Estimating and Interpreting Psychological Networks in R

Jens Lange

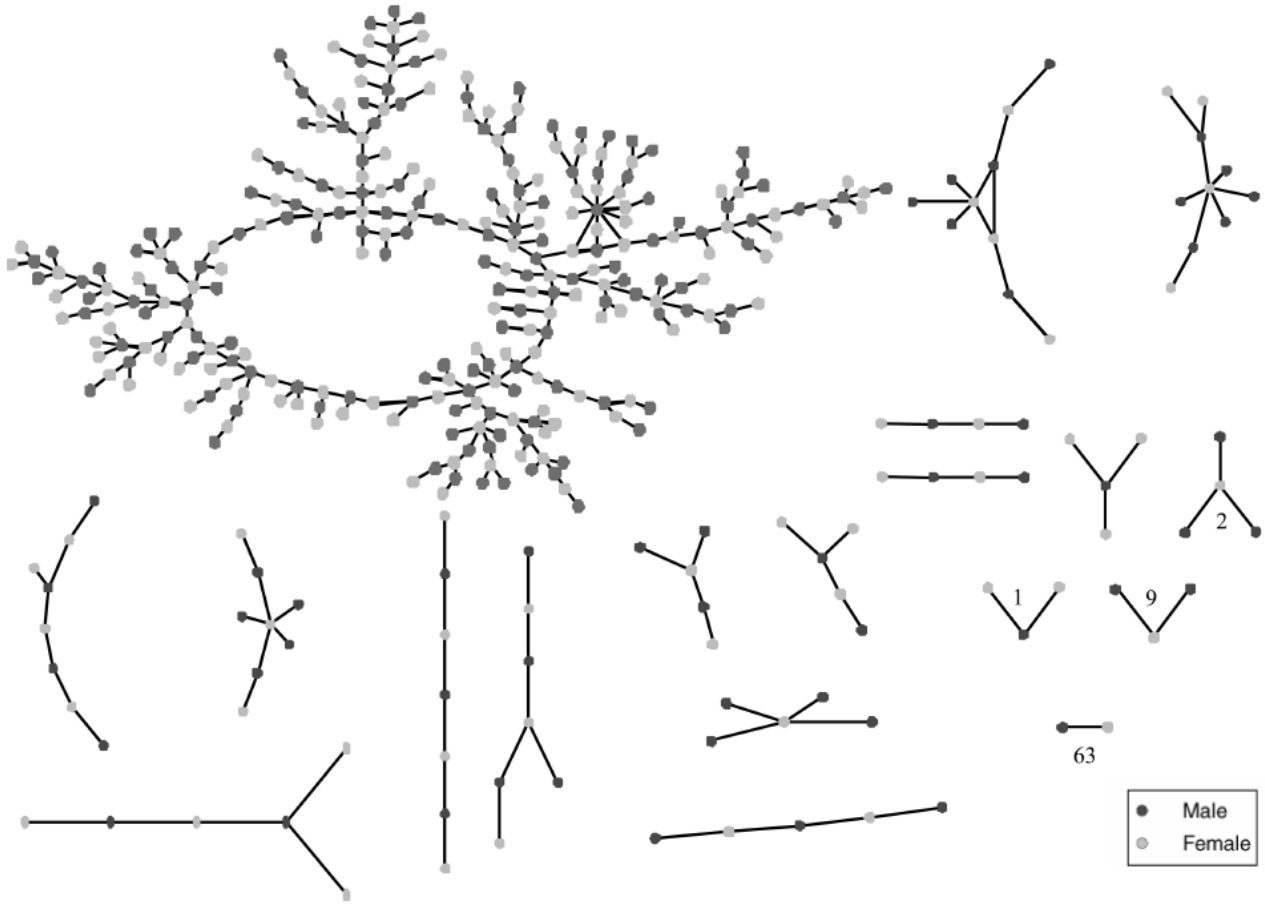
Nov 29, 2019

Introduction to Complex Networks

Complexity Science

- *complex systems* entail parts that interact with each other to produce system behavior

Complex Networks - Examples

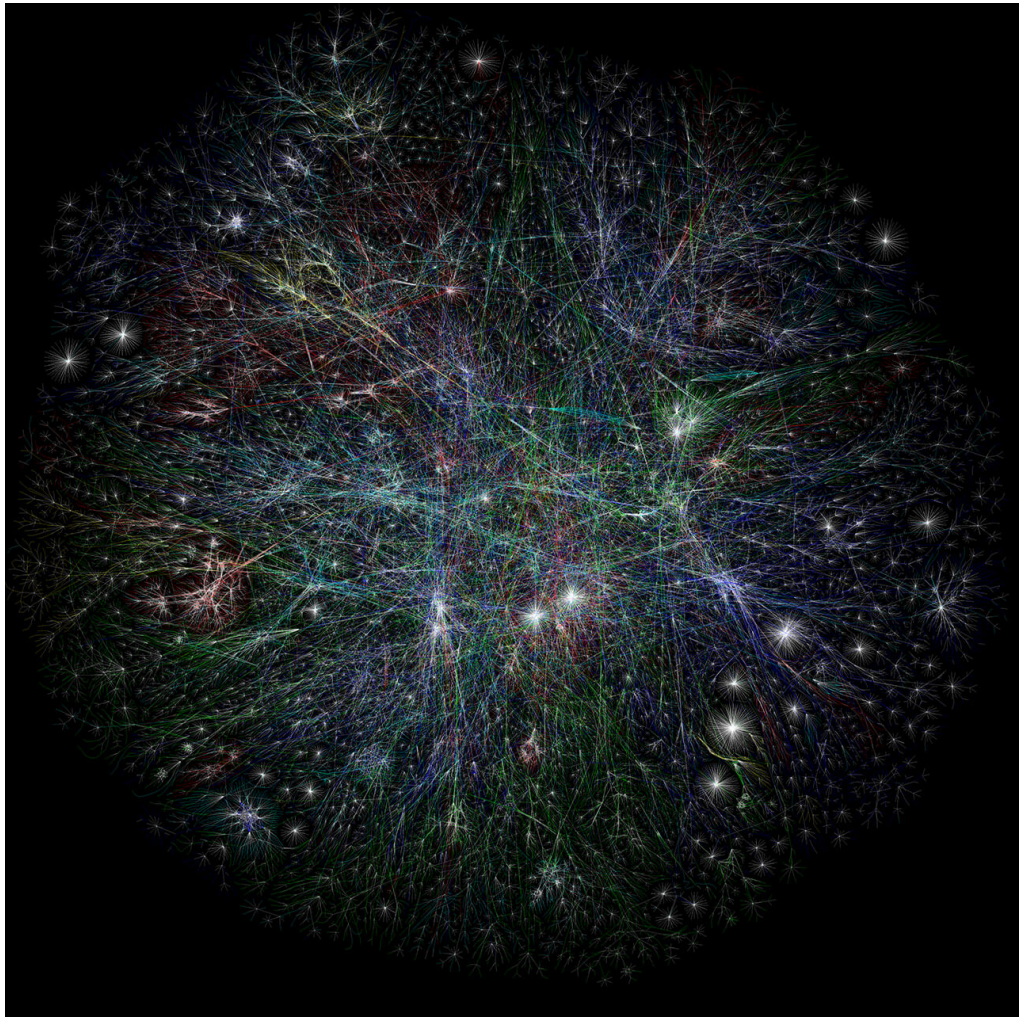


Bearman et al. (2004, AJS)

Complex Networks - Examples

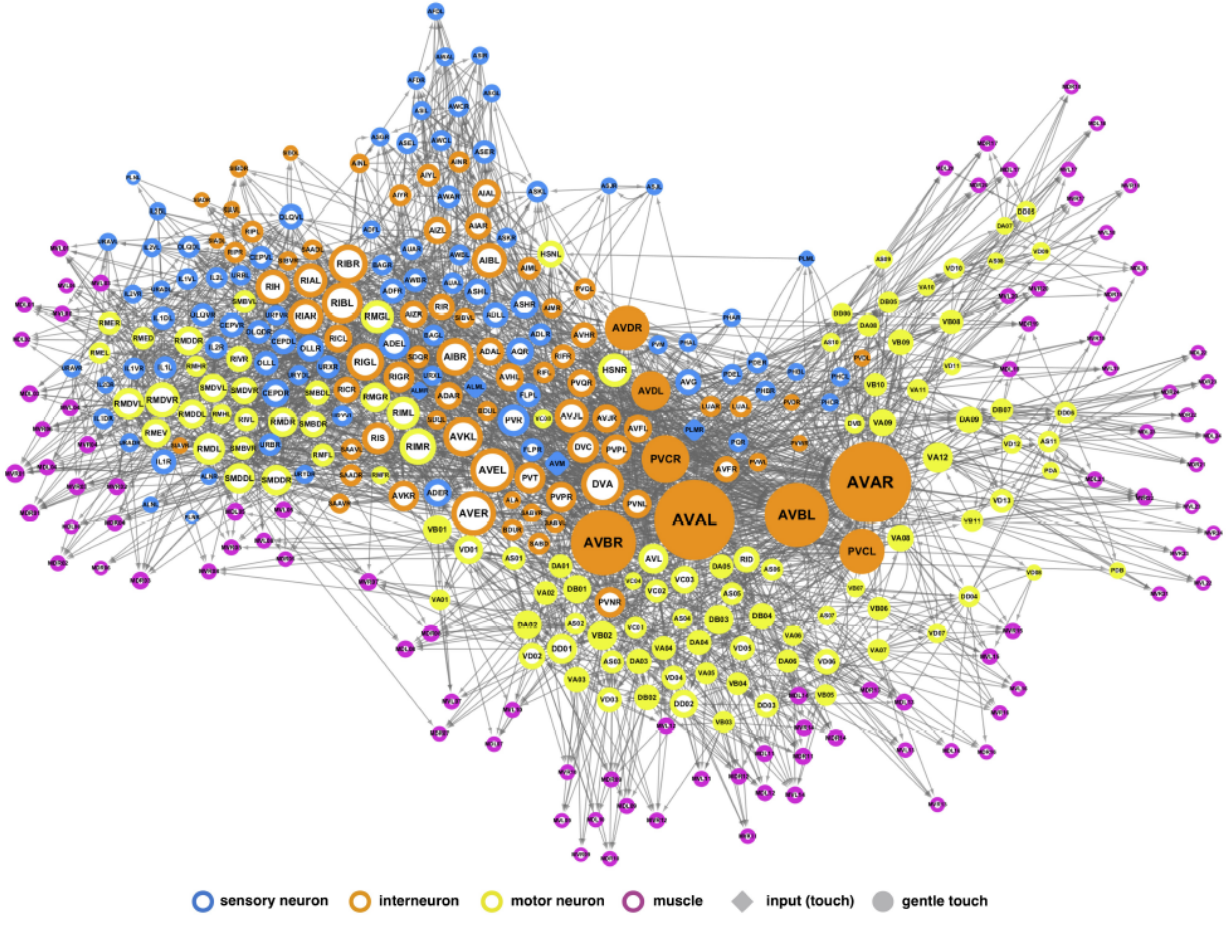
Xiaoqian et al. (2017, CJA)

Complex Networks - Examples



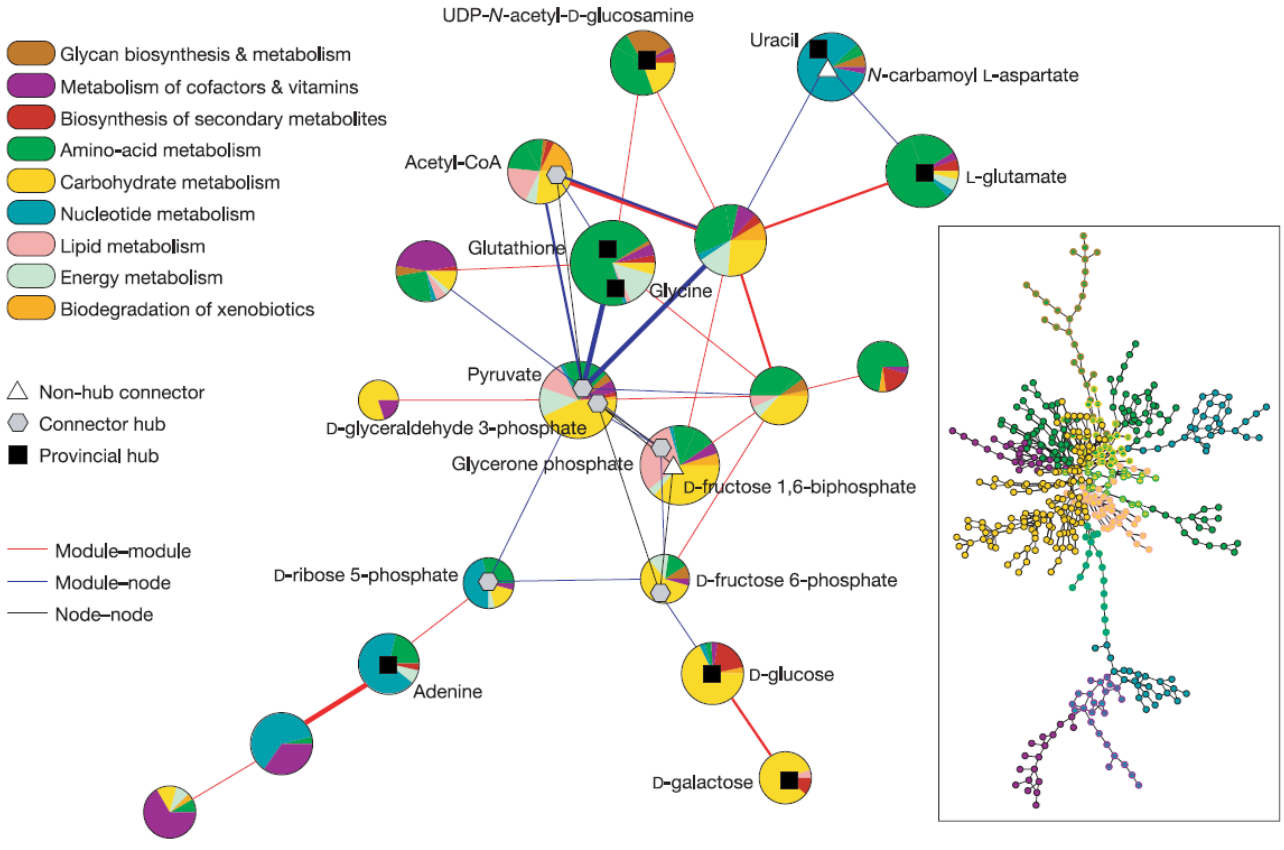
Albert et al. (1999, Nature)

Complex Networks - Examples



Yan et al. (2017, Nature)

Complex Networks - Examples



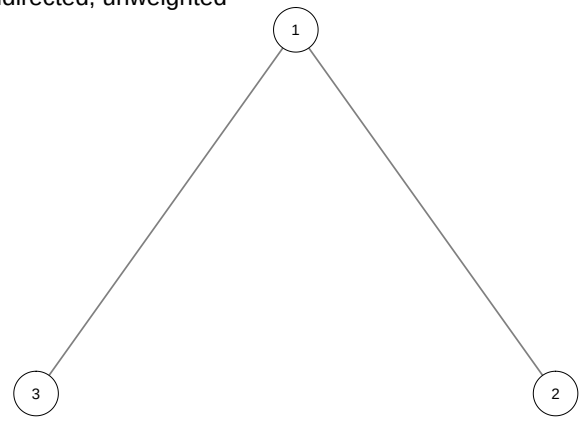
Guimerà & Amaral (2005, Nature)

Complex Networks - Definitions

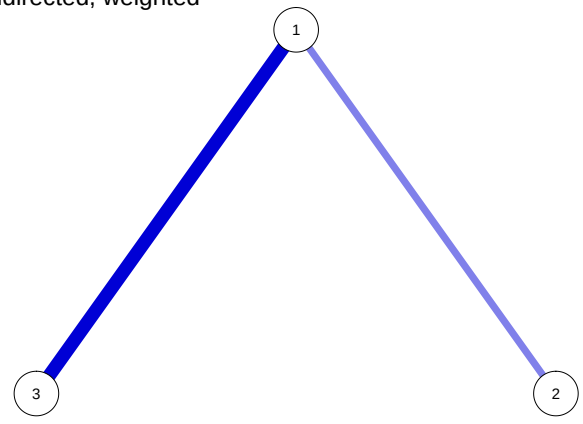
- network: *nodes* that are connected via *edges*
- nodes can be. . .
  - people
  - neurons
  - psychological variables
  - . . .
- edges can be. . .
  - friendhips
  - synapses
  - causal connections
  - . . .

Complex Networks - Definitions

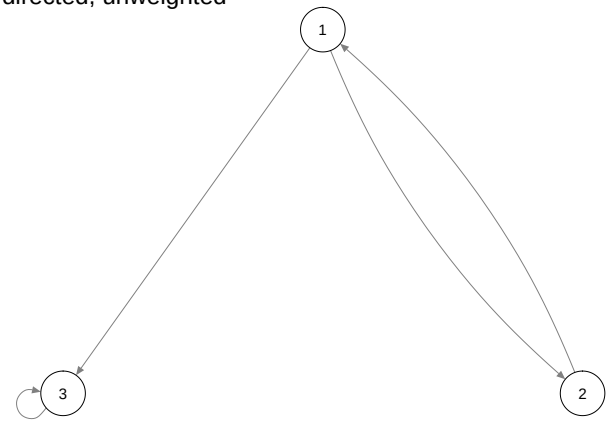
undirected, unweighted



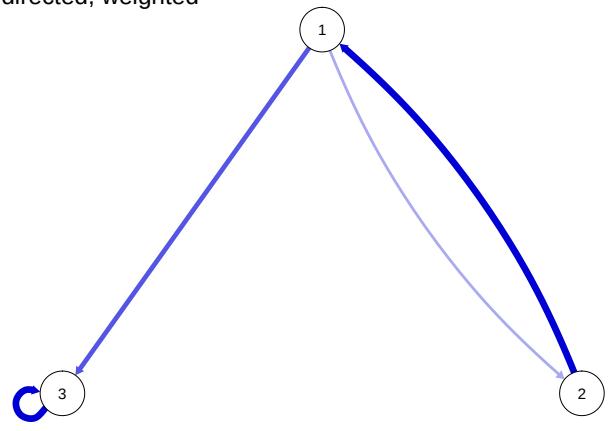
undirected, weighted



directed, unweighted

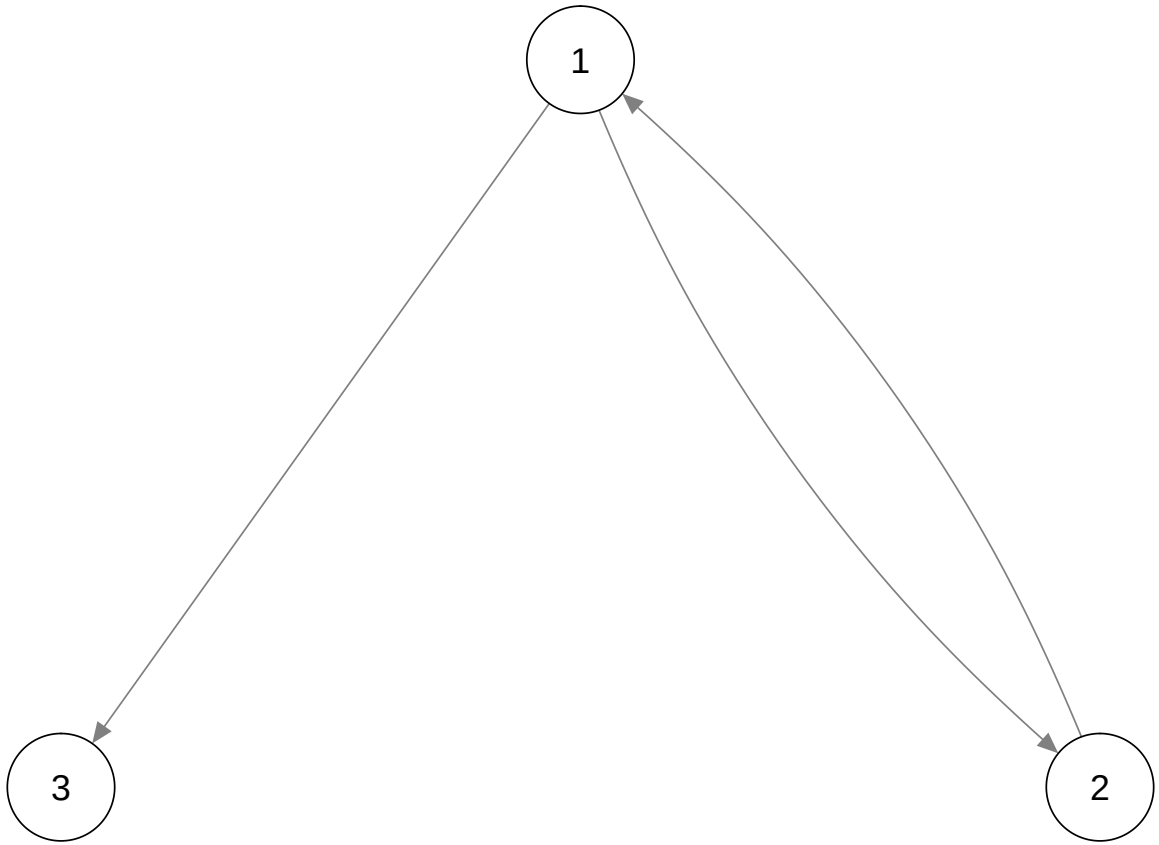


directed, weighted



Complex Networks - Mathematical Notation

- a Graph G entails sets of nodes N and edges E
  - $G = \{N, E\}$
  - $N = \{1, 2, 3\}$
  - $E = \{(1, 2), (1, 3), (2, 1)\}$

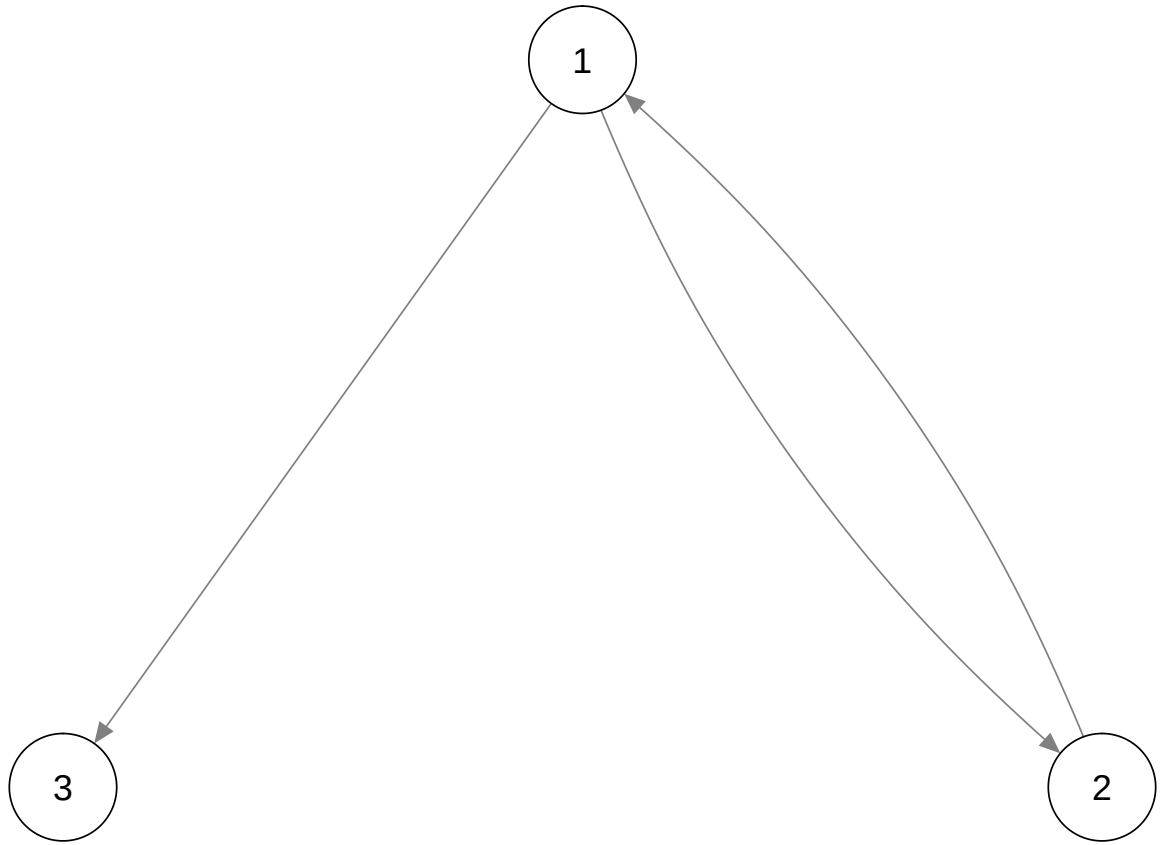


Complex Networks - Adjacency/Weights Matrix

- *adjacency matrix*: $N \times N$ matrix, with elements 0 or 1
  - 1 in row i and column j: edge from node i to j
  - undirected network has symmetric adjacency matrix
- *weights matrix* has the same form but with weights

