

# Package ‘rENA’

May 9, 2019

**Title** Epistemic Network Analysis

**Type** Package

**Author** Cody L Marquart [aut, cre],  
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** 0.1.6.1

**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) <doi:10.18608/jla.2016.33.3>.

**LazyData** TRUE

**Depends** R (>= 3.0.0)

**License** GPL-3 | file LICENSE

**LinkingTo** Rcpp, RcppArmadillo, RcppParallel, RcppEigen

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

**Suggests** testthat, knitr, rmarkdown

**RoxygenNote** 6.1.0

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2019-05-09 17:50:03 UTC

**R topics documented:**

cohens.d . . . . .	2
ena . . . . .	3
ena.accumulate.data . . . . .	4
ena.conversations . . . . .	6
ena.correlations . . . . .	7
ena.group . . . . .	8
ena.make.set . . . . .	9
ena.plot . . . . .	10
ena.plot.group . . . . .	12
ena.plot.network . . . . .	14
ena.plot.points . . . . .	16
ena.plot.trajectory . . . . .	18
ena.rotate.by.mean . . . . .	21
ena.svd . . . . .	21
ena.writeup . . . . .	22
ENadata . . . . .	23
ENApplot . . . . .	23
ENARotationSet . . . . .	24
ENaset . . . . .	24
group.stats . . . . .	25
methods_report . . . . .	26
namesToAdjacencyKey . . . . .	26
rENA . . . . .	27
RS.data . . . . .	27

**Index** **28**


---

cohens.d	<i>Cohen's d</i>
----------	------------------

---

**Description**

Calculate Conhen's d

**Usage**

cohens.d(x, y)

**Arguments**

x	[TBD]
y	[TBD]

**Details**

[TBD]

**Value**

numeric Cohen's d calculation

---

ena	<i>TBD</i>
-----	------------

---

**Description**

TBD

**Usage**

```
ena(data, codes, units, conversation, 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 = TRUE, groupCol = NULL, group1 = NULL,
     group2 = NULL, showPlots = TRUE, ...)
```

**Arguments**

data	A data frame
codes	Column names of data by which codes will be identified
units	Column names of data by which units will be identified
conversation	Column names of data by which conversations will be identified
metadata	(optional) Column names of data identifying 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
groupCol	(optional) Column name of data that includes values you wish to group and plot units by. Default is to plot a mean network for all units.
group1	(optional) Value of groupCol that you wish to group and plot units by
group2	(optional) Value of groupCol that you wish to group and plot units by
showPlots	Logical indicating if the function should show plots in the Viewer pane immediately after running. Default is TRUE
...	additional parameters addressed in inner function

**Details**

TBD

**Value**

[ENASET](#) class object that can be further processed for analysis or plotting, The [ENAPLOTS](#) requested.

**See Also**

[ENAdata](#), [ena.make.set](#), [ena.accumulate.data](#), [ena.plot.group](#), [ena.plot.network](#), [ena.plot](#)

---

`ena.accumulate.data`     *Accumulate data from a data frame into a set of adjacency (co-occurrence) vectors*

---

**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, ...)
```

**Arguments**

<code>units</code>	A data frame where the columns are the properties by which units will be identified
<code>conversation</code>	A data frame where the columns are the properties by which conversations will be identified
<code>codes</code>	A data frame where the columns are the codes used to create adjacency (co-occurrence) vectors
<code>metadata</code>	(optional) A data frame with additional columns of metadata to be associated with each unit in the data
<code>model</code>	A character, choices: EndPoint (or E), AccumulatedTrajectory (or A), or SeparateTrajectory (or S); default: EndPoint. Determines the ENA model to be constructed
<code>weight.by</code>	(optional) A function to apply to values after accumulation
<code>window</code>	A character, choices are Conversation (or C), MovingStanzaWindow (MSW, MS); default MovingStanzaWindow. Determines how stanzas are constructed, which defines how co-occurrences are modeled
<code>window.size.back</code>	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
<code>window.size.forward</code>	(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
<code>mask</code>	(optional) A binary matrix of size <code>ncol(codes) x ncol(codes)</code> . 0s in the mask matrix row <code>i</code> column <code>j</code> indicates that co-occurrence will not be modeled between code <code>i</code> and code <code>j</code>
<code>include.meta</code>	Logical indicating if unit metadata should be attached to the resulting ENAdata object, default is TRUE
<code>...</code>	additional parameters addressed in inner function

**Details**

ENADdata 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](#)

---

ena.conversations      *Find conversations by unit*

---

### 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',
              'Client.and.Consultant.Requests','Design.Reasoning',
              '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$metadata$Condition=="FirstGame",
                        accum$metadata$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**

