Package 'imaginarycss'

March 4, 2026

Version 0.1.0

Title Tools for Studying Imaginary Cognitive Social Structure

Description Provides functions to measure and test imaginary cognitive social structure (CSS) motifs, which are patterns of perceived relationships among individuals in a social network. Includes tools for calculating motif frequencies, comparing observed motifs to expected distributions, and visualizing motif structures. Implements methods described in Tanaka and Vega Yon (2023) <doi:10.1016/j.socnet.2023.11.005>.

URL https://gvegayon.github.io/imaginary-structures/

BugReports https://github.com/gvegayon/imaginary-structures/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

LinkingTo Rcpp, barry

Imports graphics, Rcpp, stats

Suggests knitr, rmarkdown, tinytest, quarto

Depends R (>= 4.0.0)

VignetteBuilder quarto

NeedsCompilation yes

Author Sima Najafzadehkhoei [aut, cre] (ORCID:
<https://orcid.org/0009-0002-6253-2910>),
George Vega Yon [aut] (ORCID: <https://orcid.org/0000-0002-3171-0844>),
Kyosuke Tanaka [aut] (ORCID: <https://orcid.org/0000-0002-1850-6814>)

Maintainer Sima Najafzadehkhoei <sima.njf@utah.edu>

Repository CRAN

Date/Publication 2026-03-04 09:50:02 UTC

Contents

`barray_to_edgelist` *Retrieves the edgelist of a barry_graph*

Description

Retrieves the edgelist of a barry_graph

Usage

```
barray_to_edgelist(x)
```

Arguments

`x`	An object of class barry_graph.

Value

A matrix with two columns, the first one with the source and the second one with the target.

Examples

```
data(krackhardt_advice)
data(krackhardt_advice_perceptions)

n_people <- 21
advice_matrix <- matrix(0L, nrow = n_people, ncol = n_people)
advice_matrix[cbind(krackhardt_advice$from, krackhardt_advice$to)] <-
  krackhardt_advice$value

krack_graph <- new_barry_graph(
```

```
  c(list(advice_matrix), krackhardt_advice_perceptions)
)
head(barray_to_edgelist(krack_graph))
```

`count_imaginary_census` *Computes census of imaginary errors*

Description

Computes census of imaginary errors

Usage

```
count_imaginary_census(x, counter_type = 0L)
```

Arguments

`x`	An object of class barry_graph.
`counter_type`	An integer indicating the type of census to compute (see details).

Details

We can also separate the counts as a function of whether the perceiver is looking into all ties, only ties including them, or only ties not including them. This is controlled by the `counter_type` argument:

- 0: All ties
- 1: Only ties including the perceiver
- 2: Only ties not including the perceiver

There are ten (10) values:

- (01) Accurate null
- (02) Partial false positive (null)
- (03) Complete false positive (null)
- (04) Partial false negative (assym)
- (05) Accurate assym
- (06) Mixed assym
- (07) Partial false positive (assym)
- (08) Complete false negative (full)
- (09) Partial false negative (full)
- (10) Accurate full

Value

A data frame of class "imaginary_census" with columns id, name, and value.

Examples

```
data(krackhardt_advice)
data(krackhardt_advice_perceptions)

n_people <- 21
advice_matrix <- matrix(0L, nrow = n_people, ncol = n_people)
advice_matrix[cbind(krackhardt_advice$from, krackhardt_advice$to)] <-
  krackhardt_advice$value

krack_graph <- new_barry_graph(
  c(list(advice_matrix), krackhardt_advice_perceptions)
)
census <- count_imaginary_census(krack_graph)
head(census)
summary(census)
```

---

count_recip_errors *Add a counter for reciprocity errors*

Description

Add a counter for reciprocity errors

Usage

```
count_recip_errors(x, counter_type = 0L)
```

Arguments

x	An object of class barry_graph.
counter_type	An integer indicating the type of census to compute (see details).

Details

We can also separate the counts as a function of whether the perceiver is looking into all ties, only ties including them, or only ties not including them. This is controlled by the counter_type argument:

- 0: All ties
- 1: Only ties including the perceiver
- 2: Only ties not including the perceiver

Value

A data frame with columns `id` (integer perceiver identifier), `name` (character label for the reciprocity error type), and `value` (numeric count).

