

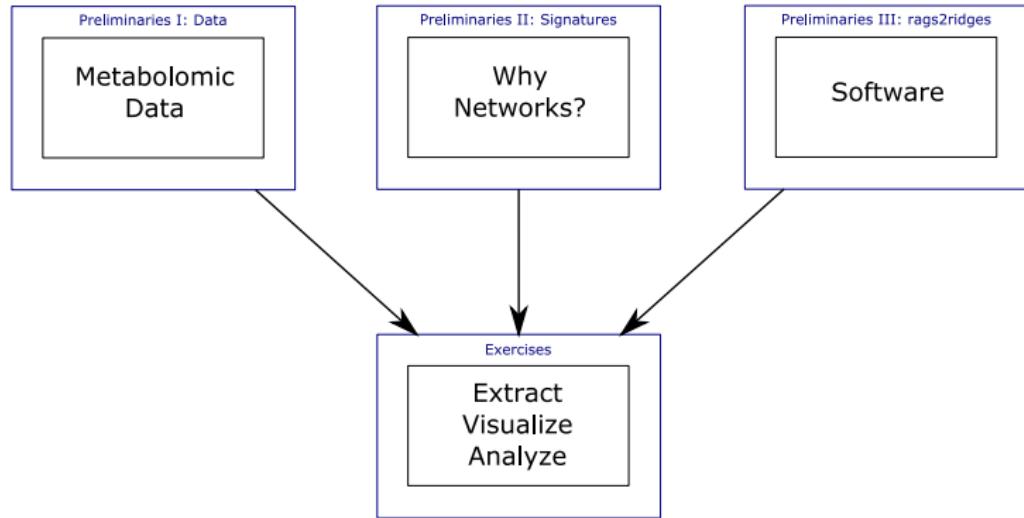
# rags2ridges

## Network Modeling of Omics Data

Carel F.W. Peeters  
Dept. of Epidemiology & Biostatistics  
Amsterdam University medical centers, Location VUmc  
[cf.peeters@amsterdamumc.nl](mailto:cf.peeters@amsterdamumc.nl)

Statistics for Omics course  
Amsterdam, the Netherlands  
June 27, 2019

# 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

# Omics and omics data

## -ome

A totality of some (molecular biological) sort

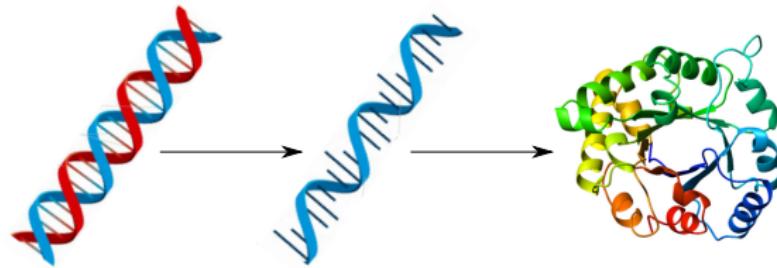
## -omics

Collective quantification of some pool of molecular molecules

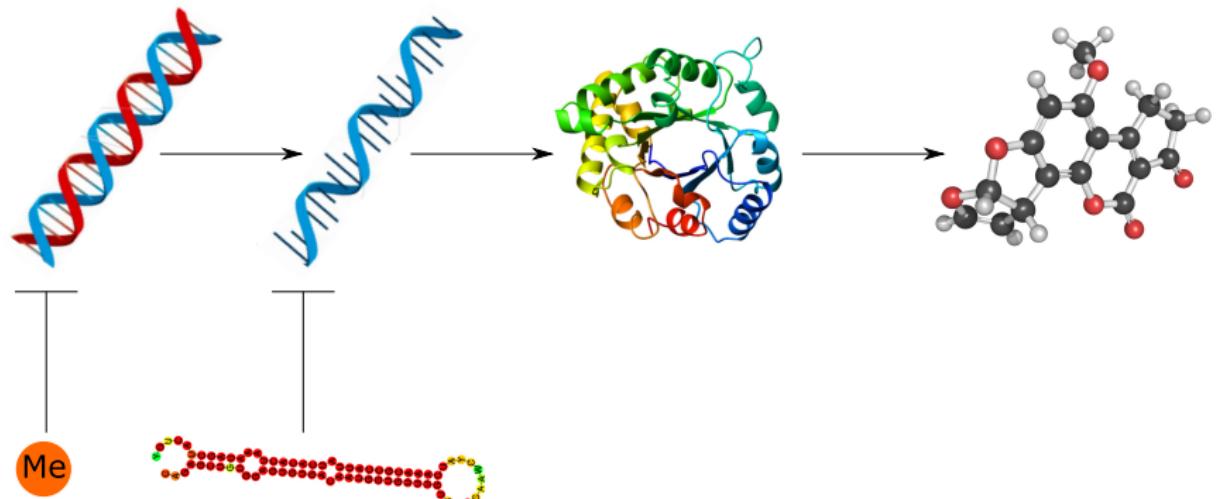
## Genomics

The omics of the genome (of some organism)

