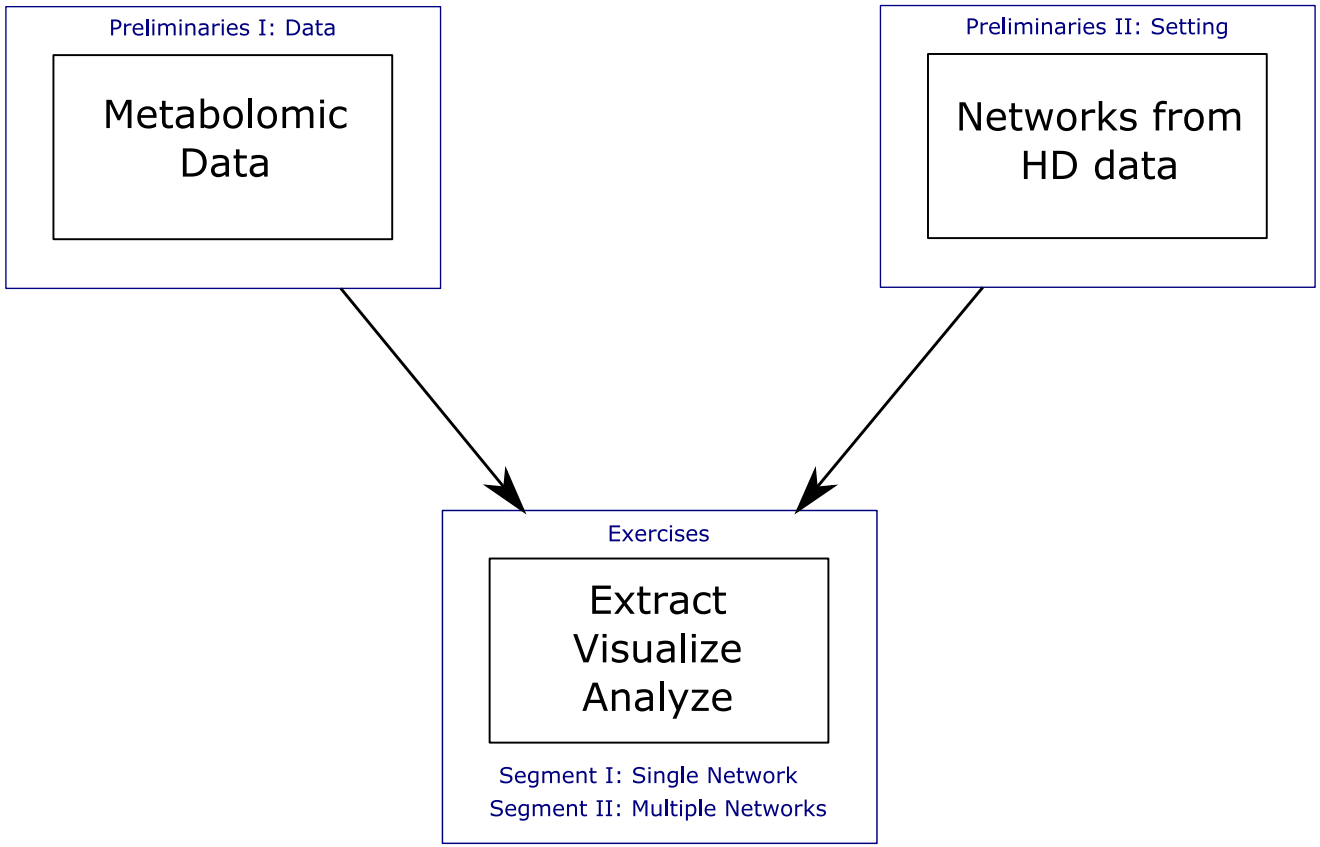
# rags2ridges
# Network Modeling of High-Dimensional Data

Carel F.W. Peeters
Dept. of Epidemiology & Biostatistics
VU University medical center, Amsterdam
cf.peeters@vumc.nl

IBS Channel Network Conference 2017 short-course
Hasselt, Belgium
April 24, 2017

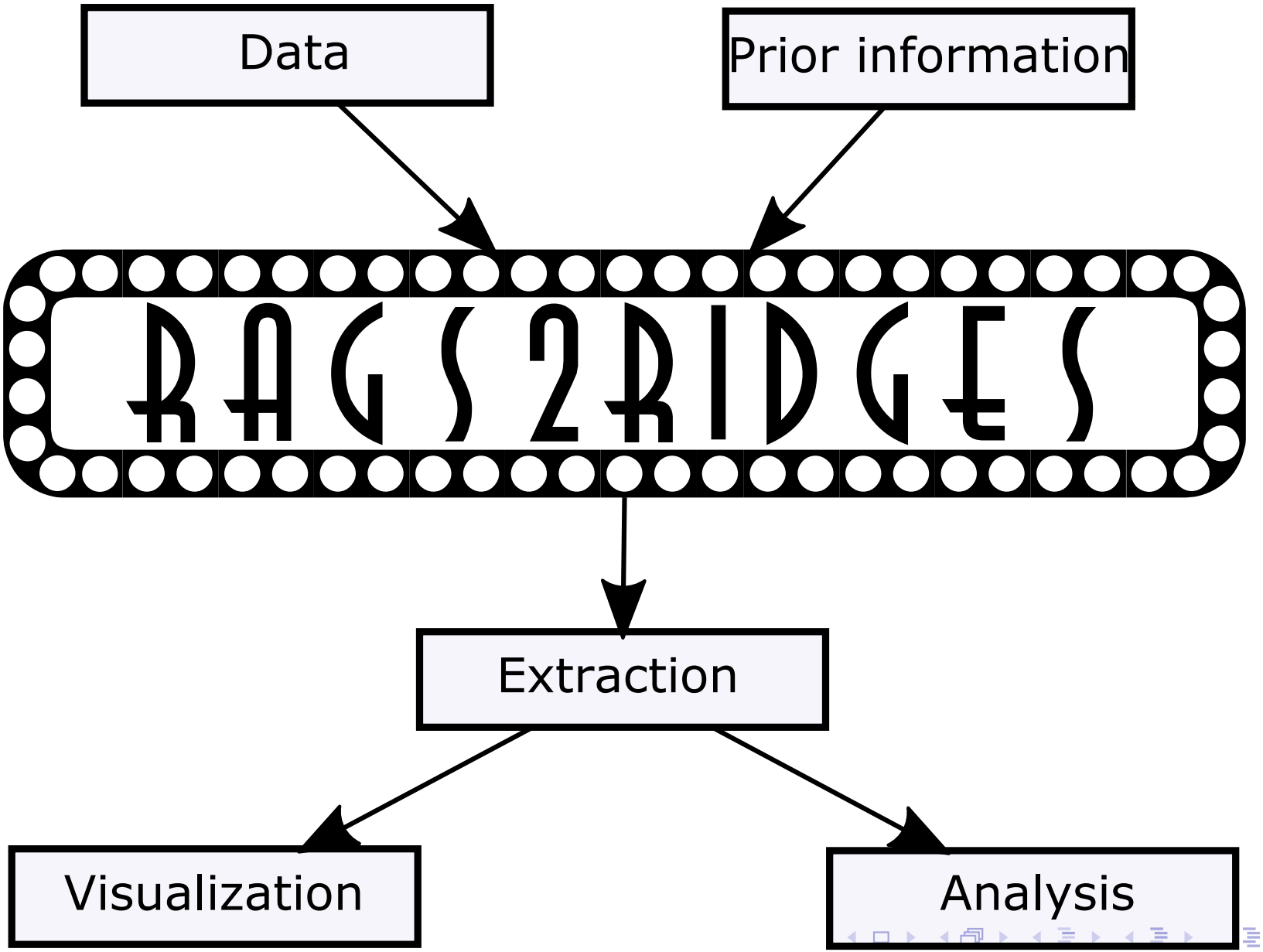
# Overview




## Materials

- These slides: Contain explanation theory/methodology and exercises
- `rags2ridges_Practical.HTML`: Contains solutions to exercises with additional explanations
- `NetworksExercises.R`: Contains bare solutions to exercises

# rags2ridges: One-Stop-Go



# Omics and omics data

**-ome**

A totality of some (molecular biological) sort

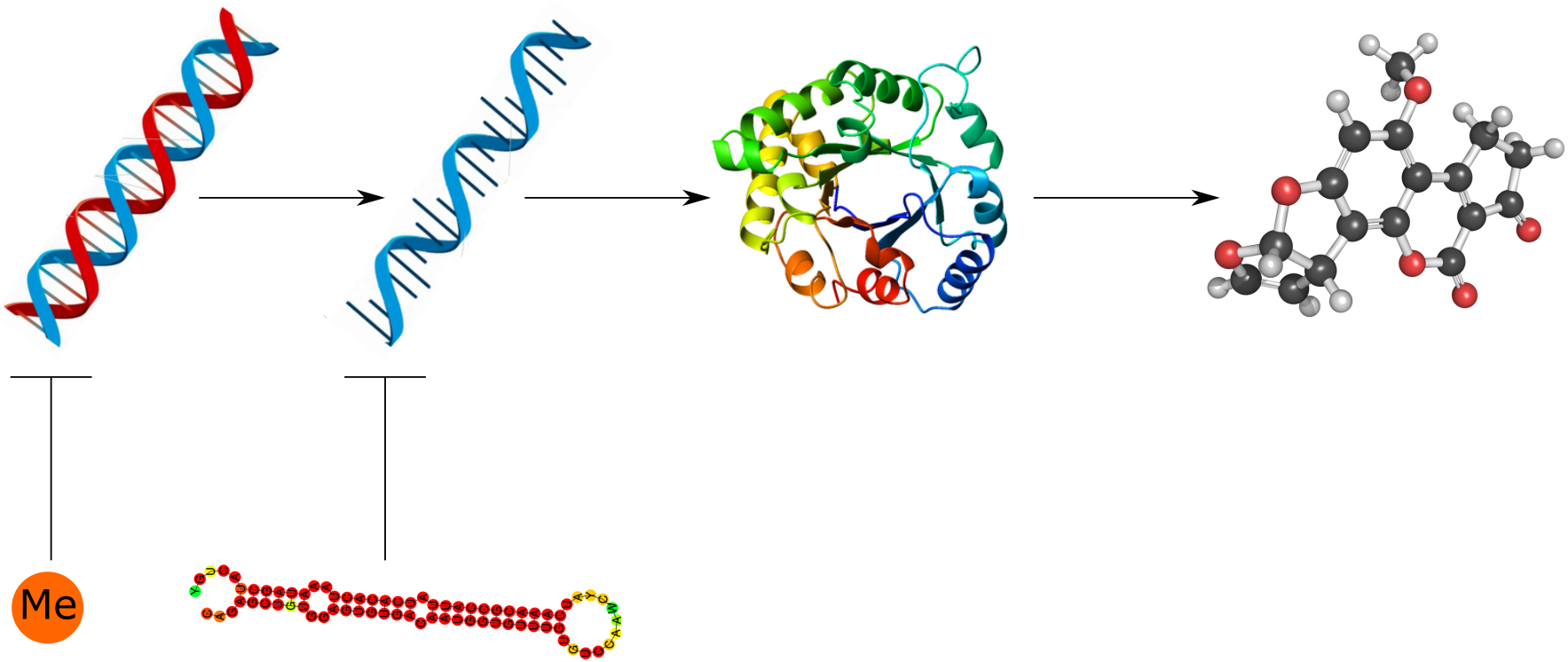
**-omics**

Collective quantification of some pool of molecular molecules

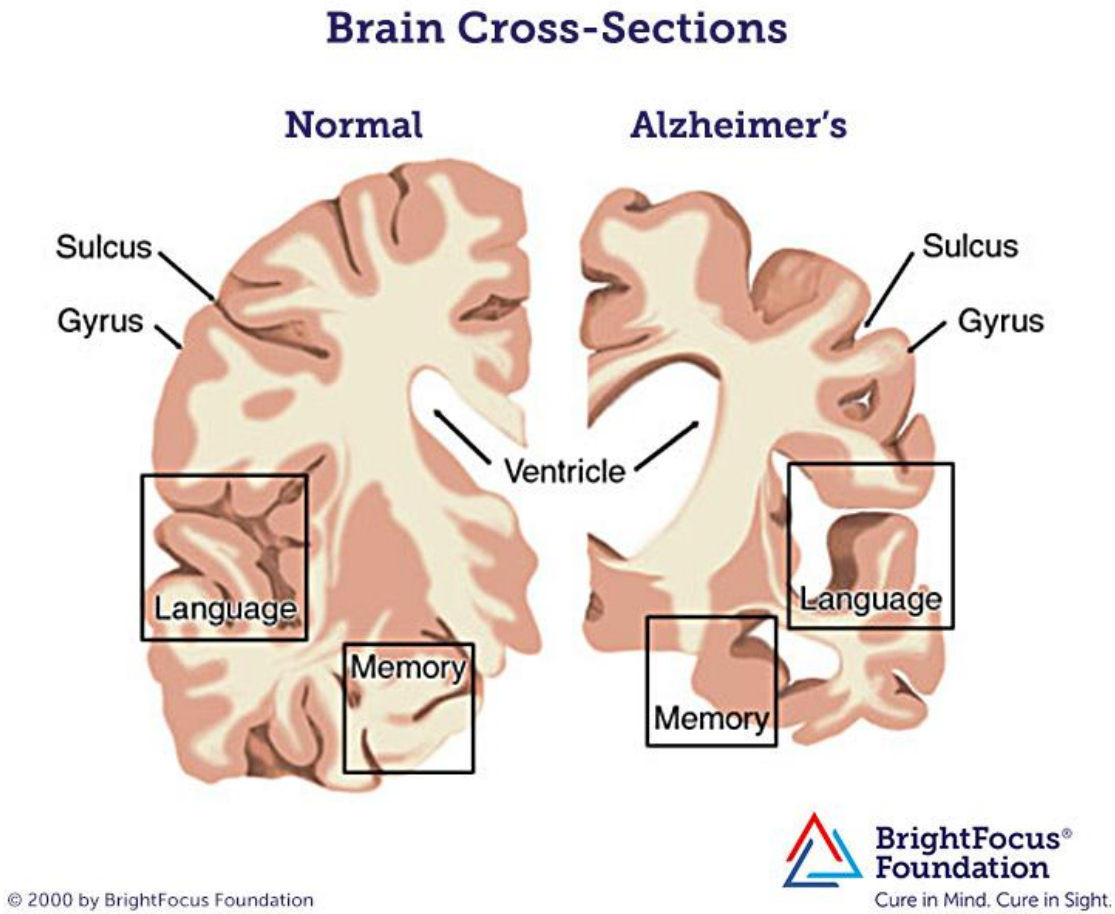
**Genomics**

The omics of the genome (of some organism)

