Package: rENA (via r-universe)

October 12, 2026

Title Epistemic Network Analysis

Type Package

Author Cody L Marquart [aut, cre] (ORCID: <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 0.3.1

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).

LazyData TRUE

Encoding UTF-8

Depends R (>= 4.1.0)

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, stats, plotly, doParallel, parallel, scales, glmnet, tma

Suggests testthat (>= 2.1.0), knitr, rmarkdown

RoxygenNote 7.3.3

NeedsCompilation yes

Repository https://cran.r-universe.dev

Date/Publication 2026-02-11 16:20:09 UTC

RemoteUrl https://github.com/cran/rENA

RemoteRef HEAD

RemoteSha f666a52f68f744f8573e52c47d640e6f2531d63f

Contents

*.ena.matrix . . . 4
$.ena.matrix . . . 5
$.ena.metadata . . . 5
$.ena.points . . . 6
$.line.weights . . . 6
accumulate . . . 7
add_group . . . 8
add_network . . . 9
add_nodes . . . 11
add_points . . . 12
add_trajectory . . . 13
as.ena.co.occurrence . . . 15
as.ena.matrix . . . 15
as.ena.metadata . . . 16
as.matrix.ena.connections . . . 16
as.matrix.ena.line.weights . . . 17
as.matrix.ena.matrix . . . 17
as.matrix.ena.nodes . . . 18
as.matrix.ena.points . . . 18
as.matrix.ena.rotation.matrix . . . 19
as.matrix.row.connections . . . 19
as.qe.code . . . 20
as.qe.data . . . 20
as.qe.horizon . . . 21
as.qe.metadata . . . 22
as.qe.unit . . . 22
as_trajectory . . . 23
center . . . 23
check_range . . . 24
clear . . . 25
codes . . . 26
combn_c2 . . . 27
compute_SB . . . 27
connection.matrix . . . 28
define . . . 28
directed_node_positions . . . 29
directed_node_positions_with_ground_response_added . . . 30
ena . . . 30

ena.accumulate.data . . . 33
ena.conversations . . . 35
ena.correlations . . . 36
ena.group . . . 37
ena.make.set . . . 38
ena.plot . . . 40
ena.plot.group . . . 41
ena.plot.network . . . 43
ena.plot.points . . . 46
ena.plot.trajectory . . . 49
ena.plotter . . . 51
ena.rotate.by.generalized . . . 52
ena.rotate.by.hena.regression . . . 53
ena.rotate.by.hena.regression_2 . . . 53
ena.rotate.by.mean . . . 54
ena.rotation.h . . . 55
ena.set.creator . . . 55
ena.svd . . . 57
ena.writeup . . . 58
ena_correlation . . . 59
ENAdata . . . 59
ENAplot . . . 61
ENARotationSet . . . 63
ENAset . . . 64
find_binary_cols . . . 66
find_code_cols . . . 67
find_dimension_cols . . . 67
find_meta_cols . . . 68
fun_cohens.d . . . 68
fun_skip_sphere_norm . . . 69
fun_sphere_norm . . . 69
get_x1_main_effect . . . 70
gmr . . . 71
group . . . 72
horizon . . . 73
is.qe.code . . . 74
is.qe.data . . . 74
is.qe.horizon . . . 75
is.qe.metadata . . . 76
is.qe.unit . . . 76
means_rotate . . . 77
merge_columns_c . . . 77
metadata . . . 78
methods_report . . . 78
methods_report_stream . . . 79
model . . . 80
move_nodes_to_unit_circle . . . 81
move_nodes_to_unit_circle_with_equal_space . . . 82

namesToAdjacencyKey 82
optimize 83
plot.ena.set 84
prepare_trajectory_data 85
print.ena.set 87
project 87
project_in 88
reclassify 89
remove_meta_data 89
rENA 90
rotate 90
RS.data 91
scale.ENAplot 91
show 92
sphere_norm 92
units 93
vector_to_ut 94
with.ena.matrix 94
with_means 95
with_trajectory 95

Index **97**

`*.ena.matrix` *Multiply ena.matrix objects Element-wise multiplication of dimension columns in an ena.matrix by another ena.matrix or numeric matrix. If e2 is an ena.matrix, it is converted to a standard matrix before multiplication. The multiplication is applied only to the dimension columns of e1, while other columns remain unchanged.*

Description

Multiply ena.matrix objects Element-wise multiplication of dimension columns in an ena.matrix by another ena.matrix or numeric matrix. If e2 is an ena.matrix, it is converted to a standard matrix before multiplication. The multiplication is applied only to the dimension columns of e1, while other columns remain unchanged.

Usage

```
## S3 method for class 'ena.matrix'
e1 * e2
```

Arguments

`e1`	An ena.matrix object whose dimension columns will be multiplied.
`e2`	An ena.matrix or numeric matrix to multiply with the dimension columns of e1.

Value

An ena.matrix object with the dimension columns of e1 multiplied by e2.

$.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

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

Arguments

x	[TBD]
i	[TBD]

Value

[TBD]

$.line.weights *Extract line.weignts easily*

Description

Extract line.weignts easily

Usage

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

Arguments

x	[TBD]
i	[TBD]

Value

[TBD]

accumulate *Accumulate Connection Counts for ENA*

Description

This function takes a data.frame and accumulates co-occurrences of codes within specified units and conversations (horizon), preparing it for ENA. It's designed to be used with pipes ('|>')..

Usage

```
accumulate(
  x,
  units = rENA::units(x),
  codes = rENA::codes(x),
  horizon = rENA::horizon(x),
  ...,
  ordered = FALSE,
  binary = TRUE
)
```

Arguments

x	A data.frame or similar object containing the data to be analyzed.
units	A character vector specifying the columns that define the units of analysis.
codes	A character vector specifying the columns that contain the codes for co-occurrence analysis.
horizon	A character vector specifying the columns that define the conversational boundaries (horizon).
...	Additional arguments passed to underlying accumulation functions.
ordered	A logical value. If TRUE, creates ordered networks (A -> B is different from B -> A). Defaults to FALSE.
binary	A logical value. If TRUE, connection counts are binarized (0 or 1). Defaults to TRUE.

Value

An ena.set object containing the accumulated connection counts and metadata.

Examples

```
data(RS.data)

codes <- c("Data", "Technical.Constraints", "Performance.Parameters",
           "Client.and.Consultant.Requests", "Design.Reasoning",
           "Collaboration")
units <- c("Condition", "UserName")
```

```
horizon <- c("Condition", "GroupName")
enaset <- RS.data |>
  accumulate(units, codes, horizon)
```

---

add_group *Add a group mean to an ENA plot*

Description

This function adds a group mean to an existing ENA plot or ENA set. It supports various input types for the ‘wh‘ parameter, including unevaluated expressions and language objects.

Usage

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

Arguments

x	An ‘ENAplot‘ object or an ENA set containing plots.
wh	Specifies the group to plot. Can be an unevaluated expression or a language object.
...	Additional parameters passed to the plotting functions.

Details

The function determines the type of the ‘wh‘ parameter and processes it accordingly: - If ‘wh‘ is an unevaluated expression, it is captured and evaluated in the parent frame. - If ‘wh‘ is a language object, it is processed to extract the relevant group information.

The function updates the plot with the new group mean and stores the updated plot back in the ENA set.

Value

Invisibly returns the modified ENA set.

Examples

```
# Load test data
data(RS.data);

# Define codenames for the test
codenames <- c("Data", "Technical.Constraints", "Performance.Parameters",
  "Client.and.Consultant.Requests", "Design.Reasoning", "Collaboration");

# Standard ENA accumulation
accum <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
```

```
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames
);
# Create a standard ENA set
newset <- ena.make.set(accum);

# Simple plot for a set of points, and their mean
newplot <- plot(newset) |>
  add_points(Condition$FirstGame, mean = TRUE, colors = "blue") |>
  add_network();

# Plot a single unit's point and it's line.weights
plot(newset) |>
  add_points(UserName$`steven z`) |>
  add_network();

# Trajectory accumulation and plotting
trajectory_accumulation <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames,
  model = "A"
);
trajectory_model <- ena.make.set(trajectory_accumulation);

newplot3 <- plot(trajectory_model) |>
  add_trajectory(Condition$FirstGame);
```

---

add_network *Add a network to an ENA plot*

Description

Adds a network (set of edges) to an existing ENA plot or ENA set. The network can be specified in several ways, including as an unevaluated expression, a numeric matrix, or a language object. This function is typically used to visualize group means, differences between groups, or custom networks on an ENA plot.