# The omic cascade

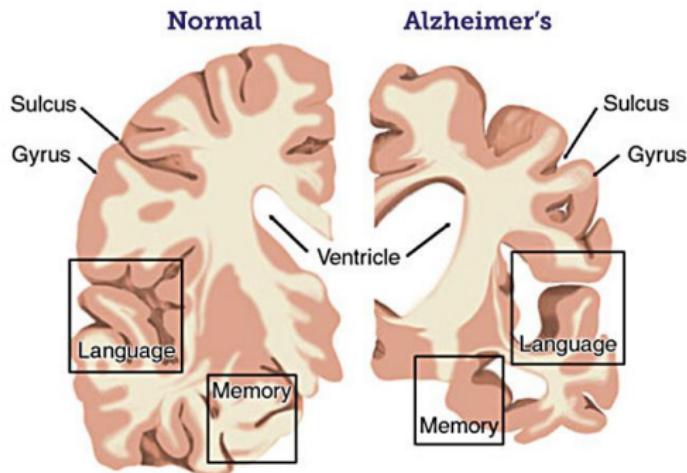


# The omic cascade



# Alzheimer's Disease

## Brain Cross-Sections



© 2000 by BrightFocus Foundation



# Metabolite quantification

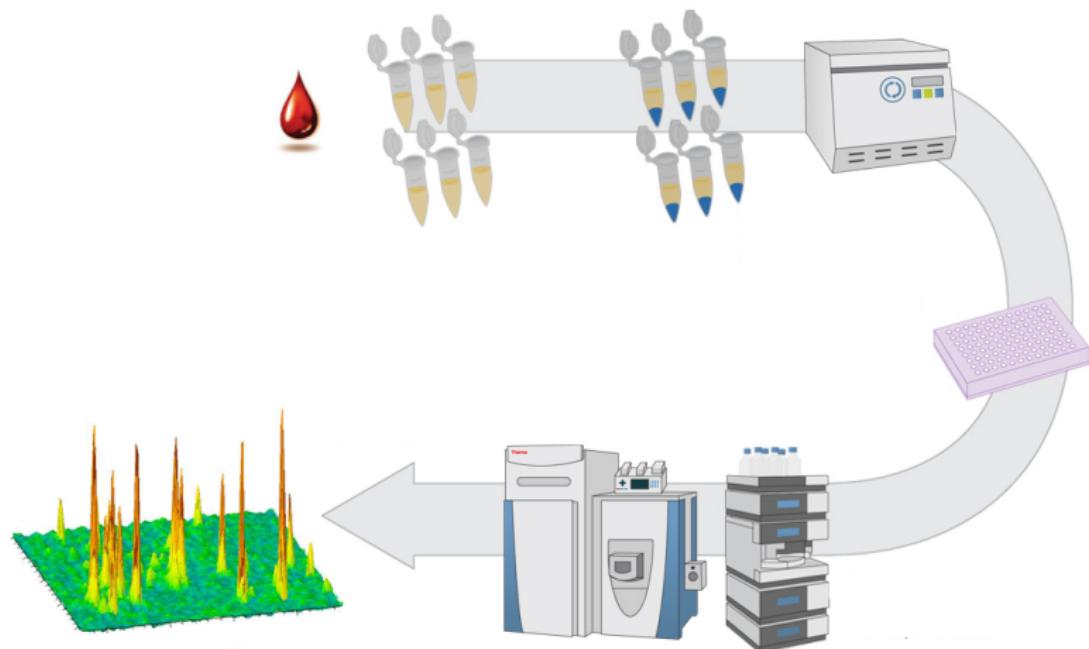


Illustration adapted from: <http://planetorbitrap.com/untargeted-metabolomics#.Vzw6yfmLRaQ> &  
<http://metabolomicsplatform.com/metabolomics-overview/>

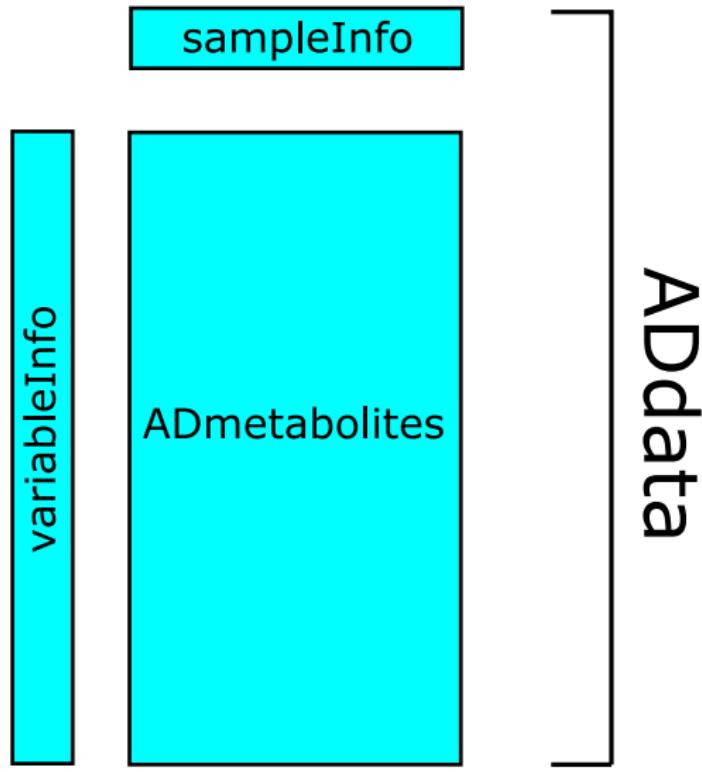
# Metabolomic Data

	Variables (features)						
	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: Look at Data

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

```
## Needed libraries  
library(rags2ridges)
```

Invoke data

```
load()
```

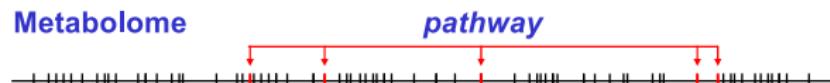
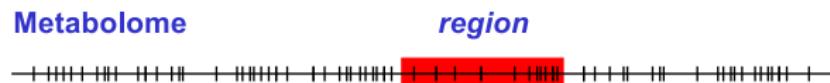
Which objects?

```
objects()
```

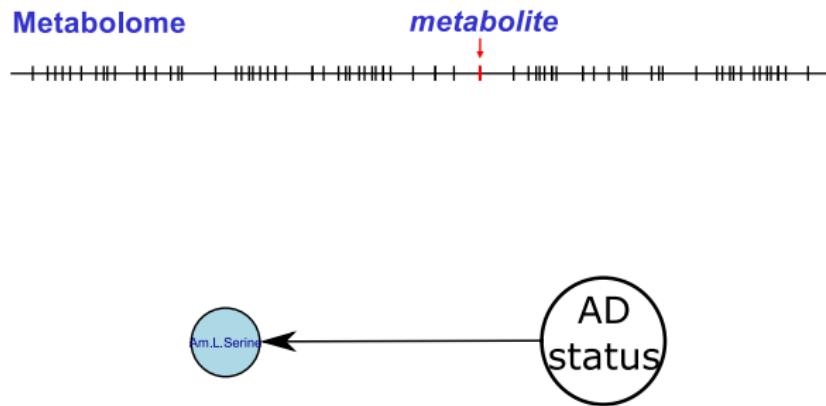
Simple exploration objects

```
head()
```

