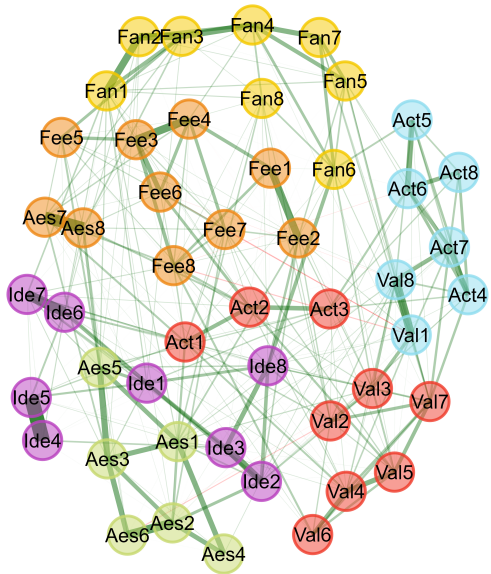
Exploratory Graph Analysis

DS-5740 Advanced Statistics



Overview: Week 10

Goals for the Week

- Learn about the inner workings of Exploratory Graph Analysis (EGA)
- Understand the graphical LASSO and model selection in EGA
- Cover different community detection algorithms

{EGAnet} Website

Dimension Reduction

Dimension Reduction

One of the first steps with survey data is to understand how variables are related and whether the measurement *appears* to be valid

Most often, we are measuring *something* and have some idea about how variables are related as well as whether they coalesce into higher-order measures

Few things we measure can be accurately captured with a single survey item (most things are *complex* and *multifaceted*)

Dimension reduction is usually the first step to understanding all of these things

Dimension Reduction

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

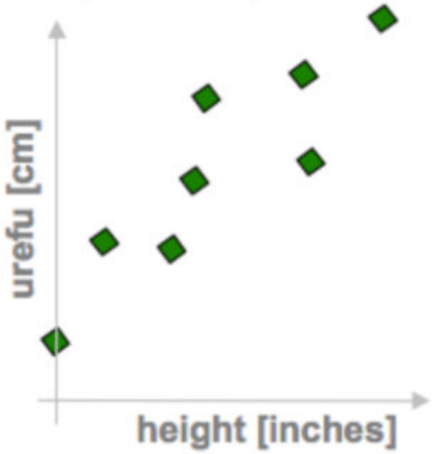
- Principal Component Analysis (PCA)
- Independent Component Analysis (ICA)
- Factor Analysis (FA)
- Exploratory Graph Analysis (EGA)

Principal Component Analysis

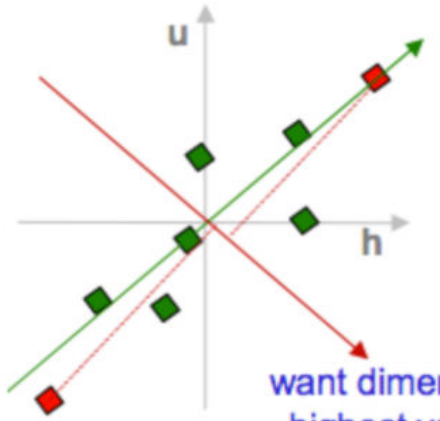
- Seeks to identify a linear combination of variables that maximizes variance on each consecutive component
- Each component is *orthogonal* (no correlations between components)
- Useful for creating clear and unique dimensions (but not necessary *valid*)

PCA in a nutshell

1. correlated hi-d data
("urefu" means "height" in Swahili)



2. center the points



3. compute covariance matrix

$$\begin{array}{c} \\ h \\ u \end{array}\begin{array}{c} \begin{array}{cc} h & u \end{array} \\ \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \end{array} \rightarrow \text{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

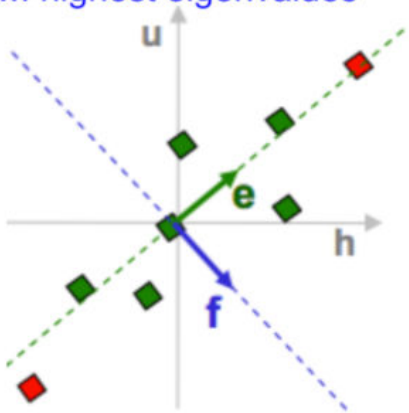
4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

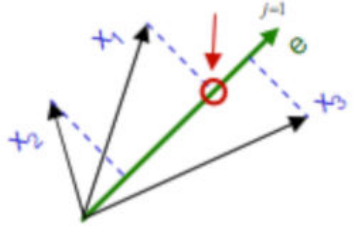
```
eig(cov(data))
```

5. pick m<d eigenvectors
w. highest eigenvalues

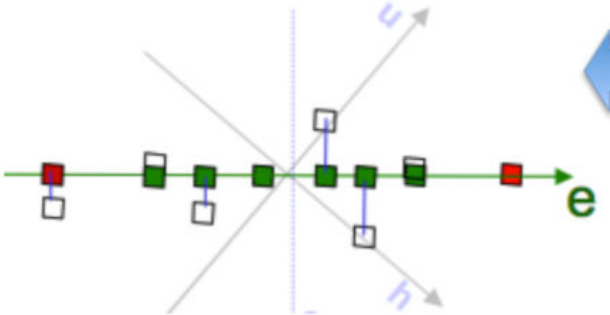


6. project data points to
those eigenvectors

$$x_e' = x^T e = \sum_{j=1}^{a} x_{ij} e_j$$



7. uncorrelated low-d data



Independent Component Analysis

- Similar to PCA, seeks to identify a linear combination of variables
- Each component is statistically *independent*
- Allows for nonlinear relationships and non-Gaussian variables
- Often used to separate signals in a multi-signal source (e.g., audio)

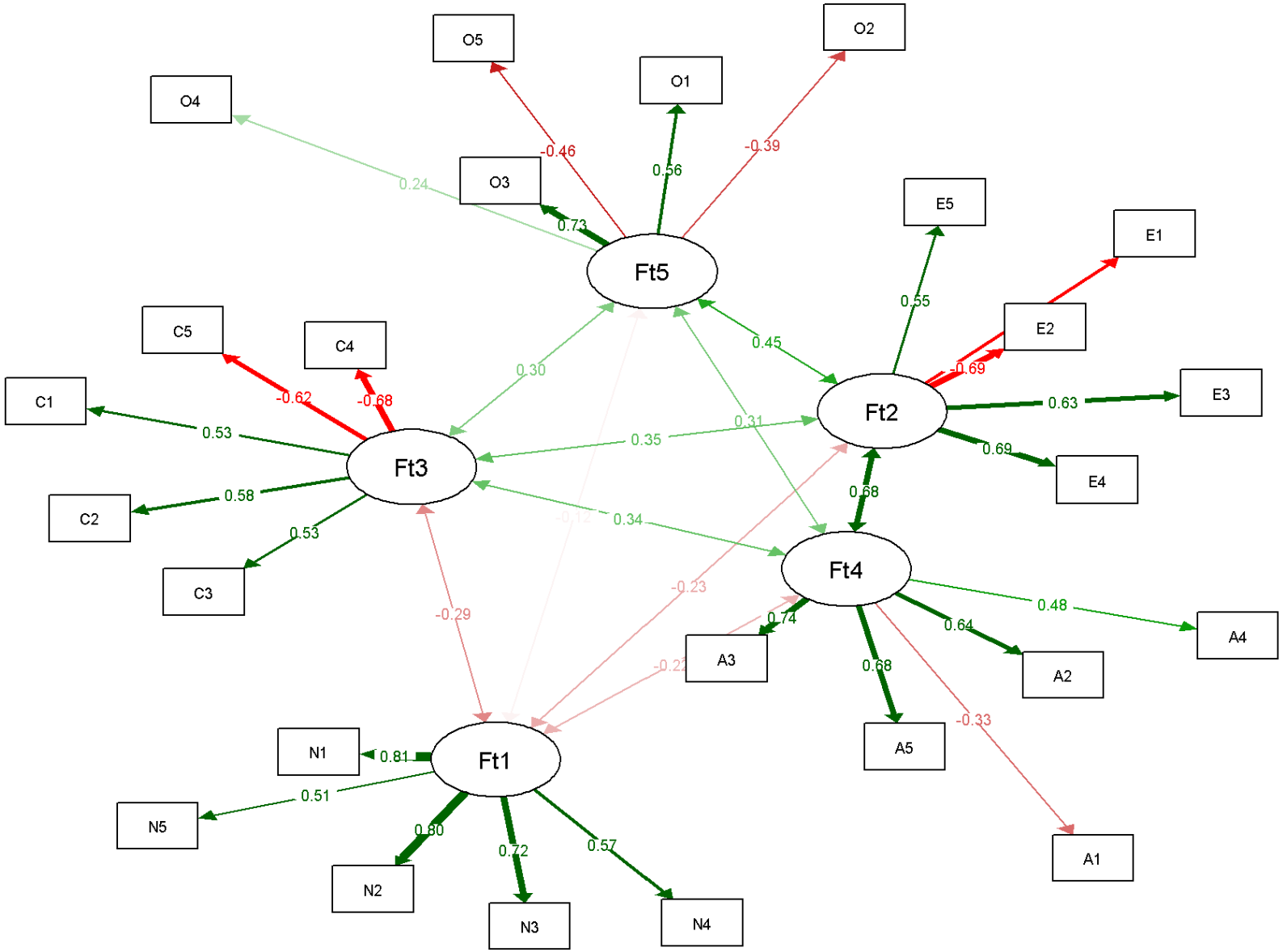
Factor Analysis

- 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

Factor Analysis

- Empirical example: Big Five Inventory (25 items)
- Five theoretical factors: openness to experience, conscientiousness, extraversion, agreeableness, neuroticism
- (Cor)related but separate factors

Dimension Reduction | Factor Analysis

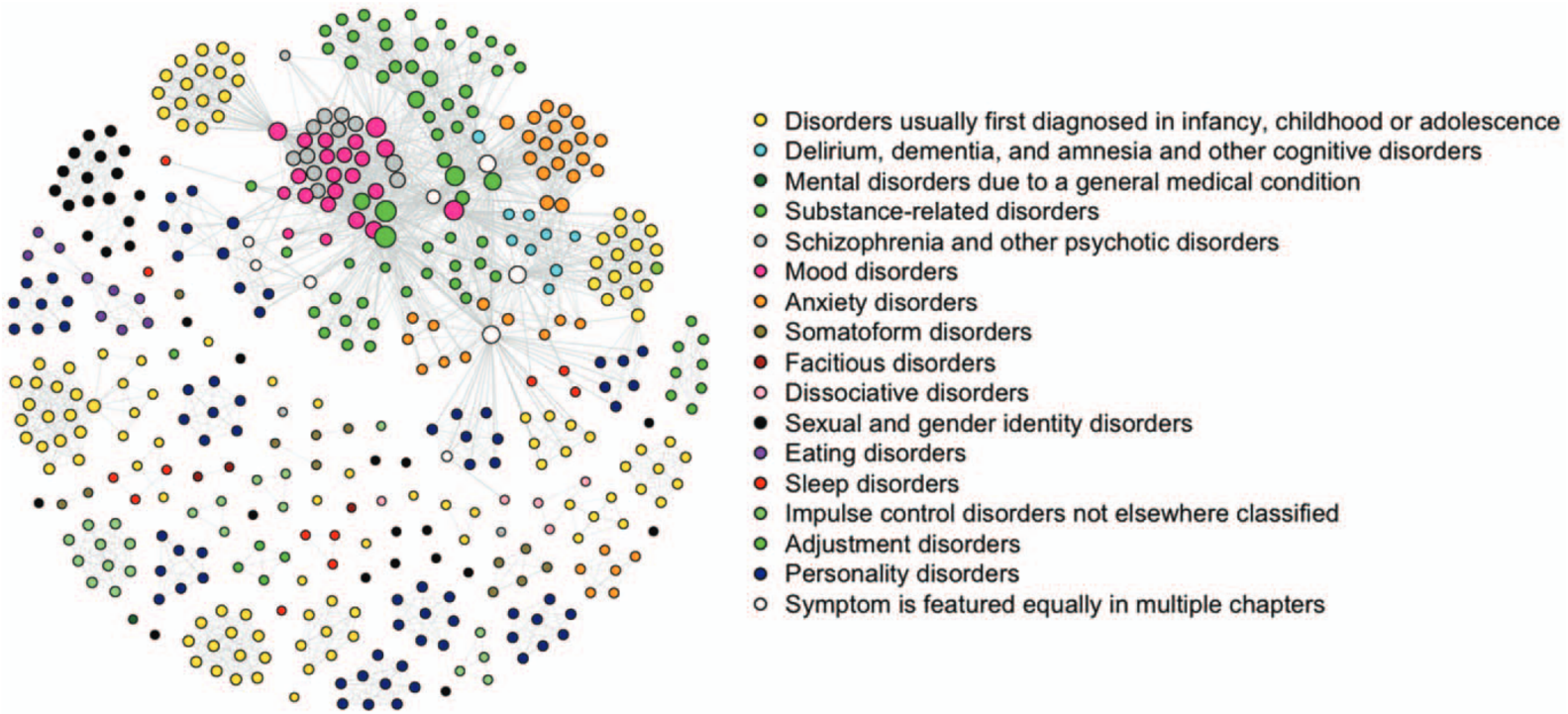


Exploratory Graph Analysis

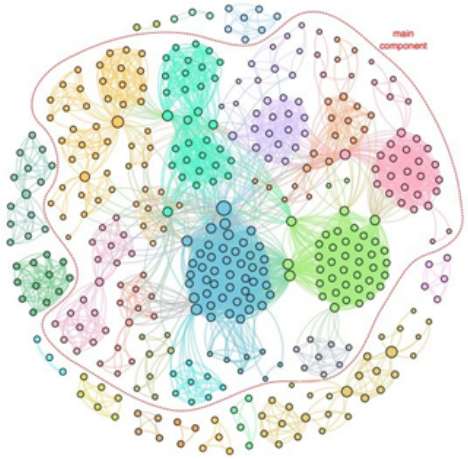
- Aims to model the conditional relationships between variables
- Uses regularized partial correlations and model selection to estimate model
- Applies community detection algorithm to identify "communities" or dimensions in a network