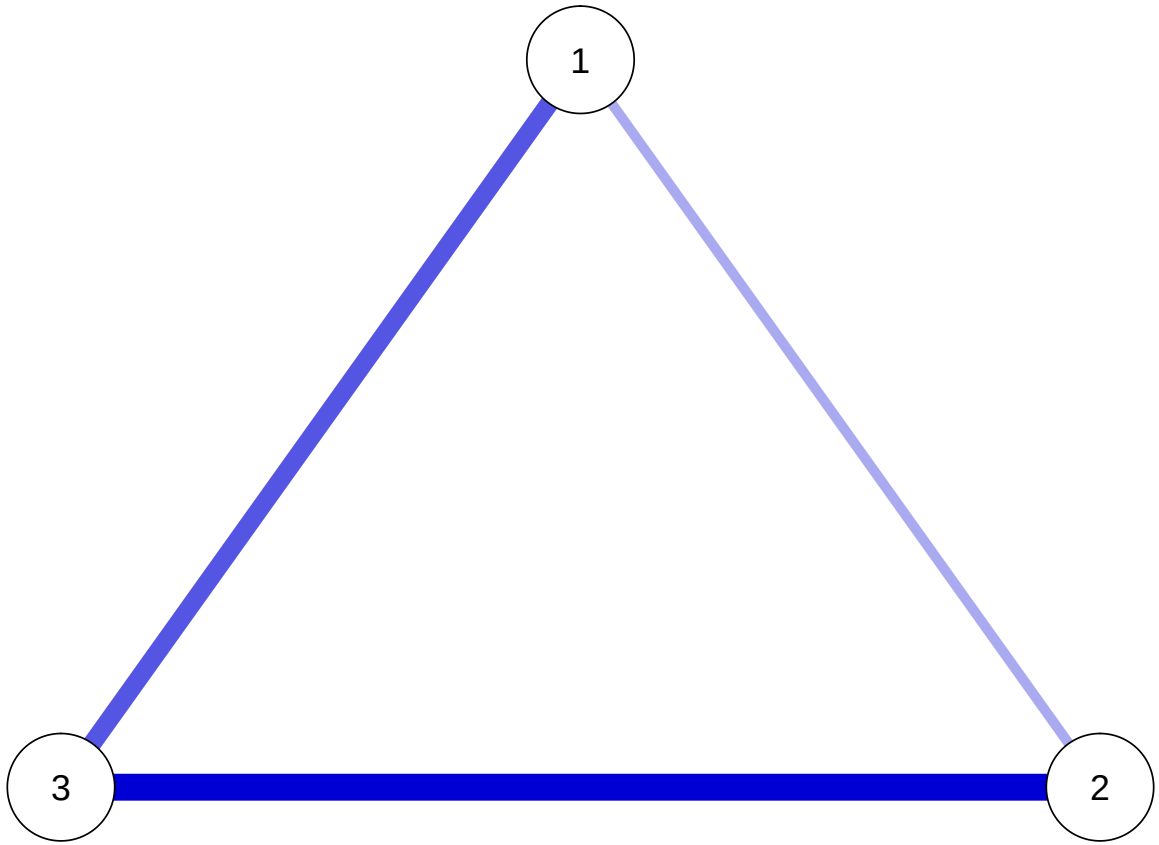
Complex Networks - Adjacency/Weights Matrix

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



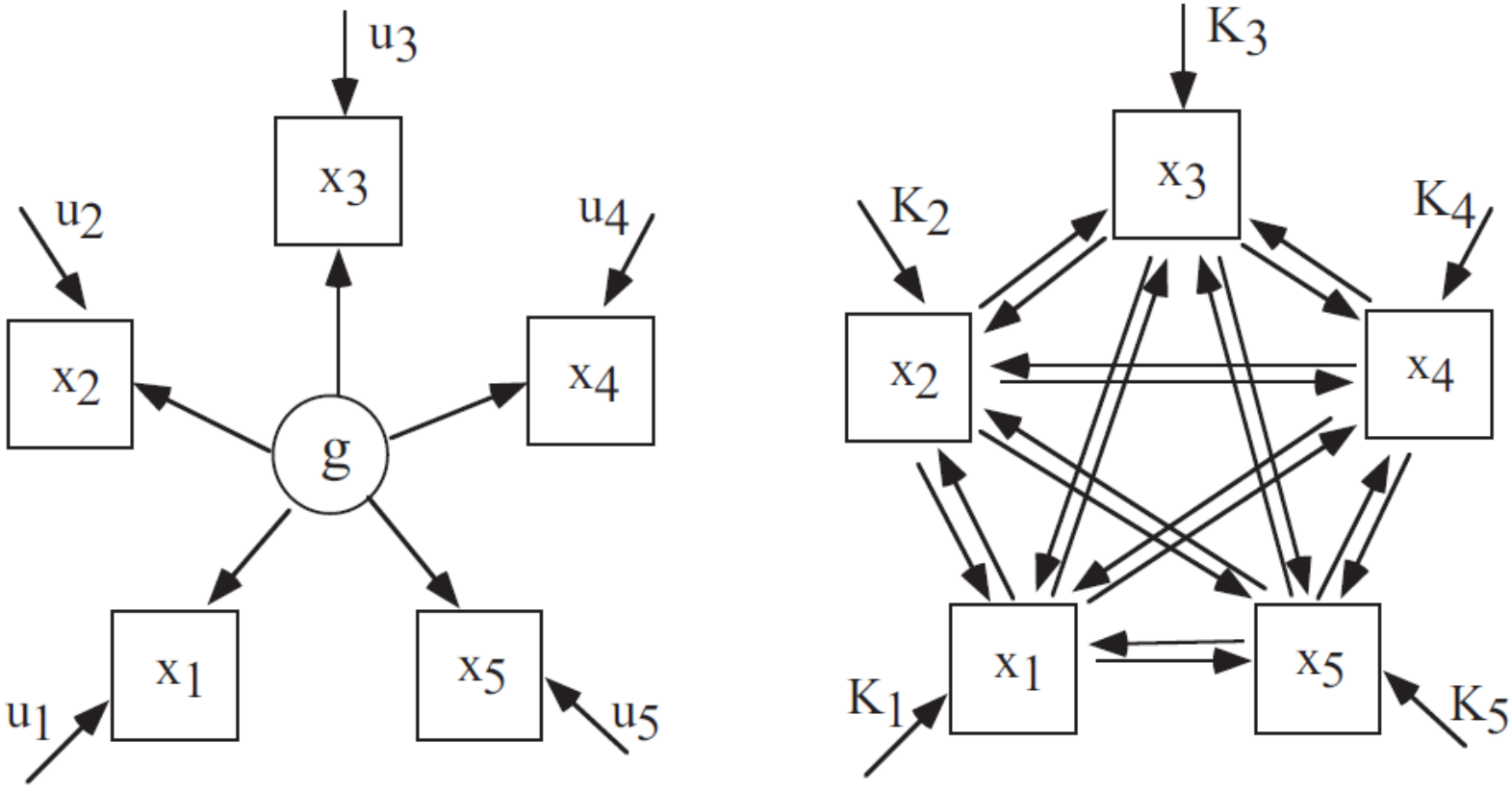
Complex Networks - Adjacency/Weights Matrix

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix}$$



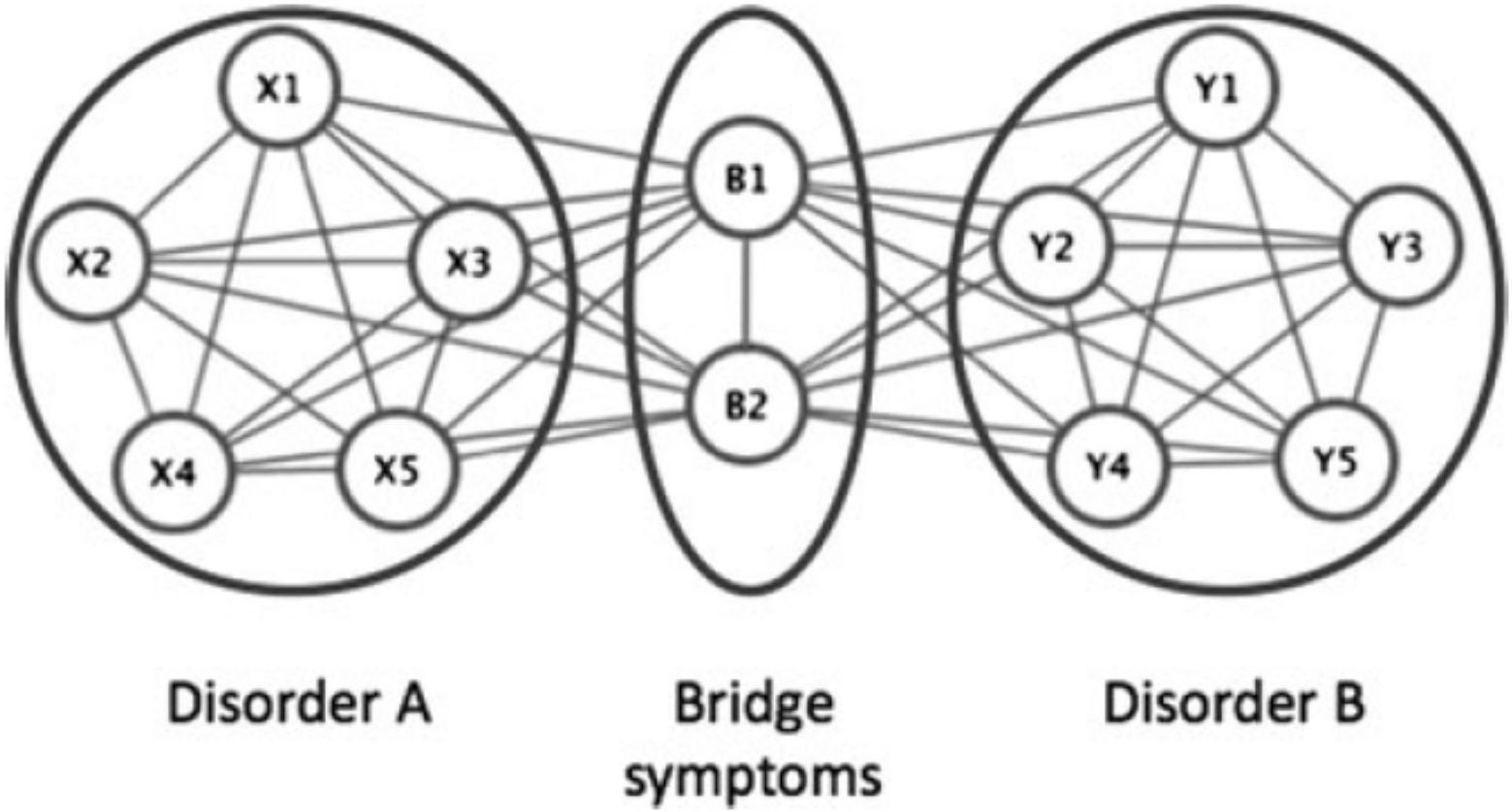
Psychological Networks

Psychological Networks - History



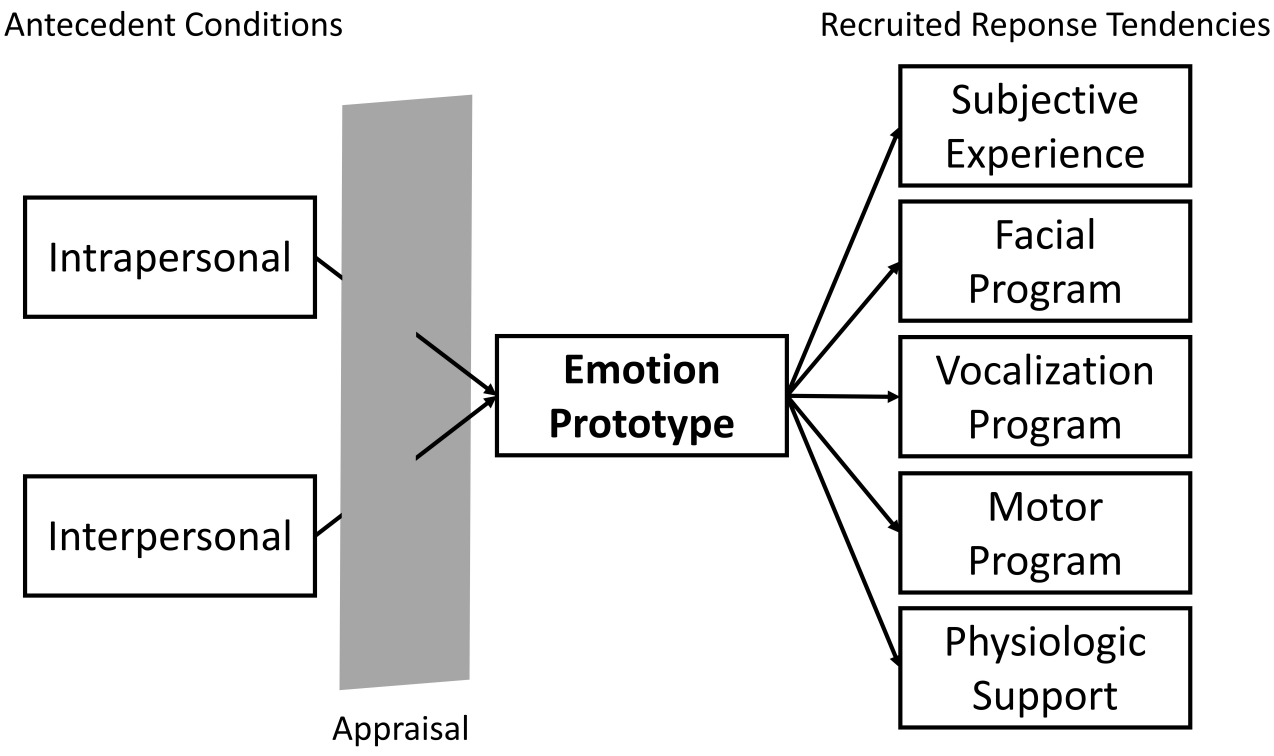
Van der Maas et al. (2006, Psych Review)

Psychological Networks - History



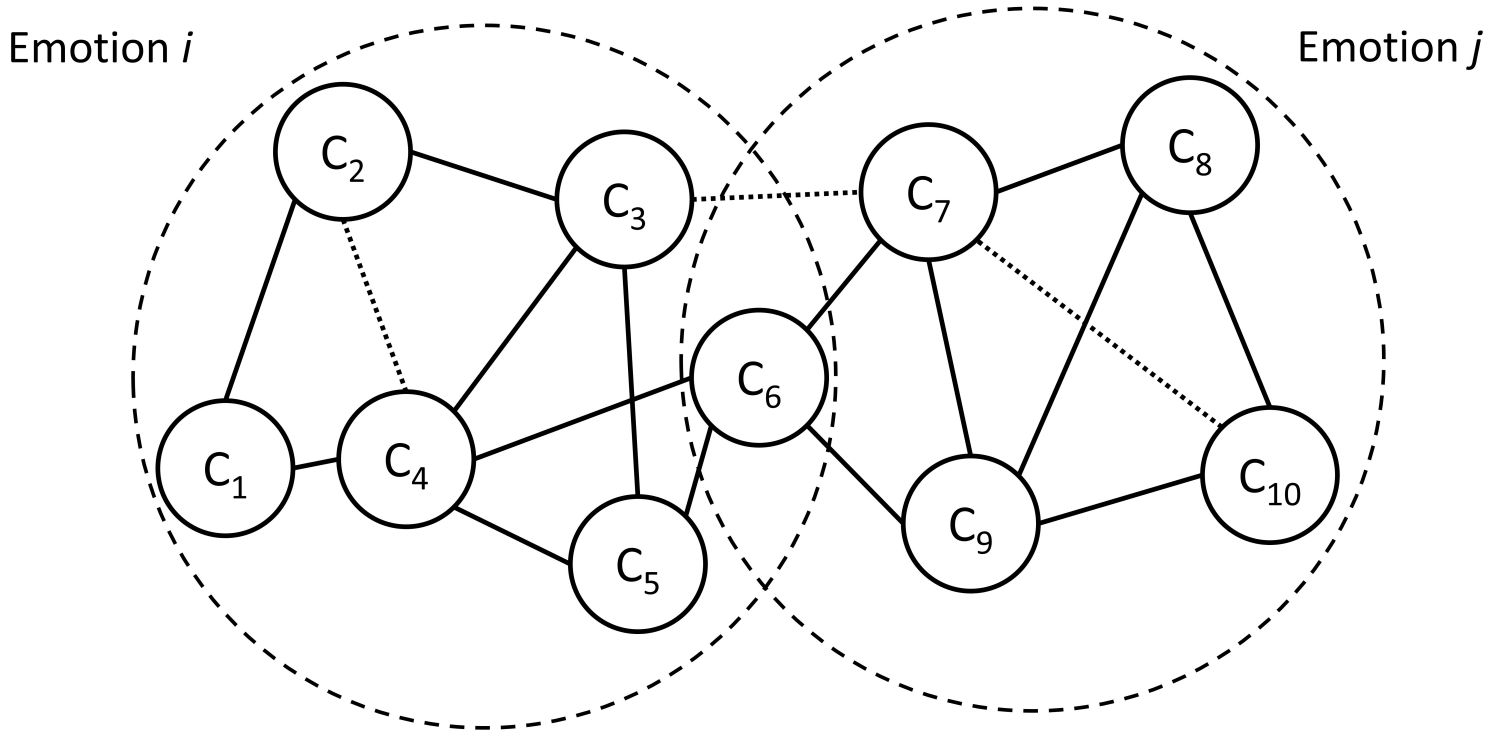
Cramer et al. (2010, BBS)

Psychological Networks - Emotions



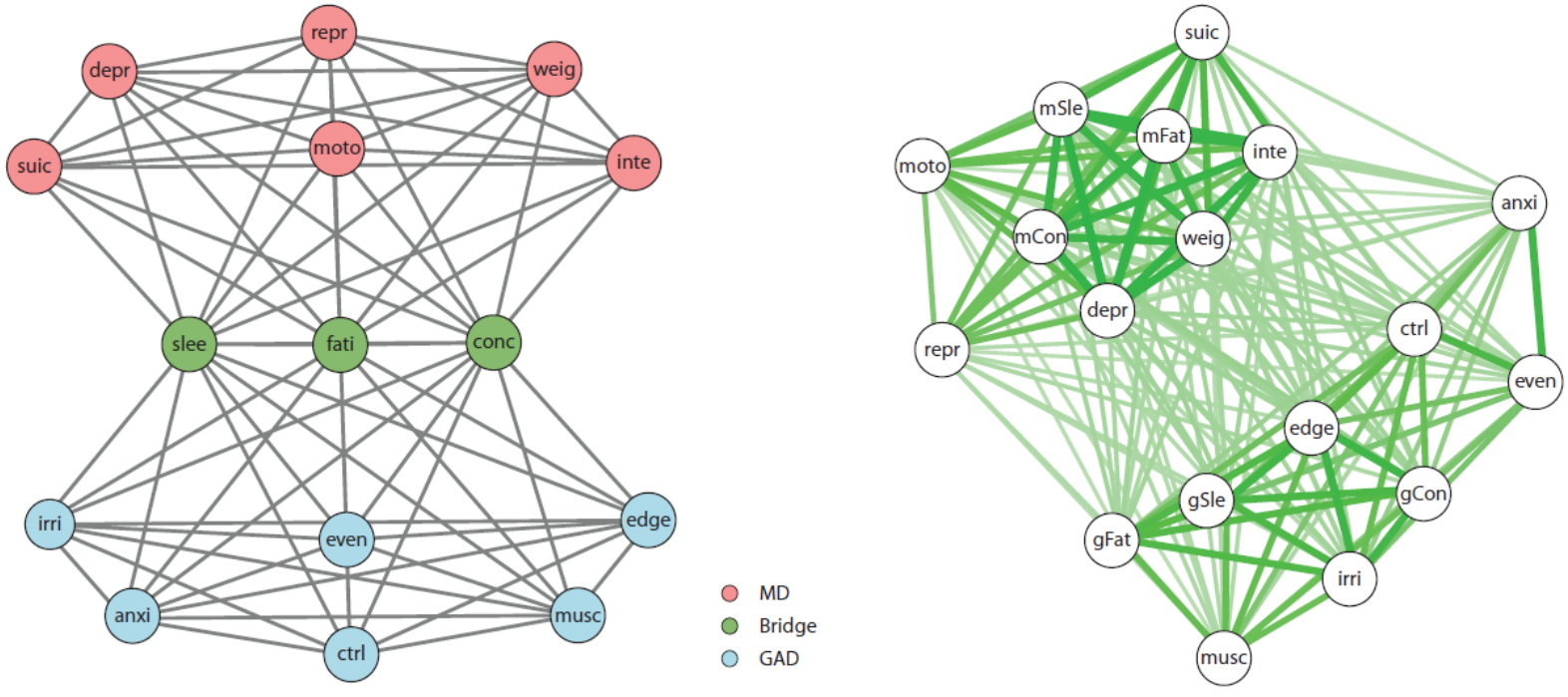
Levenson (1994, book chapter)

Psychological Networks - Emotions



Lange et al. (in press, PPS); Lange & Zickfeld (in prep.)