Usage

```
add_network(
  x,
  wh = NULL,
  ...,
  with.mean = F,
  edge.multiplier = 1,
  colors = NULL
)
```

Arguments

x	An ‘ENAplot‘ object or an ENA set containing plots.
wh	Specifies the network to plot. Can be: • An unevaluated expression (e.g., ‘Condition$FirstGame - Condition$SecondGame‘) • A numeric matrix or data.frame of edge weights • A language object • NULL (defaults to the mean network)
...	Additional parameters passed to the plotting functions.
with.mean	Logical; if ‘TRUE‘, also plots the mean for the points in the network. Default is ‘FALSE‘.
edge.multiplier	Numeric scalar to multiply the edge weights. Useful for scaling the network visualization. Default is 1.
colors	Optional vector of colors for the network. If not specified, colors are chosen from the plot palette.

Details

The function determines the type of the ‘wh‘ parameter and processes it accordingly:

- If ‘wh‘ is an unevaluated expression, it is captured and evaluated in the parent frame. This allows for flexible specification of group means or differences.
- If ‘wh‘ is a numeric matrix or data.frame, it is used directly as the network data.
- If ‘wh‘ is a language object, it is processed to extract the relevant network information.
- If ‘wh‘ is NULL, the mean network is plotted.

The function updates the plot with the new network and returns the modified plot object. The ENA set is not modified in-place.

Value

The modified ENAplot object with the new network added.

Examples

See ‘inst/examples/example-plot-piping.R‘ for usage examples.

add_nodes *Add nodes to an ENA plot*

Description

This function adds nodes to an existing ENA plot or ENA set. It can be used to customize the nodes displayed on the plot, including their size and other graphical parameters.

Usage

```
add_nodes(x, ..., return_plot = FALSE)
```

Arguments

x	An ENAplot object or an ENA set containing plots.
...	Additional arguments passed to ena.plot.points, such as nodes, size, and other graphical parameters.
return_plot	Logical; if TRUE, returns the modified ENA set. If FALSE (default), returns the modified plot invisibly.

Details

If x is an ENAplot, the function extracts the associated ENA set and plot. Otherwise, it assumes x is an ENA set and uses the last plot in the set. The nodes to be added can be specified via the nodes argument; otherwise, the default nodes from the set's rotation are used. Node size can be customized via the size argument.

The function updates the plot with the new nodes and stores the updated plot back in the ENA set.

Value

Invisibly returns the modified plot or ENA set, depending on the value of return_plot.

Examples

```
# Load test data
data(RS.data);

# Define codenames for the test
codenames <- c("Data", "Technical.Constraints", "Performance.Parameters",
  "Client.and.Consultant.Requests", "Design.Reasoning", "Collaboration");

# Standard ENA accumulation
accum <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames
);
# Create a standard ENA set
```

```
newset <- ena.make.set(accum);

# Simple plot for a set of points, and their mean
newplot <- plot(newset) |>
  add_points(Condition$FirstGame, mean = TRUE, colors = "blue") |>
  add_network();

# Plot a single unit's point and it's line.weights
plot(newset) |>
  add_points(UserName$`steven z`) |>
  add_network();

# Trajectory accumulation and plotting
trajectory_accumulation <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames,
  model = "A"
);
trajectory_model <- ena.make.set(trajectory_accumulation);

newplot3 <- plot(trajectory_model) |>
  add_trajectory(Condition$FirstGame);
```

`add_points` *Add points to an ENA plot*

Description

This function adds points to an existing ENA plot or ENA set. It supports various input types for the ‘wh‘ parameter, including unevaluated expressions and language objects.

Usage

```
add_points(x, wh = NULL, ..., colors = NULL)
```

Arguments

`x`	An ‘ENAplot‘ object or an ENA set containing plots.
`wh`	Specifies the points to plot. Can be an unevaluated expression or a language object.
`...`	Additional parameters passed to the plotting functions.
`colors`	A vector of colors for the plotted points. Default is ‘NULL‘.

Details

The function determines the type of the ‘wh‘ parameter and processes it accordingly: - If ‘wh‘ is an unevaluated expression, it is captured and evaluated in the parent frame. - If ‘wh‘ is a language object, it is processed to extract the relevant points information.

The function updates the plot with the new points and stores the updated plot back in the ENA set.

Value

Invisibly returns the modified ENA set.

Examples

```
# Load test data
data(RS.data);

# Define codenames for the test
codenames <- c("Data", "Technical.Constraints", "Performance.Parameters",
  "Client.and.Consultant.Requests", "Design.Reasoning", "Collaboration");

# Standard ENA accumulation
accum <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames
);
# Create a standard ENA set
newset <- ena.make.set(accum);

# Simple plot for a set of points, and their mean
newplot <- plot(newset) |>
  add_points(Condition$FirstGame, mean = TRUE, colors = "blue") |>
  add_network();

# Plot a single unit's point and it's line.weights
plot(newset) |>
  add_points(UserName$`steven z`) |>
  add_network();

# Trajectory accumulation and plotting
trajectory_accumulation <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames,
  model = "A"
);
trajectory_model <- ena.make.set(trajectory_accumulation);

newplot3 <- plot(trajectory_model) |>
  add_trajectory(Condition$FirstGame);
```

`add_trajectory` *Add a trajectory to an ENA plot*

Description

This function adds a trajectory to an existing ENA plot or ENA set. It supports various input types for the ‘wh‘ parameter, including unevaluated expressions and language objects.

Usage

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

Arguments

x	An 'ENAplot' object or an ENA set containing plots.
wh	Specifies the trajectory to plot. Can be an unevaluated expression or a language object.
...	Additional parameters passed to the plotting functions.
name	A character string specifying the name of the plot. Default is "plot".

Details

The function determines the type of the 'wh' parameter and processes it accordingly: - If 'wh' is an unevaluated expression, it is captured and evaluated in the parent frame. - If 'wh' is a language object, it is processed to extract the relevant trajectory information.

The function updates the plot with the new trajectory and stores the updated plot back in the ENA set.

Value

Invisibly returns the modified ENA set.

Examples

```
# Load test data
data(RS.data);

# Define codenames for the test
codenames <- c("Data", "Technical.Constraints", "Performance.Parameters",
  "Client.and.Consultant.Requests", "Design.Reasoning", "Collaboration");

# Standard ENA accumulation
accum <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames
);
# Create a standard ENA set
newset <- ena.make.set(accum);

# Simple plot for a set of points, and their mean
newplot <- plot(newset) |>
  add_points(Condition$FirstGame, mean = TRUE, colors = "blue") |>
  add_network();

# Plot a single unit's point and it's line.weights
plot(newset) |>
  add_points(UserName$`steven z`) |>
  add_network();
```

```
# Trajectory accumulation and plotting
trajectory_accumulation <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames,
  model = "A"
);
trajectory_model <- ena.make.set(trajectory_accumulation);

newplot3 <- plot(trajectory_model) |>
  add_trajectory(Condition$FirstGame);
```

`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 *ENA nodes as matrix*

Description

ENA nodes as matrix

Usage

```
## 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 *ENA points as matrix*

Description

ENA points as matrix

Usage

```
## 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.ena.rotation.matrix

ENA rotations as matrix

Description

ENA rotations as matrix

Usage

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

Arguments

x	ena.rotation.matrix to conver to matrix
...	additional arguments to be passed to or from methods

Value

matrix

as.matrix.row.connections

ENA row connections as matrix

Description

ENA row connections as matrix

Usage

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

Arguments

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

Value

matrix

as.qe.code *Convert a vector to 'qe.code' class*

Description

This function converts a vector to the 'qe.code' class. If the vector is a factor, it is first converted to a character vector.

Usage

```
as.qe.code(x)
```

Arguments

x	A vector. The vector to be converted to 'qe.code' class.

Value

The modified vector with the 'qe.code' class.

Examples

```
vec <- factor(c("A", "B", "C"))
vec <- as.qe.code(vec)
class(vec) # Should show 'qe.code' along with other classes
```

as.qe.data *Convert an object to 'qe.data' class*

Description

This function converts an object to the 'qe.data' class. If the object is not a data.frame or matrix, it is first converted to a data.table.

Usage

```
as.qe.data(x)
```

Arguments

x	An object. The object to be converted to 'qe.data' class.

Value

The modified object with the 'qe.data' class.

Examples

```
library(data.table)

dt <- data.table(
  ID = 1:5,
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45),
  Score = c(85, 90, 95, 80, 75)
)
dt <- as.qe.data(dt);
class(dt) # Should show 'qe.data' along with other classes
```

---

as.qe.horizon *Convert a vector to 'qe.horizon' class*

Description

This function converts a vector to the 'qe.horizon' class. If the vector is a factor, it is first converted to a character vector.