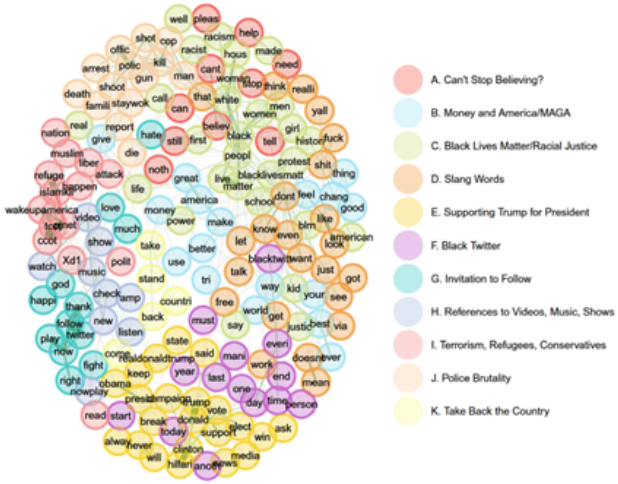
Network Analysis

Networks are everywhere



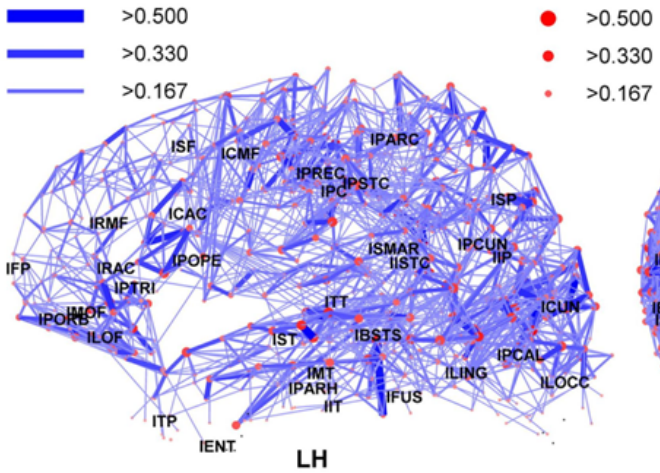
Networks can represent many different phenomena