Psychological Networks - Network analysis



Borsboom & Cramer (2013, ARCP); Epskamp & Fried (2018, Psych Methods); Epskamp et al. (2018, MBR); Van Borkulo et al. (2015, SR)

Summary for Introduction

Summary

- networks are versatile approach to complex systems
- networks entail sets of nodes and edges
- in psychological networks...
  - nodes: indicators of a psychological construct
  - edges: causal connections between indicators
- network analysis as methodological tool to estimate causal networks

Estimating Cross-sectional Psychological Networks

Data Structure

- Person X Node-Rating Matrix

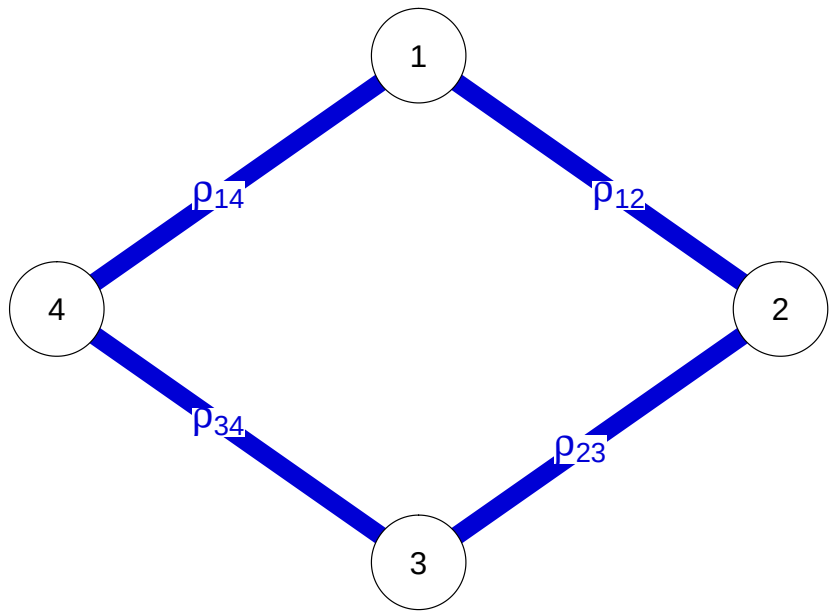
```
#>   Subject Node1 Node2 Node3 Node4
#> 1       1     3     2     5     1
#> 2       2     2     1     6     2
#> 3       3     1     1     4     2
#> 4     ...   ...   ...   ...   ...
#> 5       n     3     3     5     4
```

pairwise Markov Random Field Model (pMRF)

- unique relationships of two nodes

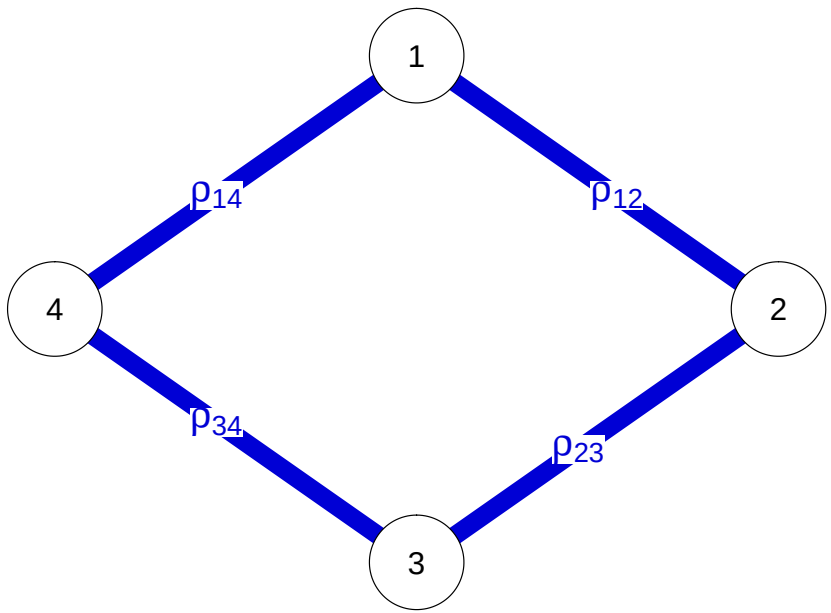
pairwise Markov Random Field Model (pMRF)

- unique relationships of two nodes
- estimation of *conditional dependencies*



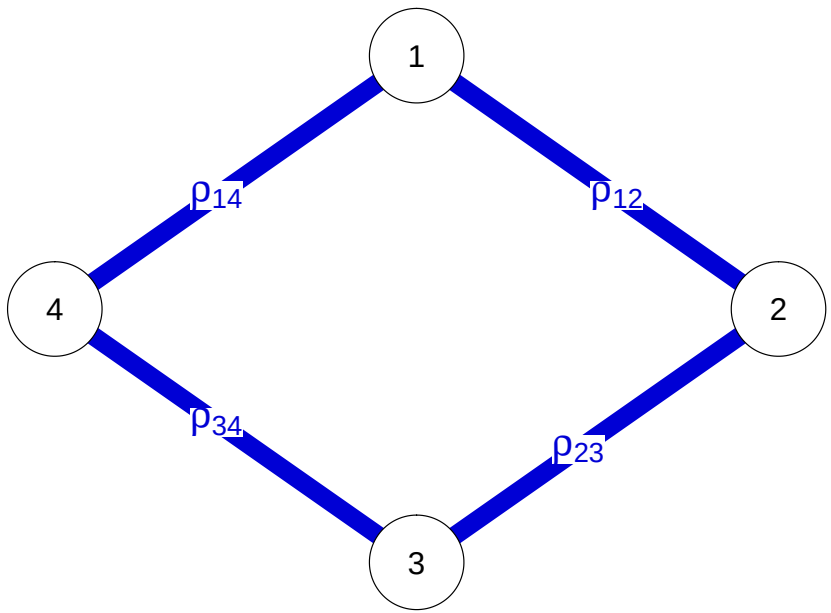
pairwise Markov Random Field Model (pMRF)

- unique relationships of two nodes
- estimation of *conditional dependencies*
  - edge: e.g. $Node1 \not\perp\!\!\!\perp Node2 \mid Node3, Node4$



pairwise Markov Random Field Model (pMRF)

- unique relationships of two nodes
- estimation of *conditional dependencies*
  - edge: e.g. $Node1 \not\perp\!\!\!\perp Node2 \mid Node3, Node4$
  - no edge: e.g. $Node1 \perp\!\!\!\perp Node3 \mid Node2, Node4$



Gaussian Graphical Model

- pMRF for continuous data

Gaussian Graphical Model

- pMRF for continuous data
- Option 1: conditional dependence relationships estimated via *partial correlations*

Gaussian Graphical Model

- pMRF for continuous data
- Option 1: conditional dependence relationships estimated via *partial correlations*
  1. take variance-covariance matrix (unstandardized correlation matrix) of all nodes

Gaussian Graphical Model

- pMRF for continuous data
- Option 1: conditional dependence relationships estimated via *partial correlations*
  1. take variance-covariance matrix (unstandardized correlation matrix) of all nodes
  2. inverse of the matrix is called *Gaussian Graphical Model*

Gaussian Graphical Model

- pMRF for continuous data
- Option 1: conditional dependence relationships estimated via *partial correlations*
  1. take variance-covariance matrix (unstandardized correlation matrix) of all nodes
  2. inverse of the matrix is called *Gaussian Graphical Model*
  3. standardized inverted matrix: partial correlation matrix

Gaussian Graphical Model

- pMRF for continuous data
- Option 1: conditional dependence relationships estimated via *partial correlations*
  1. take variance-covariance matrix (unstandardized correlation matrix) of all nodes
  2. inverse of the matrix is called *Gaussian Graphical Model*
  3. standardized inverted matrix: partial correlation matrix
- Option 2: conditional dependence relationships estimated via *node-wise regression*

Gaussian Graphical Model

- pMRF for continuous data
- Option 1: conditional dependence relationships estimated via *partial correlations*
  1. take variance-covariance matrix (unstandardized correlation matrix) of all nodes
  2. inverse of the matrix is called *Gaussian Graphical Model*
  3. standardized inverted matrix: partial correlation matrix
- Option 2: conditional dependence relationships estimated via *node-wise regression*
  1. regress each node on all other nodes

Gaussian Graphical Model

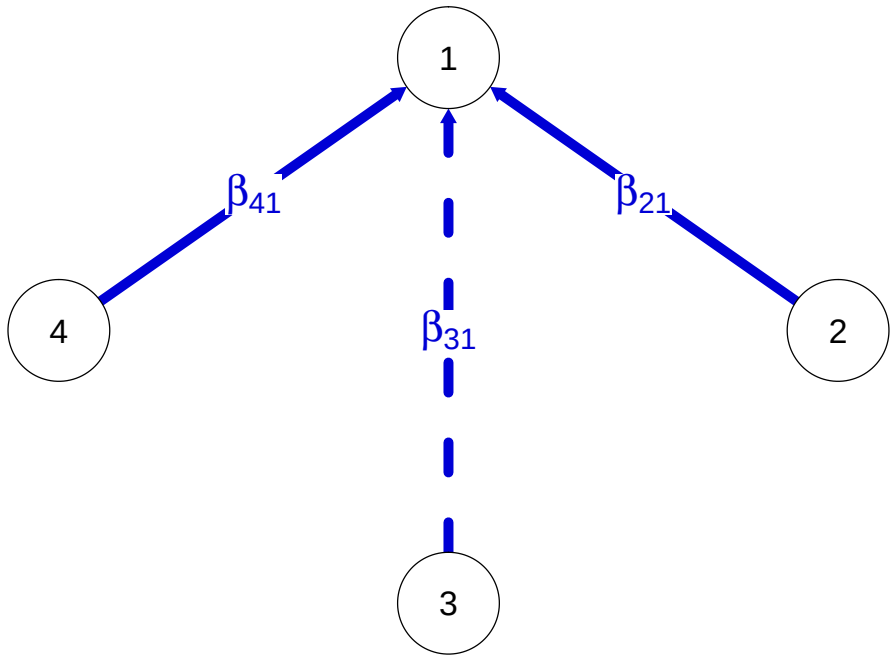
- pMRF for continuous data
- Option 1: conditional dependence relationships estimated via *partial correlations*
  1. take variance-covariance matrix (unstandardized correlation matrix) of all nodes
  2. inverse of the matrix is called *Gaussian Graphical Model*
  3. standardized inverted matrix: partial correlation matrix
- Option 2: conditional dependence relationships estimated via *node-wise regression*
  1. regress each node on all other nodes
  2. save regression weights

Gaussian Graphical Model

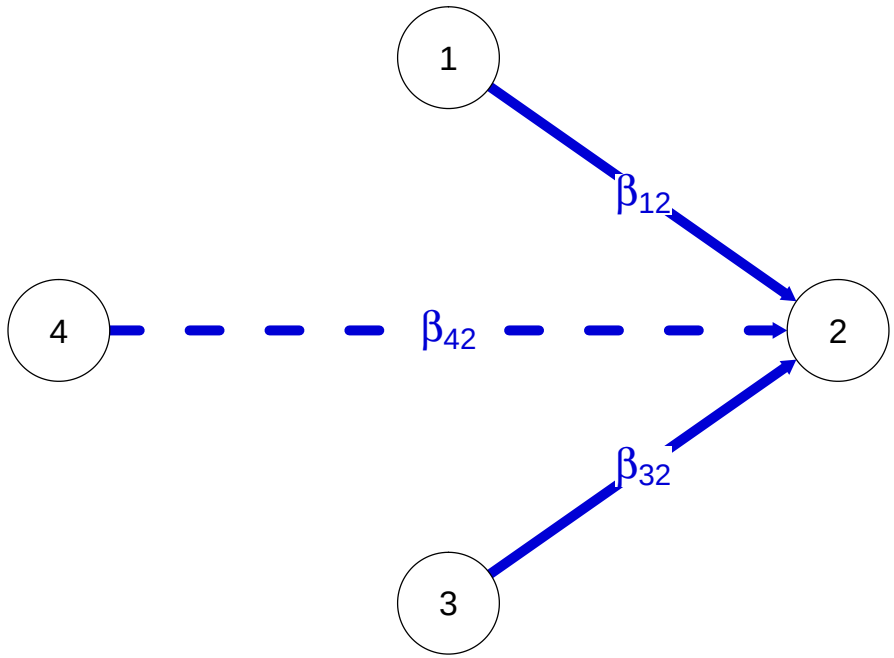
- pMRF for continuous data

- Option 1: conditional dependence relationships estimated via *partial correlations*
  1. take variance-covariance matrix (unstandardized correlation matrix) of all nodes
  2. inverse of the matrix is called *Gaussian Graphical Model*
  3. standardized inverted matrix: partial correlation matrix
- Option 2: conditional dependence relationships estimated via *node-wise regression*
  1. regress each node on all other nodes
  2. save regression weights
  3. apply AND or OR rule to determine edge weights

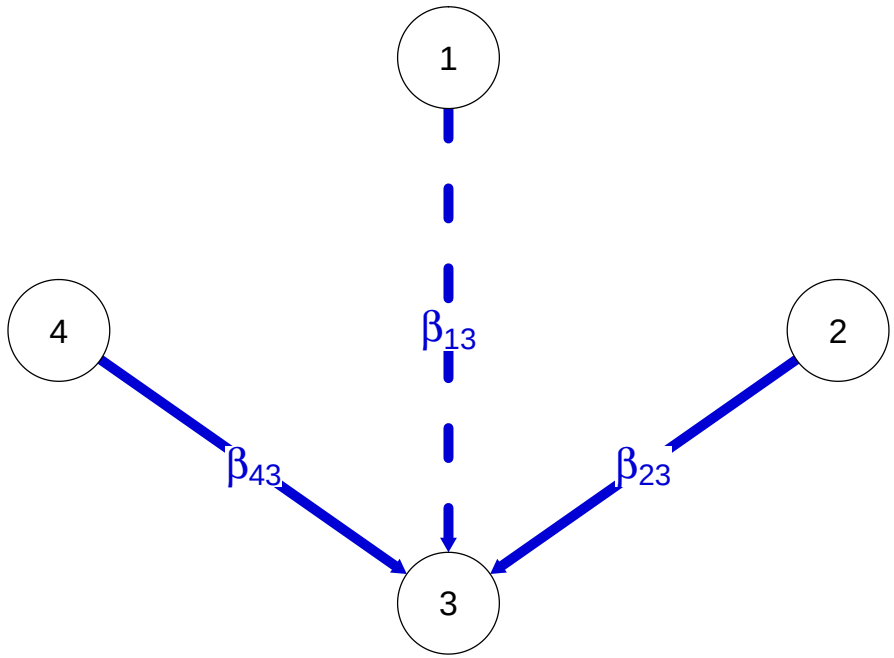
Gaussian Graphical Model – Node-wise Regression



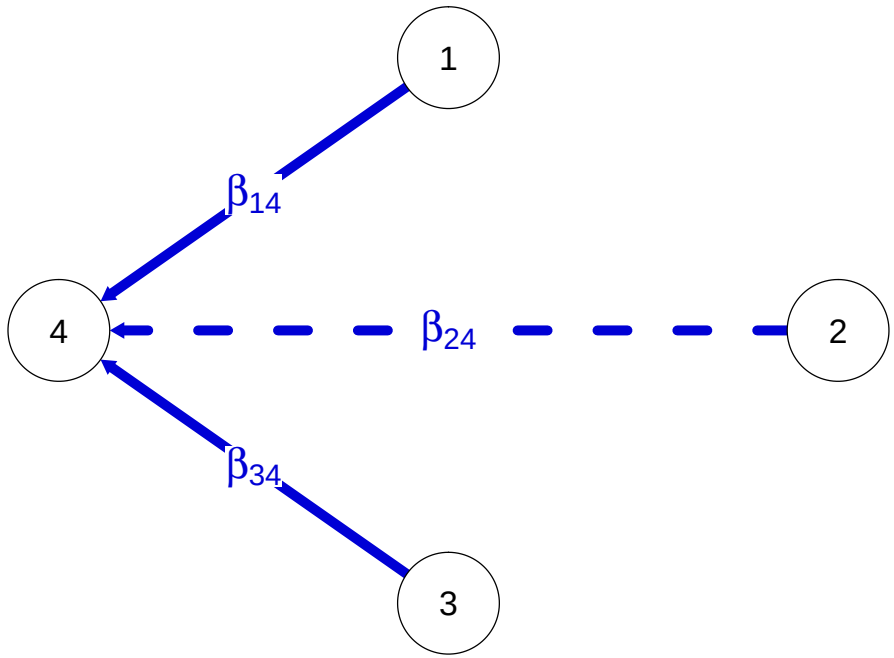
Gaussian Graphical Model – Node-wise Regression



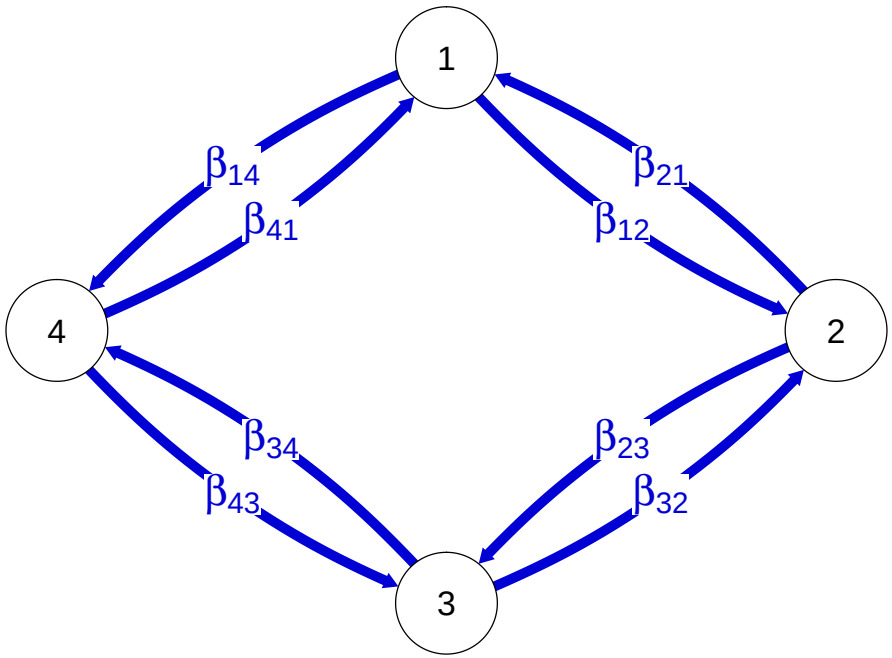
Gaussian Graphical Model – Node-wise Regression



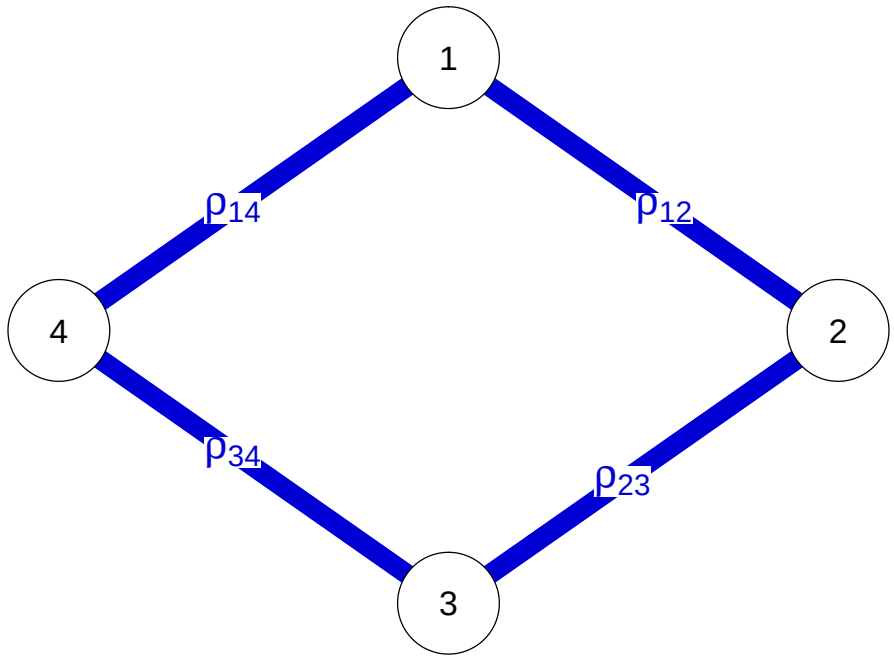
Gaussian Graphical Model – Node-wise Regression



Gaussian Graphical Model – Node-wise Regression



Gaussian Graphical Model – Node-wise Regression



Regularization

- estimation of many parameters

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)
  - limits the size of all edges

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)
  - limits the size of all edges
  - sets small edges to exactly zero

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)
  - limits the size of all edges
  - sets small edges to exactly zero
  - varying tuning parameter λ changes sparsity (100 networks)

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)
  - limits the size of all edges
  - sets small edges to exactly zero
  - varying tuning parameter λ changes sparsity (100 networks)
  - minimize Extended Bayesian Information Criterion (EBIC)

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)
  - limits the size of all edges
  - sets small edges to exactly zero
  - varying tuning parameter λ changes sparsity (100 networks)
  - minimize Extended Bayesian Information Criterion (EBIC)
  - hyperparamter γ controls EBIC

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)
  - limits the size of all edges
  - sets small edges to exactly zero
  - varying tuning parameter λ changes sparsity (100 networks)
  - minimize Extended Bayesian Information Criterion (EBIC)
  - hyperparamter γ controls EBIC
    - 0: err on the side of discovery

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)
  - limits the size of all edges
  - sets small edges to exactly zero
  - varying tuning parameter λ changes sparsity (100 networks)
  - minimize Extended Bayesian Information Criterion (EBIC)
  - hyperparamter γ controls EBIC
    - 0: err on the side of discovery
    - 0.5: err on the side of caution

Regularization

- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

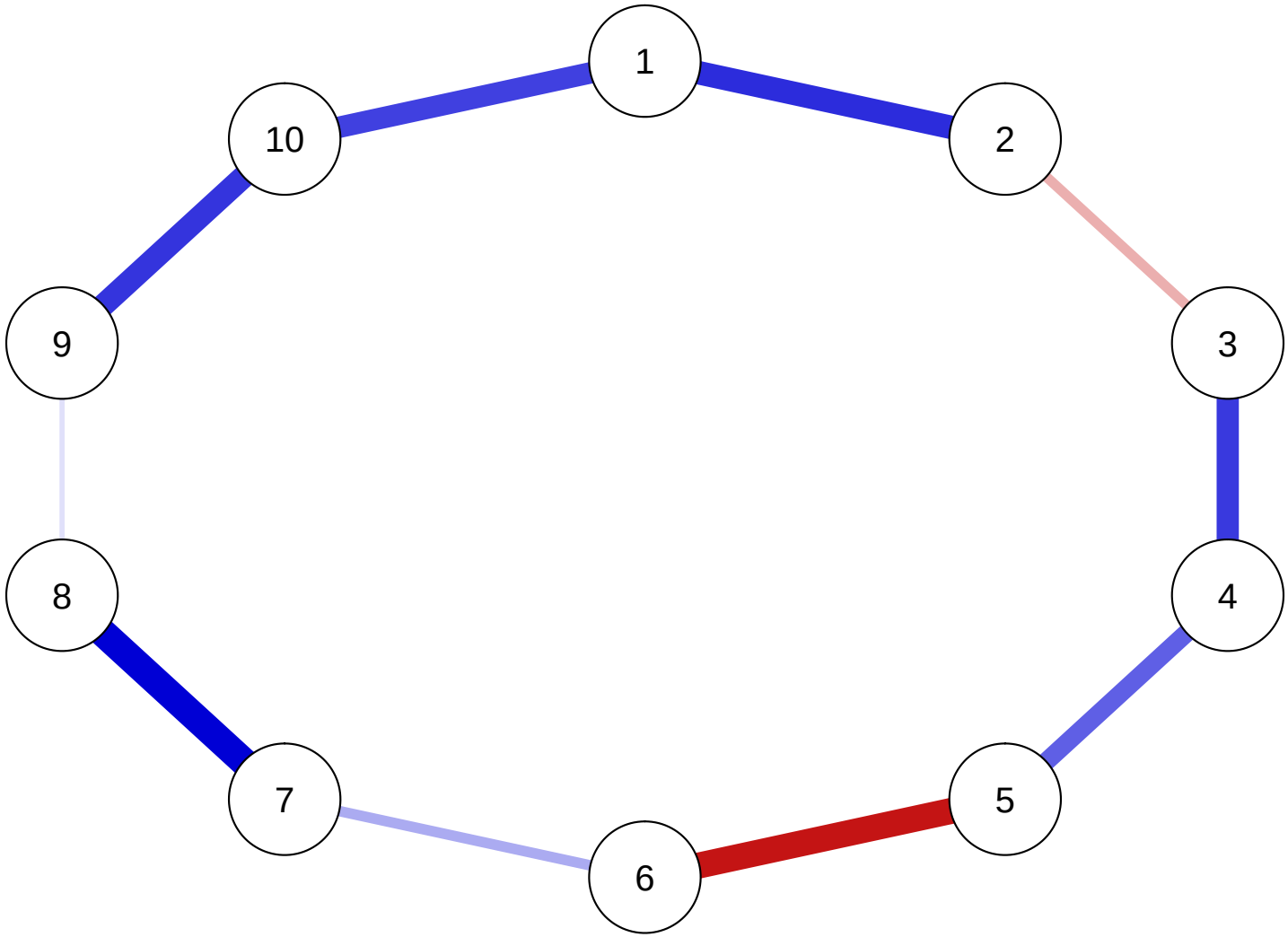
- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)
  - limits the size of all edges
  - sets small edges to exactly zero
  - varying tuning parameter λ changes sparsity (100 networks)
  - minimize Extended Bayesian Information Criterion (EBIC)
  - hyperparamter γ controls EBIC
    - 0: err on the side of discovery
    - 0.5: err on the side of caution
    - typically set 0.25 or 0.5

