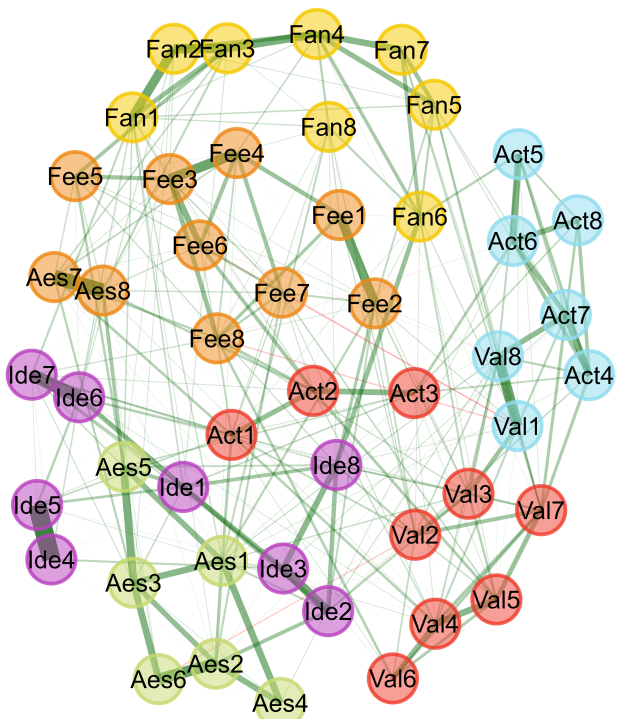
Exploratory Graph Analysis

PSY-GS 8875 Behavioral Data Science



Overview

Overview: Week 11

Readings

- ESL Chapters: 17.1-17.3
- Epskamp and Fried - 2018
- Golino and Epskamp - 2017

Optional

- Pons and Latapy -2006
- Christensen - 2024
- Schmittmann et al. - 2013
- Blondel et al. - 2008
- Christensen et al. - 2023 - community
- Golino et al. - 2020

- Network analysis
- Network estimation
- Community detection
- Exploratory Graph Analysis (EGA)
- Unidimensionality
- Total Entropy Fit (TEFI)

Dimension Reduction

Dimension Reduction

Goal: Dimension reduction is useful for reducing a large set of *variables* to a smaller summary set of variables

Many different approaches exist to accomplish this task

- Principal Component Analysis (PCA)
- Factor Analysis (FA)
- Exploratory Graph Analysis (EGA)

Principal Component Analysis

- Seeks to identify a linear combination of variables that maximizes variance on each consecutive component
- Each component is *orthogonal* (no correlations between components)
- Useful for creating clear and unique dimensions (but not necessary *valid*)

Factor Analysis

- Seeks to identify latent variables that *underlie* the relationships between variables
- Often interpreted as a "common cause" of the relationships between variables
- Assumes that after accounting for the latent variables, observed variables are no longer correlated (local dependence assumption)
- Most commonly used and assumed model in psychometrics
- Nearly *all* scales are developed and validated with this model in mind

Big Five Personality Inventory

- Five theoretical factors: openness to experience, conscientiousness, extraversion, agreeableness, neuroticism
- 25 items (5 items per factor)
- Sample size = 4,000

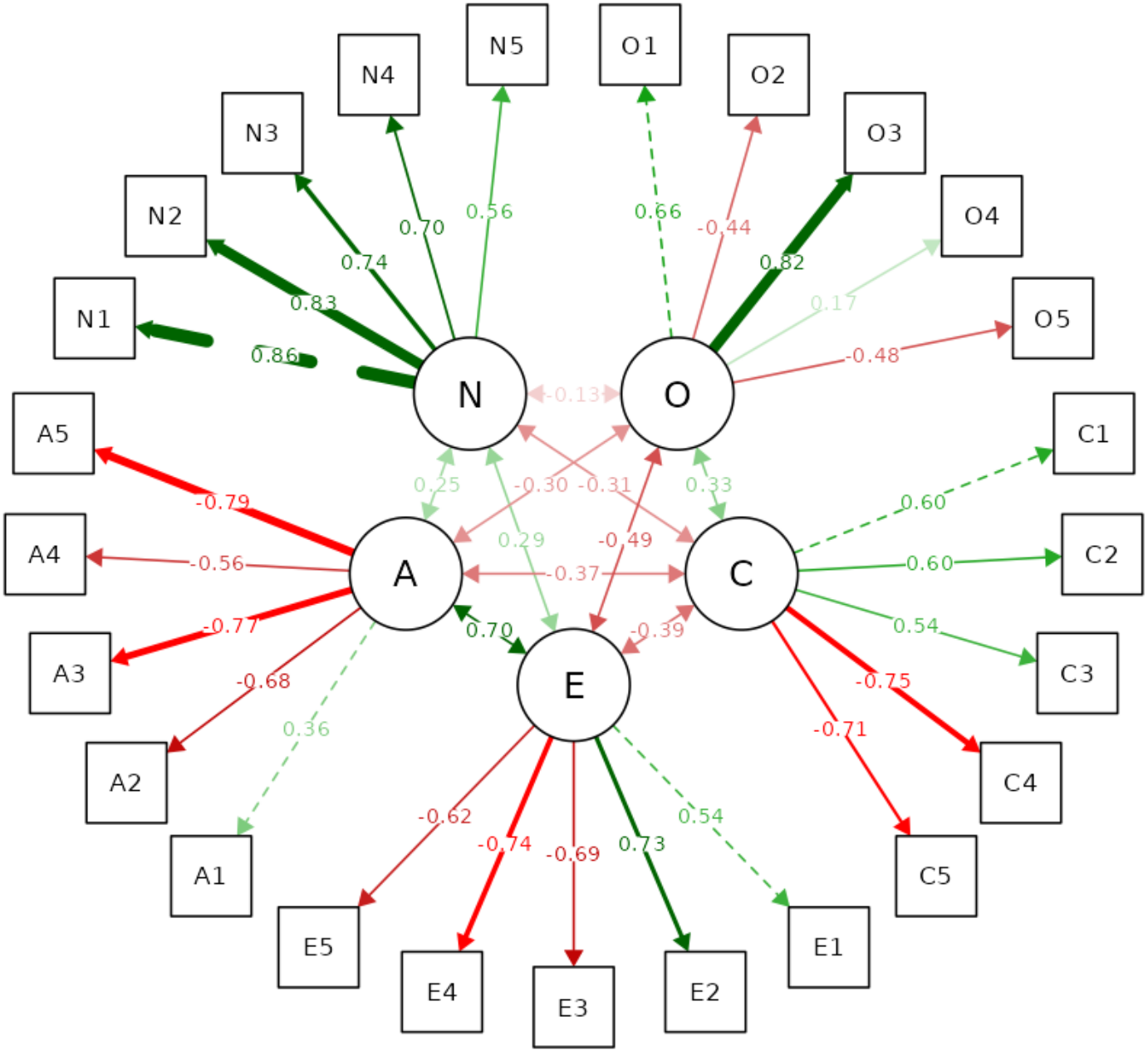
☑ Available in the {psych} package

```r
# Load packages
library(psych); library(lavaan); library(semPlot)

# Load bfi data
data <- bfi[,1:25]

# Set up correlated factor model
model <- paste0(
  "O =~ ", paste0("O", 1:5, collapse = " + "), "\n",
  "C =~ ", paste0("C", 1:5, collapse = " + "), "\n",
  "E =~ ", paste0("E", 1:5, collapse = " + "), "\n",
  "A =~ ", paste0("A", 1:5, collapse = " + "), "\n",
  "N =~ ", paste0("N", 1:5, collapse = " + ")
)

# Fit CFA model
fit <- cfa(
  model = model, data = data,
  ordered = colnames(data), # ensure data are treated as ordinal
  estimator = "WLSMV" # use categorical estimator
)
```

```r
# Summary
round(
  fitMeasures(fit)[c(
    "baseline.chisq.scaled", "baseline.df.scaled",
    "baseline.pvalue.scaled",
    "cfi.scaled", "tli.scaled",
    "rmsea.scaled", "rmsea.ci.lower.scaled",
    "rmsea.ci.upper.scaled", "rmsea.pvalue.scaled",
    "srmr", "srmr_bentler"
  )], 3
)
```

```
 baseline.chisq.scaled     baseline.df.scaled baseline.pvalue.scaled
             33250.704                300.000                   0.000
            cfi.scaled            tli.scaled           rmsea.scaled
                 0.824                  0.801                   0.095
rmsea.ci.lower.scaled rmsea.ci.upper.scaled    rmsea.pvalue.scaled
                 0.093                  0.097                   0.000
                  srmr           srmr_bentler
                 0.083                  0.070
```

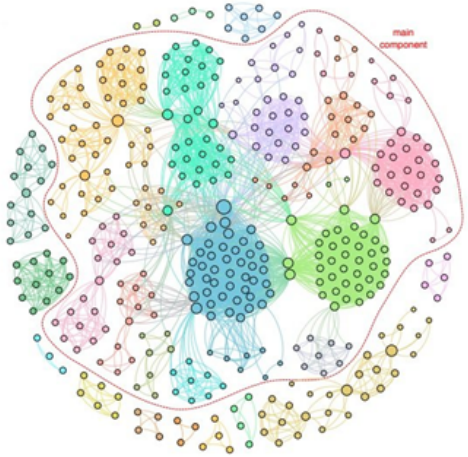
```r
# Plot CFA
semPaths(
    fit, what = "std",
    intercepts = FALSE, residuals = FALSE,
    thresholds = FALSE, sizeLat = 7, sizeMan = 5,
    node.width = 1, layout = "circle"
)
```

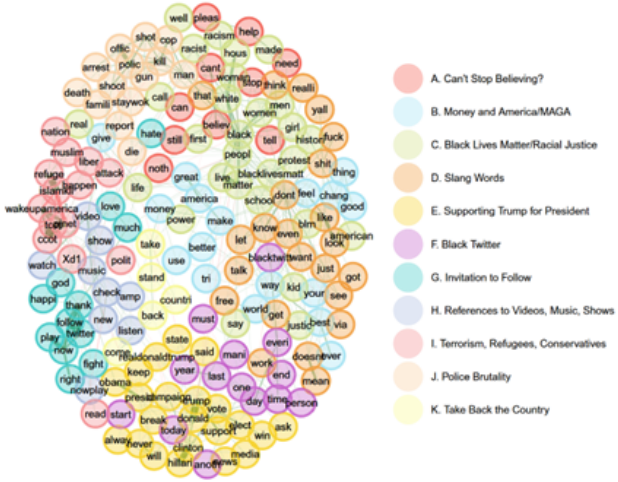

N5
O1
O2
N4
O3
N3
O4
N2
N1
O5
0.70
0.56
0.66
-0.44
0.74
0.82
0.83
0.17
0.86
-0.48
N
O
-0.13
A5
C1
0.25
-0.30
-0.31
0.33
-0.79
0.60
A4
C2
0.29
-0.49
-0.56
A
C
-0.37
0.60
-0.77
0.70
-0.39
0.54
A3
C3
E
-0.68
-0.75
0.36
-0.71
A2
C4
-0.62
0.54
-0.74
-0.69
0.73
A1
C5
E5
E1
E4
E3
E2