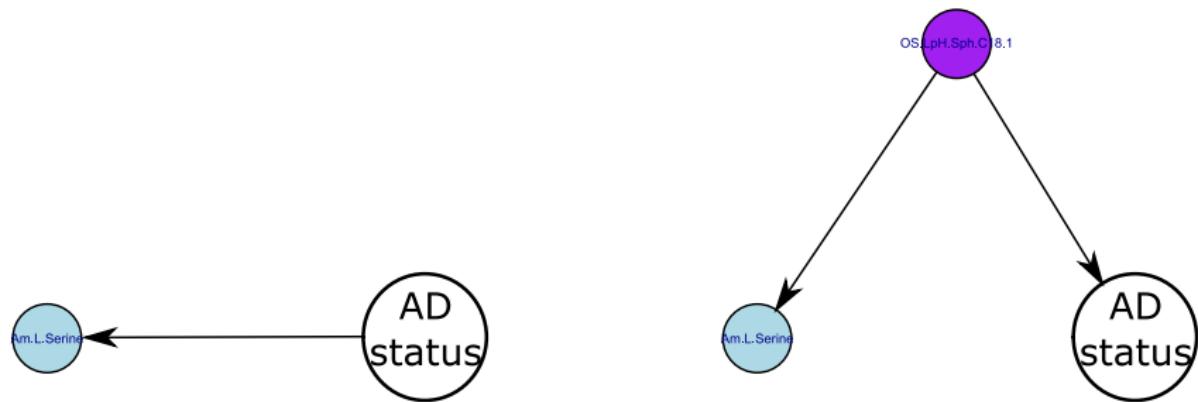
## Unit of analysis



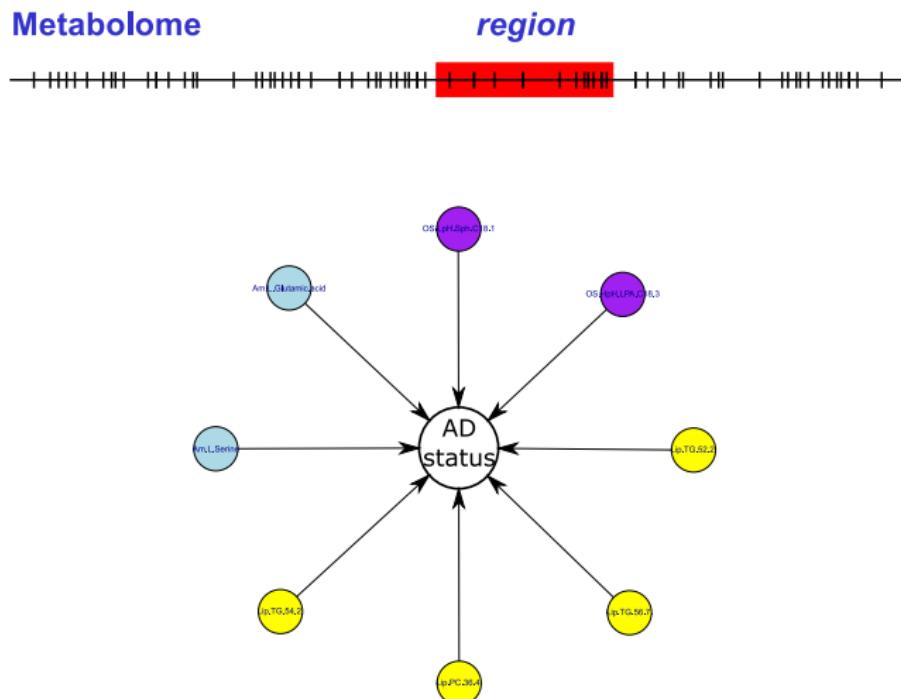
# Differential Expression



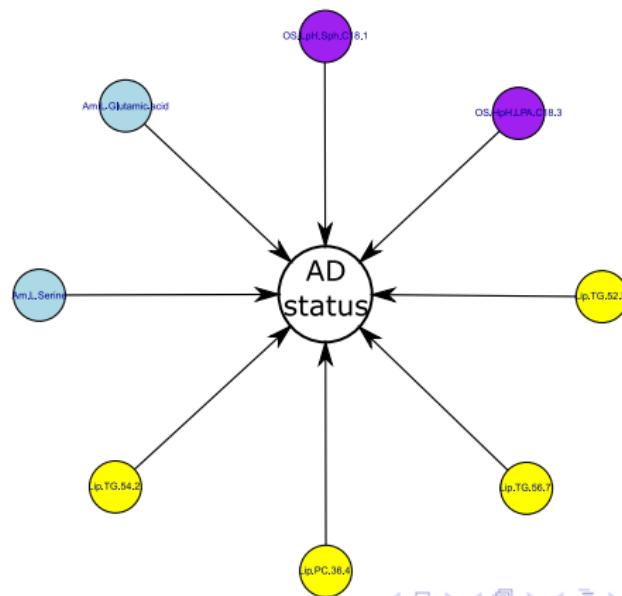
# Not the complete picture



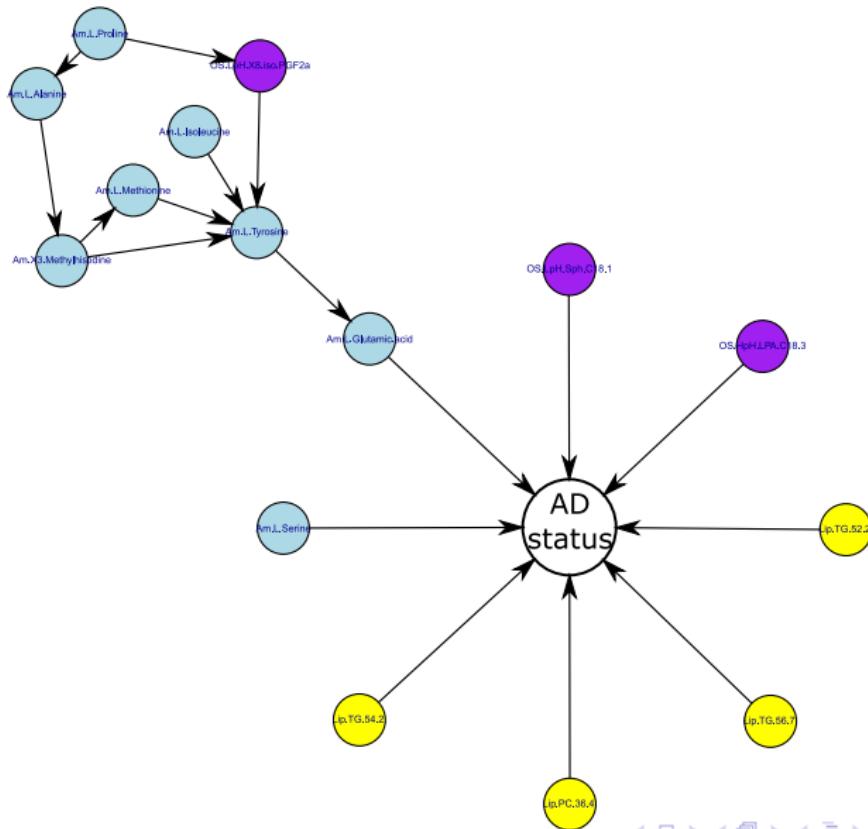
# Classification



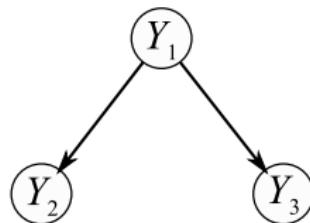
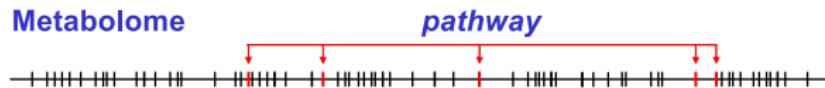
# Not the complete picture