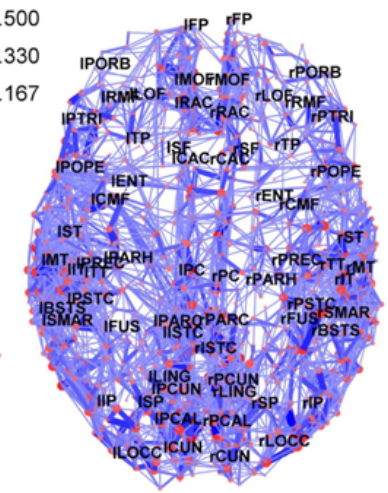


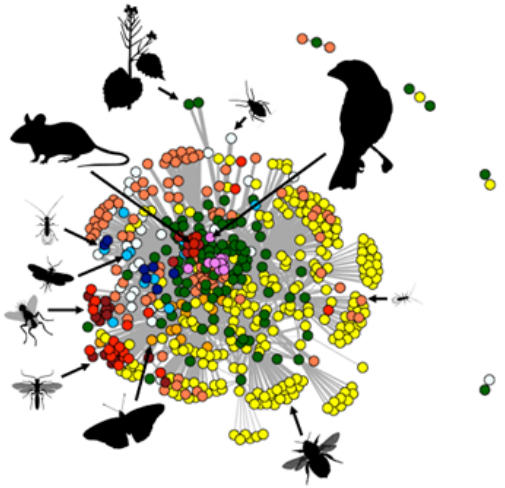


Political Corruption

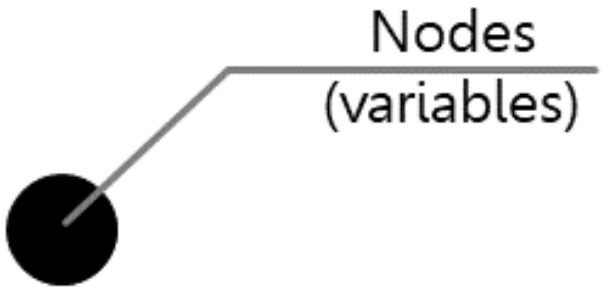
Twitter Data



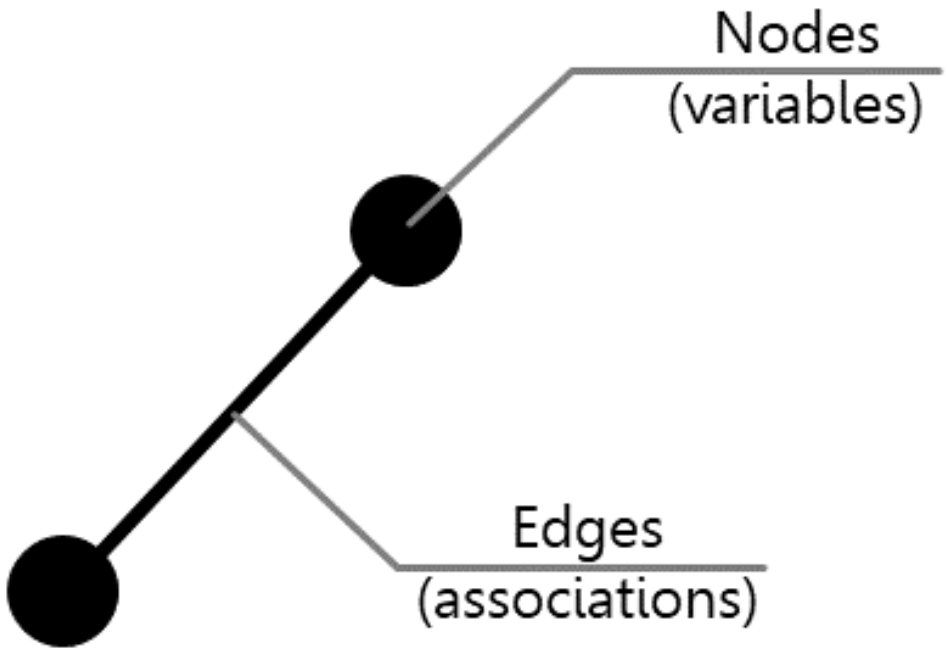




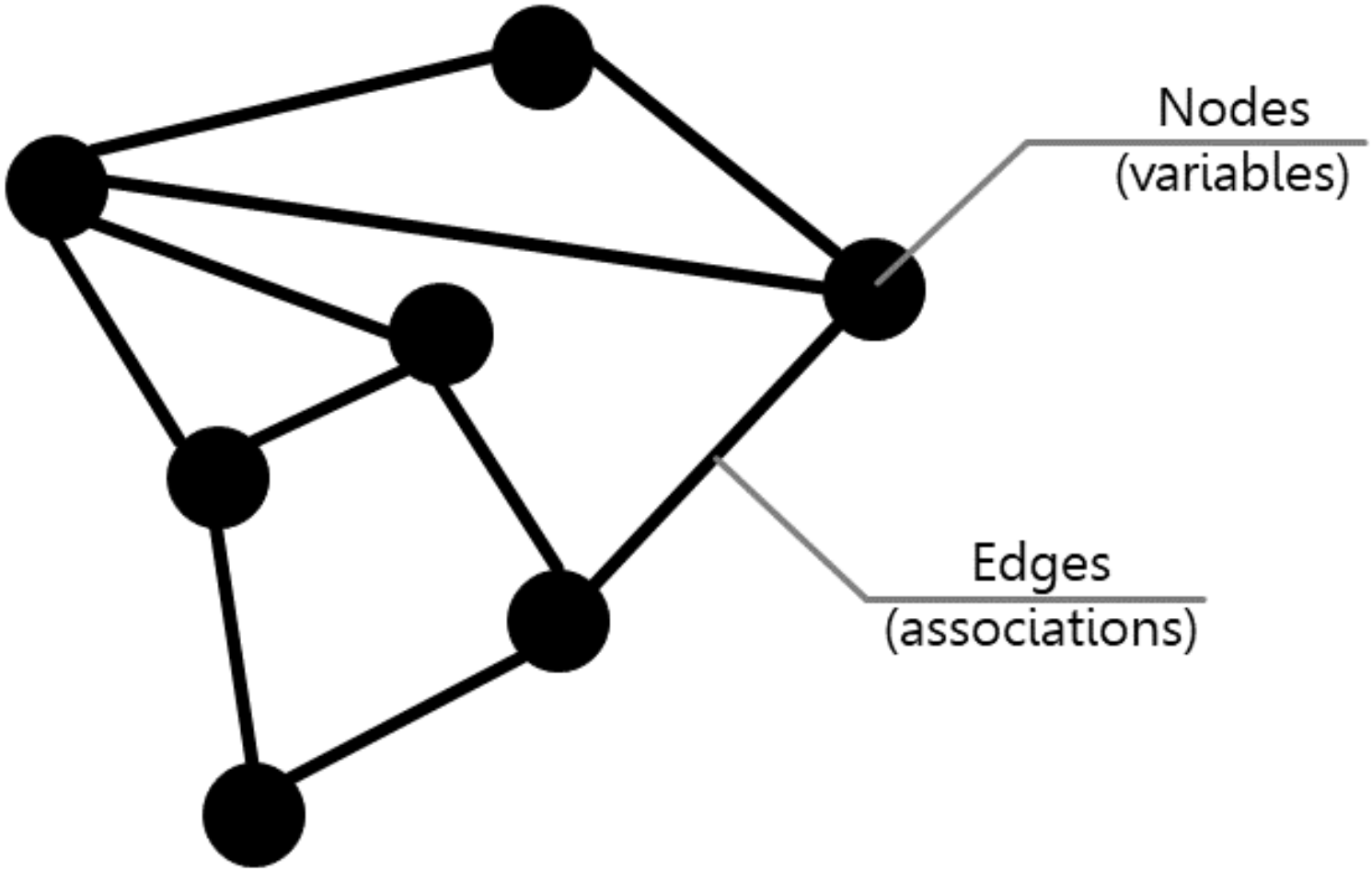
Breaking Down Networks



Breaking Down Networks



Breaking Down Networks



The data determine the interpretations of the network

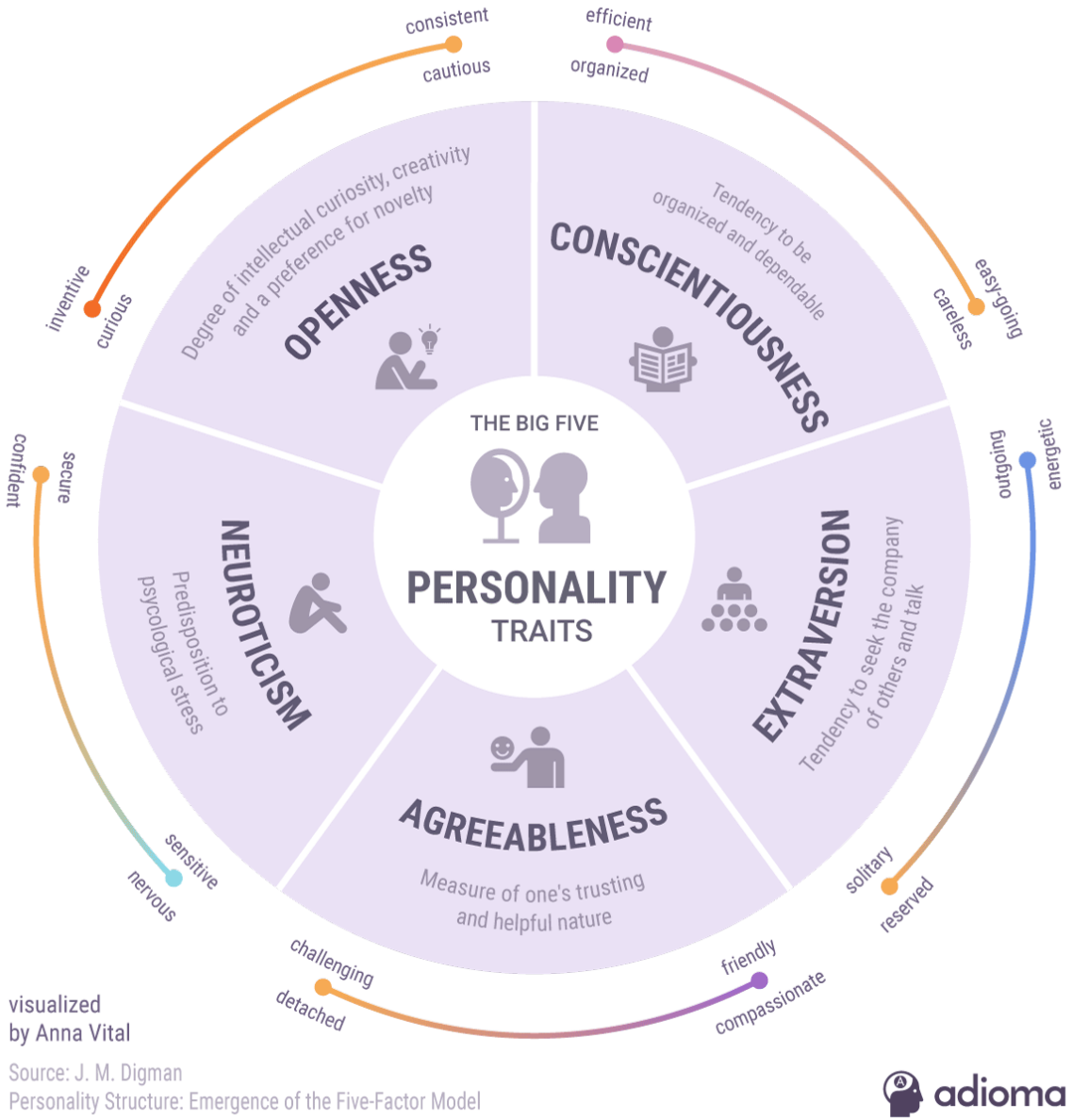
- social network: edges are friendships or relationships between people (nodes)
- brain network: edges are co-activations between brain regions (nodes)
- ecological network: edges are interactions between different species (nodes)
- **psychometric network**: edges are associations between variables (nodes)

The data determine the estimation of networks

- social network: people can directly report relationships
- brain network: co-activations can be directly measured (and correlated)
- ecological network: interactions between species can be observed
- **psychometric network**: associations between variables are *unknown!* and must be *estimated*

Empirical Example

Exploratory Graph Analysis | Empirical Example



Our Data

Openness to Experience

- Described as: creative, imaginative, intellectual, curious, open-minded
- Associated with: creative thinking, intelligence, humor, flexible thinking

Sample Characteristics

- Sample (N = 1588) from Brazil (n = 829) and Denmark (n = 926)
- Age: $M = 29.00, SD = 8.88, range = 18 - 70$
- Sex: 60% female; 40% male (other options were not provided)
- Between the years 2001-2011

Theoretical Structure

4 items each (24 items total)

- adventurousness: “prefer variety to routine”
- artistic interests: “believe in the importance of art”
- emotionality: “experience my emotions intensely”
- imagination: “have a vivid imagination”
- intellect: “have difficulty understanding abstract ideas” (reverse)
- liberalism: “believe that there is no absolute right or wrong”

All scored using 6-point Likert scale from 1 (Strongly Disagree) to 6 (Strongly Agree)

Exploratory Graph Analysis

Exploratory Graph Analysis

Steps

1. Estimate associations (e.g., correlations)
2. Estimate network (using an unsupervised learning algorithm)
3. Apply a community detection algorithm

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

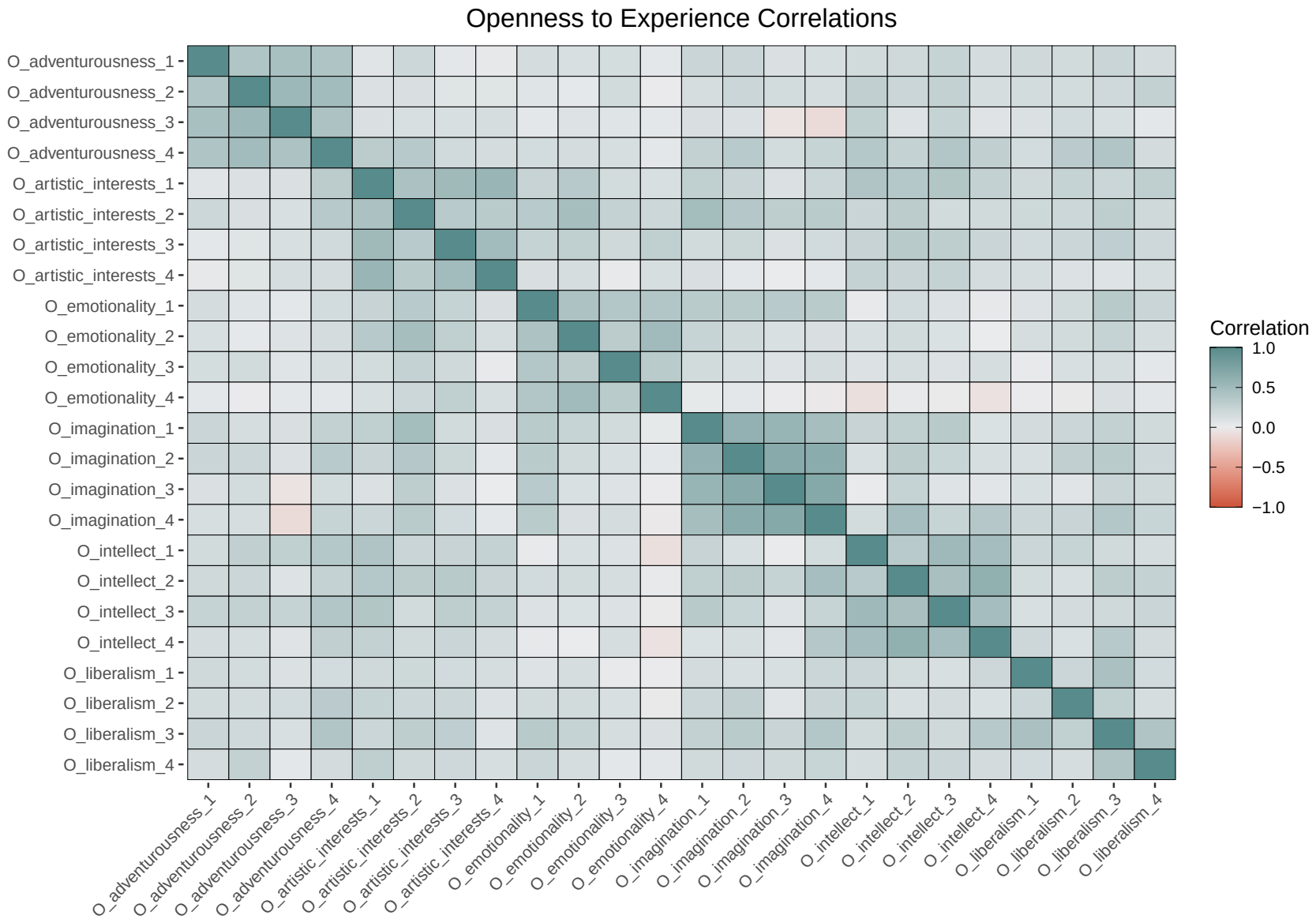
- **Ordinal data** (3-7 categories): Polychoric correlation
- Binary data (2 categories): Tetrachoric correlation
- Non-parametric: Spearman's rho

In {EGAnet}, `auto.correlate` will *automatically* compute the appropriate correlations for you

Use custom function `setup_ggplot2_heatmap` to set up correlation matrix (in script)

```r
# Load {EGAnet} and {ggplot2}
library(EGAnet); library(ggplot2)

# Load data
load("../data/openness_br_dk.RData")

# Select variables of interest
openness_voi <- openness_br_dk[, grep("O", colnames(openness_br_dk))]

# Reorder variables
openness_voi <- openness_voi[,order(colnames(openness_voi))]

# Set seed
set.seed(1)

# Subsample to increase variability
# (demonstration purposes only -- do not subsample on your own data!!)
openness_voi <- openness_voi[sample(1:nrow(openness_voi), 300),]

# Compute correlations
openness_corr <- auto.correlate(openness_voi)
```

Exploratory Graph Analysis | Estimate Associations



Estimate Network

Apply the graphical least absolute shrinkage and selection operator (GLASSO)

- Goal: Limit the number of *spurious* relationships between variables
- Attempts to maximizing the penalized log-likelihood function

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

- **K** = inverse covariance matrix ($\mathbf{R}^{-1}$)
- **S** = covariance matrix
- λ = penalty on the absolute sum of the parameters (controls *sparsity* or number of connections in the network)

Block Coordinate Descent

$$\begin{pmatrix} \kappa_{11} & \kappa_{12} \\ \kappa_{21} & \kappa_{22} \end{pmatrix}$$

- Partition the inverse covariance matrix where κ_{11} represents **K** except for the last row and column

- κ_{12} represents the partial correlations between the last variable and all other variables

$$\begin{pmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{pmatrix}$$

Block Coordinate Descent

- $\beta = -\frac{\kappa_{12}}{\kappa_{22}}$
- κ_{12} is regularized by minimizing:

$$min_{\beta}\{\frac{1}{2}||\mathbf{W}_{11}^{1/2}\beta - b||^2 + \rho||\beta||_1\}$$

- **W** = penalized covariance matrix
- We've turned regularization of the covariance matrix into a standard regularization problem

Block Coordinate Descent

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

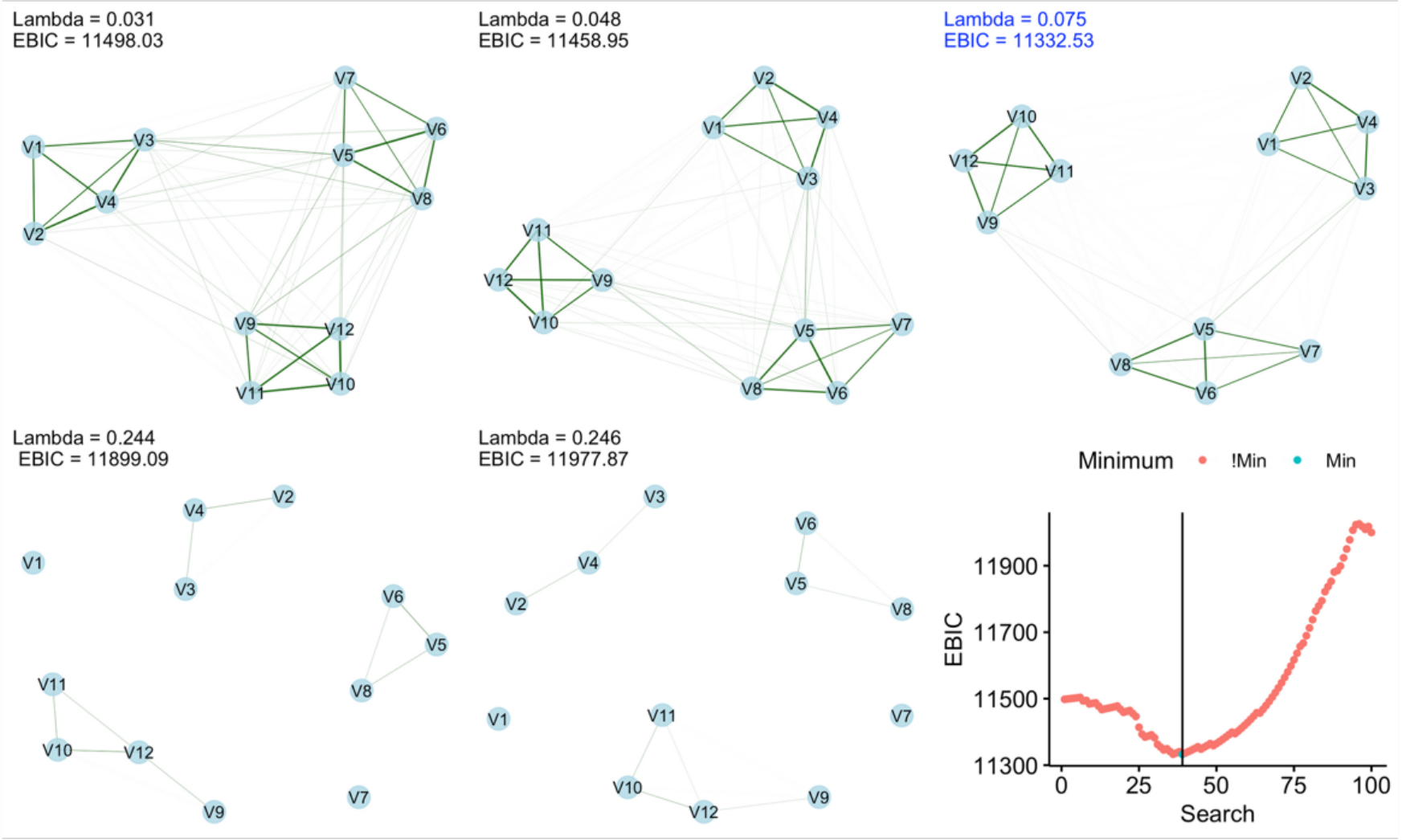
$$EBIC = -2L + E\log(N) + 4\gamma E\log(P)$$