```
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	<i>Compute summary statistic for groupings of units using given method (typically, mean)</i>
-----------	--

---

**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)
```

**Arguments**

enaset	An <a href="#">ENASET</a> 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 groups using 'mean' regardless of 'method' provided

**Value**

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

**Examples**

```
data(RS.data)

codeNames = c('Data', 'Technical.Constraints', 'Performance.Parameters',
              'Client.and.Consultant.Requests', 'Design.Reasoning', 'Collaboration');

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, by=accum$metadata$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

```

ena.make.set(enadata, dimensions = 2, norm.by = sphere_norm_c,
  rotation.by = ena.svd, rotation.params = NULL, rotation.set = NULL,
  endpoints.only = T, node.position.method = lws.positions.sq, ...)

```

### Arguments

enadata	<a href="#">ENAdata</a> 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: <code>sphere_norm_c()</code>
rotation.by	A function to be used to compute the dimensional reduction, default: <code>ena.svd()</code>
rotation.params	(optional) A character vector containing additional parameters for the function in <code>rotation.by</code> , if needed
rotation.set	A previously-constructed <code>ENARotationSet</code> object to use for the dimensional reduction
endpoints.only	A logical variable which determines whether to only show endpoints for trajectory models
node.position.method	A function to be used to determine node positions based on the dimensional reduction, default: <code>lws.position.es()</code>
...	additional parameters addressed in inner function

### Details

This function generates an `ENAsset` 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)

codeNames = c('Data', 'Technical.Constraints', 'Performance.Parameters',
              'Client.and.Consultant.Requests', 'Design.Reasoning', 'Collaboration');

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$metadata$Condition=="FirstGame",
    accum$metadata$Condition=="SecondGame"
  )
)
```

---

ena.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("X", "Y"),
        font.size = 10, font.color = "#000000", font.family = c("Arial",
        "Courier New", "Times New Roman"), scale.to = c("network", "points"),
        ...)
```

**Arguments**

enaset	The <a href="#">ENASET</a> 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	Either network or points. Default: "network"
...	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 be 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)

codeNames = c('Data', 'Technical.Constraints', 'Performance.Parameters',
              'Client.and.Consultant.Requests', 'Design.Reasoning', 'Collaboration');

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 = "black", 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,
  ...)
```

### Arguments

enaplot	<a href="#">ENApplot</a> 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
...	Additional parameters

**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)

codeNames = c('Data', 'Technical.Constraints', 'Performance.Parameters',
              'Client.and.Consultant.Requests', 'Design.Reasoning', 'Collaboration');

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$metadata$Condition=="FirstGame",
    accum$metadata$Condition=="SecondGame"
  )
)

plot = ena.plot(set)

unitNames = set$enadata$units

### Subset rotated points and plot Condition 1 Group Mean
first.game = unitNames$Condition == "FirstGame"
first.game.points = set$points.rotated[first.game,]
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 = unitNames$Condition == "SecondGame"
second.game.points = set$points.rotated[second.game,]
plot = ena.plot.group(plot, second.game.points, 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

```
ena.plot.network(enaplot = NULL, network = NULL,
  node.positions = enaplot$enaset$node.positions,
  adjacency.key = namesToAdjacencyKey(rownames(node.positions)),
  colors = c(pos = "red", "blue"), 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 = rownames(node.positions),
  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 = T, ...)
```

## Arguments

enaplot	<a href="#">ENApplot</a> 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
...	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)

codeNames = c('Data', 'Technical.Constraints', 'Performance.Parameters',
              'Client.and.Consultant.Requests', 'Design.Reasoning', 'Collaboration');

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$metadata$Condition=="FirstGame",
    accum$metadata$Condition=="SecondGame"
  )
)

plot = ena.plot(set)

unitNames = set$enadata$units

### Subset rotated points and plot Condition 1 Group Mean
first.game = unitNames$Condition == "FirstGame"
first.game.points = set$points.rotated[first.game,]
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 = unitNames$Condition == "SecondGame"
second.game.points = set$points.rotated[second.game,]
plot = ena.plot.group(plot, second.game.points, labels = "SecondGame",
  colors = "blue", confidence.interval = "box")

### get mean network plots
first.game.lineweights = set$line.weights[first.game,]
first.game.mean = colMeans(first.game.lineweights)

second.game.lineweights = set$line.weights[second.game,]
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

*Plot points on an ENAplot*


---

## Description

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



**Usage**