# Not the complete picture



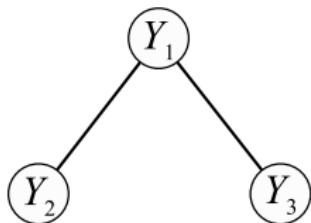
## Regulatory



$$\text{cor}(Y_1, Y_2 | Y_3) \neq 0$$

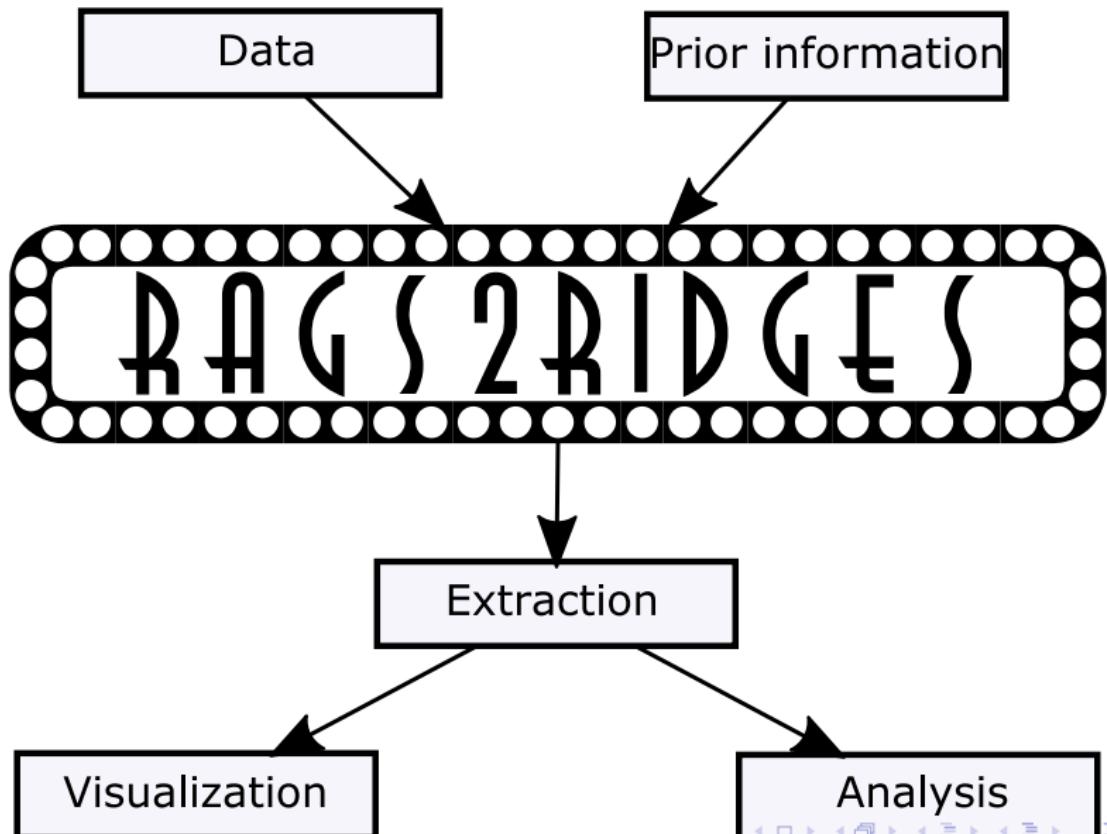
$$\text{cor}(Y_1, Y_3 | Y_2) \neq 0$$

$$\text{cor}(Y_2, Y_3 | Y_1) = 0$$

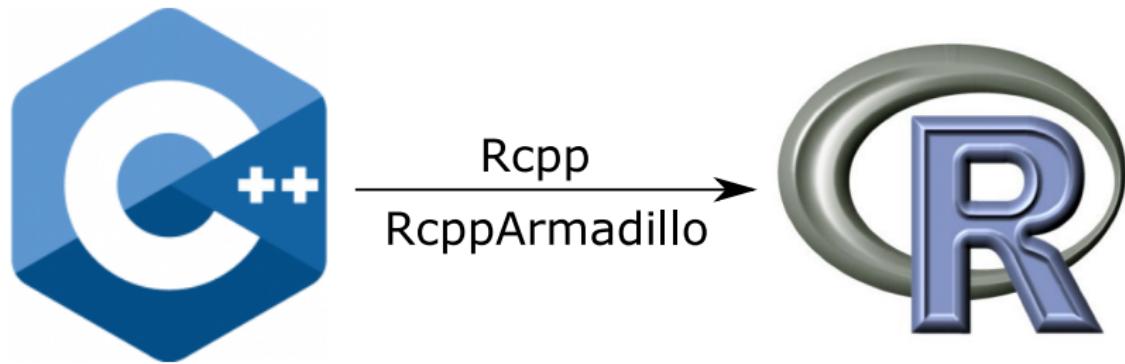




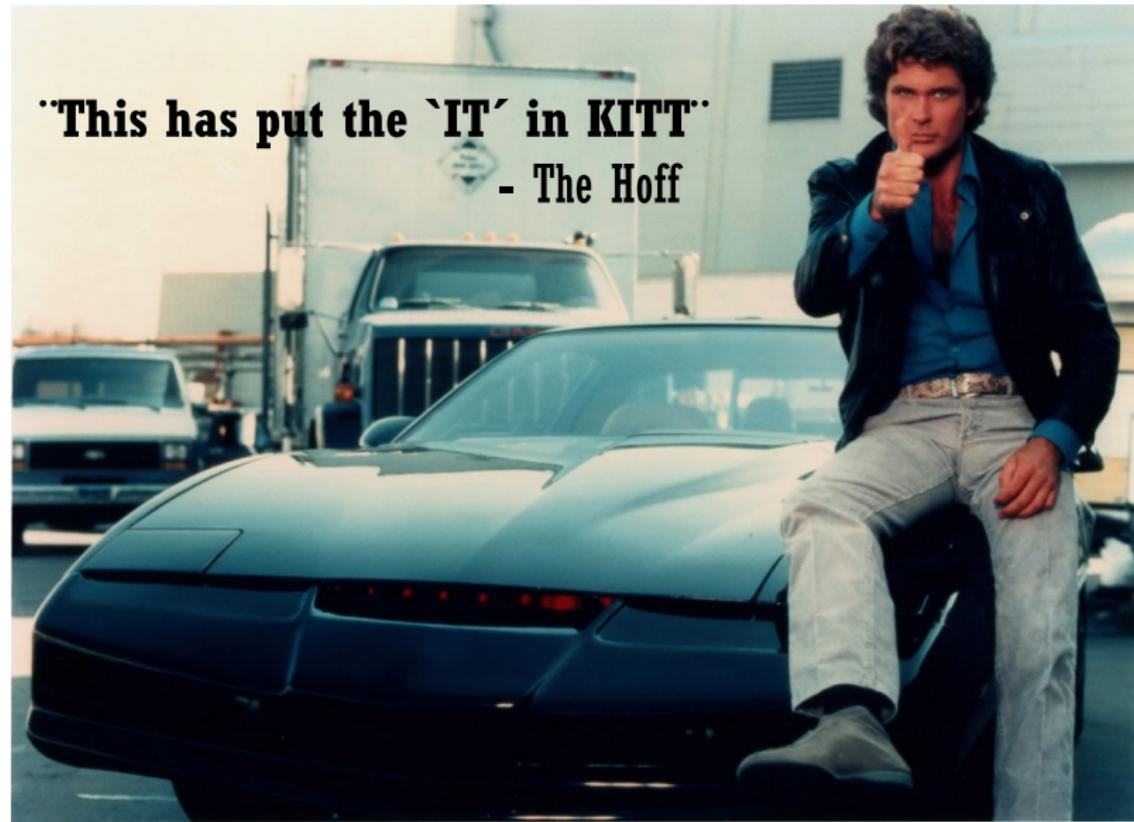
# One-Stop-Go



# C++ core



## The Hoff Approves



# Exercise Overview

## Segment I: Single Network

Extract: 2-4

Visualize: 5

Analyze: 6-8

## Segment 2: Multiple Networks

Extract: 9-11

Visualize: 12

Analyze: 13-15

# Explaining the Inverse

## The scalar inverse

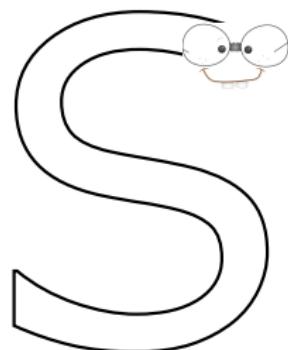
- Let  $a$  denote a number (excluding 0)
- The inverse is then the number  $b$  such that  $a \times b = 1$
- Clearly,  $b = \frac{1}{a}$

## Matrix

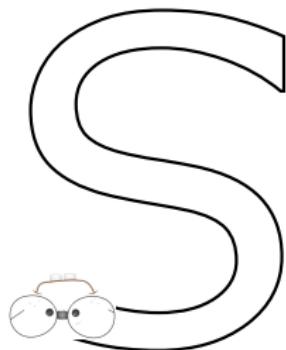
A matrix is a generalization of a number, an array of numbers

