rENA’

March 25, 2022

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  0.2.4
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
URL  https://gitlab.com/epistemic-analytics/qe-packages/rENA
BugReports  https://gitlab.com/epistemic-analytics/qe-packages/rENA/-/issues
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  2022-03-25 16:50:02 UTC
R topics documented:

add_group .................................................. 3
add_network .................................................. 4
add_nodes .................................................... 4
add_points ................................................... 5
add_trajectory .............................................. 5
as.ena.co.ooccurrence ...................................... 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
combn_c2 ..................................................... 12
connection.matrix ......................................... 12
ena .......................................................... 13
ena.accumulate.data ........................................ 15
ena.conversations .......................................... 17
ena.correlations ............................................ 18
ena.group .................................................... 19
ena.make.set ................................................. 20
ea.plot ....................................................... 22
ena.plot.group ............................................... 23
ena.plot.network ........................................... 25
ena.plot.points ............................................. 28
ea.plot.trajectory .......................................... 31
ea.plotter .................................................... 33
ea.rotate.by.mean .......................................... 34
ea.set.creator ............................................... 35
ea.svd ......................................................... 36
ena.writeup .................................................. 37
ENAdata ...................................................... 38
ENApplot ..................................................... 40
ENARotationSet ............................................. 42
ENAsset ...................................................... 43
ena_correlation ............................................. 45
find_code_cols ............................................. 45
find_dimension_cols ....................................... 46
find_meta_cols .............................................. 46
fun_cohens.d ............................................... 47
fun_skip_sphere_norm ..................................... 47
fun_sphere_norm ............................................ 48
add_group

Description

Add a group mean to an ena.plot

Usage

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

Re-class vector as ena.co.occurrence

Description

Re-class vector as ena.co.occurrence

Usage

as.ena.co.occurrence(x)

Arguments

x Vector to re-class

Value

re-classed vector

as.ena.matrix

Re-class matrix as ena.matrix

Description

Re-class matrix as ena.matrix

Usage

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
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
## 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.points

### 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.row.connections**

*ENa row connections as matrix*

### Description

ENa row connections as matrix

### Usage

```r
## 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.matrix.row.connections**

*ENa row connections as matrix*

### Description

ENa row connections as matrix

### Usage

```r
## 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

```r
as_trajectory(
  x,
  by = x$`_function.params`$conversation[1],
  model = c("AccumulatedTrajectory", "SeparateTrajectory"),
  ...
)
```

#### Arguments

- **x**: [TBD]
- **by**: [TBD]
- **model**: [TBD]
- **...**: [TBD]

#### Value

TBD

### clear

#### Description

Title

#### Usage

```r
clear(x, wh = seq(x$plots))
```

#### Arguments

- **x**: [TBD]
- **wh**: [TBD]

#### Value

TBD
### combn_c2

**Fast combn choose 2**

**Description**

faster combn alternative

**Usage**

```r
combn_c2(n)
```

**Arguments**

- `n`: TBD

---

### connection.matrix

**Connection counts as square matrix**

**Description**

Connection counts as square matrix

**Usage**

```r
connection.matrix(x)
```

**Arguments**

- `x`: ena.set or ena.connections (i.e. set$connection.counts)

**Value**

- matrix
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

```r
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"),
...
ena.accumulate.data