Examples

```
data(krackhardt_advice)
data(krackhardt_advice_perceptions)

n_people <- 21
advice_matrix <- matrix(0L, nrow = n_people, ncol = n_people)
advice_matrix[cbind(krackhardt_advice$from, krackhardt_advice$to)] <-
  krackhardt_advice$value

krack_graph <- new_barry_graph(
  c(list(advice_matrix), krackhardt_advice_perceptions)
)
count_recip_errors(krack_graph)
```

`krackhardt_advice` *Krackhardt High-Tech Managers Advice Network*

Description

Advice-seeking relationships among 21 managers in a high-tech company. Data represents who seeks advice from whom (directed network).

Usage

```
krackhardt_advice
```

Format

A data frame with 441 rows and 3 variables:

from Integer, source node ID (1-21)

to Integer, target node ID (1-21)

value Integer, 1 if advice relationship exists, 0 otherwise

Source

Krackhardt, D. (1987). Cognitive social structures. Social Networks, 9(2), 109-134.

References

Krackhardt, D. (1987). Cognitive social structures. Social Networks, 9(2), 109-134.

krackhardt_advice_perceptions *Krackhardt Advice Network Perception Errors*

Description

Individual perceptions of the Krackhardt advice network containing systematic perception errors and biases.

Usage

```
krackhardt_advice_perceptions
```

Format

A list of 21 matrices (21x21 each) representing individual perceptions

Source

Generated perception errors based on Krackhardt advice data

krackhardt_attributes *Krackhardt High-Tech Managers Attributes*

Description

Node attributes for the 21 managers in Krackhardt's high-tech company study. Contains demographic and organizational information for each manager.

Usage

```
krackhardt_attributes
```

Format

A data frame with 21 rows and 5 variables:

ID Integer, manager ID (1-21)

AGE Numeric, age of the manager

TENURE Numeric, tenure of the manager in the company (in years)

LEVEL Factor, hierarchical level of the manager (e.g., "Top", "Middle", "Lower")

DEPT Factor, department of the manager (e.g., "R&D", "Marketing", etc.)

Source

Krackhardt, D. (1987). Cognitive social structures. Social Networks, 9(2), 109-134.

krackhardt_friendship *Krackhardt High-Tech Managers Friendship Network*

Description

Friendship relationships among 21 managers in a high-tech company. Data represents who considers whom a friend (directed network).

Usage

```
krackhardt_friendship
```

Format

A data frame with 441 rows and 3 variables:

from Integer, source node ID (1-21)

to Integer, target node ID (1-21)

value Integer, 1 if friendship exists, 0 otherwise

Source

Krackhardt, D. (1987). Cognitive social structures. Social Networks, 9(2), 109-134.

krackhardt_friendship_perceptions *Krackhardt Friendship Network Perception Errors*

Description

Individual perceptions of the Krackhardt friendship network containing systematic perception errors and biases.

Usage

```
krackhardt_friendship_perceptions
```

Format

A list of 21 matrices (21x21 each) representing individual perceptions

Source

Generated perception errors based on Krackhardt friendship data

krackhardt_reports *Krackhardt High-Tech Managers Reporting Network*

Description

Formal reporting relationships among 21 managers in a high-tech company. Data represents who reports to whom (directed network).

Usage

```
krackhardt_reports
```

Format

A data frame with 441 rows and 3 variables:

from Integer, source node ID (1-21)

to Integer, target node ID (1-21)

value Integer, 1 if reporting relationship exists, 0 otherwise

Source

Krackhardt, D. (1987). Cognitive social structures. Social Networks, 9(2), 109-134.

new_barry_graph *Binary Array Graph*

Description

Binary Array Graph

Usage

```
new_barry_graph(x, ...)

## S3 method for class 'matrix'
new_barry_graph(x, n, ...)

## S3 method for class 'list'
new_barry_graph(x, ...)

netsize(x)

nnets(x)
```

Arguments

x	Either a matrix or a list of matrices, or an object of class barry_graph.
...	Currently ignored.
n	Integer. The size of the original network.

Details

When x is a matrix, it is assumed that it will be a block diagonal matrix, with the first block corresponding to the reference (true) network.

If x is a list, the first matrix is assumed to be the reference (true) network.

Value

new_barry_graph() returns an external pointer object of class "barry_graph" with attributes netsize (integer scalar giving the size of each individual network) and endpoints (integer vector marking the boundary rows of the stacked networks).

The function netsize() returns the size of individual networks (all matching).

nnets() returns the number of graphs contained in the barry_graph object.