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pp} \end{bmatrix}$$

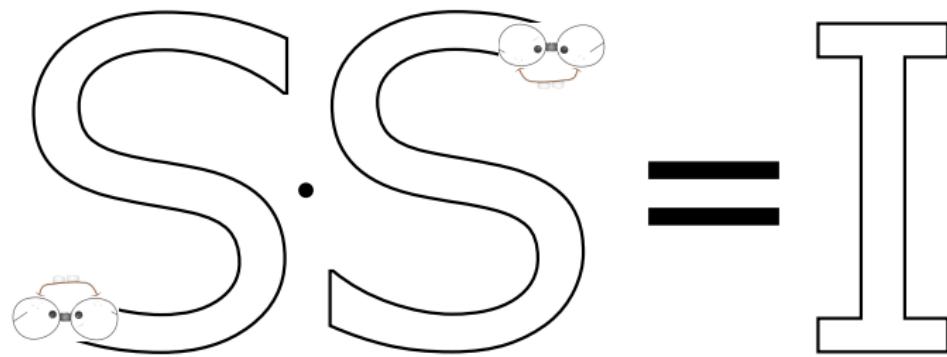
# Explaining the Inverse



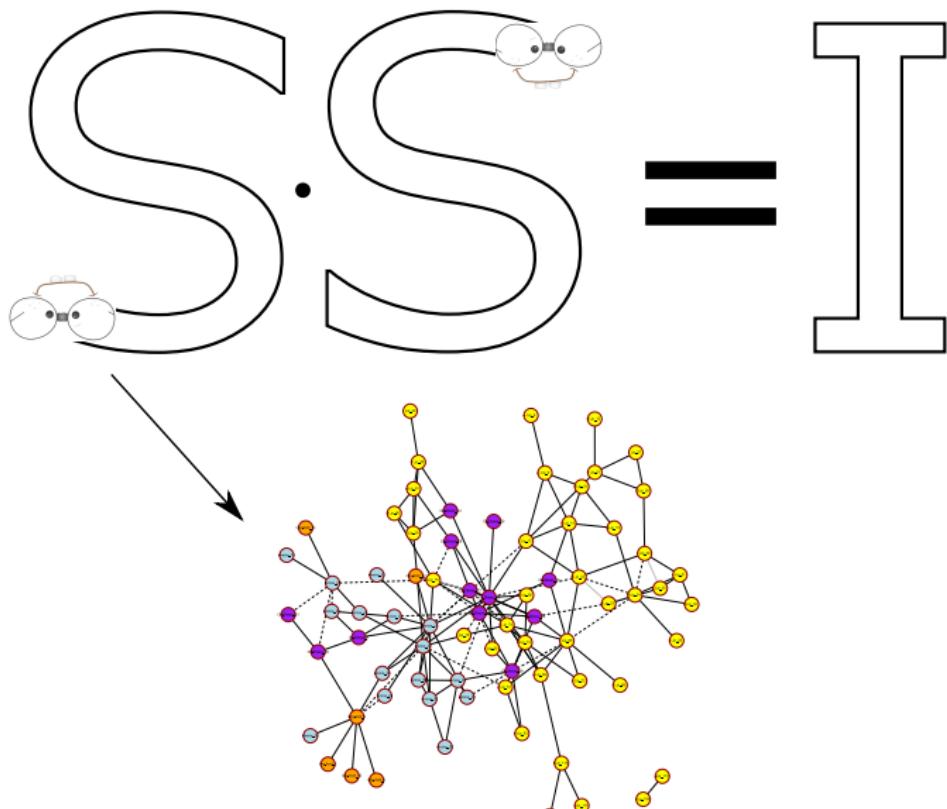
# Explaining the Inverse



# Explaining the Inverse



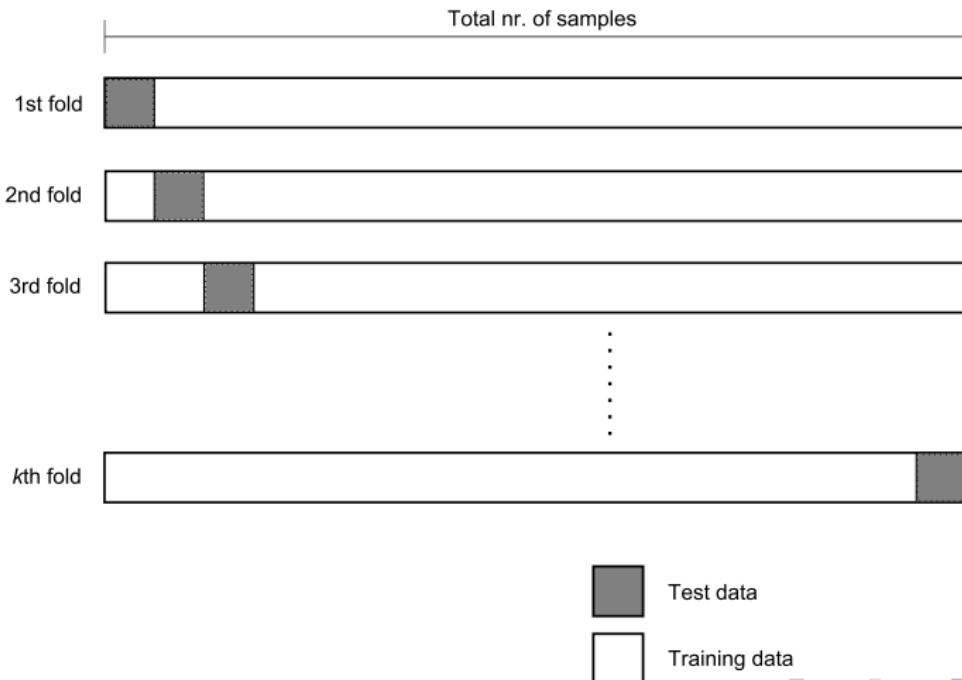
# Explaining the Inverse



# Choosing the penalty value

## K-fold cross-validation (CV)

### Single iteration of K-fold CV



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

Subset ADmetabolites with information from sampleInfo

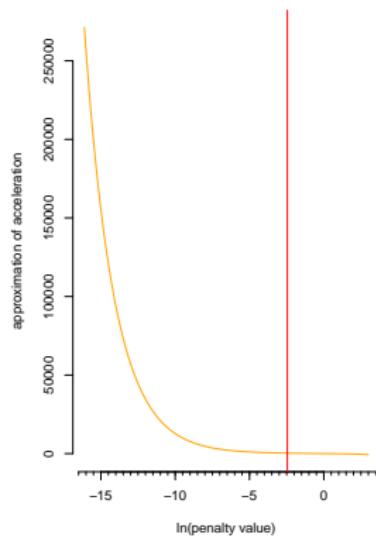
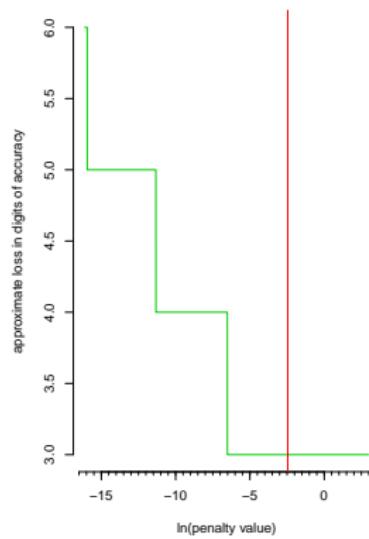
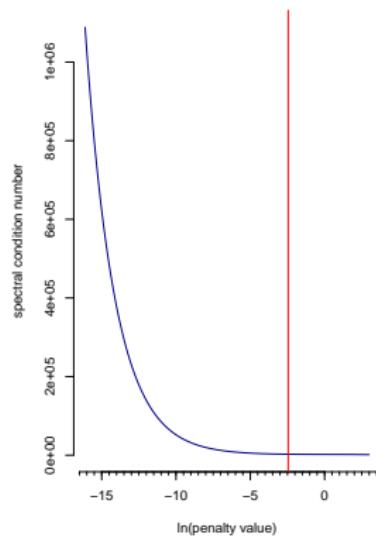
ADmetabolites[, XXX]

```
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

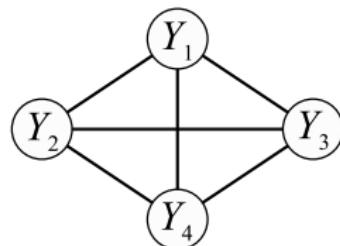


## Exercise 3: Assess the conditioning of the optimal precision matrix

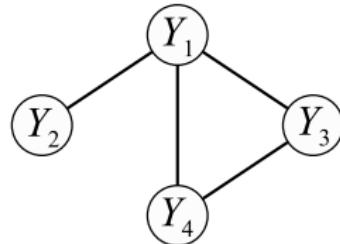
```
CNplot(S,           ← covariance matrix data  
       lambdaMin,   ← min. λ  
       lambdaMax,   ← max. λ  
       step,        ← coarseness grid  
       target,      ← T  
       vertical,    ← logical  
       value)
```

# Support Determination

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

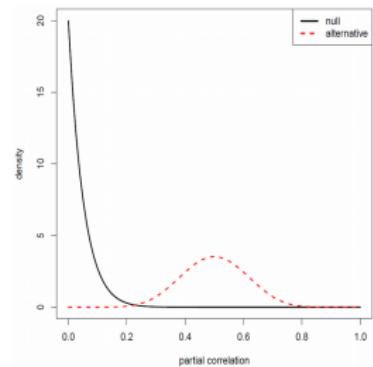