Network Analysis

Network Analysis

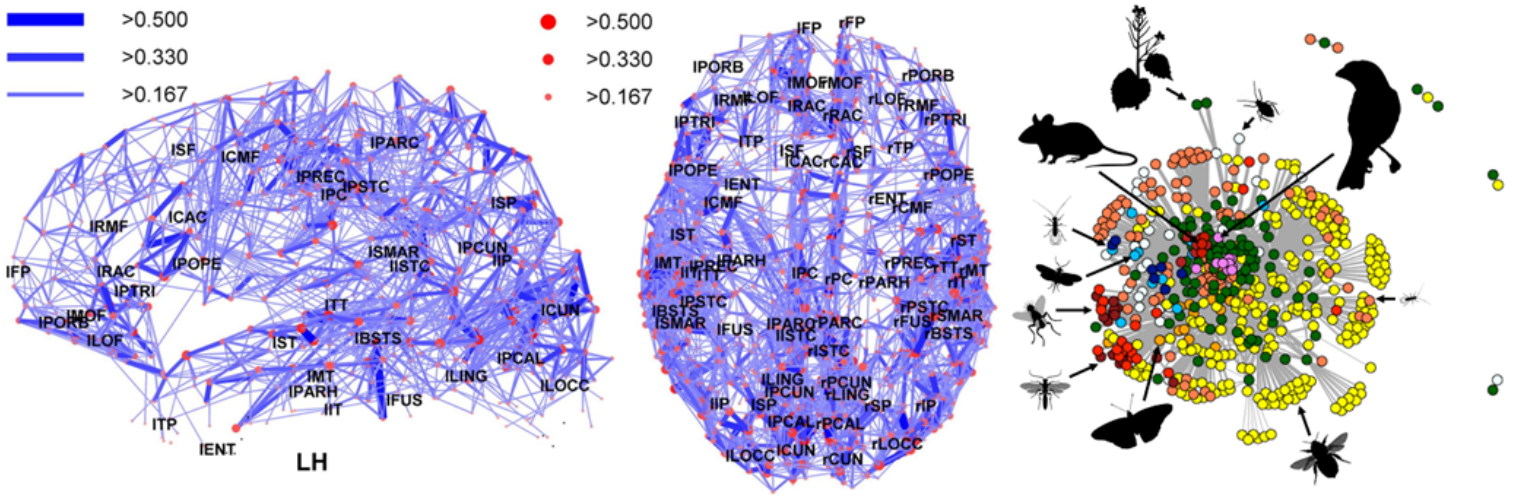
Networks are everywhere



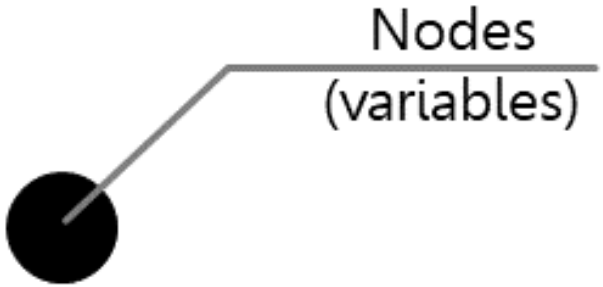


Political Corruption

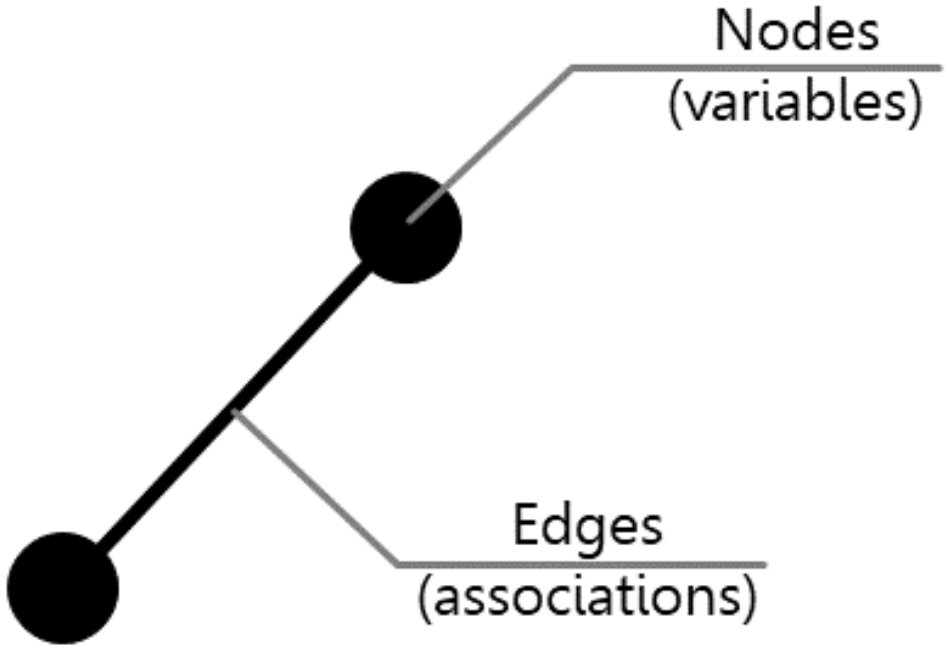
Twitter Data



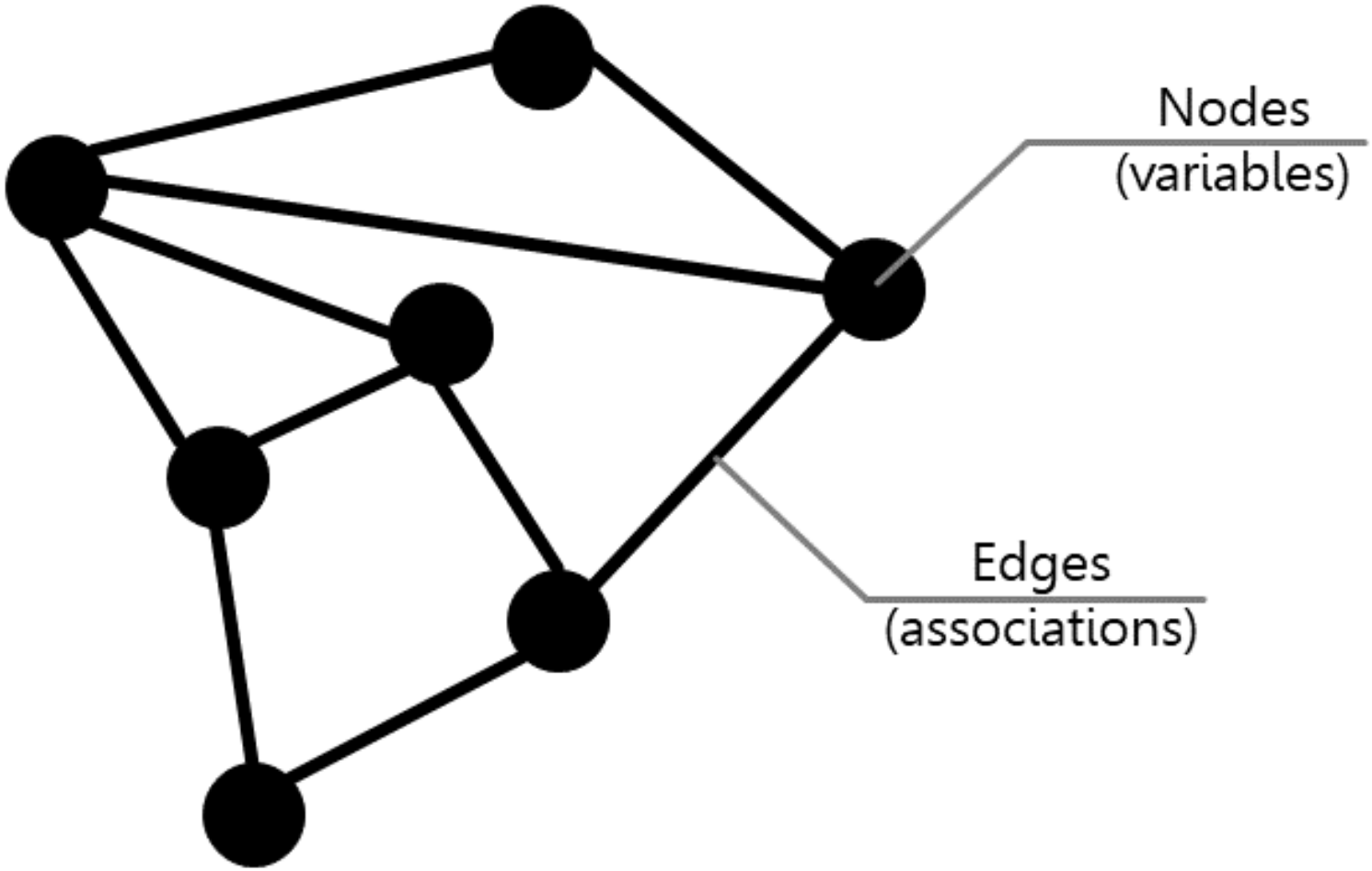
Breaking Down Networks



Breaking Down Networks



Breaking Down Networks



Types of networks

Social network

node: people
edge: relationship
sub-network: group of people or ***community***

Brain network

node: neuron, region of interest
edge: co-activation
sub-network: default mode

Semantic network

node: concept stored in memory
edge: association
sub-network: category

Psychometric network

node: observable variable
edge: (partial) correlation
sub-network: dimension (or factor)

Network Analysis

- Networks are defined and interpreted by their constituent elements (nodes and edges)
- There are few inherent assumptions
- Gaussian graphical models (most common in the social sciences)
  - multivariate normal
  - (in)conditional relationships

Exploratory Graph Analysis

Exploratory Graph Analysis

- Started as a network science method combining Gaussian graphical models with community detection algorithms
- Since expanded into a full-fledged framework based on the premise of connecting traditional psychometrics to network psychometrics
- Relatively new (circa 2017) and actively developing