# The omic cascade



# Alzheimer's Disease



# Metabolite quantification

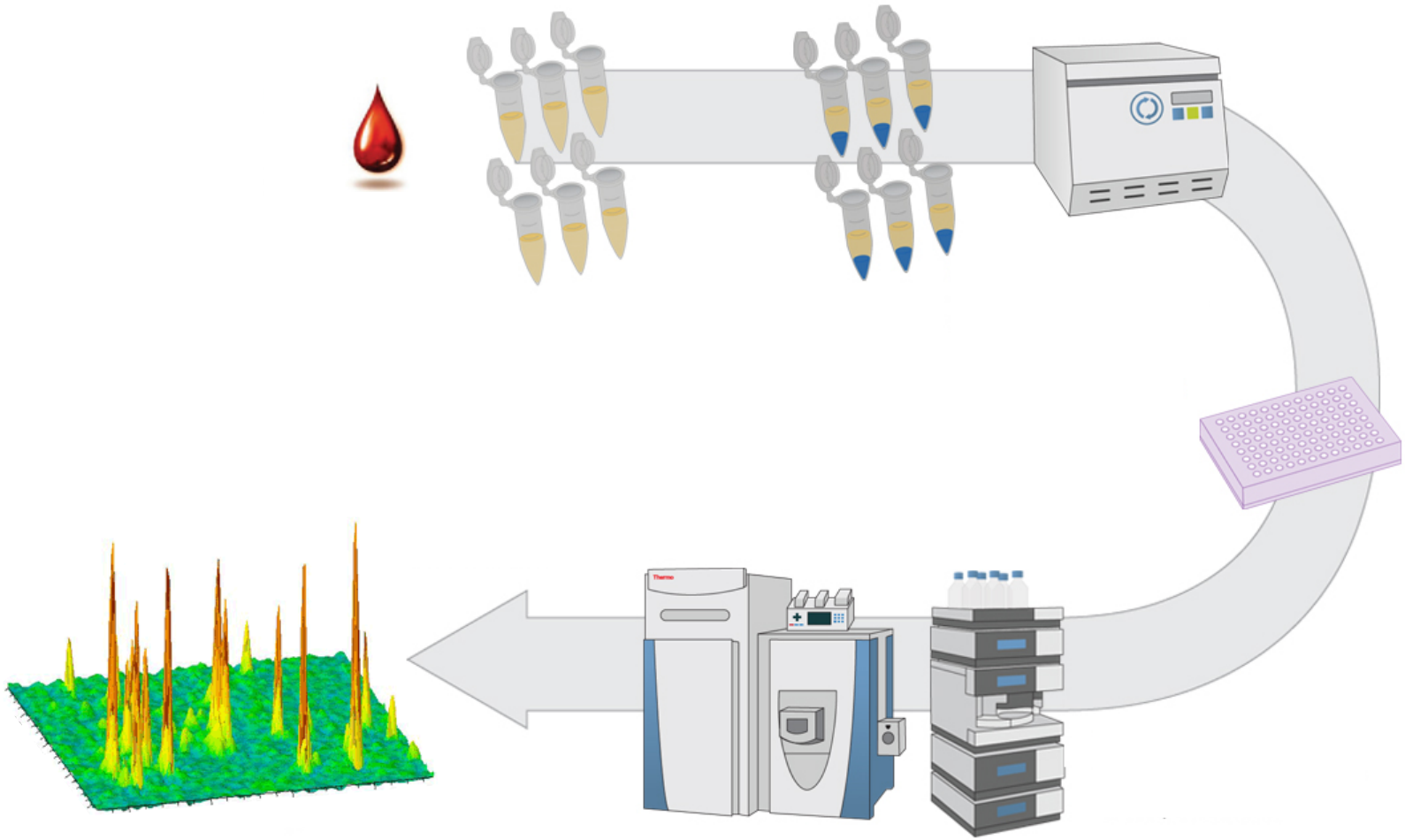


Illustration adapted from: http://planetorbitrap.com/untargeted-metabolomics#.Vzw6yfmLRaQ &

http://metabolomicsplatform.com/metabolomics-overview/

# Metabolomic Data

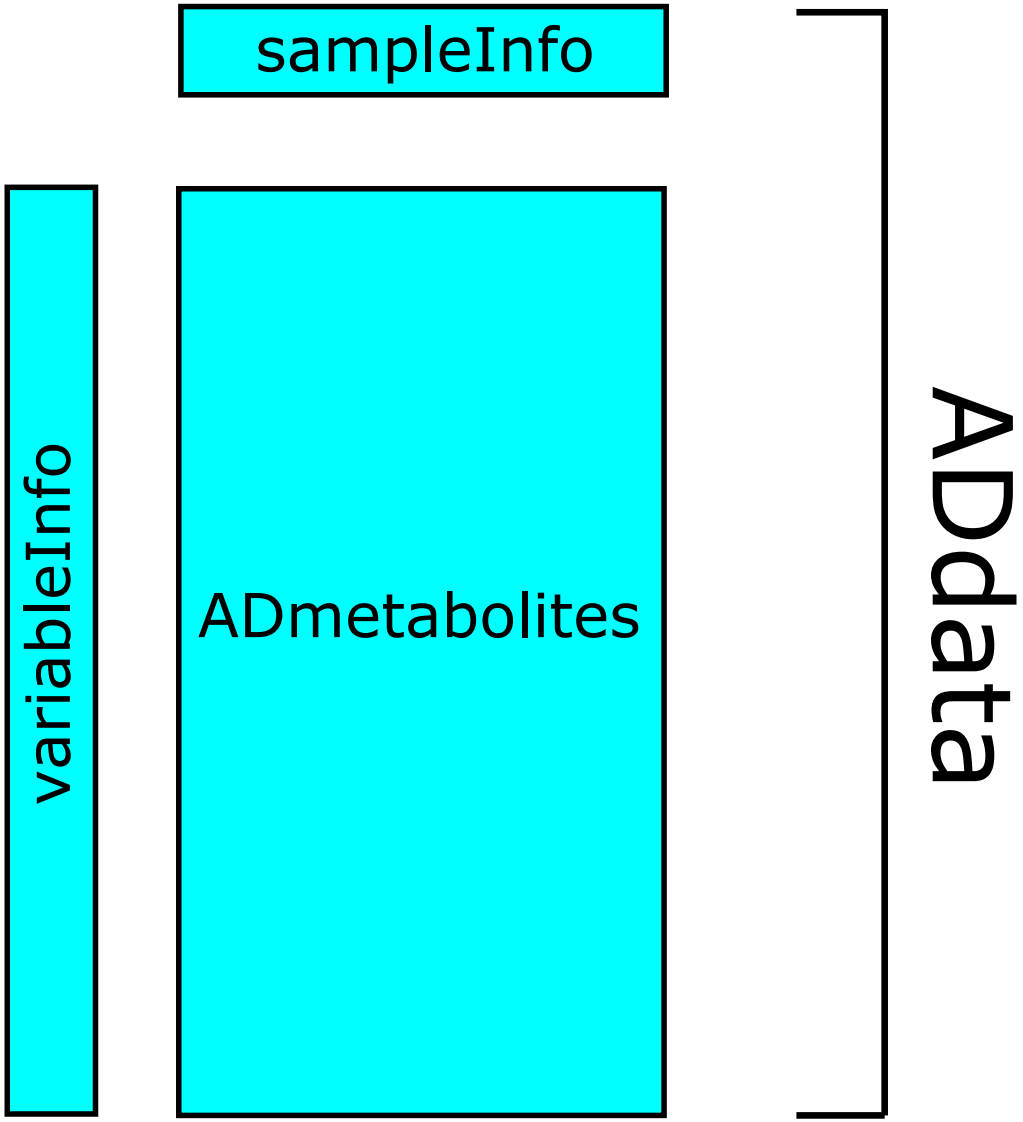
Variables (features)

| Observations | 1 | 2 | 3 | 4 | 5 | ... | $p$ |
|---|---|---|---|---|---|---|---|
| 1 | $y_{11}$ | $y_{12}$ | $y_{13}$ | $y_{14}$ | $y_{15}$ | ... | $y_{1p}$ |
| 2 | $y_{21}$ | $y_{22}$ | $y_{23}$ | $y_{24}$ | $y_{25}$ | ... | $y_{2p}$ |
| 3 | $y_{31}$ | $y_{32}$ | $y_{33}$ | $y_{34}$ | $y_{35}$ | ... | $y_{3p}$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ |
| $n$ | $y_{n1}$ | $y_{n2}$ | $y_{n3}$ | $y_{n4}$ | $y_{n5}$ | ... | $y_{np}$ |

- Amines
- Lipids
- Organic Acids
- Oxidative Stress

$p = 230$: 53 Amines, 116 Lipids, 22 Organic acids, 39 Oxidative stress compounds
$n = 127$: 40 AD class 1, 87 AD class 2

# Metabolomic Data Objects



# Exercise 1: Get acquainted with the data

```
## Set working directory
setwd("")

## Needed libraries
library(rags2ridges)
```

## Invoke data

```
load()
```

## Which objects?

```
objects()
```

## Simple exploration objects

```
head()
```

# Gaussian graphical modeling

## Graphical modeling

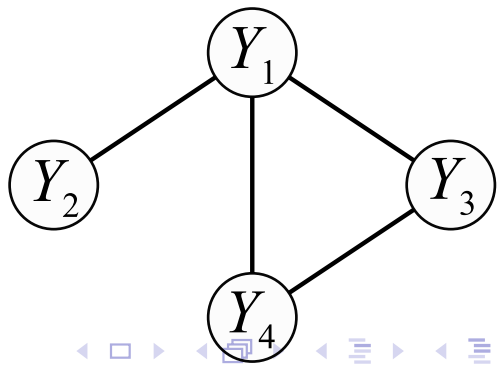
Class of models using graphs to express conditional (in)dependence relations between random variables

## Gaussian setting

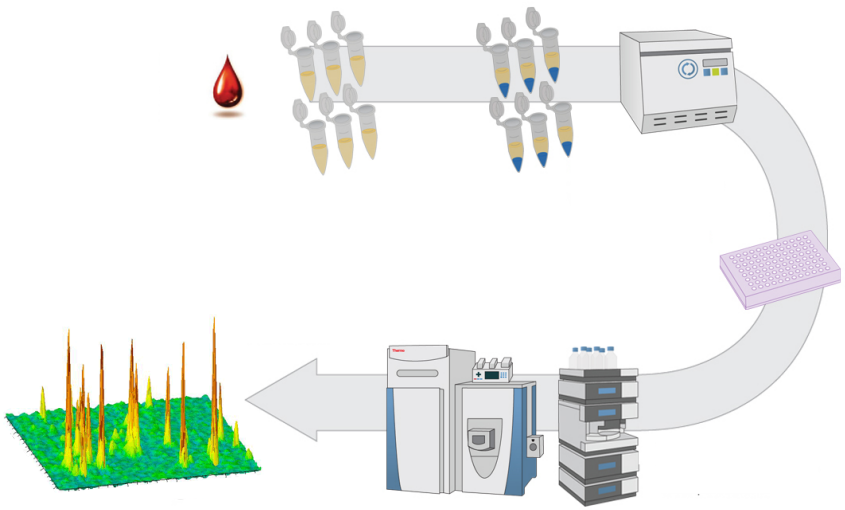
- Vertices: Correspond to random variables with normal distribution
- Edges: Correspond to the dependence structure
- Say $\mathbf{y} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{\Sigma})$, and define $\mathbf{\Sigma}^{-1} \equiv \mathbf{\Omega}$. Then, for $a, b \in$ vertex set $V$, $a \neq b$

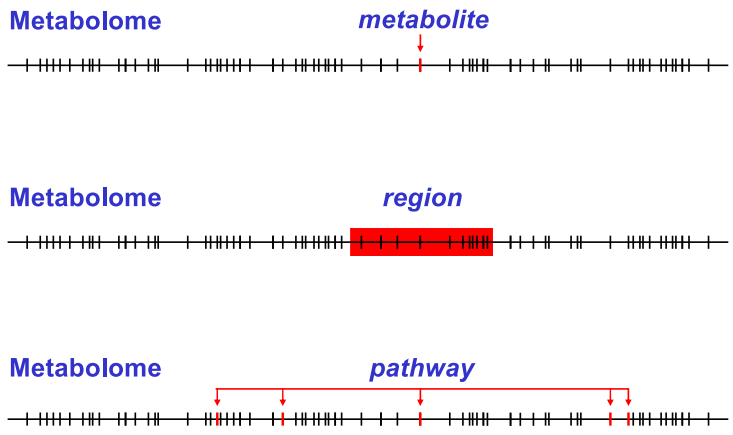
$$-\frac{\omega_{ab}}{\sqrt{\omega_{aa}\omega_{bb}}} = 0 \iff \omega_{ab} = 0 \iff a \perp\!\!\!\perp b | V \setminus \{a, b\} \iff a \not\sim b$$

$$\begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \omega_{14} \\ \omega_{21} & \omega_{22} & 0 & 0 \\ \omega_{31} & 0 & \omega_{33} & \omega_{34} \\ \omega_{41} & 0 & \omega_{43} & \omega_{44} \end{bmatrix}$$



# Challenge






## Problem

- Let $\mathbf{S} \equiv \hat{\mathbf{\Sigma}}$ denote the sample covariance matrix on $\mathbf{y}_i$
- When $p \approx n$ or $p > n$, $\mathbf{S}$ is ill-behaved or singular
- The precision $\mathbf{S}^{-1} \equiv \hat{\mathbf{\Omega}}$ is then undefined

# $\ell_2$-Penalization: Proper ridge

Maximize

$$\underbrace{\ln|\boldsymbol{\Omega}| - \text{tr}(\mathbf{S}\boldsymbol{\Omega})}_{\log-\text{likelihood}} - \underbrace{\frac{\lambda}{2}\|\boldsymbol{\Omega} - \mathbf{T}\|_2^2}_{\ell_2-\text{penalty}}$$

- $\mathbf{T}$ denotes a p.d. symmetric target matrix
- $\lambda \in (0, \infty)$ denotes a penalty parameter

Analytic penalized ML estimator

$$\hat{\boldsymbol{\Omega}}(\lambda) = \left\{ \left[ \lambda \mathbf{I}_p + \frac{1}{4}(\mathbf{S} - \lambda\mathbf{T})^2 \right]^{1/2} + \frac{1}{2}(\mathbf{S} - \lambda\mathbf{T}) \right\}^{-1}$$

# Properties

## Behavior

i. $\hat{\mathbf{\Omega}}(\lambda) \succ 0$, for all $\lambda \in (0, \infty)$;

ii. $\lim_{\lambda \to 0^+} \hat{\mathbf{\Omega}}(\lambda) = \mathbf{S}^{-1}$ if $p < n$;

iii. $\lim_{\lambda \to \infty} \hat{\mathbf{\Omega}}(\lambda) = \mathbf{T}$.