Usage

```
as.qe.horizon(x)
```

Arguments

x	A vector. The vector to be converted to 'qe.horizon' class.

Value

The modified vector with the 'qe.horizon' class.

Examples

```
vec <- factor(c("A", "B", "C"))
vec <- as.qe.horizon(vec)
class(vec) # Should show 'qe.horizon' along with other classes
```

as.qe.metadata *Convert a vector to 'qe.metadata' class*

Description

This function converts a vector to the 'qe.metadata' class. If the vector is a factor, it is first converted to a character vector.

Usage

```
as.qe.metadata(x)
```

Arguments

x	A vector. The vector to be converted to 'qe.metadata' class.

Value

The modified vector with the 'qe.metadata' class.

Examples

```
vec <- factor(c("A", "B", "C"))
vec <- as.qe.metadata(vec)
class(vec) # Should show 'qe.metadata' along with other classes
```

as.qe.unit *Convert a vector to 'qe.unit' class*

Description

This function converts a vector to the 'qe.unit' class. If the vector is a factor, it is first converted to a character vector.

Usage

```
as.qe.unit(x)
```

Arguments

x	A vector. The vector to be converted to 'qe.unit' class.

Value

The modified vector with the 'qe.unit' class.

Examples

```
vec <- factor(c("A", "B", "C"))
vec <- as.qe.unit(vec)
class(vec) # Should show 'qe.unit' along with other classes
```

`as_trajectory` *Title*

Description

Title

Usage

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

Arguments

`x`	[TBD]
`by`	[TBD]
`model`	[TBD]
`...`	[TBD]

Value

[TBD]

`center` *Center ENA Data*

Description

This function centers the line weights of an ‘ena.set‘ by subtracting the mean of each connection from all units. This is a standard step in preparing data for rotation.

Usage

```
center(x, add.meta = TRUE)
```

Arguments

x	An ‘ena.set‘ object (typically after ‘sphere_norm()‘) or a numeric matrix.
add.meta	A logical value. If ‘TRUE‘ (the default), metadata is preserved. Ignored if ‘x‘ is a matrix.

Value

If ‘x‘ is an ‘ena.set‘, it returns the modified ‘ena.set‘ with the centered data stored in ‘x$model$points.for.projection‘. If ‘x‘ is a matrix, it returns a centered matrix.

Examples

```
data(RS.data)

codes <- c("Data", "Technical.Constraints", "Performance.Parameters",
           "Client.and.Consultant.Requests", "Design.Reasoning",
           "Collaboration")
units <- c("Condition", "UserName")
horizon <- c("Condition", "GroupName")
enaset <- RS.data |>
  accumulate(units, codes, horizon) |>
  sphere_norm() |>
  center()
```

---

check_range *Updates the axis ranges of an ENA plot based on the plotted data.*

Description

This function adjusts the x and y axis ranges of the ENA plot to ensure that all plotted points, networks, and means are visible.

Usage

```
check_range(x)
```

Arguments

x	An ENA plot object containing the plotted data and axis configurations.

Value

The updated ENA plot object with adjusted axis ranges.

clear *Clears specified plots from an ENA set.*

Description

This function removes the plots specified by their indices from the ‘plots‘ field of the ENA set.

Usage

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

Arguments

x	An ENA set object containing the plots.
wh	A numeric vector specifying the indices of the plots to clear. Default is all plots.

Value

Invisibly returns the modified ENA set object with the specified plots removed.

Examples

```
# Load test data
data(RS.data);

# Define codenames for the test
codenames <- c("Data", "Technical.Constraints", "Performance.Parameters",
  "Client.and.Consultant.Requests", "Design.Reasoning", "Collaboration");

# Standard ENA accumulation
accum <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames
);
# Create a standard ENA set
newset <- ena.make.set(accum);

# Simple plot for a set of points, and their mean
newplot <- plot(newset) |>
  add_points(Condition$FirstGame, mean = TRUE, colors = "blue") |>
  add_network();

# Plot a single unit's point and it's line.weights
plot(newset) |>
  add_points(UserName$`steven z`) |>
  add_network();

# Trajectory accumulation and plotting
```

```
trajectory_accumulation <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames,
  model = "A"
);
trajectory_model <- ena.make.set(trajectory_accumulation);

newplot3 <- plot(trajectory_model) |>
  add_trajectory(Condition$FirstGame);
```

`codes` *Reclassify specified columns as codes or list codes columns in a data.table*

Description

This function reclassifies specified columns of a data.table to the 'qe.code' format if column names are provided. If no column names are provided, it returns the names of columns that are already classified as 'qe.code'.

Usage

```
codes(x, ...)
```

Arguments

`x`	A data.table. The data.table containing the columns to be reclassified or checked.
`...`	Additional arguments specifying the names of the columns to be reclassified.

Value

The modified data.table with specified columns reclassified as 'qe.code', or a character vector of column names already classified as 'qe.unit'.

Examples

```
library(data.table)
dt <- data.table(a = 1:5, b = 6:10)
# Reclassify columns 'a' and 'b' as 'qe.code'
dt <- codes(dt, "a", "b")
# List columns classified as 'qe.code'
code_columns <- codes(dt)
```

combn_c2 *Fast combn choose 2*

Description

faster combn alternative

Usage

```
combn_c2(n)
```

Arguments

n	TBD

compute_SB *Compute Between-Group Scatter Matrix*

Description

This function calculates the between-group scatter matrix (SB) for a given numeric matrix and grouping variable.

Usage

```
compute_SB(A, g)
```

Arguments

A	A numeric matrix of dimensions m x n, where rows represent observations and columns represent features.
g	A grouping variable of length m, either a factor or a character vector, indicating group membership for each observation.

Details

The function computes the total mean of the matrix A and the mean for each group defined by g. It then calculates the between-group scatter matrix by summing the outer product of the mean differences, weighted by the group sizes.

Value

A numeric matrix representing the between-group scatter matrix (SB).

Examples

```
# Example usage:
A <- matrix(rnorm(20), nrow = 5, ncol = 4)
g <- factor(c("A", "B", "A", "B", "A"))
SB <- rENA:::compute_SB(A, g)
```

`connection.matrix` *Connection counts as square matrix*

Description

Connection counts as square matrix

Usage

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

Arguments

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

Value

matrix

`define` *Apply metadata and code transformations to a data.table*

Description

This function applies metadata and code transformations to a data.table if provided. It checks if the metadata and codes are supplied as vectors of column names.

Usage

```
define(
  x,
  metadata_cols = find_meta_cols(x),
  codes_cols = find_binary_cols(x),
  horizon_cols = NULL,
  units_cols = NULL
)
```

Arguments

x	A data.table. The data.table to be transformed.
metadata_cols	A vector of column names or NULL. A vector specifying the columns for metadata transformations.
codes_cols	A vector of column names or NULL. A vector specifying the columns for code transformations.
horizon_cols	A vector of column names or NULL. A vector specifying the columns for horizon transformations.
units_cols	A vector of column names or NULL. A vector specifying the columns for unit transformations.

Value

The modified data.table after applying the metadata and code transformations.

Examples

```
library(data.table)
dt <- data.table(a = 1:5, b = 6:10)
dt <- define(dt, metadata = c("a"), codes = c("b"))
```

---

directed_node_positions

Multiobjective, Component by Component, with Ellipsoidal Scaling, for directed ENA

Description

TBD

Usage

```
directed_node_positions(line_weights, points, numDims)
```

Arguments

line_weights	TBD
points	TBD
numDims	TBD

Details

Multiobjective, Component by Component, with Ellipsoidal Scaling, for directed ENA

directed_node_positions_with_ground_response_added

Node position optimization with ground and response weights/points added

Description

TBD

Usage

```
directed_node_positions_with_ground_response_added(
  line_weights,
  points,
  numDims
)
```

Arguments

line_weights	TBD
points	TBD
numDims	TBD

Details

Node position optimization with ground and response weights/points added

ena

Wrapper to generate, and optionally plot, an ENA model

Description

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

Usage

```
ena(
  data,
  codes,
  units,
  conversation,
  metadata = NULL,
  model = c("EndPoint", "AccumulatedTrajectory", "SeparateTrajectory"),
  weight.by = "binary",
```

```
  window = c("MovingStanzaWindow", "Conversation"),
  window.size.back = 1,
  include.meta = TRUE,
  groupVar = NULL,
  groups = NULL,
  runTest = FALSE,
  points = FALSE,
  mean = FALSE,
  network = TRUE,
  networkMultiplier = 1,
  subtractionMultiplier = 1,
  unit = NULL,
  include.plots = T,
  print.plots = F,
  ...
)
```