Exploratory Graph Analysis

Steps

1. Estimate associations
2. Estimate network
3. Apply community detection algorithm

1. Estimate associations

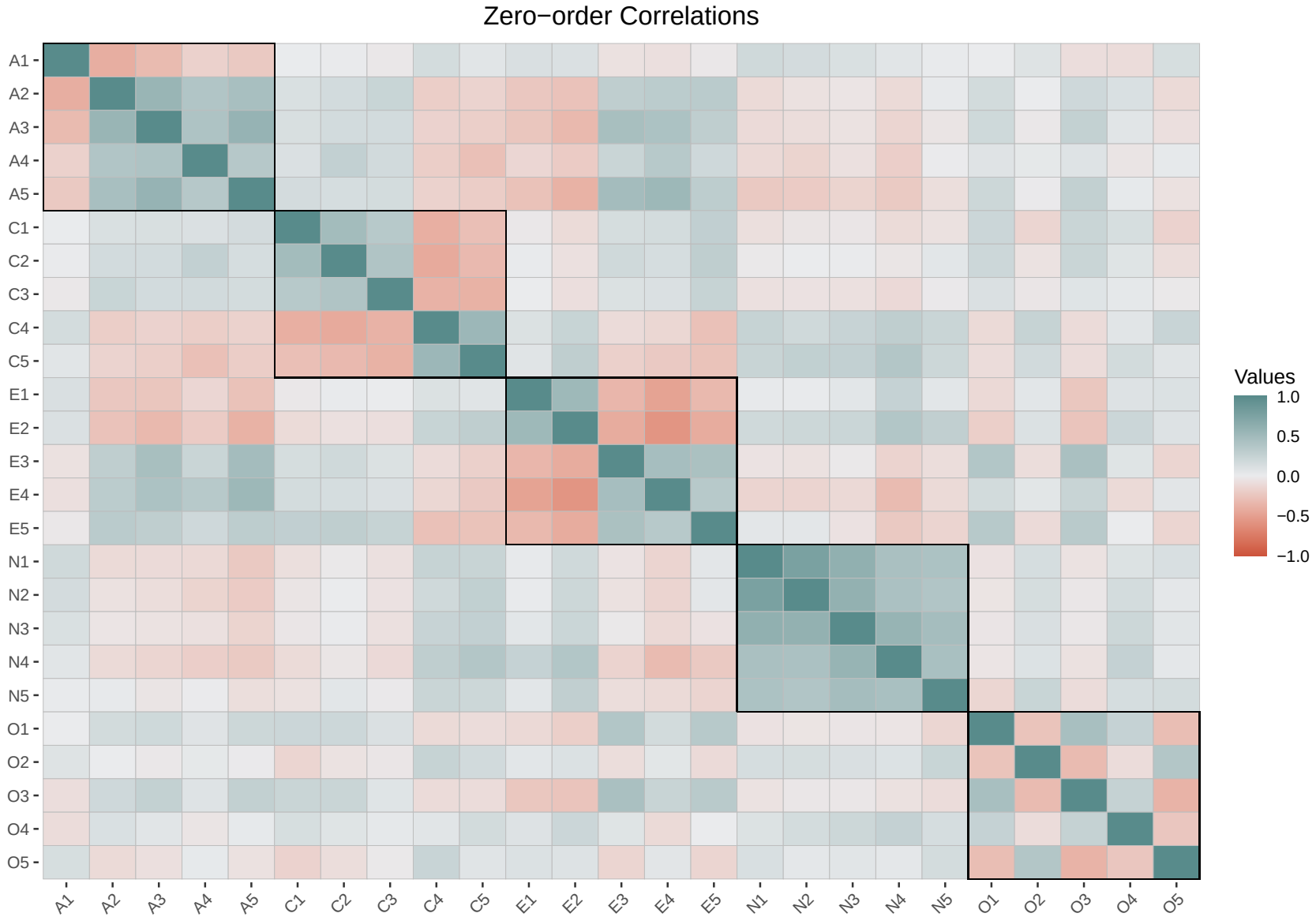
$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

- Continuous data (8 or more categories): Pearson's correlation

- **Polytomous** (3-7 categories): polychoric correlation
- Dichotomous (2 categories): tetrachoric correlation
- Polytomous/Dichotomous-Continuous: poly-/bi-serial correlation
- Non-parametric: Spearman's rho

{EGAnet}'s `auto.correlate` *automatically* computes the appropriate correlations for you

```r
# Load packages
library(EGAnet); library(ggplot2)

# Compute correlations
correlations <- auto.correlate(data)
```


Zero−order Correlations
A1
A2
A3
A4
A5
C1
C2
C3
C4
C5
E1
E2
E3
E4
E5
N1
N2
N3
N4
N5
O1
O2
O3
O4
O5
Values
1.0
0.5
0.0
−0.5
−1.0

- Psychometric networks use partial correlations given all other variables
- These partial correlations correspond to the conditional relationship between two variables
- Can be converted directly to βs from linear regression

Let $\mathbf{S}$ represent the sample covariance matrix, then the correlation matrix:

$$\mathbf{R} = [\text{diag}(\mathbf{S})]^{-1/2} \; \mathbf{S} \; [\text{diag}(\mathbf{S})]^{-1/2}$$

The inverse covariance matrix equals the precision or conditional relationships between variables:

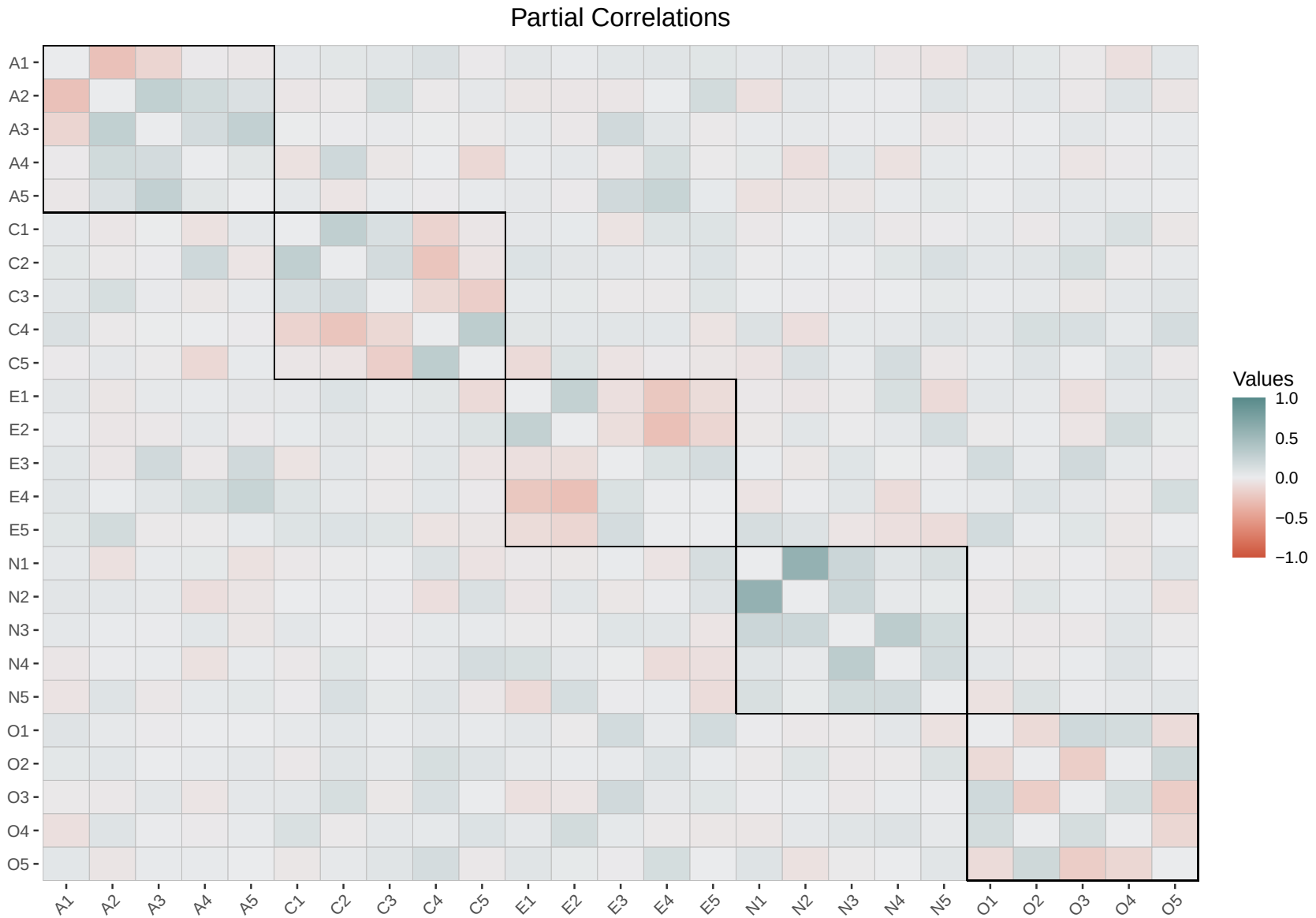
$$\mathbf{K} = \mathbf{S}^{-1}$$

The inverse covariance matrix can be converted to partial correlations:

$$\mathbf{P} = -\left([\text{diag}(\mathbf{K})]^{-1/2} \; \mathbf{K} \; [\text{diag}(\mathbf{K})]^{-1/2} \right)$$

Why partial correlations?

Why partial correlations?



The inverse covariance matrix can be used to compute βs:

$$\beta_{ij} = \frac{-\kappa_{ij}}{\kappa_{ii}}$$

Similarly, the partial correlations can be used to compute βs:

$$\beta_{ij} = p_{ij} \sqrt{\frac{\kappa_{ii}}{\kappa_{jj}}}$$

Guttman (1953) has an amazing paper on these transfers between correlation, regression, and partial correlation worlds

Graphical LASSO

- Applies the LASSO to the inverse covariance matrix ($\mathbf{K}$)
- **Goal**: reduce overfitting but also create a sparse network structure
- (G)LASSO sets small coefficients to zero making *sparsity* a natural consequence

Formal Notation

$$\log\det(\mathbf{K}) - \text{tr}(\mathbf{SK}) - \lambda \sum_{<i,j>} |_{ij}|$$

Algorithm

- Apply standard LASSO regularization to each variable and permutate the last variable (column and row) to the first
- Solve: $\mathbf{S}_{11}\beta - s_{12} + \lambda \cdot \text{sign}(\beta) = 0$
- Repeat until convergence

GLASSO Model Selection

- λ affects the sparsity (how densely connected) of the network
- This parameter should be chosen with care
  - Too sparse and the model may detect the "true" underlying structure
  - Too dense and the model is overparameterized
- Model selection criterion:
  - Akaike Information Criterion (AIC)
  - Corrected AIC (AICc)
  - Bayesian Information Criterion (BIC)
  - Extended Bayesian Information Criterion (EBIC)

GLASSO Model Selection

- EBIC tends to be the standard approach

$$L = \frac{N}{2} \log\det(\mathbf{K}) - \text{tr}(\mathbf{SK})$$

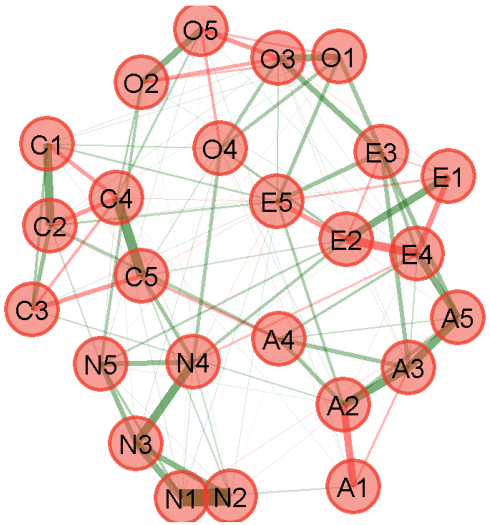
$$EBIC = -2L + E\log(N) + 4\gamma E\log(V)$$

- L = log-likelihood
- E = number of edges (connections)
- N = sample size
- V = number of variables
- γ = preference for more or less complex models ($\gamma = 0 = BIC$)
  - smaller γ = more complex
  - larger γ = more parsimonious