## Consistency

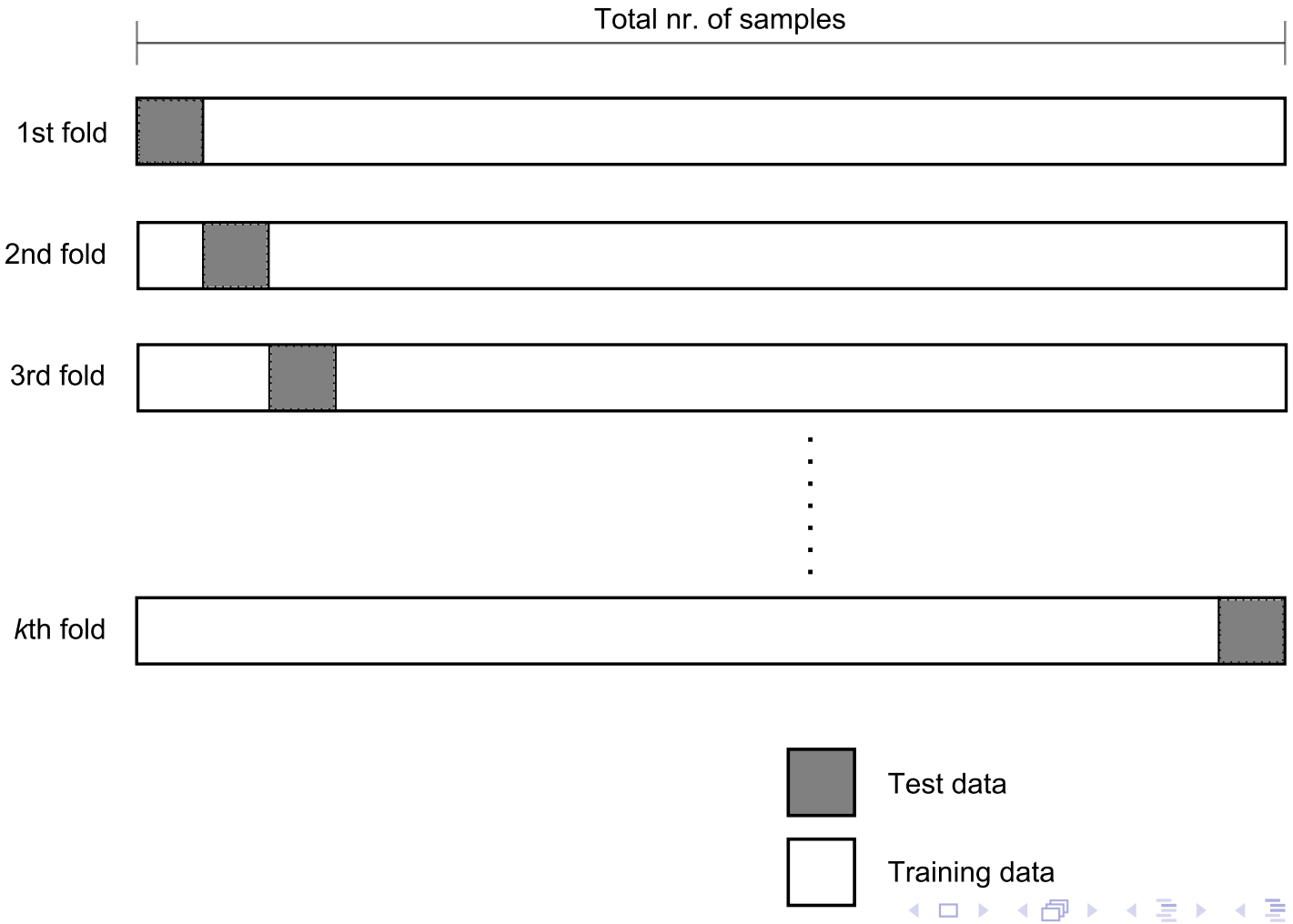
i. $\lim_{n \to \infty} \mathbb{E}\left[\hat{\mathbf{\Omega}}_n(\lambda_n)\right] \longrightarrow \lim_{n \to \infty} \mathbb{E}\left(\mathbf{S}_n^{-1}\right) = \mathbf{\Omega}$;

ii. $\lim_{n \to \infty} \mathbb{E}\left(\|\hat{\mathbf{\Omega}}_n(\lambda_n) - \mathbf{\Omega}\|_F^2\right) = 0$.

# Choosing the penalty value

## *K*-fold cross-validation (CV)

### Single iteration of *K*-fold CV



# Choosing the penalty value

## $K$-fold CV score

$$\varphi^K(\lambda) = \sum_{k=1}^{K} n_k \left\{ -\ln |\hat{\mathbf{\Omega}}(\lambda)_{-k}| + \text{tr}[\hat{\mathbf{\Omega}}(\lambda)_{-k}\mathbf{S}_k] \right\},$$

$n_k$ is the size of subset $k$, for $k = 1, \ldots, K$ disjoint subsets;
$\mathbf{S}_k$ denotes the sample covariance matrix on $kth$ test set;
$\hat{\mathbf{\Omega}}(\lambda)_{-k}$ denotes the estimated regularized precision matrix on $k$th training set

## Choose

$\lambda^* = \arg\min_{\lambda \in \mathbb{R}^+} \varphi^K(\lambda)$

## Efficiency

- Implementation uses C++ at its core
- Makes use of a root-finding (Brent) algorithm
- Utilizes rotational equivariance property when possible

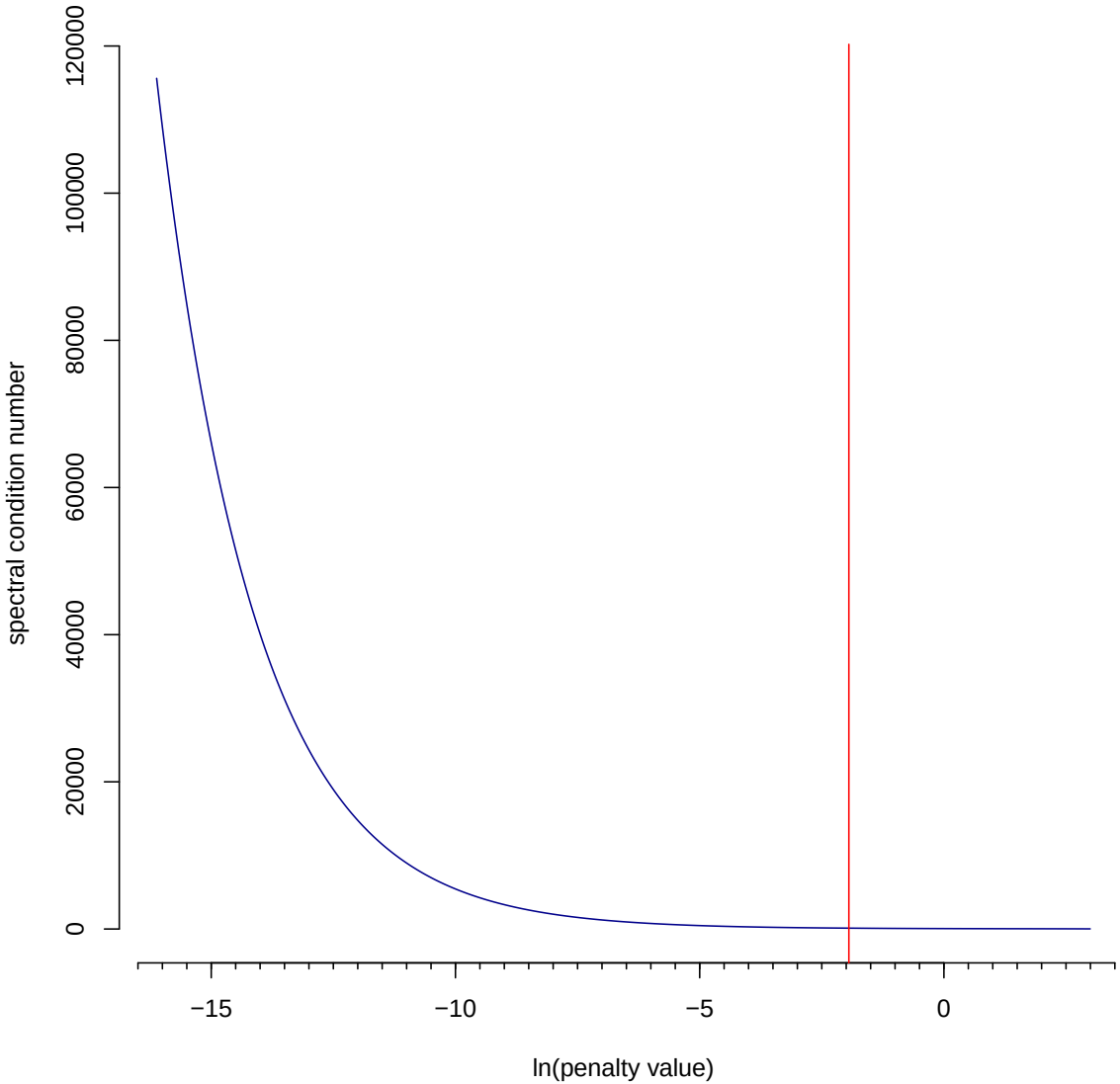
# Exercise 2: Find an optimal precision matrix for the Class 2 AD data

```
optPenalty.kCVauto(Y,          <-- data (matrix)
                   lambdaMin,  <-- min. λ
                   lambdaMax,  <-- max. λ
                   target)     <-- T (use default.target())
```

## Returns list object

- `$optLambda`: Optimal penalty parameter
- `$optPrec`: Precision estimate under optimal penalty parameter

# Assessing the Conditioning of the Estimate: Condition Number Plot



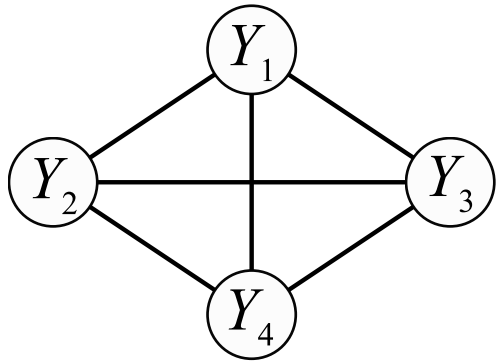
# Exercise 3: Assess the conditioning of the optimal precision matrix

```
CNplot(S,
       lambdaMin,
       lambdaMax,
       step,
       target,
       vertical,
       value)
```

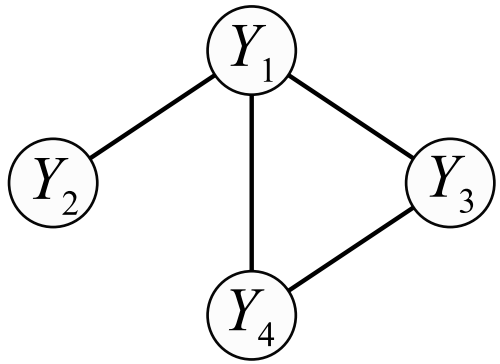
← covariance matrix data
← min. λ
← max. λ
← coarseness grid
← T
← logical

# Support Determination

$$\begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \omega_{14} \\ \omega_{21} & \omega_{22} & \omega_{23} & \omega_{24} \\ \omega_{31} & \omega_{32} & \omega_{33} & \omega_{34} \\ \omega_{41} & \omega_{42} & \omega_{43} & \omega_{44} \end{bmatrix}$$




$$\begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \omega_{14} \\ \omega_{21} & \omega_{22} & 0 & 0 \\ \omega_{31} & 0 & \omega_{33} & \omega_{34} \\ \omega_{41} & 0 & \omega_{43} & \omega_{44} \end{bmatrix}$$



# Support determination

## Scaling

$\hat{\mathbf{P}}(\lambda)$: Regularized precision estimate scaled to partial correlation form