```
ena.plot.points(enaplot, points = NULL, point.size = 5,
  labels = NULL, label.offset = "top left", label.group = NULL,
  label.font.size = NULL, label.font.color = NULL,
  label.font.family = NULL, shape = "circle",
  colors = default.colors[1], confidence.interval.values = NULL,
  confidence.interval = c("none", "crosshairs", "box"),
  outlier.interval.values = NULL, outlier.interval = c("none",
  "crosshairs", "box"), show.legend = T, ...)
```

**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 ENApplot
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
...	additional parameters addressed in inner function

**Value**

[ENApplot](#) The ENApplot provided to the function, with its plot updated to include the new points.

**See Also**

[ena.plot](#), [ENApplot](#), [ena.plot.group](#)

**Examples**

```

data(RS.data)

codeNames = c('Data','Technical.Constraints','Performance.Parameters',
  'Client.and.Consultant.Requests','Design.Reasoning','Collaboration');

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$metadata$Condition=="FirstGame",
    accum$metadata$Condition=="SecondGame"
  )
)

plot = ena.plot(set)

group1.points = set$points.rotated[set$enadata$units$Condition == "FirstGame",]
group2.points = set$points.rotated[set$enadata$units$Condition == "SecondGame",]
plot = ena.plot.points(plot, points = group1.points);
plot = ena.plot.points(plot, points = group2.points);
print(plot);

```

---

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	<a href="#">ENApplot</a> 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 [ENApplot](#) provided to the function, with its plot updated to include the trajectories

**See Also**

[ena.plot](#)

**Examples**

```

data(RS.data)

codeNames = c('Data','Technical.Constraints','Performance.Parameters',
  'Client.and.Consultant.Requests','Design.Reasoning','Collaboration');

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);

unitNames = set$enadata$units

### Subset rotated points and plot Condition 1 Group Mean
first.game = unitNames$Condition == "FirstGame"
first.game.points = set$points.rotated[first.game,]

### Subset rotated points and plot Condition 2 Group Mean
second.game = unitNames$Condition == "SecondGame"
second.game.points = set$points.rotated[second.game,]

### get mean network plots
first.game.lineweights = set$line.weights[first.game,]
first.game.mean = colMeans(first.game.lineweights)

second.game.lineweights = set$line.weights[second.game,]
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(
  set$points.rotated[,1],
  set$enadata$trajectories$step$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$enadata$units$UserName),
  by = set$enadata$units$UserName
);
print(plot)

```

---

ena.rotate.by.mean	<i>ENA Rotate by mean</i>
--------------------	---------------------------

---

**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 a the original space. Subsequent dimensions of the projected space are computed using ena.svd

**Usage**

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

**Arguments**

enaset	An <a href="#">ENAsset</a>
groups	A list containing two logical vectors of length <code>nrow(ENA.set\$ena.data\$units)</code> , 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.svd	<i>ENA SVD</i>
---------	----------------

---

**Description**

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

**Usage**

```
ena.svd(enaset, ...)
```

**Arguments**

enaset	An <a href="#">ENAsset</a>
...	Unused, necessary for ena.make.set

---

ena.writeup	<i>Calculate the correlations</i>
-------------	-----------------------------------

---

## Description

Calculate both Spearman and Pearson correlations for the provided ENAset

## Usage