Regularization

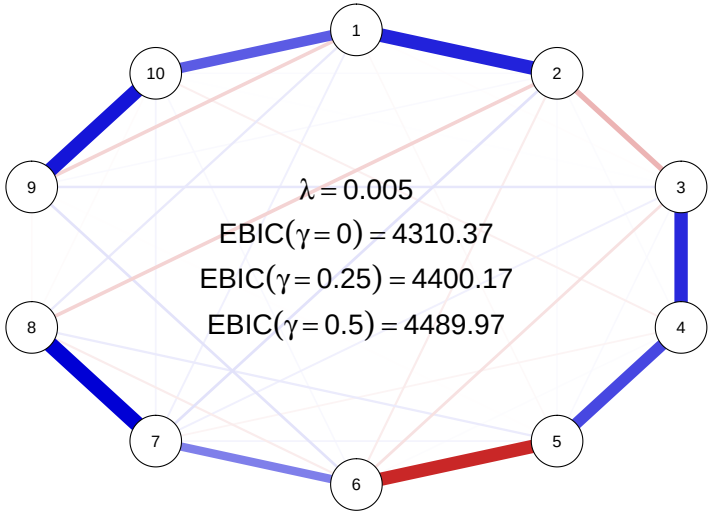
- estimation of many parameters
  - k nodes: $k * (k-1)/2$ edges
  - preferred model should be sparse and true

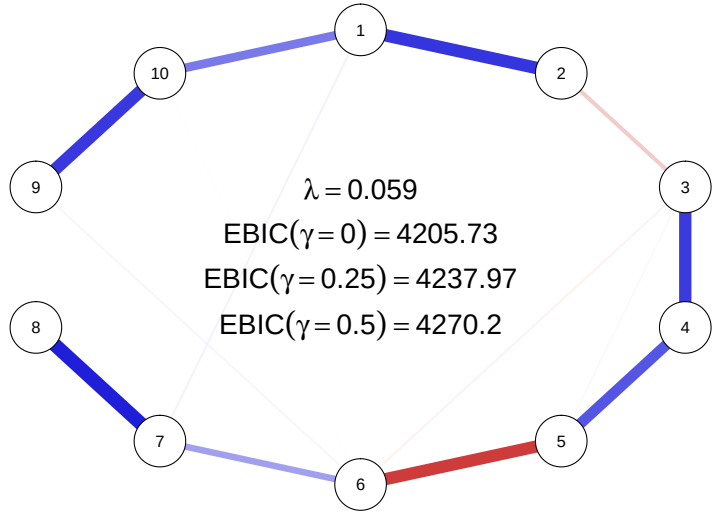
- regularization: Least Absolute Shrinkage and Selection Operator (LASSO)
  - limits the size of all edges
  - sets small edges to exactly zero
  - varying tuning parameter λ changes sparsity (100 networks)
  - minimize Extended Bayesian Information Criterion (EBIC)
  - hyperparamter γ controls EBIC
    - 0: err on the side of discovery
    - 0.5: err on the side of caution
    - typically set 0.25 or 0.5
  - LASSO for partial correlations (gLASSO) and node-wise regression (eLASSO)

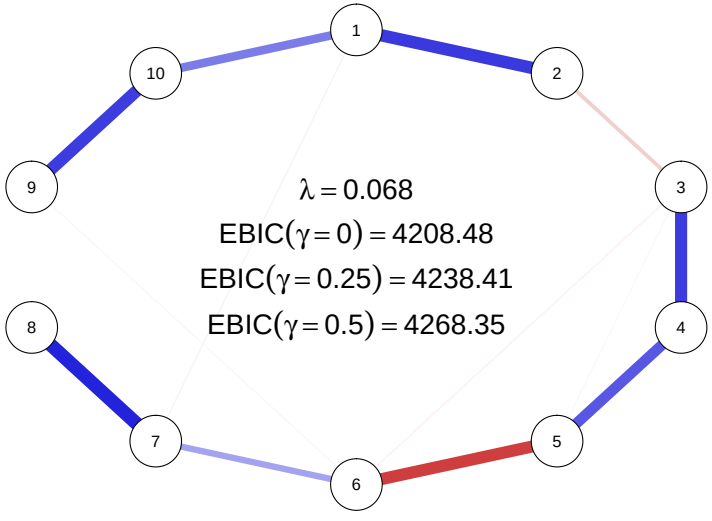
Regularization – Example

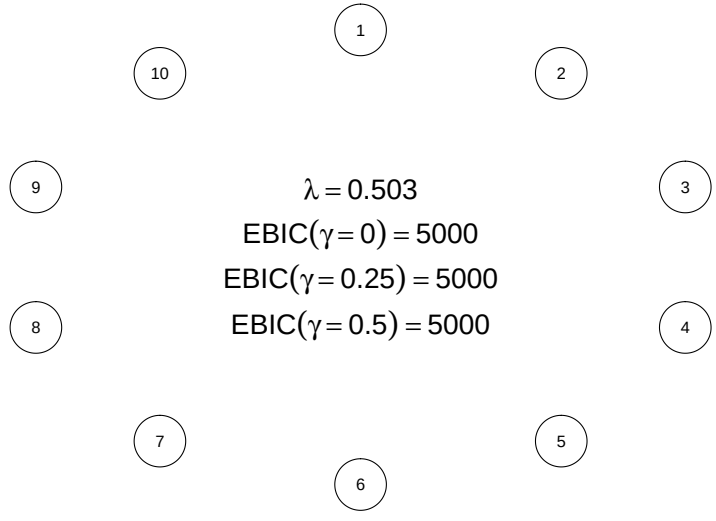


Regularization – Example









Ising Model

- pMRF for binary data

Ising Model

- pMRF for binary data
- estimation via node-wise regression

Ising Model

- pMRF for binary data
- estimation via node-wise regression
  - logistic regression

Ising Model

- pMRF for binary data
- estimation via node-wise regression
  - logistic regression
  - regularization via eLASSO

Clique Percolation Community Detection

Community Detection

- *communities*: strongly connected subgraphs

Community Detection

- *communities*: strongly connected subgraphs
- *clique percolation algorithm*

Community Detection

- *communities*: strongly connected subgraphs
- *clique percolation algorithm*
  1. Identify all k-cliques.

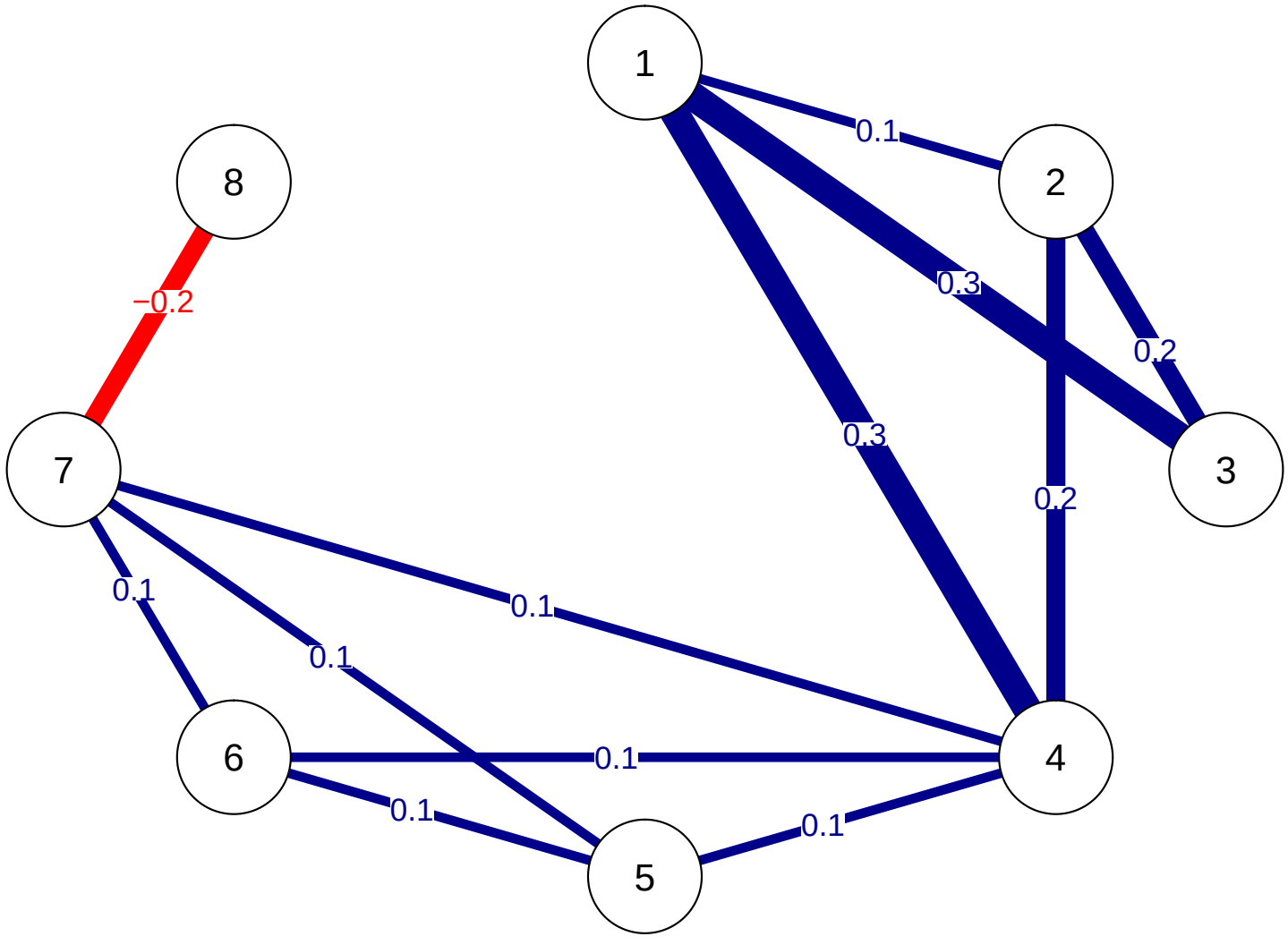
Community Detection

- *communities*: strongly connected subgraphs
- *clique percolation algorithm*
  1. Identify all k-cliques.
  2. Check whether Intensity of k-cliques exceeds threshold I.

Community Detection

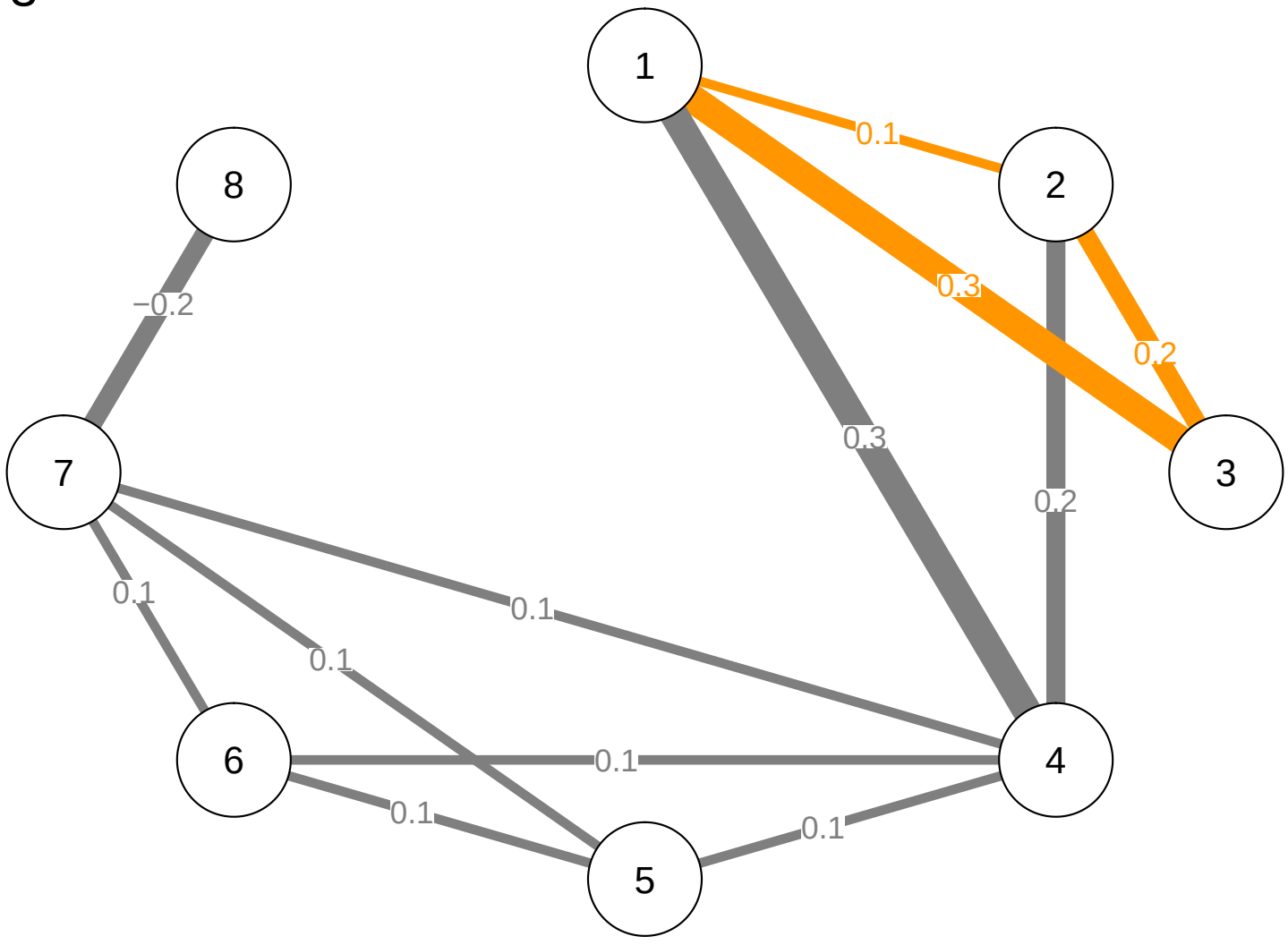
- *communities*: strongly connected subgraphs
- *clique percolation algorithm*
  1. Identify all k-cliques.
  2. Check whether Intensity of k-cliques exceeds threshold I.
  3. Put k-cliques that share k-1 nodes in a community.