Examples

```
# Using the Krackhardt advice network
data(krackhardt_advice)
data(krackhardt_advice_perceptions)

# Convert edge-list data frame to adjacency matrix
n_people <- 21
advice_matrix <- matrix(0L, nrow = n_people, ncol = n_people)
advice_matrix[cbind(krackhardt_advice$from, krackhardt_advice$to)] <-
  krackhardt_advice$value

krack_graph <- new_barry_graph(
  c(list(advice_matrix), krackhardt_advice_perceptions)
)
krack_graph

# Network size and number of networks
netsize(krack_graph)
nnets(krack_graph)
```

---

print.barry_graph *Print Barry Graph*

Description

Print method for barry_graph objects.

Usage

```
## S3 method for class 'barry_graph'
print(x, n = min(10, netsize(x)), ...)
```

Arguments

x	A barry_graph object.
n	Integer. Number of nodes to display (default: min of 10 and the network size).
...	Additional arguments passed to print (currently ignored).

Value

Invisibly returns the input object. Called for its side effect of printing.

Examples

```
data(krackhardt_advice)
data(krackhardt_advice_perceptions)

n_people <- 21
advice_matrix <- matrix(0L, nrow = n_people, ncol = n_people)
advice_matrix[cbind(krackhardt_advice$from, krackhardt_advice$to)] <-
  krackhardt_advice$value

krack_graph <- new_barry_graph(
  c(list(advice_matrix), krackhardt_advice_perceptions)
)
print(krack_graph)
```

---

summary.imaginary_census

Summarize an imaginary census

Description

Aggregates the per-perceiver motif counts returned by count_imaginary_census() into totals per motif type.

Usage

```
## S3 method for class 'imaginary_census'
summary(object, ...)
```

Arguments

object	A data frame returned by count_imaginary_census().
...	Currently ignored.

Value

A named numeric vector with total counts per motif type (sorted in decreasing order).

Examples

```
data(krackhardt_advice)
data(krackhardt_advice_perceptions)

n_people <- 21
advice_matrix <- matrix(0L, nrow = n_people, ncol = n_people)
advice_matrix[cbind(krackhardt_advice$from, krackhardt_advice$to)] <-
  krackhardt_advice$value

krack_graph <- new_barry_graph(
  c(list(advice_matrix), krackhardt_advice_perceptions)
)
census <- count_imaginary_census(krack_graph)
summary(census)
```

---

test_imaginary_census *Test imaginary census motifs against a null model*

Description

Generates a null distribution of imaginary-census motif counts by repeatedly sampling CSS networks with `sample_css_network()` and compares the observed counts to this null distribution. Returns an S3 object of class `"imaginarycss_test"` with `print`, `summary`, and `plot` methods.

Usage

```
test_imaginary_census(graph, n_sim = 100L, alpha = 0.05, counter_type = 0L)

## S3 method for class 'imaginarycss_test'
print(x, ...)

## S3 method for class 'imaginarycss_test'
summary(object, ...)

## S3 method for class 'imaginarycss_test'
plot(x, main = "Motif Z-Scores vs Null", ...)
```

Arguments

`graph`	A `barry_graph` object.
`n_sim`	Integer. Number of null-model simulations (default `100`).
`alpha`	Numeric. Significance level for the two-sided test (default `0.05`).

counter_type	Integer passed to count_imaginary_census() (default 0L).
x	An object of class "imaginarycss_test".
...	Currently ignored.
object	An object of class "imaginarycss_test".
main	Character. Plot title.

Value

An object of class "imaginarycss_test", which is a list containing:

results A data frame with columns motif, observed, null_mean, null_sd, z_score, p_value, and significant.

observed Named numeric vector of observed motif totals.

null_matrix Matrix of null motif totals (motifs x simulations).

n_sim Number of simulations used.

alpha Significance level used.

print.imaginarycss_test() returns the input object invisibly. Called for its side effect of printing a summary of significant motifs.

summary.imaginarycss_test() returns the input object invisibly. Called for its side effect of printing the full results table.

plot.imaginarycss_test() returns the input object invisibly. Called for its side effect of drawing a horizontal barplot of z-scores.