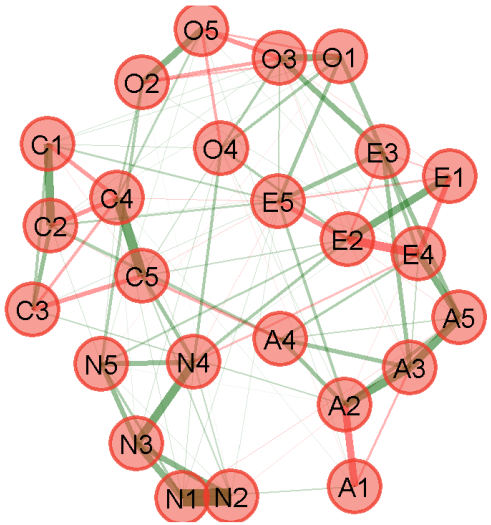
GLASSO Model Selection

- Using EBIC, a model search over many $lambda$ parameters is performed
- This search is over a logarithmic number of λ parameters with a "min-max" ratio
- Default of this ratio in $\{\text{EGAnet}\} = 0.01$

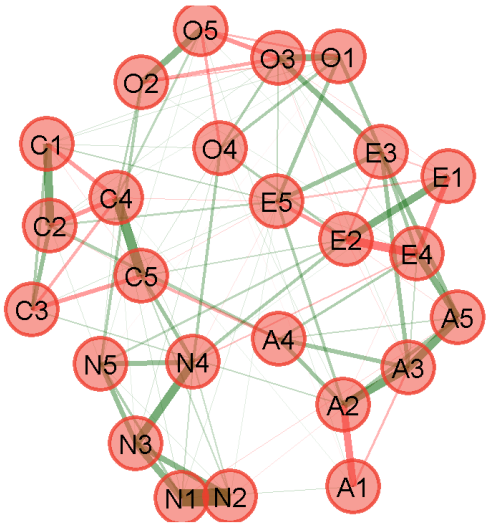
lambda = 0.076
EBIC = 48305.432



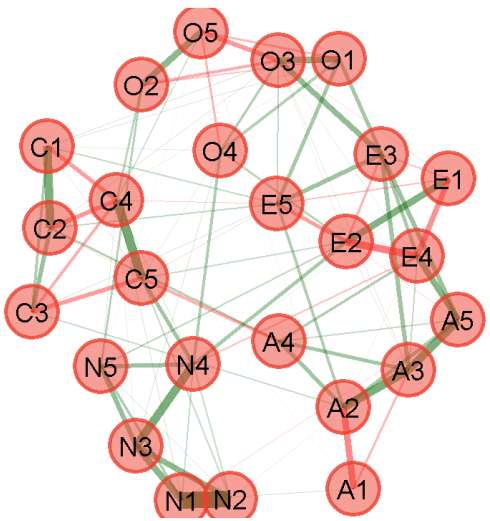
lambda = 0.084
EBIC = 48502.152



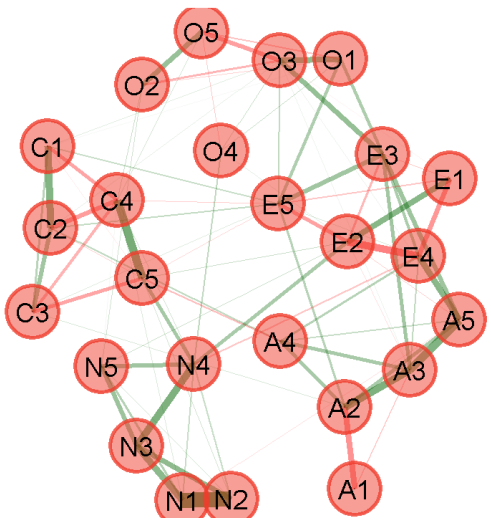
lambda = 0.094
EBIC = 48821.611



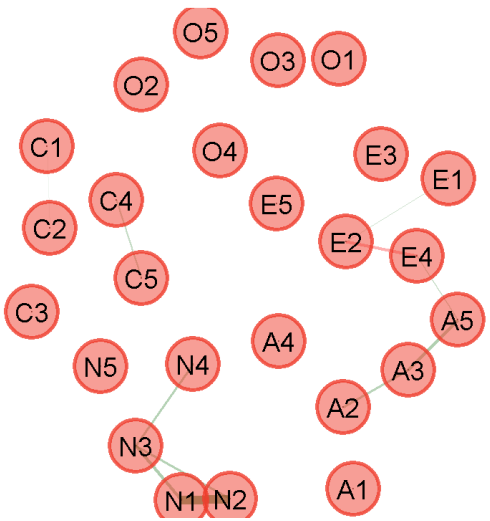
lambda = 0.119
EBIC = 49563.89



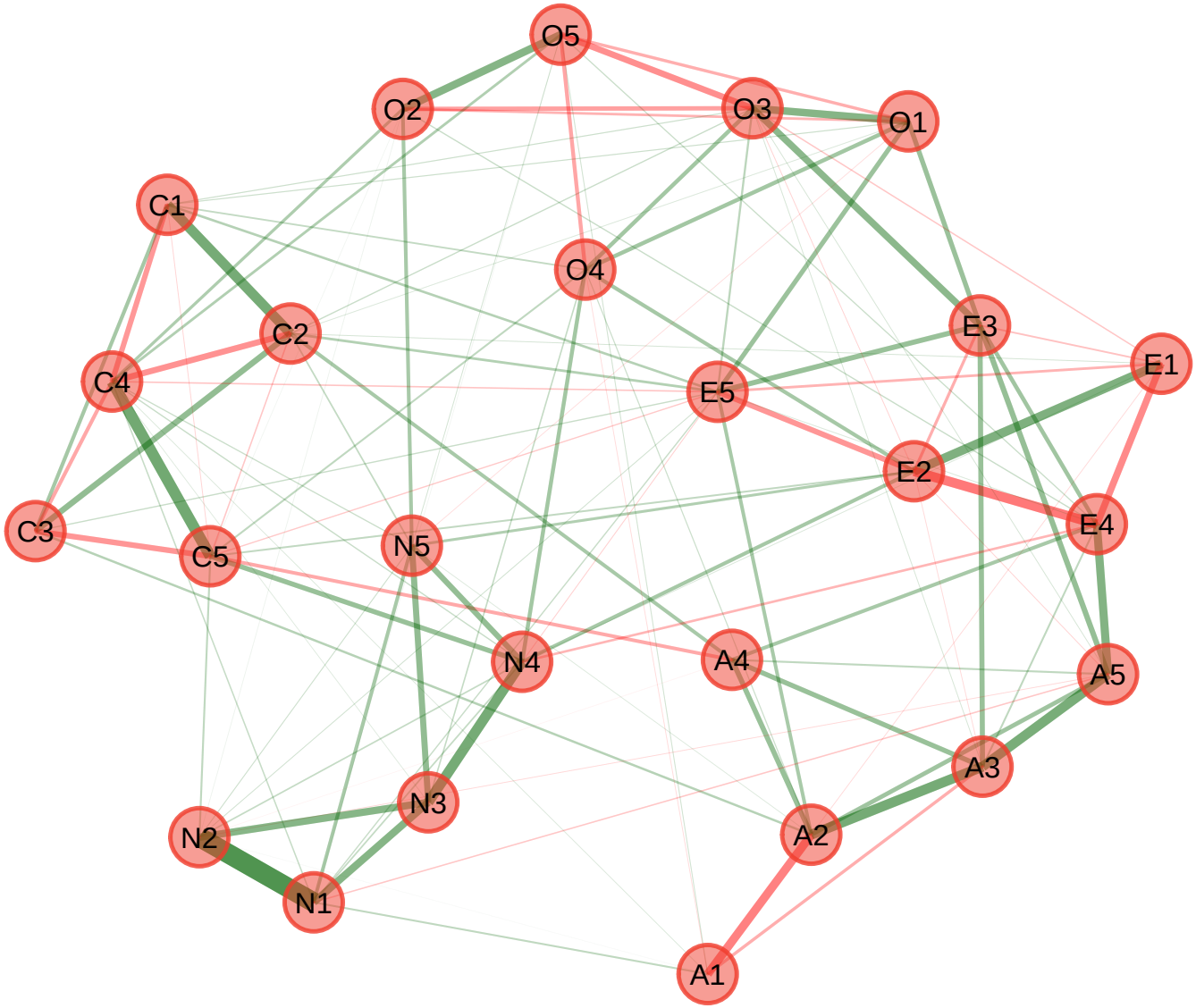
lambda = 0.189
EBIC = 52236.811



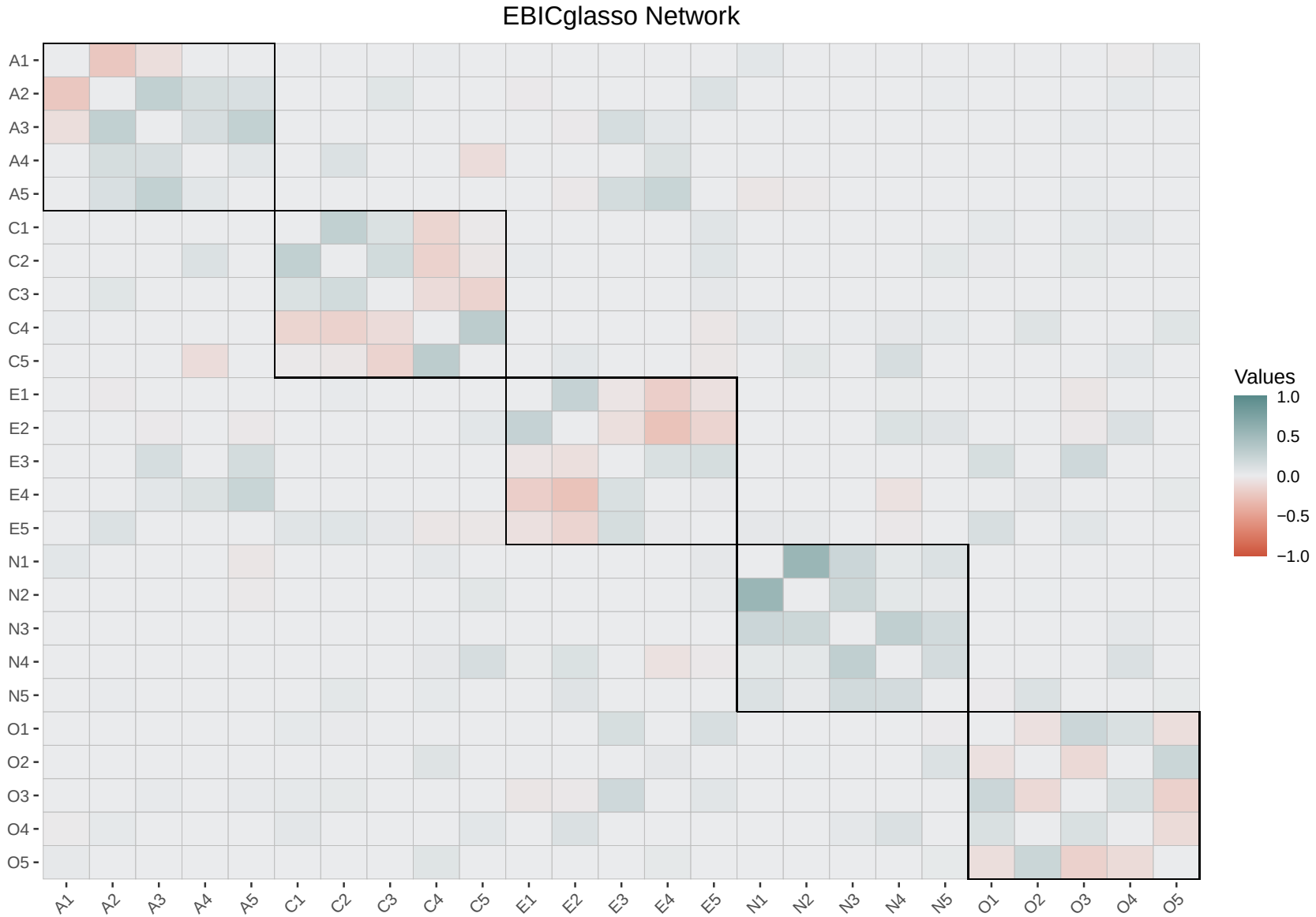
lambda = 0.48
EBIC = 67119.596



```r
# On the correlation matrix
bfi_network <- network.estimation(
  data = correlations, n = nrow(data), model = "glasso"
)

# On the data
bfi_network <- network.estimation(data, model = "glasso")
```