## Assume

Nonredundant coefficients (indexed by $j < j'$) follow a mixture distribution:

$$f\left\{[\hat{\mathbf{P}}(\lambda^*)]_{jj'}\right\} = \eta_0 f_0\left\{[\hat{\mathbf{P}}(\lambda^*)]_{jj'}; \kappa\right\} + (1-\eta_0) f_{\mathcal{E}}\left\{[\hat{\mathbf{P}}(\lambda^*)]_{jj'}\right\}$$

- $\eta_0 \in [0,1]$ is the mixture weight
- $f_0\{\cdot\}$ denotes the distribution of a null-edge
- $f_{\mathcal{E}}\{\cdot\}$ denotes the distribution of a present edge
- $\kappa$ denotes degrees of freedom

## Determine

$$P\left(Y_j \neq Y_{j'} | [\hat{\mathbf{P}}(\lambda^*)]_{jj'}\right) = \frac{\hat{\eta}_0 f_0\left\{[\hat{\mathbf{P}}(\lambda^*)]_{jj'}; \hat{\kappa}\right\}}{\hat{\eta}_0 f_0\left\{[\hat{\mathbf{P}}(\lambda^*)]_{jj'}; \hat{\kappa}\right\} + (1-\hat{\eta}_0)\hat{f}_{\mathcal{E}}\left\{[\hat{\mathbf{P}}(\lambda^*)]_{jj'}\right\}}$$

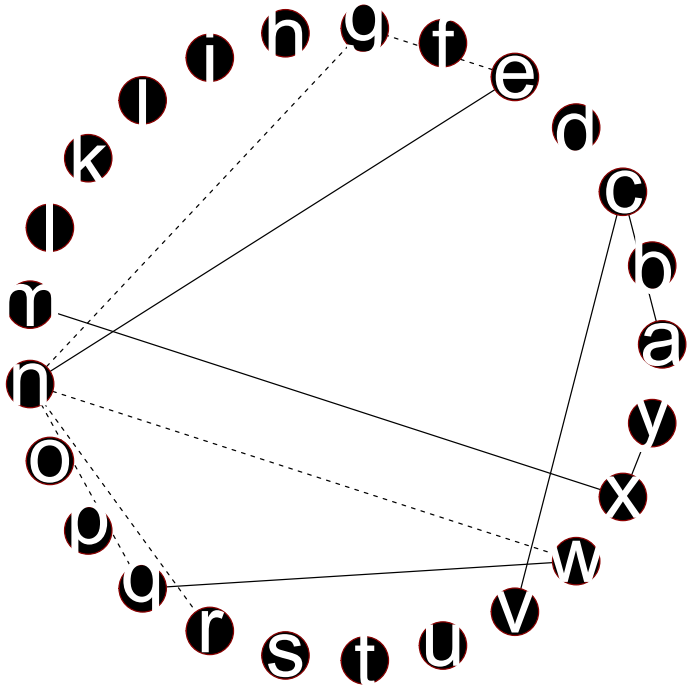
# Exercise 4: Extract a network from the optimal precision matrix by retaining elements whose posterior probability of being present $\geq .999$

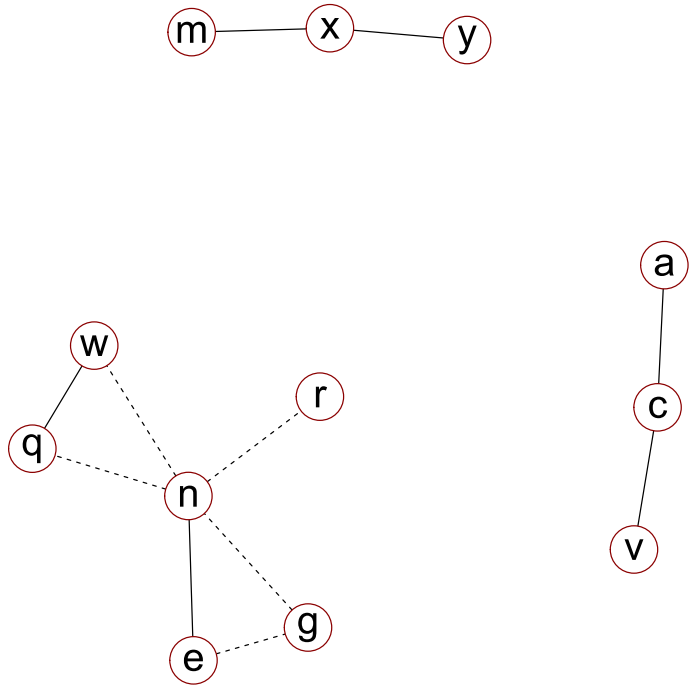
```
sparsify(P,          <-- estimated precision matrix
         threshold,  <-- type: "localFDR", "top"
         FDRcut)     <-- cut-off for 1 - lFDR
```

Returns list object

- `$sparsePrecision`: Sparsified precision matrix
- `$sparseParCor`: Sparsified partial correlation matrix

# Visualization




## Things to Consider

- Layout
- Size of vertices
- colorings
- ...

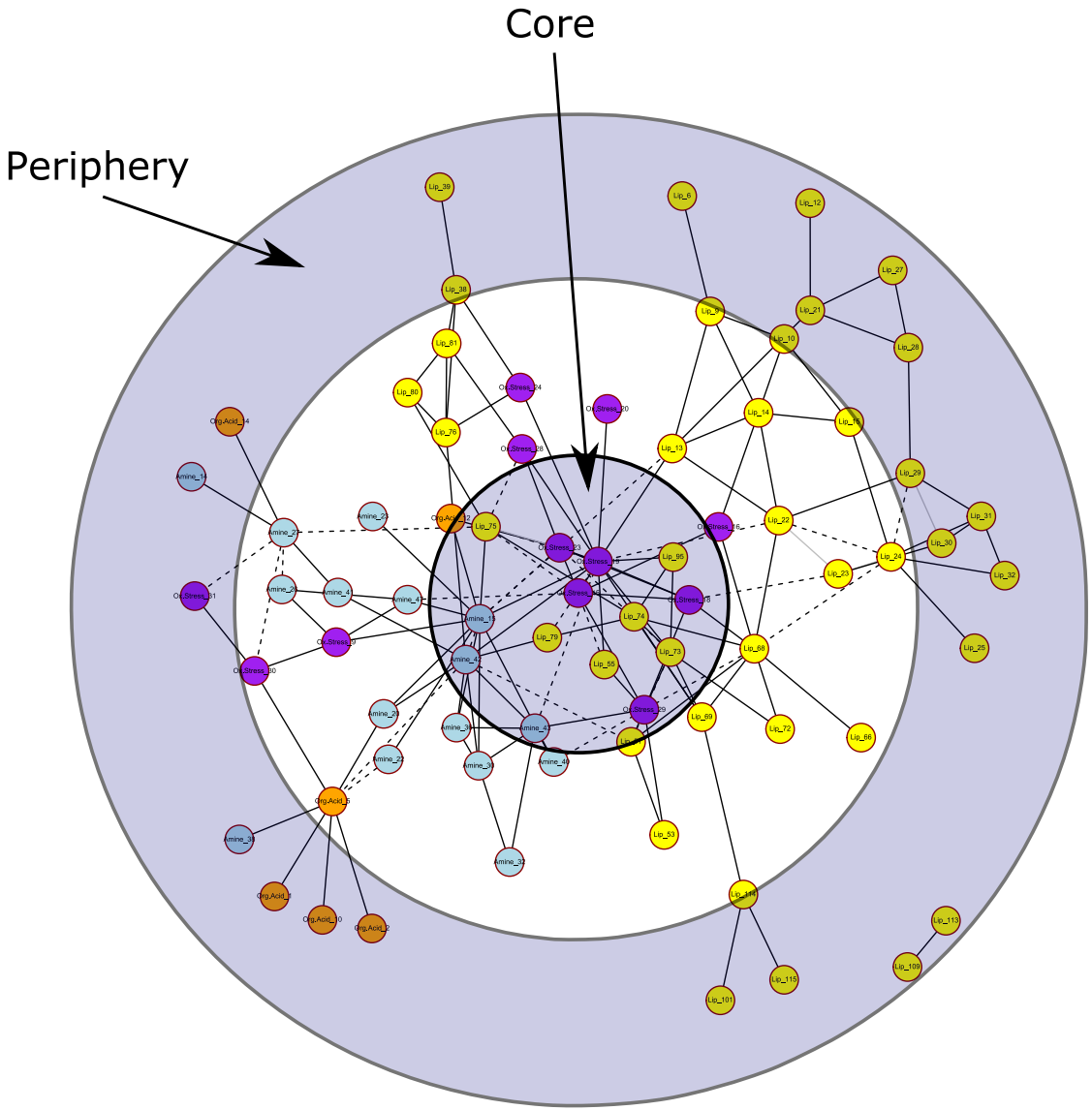
# Exercise 5: Visualize the extracted network

```
Ugraph(M,          <-- sparse matrix to be visualized
       type,       <-- "plain", "fancy", "weighted"
       lay,        <-- specifies layout
       Vsize,      <-- size of vertices
       Vcex,       <-- size vertex labels
       Vcolor,     <-- vertex color
       VBcolor,    <-- vertex boundary color
       VLcolor,    <-- color vertex labels
       prune,      <-- removes unconnected vertices?
       ...)
```