Arguments

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

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

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

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

include.plots
: logical, TRUE will generate plots based on the model (default: TRUE)

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

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

Details

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

Value

ena.set object

Examples

```
data(RS.data)

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

`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,
  as.list = T,
  ...
)
```

Arguments

`units`	A data frame where the columns are the properties by which units will be identified
`conversation`	A data frame where the columns are the properties by which conversations will be identified
`codes`	A data frame where the columns are the codes used to create adjacency (co-occurrence) vectors
`metadata`	(optional) A data frame with additional columns of metadata to be associated with each unit in the data
`model`	A character, choices: EndPoint (or E), AccumulatedTrajectory (or A), or SeparateTrajectory (or S); default: EndPoint. Determines the ENA model to be constructed
`weight.by`	(optional) A function to apply to values after accumulation
`window`	A character, choices are Conversation (or C), MovingStanzaWindow (MSW, MS); default MovingStanzaWindow. Determines how stanzas are constructed, which defines how co-occurrences are modeled

`window.size.back`
: A positive integer, Inf, or character (INF or Infinite), default: 1. Determines, for each line in the data frame, the number of previous lines in a conversation to include in the stanza window, which defines how co-occurrences are modeled

`window.size.forward`
: (optional) A positive integer, Inf, or character (INF or Infinite), default: 0. Determines, for each line in the data frame, the number of subsequent lines in a conversation to include in the stanza window, which defines how co-occurrences are modeled

`mask`
: (optional) A binary matrix of size ncol(codes) x ncol(codes). 0s in the mask matrix row i column j indicates that co-occurrence will not be modeled between code i and code j

`include.meta`
: Locigal indicating if unit metadata should be attached to the resulting ENAdata object, default is TRUE

`as.list`
: R6 objects will be deprecated, but if this is TRUE, the original R6 object will be returned, otherwise a list with class 'ena.set'

`...`
: additional parameters addressed in inner function

Details

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

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

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

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

Value

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

See Also

`ENAdata`, `ena.make.set`

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$meta.data$Condition=="FirstGame",
                         accum$meta.data$Condition=="SecondGame")
);
ena.conversations(set = RS.data,
  units = c("FirstGame.steven z"), units.by=c("Condition","UserName"),
  conversation.by = c("Condition","GroupName"),
  codes=codeNames, window = 4
)
```

`ena.correlations` *Calculate the correlations*

Description

Calculate both Spearman and Pearson correlations for the provided ENAset

Usage

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

Arguments

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

Value

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

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

Description

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

Usage

```
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 groups 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)

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, "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 = 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

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)

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$meta.data$Condition=="FirstGame",
      accum$meta.data$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("", ""),
  font.size = 10,
  font.color = "#000000",
  font.family = c("Arial", "Courier New", "Times New Roman"),
  scale.to = "network",
  ...
)
```

Arguments

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

Details

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

Value

ENAplot used for plotting an ENAset

See Also

ena.make.set, ena.plot.points

Examples

```
data(RS.data)

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 = default.colors[1],
```

```
  shape = c("square", "triangle-up", "diamond", "circle"),
  confidence.interval = c("none", "crosshairs", "box"),
  outlier.interval = c("none", "crosshairs", "box"),
  label.offset = "bottom right",
  label.font.size = NULL,
  label.font.color = NULL,
  label.font.family = NULL,
  show.legend = T,
  legend.name = NULL,
  ...
)
```

Arguments

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
...	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$meta.data$Condition=="FirstGame",
      accum$meta.data$Condition=="SecondGame"
  )
)

plot = ena.plot(set)

unitNames = set$enadata$units

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

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

print(plot);
```

---

ena.plot.network *Plot an ENA network*

Description

Plot an ENA network: nodes and edges

Usage

```
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$line.weights
node.positions	matrix containing the positiions of the nodes. Defaults to enaplot$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

`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)

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$meta.data$Condition=="FirstGame",
    accum$meta.data$Condition=="SecondGame"
  )
)

plot = ena.plot(set)

### Subset rotated points and plot Condition 1 Group Mean
as.matrix(set$points$Condition$FirstGame)

first.game.points = as.matrix(set$points$Condition$FirstGame)
plot = ena.plot.group(plot, first.game.points, labels = "FirstGame",
    colors = "red", confidence.interval = "box")

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

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

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

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

`ena.plot.points` *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 = 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	ENAplot 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

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

```
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$meta.data$Condition=="FirstGame",
      accum$meta.data$Condition=="SecondGame"
```

```
  )
)

plot = ena.plot(set)

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

`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)

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);

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

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

```
second.game.mean = colMeans(second.game.lineweights)

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

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

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

`ena.plotter` *Wrapper to generate plots of units, groups, and networks*

Description

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

Usage

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

Arguments

`set`	an ena.set object
`groupVar`	vector, character, of column name containing group identifiers.

groups	vector, character, of values of groupVar column you wish to plot. Maxium of two groups allowed.
points	logical, TRUE will plot points (default: FALSE)
mean	logical, TRUE will plot the mean position of the groups defined in the groups argument (default: FALSE)
network	logical, TRUE will plot networks (default: TRUE)
networkMultiplier	numeric, scaling factor for non-subtracted networks (default: 1)
subtractionMultiplier	numeric, scaling factor for subtracted networks (default: 1)
unit	vector, character, name of a single unit to plot
print.plots	logical, TRUE will show plots in the Viewer (default: FALSE)
...	Additional parameters passed to set creation and plotting functions

Details

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

Value

ena.set object

`ena.rotate.by.generalized`

ENA Rotate by generalized means rotation (gmr)

Description

ENA Rotate by generalized means rotation (gmr)

Usage

```
ena.rotate.by.generalized(enaset, params)
```

Arguments

enaset	An ENAset
params	list of parameters, may include: x_var: data.frame used for calling gmr() on the first dimension y_var: data.frame used for calling gmr() on the second dimension (optional).

Value

ENARotationSet

ena.rotate.by.hena.regression

ENA Rotate by regression

Description

This function allows user to provide a regression formula for rotation on x and optionally on y. If regression formula for y is not provide, svd is applied to the residual data deflated by x to get y coordinates. The regression formula uses ENA dimensions are dependent variables. The first predictor has to be two-group categorical, binary, or numerical.

Usage

```
ena.rotate.by.hena.regression(enaset, params)
```

Arguments

enaset	An ENAset
params	list of parameters, may include: x_var: Regression formula for x direction, such as "lm(formula=V ~ Condition + GameHalf + Condition : GameHalf)", where V always stands for the ENA points. y_var: Regression formula, similar to x_var, for y direction (optional).

Value

ENARotationSet

ena.rotate.by.hena.regression_2

ENA Rotate by regression (second way)

Description

This function allows user to provide a regression formula for rotation on x and optionally on y. If regression formula for y is not provide, svd is applied to the residual data deflated by x to get y coordinates. The regression formula should use ENA points as major predictors and a binary or numerical variable as dependent variable. Control and interaction variables are allowed to be included as predictors in the formula.

Usage

```
ena.rotate.by.hena.regression_2(enaset, params)
```

Arguments

enaset	An ENAset
params	list of parameters, may include: x_var: Regression formula for x direction, such as "lm(formula= Condition ~ V + GameHalf + Condition : GameHalf)", where V always stands for the ENA points. y_var: Regression formula, similar to x_var for y direction (optional).

Value

ENARotationSet

---

ena.rotate.by.mean *ENA Rotate by mean*

Description

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

Usage

```
ena.rotate.by.mean(enaset, groups = NULL, params = 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
params	A list containing two vectors of length nrow(ENA.set$ena.data$units),

Value

ENARotationSet

ena.rotation.h *hENA rotation for ENA*

Description

hENA rotation function.