```r
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**

```r
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
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
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
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
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'

Details
ENADatasets 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

data(RS.data)

codeNames = c('Data','Technical.Constraints','Performance.Parameters',
               'Collaboration');

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.correlations(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

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.
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

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

Arguments

enaset An ENAset or a vector 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 = TRUE, 
    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
ena.make.set

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

Value

ENAset class object that can be further processed for analysis or plotting

See Also

ena.accumulate.data, ENAset

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
)

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


ten.plot Generate a plot of an ENAset

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
ena.plot.group

Value

ENAplot used for plotting an ENAset

See Also

ena.make.set, ena.plot.points

Examples

data(RS.data)

codeNames = c('Var 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
)

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

plot = ena.plot(set)

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], 
)
Arguments

enaplot 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$color
shape A character, determines shape of the group’s point, choices: square, triangle, diamond, circle, default: square
confidence.interval A character that determines how the confidence interval is displayed, choices: none, box, crosshair, default: none
outlier.interval A character that determines how outlier interval is displayed, choices: none, box, crosshair, default: none
label.offset character: top left (default), top center, top right, middle left, middle center, middle right, bottom left, bottom center, bottom right
label.font.size An integer which determines the font size for label, default: enaplot$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 ENAplot provided to the function, with its plot updated to include the new group point.

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)

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);
Description

Plot an ENA network: nodes and edges

Usage

ena.plot.network(
enaplot = NULL,
network = NULL,
node.positions = enaplot$enaset$rotation$nodes,
adjacency.key = NULL,
colors = c(pos = enaplot$palette[1], enaplot$palette[2]),
edge_type = "line",
show.all.nodes = T,
threshold = c(0),
thin.lines.in.front = T,
layers = c("nodes", "edges"),
thickness = c(min(abs(network)), max(abs(network))),
opacity = thickness,
saturation = thickness,
scale.range = c(ifelse(min(network) == 0, 0, 0.1), 1),
node.size = c(3, 10),
labels = NULL,
label.offset = "middle right",
label.font.size = enaplot$get("font.size"),
label.font.color = enaplot$get("font.color"),
label.font.family = enaplot$get("font.family"),
legend.name = NULL,
legend.include.edges = F,
scale.weights = F,
...)

Arguments

enaplot ENAplot 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.
ena.plot.network

thin.lines.in.front
  A logical, default: true

layers
  ordering of layers, default: c("nodes", "edges")

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

... Additional parameters

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)
**Description**

Plot all or a subset of the points of an ENAplot using the plotly plotting library

**Usage**

```r
ena.plot.points(
  enaplot,
  points = NULL,
  point.size = enaplot$point$size,
  labels = NULL,
  label.offset = "top left",
  label.group = NULL,
  label.font.size = NULL,
  label.font.color = NULL,
  label.font.family = NULL,
  shape = "circle",
  colors = NULL,
  confidence.interval.values = NULL,
  confidence.interval = c("none", "crosshairs", "box"),
  outlier.interval.values = NULL,
  outlier.interval = c("none", "crosshairs", "box"),
  show.legend = T,
  legend.name = "Points",
  texts = NULL,
  ...)
```

**Arguments**

- **enaplot** `ENApplot` object to use for plotting
- **points** A dataframe of matrix where the first two column are X and Y coordinates
- **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
### ena.plot.points

**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]

...  
additional parameters addressed in inner function

### 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"
)
```
ena.plot.trajectory

Plot of ENA trajectories

Description

Function used to plot trajectories

Usage

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

ena.plot

Examples

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)

description

Plots individual units, all units, groups of units, networks, and network subtractions

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.
ena.rotate.by.mean

- **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

---

**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
**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**

```r
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,
  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**  
*ENA SVD*

**Description**

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

**Usage**

ena.svd(enaset, ...)

**Arguments**

ena.set  
An ENAset

...  
Unused, necessary for ena.make.set
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, 
output_format = 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
- **output_format**: character
Value
String representing the methods used to generate the model

---

### ENAdata R6class

#### Description
ENAdata R6class
- ENAdata 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

#### Methods

### Public methods:
- ENAdata$new()
- ENAdata$process()
- ENAdata$get()
- ENAdata$add.metadata()
- ENAdata$clone()

### Method `new()`:

#### Usage:
ENAdat$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:
ENAdat$process()

Returns: ENAdat Get property from object

Method get():
Usage:
ENAdat$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.

Description
ENAset R6class
ENAset R6class

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
showticklabels TBD
autosize TBD
automargin TBD
axispadding 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:
deept 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

ENAs et R6class
ENAs et 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 Create ENAset

Methods

Public methods:

- ENAs et$new()
- ENAset$process()
- ENAset$get()
- ENAset$clone()
**Method** `new()`:

*Usage:*