- L = log-likelihood
- E = number of edges (connections)
- N = sample size
- P = 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 {EGAnet} = 0.01

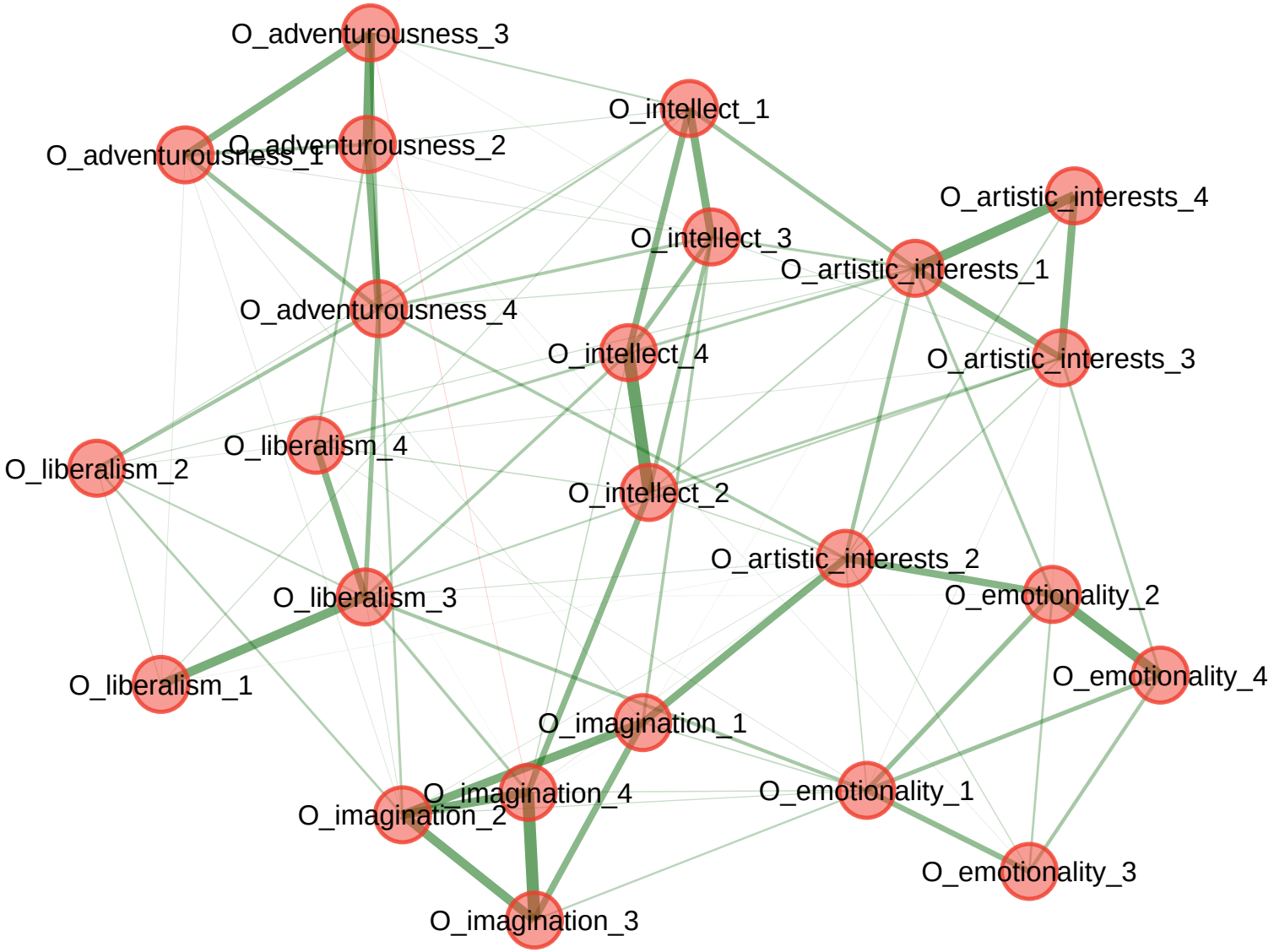
Exploratory Graph Analysis | Estimate Network



Exploratory Graph Analysis | Estimate Network

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

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

# Create class to plot
network_class <- list(
  network = openness_network,
  wc = rep(1, ncol(openness_network))
)
class(network_class) <- "EGA"

# Plot
plot(network_class) + theme(legend.position = "none")
```

Exploratory Graph Analysis | Estimate Network



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)

In social networks, these reflect actual friend groups, communities, real-world "clusters" of people

Common algorithms:

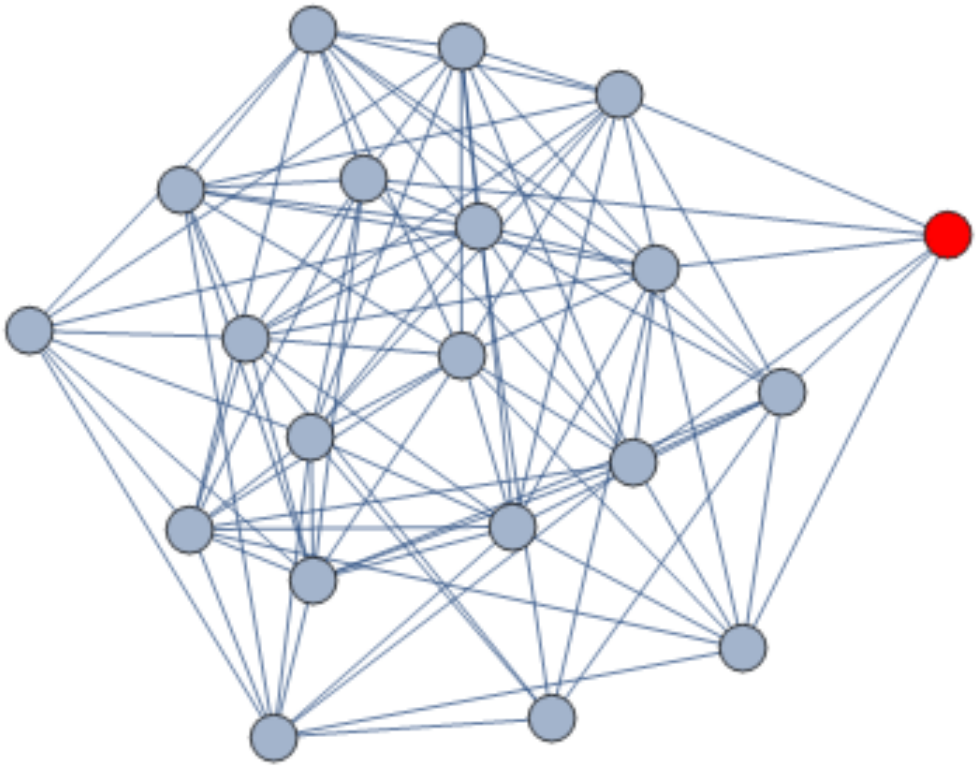
- **Walktrap**: uses hierarchical clustering to identify different clusters
- **Louvain**: uses local moves to maximize *modularity*
- Infomap: uses information theory to determine community cut-offs
- Spinglass: uses statistical mechanics and annealing processes

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

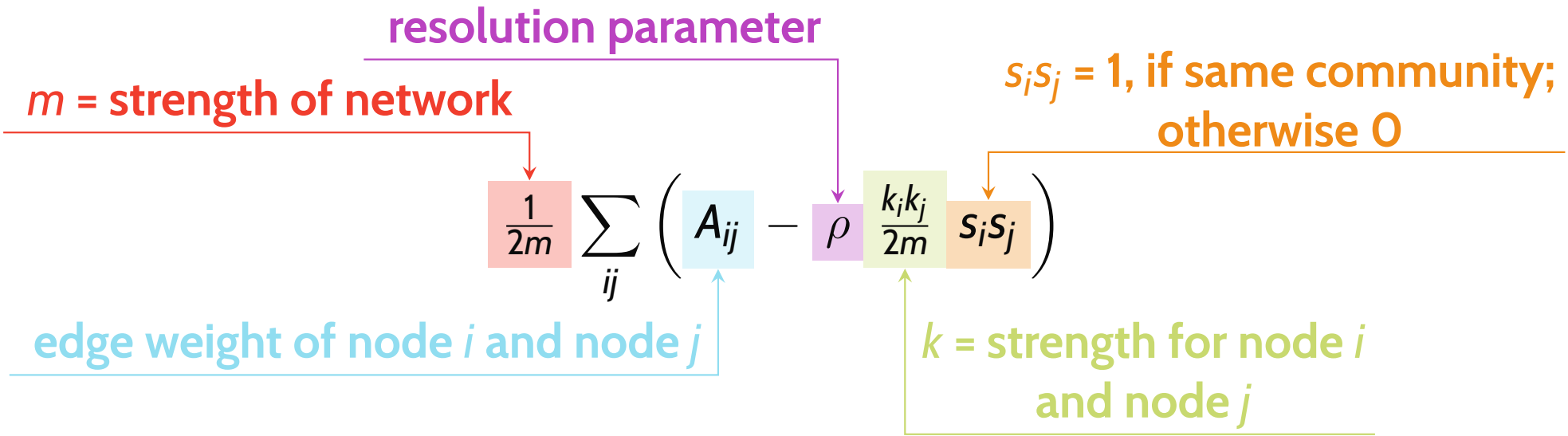


Walktrap Algorithm

- Performs "random walks" over the network
- Stronger (absolute) relations are more likely to "steps" in the walk
- The more often two nodes are used consecutively, the more likely they belong to the same community
- After, Ward's hierarchical clustering is applied to the "transition matrix"

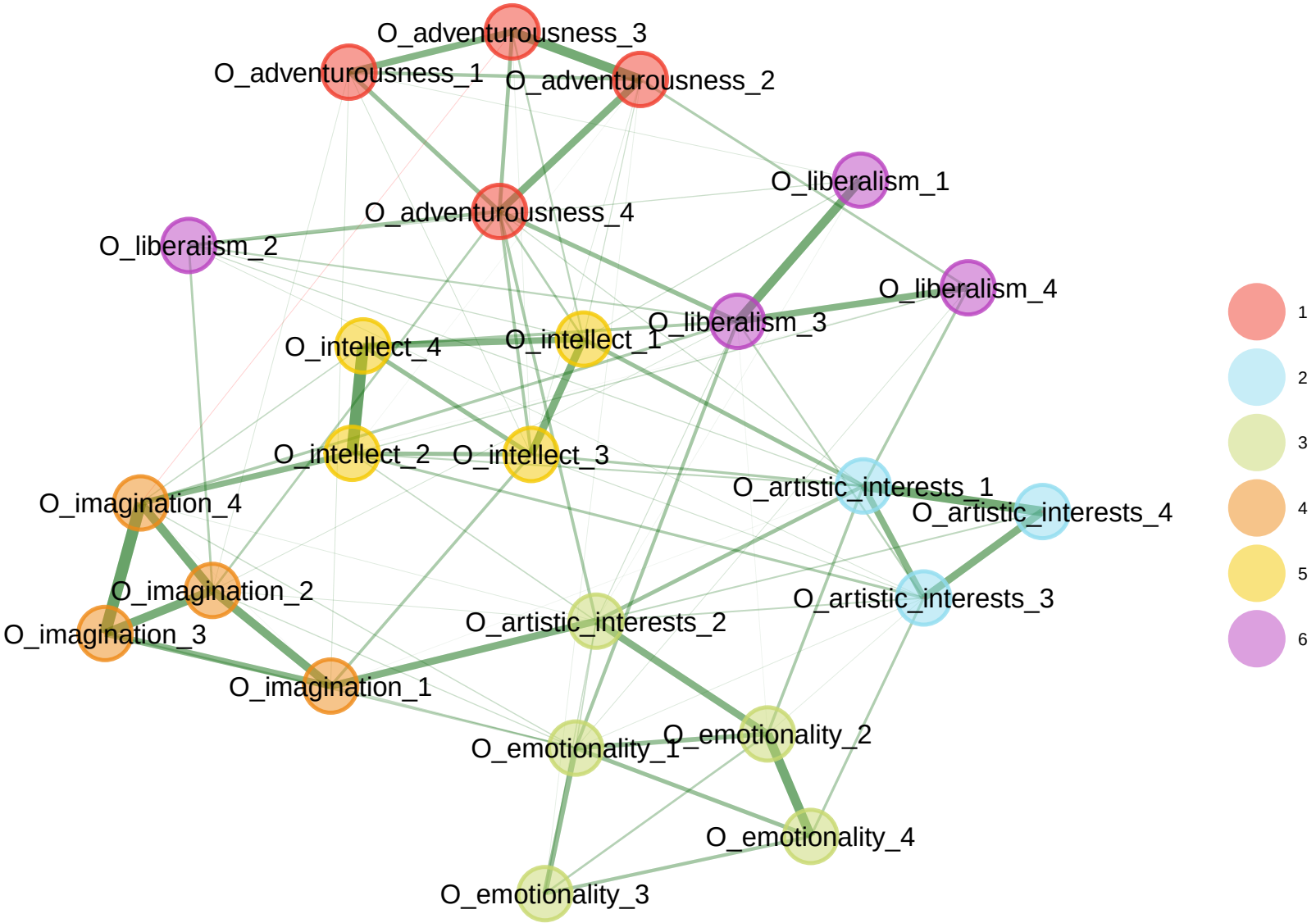
Walktrap Algorithm

- To select the proper number of clusters from the hierarchical clustering, modularity is used