O5
O2
O3
O1
C1
O4
C2
E3
E1
C4
E5
E2
C3
E4
N5
C5
N4
A4
A5
A3
N3
N2
A2
N1
A1

Exploratory Graph Analysis | Estimate Network



Exploratory Graph Analysis

R Script

Community Detection

- There are many different metrics that can be applied to network to quantify them (graph theory)
- Community detection algorithms are used to identify *sets of connected nodes* that have *more connections within the set than between the set*
- In scales, these reflect "dimensions" or "factors" (consistent with PCA and factor analysis, respectively)

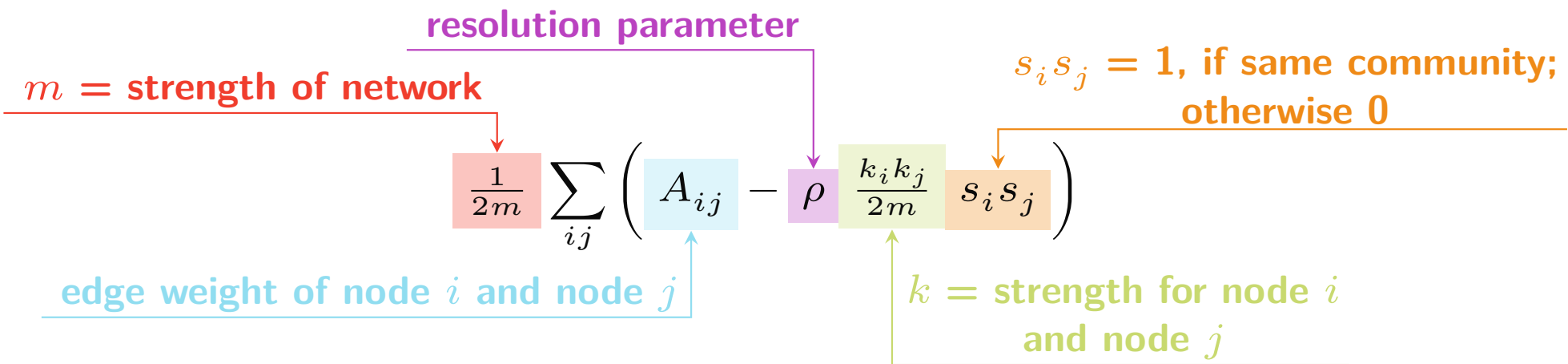
Common algorithms:

- **Walktrap**: uses hierarchical clustering to identify different clusters
- **Louvain**: uses local moves to maximize *modularity*
- Spinglass: uses statistical mechanics and annealing processes

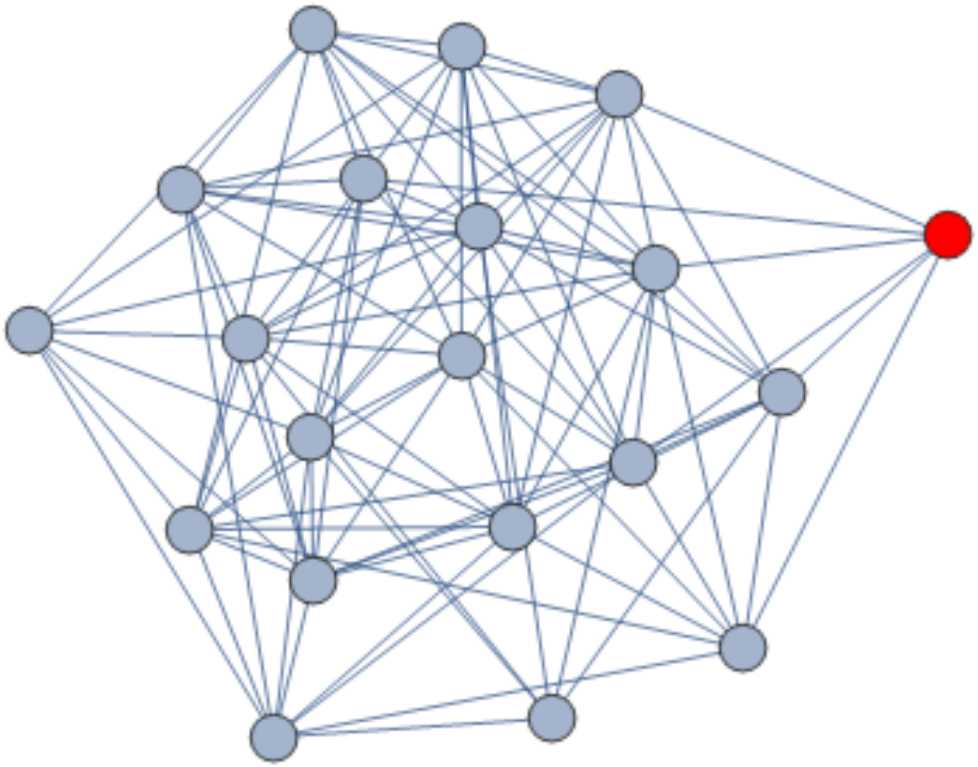
🔁 Most algorithms aim to maximize the number of connections *within* communities while minimizing the number of connections *between* communities

communities: sets of densely connected nodes (sometimes referred to as clusters)

modularity: metric to quantify the extent to which there are more within-community connections than between-community connections



Walktrap Algorithm



Rather than pursuing this process, the problem of the long-run expected walks can be obtained:

$$\mathbf{T}_{ij} = \frac{\mathbf{W}_{ij}}{\sum_{i=1}^{n} |w_i|}$$

where:

- $\mathbf{W}$ = network
- $\mathbf{T}$ = transition probability between node i and j
- $\sum_{i=1}^{n} |w_i|$ = *node strength* or absolute sum of a node's connections to all other nodes in the network

Convert transition matrix, $\mathbf{T}$, to distance metric

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(\mathbf{T}_{ik} - \mathbf{T}_{jk})^2}{\sum_{i=1}^{n} |w_i|}}$$

Convert transition matrix, $\mathbf{T}$, to distance metric

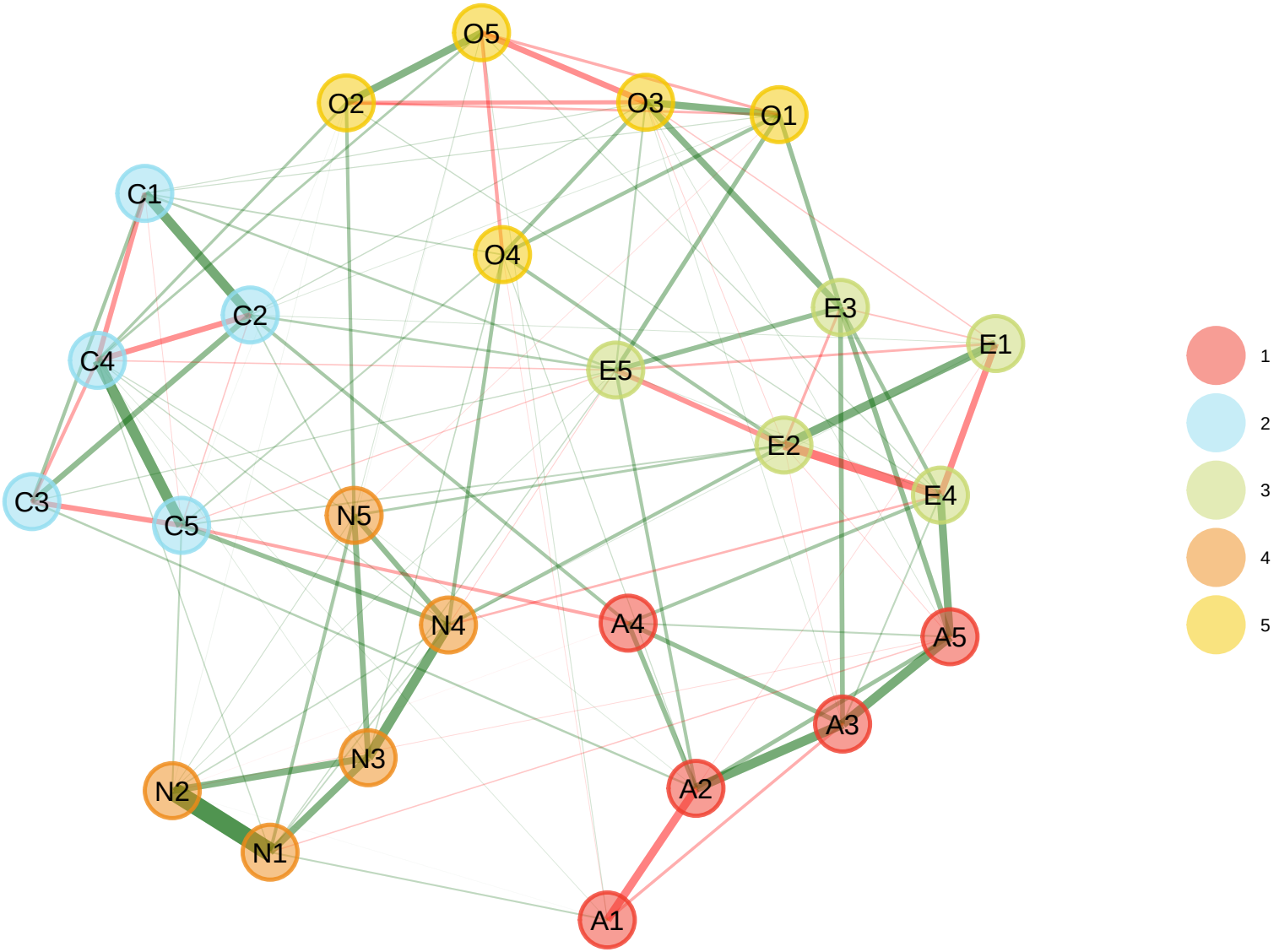
$$d_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(\mathbf{T}_{ik} - \mathbf{T}_{jk})^2}{\sum_{i=1}^{n} |w_i|}}$$