Usage

```
ena.rotation.h(enaset, params)
```

Arguments

enaset	ena set
params	list of parameters

Value

ena set

ena.set.creator *Wrapper to generate an ENA model*

Description

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

Usage

```
ena.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

Computes a dimensional reduction of points in an ENA set using Singular Value Decomposition (SVD).

Usage

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

Arguments

enaset	An ENAset object containing the points to be reduced.
params	A list of parameters. Use params$as_object = TRUE to return an ENARotationSet object, or FALSE (default) to return a list.

Details

This function computes the SVD of the points in the ENA set and returns either an ENARotationSet object or a list with the rotation matrix, codes, node positions, and eigenvalues, depending on params$as_object.

Value

An ENARotationSet object or a list containing:

rotation	The rotation matrix from SVD
codes	The code names used for the matrix
node.positions	(Currently NULL) Node positions
eigenvalues	The eigenvalues (squared singular values) from SVD

Examples

```
data(RS.data)
codes <- c("Data", "Technical.Constraints", "Performance.Parameters",
           "Client.and.Consultant.Requests", "Design.Reasoning",
           "Collaboration")
units <- c("Condition", "UserName")
horizon <- c("Condition", "GroupName")
enaset <- RS.data |>
  accumulate(units, codes, horizon) |>
  model()
# SVD as list:
svd_result <- ena.svd(enaset, list(as_object = FALSE))
# SVD as ENARotationSet object:
svd_obj <- ena.svd(enaset, list(as_object = TRUE))
```

`ena.writeup` *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

`ena_correlation` *Calculate the correlations*

Description

Calculate both Pearson correlations for the provided points and centorids

Usage

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

Arguments

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

`ENAdata` *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 Construct ENAdata

Methods

Public methods:

- `ENAdata$new()`
- `ENAdata$process()`
- `ENAdata$get()`
- `ENAdata$add.metadata()`
- `ENAdata$clone()`

Method `new()`:

Usage:

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

Arguments:

`file` TBD

`units` TBD

`units.used` TBD

`units.by` TBD

`conversations.by` TBD

`codes` TBD

`model` TBD

`weight.by` TBD

`window.size.back` TBD

`window.size.forward` TBD

`mask` TBD

`include.meta` TBD

... TBD

Returns: Process accumulation

Method `process()`**:**

Usage:

```
ENAdata$process()
```

Returns: ENAdata Get property from object

Method `get()`**:**

Usage:

```
ENAdata$get(x = "data")
```

Arguments:

`x` character key to retrieve from object

Returns: value from object at x Add metadata

Method `add.metadata()`**:**

Usage:

```
ENAdata$add.metadata(merge = F)
```

Arguments:

`merge` logical (default: FALSE)

Returns: data.frame

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

Usage:

```
ENAdata$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ENAplot` *ENAplot Class*

Description

The ENAplot R6 class provides a structure for visualizing ENAset objects using plotly. It encapsulates the ENAset data, the plotly visualization, and related plotting parameters.

Fields

enaset The `ENAset` object from which the ENAplot was constructed.

plot The plotly object used for data visualization.

axes Axes information for the plot (TBD).

point Point information for the plot (TBD).

palette Color palette used for plotting (TBD).

plotted Indicates whether the plot has been rendered (TBD).

Public fields

`enaset` - The `ENAset` object from which the ENAplot was constructed

`plot` - The plotly object used for data visualization

`axes` A list or object specifying the axes configuration for the ENA plot, such as axis labels, limits, or scaling.

`point` A structure representing the data points to be plotted, including coordinates and visual properties.

`palette` A set of colors or a function defining the color scheme used for plotting elements in the ENA plot.

`plotted` A logical or status indicator showing whether the plot has been rendered or updated.

`showticklabels` Logical. Indicates whether to show tick labels on the axes.

`autosize` Logical. Indicates whether the plot should automatically resize.

`automargin` Logical. Indicates whether the plot should automatically adjust margins.

`axispadding` Numeric. Padding factor for the axes. Create ENApolot

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` An ENA set object containing the data to be plotted.

`title` The title of the plot.

`dimension.labels` Labels for the dimensions shown in the plot.

`font.size` Numeric value specifying the font size for plot text.

`font.color` Color value for the plot text.

`font.family` Font family to use for plot text.

`scale.to` Numeric value to scale the plot axes.

... Additional arguments passed to the plotting function. #'
showticklabels Logical; whether to display axis tick labels.
autosize Logical; whether the plot should automatically size itself.
automargin Logical; whether the plot should automatically adjust margins.
axispadding Numeric value specifying padding around axes.

Returns: ENAplot Print ENA plot

Method `print()`**:**

Usage:

```
ENAplot$print()
```

Returns: Get property from object

Method `get()`**:**

Usage:

```
ENAplot$get(x)
```

Arguments:

x character key to retrieve from object

Returns: value from object at x

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

Usage:

```
ENAplot$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
# Example usage:
# enaplot <- ENAplot$new(enaset = myENAset)
```

`ENARotationSet` *ENARotationSet R6class*

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.

`ENAset` *ENAset R6class*

Description

ENAset R6class

ENAset R6class

Public fields

`enadata` An ENAdata object originally used to create the set

`points.raw` A data frame containing accumulated adjacency (co-occurrence) vectors per unit

`points.normed.centered` A data frame of centered normed accumulated adjacency (co-occurrence) vectors for each unit

`points.rotated` A data frame of point positions for number of dimensions specified in ena.make.set (i.e., the centered, normed, and rotated data)

`line.weights` A data frame of connections strengths per unit (Data frame of normed accumulated adjacency (co-occurrence) vectors for each unit)

`node.positions` - A data frame of positions for each code

`codes` - A vector of code names

rotation.set - An ENARotationSet object

variance - A vector of variance accounted for by each dimension specified

centroids - A matrix of the calculated centroid positions

function.call - The string representation of function called

function.params - A list of all parameters sent to function call

rotation_dists TBD

points.rotated.scaled TBD

points.rotated.non.zero TBD

line.weights.unrotated TBD

line.weights.non.zero TBD

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

center.align.to.origin - align point and centroid centers to origin Create ENAset

Methods

Public methods:

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

Method new():

Usage:

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

```
ENAset$process()
```

Returns: ENASet Get property from object

Method `get()`**:**

Usage:

```
ENAset$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:

```
ENAset$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`find_binary_cols` *Find Binary Columns*

Description

Identifies columns in a data.frame or data.table that are binary (i.e., contain only two unique values), optionally including logical columns.

Usage

```
find_binary_cols(x, include_logical = FALSE)
```

Arguments

x	A data.frame or data.table to search for binary columns.
include_logical	Logical. If TRUE, logical columns are also considered binary. Default is FALSE.

Value

A character vector of column names that are binary, or NULL if none are found.

Examples

```
df <- data.frame(a = c(0, 1, 1), b = c(TRUE, FALSE, TRUE), c = c(1, 2, 3))
find_binary_cols(df)
find_binary_cols(df, include_logical = TRUE)
```

`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 *Row-wise Max-Norm Scaling*

Description

Scales all rows of a numeric dataframe by dividing by the largest row vector length (L2 norm) found in the dataframe. This preserves the relative magnitudes between rows but does not normalize each row to unit length. Useful for analyses where relative scale is important but full normalization is not desired.

Usage

```
fun_skip_sphere_norm(dfM)
```

Arguments

dfM	A data.frame or matrix. Each row is treated as a vector to compute its L2 norm.

Details

Row-wise Max-Norm Scaling

This function finds the row with the largest L2 norm (Euclidean length) and divides all entries in the matrix by this value. It does not normalize each row individually.

Value

A numeric matrix with the same dimensions as 'dfM', with all values divided by the largest row L2 norm.

Examples

```
df <- data.frame(a = c(3, 4), b = c(0, 0))
fun_skip_sphere_norm(df)
```

fun_sphere_norm *Row-wise L2 (Sphere) Normalization*

Description

Normalizes each row of a numeric dataframe or matrix to have unit L2 norm (Euclidean length). Each row is divided by its own length, projecting all rows onto the unit hypersphere. Useful for analyses where direction is important but magnitude should be removed.

Usage

```
fun_sphere_norm(dfM)
```

Arguments

dfM	A data.frame or matrix. Each row is treated as a vector to compute its L2 norm.

Details

Row-wise L2 (Sphere) Normalization

This function computes the L2 norm (Euclidean length) of each row and divides the row by this value. Rows with zero length are left unchanged.

Value

A numeric matrix with the same dimensions as 'dfM', with each row normalized to unit length (L2 norm = 1), unless the row is all zeros (in which case it remains zeros).

Examples

```
df <- data.frame(a = c(3, 4), b = c(0, 0))
fun_sphere_norm(df)
```

---

get_x1_main_effect *Extract Main Effect Contribution of the First Predictor Using Elastic Net*

Description

This function fits a multivariate elastic net regression model (using `glmnet`) to estimate the main effect contribution of the first predictor variable (`x1`) in the provided data frame `X` for each response variable in `V`. The main effect is forced into the model by setting its penalty factor to zero.

Usage

```
get_x1_main_effect(V, X, alpha = 1, lambda = "lambda.min")
```

Arguments

V	A numeric matrix of response variables (dimensions: observations x responses).
X	A data frame containing all predictor variables. The first column is treated as the target predictor (`x1`).
alpha	Elastic net mixing parameter (0 = ridge, 1 = lasso). Default is 1.
lambda	The value of the regularization parameter to use. Can be `"lambda.min"`, `"lambda.1se"`, or a specific numeric value. Default is `"lambda.min"`.