Identify k-cliques



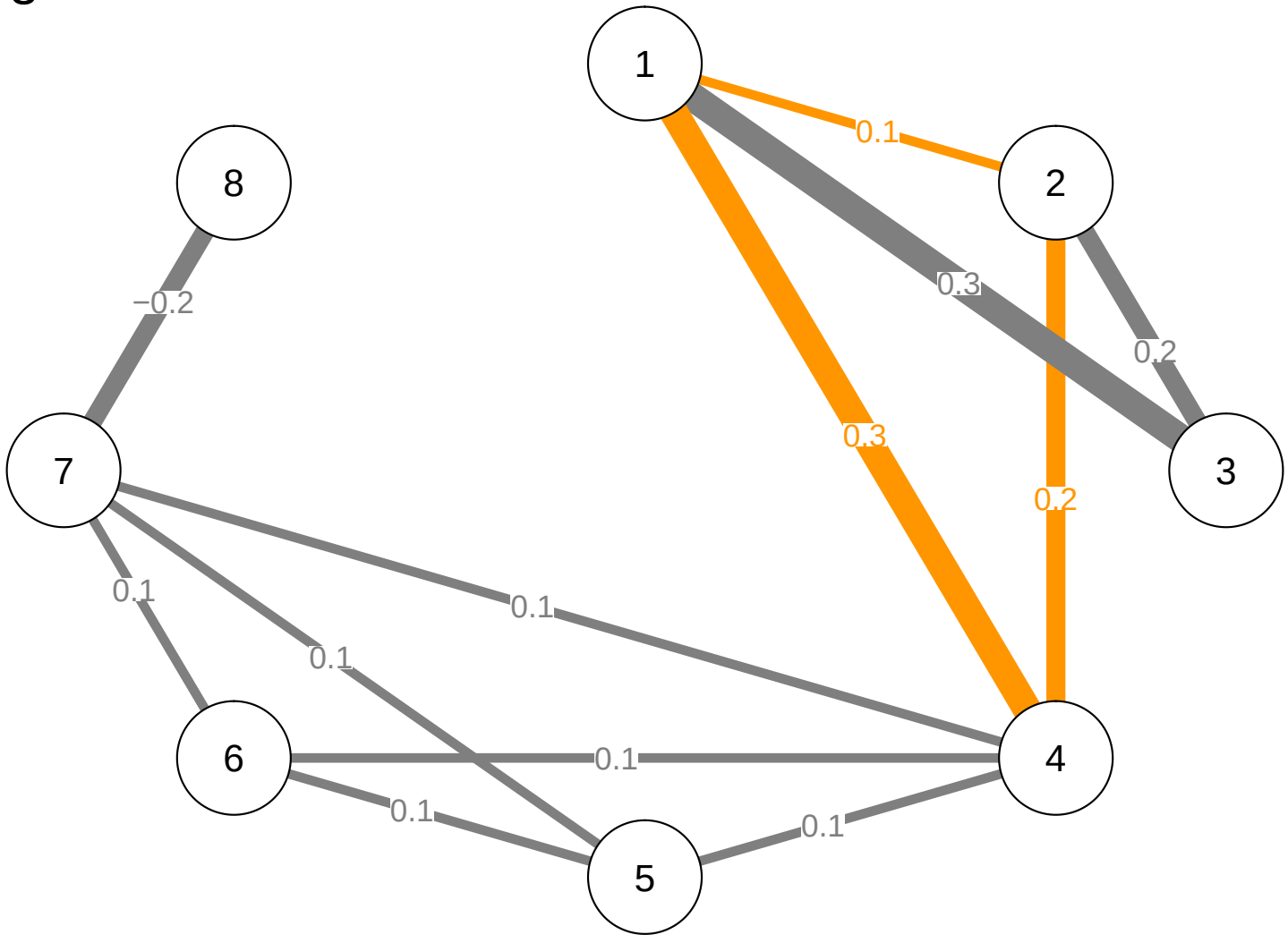
Identify k-cliques

k = 3



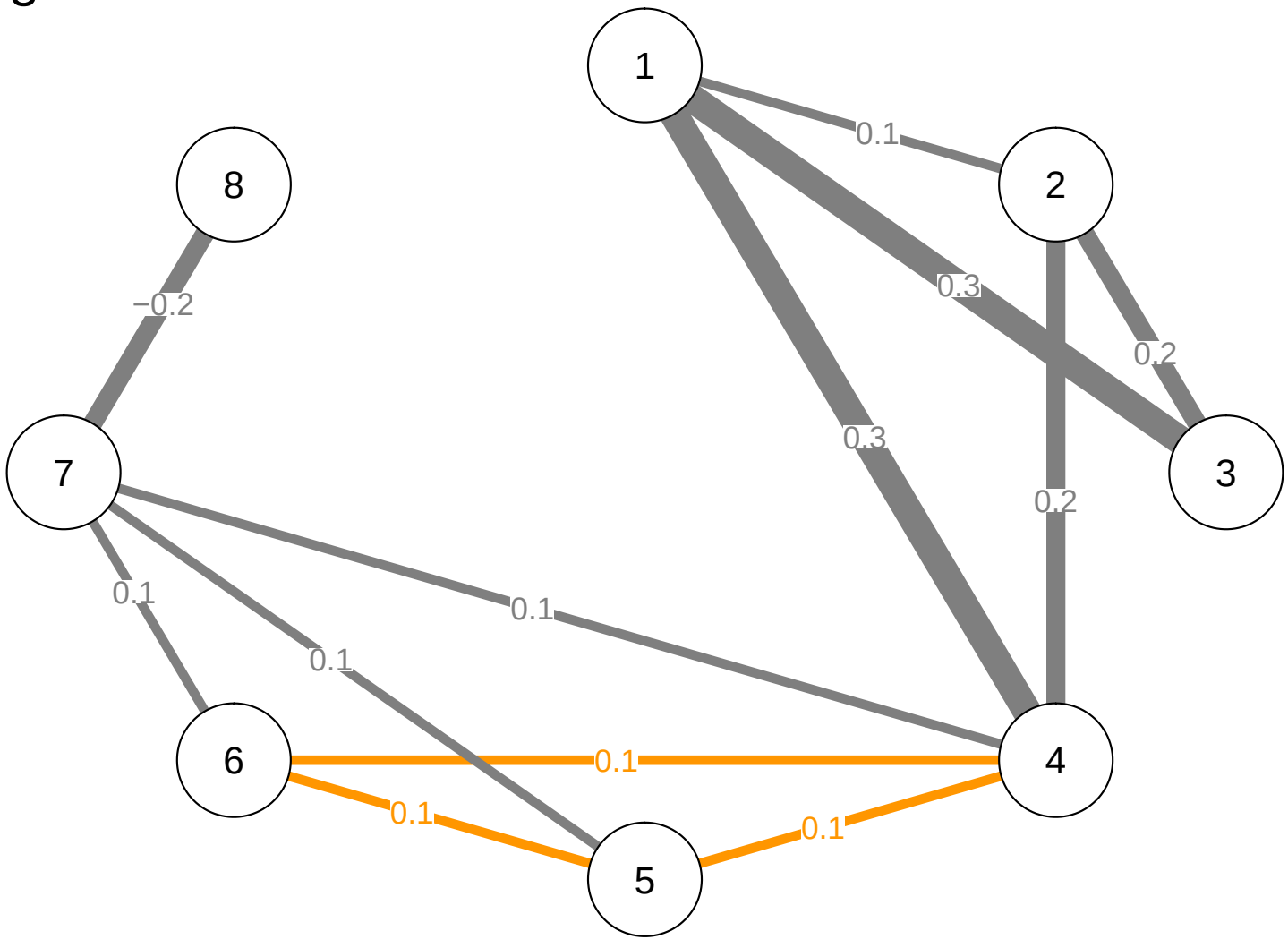
Identify k-cliques

k = 3



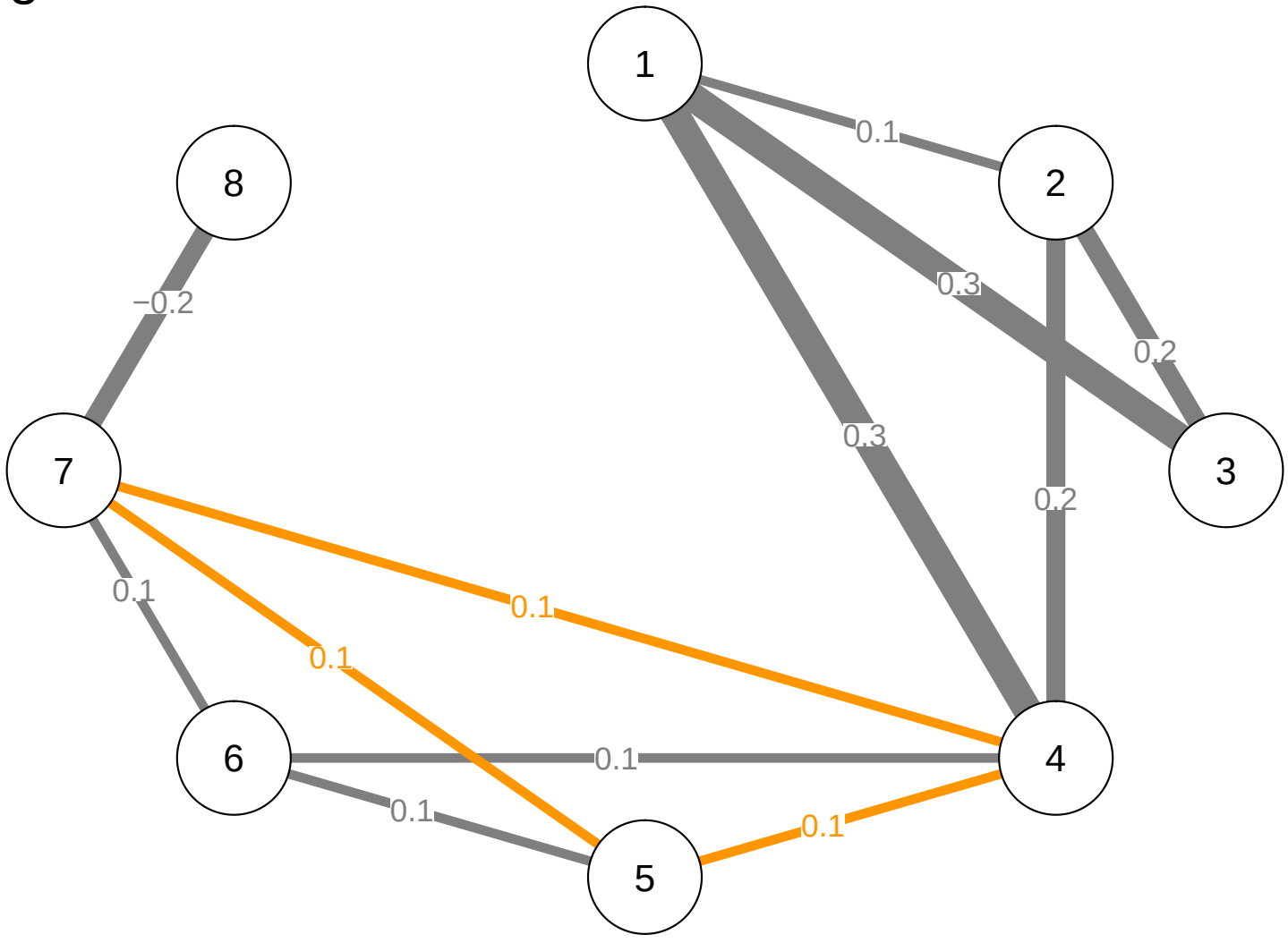
Identify k-cliques

k = 3



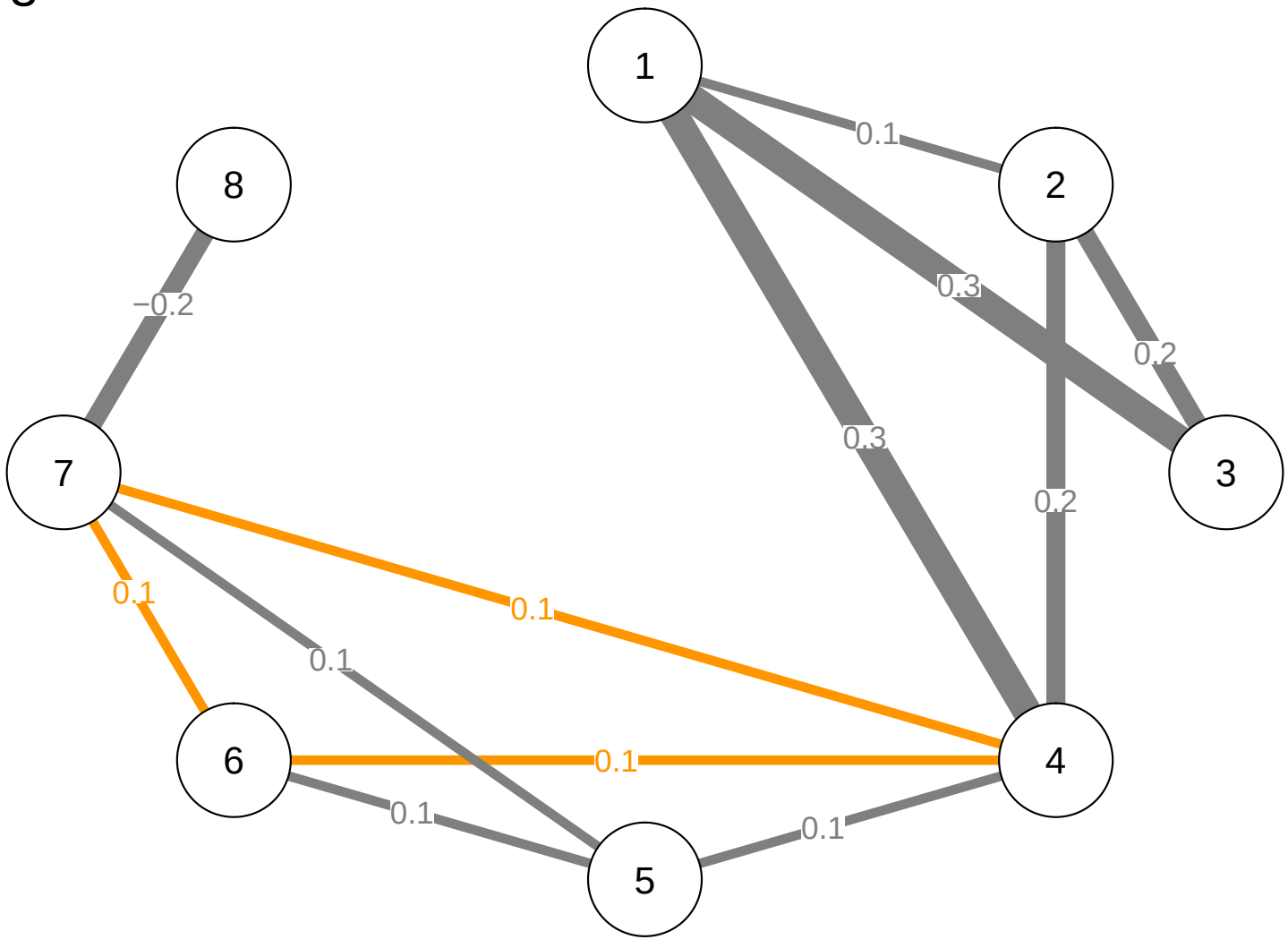
Identify k-cliques

k = 3



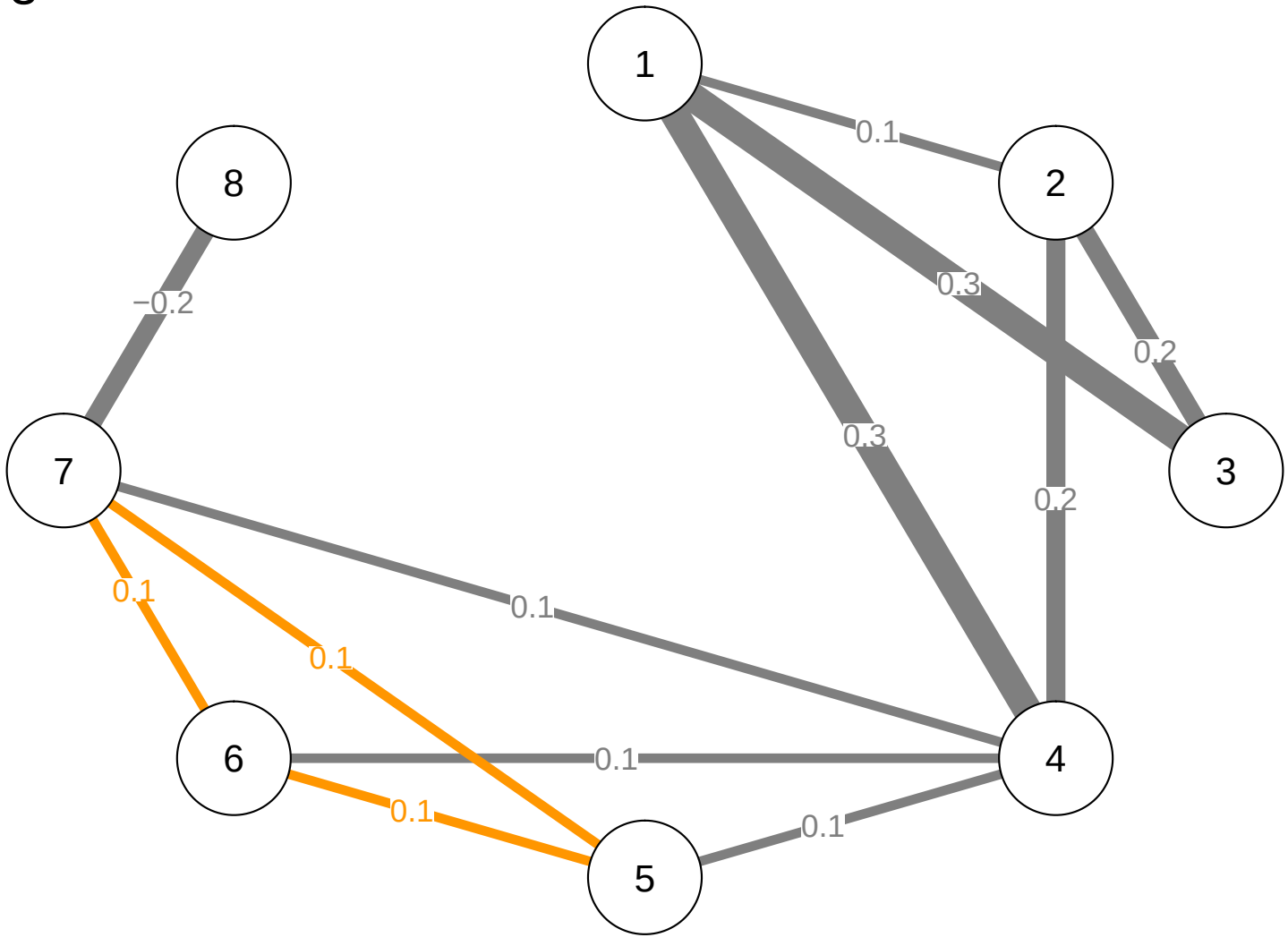
Identify k-cliques

k = 3



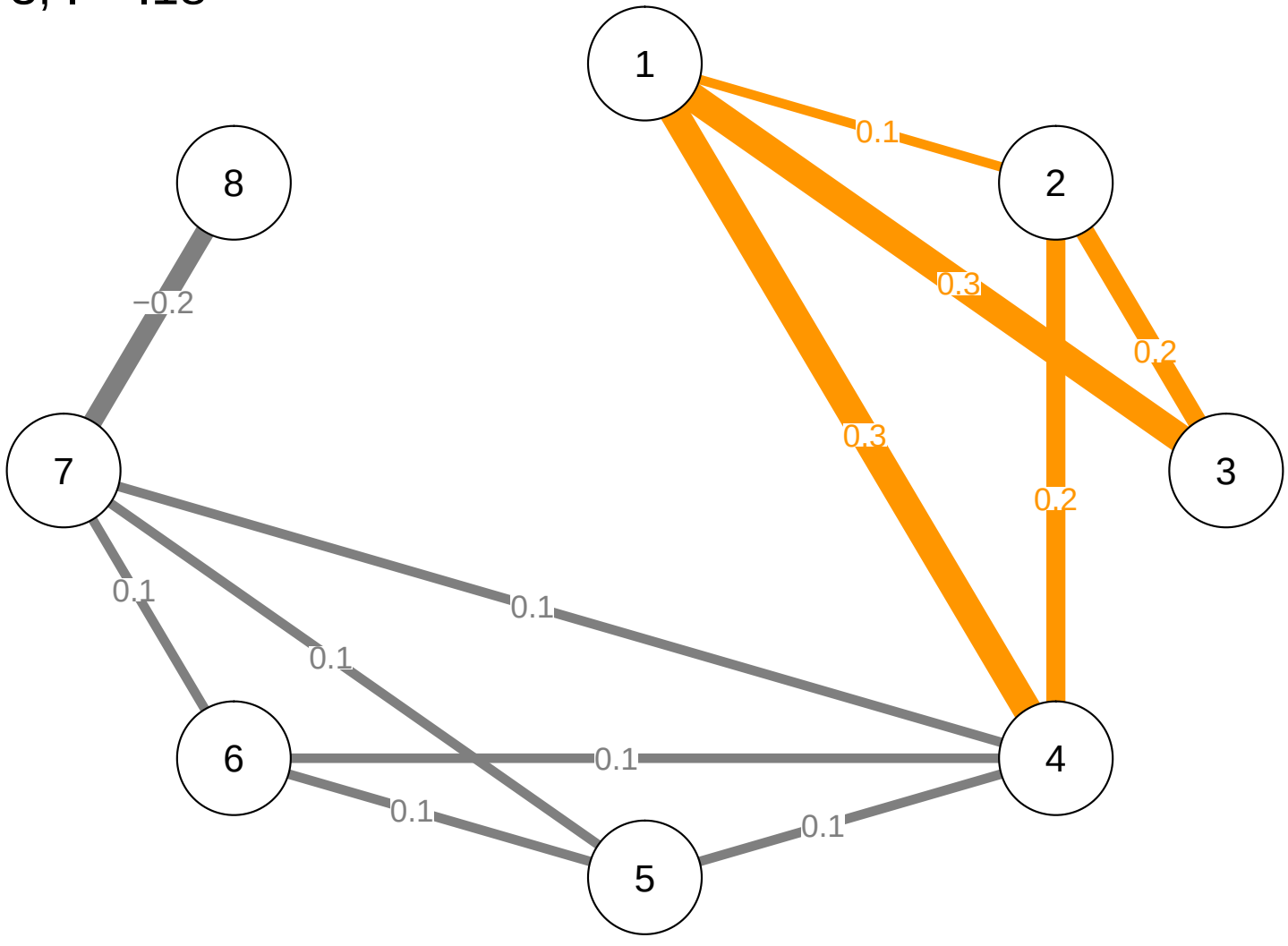
Identify k-cliques

k = 3



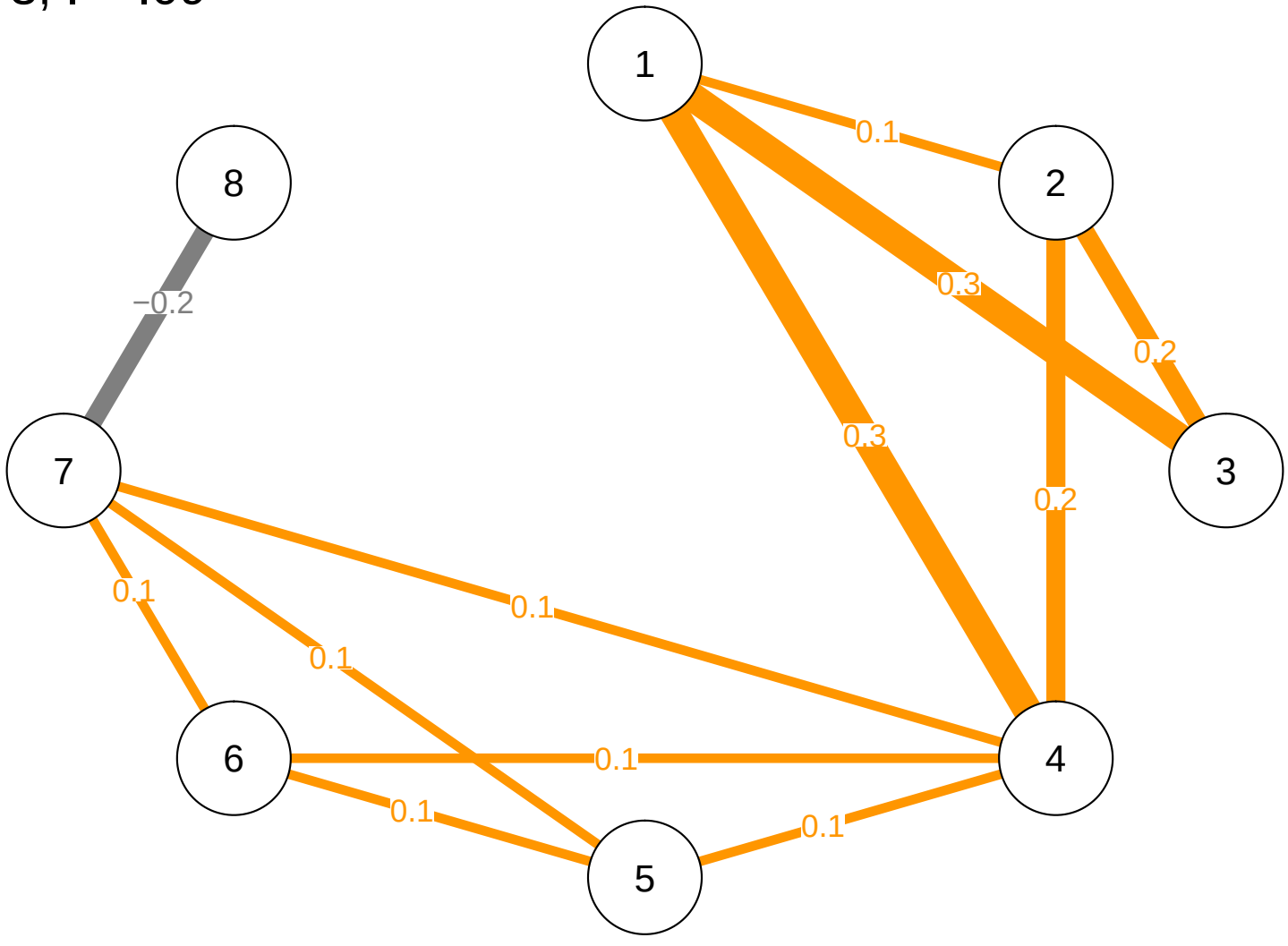
k-cliques whose Intensity Exceeds I

k = 3, I = .18



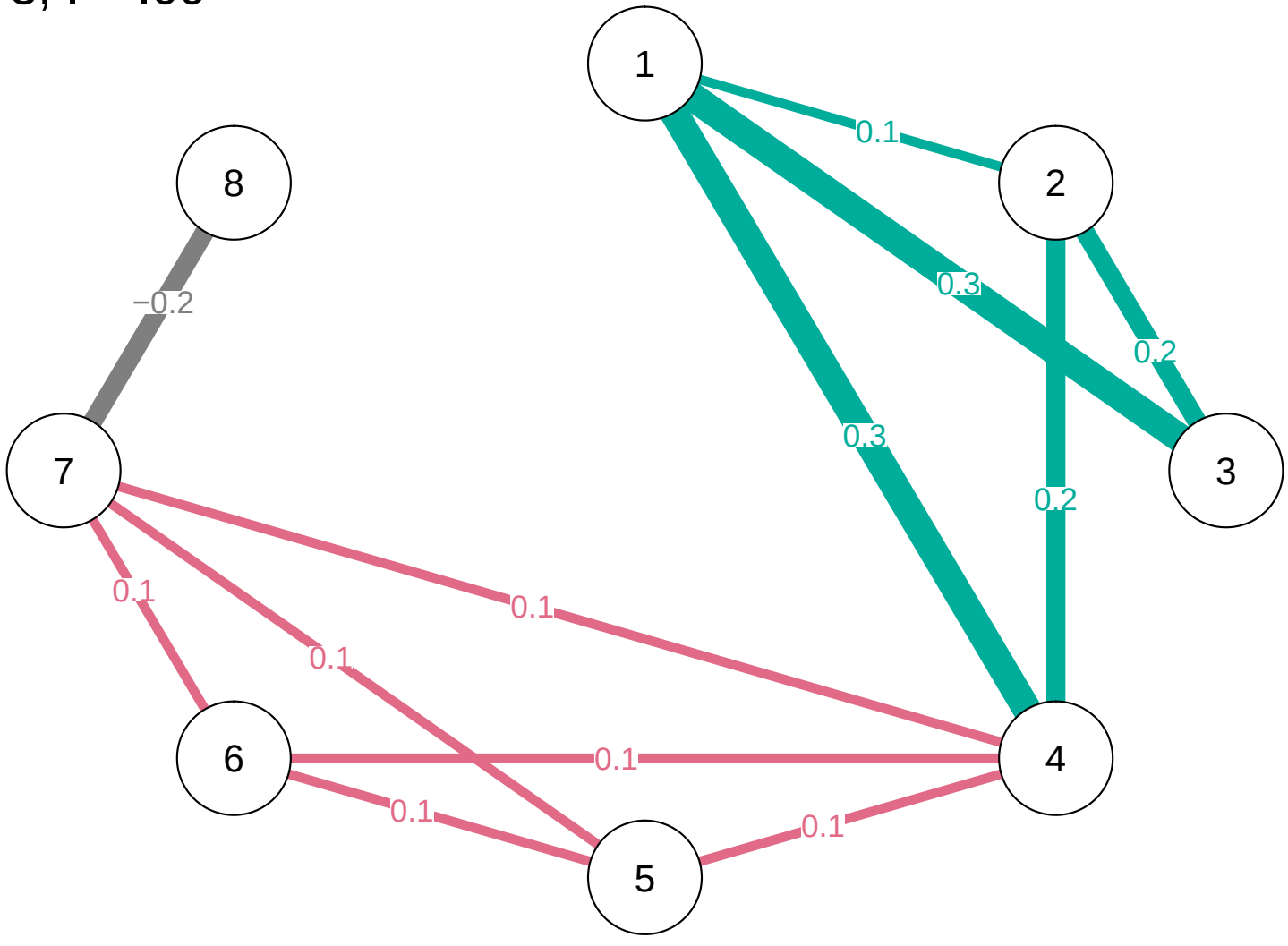
k-cliques whose Intensity Exceeds I

k = 3, I = .09



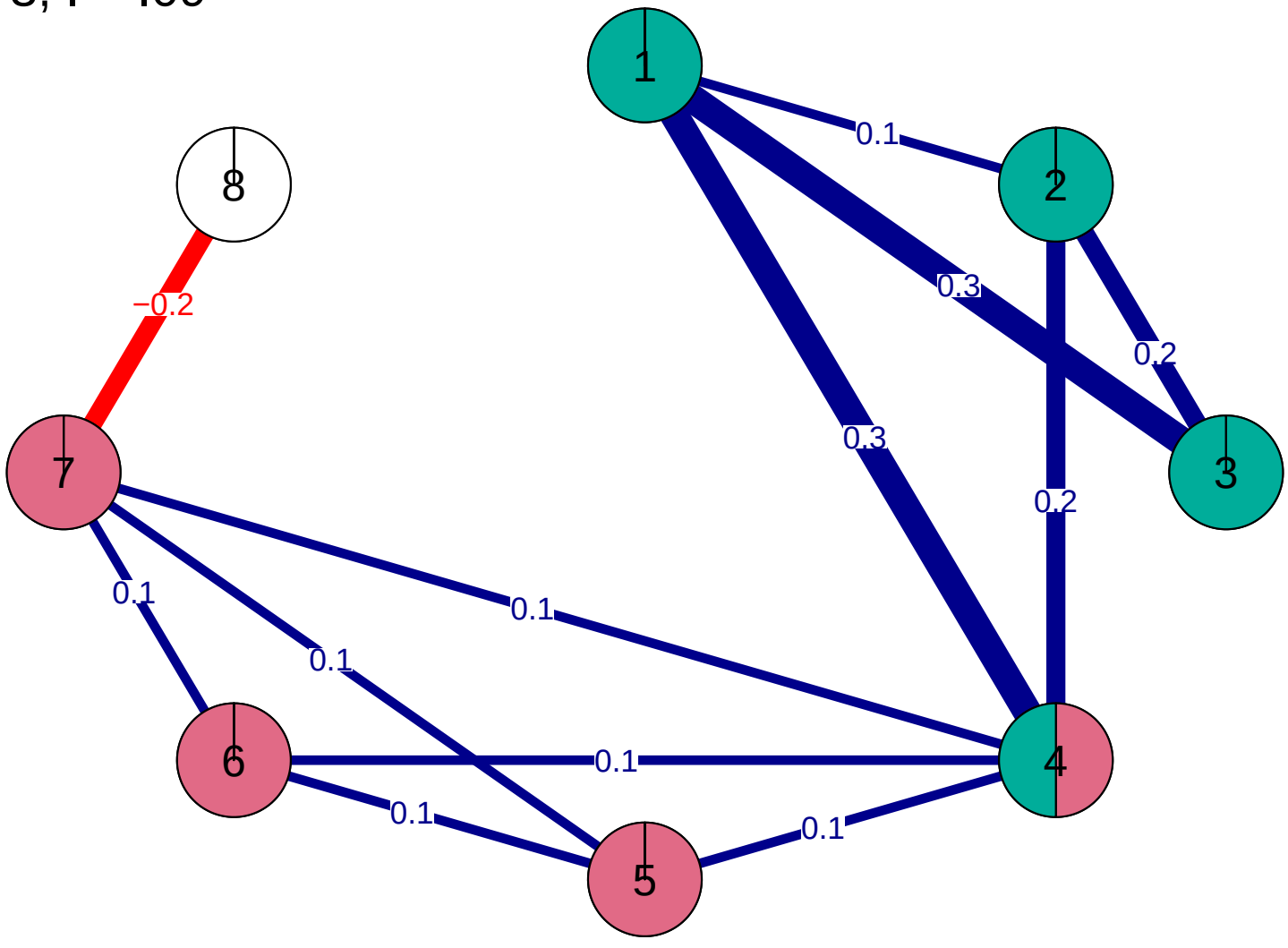
Identify Communities

k = 3, I = .09



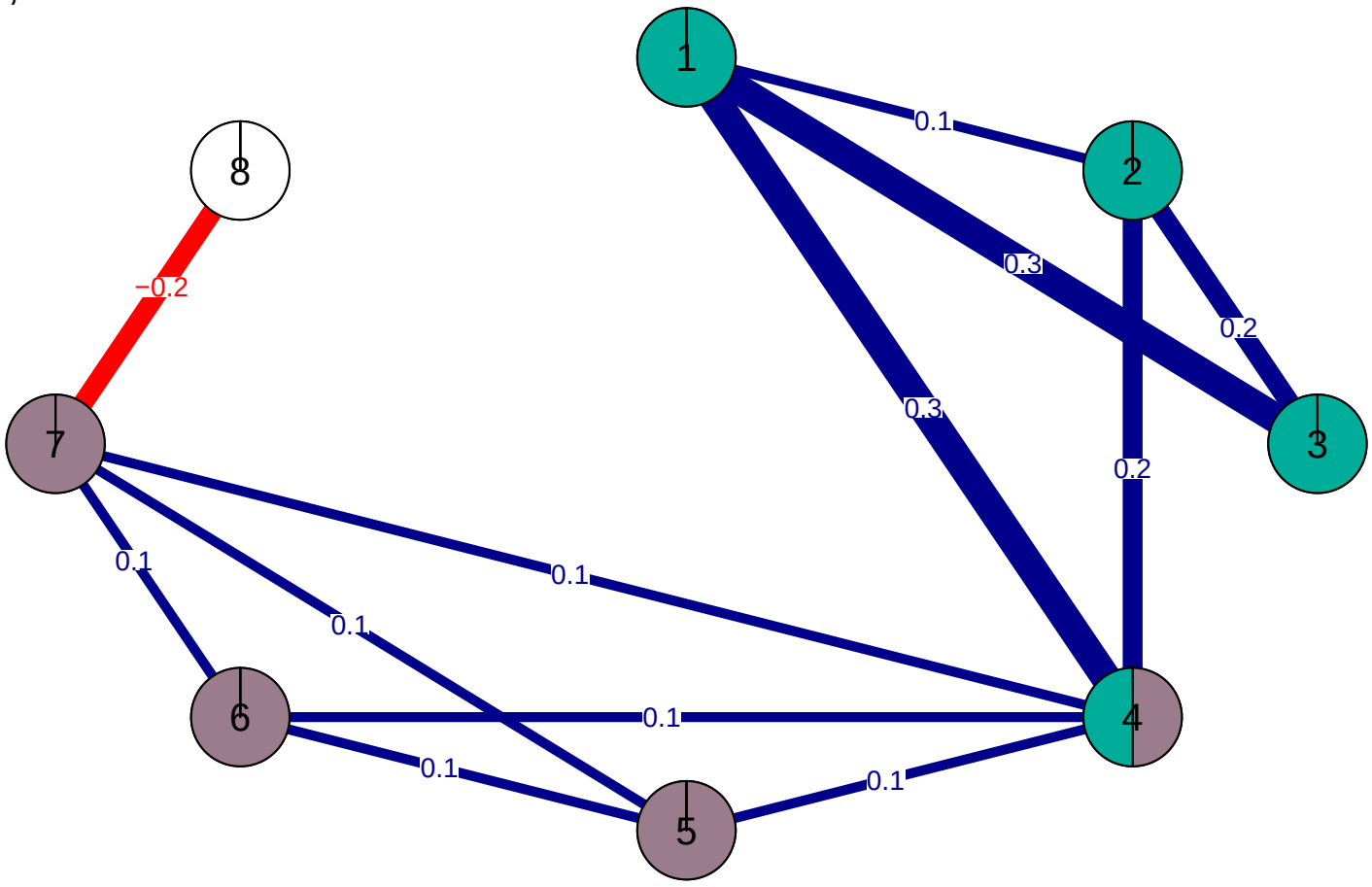
Identify Communities

k = 3, I = .09



Identify Communities

k = 3, l = .09



Emotion 1

Emotion 2

Optimizing k and l

- large networks

Optimizing k and l

- large networks
  - ratio of largest to second largest community

Optimizing k and I

- large networks
  - ratio of largest to second largest community
  - optimal I for a k: ratio crosses 2

Optimizing k and I

- large networks
  - ratio of largest to second largest community
  - optimal I for a k: ratio crosses 2
  - choose k & I with broadest distribution

Optimizing k and I

- large networks
  - ratio of largest to second largest community
  - optimal I for a k: ratio crosses 2
  - choose k & I with broadest distribution
- small networks

Optimizing k and I

- large networks
  - ratio of largest to second largest community
  - optimal I for a k: ratio crosses 2
  - choose k & I with broadest distribution
- small networks
  - variance χ of community sizes (after excluding largest community)

Optimizing k and I

- large networks
  - ratio of largest to second largest community
  - optimal I for a k: ratio crosses 2
  - choose k & I with broadest distribution
- small networks
  - variance χ of community sizes (after excluding largest community)
  - optimal I for a k: maximal χ

Optimizing k and I

- large networks
  - ratio of largest to second largest community
  - optimal I for a k: ratio crosses 2
  - choose k & I with broadest distribution
- small networks
  - variance χ of community sizes (after excluding largest community)
  - optimal I for a k: maximal χ
  - choose k & I with broadest distribution

Optimizing k and l

- large networks
  - ratio of largest to second largest community
  - optimal l for a k: ratio crosses 2
  - choose k & l with broadest distribution
- small networks
  - variance χ of community sizes (after excluding largest community)
  - optimal l for a k: maximal χ
  - choose k & l with broadest distribution
- very small networks (as in psychology)

Optimizing k and l

- large networks
  - ratio of largest to second largest community
  - optimal l for a k: ratio crosses 2
  - choose k & l with broadest distribution
- small networks
  - variance χ of community sizes (after excluding largest community)
  - optimal l for a k: maximal χ
  - choose k & l with broadest distribution
- very small networks (as in psychology)
  - entropy (i.e., surprisingness) of community sizes (treating isolated nodes as community)

Optimizing k and l

- large networks
  - ratio of largest to second largest community
  - optimal l for a k: ratio crosses 2
  - choose k & l with broadest distribution
- small networks
  - variance χ of community sizes (after excluding largest community)