$$\begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \omega_{14} \\ \omega_{21} & \omega_{22} & 0 & 0 \\ \omega_{31} & 0 & \omega_{33} & \omega_{34} \\ \omega_{41} & 0 & \omega_{43} & \omega_{44} \end{bmatrix}$$



# Support Determination

$$\begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \omega_{14} \\ \omega_{21} & \omega_{22} & \omega_{23} & \omega_{24} \\ \omega_{31} & \omega_{32} & \omega_{33} & \omega_{34} \\ \omega_{41} & \omega_{42} & \omega_{43} & \omega_{44} \end{bmatrix} \longrightarrow \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \omega_{14} \\ \omega_{21} & \omega_{22} & 0 & 0 \\ \omega_{31} & 0 & \omega_{33} & \omega_{34} \\ \omega_{41} & 0 & \omega_{43} & \omega_{44} \end{bmatrix}$$



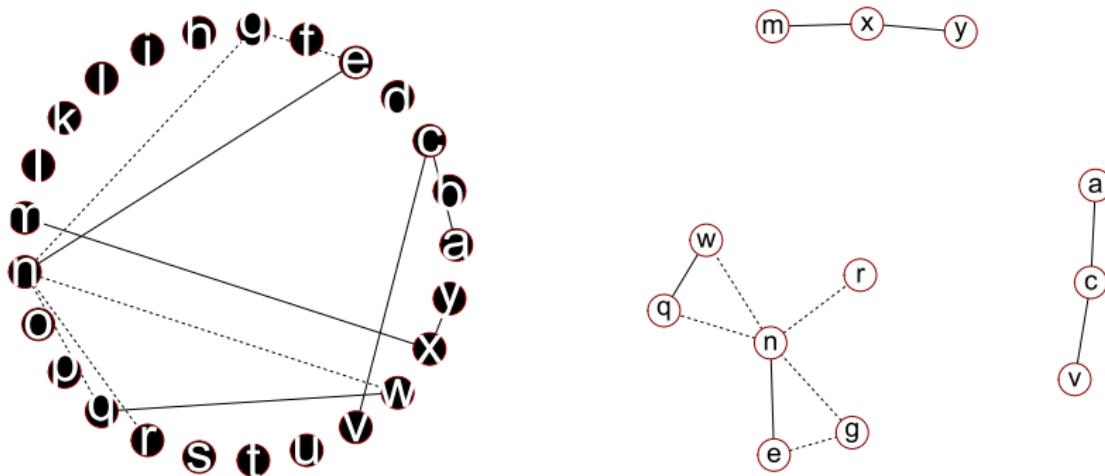
## Exercise 4: Extract a network from the optimal precision matrix by retaining elements whose posterior probability of being present $\geq .999$

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

Returns list object

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

# Visualization



## Things to Consider

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

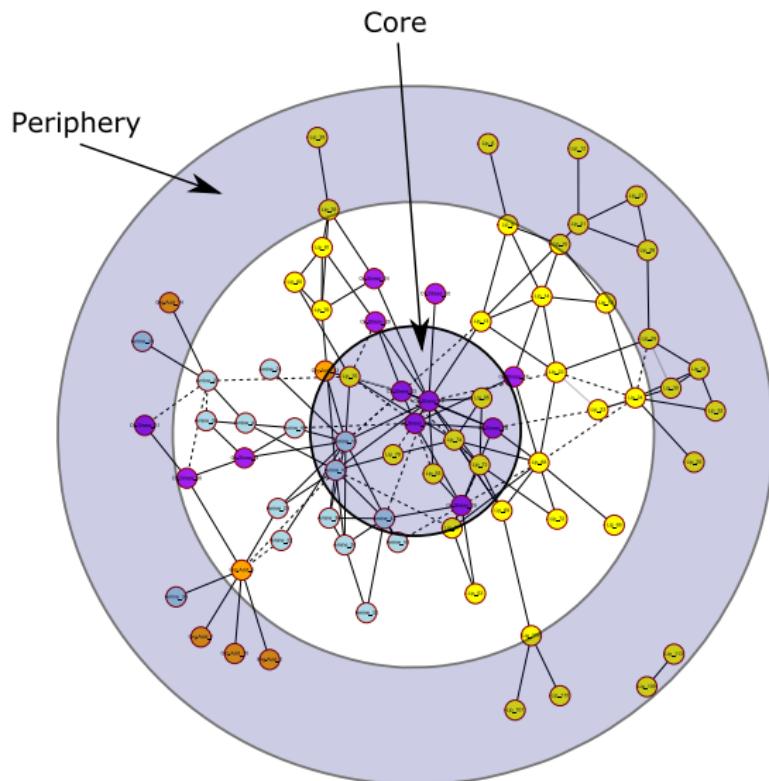
## Exercise 5: Visualize the extracted network