Details

The function constructs a model matrix including all main effects and two-way interactions. It then fits a multivariate elastic net model using `cv.glmnet`, forcing the inclusion of the main effect of `x1` by setting its penalty factor to zero. The resulting coefficients for `x1` are used to compute its contribution to the fitted values for each response variable.

Value

A numeric matrix of the same dimensions as V, where each column contains the estimated main effect contribution of x1 for the corresponding response variable.

Examples

```
## Not run:
set.seed(123)
V <- matrix(rnorm(100 * 2), ncol = 2)
X <- data.frame(x1 = rnorm(100), x2 = sample(letters[1:3], 100, TRUE))
result <- get_x1_main_effect(V, X)

## End(Not run)
```

---

gmr *Generalized Means Rotation (GMR)*

Description

Computes the generalized means rotation for a given set of ENA points and predictor variables.

Usage

```
gmr(V, X)
```

Arguments

V	A matrix containing ENA set points for projection.
X	A data frame containing all predictor variables, with the first column as the target variable.

Details

If X has only one column, a linear model is fit between V and the single predictor. Otherwise, the main effect of the first predictor is extracted using `get_x1_main_effect`. Singular value decomposition (SVD) is then performed, and the first right singular vector is used to project the data. A linear model is fit to the projected data, and the coefficients are normalized to produce the rotation vector.

Value

A numeric vector representing the rotation.

See Also

`get_x1_main_effect`

Examples

```
## Not run:
V <- matrix(rnorm(100), ncol = 5)
X <- data.frame(target = rnorm(20), predictor1 = rnorm(20), predictor2 = rnorm(20))
r <- gmr(V, X)

## End(Not run)
```

---

group *Add all groups to an ENA plot*

Description

This function iterates over all unique values of the first metadata column (excluding 'QEUNIT' and 'ENA_UNIT') in the ENA set and adds each group as a set of points to the ENA plot. This is useful for quickly visualizing all groups in a categorical variable on the same plot.

Usage

```
group(x, wh = NULL)
```

Arguments

x	An 'ENAplot' object (as returned by 'plot.ena.set').
wh	(Ignored) Included for compatibility with other plotting functions.

Details

The function finds the first metadata column in the ENA set (excluding 'QEUNIT' and 'ENA_UNIT'), and for each unique value in that column, calls 'add_points()' to add the group's points to the plot.

Value

The modified 'ENAplot' object with all groups added as points.

Examples

```
# Load test data
data(RS.data);

# Define codenames for the test
codenames <- c("Data", "Technical.Constraints", "Performance.Parameters",
  "Client.and.Consultant.Requests", "Design.Reasoning", "Collaboration");

# Standard ENA accumulation
accum <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
```

```
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames
);
# Create a standard ENA set
newset <- ena.make.set(accum);

# Simple plot for a set of points, and their mean
newplot <- plot(newset) |>
  add_points(Condition$FirstGame, mean = TRUE, colors = "blue") |>
  add_network();

# Plot a single unit's point and it's line.weights
plot(newset) |>
  add_points(UserName$`steven z`) |>
  add_network();

# Trajectory accumulation and plotting
trajectory_accumulation <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames,
  model = "A"
);
trajectory_model <- ena.make.set(trajectory_accumulation);

newplot3 <- plot(trajectory_model) |>
  add_trajectory(Condition$FirstGame);
```

`horizon` *Reclassify specified columns as horizon or list horizon columns in a data.table*

Description

This function reclassifies specified columns of a data.table to the 'qe.horizon' format if column names are provided. If no column names are provided, it returns the names of columns that are already classified as 'qe.horizon'.

Usage

```
horizon(x, ...)
```

Arguments

`x`	A data.table. The data.table containing the columns to be reclassified or checked.
`...`	Additional arguments specifying the names of the columns to be reclassified.

Value

The modified data.table with specified columns reclassified as 'qe.horizon', or a character vector of column names already classified as 'qe.horizon'.

Examples

```
library(data.table)
dt <- data.table(a = 1:5, b = 6:10)
# Reclassify columns 'a' and 'b' as 'qe.horizon'
dt <- horizon(dt, "a", "b")
# List columns classified as 'qe.horizon'
horizon_columns <- horizon(dt)
```

---

is.qe.code *Check if an object is of class 'qe.code'*

Description

This function checks if an object is of class 'qe.code'.

Usage

```
is.qe.code(x)
```

Arguments

x	An object. The object to be checked.

Value

A logical value. TRUE if the object is of class 'qe.code', otherwise FALSE.

Examples

```
dt <- 1:5
class(dt) <- c("qe.code", class(dt))
is.qe.code(dt) # Should return TRUE
```

---

is.qe.data *Check if an object is of class 'qe.data'*

Description

This function checks if an object is of class 'qe.data'.

Usage

```
is.qe.data(x)
```

Arguments

x	An object. The object to be checked.

Value

A logical value. TRUE if the object is of class 'qe.data', otherwise FALSE.

Examples

```
library(data.table)

dt <- data.table(ID = 1:5)
class(dt) <- c("qe.data", class(dt))
is.qe.data(dt) # Should return TRUE
```

---

is.qe.horizon *Check if an object is of class 'qe.horizon'*

Description

This function checks if an object is of class 'qe.horizon'.

Usage

```
is.qe.horizon(x)
```

Arguments

x	An object. The object to be checked.

Value

A logical value. TRUE if the object is of class 'qe.horizon', otherwise FALSE.

Examples

```
dt <- 1:5
class(dt) <- c("qe.horizon", class(dt))
is.qe.horizon(dt) # Should return TRUE
```

is.qe.metadata *Check if an object is of class 'qe.metadata'*

Description

This function checks if an object is of class 'qe.metadata'.

Usage

```
is.qe.metadata(x)
```

Arguments

x	An object. The object to be checked.

Value

A logical value. TRUE if the object is of class 'qe.metadata', otherwise FALSE.

Examples

```
dt <- 1:5
class(dt) <- c("qe.metadata", class(dt))
is.qe.metadata(dt) # Should return TRUE
```

is.qe.unit *Check if an object is of class 'qe.unit'*

Description

This function checks if an object is of class 'qe.unit'.

Usage

```
is.qe.unit(x)
```

Arguments

x	An object. The object to be checked.

Value

A logical value. TRUE if the object is of class 'qe.unit', otherwise FALSE.

Examples

```
dt <- 1:5
class(dt) <- c("qe.unit", class(dt))
is.qe.unit(dt) # Should return TRUE
```

means_rotate *Title*

Description

Title

Usage

```
means_rotate(x, on = NULL)
```

Arguments

x	[TBD]
on	[TBD]

Value

[TBD]

merge_columns_c *Merge data frame columns*

Description

TBD

Usage

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

Arguments

df	Dataframe
cols	Vector
sep	Character seperator

Details

Merge data frame columns

`metadata` *Reclassify specified columns as metadata or list metadata columns in a data.table*

Description

This function reclassifies specified columns of a data.table to the 'qe.metadata' format if column names are provided. If no column names are provided, it returns the names of columns that are already classified as 'qe.metadata'.

Usage

```
metadata(x, ...)
```

Arguments

`x`	A data.table. The data.table containing the columns to be reclassified or checked.
`...`	Additional arguments specifying the names of the columns to be reclassified.

Value

The modified data.table with specified columns reclassified as 'qe.metadata', or a character vector of column names already classified as 'qe.metadata'.

Examples

```
library(data.table)
dt <- data.table(a = 1:5, b = 6:10)
# Reclassify columns 'a' and 'b' as 'qe.metadata'
dt <- metadata(dt, "a", "b")
# List columns classified as 'qe.metadata'
metadata_columns <- metadata(dt)
```

`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]

---

methods_report_stream *methods_report_stream*

Description

Methods report for rmarkdwon

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]

---

model *Build a Complete ENA Model*

Description

This function applies a full ENA modeling pipeline to accumulated data. It is a convenience wrapper that chains together normalization, centering, rotation, projection, and optional optimization. Each step can be customized by supplying an alternative function.