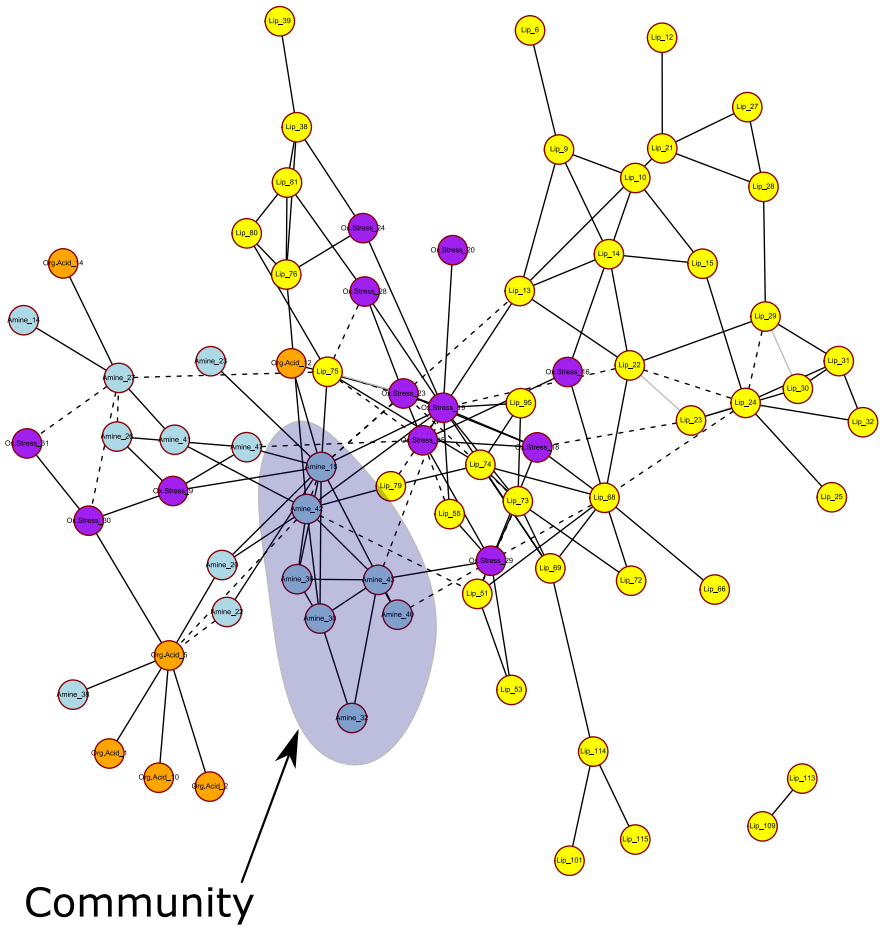
**Returns matrix object**

Containing the coordinates of the vertices in the given graph/network

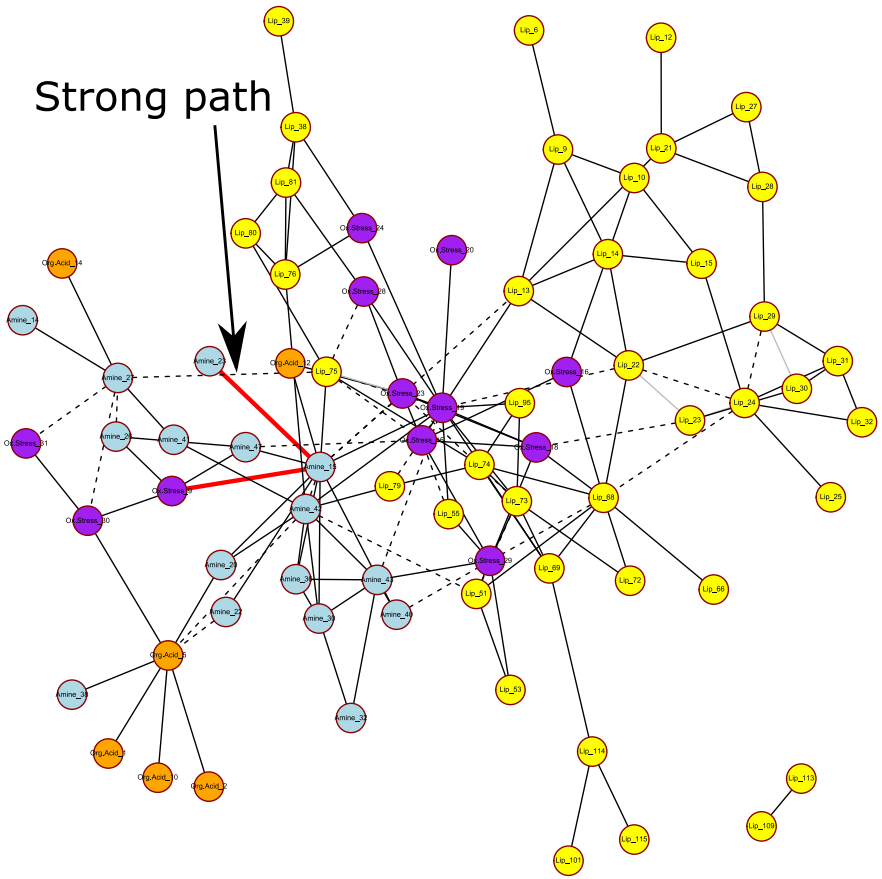
# Analysis: Global Level



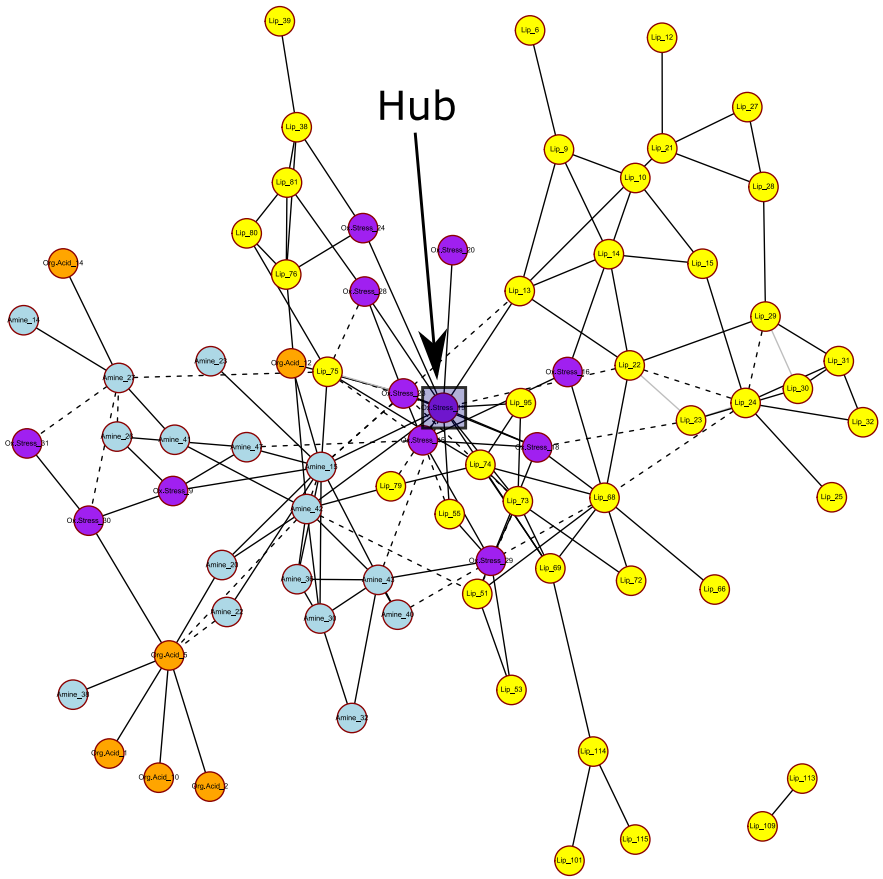
# Analysis: Group Level



# Analysis: Path Level



# Analysis: Node Level



# Node-Level Analysis: Centrality

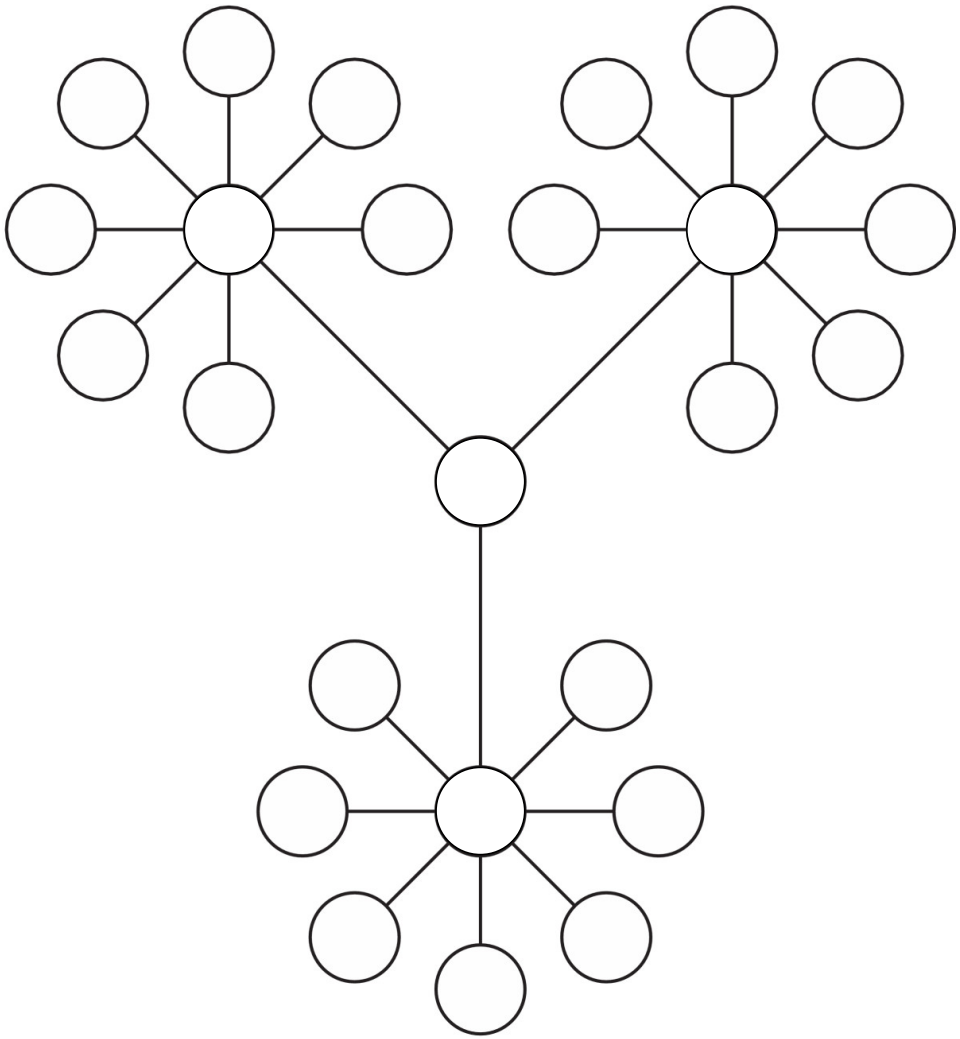


Illustration adapted from: http://shareablespaces.blogspot.nl/2010/03/social-networking-tools.html

# Node-Level Analysis: Degree Centrality

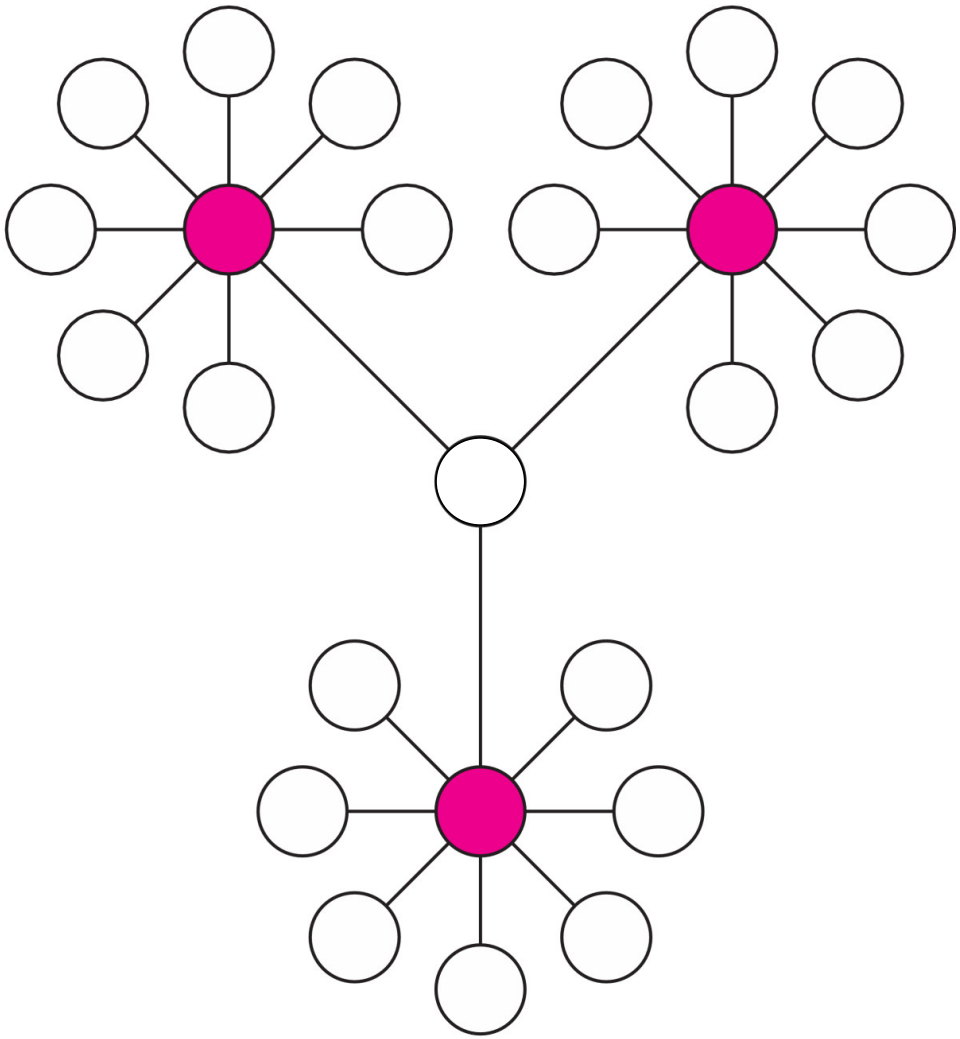


Illustration adapted from: http://shareablespaces.blogspot.nl/2010/03/social-networking-tools.html

# Node-Level Analysis: Betweenness Centrality

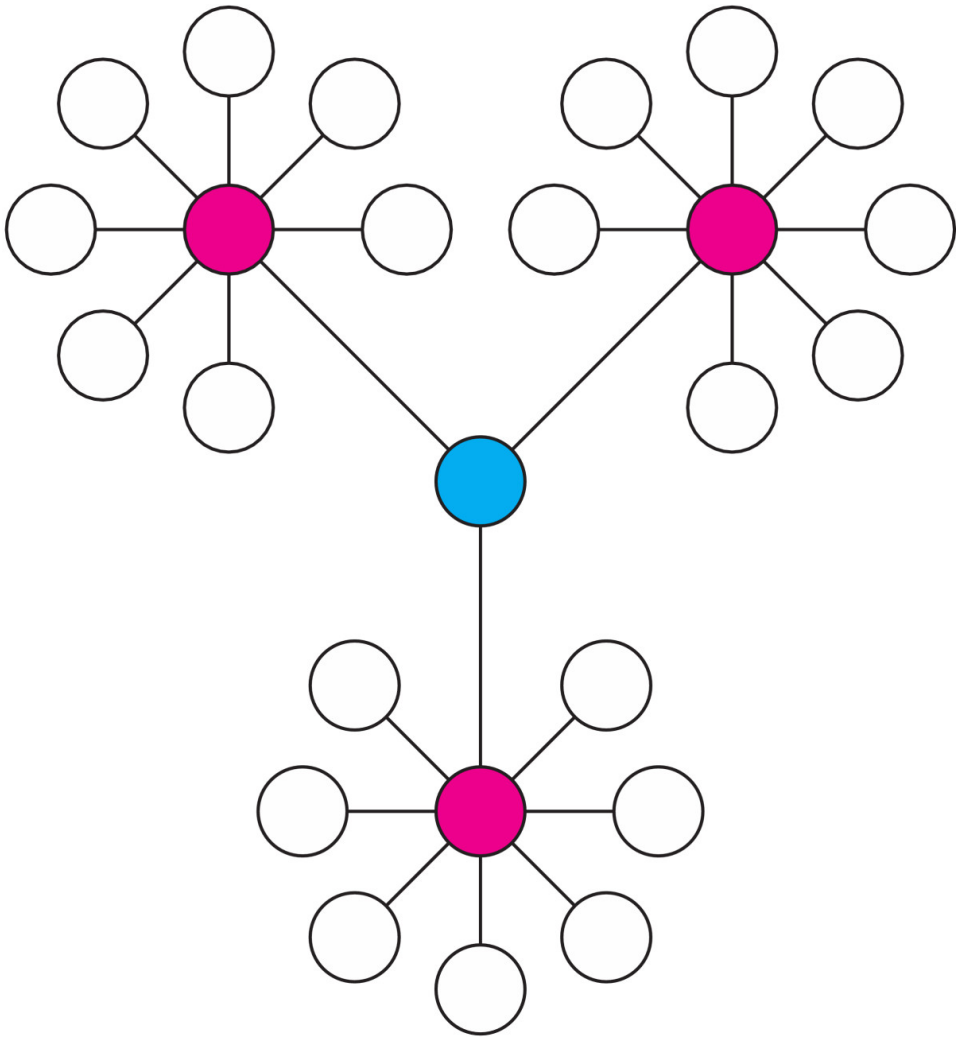
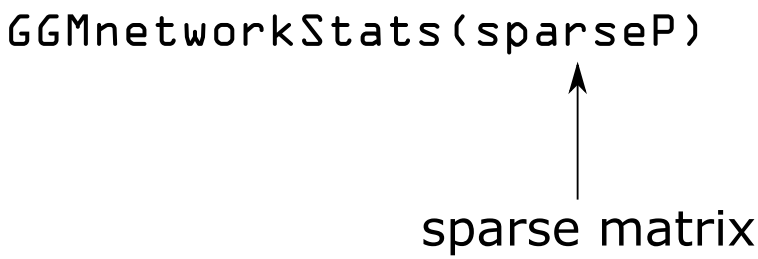