```r
ENOset$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 = TRUE, 
    ... 
)
```

*Arguments:*

- `enadata`: TBD
- `dimensions`: TBD
- `norm.by`: TBD
- `rotation.by`: TBD
- `rotation.params`: TBD
- `rotation.set`: TBD
- `node.position.method`: TBD
- `endpoints.only`: TBD
- `center.align.to.origin`: TBD
- `...`: TBD

*Returns:* ENAset Process ENAset

**Method** `process()`:

*Usage:*

```r
ENOset$process()
```

*Returns:* ENASet Get property from object

**Method** `get()`:

*Usage:*

```r
ENOset$get(x = "enadata")
```

*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:*

```r
ENOset$clone(deep = FALSE)
```

*Arguments:*

- `deep`: Whether to make a deep clone.
**ena_correlation**

*Calculate the correlations*

**Description**

Calculate both Pearson correlations for the provided points and centroids

**Usage**

`ena_correlation(points, centroids, conf_level = 0.95)`

**Arguments**

- `points` TBD
- `centroids` TBD
- `conf_level` TBD

**find_code_cols**

*Find code columns*

**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
find_dimension_cols  Find dimension columns

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  Find metadata columns

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**  
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**  
fun_skip_sphere_norm(dfM)

**Arguments**
- dfM Dataframe

**Details**  
Non sphere norm
fun_sphere_norm  Sphere norm

Description
TBD

Usage
fun_sphere_norm(dfM)

Arguments
- dfM  Dataframe

Details
Sphere norm

means_rotate  Title

Description
Title

Usage
means_rotate(x, on = NULL)

Arguments
- x  [TBD]
- on  [TBD]

Value
TBD
merge_columns_c

**Description**

TBD

**Usage**

```r
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**

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

Methods report for rmarkdown

Usage

methods_report_stream(
  toc = FALSE,
  toc_depth = 3,
  fig_width = 5,
  fig_height = 4,
  keep_md = FALSE,
  md_extensions = NULL,
  pandoc_args = NULL
)

Arguments

toc [TBD]
toc_depth [TBD]
fig_width [TBD]
fig_height [TBD]
keep_md [TBD]
md_extensions [TBD]
pandoc_args [TBD]
move_nodes_to_unit_circle

Title

Description
Title

Usage
move_nodes_to_unit_circle(set, dimension_name_1, dimension_name_2)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>set</td>
<td>TBD</td>
</tr>
<tr>
<td>dimension_name_1</td>
<td>TBD</td>
</tr>
<tr>
<td>dimension_name_2</td>
<td>TBD</td>
</tr>
</tbody>
</table>

Value
TBD

move_nodes_to_unit_circle_with_equal_space

Title

Description
Title

Usage
move_nodes_to_unit_circle_with_equal_space(set, dimension_name_1, dimension_name_2)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>set</td>
<td>TBD</td>
</tr>
<tr>
<td>dimension_name_1</td>
<td>TBD</td>
</tr>
<tr>
<td>dimension_name_2</td>
<td>TBD</td>
</tr>
</tbody>
</table>

Value
TBD
Value

TBD

\begin{itemize}
\item \texttt{namesToAdjacencyKey} \hspace{1em} \textit{Names to Adjacency Key}
\end{itemize}

\textbf{Description}

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

\textbf{Usage}

\begin{verbatim}
namesToAdjacencyKey(vector, upper_triangle = TRUE)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
\item \texttt{vector} \hspace{1em} Vector representing the names of a square matrix
\item \texttt{upper_triangle} \hspace{1em} Not Implemented
\end{itemize}

\textbf{Details}

Returns a matrix of 2 rows by \texttt{choose(length(vector), 2)} columns

\begin{itemize}
\item \texttt{plot.ena.set} \hspace{1em} \textit{Plot an ena.set object}
\end{itemize}

\textbf{Description}

Plot an ena.set object

\textbf{Usage}

\begin{verbatim}
## S3 method for \texttt{class} 'ena.set'
plot(x, y, \ldots)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
\item \texttt{x} \hspace{1em} ena.set to plot
\item \texttt{y} \hspace{1em} ignored.
\item \texttt{\ldots} \hspace{1em} Additional parameters passed along to ena.plot functions
\end{itemize}