- Euclidean norm

Using the distance matrix, Ward's hierarchical clustering algorithm is applied and modularity is used to select the number of clusters

```
# Apply Walktrap algorithm
bfi_walktrap <- community.detection(bfi_network, algorithm = "walktrap")

# Print summary
summary(bfi_walktrap)
```

```
Algorithm:  Walktrap

Number of communities:  5

A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3 O4 O5
 1  1  1  1  1  2  2  2  2  2  3  3  3  3  3  4  4  4  4  4  5  5  5  5  5
```


O5
O2
O3
O1
C1
O4
E3
C2
E1
C4
E5
E2
E4
C3
C5
N5
N4
A4
A5
A3
N3
N2
A2
N1
A1
1
2
3
4
5

Louvain Algorithm

Iterative algorithm that takes multiple "passes" over merging the nodes in the network

1. For one node, identify the community that *maximizes* the gain in modularity
2. If there is a gain, then add that node to the community; otherwise, leave in current community
3. Repeat for each node

➔ This process constitutes one "pass"

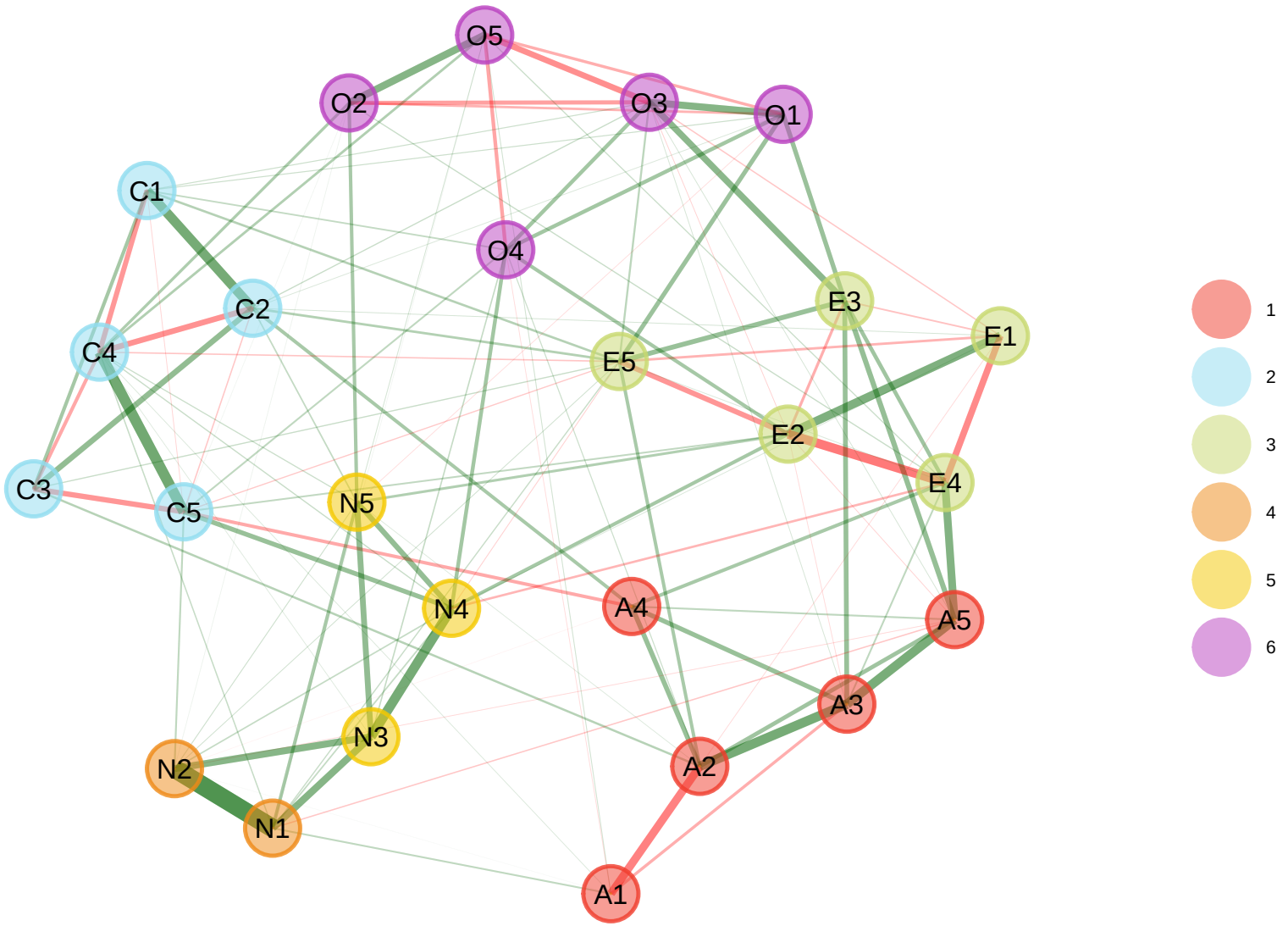
Louvain Algorithm

1. After a pass, "merge" nodes by summing the connections between nodes in their respective communities

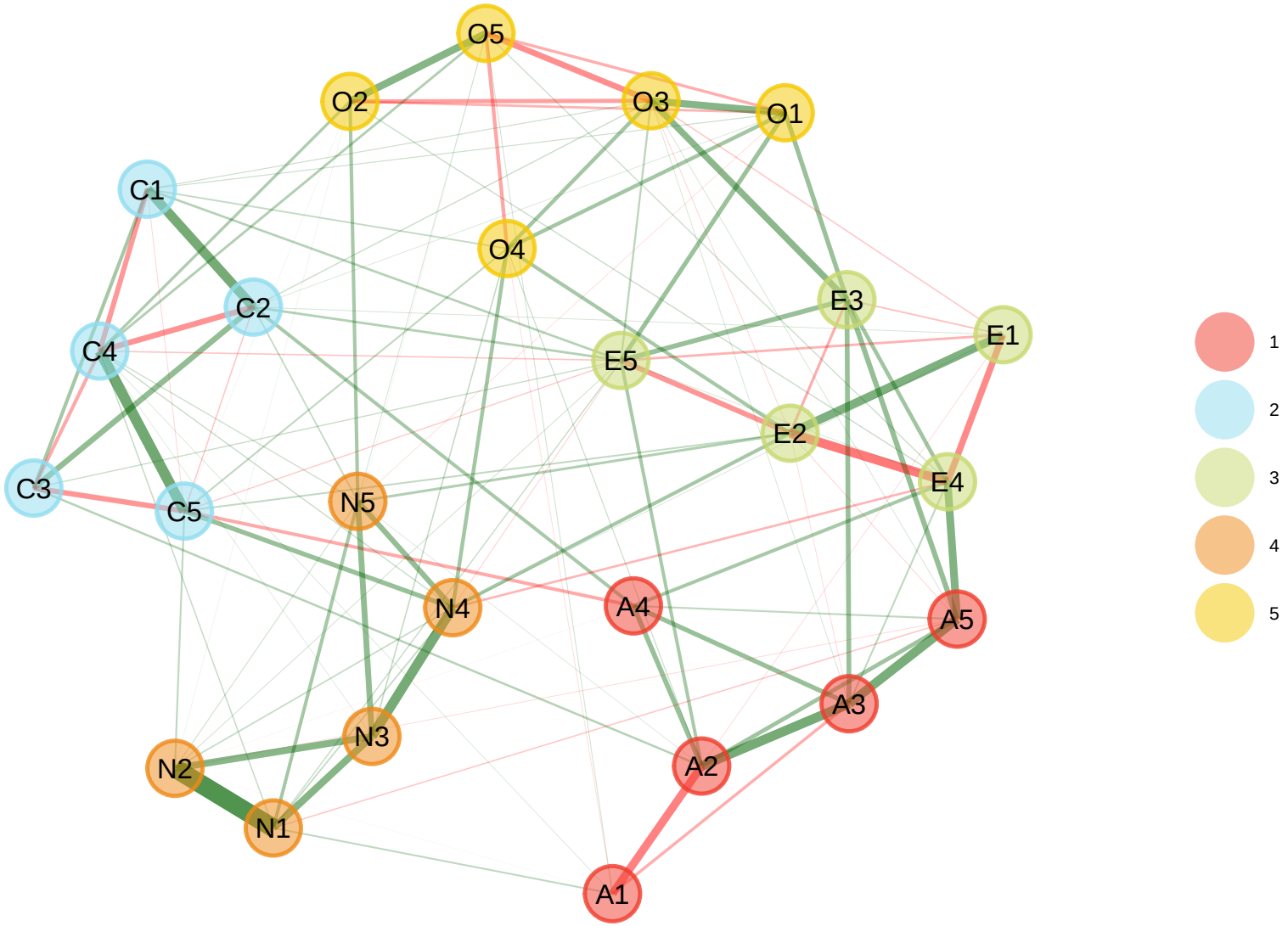
2. Repeat process until modularity cannot be increased *or* structure is unidimensional (all one community)

Because the passes start more granular and end broader, the algorithm is sometimes referred to as "multi-level" (we'll come back to this notion with hierarchical EGA)

First Pass


O5
O2
O3
O1
C1
O4
E3
C2
E1
C4
E5
E2
E4
C3
C5
N5
N4
A4
A5
A3
N3
N2
A2
N1
A1
1
2
3
4
5
6

Second Pass


O5
O2
O3
O1
C1
O4
E3
C2
E1
C4
E5
E2
C3
E4
N5
C5
N4
A4
A5
A3
N3
N2
A2
N1
A1
1
2
3
4
5

```r
# Apply Louvain algorithm
bfi_louvain <- community.detection(bfi_network, algorithm = "louvain")

# Print summary
summary(bfi_louvain)
```

```
Algorithm:  Louvain

Number of communities:  5

A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3 O4 O5
 1  1  1  1  1  2  2  2  2  2  3  3  3  3  3  4  4  4  4  4  5  5  5  5  5
```

Notes on the Louvain Algorithm

- Works sequentially, node-by-node, meaning node order can change the result
- {igraph}'s implementation (used in {EGAnet}) randomizes the order *outside* of the scope of `R`
- Other methods, such as Leiden, have improved on the original Louvain algorithm to avoid node order issues