  - optimal l for a k: maximal χ
  - choose k & l with broadest distribution
- very small networks (as in psychology)
  - entropy (i.e., surprisingness) of community sizes (treating isolated nodes as community)
  - optimal l for a k: maximal entropy that is higher than chance (permutation test)

Optimizing k and l

- large networks
  - ratio of largest to second largest community
  - optimal l for a k: ratio crosses 2
  - choose k & l with broadest distribution
- small networks
  - variance χ of community sizes (after excluding largest community)
  - optimal l for a k: maximal χ
  - choose k & l with broadest distribution
- very small networks (as in psychology)
  - entropy (i.e., surprisingness) of community sizes (treating isolated nodes as community)
  - optimal l for a k: maximal entropy that is higher than chance (permutation test)
  - choose k & l with highest entropy and fewest isolated nodes

Network Analysis in R

OSF Files

https://osf.io/b4gc7/

Get Started

- set working directory

```
setwd("...")
```

- load packages

```
library(qgraph)            #plotting networks; version 1.6.3
library(bootnet)           #estimating networks; version 1.2.3
library(psych)             #psych statistics; version 1.8.12
library(rio)               #import/export data; version 0.5.16
library(CliquePercolation) #clique percolation community detection; version 0.2.0
```

Load Data

```r
#load data - ratings of Obama
data <- import("Data_mixed.csv")

#Items
#Scale from from 0 (not at all) to 6 (very much)
#Awe
#AS_1 = "I feel my jaw drop"
#AS_2 = "I gasp"
#AS_3 = "I feel that I am in the presence of something grand"
#AS_4 = "I sense things momentarily slow down"
#AS_5 = "I have goosebumps"
#AS_6 = "I feel challenged to mentally process what I am experiencing"
#AS_7 = "I feel my eyes widen"
#AS_8 = "I have the sense of being connected to everything"
#AS_9 = "I have chills"
#AS_10 = "I feel awed"
#AS_11 = "I feel that my sense of self is diminished"
#
#Kama Muta
#KA_1 = "I have moist eyes"
#KA_2 = "I have positive feelings"
#KA_3 = "I feel like telling someone how much I care about them"
#KA_4 = "I have difficulty speaking"
#KA_5 = "A warm feeling in the center of the chest"
#KA_6 = "I feel refreshed, energized, or exhilarated"
#KA_7 = "A lump in the throat"
#KA_8 = "I am moved"
#KA_9 = "I feel buoyant or light"
#KA_10 = "Some feeling in the center of the chest"
#KA_11 = "I feel/observed an incredible bond"
#KA_12 = "It is heartwarming"
#KA_13 = "I smile"
#KA_14 = "I shed tears"
#KA_15 = "I feel choked up"
#KA_16 = "I am touched"
```

Descriptive Statistics

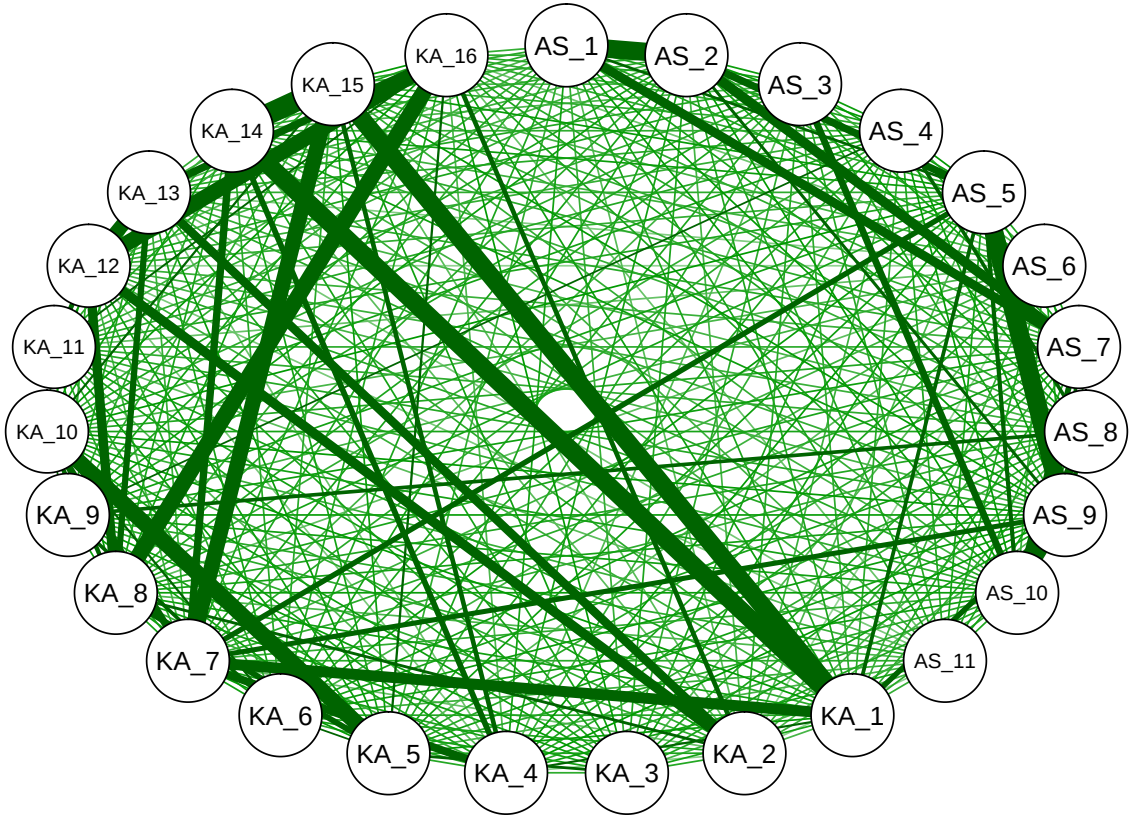
```
describe(data)
#-> one missing value
#-> no obvious skewness or kurtosis
```

Zero-order correlations

```
#automatic detection of reasonable kind of correlation
#here polychoric correlations
correlations <- cor_auto(data)
correlations
```

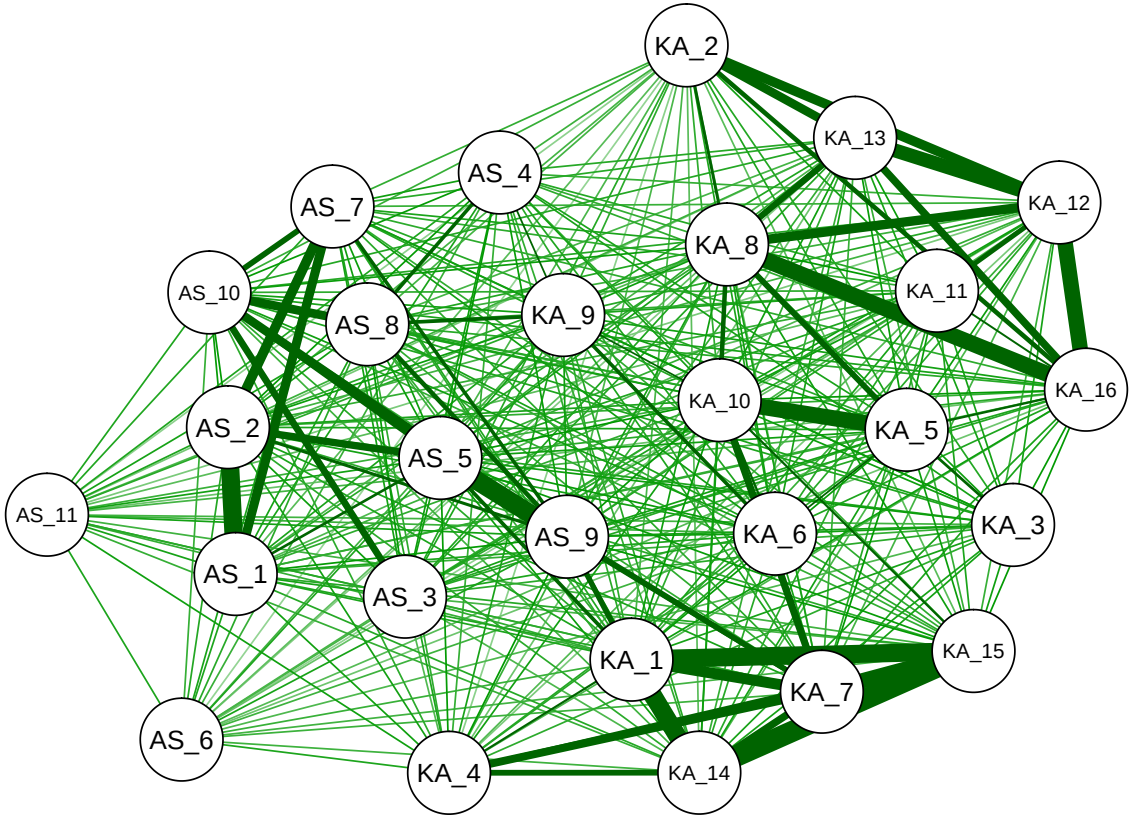
Plotting Correlations with qgraph

```
#plotting correlations with qgraph
cor_graph <- qgraph(correlations)
```



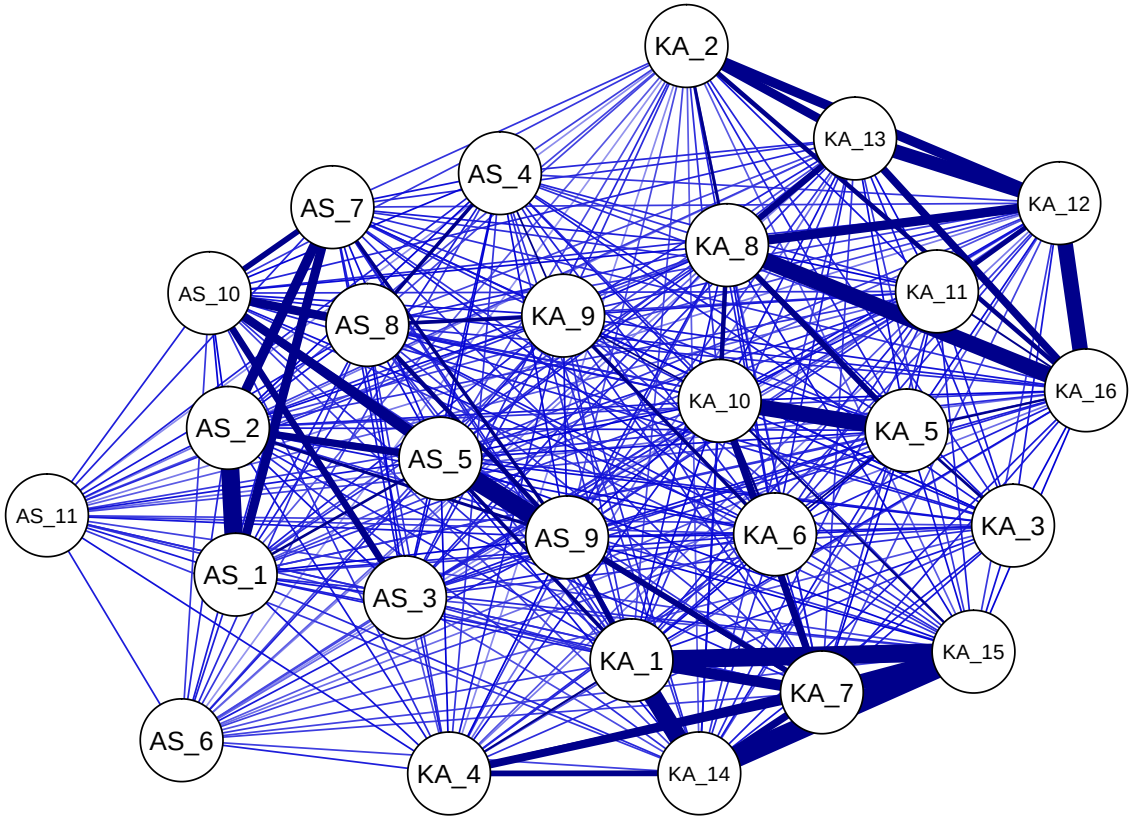
Plotting Correlations with qgraph

```
#+ using Fruchterman-Reingold
cor_graph <- qgraph(correlations, layout = "spring")
```



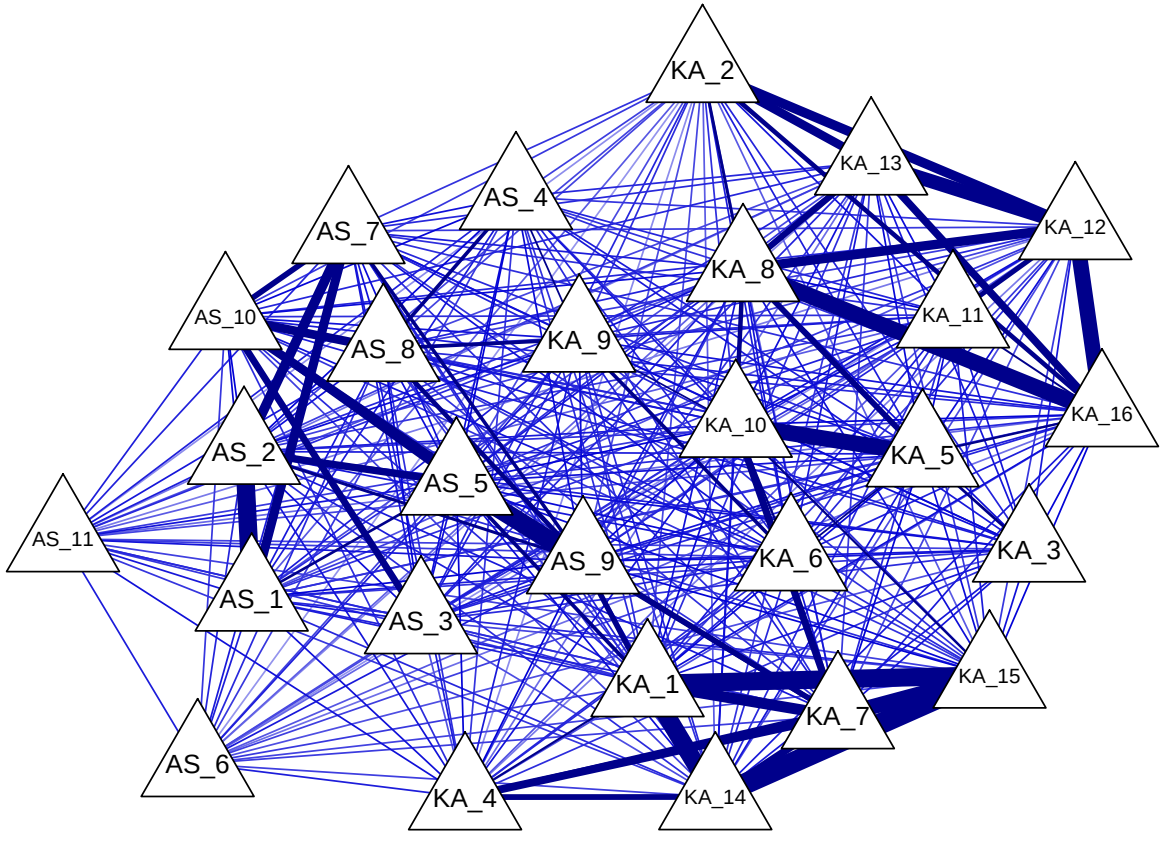
Plotting Correlations with qgraph

```
#+ change for color-blind people
cor_graph <- qgraph(correlations, layout = "spring",
                    theme = "colorblind")
```