\textbf{Value}

ena.plot.object
Examples

library(magrittr)

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
)

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

plot(set) %>%
  add_points(Condition$FirstGame, colors = "blue", with.mean = TRUE) %>%
  add_points(Condition$SecondGame, colors = "red", with.mean = TRUE)

plot(set) %>%
  add_network(Condition$FirstGame - Condition$SecondGame)

prepare_trajectory_data

Title

Description

Title

Usage

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.ena.set

<table>
<thead>
<tr>
<th>Description</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td></td>
</tr>
</tbody>
</table>

Usage

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

Arguments

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

Value

TBD
### project_in

**Title**

**Description**

Title

**Usage**

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

**Arguments**

- **x**: [TBD]
- **by**: [TBD]
- **...**: [TBD]

**Value**

TBD

---

### remove_meta_data

**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

---

### Description

Title

### Usage

`show(x, ...)`

### Arguments

- `x` [TBD]
- `...` [TBD]

### Value

TBD

---

vector_to_ut

---

### Description

TBD

### Usage

`vector_to_ut(v)`

### Arguments

- `v` [TBD]

### Details

Upper Triangle from Vector
with_means

### Description
Title

### Usage
```r
with_means(x)
```

### Arguments
- `x` [TBD]

### Value
TBD

---

with_trajectory

### Description
Title

### Usage
```r
with_trajectory(
    x,
    ..., 
    by = x$\_function.params\_conversation[1],
    add_jitter = TRUE,
    frame = 1100,
    transition = 1000,
    easing = "circle-in-out"
)
```

### Arguments
- `x` [TBD]
- `...` [TBD]
- `by` [TBD]
- `add_jitter` [TBD]
- `frame` [TBD]
- `transition` [TBD]
- `easing` [TBD]
$.ena.matrix

Value
TBD

---

$.ena.matrix *Extract from ena.matrix easily using metadata*

Description
Extract from ena.matrix easily using metadata

Usage
## S3 method for class 'ena.matrix'
x$i

Arguments
x [TBD]
i [TBD]

Value
TBD

---

$.ena.metadata *Extract metadata easily*

Description
Extract metadata easily

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

Arguments
x [TBD]
i [TBD]

Value
TBD
$.ena.points  

Extract points easily

**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, 56
  - .ena.matrix, 59
  - .ena.metadata, 59
  - .ena.points, 60
  - .line.weights, 60

  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
  combn_c2, 12
  connection.matrix, 12

  ena, 13
  ena.accumulate.data, 15, 21
  ena.conversations, 17
  ena.correlations, 18
  ena.group, 19
  ena.make.set, 17, 20, 23
  ena.plot, 22, 25, 27, 30, 32
  ena.plot.group, 23, 30
  ena.plot.network, 25
  ena.plot.points, 23, 27, 28
  ena.plot.trajectory, 31
  ena.plotter, 33

  ena.rotate.by.mean, 34
  ena.set.creator, 35
  ena.svd, 36
  ena.writeup, 37
  ena.correlation, 45
  ENAdata, 17, 20, 38, 43
  ENAplot, 23–27, 29–32, 40
  ENARotationSet, 34, 42, 43
  ENAset, 19, 21, 22, 34, 36, 40, 43

  find_code_cols, 45
  find_dimension_cols, 46
  find_meta_cols, 46
  fun_cohens.d, 47
  fun_skip_sphere_norm, 47
  fun_sphere_norm, 48

  means_rotate, 48
  merge_columns_c, 49
  methods_report, 49
  methods_report_stream, 50
  move_nodes_to_unit_circle, 51
  move_nodes_to_unit_circle_with_equal_space, 51

  namesToAdjacencyKey, 52
  plot.ena.set, 52
  prepare_trajectory_data, 53
  print.ena.set, 54
  project_in, 55
  remove_meta_data, 55
  rENA, 56
  RS.data, 56

  scale.ena.set, 56
  show, 57

  vector_to_ut, 57

  with_means, 58
  with_trajectory, 58

61