Illustration adapted from: http://shareablespaces.blogspot.nl/2010/03/social-networking-tools.html

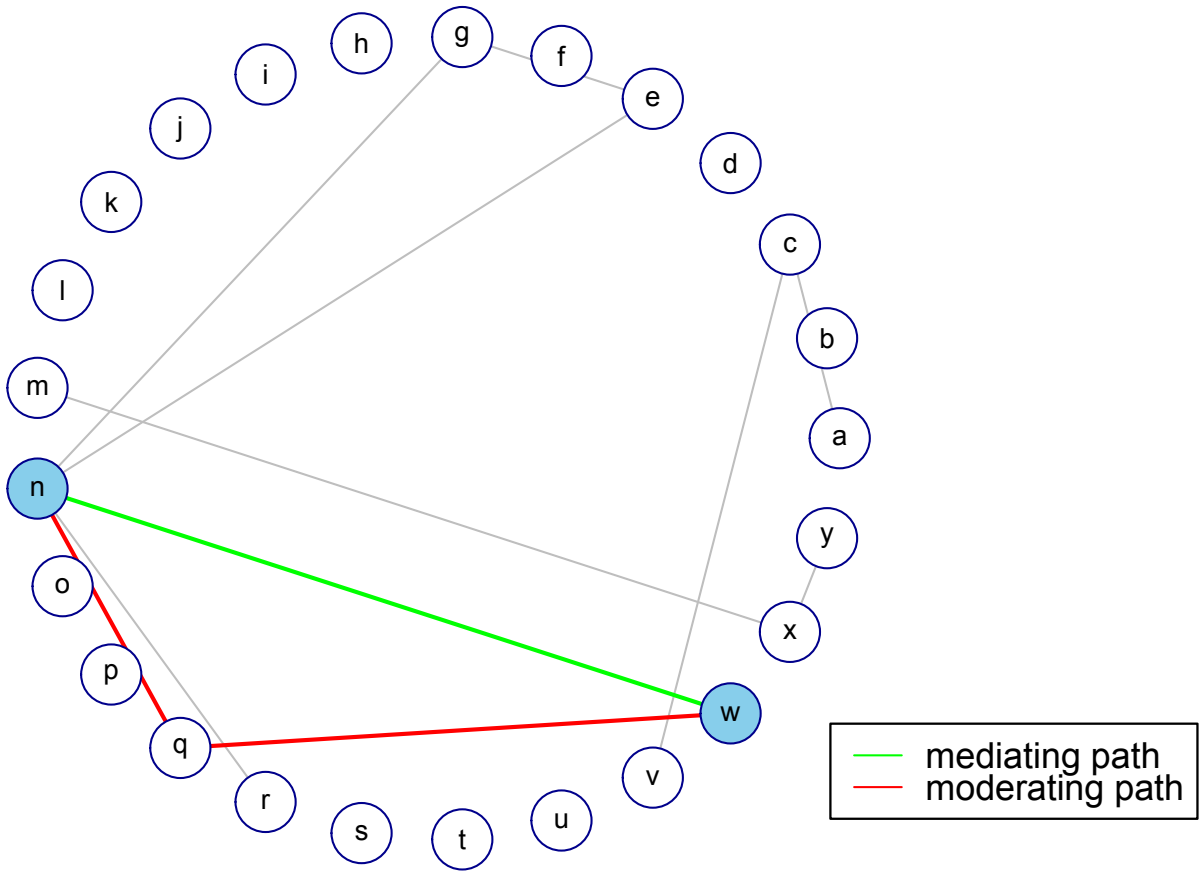
# Exercise 6: Find the nodes with the top degree and betweenness centralities




## Returns list object

- `$degree`: Degree centrality
- `$betweenness`: Betweenness centrality
- ...

# Path-Level Analysis: Mediating and Moderating Paths



# Exercise 7: Find 2 strongest paths between Amines 1 and 2

```
GGMpathStats(P0,      <-- sparse precision matrix
             node1,   <-- endpoint 1
             node2,   <-- endpoint 2
             graph,   <-- logical, should graph be produced?
             ...)     <-- arguments passed to Ugraph
```

## Returns list object

- `$pathStats`: Matrix specifying paths
- ...

# Group-Level Analysis: Finding Communities

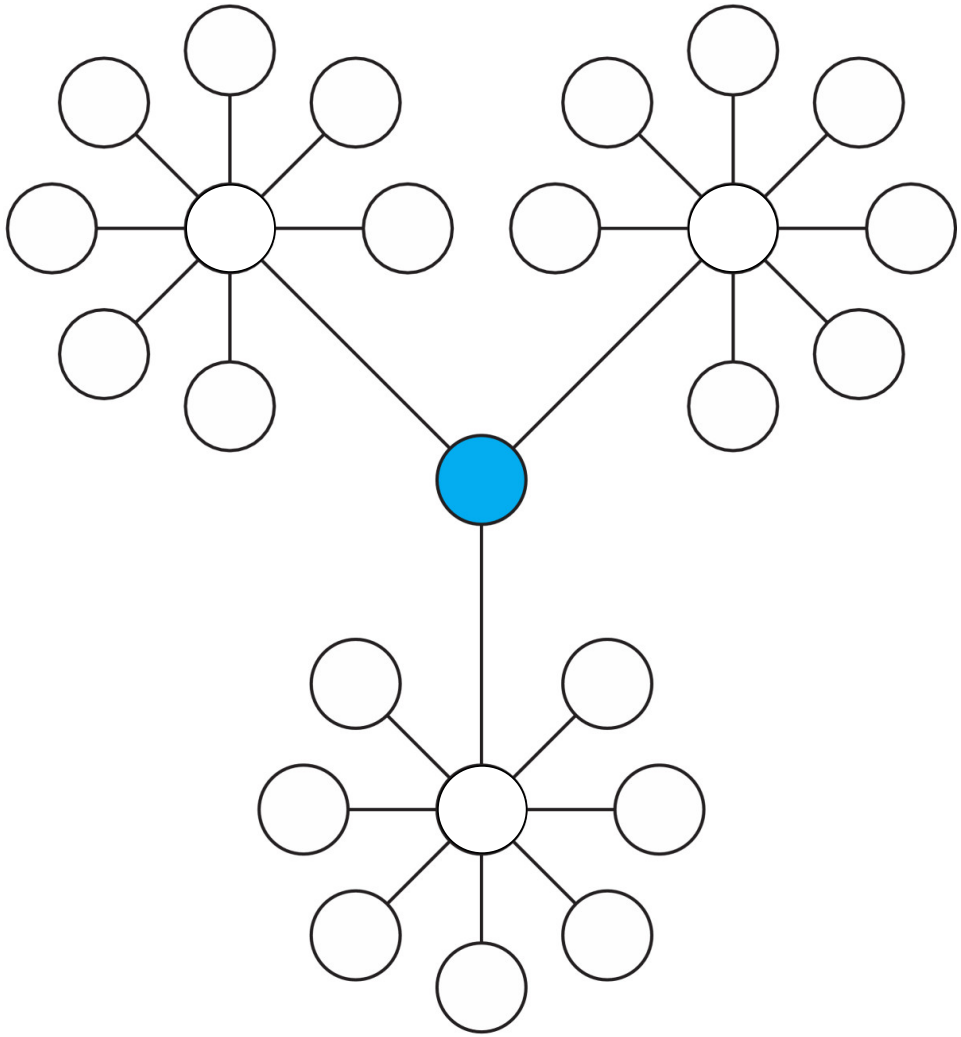


Illustration adapted from: http://shareablespaces.blogspot.nl/2010/03/social-networking-tools.html

# Group-Level Analysis: Finding Communities

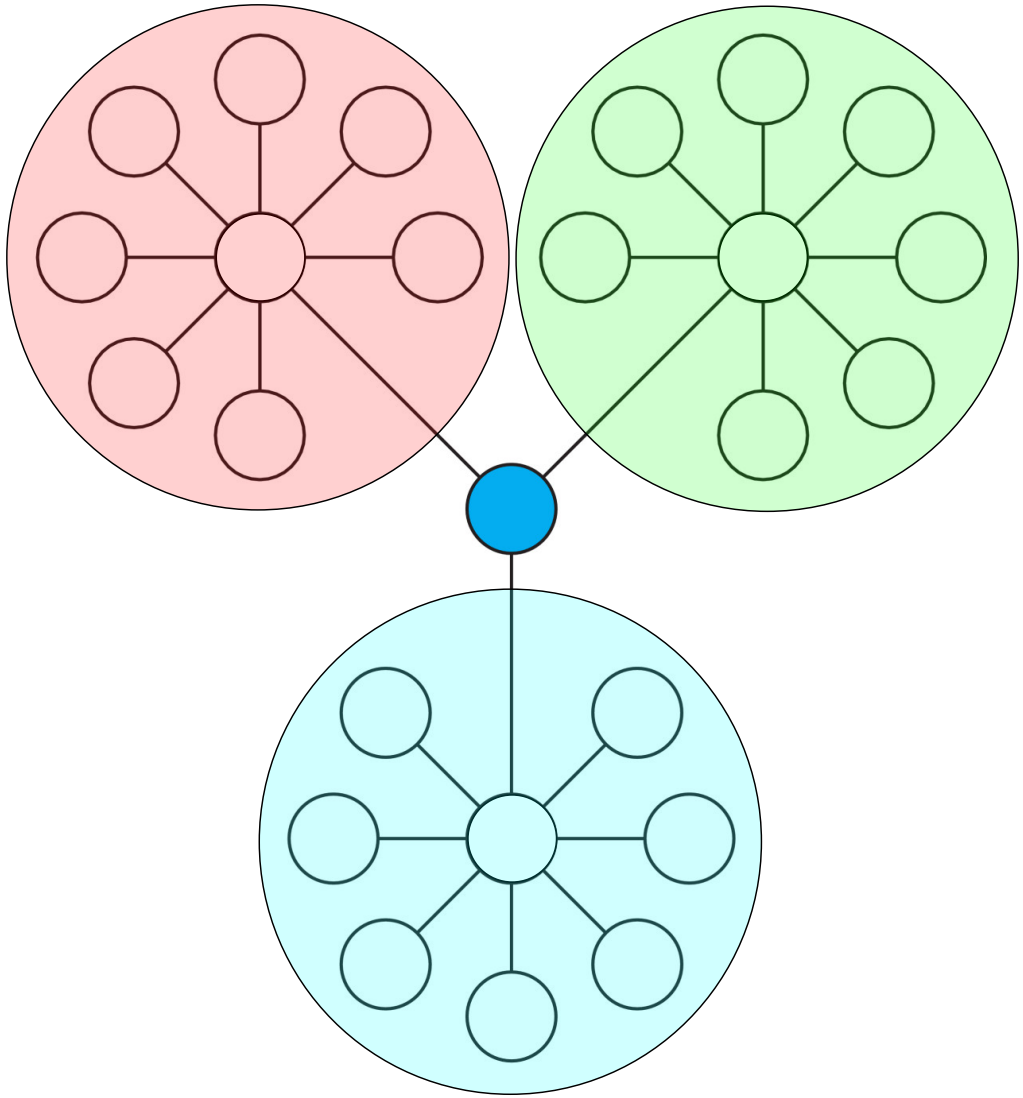


Illustration adapted from: http://shareablespaces.blogspot.nl/2010/03/social-networking-tools.html

# Exercise 8: Find and visualize communities for the extracted network

```
Communities(P,
            graph,
            lay,
            Vsize,
            Vcex,
            Vcolor,
            VBcolor,
            VLcolor,
            main)
```

P: sparse matrix

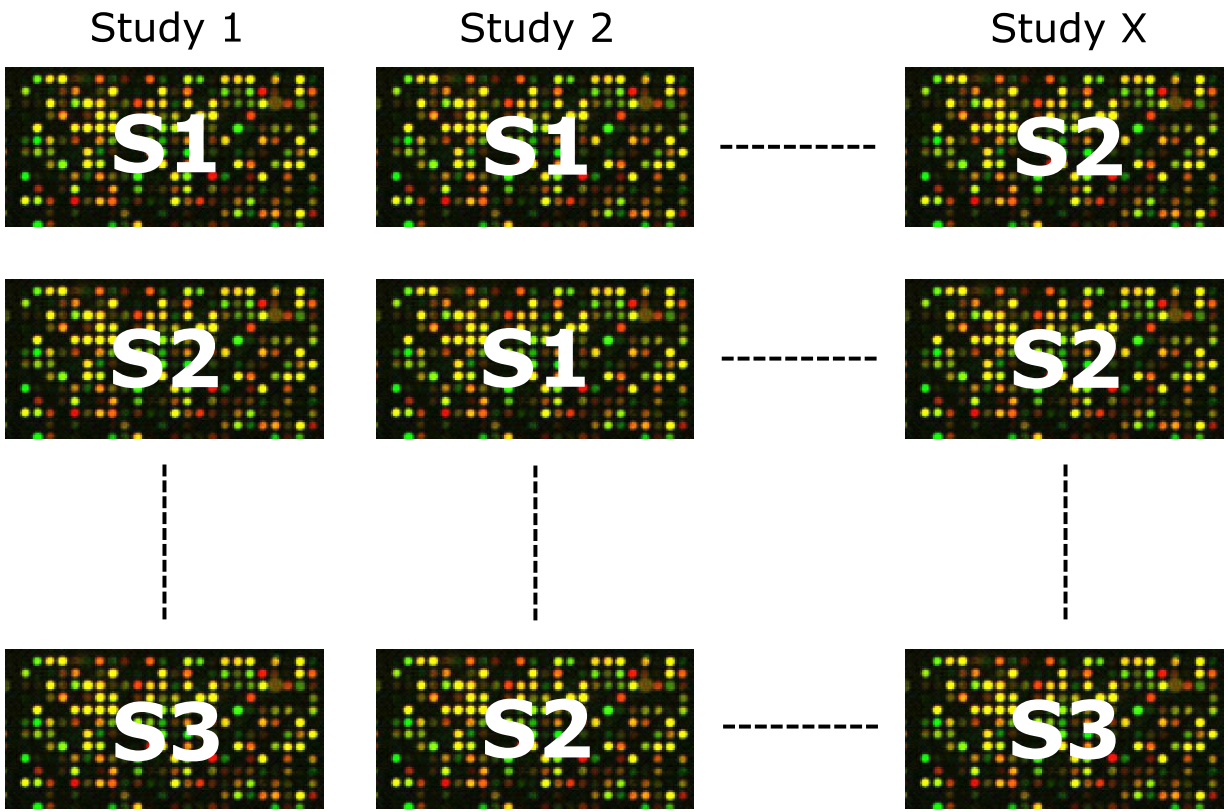
graph: logical, if TRUE, then graph is given

lay, Vsize, Vcex, Vcolor, VBcolor, VLcolor, main: arguments Ugraph

## Returns list object

- `$membership`: Community membership for each feature
- `$modularityscore`: Modularity score