Usage

```
model(
  data,
  ...,
  normalize = sphere_norm,
  center_with = center,
  rotate_with = rotate,
  project_with = project,
  optimize_with = optimize,
  rotate_fun = ena.rotate.by.generalized,
  rotate_params = list()
)
```

Arguments

data	An ‘ena.set‘ object, typically the result of ‘accumulate()‘.
...	Additional arguments passed to the rotation function specified by ‘rotate_fun‘.
normalize	A function to normalize the connection counts. Defaults to ‘sphere_norm‘.
center_with	A function to center the normalized data. Defaults to ‘center‘.
rotate_with	A function to perform the rotation (e.g., SVD). Defaults to ‘rotate‘.
project_with	A function to project the points into the rotated space. Defaults to ‘project‘.
optimize_with	A function to optimize node positions. Defaults to ‘optimize‘. Can be set to ‘NULL‘ or ‘FALSE‘ to skip.
rotate_fun	The specific rotation function to be used by ‘rotate_with‘. Defaults to ‘ena.rotate.by.generalized‘.
rotate_params	A list of additional parameters to pass to the ‘rotate_fun‘.

Value

An 'ena.set' object with a complete ENA model, including projected points and node positions.

Examples

```
data(RS.data)

codes <- c("Data", "Technical.Constraints", "Performance.Parameters",
           "Client.and.Consultant.Requests", "Design.Reasoning",
           "Collaboration")
units <- c("Condition", "UserName")
horizon <- c("Condition", "GroupName")
enaset <- RS.data |>
  accumulate(units, codes, horizon) |>
  model()
```

---

move_nodes_to_unit_circle

Title

Description

Title

Usage

```
move_nodes_to_unit_circle(
  set,
  dimension_name_1 = colnames(as.matrix(set$rotation$nodes))[1],
  dimension_name_2 = colnames(as.matrix(set$rotation$nodes))[2]
)
```

Arguments

set	TBD
dimension_name_1	TBD
dimension_name_2	TBD

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 = colnames(as.matrix(set$rotation$nodes))[1],
  dimension_name_2 = colnames(as.matrix(set$rotation$nodes))[2]
)
```

Arguments

set	TBD
dimension_name_1	TBD
dimension_name_2	TBD

Value

TBD

namesToAdjacencyKey *Names to Adjacency Key*

Description

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

Usage

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

Arguments

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

Details

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

optimize *Optimize Node and Centroid Positions in ENA Set*

Description

This function computes and assigns node positions and centroids for an ENA set object using the current points and rotation information.

Usage

```
optimize(x, weights = NULL)
```

Arguments

x	An ena.set object for which to optimize node and centroid positions.
weights	Optional. A numeric matrix of connection weights. If provided, the function will use this matrix instead of the connection counts from the ena.set.

Value

The input ena.set object with updated node and centroid positions.

Examples

```
# Assuming 'set' is an ena.set object:
data(RS.data)

codes <- c("Data", "Technical.Constraints", "Performance.Parameters",
           "Client.and.Consultant.Requests", "Design.Reasoning",
           "Collaboration")
units <- c("Condition", "UserName")
horizon <- c("Condition", "GroupName")
enaset <- RS.data |>
  accumulate(units, codes, horizon) |>
  sphere_norm() |>
  center() |>
  rotate() |>
  project() |>
  optimize()
```

plot.ena.set *Plot an ena.set object*

Description

Plot an ena.set object

Usage

```
## S3 method for class 'ena.set'
plot(x, y, ..., empty = TRUE, title = "ENA Plot")
```

Arguments

x	ena.set to plot
y	ignored.
...	Additional parameters passed along to ena.plot functions
empty	Logical; if TRUE, creates an empty plot without points. Default is TRUE.
title	Character; title for the plot. Default is "ENA Plot".

Value

ena.plot.object

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(set) |>
  add_points(Condition$FirstGame, colors = "blue", with.mean = TRUE) |>
  add_points(Condition$SecondGame, colors = "red", with.mean = TRUE) |>
  with_means() |>
  add_nodes()
```

```
myENAplot <- plot(set) |>
  add_network(Condition$FirstGame - Condition$SecondGame)


# Add a group mean to an existing ENA plot
add_group(myENAplot, wh = Condition$FirstGame)

# Add a trajectory to an existing ENA plot
add_trajectory(myENAplot, wh = Condition$FirstGame)

# Load test data
data(RS.data);

# Define codenames for the test
codenames <- c("Data", "Technical.Constraints", "Performance.Parameters",
  "Client.and.Consultant.Requests", "Design.Reasoning", "Collaboration");

# Standard ENA accumulation
accum <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames
);
# Create a standard ENA set
newset <- ena.make.set(accum);

# Simple plot for a set of points, and their mean
newplot <- plot(newset) |>
  add_points(Condition$FirstGame, mean = TRUE, colors = "blue") |>
  add_network();

# Plot a single unit's point and it's line.weights
plot(newset) |>
  add_points(UserName$`steven z`) |>
  add_network();

# Trajectory accumulation and plotting
trajectory_accumulation <- rENA:::ena.accumulate.data.file(
  RS.data, units.by = c("UserName", "Condition"),
  conversations.by = c("ActivityNumber", "GroupName"),
  codes = codenames,
  model = "A"
);
trajectory_model <- ena.make.set(trajectory_accumulation);

newplot3 <- plot(trajectory_model) |>
  add_trajectory(Condition$FirstGame);
```

`prepare_trajectory_data` *Prepares trajectory data for an ENA plot.*

Description

This function processes and prepares trajectory data for plotting in an ENA set. It handles rotation, grouping, and filling missing steps in the trajectory.

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`	An ENA set object. If 'NULL', other parameters must be provided.
`by`	A character vector specifying the grouping variables for the trajectory. Default is the first conversation parameter in the ENA set.
`rotation_matrix`	A matrix used to rotate the points. Default is the rotation matrix from the ENA set.
`points`	A data table of points to be processed. Default is the points from the ENA set.
`units`	A data table of units corresponding to the points. Default is the trajectories or points from the ENA set.
`units_by`	A character vector specifying the unit grouping variables. Default is the unit parameters from the ENA set.
`steps`	A data table specifying the steps for the trajectory. If 'NULL', steps are generated automatically.

Value

A data table containing the processed trajectory data, including dimensions and metadata.

print.ena.set *Title*

Description

Title

Usage

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

Arguments

x	[TBD]
...	[TBD]
plot	[TBD]
set	[TBD]

Value

[TBD]

project *Project ENA Points onto Rotated Space*

Description

This function projects ENA points onto the rotated space using the rotation matrix. Optionally, metadata can be included in the resulting points matrix.

Usage

```
project(x, rotation = NULL, add.meta = TRUE)
```

Arguments

x	An ena.set object containing the points for projection and rotation matrix.
rotation	Optional. A rotation matrix to use for projection if x is not an ena.set.
add.meta	Logical. If TRUE (default), metadata will be included in the output.

Value

The input ena.set object with the projected points matrix (and metadata if requested).

Examples

```
# Assuming 'set' is an ena.set object:
data(RS.data)

codes <- c("Data", "Technical.Constraints", "Performance.Parameters",
           "Client.and.Consultant.Requests", "Design.Reasoning",
           "Collaboration")
units <- c("Condition", "UserName")
horizon <- c("Condition", "GroupName")
enaset <- RS.data |>
  accumulate(units, codes, horizon) |>
  sphere_norm() |>
  center() |>
  rotate() |>
  project()
```

`project_in` *Title*

Description

Title

Usage

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

Arguments

`x`	[TBD]
`by`	[TBD]
`...`	[TBD]

Value

[TBD]

reclassify *Reclassify specified columns in a data.table*

Description

This function reclassifies specified columns of a data.table using a provided function.

Usage

```
reclassify(x, v, ...)
```

Arguments

x	A data.table. The data.table containing the columns to be reclassified.
v	A function. The function to apply to each specified column for reclassification.
...	Additional arguments specifying the names of the columns to be reclassified.

Value

The modified data.table with specified columns reclassified.

Examples

```
library(data.table)
dt <- data.table(a = 1:5, b = 6:10)
dt <- reclassify(dt, as.qe.code, "a", "b")
```

remove_meta_data *Remove meta columns from data.table*

Description

Remove meta columns from data.table

Usage

```
remove_meta_data(x)
```

Arguments

x	[TBD]

Value

data.table withe columns of class ena.meta.data removed

rENA *rENA creates ENA sets*

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

rotate *Rotate ENA Data*

Description

Rotates ENA data using a specified rotation function (default: SVD), optionally using formulas or grouping variables.

Usage

```
rotate(x, ..., wh = ena.rotate.by.generalized)
```

Arguments

x	An ena.set object to be rotated.
...	Optional formulas or additional arguments for rotation.
wh	Function to use for rotation (default: ena.svd).

Value

The rotated ena.set object with updated rotation matrices.