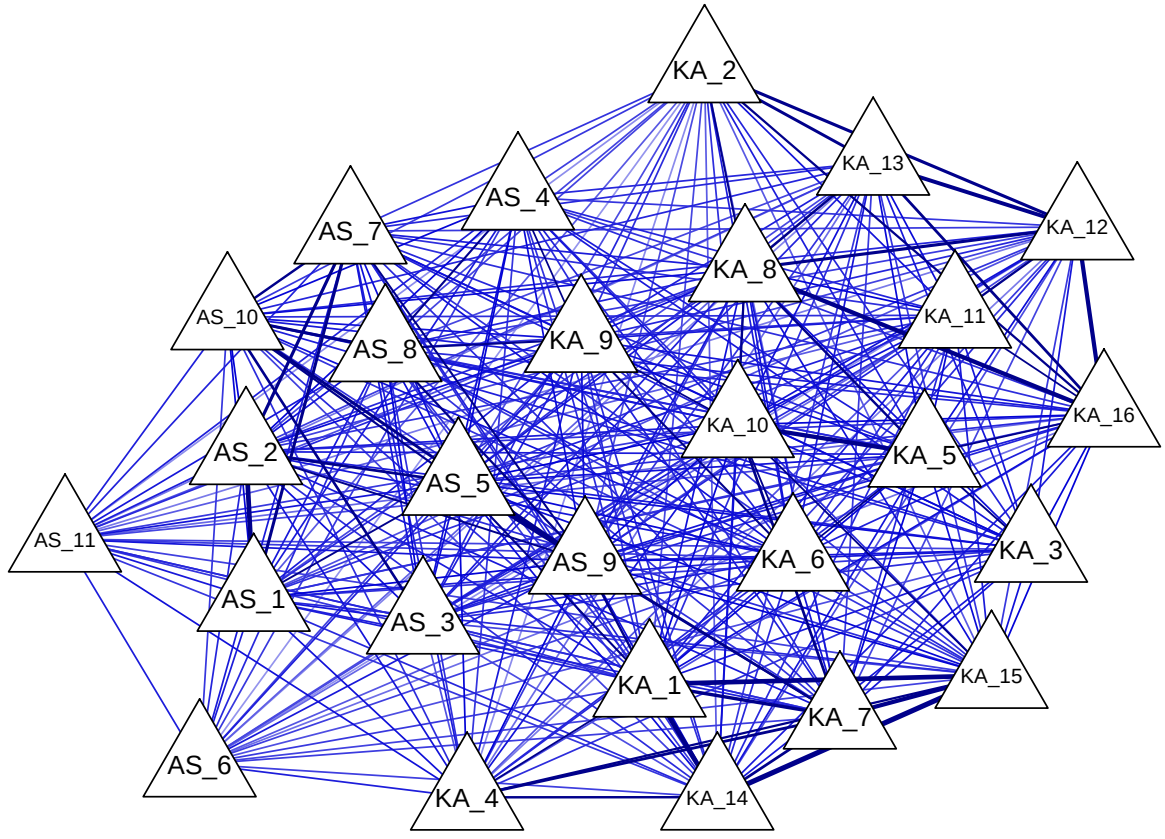
Plotting Correlations with qgraph

```
#+ change shape of nodes
cor_graph <- qgraph(correlations, layout = "spring",
                    theme = "colorblind", shape = "triangle")
```



Plotting Correlations with qgraph

```
#+ change edge width scale
cor_graph <- qgraph(correlations, layout = "spring",
                    theme = "colorblind", shape = "triangle",
                    esize = 3)
```



Plotting Correlations with qgraph

```
#for list of features see options in qgraph help page
?qgraph
```

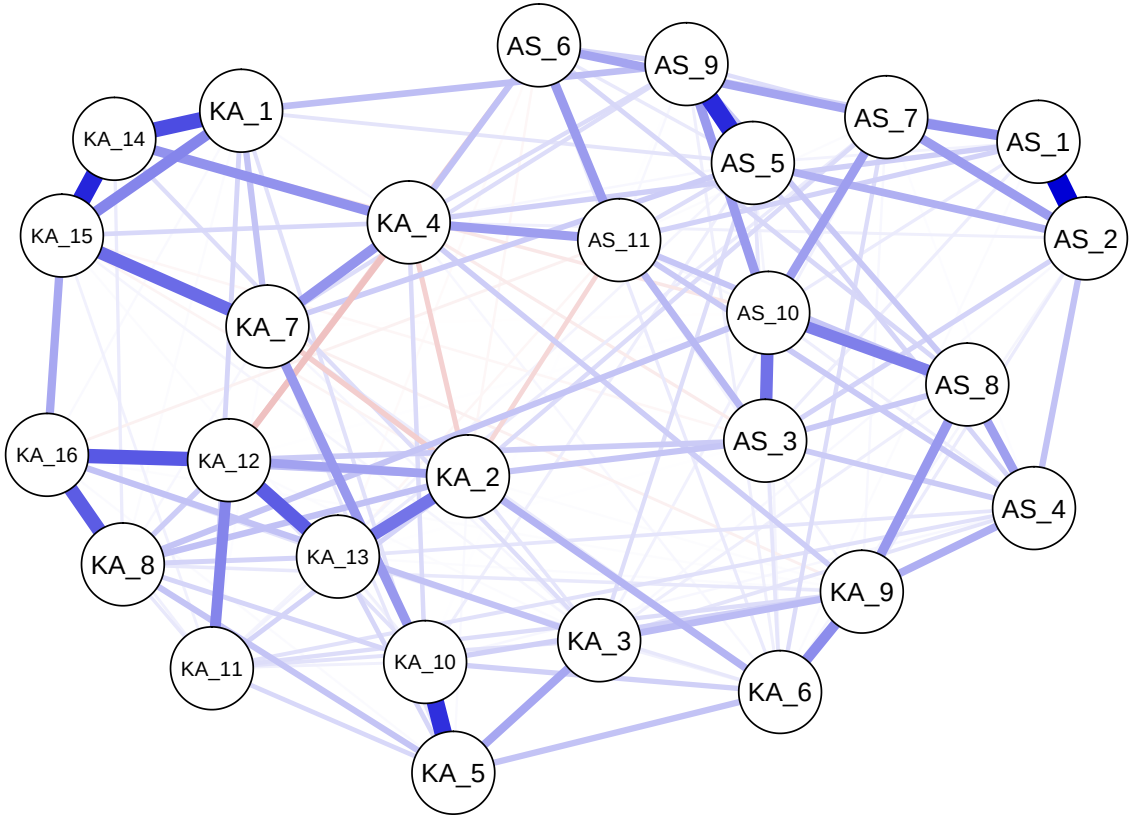
Estimating the Gaussian Graphical Model

```
#estimate regularized partial correlation network
#EBICglasso (gLASSO with EBIC model selection)
#correlations determined via cor_auto
#pairwise deletion of missing values
GGM_net <- estimateNetwork(data, default = "EBICglasso", corMethod = "cor_auto",
                           missing = "pairwise")
```

```
#> Warning in EBICglassoCore(S = S, n = n, gamma = gamma, penalize.diagonal
#> = penalize.diagonal, : A dense regularized network was selected (lambda <
#> 0.1 * lambda.max). Recent work indicates a possible drop in specificity.
#> Interpret the presence of the smallest edges with care. Setting threshold =
#> TRUE will enforce higher specificity, at the cost of sensitivity.
```

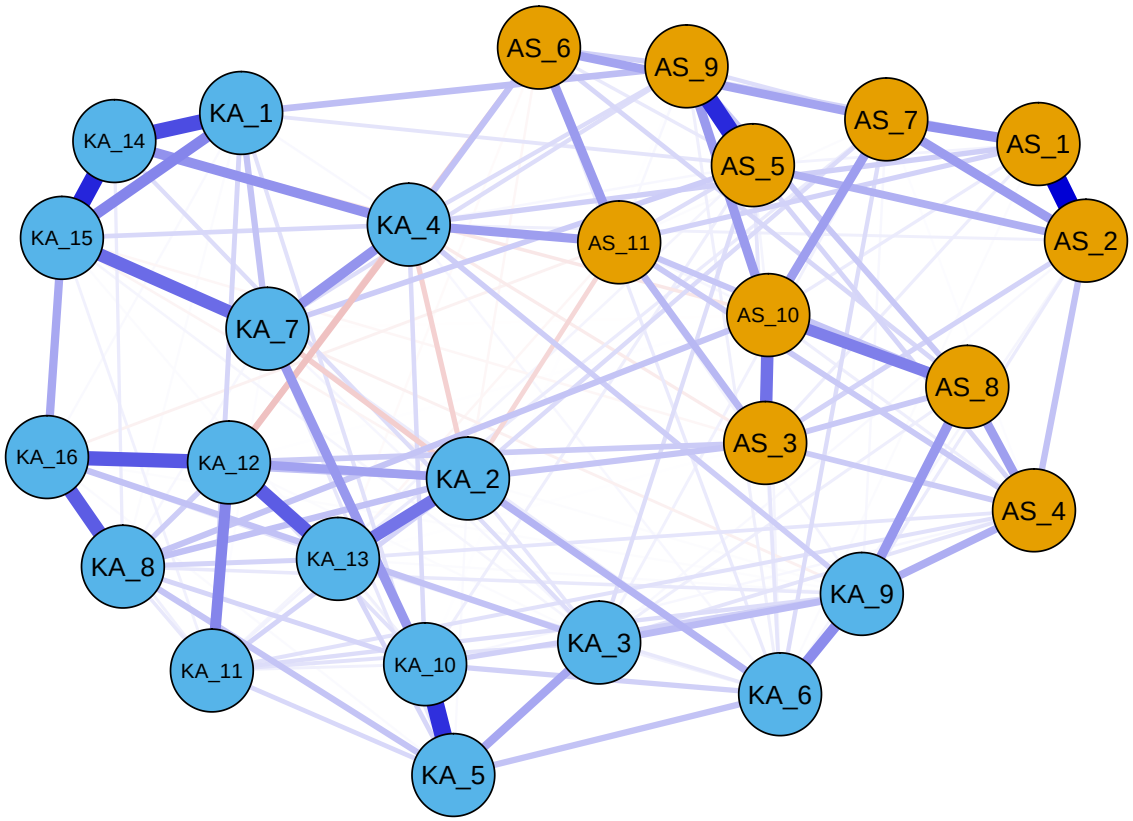
Plotting the Gaussian Graphical Model

```
#plot network with qgraph
GGM_graph <- plot(GGM_net, layout = "spring", theme = "colorblind")
```



Plotting the Gaussian Graphical Model

```
#declare awe and kama muta items separate groups
groups <- c(rep("Awe",11), rep("Kama Muta",16))

#plot network with groups; no legend
GGM_graph <- plot(GGM_net, layout = "spring", theme = "colorblind",
                  groups = groups, legend = FALSE)
```



Information about Gaussian Graphical Model

```
#save layout of the graph
layout <- GGM_graph$layout

#get weights matrix
Wmat_GGM <- getWmat(GGM_graph)
Wmat_GGM
```

Node Centrality

- (potentially) information about how important a node is

Node Centrality

- (potentially) information about how important a node is
  - *strength* (*degree*): sum of all edge weights

Node Centrality

- (potentially) information about how important a node is
  - *strength* (*degree*): sum of all edge weights
  - *closeness*: inverse of average shortest path length

Node Centrality

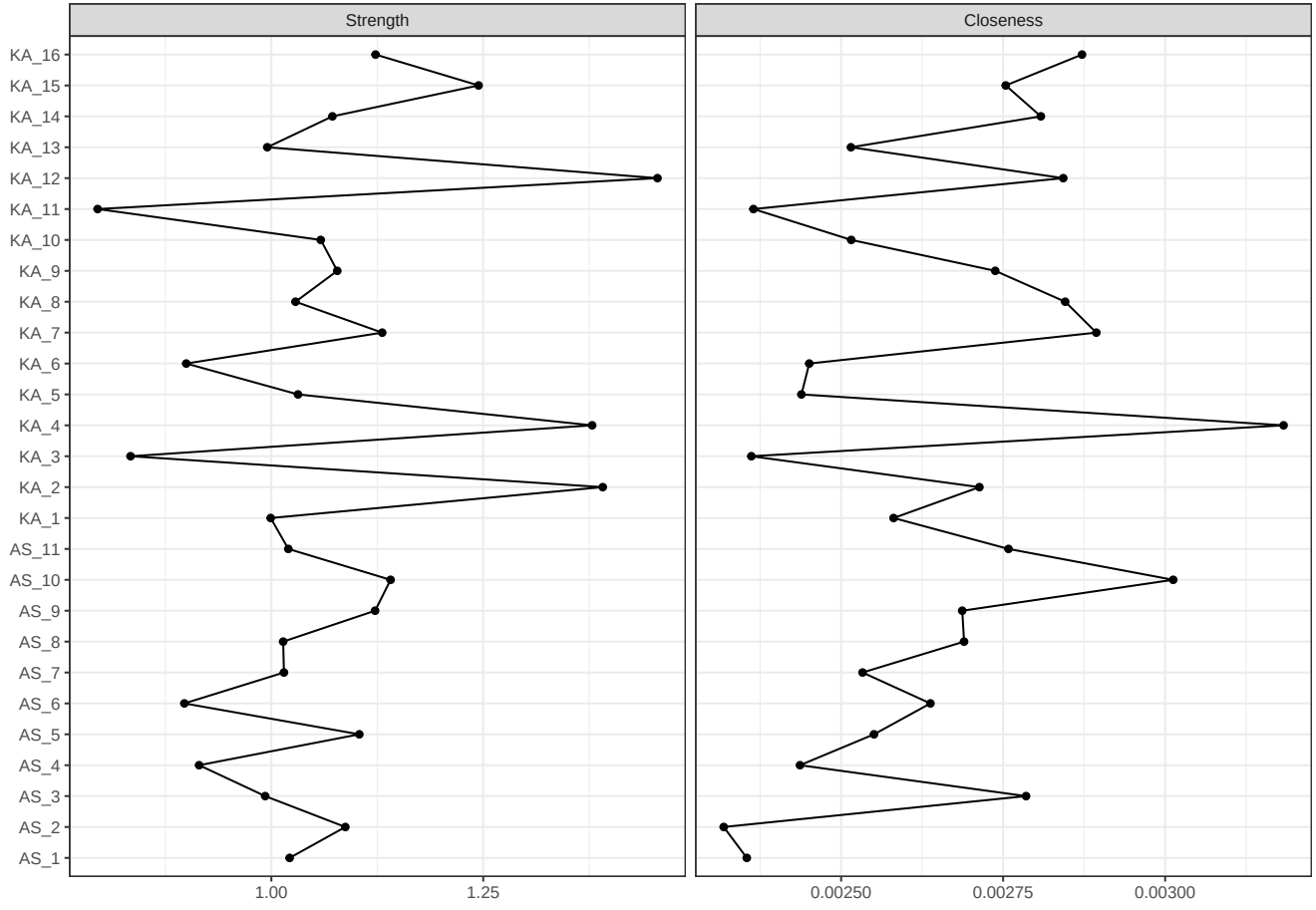
- (potentially) information about how important a node is
  - *strength* (*degree*): sum of all edge weights
  - *closeness*: inverse of average shortest path length
  - ...

Node Centrality

```
#determine values
cent <- centrality(GGM_net)

#get values
cent$OutDegree
cent$Closeness
```

Node Centrality

```
#plot values
centralityPlot(GGM_net, scale = "raw",
               include = c("Strength","Closeness"))
```



Network Stability and Difference Tests

- edges in regularized networks are significant

Network Stability and Difference Tests

- edges in regularized networks are significant
- nevertheless, construct CIs around edge weights

Network Stability and Difference Tests

- edges in regularized networks are significant
- nevertheless, construct CIs around edge weights
- test whether two edges differ

Network Stability and Difference Tests

- edges in regularized networks are significant
- nevertheless, construct CIs around edge weights
- test whether two edges differ
- test stability of centrality indices

Network Stability and Difference Tests

- edges in regularized networks are significant
- nevertheless, construct CIs around edge weights
- test whether two edges differ
- test stability of centrality indices
- compare centrality indices

Network Stability and Difference Tests

- edges in regularized networks are significant
- nevertheless, construct CIs around edge weights
- test whether two edges differ
- test stability of centrality indices
- compare centrality indices

- bootstrap methods

Network Stability and Difference Tests

- edges in regularized networks are significant
- nevertheless, construct CIs around edge weights
- test whether two edges differ
- test stability of centrality indices
- compare centrality indices

- bootstrap methods
  - non-parametric bootstrap: re-sampling from data with replacement

Network Stability and Difference Tests

- edges in regularized networks are significant
- nevertheless, construct CIs around edge weights
- test whether two edges differ
- test stability of centrality indices
- compare centrality indices

- bootstrap methods
  - non-parametric bootstrap: re-sampling from data with replacement
    - for edges and edges/centrality difference tests

Network Stability and Difference Tests

- edges in regularized networks are significant
- nevertheless, construct CIs around edge weights
- test whether two edges differ
- test stability of centrality indices
- compare centrality indices

- bootstrap methods
  - non-parametric bootstrap: re-sampling from data with replacement
    - for edges and edges/centrality difference tests
  - case-dropping bootstrap: successively drop cases; check parameter consistency

Network Stability and Difference Tests

- edges in regularized networks are significant
- nevertheless, construct CIs around edge weights
- test whether two edges differ
- test stability of centrality indices
- compare centrality indices

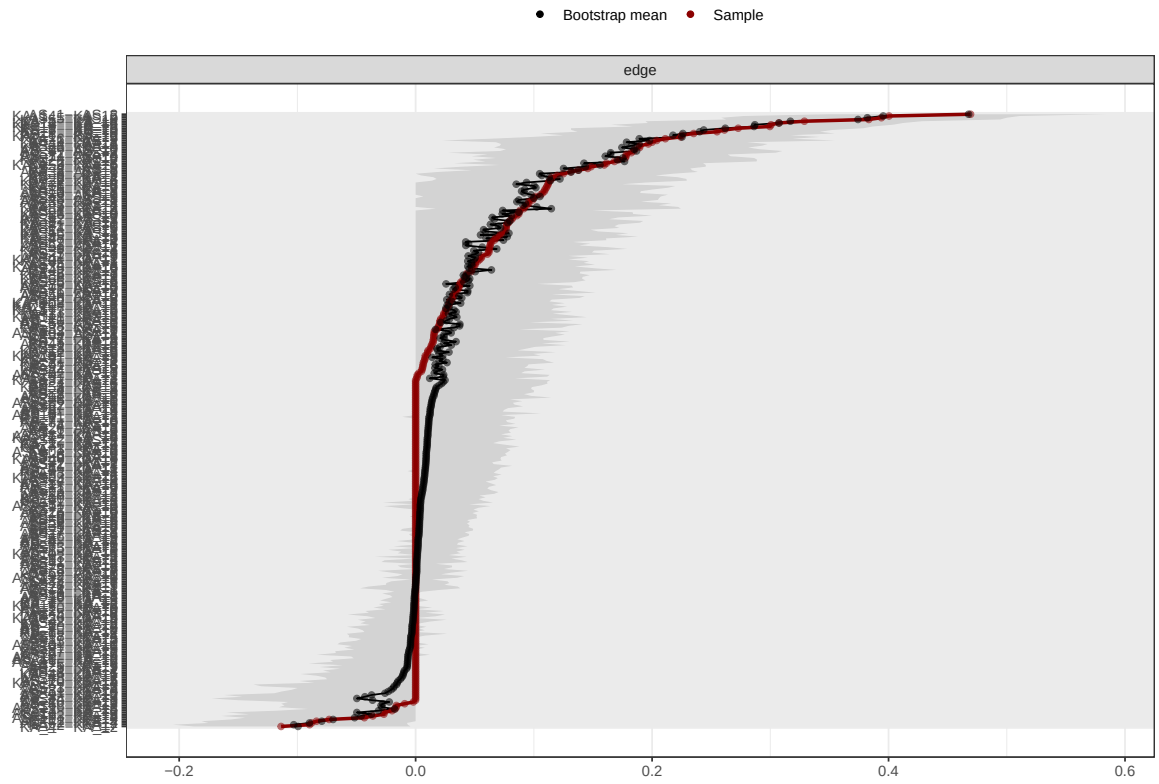
- bootstrap methods
  - non-parametric bootstrap: re-sampling from data with replacement
    - for edges and edges/centrality difference tests
  - case-dropping bootstrap: successively drop cases; check parameter consistency
    - for centrality indices

Network Stability and Difference Tests

```r
#non-parametric bootstrap
#-> for edge stability and edge as well as centrality difference tests
#run bootstrap
set.seed(4186)
boot1 <- bootnet(GGM_net, statistics = c("edge","Strength","Closeness"),
                 nboots = 1000, nCores = 2, type = "nonparametric")
save(boot1, file = "boot_edges.RData") #save results

#case-dropping bootstrap
#-> for centrality stability
#run bootstrap
set.seed(4186)
boot2 <- bootnet(GGM_net, statistics = c("Strength","Closeness"),
                 nboots = 1000, nCores = 2, type = "case")
save(boot2, file = "boot_centrality.RData") #save results
```