Examples

```
data(krackhardt_advice)
data(krackhardt_advice_perceptions)

n_people <- 21
advice_matrix <- matrix(0L, nrow = n_people, ncol = n_people)
advice_matrix[cbind(krackhardt_advice$from, krackhardt_advice$to)] <-
  krackhardt_advice$value

krack_graph <- new_barry_graph(
  c(list(advice_matrix), krackhardt_advice_perceptions)
)
res <- test_imaginary_census(krack_graph, n_sim = 50)
res
summary(res)
plot(res)
```

`tie_level_accuracy` *Null distribution for Cognitive Imaginary Structures*

Description

These functions can be used to generate null distributions for testing the prevalence of imaginary CSS. The null is a function of the individual level accuracy rates, in other words. Pr(i perceives a one| there is a one) and Pr(i perceives a zero| there is a zero).

Usage

```
tie_level_accuracy(graph, which_nets = NULL)

sample_css_network(
  graph,
  prob = tie_level_accuracy(graph),
  i = 1L:attr(graph, "netsize"),
  keep_baseline = TRUE
)
```

Arguments

`graph`	A barry_graph object.
`which_nets`	Integer vector. The networks to sample from.
`prob`	A numeric vector of length 4L or a data frame (see details).
`i`	Integer vector. The network to sample from.
`keep_baseline`	Logical scalar. When `TRUE`, the function returns the baseline network as the first element of the list.

Details

There are two special cases worth mentioning. First, when the dyads in question are all present the probability of true negative is set to `NA`. On the other hand, if the dyads in question are all null, the probability of true positive is `NA` as well. This doesn't affect the `sample_css_network` function because those probabilities are unsed since tie/no tie probabilities are according to the baseline graph, meaning that, for instance, a fully connected network will never use the `p_0_ego` and `p_0_alter` probabilities and an empty network will never use the `p_1_ego` and `p_1_alter` probabilities.

The function `sample_css_network` samples perceived networks from the baseline network. The baseline network is the first network in the `graph` object. The function `tie_level_accuracy` can be used to generate the probability vector.

The probability vector is a numeric vector of length 4L. The first two elements are the probability of a tie/no tie between an ego and an alter. The third and fourth elements are the probability of a tie/no tie between two alters. When `prob` is a data frame, the function will sample from each row of the data frame (returned from the function `tie_level_accuracy`).

Value

The function `tie_level_accuracy` returns a data frame with the following columns:

- `k`: The perceiver id.
- `p_0_ego`: The probability of no tie between the perceiver (ego) and an alter.
- `p_1_ego`: The probability of a tie between the perceiver and an alter.
- `p_0_alter`: The probability of no tie between two alters.
- `p_1_alter`: The probability of a tie between two alters.

The function `sample_css_network` returns a list of square matrices of size `attr(graph, "netsize")`. If `keep_baseline = TRUE`, the first element of the list is the baseline network. Otherwise, it is not returned.

Examples

```
# Using the Krackhardt advice network
data(krackhardt_advice)
data(krackhardt_advice_perceptions)

n_people <- 21
advice_matrix <- matrix(0L, nrow = n_people, ncol = n_people)
advice_matrix[cbind(krackhardt_advice$from, krackhardt_advice$to)] <-
  krackhardt_advice$value

krack_graph <- new_barry_graph(
  c(list(advice_matrix), krackhardt_advice_perceptions)
)

# Calculate accuracy rates
accuracy <- tie_level_accuracy(krack_graph)
print(accuracy)

# Visualize accuracy patterns
boxplot(accuracy[, -1],
        main = "Accuracy Rates by Type",
        ylab = "Probability",
        names = c("P(0|0) Ego", "P(1|1) Ego",
                  "P(0|0) Alter", "P(1|1) Alter"))

# Using the Krackhardt advice network
data(krackhardt_advice)
data(krackhardt_advice_perceptions)

n_people <- 21
advice_matrix <- matrix(0L, nrow = n_people, ncol = n_people)
advice_matrix[cbind(krackhardt_advice$from, krackhardt_advice$to)] <-
  krackhardt_advice$value

krack_graph <- new_barry_graph(
  c(list(advice_matrix), krackhardt_advice_perceptions)
)
```

```
# Method 1: Using accuracy data frame (recommended)
accuracy <- tie_level_accuracy(krack_graph)
sampled_networks <- sample_css_network(krack_graph, prob = accuracy)
length(sampled_networks)

# Method 2: Using manual probability vector for a single perceiver
# p_0_ego, p_1_ego, p_0_alter, p_1_alter
manual_probs <- c(0.8, 0.9, 0.85, 0.75)
sampled_manual <- sample_css_network(
  krack_graph,
  prob = manual_probs,
  i = 1,
  keep_baseline = FALSE
)
```

Index