Examples

```
# Assuming 'set' is an ena.set object:
data(RS.data)

codes <- c("Data", "Technical.Constraints", "Performance.Parameters",
           "Client.and.Consultant.Requests", "Design.Reasoning",
           "Collaboration")
units <- c("Condition", "UserName")
horizon <- c("Condition", "GroupName")
enaset <- RS.data |>
  accumulate(units, codes, horizon) |>
  sphere_norm() |>
  center() |>
  rotate()
```

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.ENAplot *Scales the points in an ENA set.*

Description

This function adjusts the scale of the points in the ENA set to match the range of the network.

Usage

```
## S3 method for class 'ENAplot'
scale(x, center = NULL, scale = NULL)
```

Arguments

`x`	An ENAplot object containing the set to scale.
`center`	Unused parameter, included for compatibility.
`scale`	A numeric value specifying the scaling factor. If 'NULL', the function will determine the scale based on the data.

Value

The modified ENAplot object with scaled points.

show *Display and update plot objects within a custom object*

Description

This function updates the plots within the provided object by applying the ‘check_range‘ function to each plot. It then prints the updated object using custom print options and returns the object invisibly.

Usage

```
show(x, ...)
```

Arguments

x	An object containing a list of plots in the ‘plots‘ field.
...	Additional arguments passed to the ‘print‘ method.

Value

The updated object ‘x‘, returned invisibly.

sphere_norm *Apply Spherical Normalization to ENA Data*

Description

This function applies spherical normalization to the connection counts in an ‘ena.set‘ object or to a raw matrix of connection counts. Normalization is a key step before centering and rotation in ENA.

Usage

```
sphere_norm(x, add.meta = TRUE)
```

Arguments

x	An ‘ena.set‘ object or a numeric matrix of connection counts.
add.meta	A logical value. If ‘TRUE‘ (the default), metadata from the ‘ena.set‘ is preserved and included in the output. This parameter is ignored if ‘x‘ is a matrix.

Value

If ‘x‘ is an ‘ena.set‘, it returns the modified ‘ena.set‘ with a new ‘line.weights‘ matrix and an updated ‘centervec‘ in the ‘rotation‘ object. If ‘x‘ is a matrix, it returns a matrix of normalized line weights.

Examples

```
data(RS.data)

codes <- c("Data", "Technical.Constraints", "Performance.Parameters",
           "Client.and.Consultant.Requests", "Design.Reasoning",
           "Collaboration")
units <- c("Condition", "UserName")
horizon <- c("Condition", "GroupName")
enaset <- RS.data |>
  accumulate(units, codes, horizon) |>
  sphere_norm()
```

`units` *Reclassify specified columns as units or list unit columns in a data.table*

Description

This function reclassifies specified columns of a data.table to the 'qe.unit' format if column names are provided. If no column names are provided, it returns the names of columns that are already classified as 'qe.unit'.

Usage

```
units(x, ...)
```

Arguments

`x`	A data.table. The data.table containing the columns to be reclassified or checked.
`...`	Additional arguments specifying the names of the columns to be reclassified.

Value

The modified data.table with specified columns reclassified as 'qe.unit', or a character vector of column names already classified as 'qe.unit'.

Examples

```
library(data.table)
dt <- data.table(a = 1:5, b = 6:10)
# Reclassify columns 'a' and 'b' as 'qe.unit'
dt <- units(dt, "a", "b")
# List columns classified as 'qe.unit'
unit_columns <- units(dt)
```

vector_to_ut *vector to upper triangle*

Description

TBD

Usage

```
vector_to_ut(v)
```

Arguments

v	[TBD]

Details

Upper Triangle from Vector

with.ena.matrix *with.ena.matrix*

Description

This function sets up a context using the provided data (typically an ENA matrix), allowing the evaluation of an expression (‘expr‘) with access to both the matrix and its metadata. Optionally, a custom matrix ‘V‘ and other arguments can be supplied.

Usage

```
## S3 method for class 'ena.matrix'
with(data, expr, ...)
```

Arguments

data	An ENA matrix or data frame containing the data to be used.
expr	An R expression to be evaluated within the context of the ENA matrix.
...	Additional arguments, including an optional custom matrix ‘V‘ and other parameters.

Details

- If a custom matrix ‘V‘ is provided in ‘...‘, it will be used; otherwise, ‘data‘ is converted to a matrix. - Metadata columns are coerced to numeric if they are character vectors. - The expression is evaluated with access to both the matrix (‘V‘) and metadata.

Value

The result of evaluating ‘expr‘ in the constructed context.

`with_means` *Adds group means to the ENA plot.*

Description

This function iterates over the plotted points in the ENA plot and calculates the mean for each group of points. The calculated means are then added to the plot as group means.

Usage

```
with_means(x)
```

Arguments

`x`	An ENA set object containing the plots.

Value

Invisibly returns the modified ENA set object with updated plots.

`with_trajectory` *Adds trajectories to an ENA plot.*

Description

This function generates trajectories for the plotted points in the ENA plot based on the specified grouping variables. It supports options for jittering, animation, and scaling.

Usage

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

Arguments

x	An ENA set object containing the plots.
...	Additional arguments passed to the plotting functions.
by	A character vector specifying the grouping variables for the trajectories. Default is the first conversation parameter in the ENA set.
add_jitter	Logical; if 'TRUE', adds jitter to the trajectory points. Default is 'TRUE'.
frame	Numeric; the duration of each frame in the animation. Default is 1100.
transition	Numeric; the duration of the transition between frames. Default is 1000.
easing	A character string specifying the easing function for the animation. Default is "circle-in-out".

Value

Invisibly returns the modified ENA set object with updated plots.

Index

∗ **datasets**
RS.data, 91
*.ena.matrix, 4
$.ena.matrix, 5
$.ena.metadata, 5
$.ena.points, 6
$.line.weights, 6

accumulate, 7
add_group, 8
add_network, 9
add_nodes, 11
add_points, 12
add_trajectory, 13
as.ena.co.occurrence, 15
as.ena.matrix, 15
as.ena.metadata, 16
as.matrix.ena.connections, 16
as.matrix.ena.line.weights, 17
as.matrix.ena.matrix, 17
as.matrix.ena.nodes, 18
as.matrix.ena.points, 18
as.matrix.ena.rotation.matrix, 19
as.matrix.row.connections, 19
as.qe.code, 20
as.qe.data, 20
as.qe.horizon, 21
as.qe.metadata, 22
as.qe.unit, 22
as_trajectory, 23

center, 23
check_range, 24
clear, 25
codes, 26
combn_c2, 27
compute_SB, 27
connection.matrix, 28

define, 28
directed_node_positions, 29
directed_node_positions_with_ground_response_added, 30

ena, 30
ena.accumulate.data, 33, *39*
ena.conversations, 35
ena.correlations, 36
ena.group, 37
ena.make.set, *34*, 38, *41*
ena.plot, 40, *43*, *45*, *48*, *50*
ena.plot.group, 41, *48*
ena.plot.network, 43
ena.plot.points, *41*, *45*, 46
ena.plot.trajectory, 49
ena.plotter, 51
ena.rotate.by.generalized, 52
ena.rotate.by.hena.regression, 53
ena.rotate.by.hena.regression_2, 53
ena.rotate.by.mean, 54
ena.rotation.h, 55
ena.set.creator, 55
ena.svd, 57
ena.writeup, 58
ena_correlation, 59
ENAdata, *34*, *38*, 59, *64*
ENAplot, *41–45*, *47–50*, 61
ENARotationSet, *52–54*, 63, *65*
ENAset, *37*, *39, 40*, *52–54*, *61, 62*, 64

find_binary_cols, 66
find_code_cols, 67
find_dimension_cols, 67
find_meta_cols, 68
fun_cohens.d, 68
fun_skip_sphere_norm, 69
fun_sphere_norm, 69

get_x1_main_effect, 70, *71*
gmr, 71

group, 72

horizon, 73

is.qe.code, 74
is.qe.data, 74
is.qe.horizon, 75
is.qe.metadata, 76
is.qe.unit, 76

means_rotate, 77
merge_columns_c, 77
metadata, 78
methods_report, 78
methods_report_stream, 79
model, 80
move_nodes_to_unit_circle, 81
move_nodes_to_unit_circle_with_equal_space, 82

namesToAdjacencyKey, 82

optimize, 83

plot.ena.set, 84
prepare_trajectory_data, 85
print.ena.set, 87
project, 87
project_in, 88

reclassify, 89
remove_meta_data, 89
rENA, 90
rotate, 90
RS.data, 91

scale.ENAplot, 91
show, 92
sphere_norm, 92

units, 93

vector_to_ut, 94

with.ena.matrix, 94
with_means, 95
with_trajectory, 95