```
ena.writeup(ena.set, 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)
```

## Arguments

ena.set	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

## Value

String representing the methods used to generate the model

---

ENAdata	<i>ENAdata R6class</i>
---------	------------------------

---

**Description**

ENAdata R6class

**Usage**

ENAdata

**Format**

An object of class R6ClassGenerator of length 24.

**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

`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

---

ENApplot	<i>ENASET R6class</i>
----------	-----------------------

---

**Description**

ENASET R6class

**Usage**

ENApplot

**Format**

An object of class R6ClassGenerator of length 24.

**Fields**

enaset - The [ENAsset](#) object from which the ENAplot was constructed

plot - The plotly object used for data visualization

---

ENARotationSet	<i>ENARotationSet R6class</i>
----------------	-------------------------------

---

**Description**

ENARotationSet R6class

**Usage**

ENARotationSet

**Format**

An object of class R6ClassGenerator of length 24.

---

ENAsset	<i>ENAsset R6class</i>
---------	------------------------

---

**Description**

ENAsset R6class

**Usage**

ENAsset

**Format**

An object of class R6ClassGenerator of length 24.



**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  
 correlation - A data frame of spearman and pearson correlations for each dimension specified  
 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

---

 group.stats

*Group Stats*


---

**Description**

Generate comparison stats for groups within set

**Usage**

```
group.stats(groupOne, groupTwo)
```

**Arguments**

groupOne	[TBD]
groupTwo	[TBD]

**Details**

[TBD]

**Value**

list containing all of the statistics

methods\_report      *methods\_report*

---

**Description**

Methods report for rmarkdwon

**Usage**

```
methods_report(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]

---

namesToAdjacencyKey      *Names to Adjacency Key*

---

**Description**

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

**Usage**

```
namesToAdjacencyKey(vector)
```

**Arguments**

vector	Vector representing the names of a square matrix
--------	--

**Details**

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

---

rENA	<i>rENA creates ENA sets</i>
------	------------------------------

---

**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	<i>Coded Rescushell Chat Data</i>
---------	-----------------------------------

---

**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.

# Index

- \*Topic **ENA**,
  - ena.group, 8
  - ena.make.set, 9
  - ena.plot, 10
  - ena.plot.group, 12
  - ena.plot.network, 14
  - ena.plot.points, 16
  - ena.plot.trajectory, 18
- \*Topic **accumulate**,
  - ena, 3
- \*Topic **accumulate**
  - ena.accumulate.data, 4
- \*Topic **creation**,
  - ena, 3
- \*Topic **data**,
  - ena, 3
  - ena.accumulate.data, 4
- \*Topic **datasets**
  - ENAdata, 23
  - ENApplot, 23
  - ENARotationSet, 24
  - ENASet, 24
  - RS.data, 27
- \*Topic **edges**
  - ena.plot.network, 14
- \*Topic **generate**,
  - ena.make.set, 9
  - ena.plot, 10
- \*Topic **group**
  - ena.group, 8
  - ena.plot.group, 12
- \*Topic **network**,
  - ena.plot.network, 14
- \*Topic **nodes**,
  - ena.plot.network, 14
- \*Topic **plot**,
  - ena.plot.group, 12
  - ena.plot.network, 14
  - ena.plot.points, 16
  - ena.plot.trajectory, 18
- \*Topic **plotting**
  - ena, 3
- \*Topic **plot**
  - ena.plot, 10
- \*Topic **points**
  - ena.plot.points, 16
- \*Topic **set**,
  - ena.group, 8
- \*Topic **set**
  - ena, 3
  - ena.make.set, 9
- \*Topic **trajectory**
  - ena.plot.trajectory, 18
- cohens.d, 2
- ena, 3
- ena.accumulate.data, 4, 4, 10
- ena.conversations, 6
- ena.correlations, 7
- ena.group, 8
- ena.make.set, 4, 6, 9, 11
- ena.plot, 4, 10, 13, 15, 18, 19
- ena.plot.group, 4, 12, 18
- ena.plot.network, 4, 14
- ena.plot.points, 11, 15, 16
- ena.plot.trajectory, 18
- ena.rotate.by.mean, 21
- ena.svd, 21
- ena.writeup, 22
- ENAdata, 4, 6, 9, 23, 25
- ENApplot, 4, 11–15, 17–19, 23
- ENARotationSet, 21, 24, 25
- ENASet, 4, 8, 10, 11, 21, 24, 24
- group.stats, 25
- methods\_report, 26
- namesToAdjacencyKey, 26

rENA, [27](#)

RS.data, [27](#)