```r
# Apply Walktrap to the network
walktrap_wc <- community.detection(
  openness_network, algorithm = "walktrap"
)

# Create class to plot
network_class <- list(
  network = openness_network,
  wc = walktrap_wc
)
class(network_class) <- "EGA"

# Plot
plot(network_class)
```

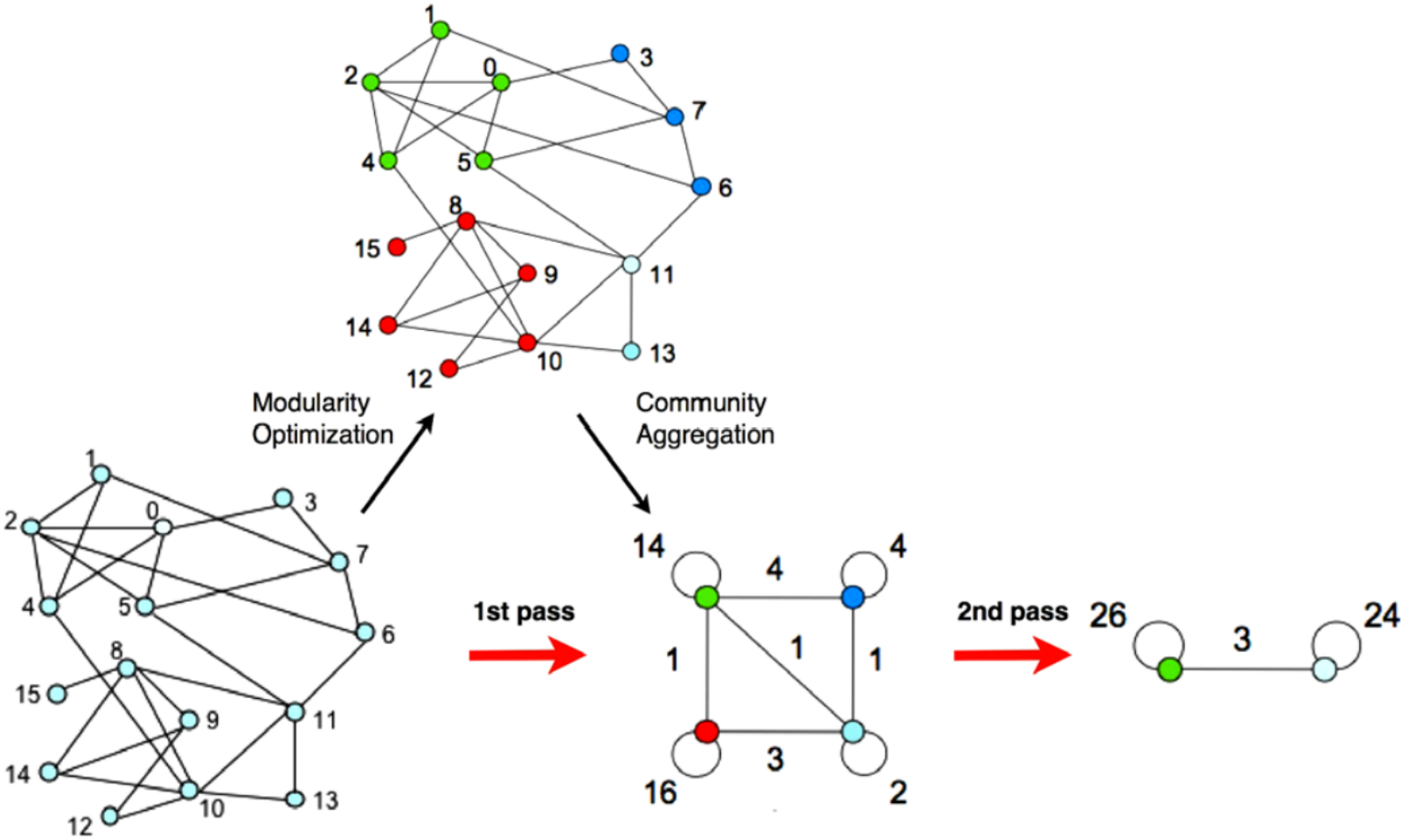
Exploratory Graph Analysis | Community Detection



Louvain Algorithm

- For each node, identify the community that *maximizes* the gain in modularity
- If there is a gain, then add that node to the community; otherwise, leave in current community
- Repeat for each node
- "Merge" nodes by summing the connections between nodes in their respective communities
- Repeat process until modularity cannot be increase *or* structure is unidimensional (all one community)

Louvain Algorithm



Louvain Algorithm

One limitation of the Louvain algorithm is that it is stochastic and depends on node ordering

	Run_1	Run_2	Run_3
O_adventurousness_1	1	1	1
O_adventurousness_2	1	1	1
O_adventurousness_3	1	1	1
O_adventurousness_4	1	1	1
O_artistic_interests_1	2	2	2
O_artistic_interests_2	3	3	3
O_artistic_interests_3	2	2	2
O_artistic_interests_4	2	2	2
O_emotionality_1	3	3	3
O_emotionality_2	3	3	3
O_emotionality_3	3	3	3
O_emotionality_4	3	3	3
O_imagination_1	4	4	4
O_imagination_2	4	4	4
O_imagination_3	4	4	4
O_imagination_4	4	4	4
O_intellect_1	2	5	1
O_intellect_2	2	5	1
O_intellect_3	2	5	1
O_intellect_4	2	5	1
O_liberalism_1	5	6	5
O_liberalism_2	5	6	5
O_liberalism_3	5	6	5
O_liberalism_4	5	6	5

Consensus Clustering

To avoid this issue, a technique called *consensus clustering* can be applied

- Repeatedly (e.g., 1000 times) apply community detection algorithm while shuffling node order (network *does not* change)

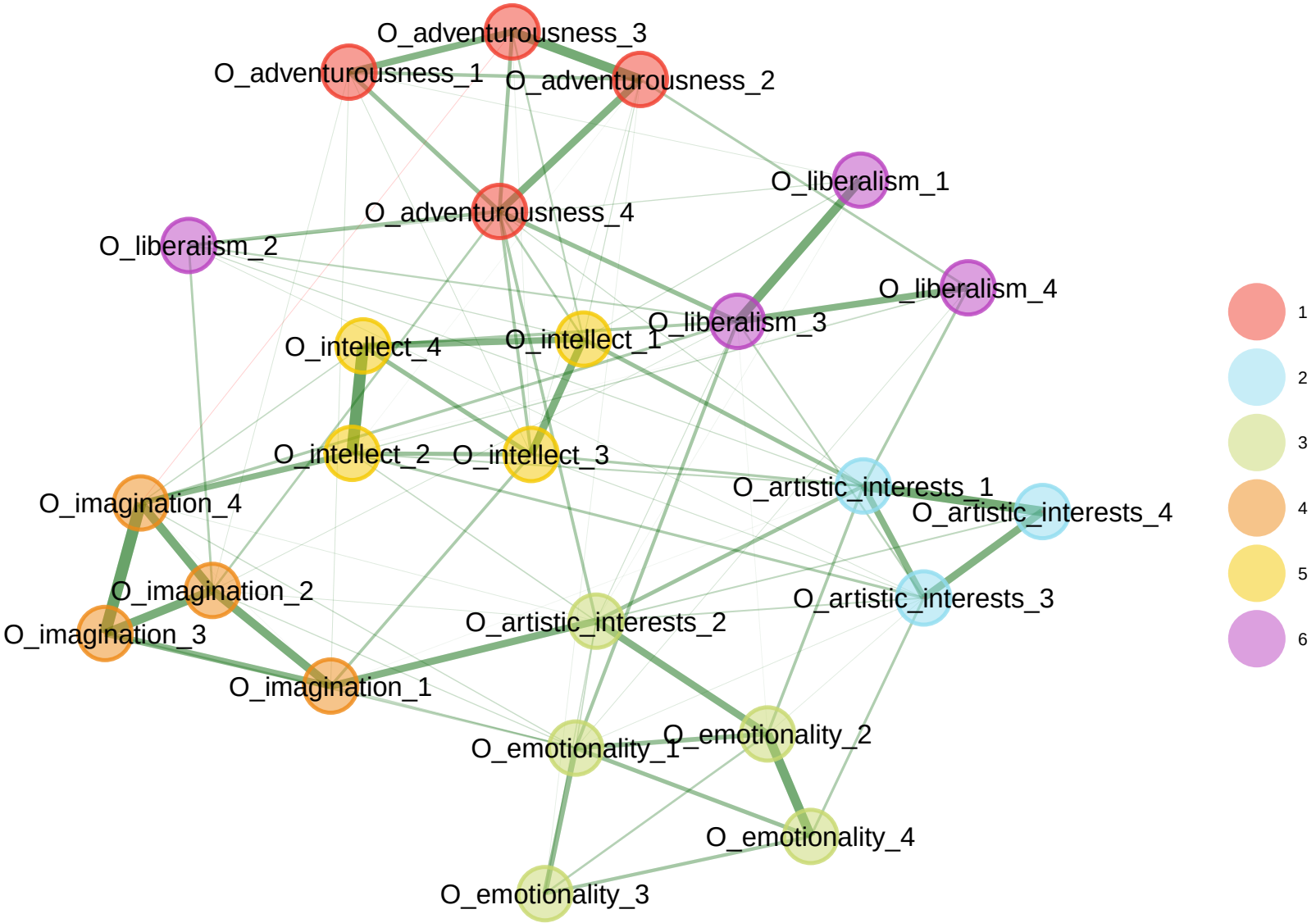
- Most efficient and generally accurate approach: accept the most common solution

Consensus Clustering

	Run_1	Run_2	Run_3
O_adventurousness_1	1	1	1
O_adventurousness_2	1	1	1
O_adventurousness_3	1	1	1
O_adventurousness_4	1	1	1
O_artistic_interests_1	2	2	2
O_artistic_interests_2	3	3	3
O_artistic_interests_3	2	2	2
O_artistic_interests_4	2	2	2
O_emotionality_1	3	3	3
O_emotionality_2	3	3	3
O_emotionality_3	3	3	3
O_emotionality_4	3	3	3
O_imagination_1	4	4	4
O_imagination_2	4	4	4
O_imagination_3	4	4	4
O_imagination_4	4	4	4
O_intellect_1	5	5	5
O_intellect_2	5	5	5
O_intellect_3	5	5	5
O_intellect_4	5	5	5
O_liberalism_1	6	6	6
O_liberalism_2	6	6	6
O_liberalism_3	6	6	6
O_liberalism_4	6	6	6

```r
# Apply Louvain to the network
louvain_wc <- community.consensus(
  openness_network, algorithm = "Louvain"
)