A reproducible Louvain algorithm written in C code is in progress

Exploratory Graph Analysis

R Script

Exploratory Graph Analysis

Recap

1. Estimate associations
2. Estimate network
3. Apply community detection algorithm

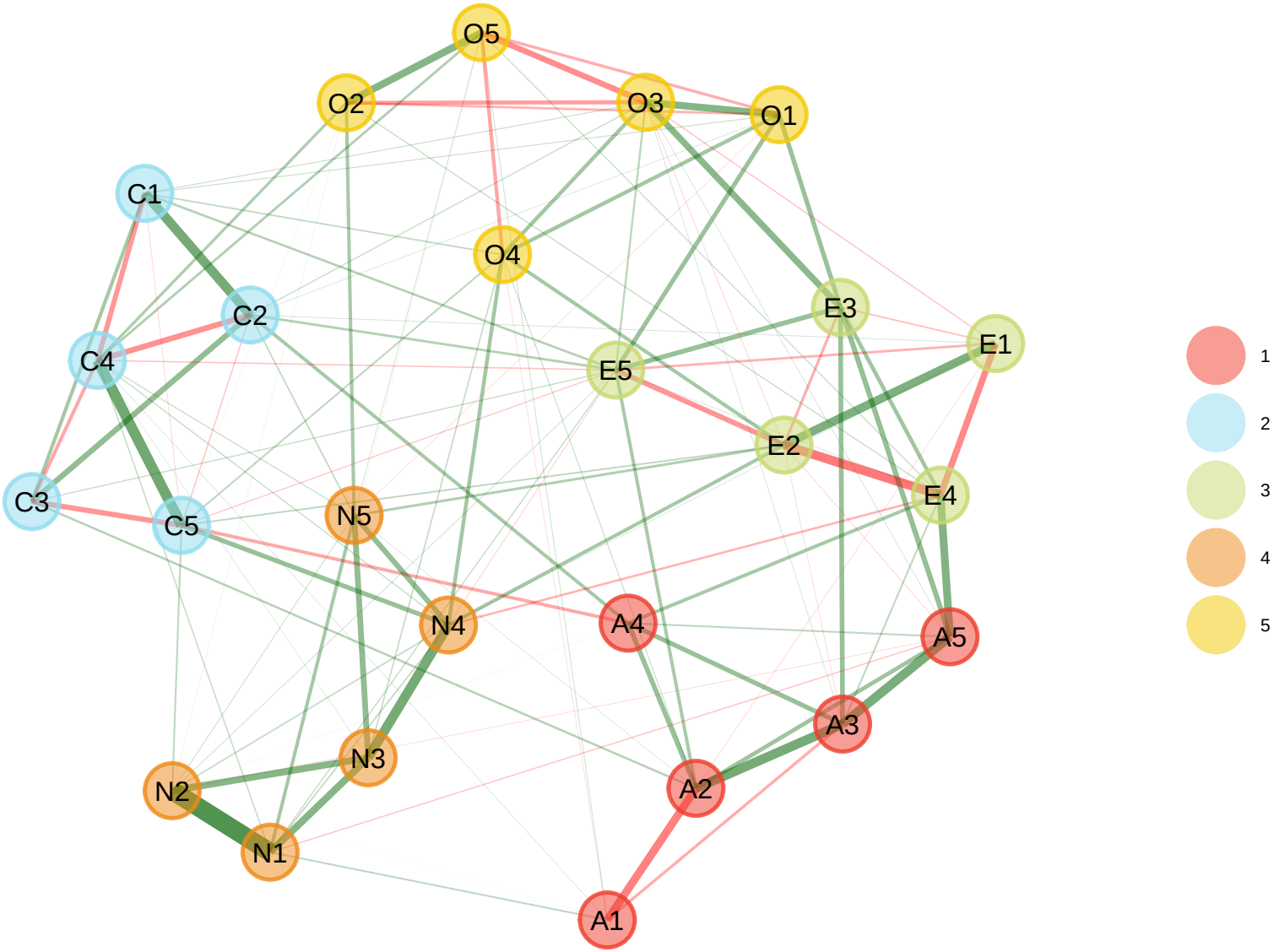
All-in-one Function

```
# Apply EGA
bfi_ega <- EGA(data)

# Print summary
summary(bfi_ega)

# Plot
plot(bfi_ega)
```

Exploratory Graph Analysis

```
Model: GLASSO (EBIC with gamma = 0.5)
Correlations: auto
Lambda: 0.0764652282008741 (n = 100, ratio = 0.1)

Number of nodes: 25
Number of edges: 117
Edge density: 0.390

Non-zero edge weights:
     M    SD    Min   Max
 0.046 0.119 -0.269 0.548

----

Algorithm:  Walktrap

Number of communities:  5

A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3 O4 O5
 1  1  1  1  1  2  2  2  2  2  3  3  3  3  3  4  4  4  4  4  5  5  5  5  5

----

Unidimensional Method: Louvain
Unidimensional: No

----

TEFI: -27.335
```

Exploratory Graph Analysis



Exploratory Graph Analysis

R Script

Unidimensionality

Unidimensionality

unidimensional: belonging to or representing a single dimension

If a measurement is unidimensional, then the assumption is that the measurement is capturing a single, unified *construct*

construct: our theoretical attribute that we measure that is *expected* to map onto some attribute that exists in the real-world

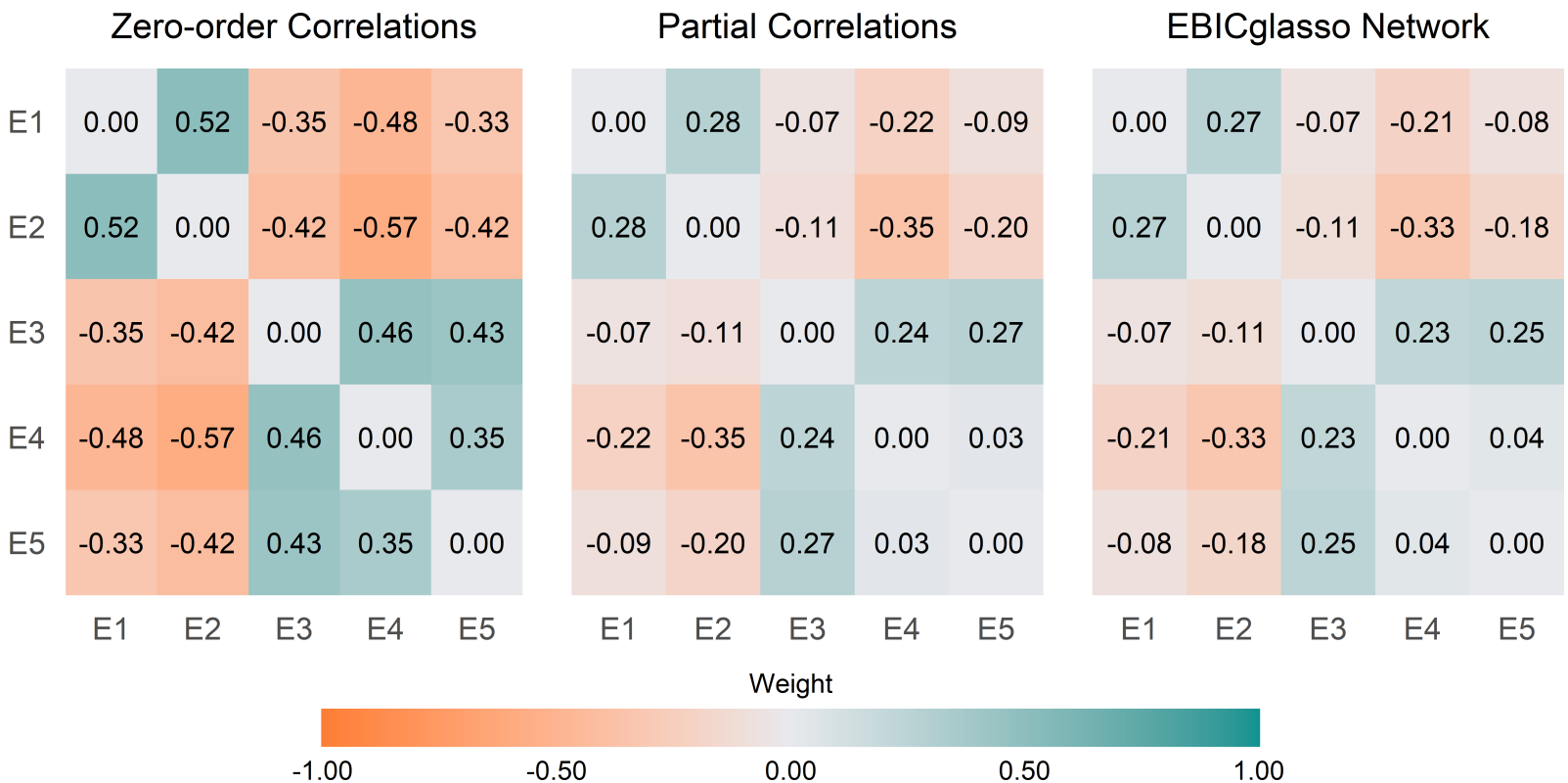
In essence, a construct is a proxy of something too difficult to directly measure through behavior

Unidimensionality

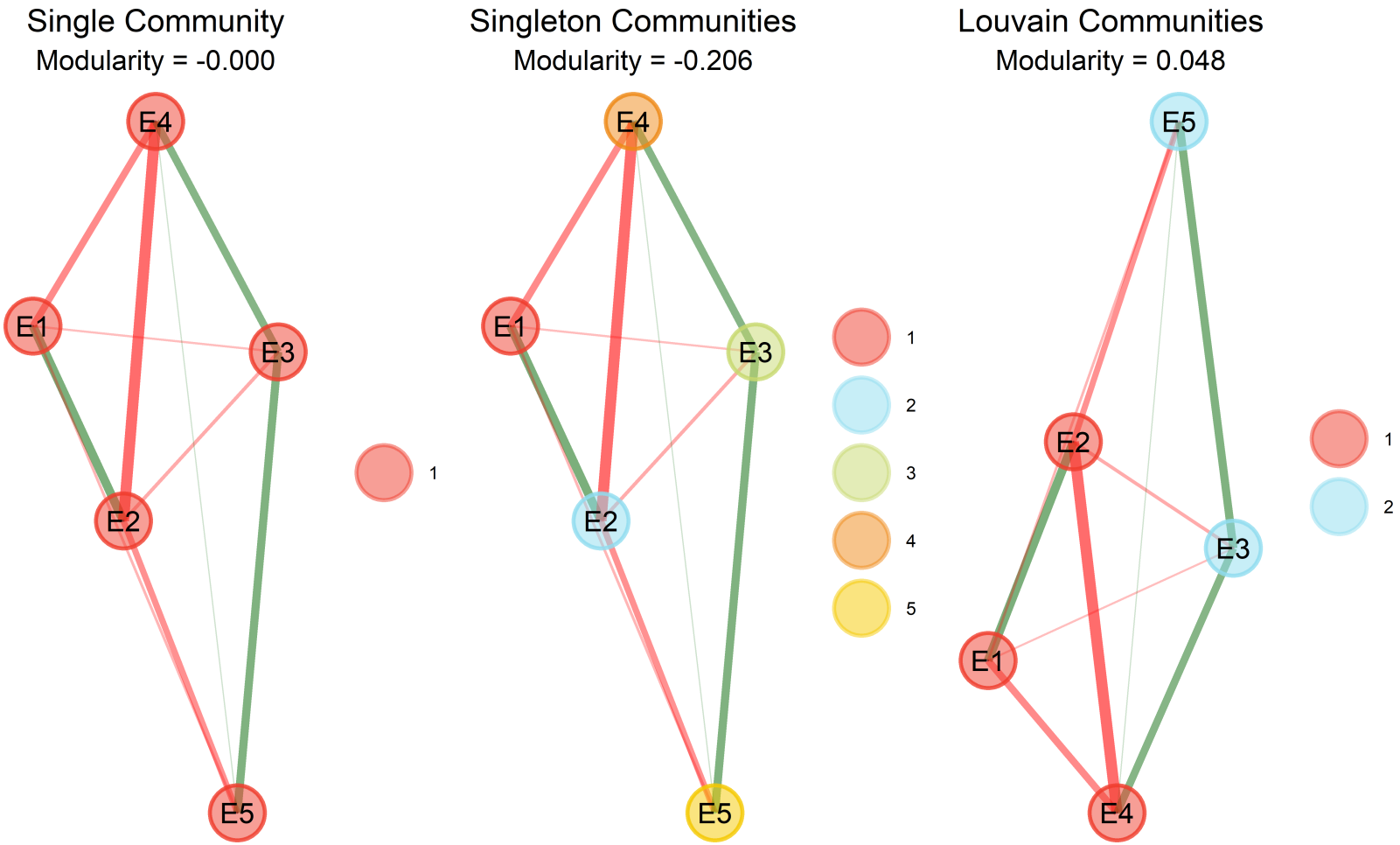
There are a few fundamental issues with networks and unidimensionality...