# Multiple data classes




# Situation

## Data

- G classes of $(n_g \times p)$-dimensional data
- Classes defined by data sets and/or (subtypes of) diseases

## Assumption

Precision matrices of constituent classes chiefly share the same structure but potentially differ in a number of locations of interest

## Desire

Integrative or meta-analytic Gaussian graphical modeling

# Targeted fused ridge estimation: General Formulation

## Maximize

$$\underbrace{\mathcal{L}\left(\{\boldsymbol{\Omega}_g\};\{\mathbf{S}_g\}\right)}_{\text{log-likelihood}} - \sum_g \underbrace{\frac{\lambda_{gg}}{2}\left\|\boldsymbol{\Omega}_g - \mathbf{T}_g\right\|_F^2}_{\text{ridge-penalty}} - \sum_{g_1,g_2} \underbrace{\frac{\lambda_{g_1g_2}}{4}\left\|(\boldsymbol{\Omega}_{g_1} - \mathbf{T}_{g_1}) - (\boldsymbol{\Omega}_{g_2} - \mathbf{T}_{g_2})\right\|_F^2}_{\text{fusion-penalty}}$$

- $\mathbf{T}_g$ indicate class-specific target matrices
- $\lambda_{gg} \in (0, \infty)$ denote class-specific ridge penalty parameters
- $\lambda_{g_1g_2} \in [0, \infty)$ denote pair-specific fusion penalty parameters, $\lambda_{g_1g_2} = \lambda_{g_2g_1}$

## Penalty matrix

All penalties can be collected into a non-negative symmetric matrix $\boldsymbol{\Lambda} = [\lambda_{g_1g_2}]$

# Targeted fused ridge estimation

## Maximizing argument for class $g_0$

$$\hat{\boldsymbol{\Omega}}_{g_0}\left(\boldsymbol{\Lambda}, \{\boldsymbol{\Omega}_g\}_{g \neq g_0}\right) = \left\{ \left[ \bar{\lambda}_{g_0} \mathbf{I}_p + \frac{1}{4}\left(\bar{\mathbf{S}}_{g_0} - \bar{\lambda}_{g_0} \mathbf{T}_{g_0}\right)^2 \right]^{1/2} + \frac{1}{2}\left(\bar{\mathbf{S}}_{g_0} - \bar{\lambda}_{g_0} \mathbf{T}_{g_0}\right) \right\}^{-1},$$

where

$$\bar{\mathbf{S}}_{g_0} = \mathbf{S}_{g_0} - \sum_{g \neq g_0} \frac{\lambda_{g g_0}}{n_{g_0}} (\boldsymbol{\Omega}_g - \mathbf{T}_g), \quad \text{and} \quad \bar{\lambda}_{g_0} = \frac{\sum_g \lambda_{g g_0}}{n_{g_0}}$$

# Properties

## Behavior

i. $\hat{\mathbf{\Omega}}_g \succ \mathbf{0}$ for all $\lambda_{gg} \in (0, \infty)$;

ii. $\lim_{\lambda_{gg} \to 0^+} \hat{\mathbf{\Omega}}_g = \mathbf{S}_g^{-1}$ if $\sum_{g' \neq g} \lambda_{gg'} = 0$ and $p \leq n_g$;

iii. $\lim_{\lambda_{gg} \to \infty} \hat{\mathbf{\Omega}}_g = \mathbf{T}_g$ if $\lambda_{gg'} < \infty$ for all $g' \neq g$;

iv. $\lim_{\lambda_{g_1 g_2} \to \infty} (\hat{\mathbf{\Omega}}_{g_1} - \mathbf{T}_{g_1}) = \lim_{\lambda_{g_1 g_2} \to \infty} (\hat{\mathbf{\Omega}}_{g_2} - \mathbf{T}_{g_2})$ if $\lambda_{g_1' g_2'} < \infty$ for all $\{g_1', g_2'\} \neq \{g_1, g_2\}$.

# Block coordinate ascent

**Input:**
*Sufficient data:* $(\mathbf{S}_1, n_1), \ldots, (\mathbf{S}_G, n_G)$
*Penalty matrix:* $\boldsymbol{\Lambda}$
*Convergence criterion:* $\varepsilon > 0$
**Output:**
*Estimates:* $\hat{\boldsymbol{\Omega}}_1, \ldots, \hat{\boldsymbol{\Omega}}_G$
**procedure** RIDGEP.FUSED$(\mathbf{S}_1, \ldots, \mathbf{S}_G, n_1, \ldots, n_G, \boldsymbol{\Lambda}, \varepsilon)$
    *Initialize*: $\hat{\boldsymbol{\Omega}}_g^{(0)}$ for all $g$.
    **for** $c = 1, 2, 3, \ldots$ **do**
        **for** $g = 1, 2, \ldots, G$ **do**
            Update $\hat{\boldsymbol{\Omega}}_g^{(c)} := \hat{\boldsymbol{\Omega}}_g\left(\boldsymbol{\Lambda}, \hat{\boldsymbol{\Omega}}_1^{(c)}, \ldots, \hat{\boldsymbol{\Omega}}_{g-1}^{(c)}, \hat{\boldsymbol{\Omega}}_{g+1}^{(c-1)}, \ldots, \hat{\boldsymbol{\Omega}}_G^{(c-1)}\right)$
        **end for**
        **if** $\max_g \left\{ \frac{\|\hat{\boldsymbol{\Omega}}_g^{(c)} - \hat{\boldsymbol{\Omega}}_g^{(c-1)}\|_F^2}{\|\hat{\boldsymbol{\Omega}}_g^{(c)}\|_F^2} \right\} < \varepsilon$ **then**
            **return** $\left(\hat{\boldsymbol{\Omega}}_1^{(c)}, \ldots, \hat{\boldsymbol{\Omega}}_G^{(c)}\right)$
        **end if**
    **end for**
**end procedure**

# Functional analogues

| Core | Fused |
|---|---|
| `ridgeP` | `ridgeP.fused` |
| `optPenalty` | `optPenalty.fused` |
| `sparsify` | `sparsify.fused` |
| `GGMnetworkStats` | `GGMnetworkStats.fused` |
| `GGMpathStats` | `GGMpathStats.fused` |

# Exercise 9: Construct lists of class-specific target and data matrices

```
## Subset
ADclass1 <- ADmetabolites[, sampleInfo$ApoEClass == "Class 1"]
ADclass2 <- ADmetabolites[, sampleInfo$ApoEClass == "Class 2"]

## Transpose and scale data
ADclass1 <- scale(t(ADclass1))
ADclass2 <- scale(t(ADclass2))

## Correlations for subsets
rAD1 <- cor(ADclass1)
rAD2 <- cor(ADclass2)

## Constructing list of correlation matrices
Rlist = list(rAD1 = rAD1, rAD2 = rAD2)
samps = c(dim(ADclass1)[1], dim(ADclass2)[1])

## Constructing list of target matrices and data
Tlist <- default.target.fused(Slist = Rlist, ns = samps, type = "DUPV")
Ylist <- list(AD1data = ADclass1, AD2data = ADclass2)
```

# Exercise 10: Find optimal precision matrices for Class 1 and Class 2 AD data

```
optPenalty.fused(Ylist,    <-- list of data matrices
                 Tlist,    <-- list of target matrices
                 cv.method)<-- CV method: choose LOOCV
```

Returns list object

- `$lambda.unique`: Optimal penalty parameters
- `$Plist`: List of precision matrices under optimal penalty parameters
- ...

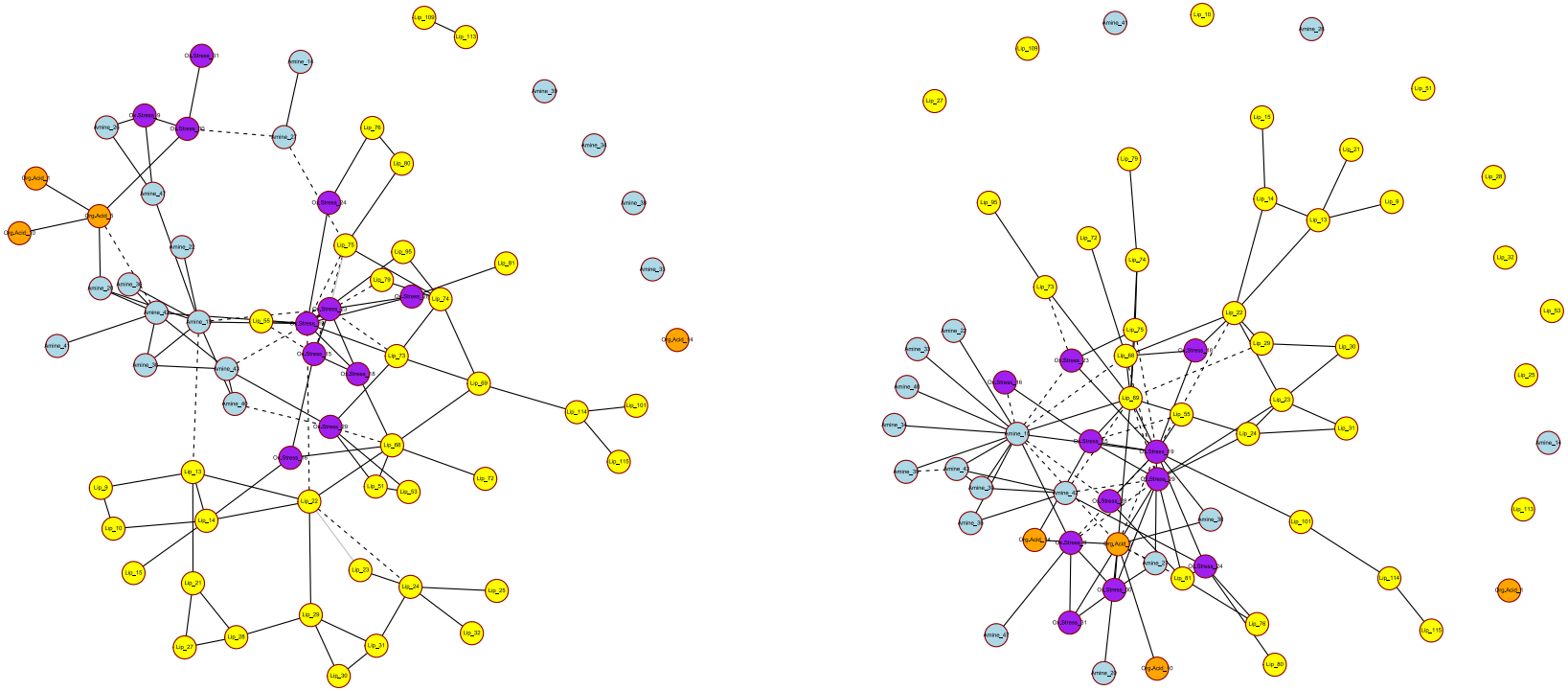
# Exercise 11: Extract class-specific networks on the basis of local FDR thresholding

```
sparsify.fused(Plist,      <-- list of estimated precision matrices
               threshold,  <-- type: "localFDR", "top"
               FDRcut)     <-- cut-off for 1 - lFDR
```