# Create class to plot
network_class <- list(
  network = openness_network,
  wc = louvain_wc
)
class(network_class) <- "EGA"

# Plot
plot(network_class)
```

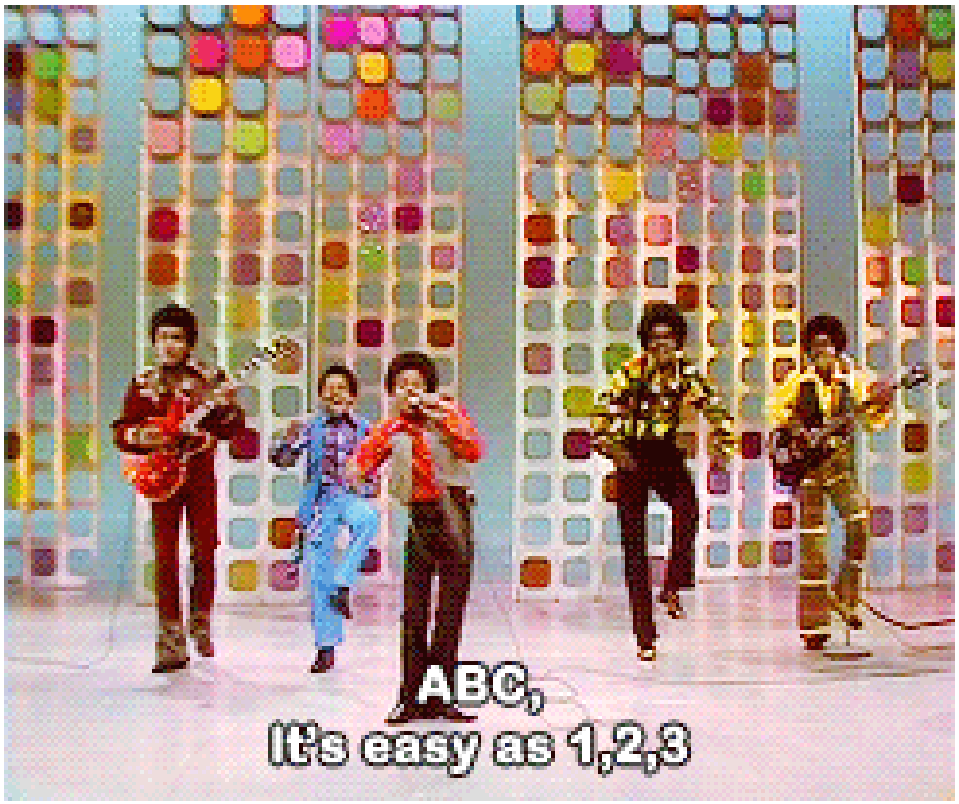
Exploratory Graph Analysis | Community Detection



Exploratory Graph Analysis

Recap

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

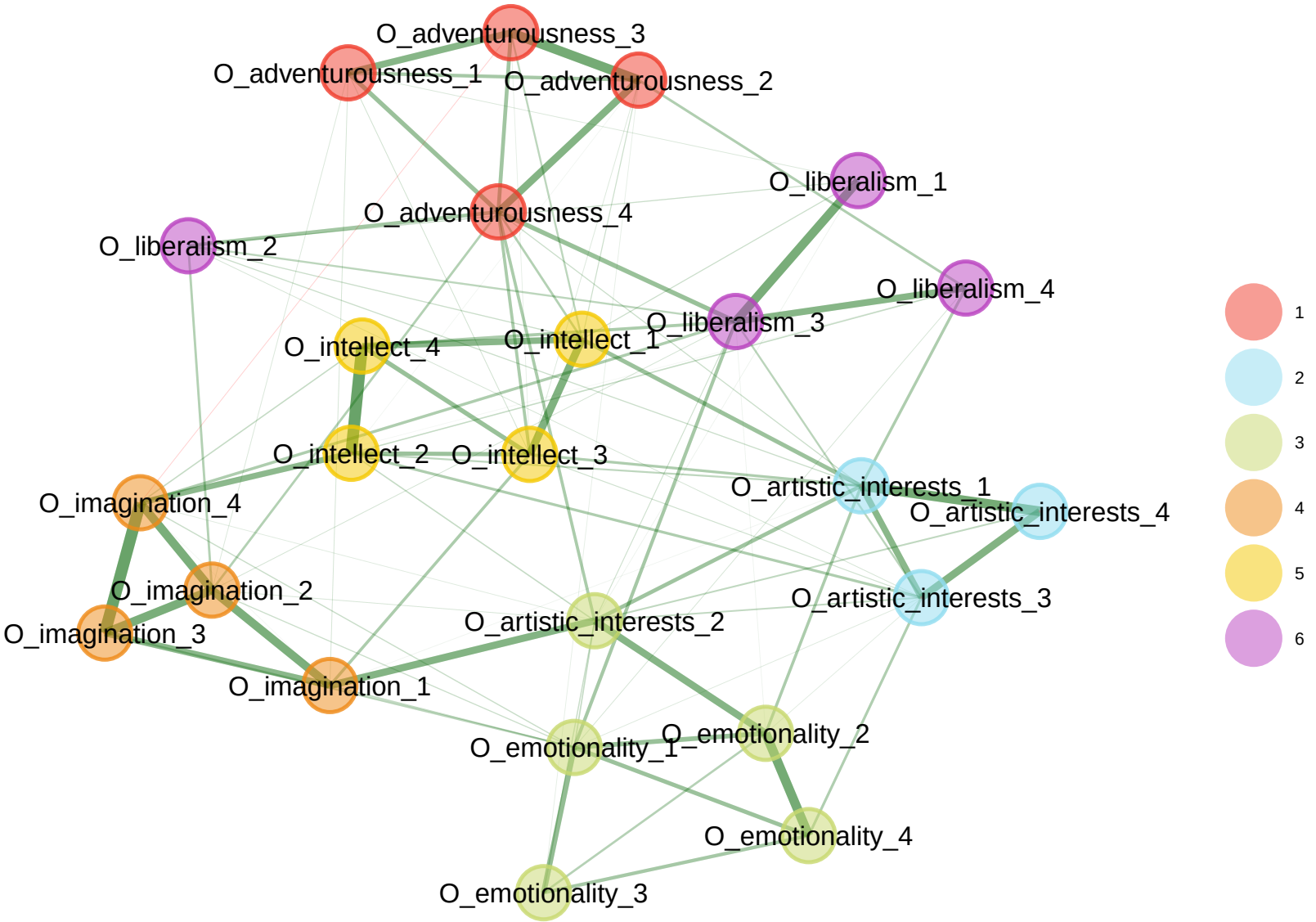


Exploratory Graph Analysis

It's actually easier than 1-2-3

```
# Apply EGA with Walktrap
openness_ega_walktrap <- EGA(openness_voi)

# Print summary
summary(openness_ega_walktrap)
```

Exploratory Graph Analysis



Exploratory Graph Analysis

```
Model: GLASSO (EBIC with gamma = 0.5)
Correlations: auto
Lambda: 0.135742007209069 (n = 100, ratio = 0.1)

Number of nodes: 24
Number of edges: 93
Edge density: 0.337

Non-zero edge weights:
     M    SD    Min   Max
 0.098 0.090 -0.026 0.356
```

Exploratory Graph Analysis

```
Algorithm:  Walktrap

Number of communities:  6

   O_adventurousness_1    O_adventurousness_2    O_adventurousness_3    O_adventurousness_4
                     1                      1                      1                      1
O_artistic_interests_1 O_artistic_interests_2 O_artistic_interests_3 O_artistic_interests_4
                     2                      3                      2                      2
      O_emotionality_1       O_emotionality_2       O_emotionality_3       O_emotionality_4
                     3                      3                      3                      3
       O_imagination_1        O_imagination_2        O_imagination_3        O_imagination_4
                     4                      4                      4                      4
         O_intellect_1          O_intellect_2          O_intellect_3          O_intellect_4
                     5                      5                      5                      5
        O_liberalism_1         O_liberalism_2         O_liberalism_3         O_liberalism_4
                     6                      6                      6                      6
```

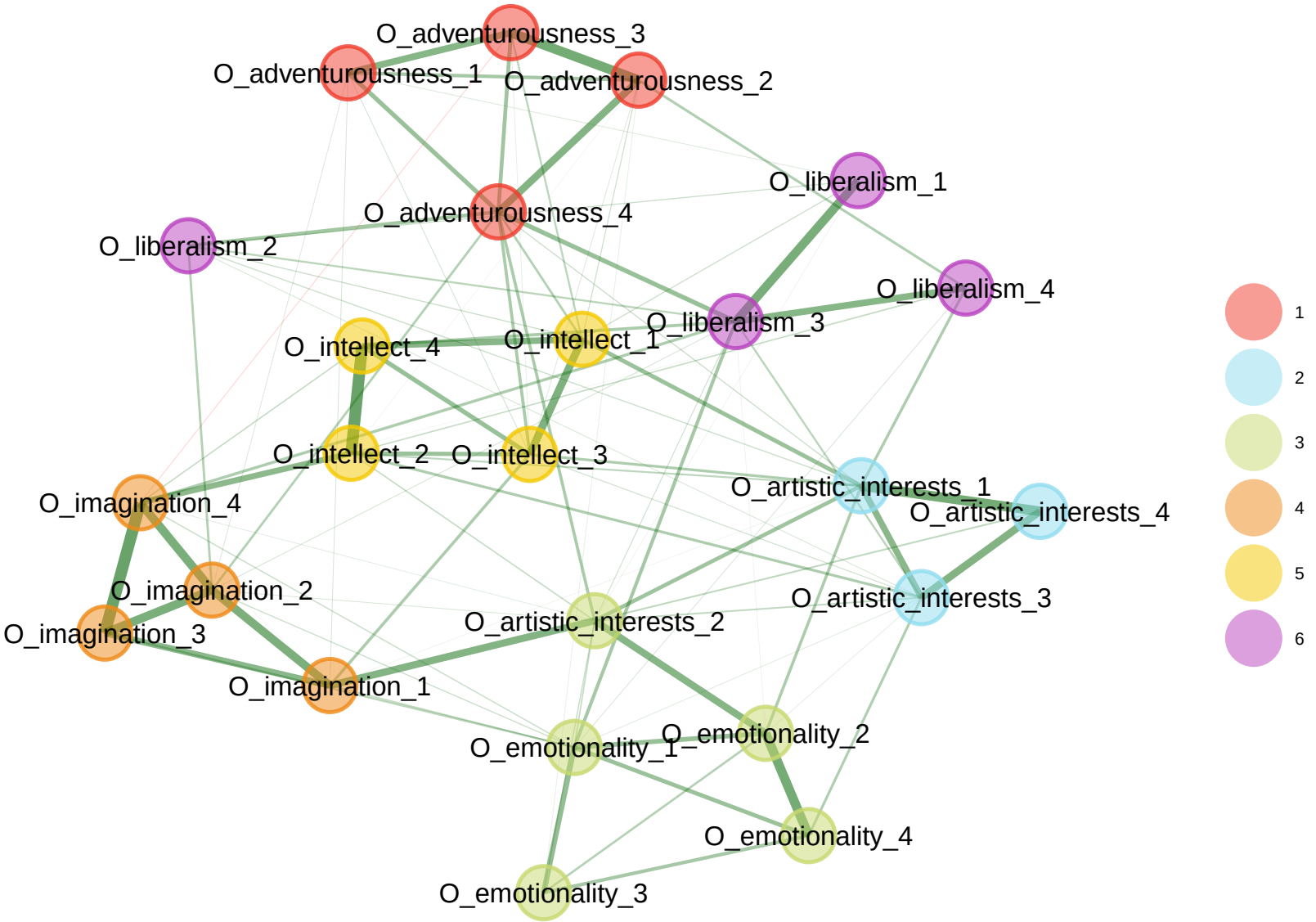
Exploratory Graph Analysis

```
Unidimensional Method: Louvain
Unidimensional: No

----

TEFI: -21.112
```

Exploratory Graph Analysis

```r
# Apply EGA with Louvain
openness_ega_louvain <- EGA(
  openness_voi, algorithm = "louvain"
)

# Print summary
summary(openness_ega_louvain)
```

Exploratory Graph Analysis



Exploratory Graph Analysis

```
Model: GLASSO (EBIC with gamma = 0.5)
Correlations: auto
Lambda: 0.0512352017222578 (n = 100, ratio = 0.1)

Number of nodes: 24
Number of edges: 154
Edge density: 0.558

Non-zero edge weights:
     M    SD    Min   Max
 0.029 0.085 -0.196 0.326
```

Exploratory Graph Analysis

```
Consensus Method: Most Common (1000 iterations)
Algorithm: Louvain
Order: Higher