1. Partial correlations unevenly and unpredictably decrease relations between some variables more than others
2. Sparse networks are inherently modular due to the lack of edges between nodes
3. Modularity as a measure penalizes unidimensionality such that modularity equals zero (so almost any modular solution will be greater than one)

(Partial) Correlations

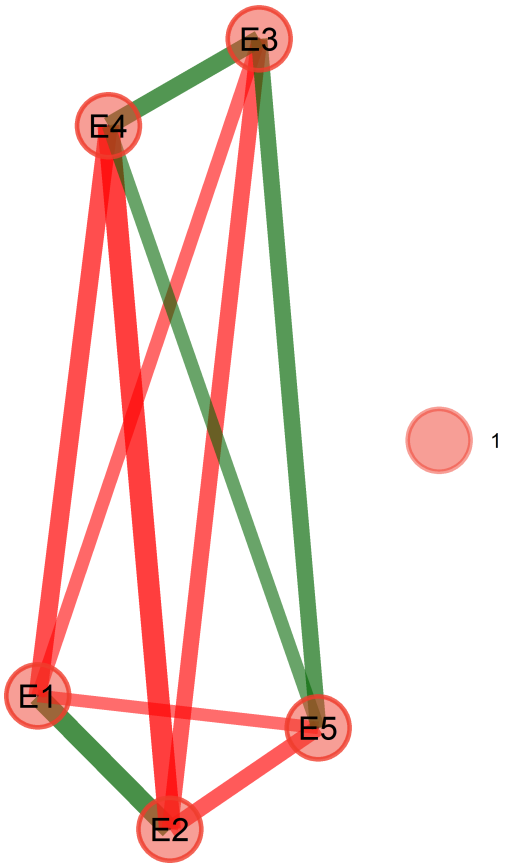


Sparsity and Modularity

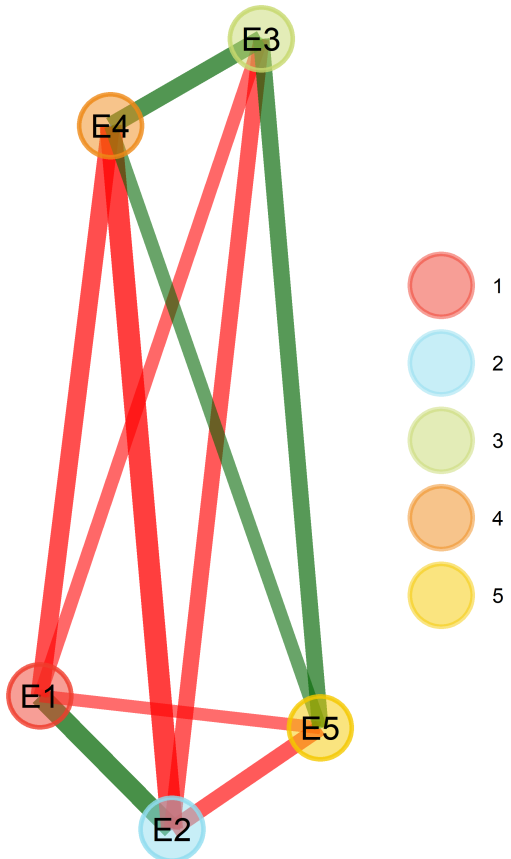


Zero-order Correlations

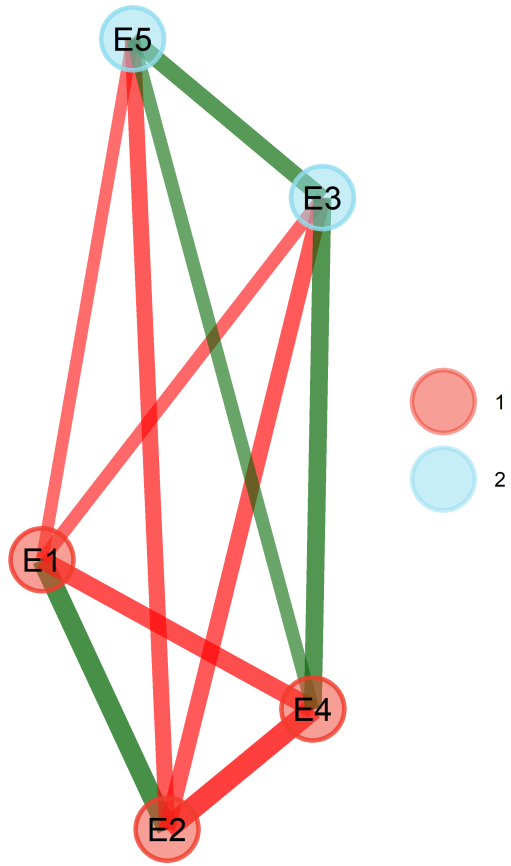
Single Community
Modularity = -0.000



Singleton Communities
Modularity = -0.201



Louvain Communities
Modularity = -0.073



Unidimensionality

```r
# Apply unidimensional
community.unidimensional(data[,grep("O", colnames(data))])
```

```
Algorithm:  Louvain

Number of communities:  1

O1 O2 O3 O4 O5
 1  1  1  1  1
```

Solution: apply community detection algorithm to the zero-order correlations

EGA under the hood

1. Estimate associations
2. Check for unidimensionality on associations
   a. If unidimensional, then stop
   b. If not unidimensional, then proceed
3. Estimate network
4. Apply community detection algorithm

Exploratory Graph Analysis

R Script

Total Entropy Fit Index

There are many community solution that can be achieved with different algorithms

Further, some algorithms have parameters that can be tuned (e.g., steps in Walktrap)

What solution should be used? What should the parameters be set to?

Total Entropy Fit Index provides an information theoretic approach to determine the best fitting solution

Entropy

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

Joint Entropy

$$H(X, Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x, y)$$

Conditional Entropy

$$H(Y|X) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) \log p(y|x)$$

Joint Entropy (reformulated)

$$H(X, Y) = H(X) + H(Y|X)$$

Total Correlation

$$C_{tot_x} = \left(\sum_{i=1}^{n} H(x_i) \right) - H(x_1, \ldots, x_n) \geq 0$$

Overall (inter)dependence of all variables

k-function

$$k(X_v, X_\omega) = n_1 H(X_v) + n_2 H(X_\omega) - (n_1 + n_2) H(X_v, X_\omega)$$

Difference of the average entropy of X_v and X_ω from the entropy of the super-set

Entropy Fit

$$EFI = \left[\frac{\sum_{i=1}^{N_F} H(S_{\eta_i})}{N_F} - H(S_{\eta_1}, \dots, S_{\eta_n})\right] + \left[\left(H_{max} - \frac{\sum_{i=1}^{N_F} H(S_{\eta_i})}{N_F}\right) \times \sqrt{N_F}\right]$$

Works directly on the values of the data

Von Neumann Entropy

$$S(\rho) = -\text{tr}(\rho \log \rho))$$

where

$$\rho = \frac{\mathbf{R}}{N}$$

where $\mathbf{R}$ is the correlation matrix and N is the number of variables

Von Neumann Entropy

Given ρ, its eigenvalues $\lambda_1, \ldots, \lambda_n \geq 0$ can be used to analytically solve for Von Neumann entropy such that

$$S(\rho) = -\text{tr}(\mathcal{L}(\mathbf{D}))$$

This approach is computationally efficient especially for large datasets

Total Entropy Fit Index

$$TEFI = \left[\frac{\sum_{i=1}^{N_F} S(\rho_i)}{N_F} - S(\rho)\right] + \left[\left(S(\rho) - \sum_{i=1}^{N_F} S(\rho_i)\right) \times \sqrt{N_F}\right]$$

Takeaways

- TEFI is a fast and efficient measure to estimate the fit of a dimensional solution
- Based on simulation studies, TEFI is as accurate or more accurate than more traditional measures commonly used in dimension reduction (e.g. ΔCFI, ΔRMSEA, ΔSRMR)
- Lower values = better solution

```r
# Apply EGA + TEFI with Walktrap
bfi_walktrap_fit <- EGA.fit(data, algorithm = "walktrap")

# Print summary
summary(bfi_walktrap_fit)
```

Exploratory Graph Analysis | TEFI

```
Model: GLASSO (EBIC with gamma = 0.5)
Correlations: auto
Lambda: 0.0764652282008741 (n = 100, ratio = 0.1)

Number of nodes: 25
Number of edges: 117
Edge density: 0.390

Non-zero edge weights:
     M    SD    Min   Max
 0.046 0.119 -0.269 0.548

----

Algorithm:  Walktrap (Steps = 3)

Number of communities:  5

A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3 O4 O5
 1  1  1  1  1  2  2  2  2  2  3  3  3  3  3  4  4  4  4  4  5  5  5  5  5


----

TEFI: -27.335
```

```r
# Apply EGA + TEFI with Louvain
bfi_louvain_fit <- EGA.fit(data, algorithm = "louvain")

# Print summary
summary(bfi_louvain_fit)
```

```
Model: GLASSO (EBIC with gamma = 0.5)
Correlations: auto
Lambda: 0.0764652282008741 (n = 100, ratio = 0.1)

Number of nodes: 25
Number of edges: 117
Edge density: 0.390

Non-zero edge weights:
     M    SD    Min   Max
 0.046 0.119 -0.269 0.548

----

Algorithm:  Louvain (Resolution = 0)

Number of communities:  5

A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3 O4 O5
 1  1  1  1  1  2  2  2  2  2  3  3  3  3  3  4  4  4  4  4  5  5  5  5  5


----

TEFI: -27.335
```

Summary

Summary

- EGA
  - estimate pairwise associations
  - check for unidimensionality
  - estimate network
  - apply community detection algorithm
- TEFI
  - grid search over community detection parameters
  - compare between multiple solutions (including theoretical and non-network solutions)

At Home Activity

At Home Activity

- Load `sapa.RData` data

- Apply `EGA` with Walktrap and Louvain
  - Report: number of communities, TEFI, and which algorithm fits better

- Apply `EGA.fit` with Walktrap and Louvain
  - Report: number of communities, TEFI, and which algorithm fits better

Theoretically, there are the Big Five factors

Readings for Next Week

Readings (Optional)

- Christensen and Golino - 2021 - bootEGA
- Christensen et al. - 2023 - UVA
- Christensen and Golino - 2021 - loadings
- Jamison et al. - 2022
- Jimenez et al. - 2023
- Samo et al. - 2023