Returns list object for each class

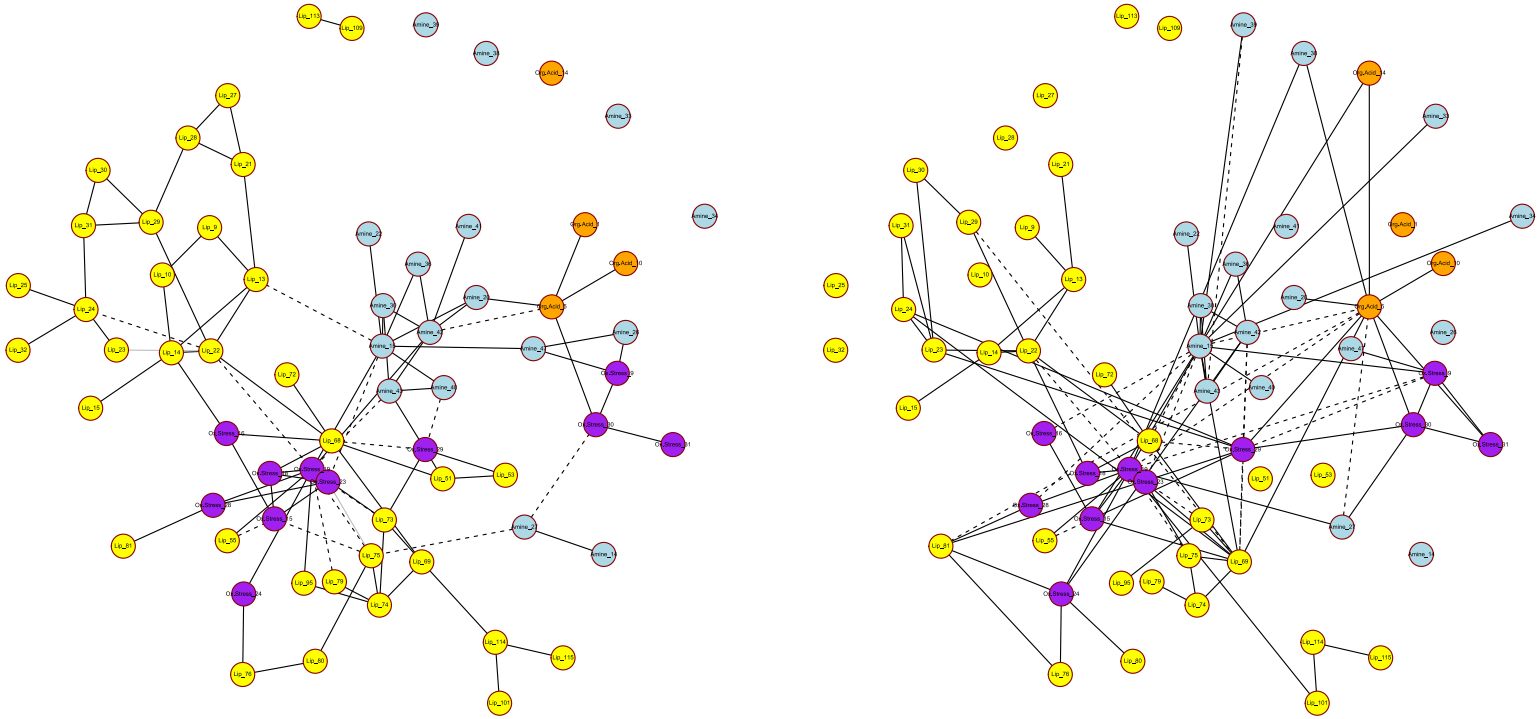
- `$sparsePrecision`: Sparsified precision matrix
- `$sparseParCor`: Sparsified partial correlation matrix

# Visualization



Two class-specific networks, independently visualized with the FR algorithm

# Visualization



## Coordinate retainment supports visual comparison

Same two class-specific networks, visualized with the same node-coordinates

# Exercise 12: Visualize the retained networks in the same coordinates

```
Ugraph(M,
       type,
       lay,
       coords,
       Vsize,
       Vcex,
       Vcolor,
       VBcolor,
       VLcolor,
       prune
       ...)
```

← if `lay = NULL`, then layout according to coordinates

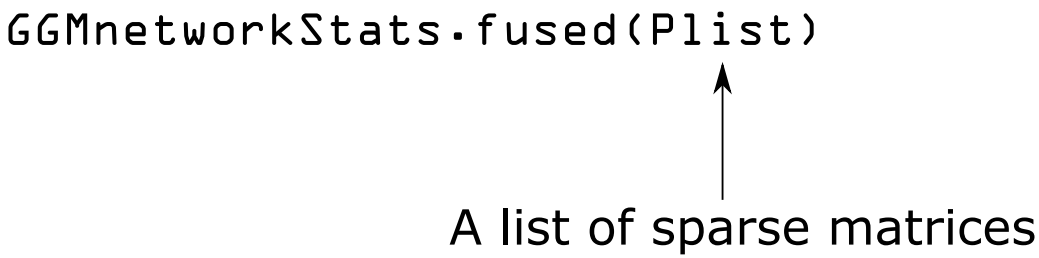
## Returns matrix object

- Containing coordinates of layout

## Tips

- Use the `union` function.
- Use coordinates return object.

# Exercise 13: Compare the top node-degrees for the class-specific networks




Returns `data.frame`

Names of `Plist` are prefixed to column-names

# Exercise 14: Find and visualize communities for the class-specific networks

```
Communities(P,
            graph,
            lay,
            Vsize,
            Vcex,
            Vcolor,
            VBcolor,
            VLcolor,
            main)
```

← sparse matrix

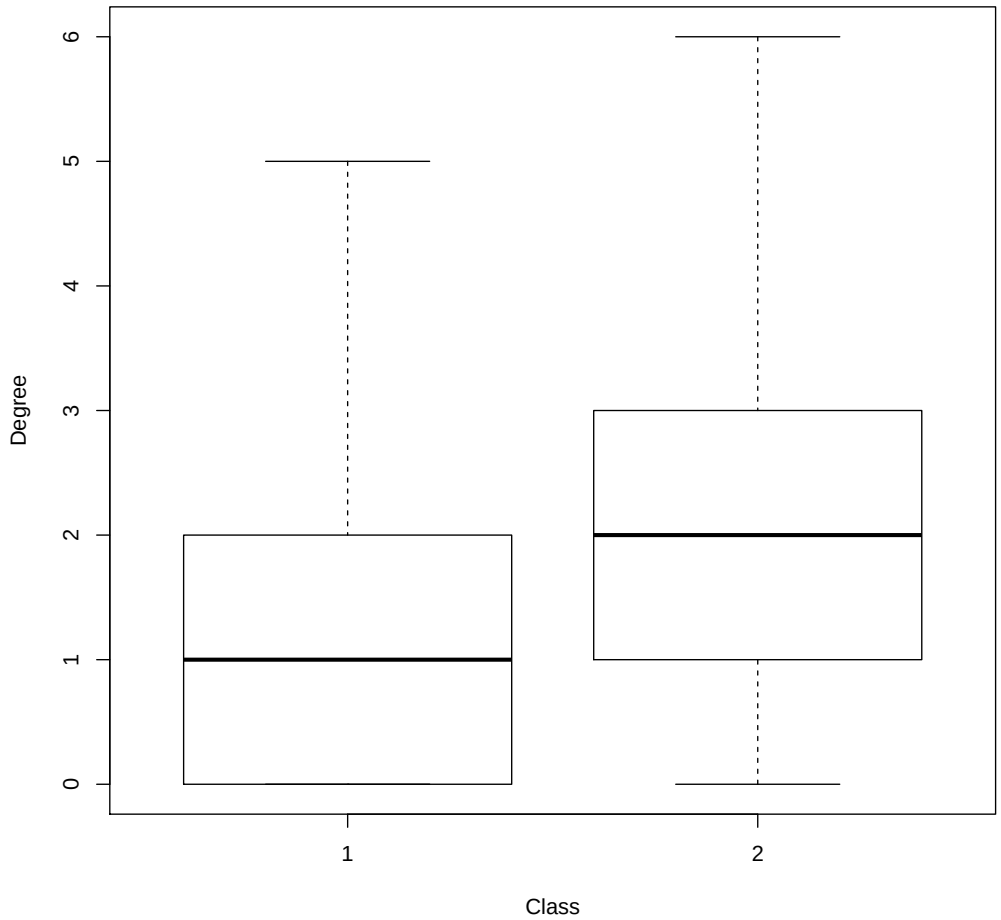
← logical, if TRUE, then graph is given

arguments Ugraph

## Returns list object

- $membership: Community membership for each feature
- $modularityscore: Modularity score

# Global Analysis: Degree Distributions



# Global Analysis: Entropy

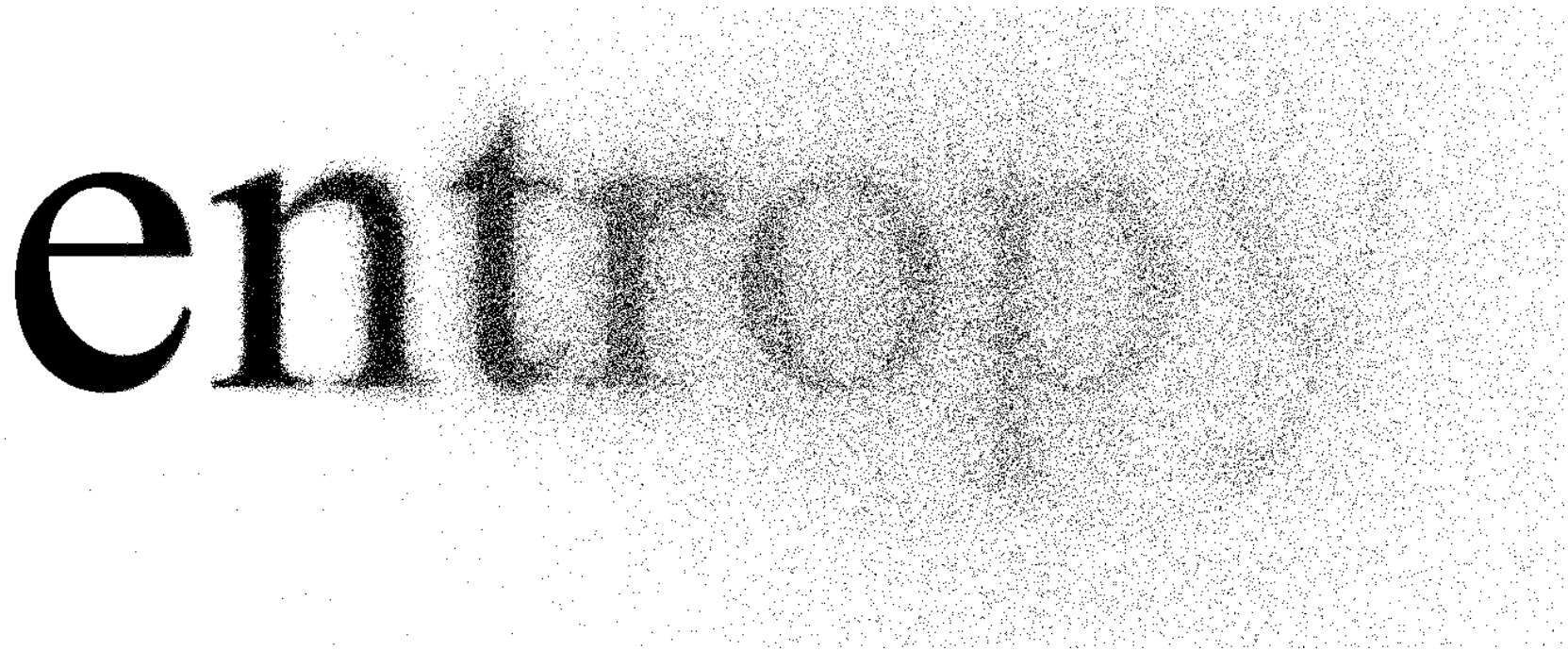



Illustration adapted from: http://creepypasta.wikia.com/wiki/Entropy

# Exercise 15: Compare either the degree distributions or the entropies for the class-specific networks

## Comparing degree distributions

Can be done with simple test, such as the Wilcoxon Signed Rank Test

## Comparing entropies

```
library(sigaR)

entropyTest(Y,      <-- data matrix, samples in rows
            id,     <-- class indicator (coded 0,1)
            nPerm,  <-- number of permutations
            method) <-- distributional assumption: use "knn"
```

## Entropy test returns test information

```
summary()
```

- Brent, R.P. (1971). "An Algorithm with Guaranteed Convergence for Finding a Zero of a Function". Computer Journal, 14: 422-425.
- Csardi, G. & Nepusz, T. (2006). "The igraph software package for complex network research". InterJournal, Complex Systems 1695. http://igraph.sf.net
- Fruchterman, T.M.J. & Reingold, E.M. (1991). "Graph Drawing by Force-Directed Placement". Software: Practice & Experience, 21: 1129-1164.
- Jones, B. & West, M. (2005). "Covariance Decomposition in Undirected Gaussian Graphical Models". Biometrika, 92: 779-786.
- Newman, M.E.J. (2010). Networks: an introduction. Oxford University Press.
- Newman, M.E.J. & Girvan, M. (2004). "Finding and evaluating community structure in networks". Physical Review E, 69: 026113.
- Schäfer, J. & Strimmer, K. (2005a). "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics". Statistical Applications in Genetics and Molecular Biology, 4: art. 32.
- Schäfer, J. & Strimmer, K. (2005b). "An empirical Bayes approach to inferring large-scale gene association networks". Bioinformatics, 21: 754764.
- van Wieringen, W.N. & van der Vaart, A.W. (2011). "Statistical analysis of the cancer cells molecular entropy using high-throughput data". Bioinformatics, 27: 556-563.

## Manual/Download


- Peeters, C.F.W., Bilgrau, A.E., & van Wieringen, W.N. (2017). "`rags2ridges`: Ridge Estimation of Precision Matrices from High-Dimensional Data". R package, version 2.2. URL: https://cran.r-project.org/package=rags2ridges.


## Theory/Methodology


- Bilgrau*, A.E., Peeters*, C.F.W., Eriksen, P.S., Bøgsted, M., & van Wieringen, W.N. (2015). "Targeted Fused Ridge Estimation of Inverse Covariance Matrices from Multiple High-Dimensional Data Classes". arXiv:1509.07982v1 [stat.ME].
- Peeters, C.F.W., van Wieringen, W.N., & van de Wiel, M.A. (in preparation). "Directed Cyclic Mixed Graph Modeling for High-Dimensional Omic Data Integration".
- van Wieringen, W.N. & Peeters, C.F.W. (2016). "Ridge Estimation of Inverse Covariance Matrices from High-Dimensional Data". Computational Statistics & Data Analysis, 103: 284-303. arXiv:1403.0904v3 [stat.ME].


## Software


- Peeters, C.F.W., van de Wiel, M.A., & van Wieringen, W.N. (2016) "The Spectral Condition Number Plot for Regularization Parameter Determination". arXiv:1608.04123v1 [stat.CO].
- van Wieringen, W.N. & Peeters, C.F.W. (2015). "Application of a New Ridge Estimator of the Inverse Covariance Matrix to the Reconstruction of Gene-Gene Interaction Networks". In: di Serio, C., Lio, P., Nonis, A., and Tagliaferri, R. (Eds.) 'Computational Intelligence Methods for Bioinformatics and Biostatistics'. Lecture Notes in Computer Science, vol. 8623. Springer, pp. 170–179.