Number of communities:  6

   O_adventurousness_1    O_adventurousness_2    O_adventurousness_3    O_adventurousness_4
                     1                      1                      1                      1
O_artistic_interests_1 O_artistic_interests_2 O_artistic_interests_3 O_artistic_interests_4
                     2                      3                      2                      2
      O_emotionality_1       O_emotionality_2       O_emotionality_3       O_emotionality_4
                     3                      3                      3                      3
       O_imagination_1        O_imagination_2        O_imagination_3        O_imagination_4
                     4                      4                      4                      4
         O_intellect_1          O_intellect_2          O_intellect_3          O_intellect_4
                     5                      5                      5                      5
        O_liberalism_1         O_liberalism_2         O_liberalism_3         O_liberalism_4
                     6                      6                      6                      6
```

Exploratory Graph Analysis

```
Unidimensional Method: Louvain
Unidimensional: No

----

TEFI: -21.112
```

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

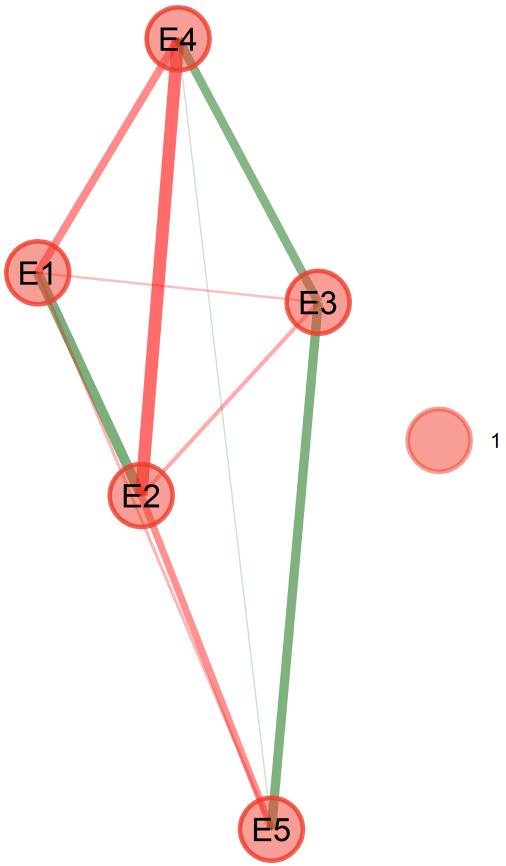
In networks, determining whether there is unidimensionality is tough…

Fundamentally, this issue arises from a few problems

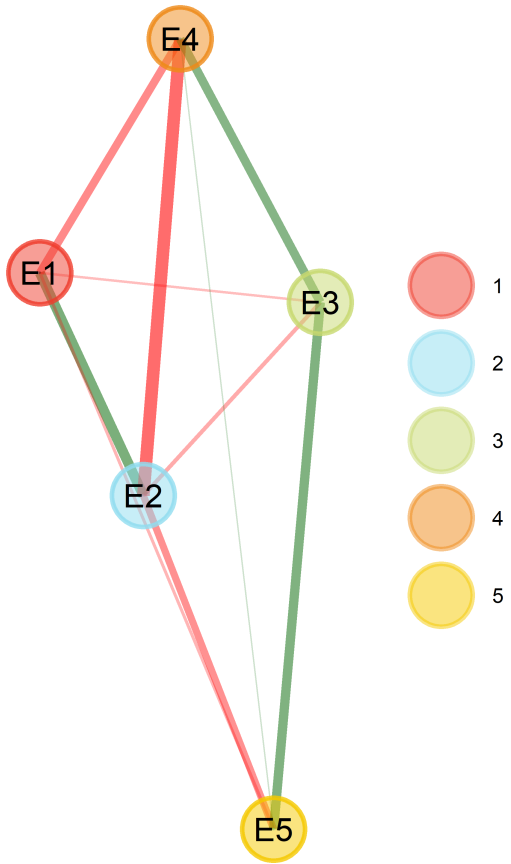
1. Modularity as a measure penalizes unidimensionality such that modularity equals zero (so almost any modular solution will be greater than one)
2. Partial correlations unevenly and unpredictably decrease relations between some variables more than others
3. Sparse networks are inherently modular due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

Modularity

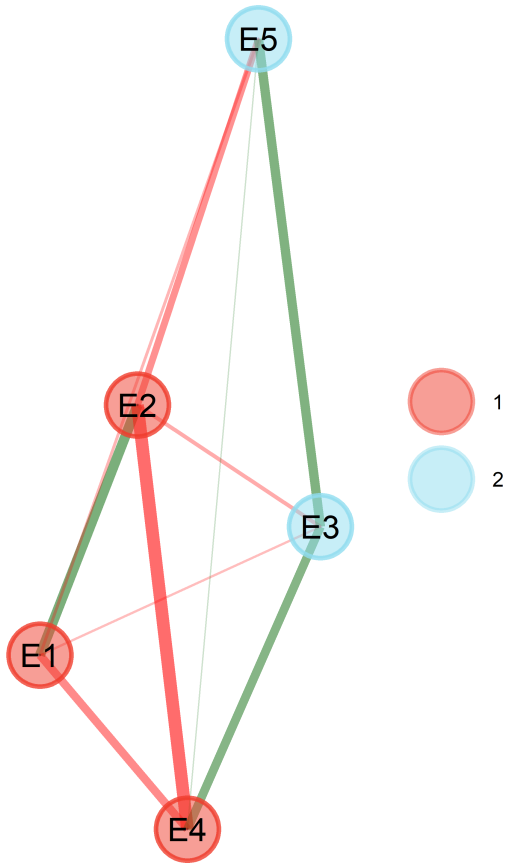
Single Community
Modularity = -0.000



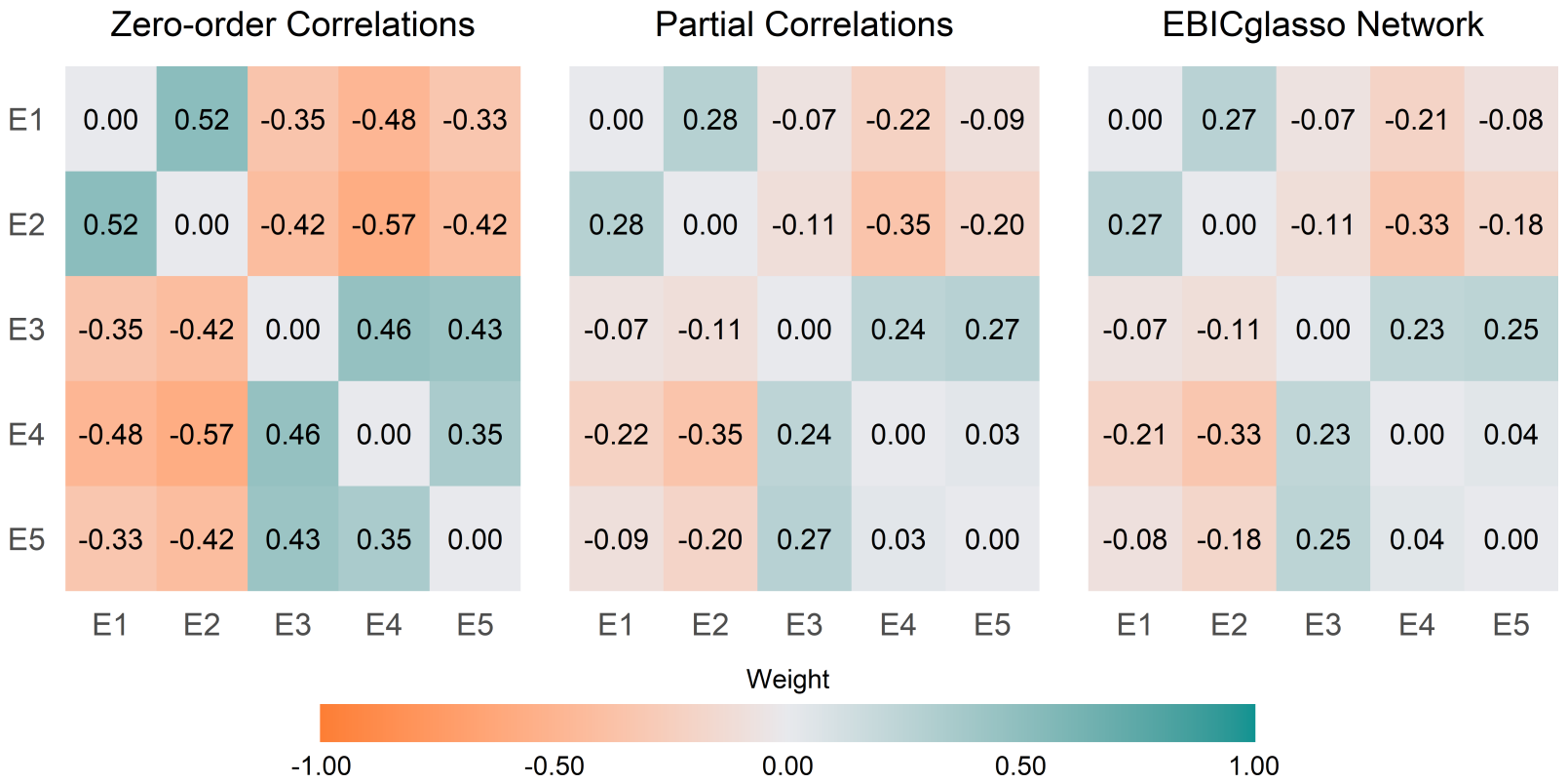
Singleton Communities
Modularity = -0.206



Louvain Communities
Modularity = 0.048

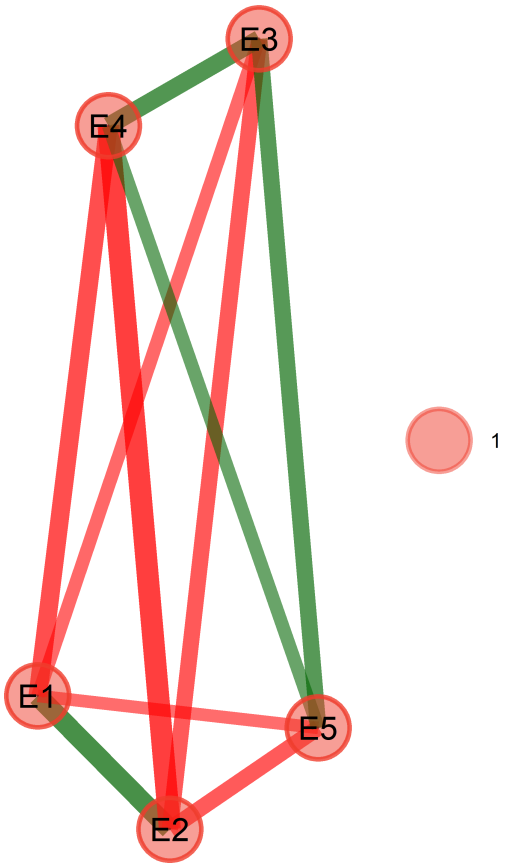


(Partial) Correlations

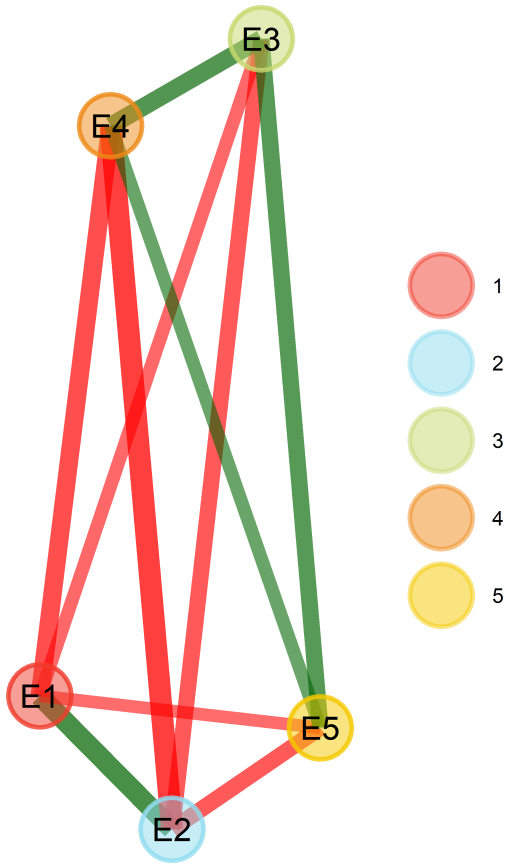


Complete Networks

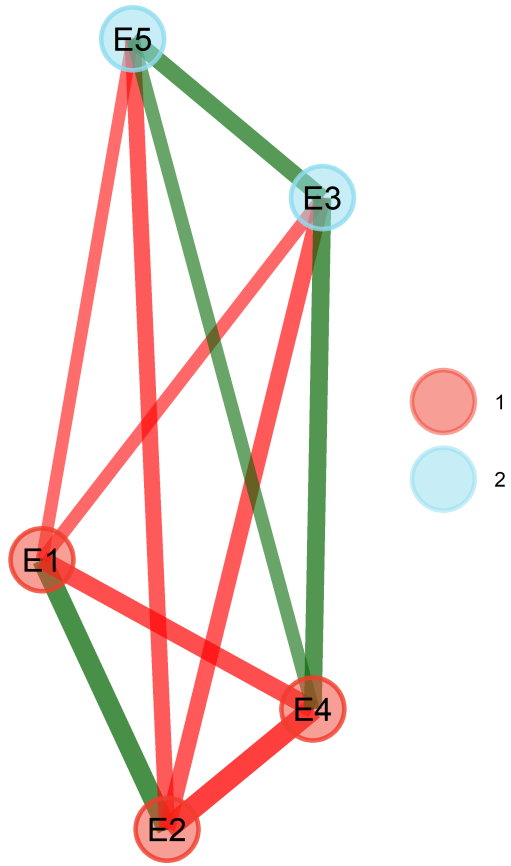
Single Community
Modularity = -0.000



Singleton Communities
Modularity = -0.201



Louvain Communities
Modularity = -0.073



Solution: apply community detection algorithms 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

Total Entropy Fit Index

Exploratory Graph Analysis | TEFI

There are two solutions from Walktrap and Louvain as well as many, many other solutions that can be achieved with different algorithms

What solution should be used?

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

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

Works directly on the values of the data

Von Neumann Entropy

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

where

$$\rho = \frac{\mathbf{R}}{\text{diag}(\mathbf{R})}$$

where **R** is the correlation matrix

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 more accurate than more traditional measures commonly used in dimension reduction

Lower values = better solution

Stepping back to our previous EGA results…

Walktrap = -21.112

Louvain = -21.112

Based on our results, the algorithms fit the same (because they produced the same solution)

EGA + TEFI

Exploratory Graph Analysis | EGA + TEFI

Some algorithms have tunable parameters

Walktrap = number of steps in its random walks (defaults to 4)

Louvain = resolution parameter (γ, defaults to 1)

The resolution parameter adjusts the extent to which smaller ($\gamma > 1$) or larger ($\gamma < 1$) communities

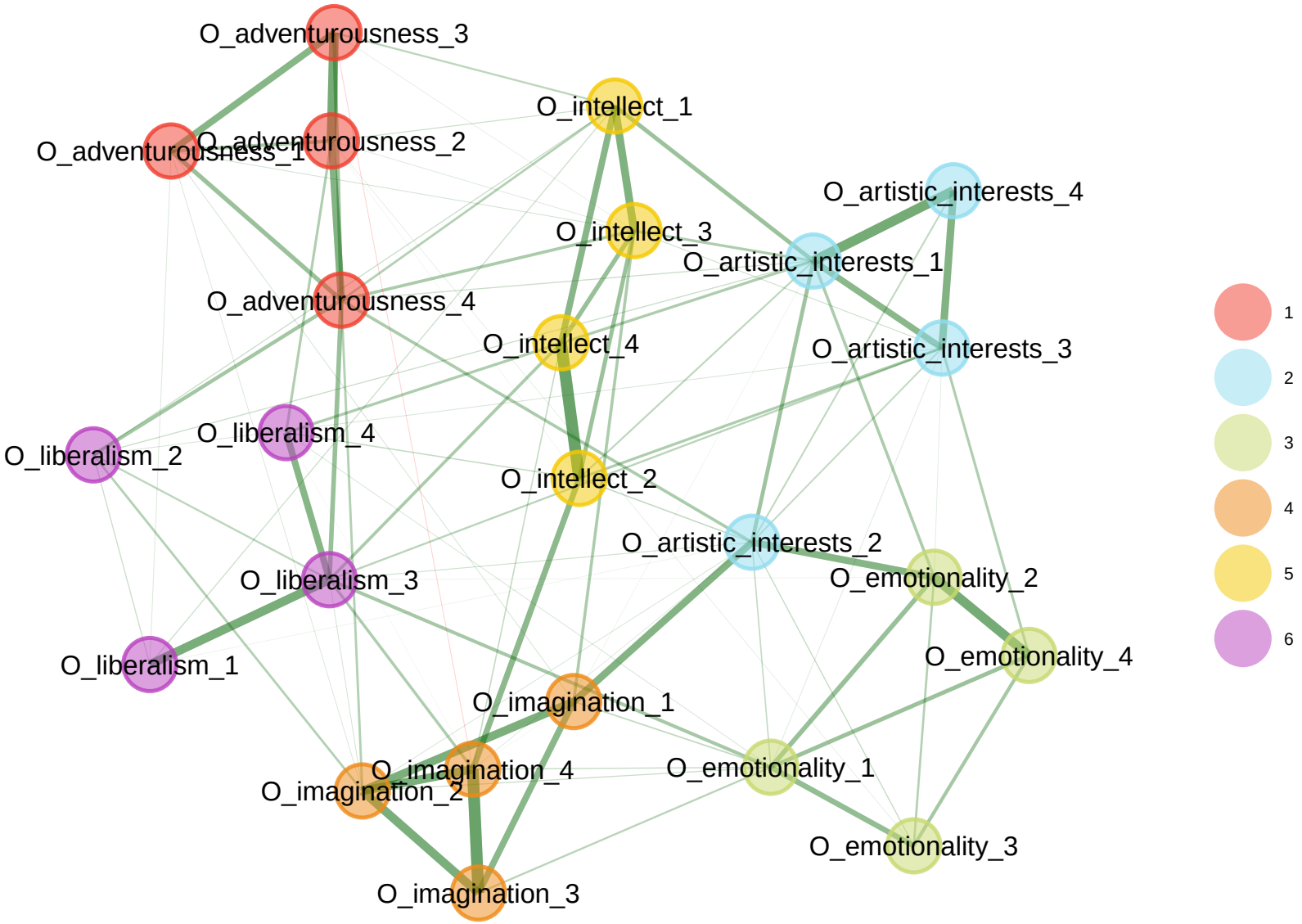
With TEFI, a grid search over parameters can be applied to find the absolute best possible fit to the data

Exploratory Graph Analysis | EGA + TEFI

Shockingly (audible gasps)… there's a function for that