Network Stability and Difference Tests

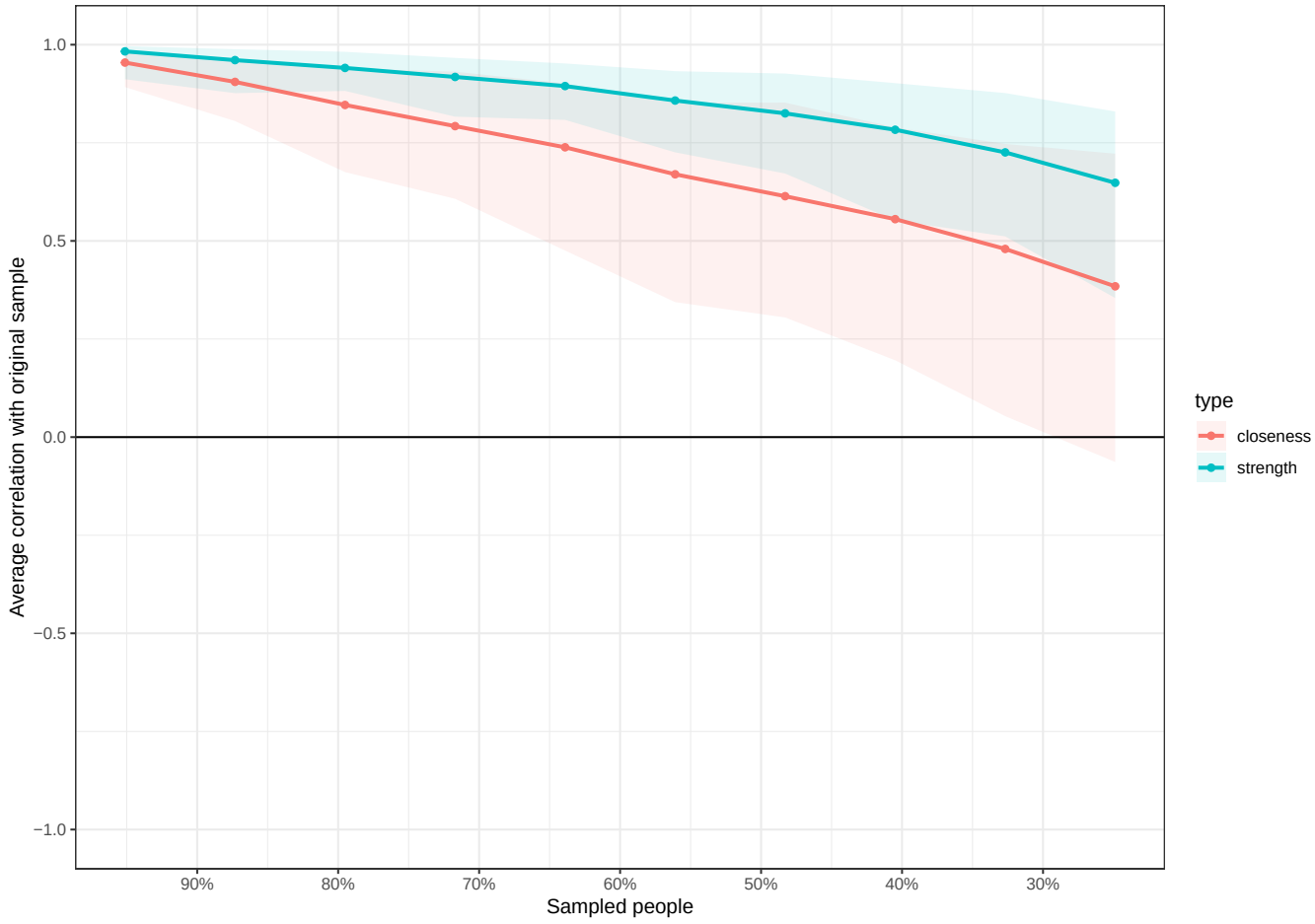
```
load("boot_edges.RData")
load("boot_centrality.RData")

#plot edge CIs
plot(boot1, statistics = "edge", labels = TRUE, order = "sample")
```




```
#plot edge CIs + make labels visible by creating large pdf
# pdf("edge_stability.pdf", height = 50)
# plot(boot1, statistics = "edge", labels = TRUE, order = "sample")
# dev.off()
```

Network Stability and Difference Tests

```
#plot centrality stability
plot(boot2, statistics = c("Strength","Closeness"))
```



Network Stability and Difference Tests

```
#check stability of centrality indices
corStability(boot2)
```

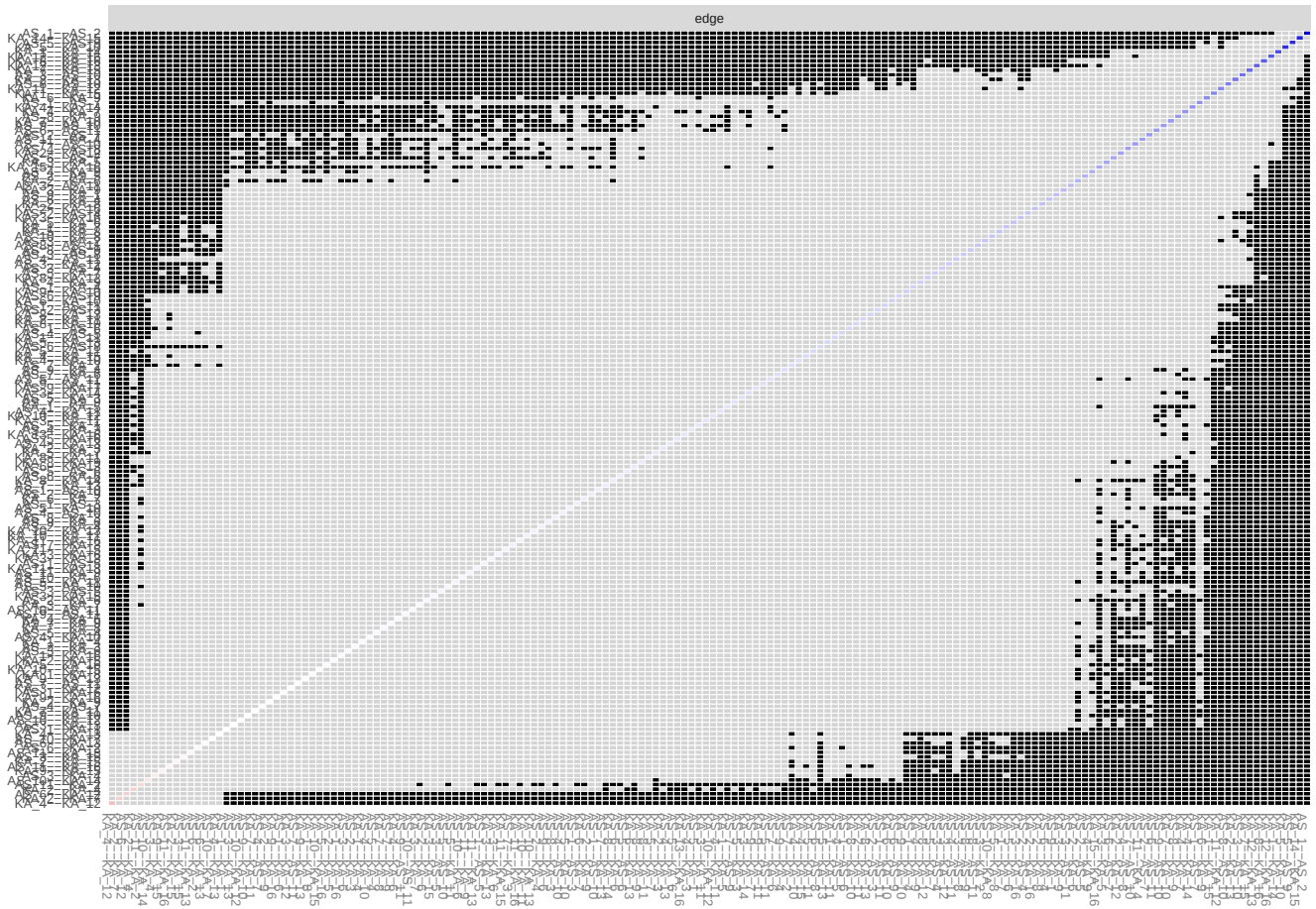
```
#> === Correlation Stability Analysis ===
#>
#> Sampling levels tested:
#>    nPerson Drop%   n
#> 1      102  75.1  93
#> 2      134  67.3 114
#> 3      166  59.5 112
#> 4      198  51.7 100
#> 5      230  43.9 105
#> 6      262  36.1  92
#> 7      294  28.3  98
#> 8      326  20.5  92
#> 9      358  12.7 104
#> 10     390   4.9  90
#>
#> Maximum drop proportions to retain correlation of 0.7 in at least 95% of the samples:
#>
#> closeness: 0.205
#>   - For more accuracy, run bootnet(..., caseMin = 0.127, caseMax = 0.283)
#>
#> strength: 0.439
#>   - For more accuracy, run bootnet(..., caseMin = 0.361, caseMax = 0.517)
#>
#> Accuracy can also be increased by increasing both 'nBoots' and 'caseN'.
```

```
#-> strength can be interpreted with care
```

Network Stability and Difference Tests

```
#plot edge weights difference test
plot(boot1, statistics = "edge", plot = "difference",
     onlyNonZero = TRUE, order = "sample")
```

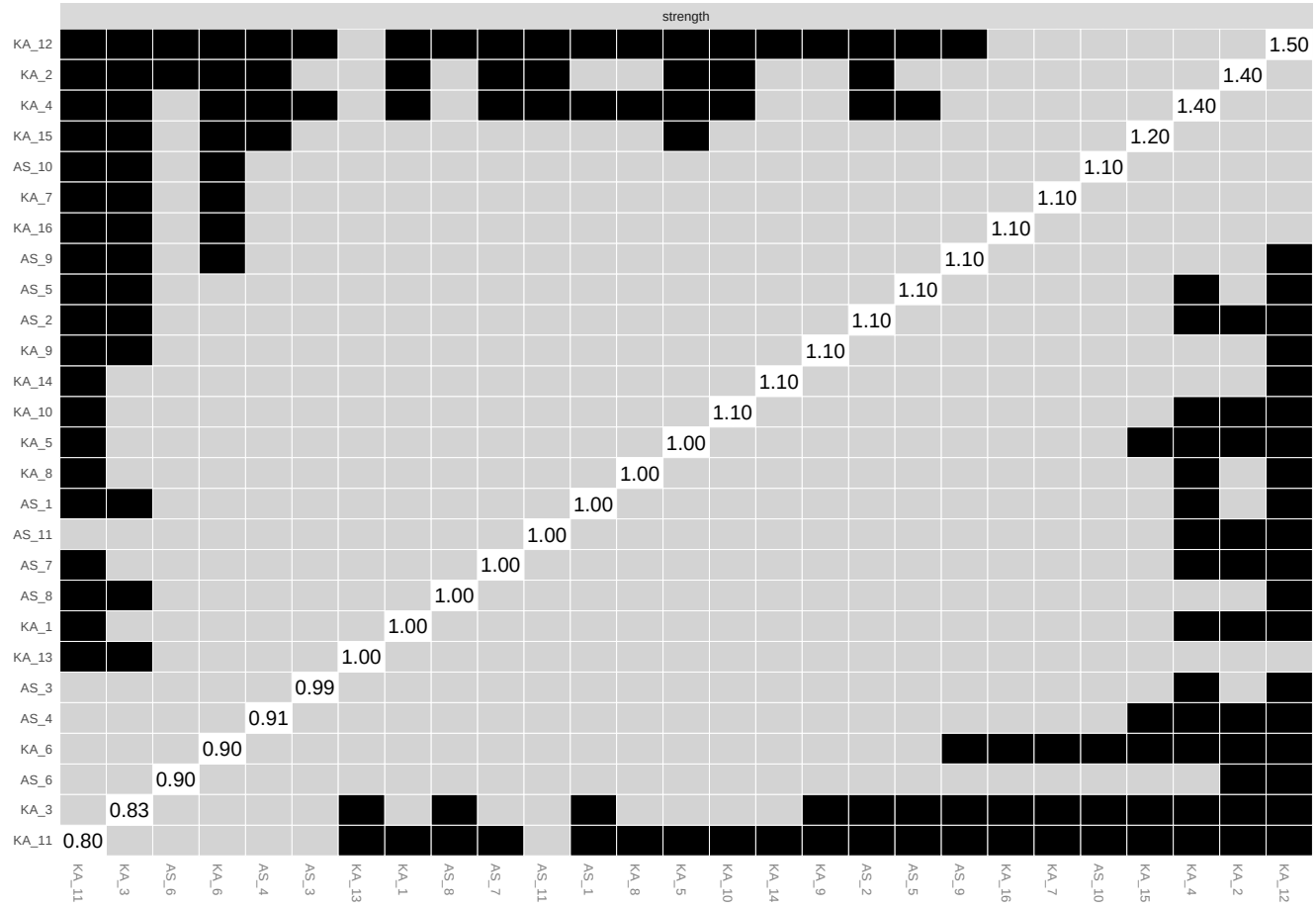
```
#> Expected significance level given number of bootstrap samples is approximately: 0.05
```



Network Stability and Difference Tests

```
#plot strength difference test
plot(boot1, statistics = "strength", plot = "difference",
     order = "sample")
```

```
#> Expected significance level given number of bootstrap samples is approximately: 0.05
```



Clique Percolation - Thresholds

```
#run threshold function to determine k and I for a range of k and I values
threshold <- cpThreshold(W = GGM_graph, method = "weighted",
                         k.range = c(3,4,5),
                         I.range = c(seq(0.25, 0.01, by = -0.001)),
                         threshold = c("largest.components.ratio","chi","entropy"))
```

Clique Percolation - Thresholds

```
#use permutation function to derive solutions more surprising than chance
set.seed(4186)
threshold_permute <- cpPermuteEntropy(W = GGM_graph, cpThreshold.object = threshold,
                                      n = 100, interval = 0.95)
save(threshold_permute, file = "threshold_permutation_mixed.Rdata")
```

Clique Percolation - Thresholds

```
load("threshold_permutation_mixed.Rdata") #loads previous permutation

#inspect results
threshold_permute$Confidence.Interval
threshold_permute$Extracted.Rows
```

Clique Percolation - Algorithm

```
#highest threshold for fewest isolated nodes obtained with k = 3 and I = .129
cp_ka_k3I.129 <- cpAlgorithm(W = GGM_graph, k = 3,
                             method = "weighted", I = 0.129)
```

Clique Percolation - Algorithm

```
#inspect communities
cp_ka_k3I.129$list.of.communities.labels
```

```
#> [[1]]
#> [1] "KA_1"  "KA_4"  "KA_5"  "KA_6"  "KA_7"  "KA_8"  "KA_10" "KA_14" "KA_15"
#>
#> [[2]]
#> [1] "KA_2"  "KA_8"  "KA_11" "KA_12" "KA_13" "KA_16"
#>
#> [[3]]
#> [1] "AS_3"  "AS_8"  "AS_9"  "AS_10"
#>
#> [[4]]
#> [1] "AS_6"  "AS_11" "KA_4"
#>
#> [[5]]
#> [1] "AS_4" "AS_8" "KA_9"
#>
#> [[6]]
#> [1] "AS_5" "AS_9" "KA_1" "KA_7"
#>
#> [[7]]
#> [1] "AS_1" "AS_2" "AS_6" "AS_7"
```

Clique Percolation - Algorithm

```
#inspect shared nodes
cp_ka_k3I.129$shared.nodes.labels
```

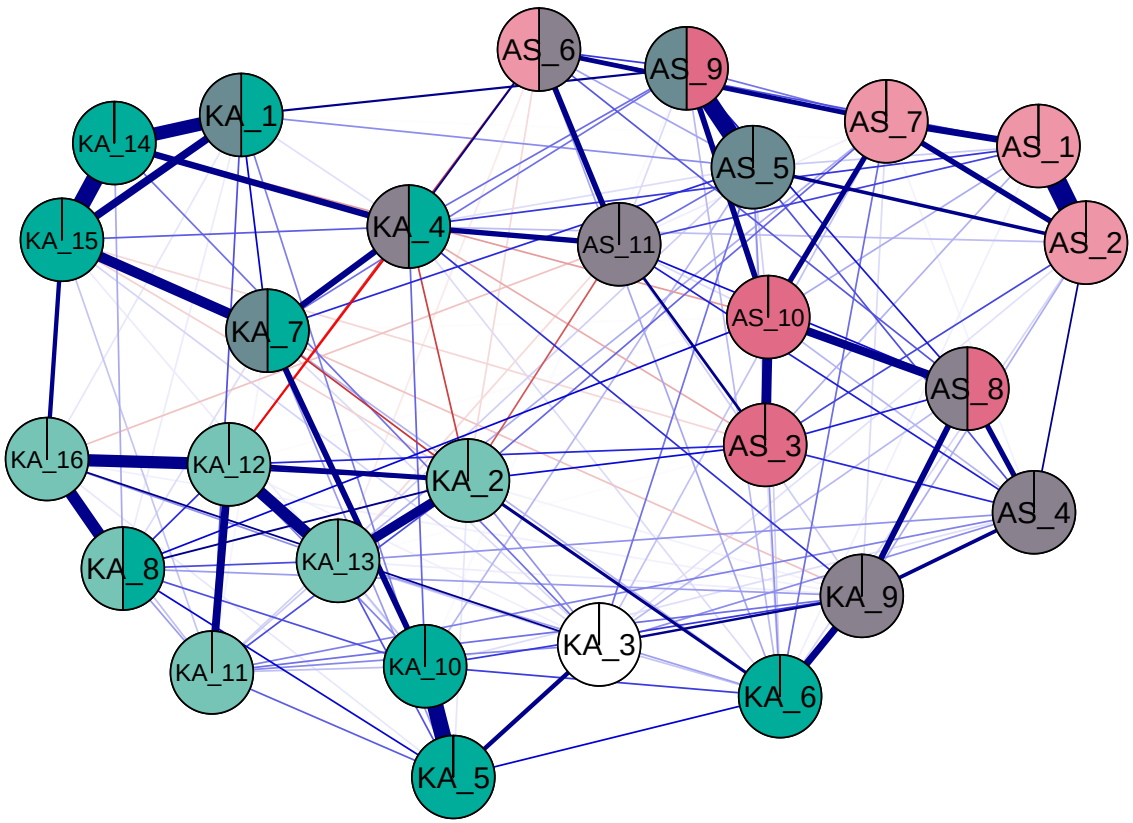
```
#> [1] "AS_6" "AS_8" "AS_9" "KA_1" "KA_4" "KA_7" "KA_8"
```

```
#inspect isolated nodes
cp_ka_k3I.129$isolated.nodes.labels
```

```
#> [1] "KA_3"
```

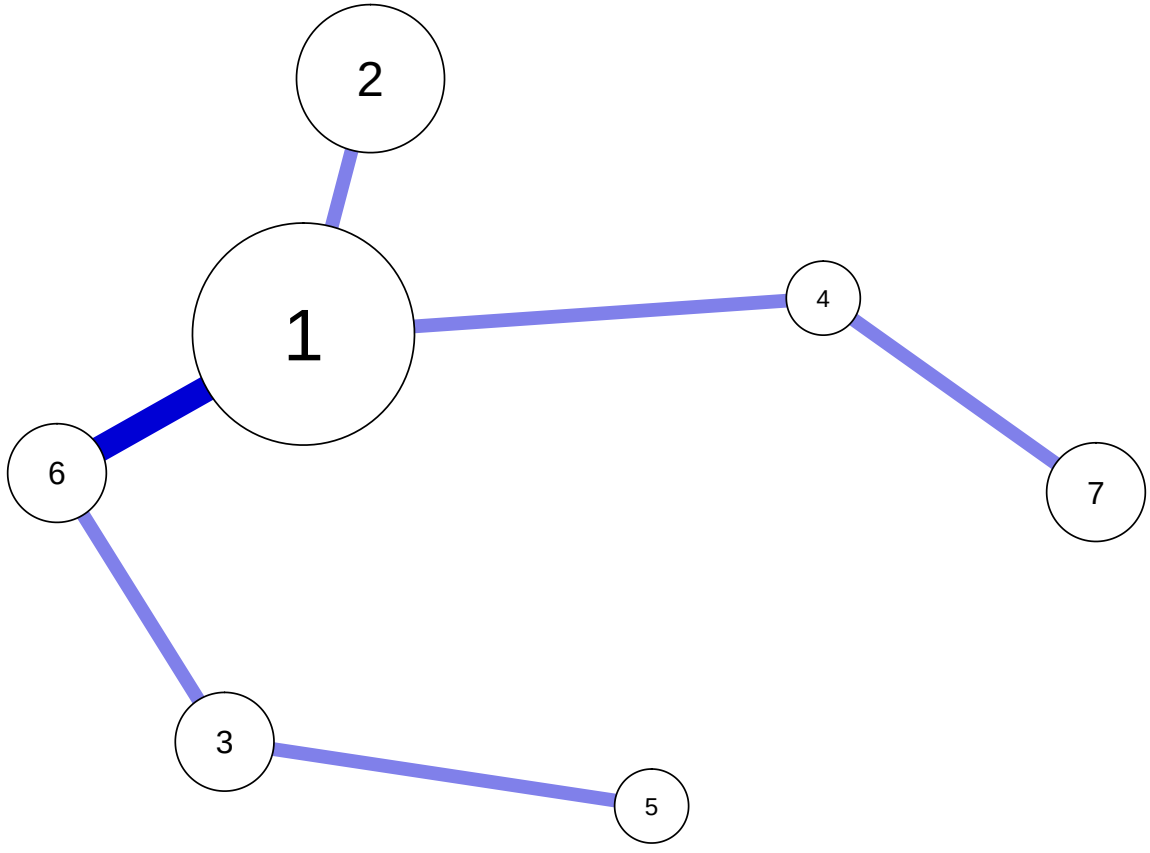
Clique Percolation - Colored Graph

```
#define which items belong to which construct, first construct awe, second kama muta.
list.of.sets <- list(seq(from = 1, to = 11),
                     seq(from = 12, to = 27))
#plot colored graph
col_graph <- cpColoredGraph(GGM_graph, cp_ka_k3I.129$list.of.communities.numbers, layout=layout,
                            list.of.sets = list.of.sets, set.palettes.size = 6,
                            theme = "colorblind")
```



Clique Percolation - Community Graph

```
#community graph
comm_graph <- cpCommunityGraph(cp_ka_k3I.129$list.of.communities.numbers,
                               node.size.method = "proportional",
                               max.node.size = 18, layout = "spring", theme = "colorblind")
```



Estimating the Ising Model

```
#can be estimated via estimateNetwork
#variables automatically binarized at median
#estimate regularized logistic node-wise regression network
#define where to binarize variables
#eLASSO (LASSO with EBIC model selection)
#listwise deletion of missing values (pairwise not possible for regressions)
Ising_net <- estimateNetwork(data, default = "IsingFit",
                             missing = "listwise", rule = "OR")
```

```
#> Estimating Network. Using package::function:
#>   - IsingFit::IsingFit for network computation
#>     - Using glmnet::glmnet
```

```
#> Warning in bootnet::binarize(data, split = split, verbose = verbose):
#> Splitting data by median
```

Information about the Ising Model

```r
#extract thresholds
thresholds <- Ising_net$intercepts

#extract weights matrix
Wmat_Ising <- getWmat(Ising_net)
Wmat_Ising
```

Comparing the Gaussian Graphical Model and Ising Model

```r
#correlating weights matrices
cor.test(Wmat_GGM[upper.tri(Wmat_GGM)], Wmat_Ising[upper.tri(Wmat_Ising)])
```

```
#>
#>  Pearson's product-moment correlation
#>
#> data:  Wmat_GGM[upper.tri(Wmat_GGM)] and Wmat_Ising[upper.tri(Wmat_Ising)]
#> t = 24.648, df = 349, p-value < 2.2e-16
#> alternative hypothesis: true correlation is not equal to 0
#> 95 percent confidence interval:
#>  0.7552786 0.8322035
#> sample estimates:
#>       cor
#> 0.7969498
```

Other developments in network analysis in R

See also...

- categorical variables (*mgm*; Haslbeck & Waldorp, in press, JSS)
- time-series data (idosyncratic and group-level; Epskamp et al., 2018, MBR)
- latent variables in networks (Epskamp et al., 2017, Psychometrika)
- moderated network models (Haslbeck et al., subm)
- network comparison (*NCT*; Van Borkulo et al., subm.)
- interpretable plots (Jones et al., 2018, Frontiers)
- ...

Summary for Network Analysis

Summary

- psychological networks can be estimated via pairwise Markov Random Field Models
  - continuous data: Gaussian Graphical Model
  - binary data: Ising Model
- estimation via regularized partial correlations and/or regularized node-wise regression
- central nodes can be structurally important
- stability of network parameters should be checked
- *bootnet* and *qgraph* can be used for estimating and plotting networks in R
- community detection helps identifying (overlapping) strongly connected subgraphs

Thank you very much for your attention!