```
# Apply EGA with Walktrap
openness_walktrap_fit <- EGA.fit(openness_voi)

# Print summary
summary(openness_walktrap_fit)
```

Exploratory Graph Analysis | EGA + TEFI



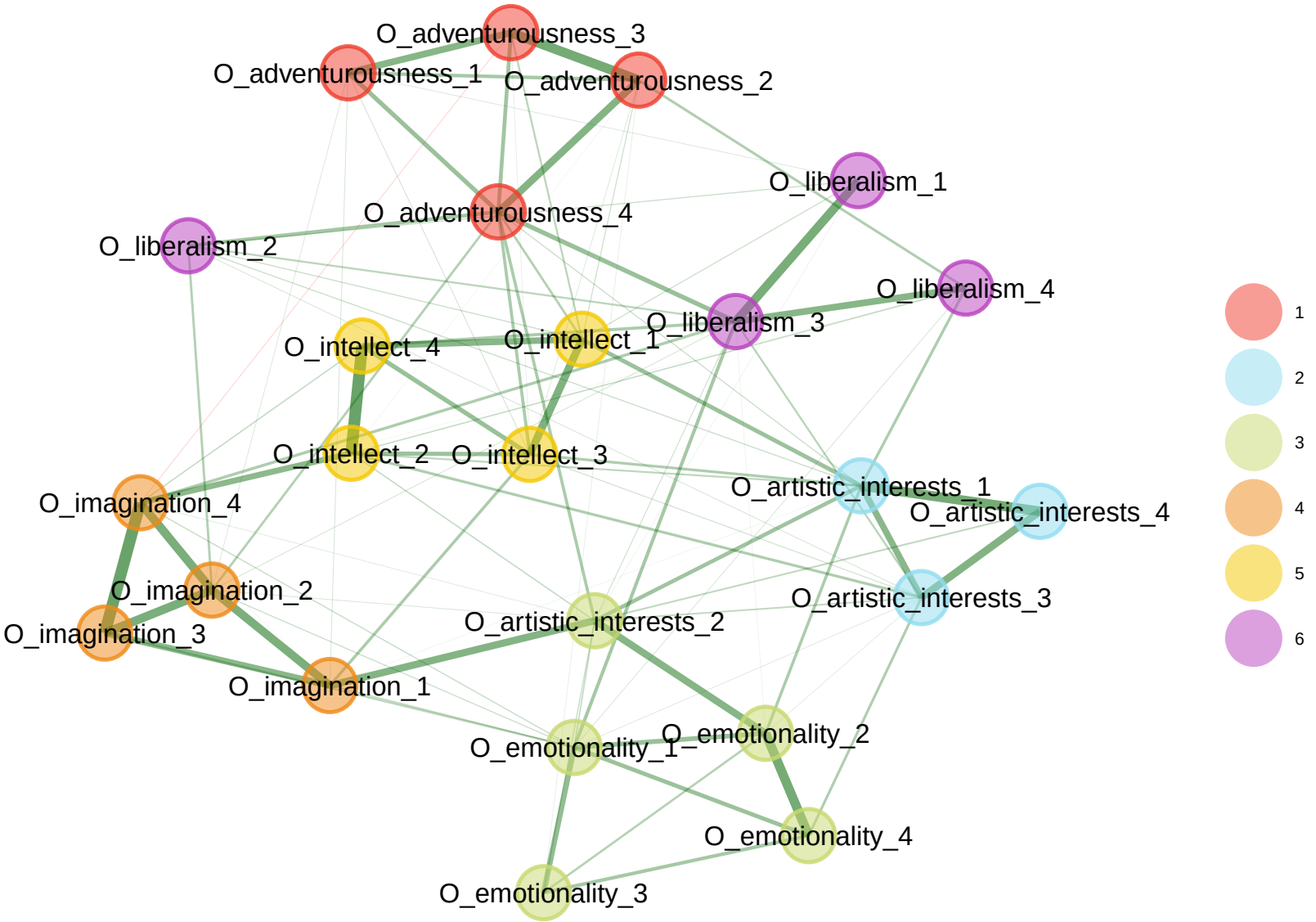
Exploratory Graph Analysis | EGA + TEFI

```
Algorithm:  Walktrap (Steps = 3)

Number of communities:  6

   O_adventurousness_1    O_adventurousness_2    O_adventurousness_3    O_adventurousness_4
                     1                      1                      1                      1
O_artistic_interests_1 O_artistic_interests_2 O_artistic_interests_3 O_artistic_interests_4
                     2                      2                      2                      2
      O_emotionality_1       O_emotionality_2       O_emotionality_3       O_emotionality_4
                     3                      3                      3                      3
       O_imagination_1        O_imagination_2        O_imagination_3        O_imagination_4
                     4                      4                      4                      4
         O_intellect_1          O_intellect_2          O_intellect_3          O_intellect_4
                     5                      5                      5                      5
        O_liberalism_1         O_liberalism_2         O_liberalism_3         O_liberalism_4
                     6                      6                      6                      6


----

TEFI: -21.336
```

Exploratory Graph Analysis | EGA + TEFI

```r
# Apply EGA with Louvain
openness_louvain_fit <- EGA.fit(
  openness_voi, algorithm = "louvain"
)

# Print summary
summary(openness_louvain_fit)
```

Exploratory Graph Analysis | EGA + TEFI



Exploratory Graph Analysis | EGA + TEFI

```
Algorithm:  Louvain (Resolution = 0)

Number of communities:  6

   O_adventurousness_1    O_adventurousness_2    O_adventurousness_3    O_adventurousness_4
                     1                      1                      1                      1
O_artistic_interests_1 O_artistic_interests_2 O_artistic_interests_3 O_artistic_interests_4
                     2                      3                      2                      2
      O_emotionality_1       O_emotionality_2       O_emotionality_3       O_emotionality_4
                     3                      3                      3                      3
       O_imagination_1        O_imagination_2        O_imagination_3        O_imagination_4
                     4                      4                      4                      4
         O_intellect_1          O_intellect_2          O_intellect_3          O_intellect_4
                     5                      5                      5                      5
        O_liberalism_1         O_liberalism_2         O_liberalism_3         O_liberalism_4
                     6                      6                      6                      6


----

TEFI: -21.112
```

Exploratory Graph Analysis | EGA + TEFI

Stepping back to our previous EGA results…

Walktrap = -21.112

Louvain = -21.112

Now, with our TEFI results…

Walktrap (TEFI) = -21.336

Louvain (TEFI) = -21.112

Walktrap with TEFI selection would be the preferred solution (and actually matches the theoretical solution!)