```
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?  
    ...)
```

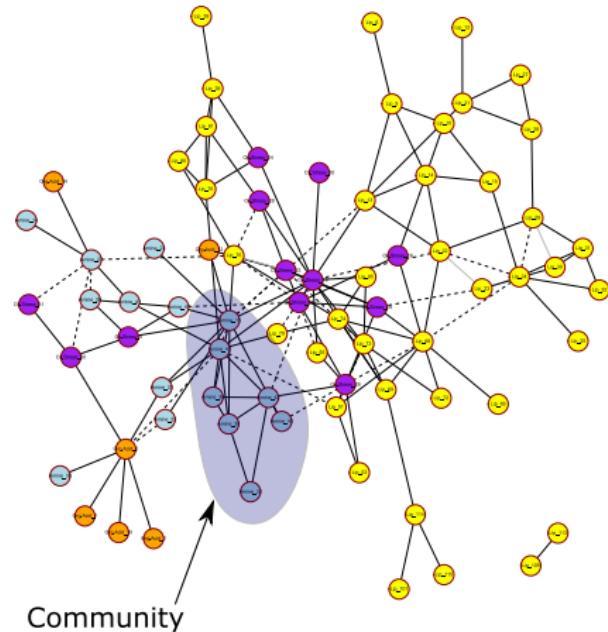
Returns matrix object

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

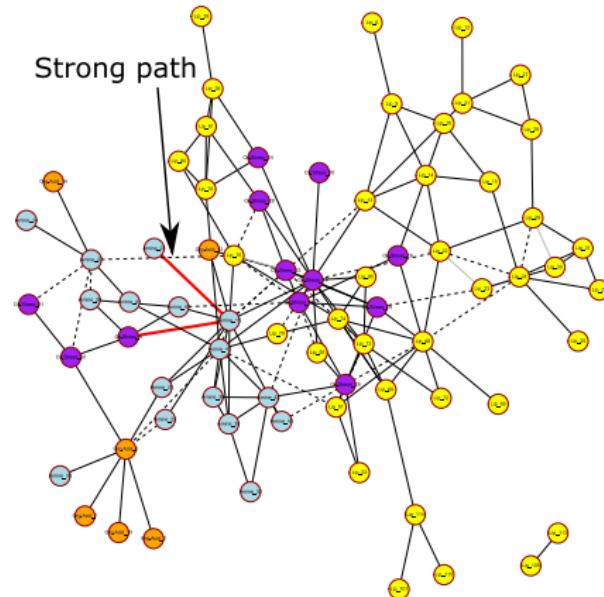
## Analysis: Global Level



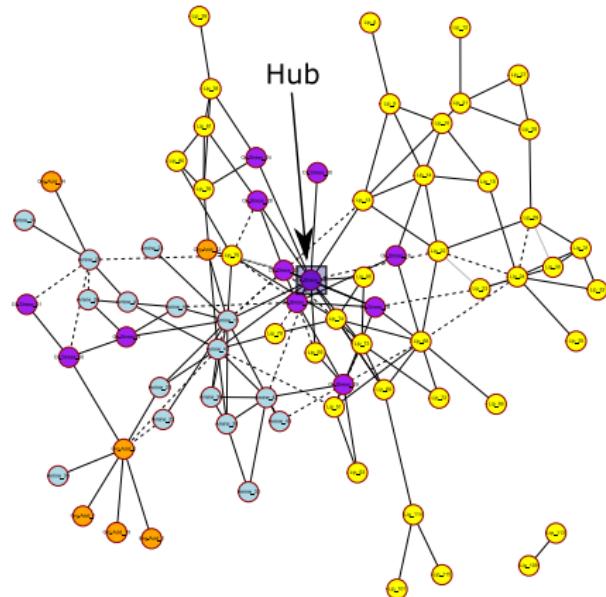
## Analysis: Group Level



## Analysis: Path Level



## Analysis: Node Level



## Node-Level Analysis: Centrality

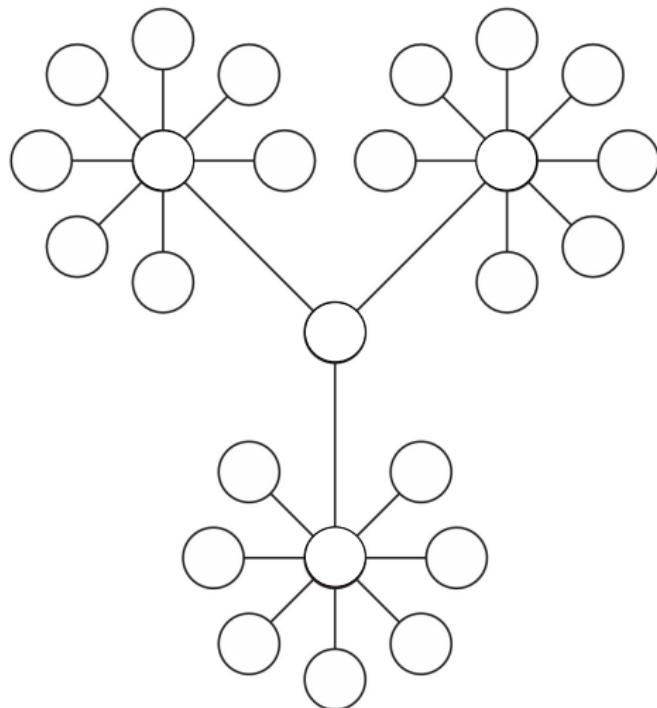


Illustration adapted from: <http://shareablespaces.blogspot.nl/2010/03/social-networking-tools.html>

## Node-Level Analysis: Degree Centrality

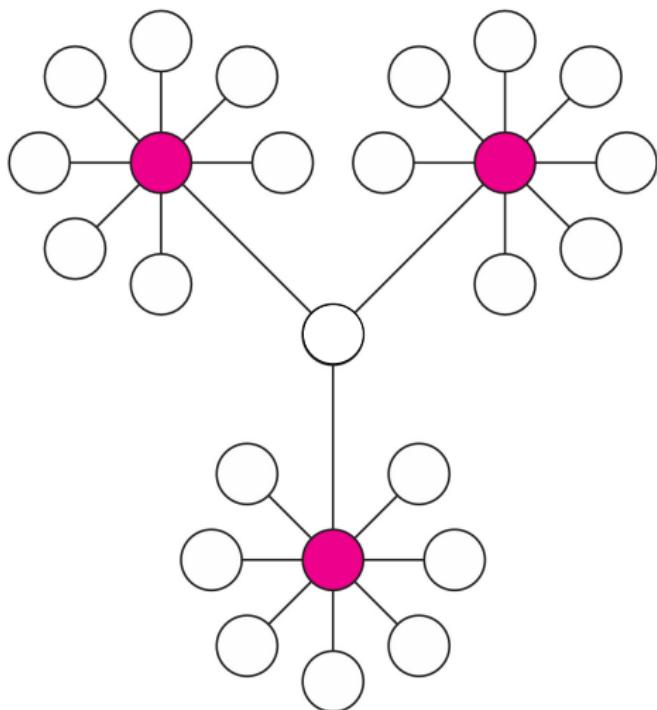


Illustration adapted from: <http://shareablesplaces.blogspot.nl/2010/03/social-networking-tools.html>

## Node-Level Analysis: Betweenness Centrality

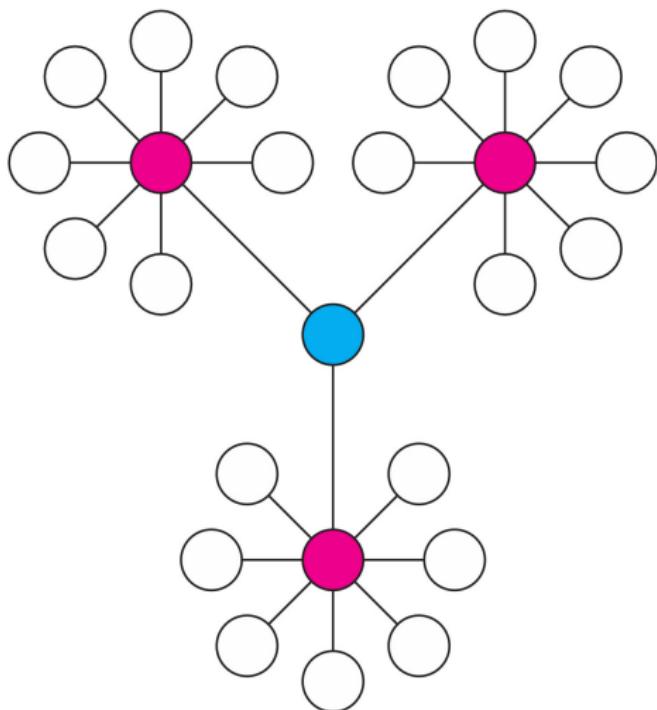


Illustration adapted from: <http://shareablespace.blogspot.nl/2010/03/social-networking-tools.html>

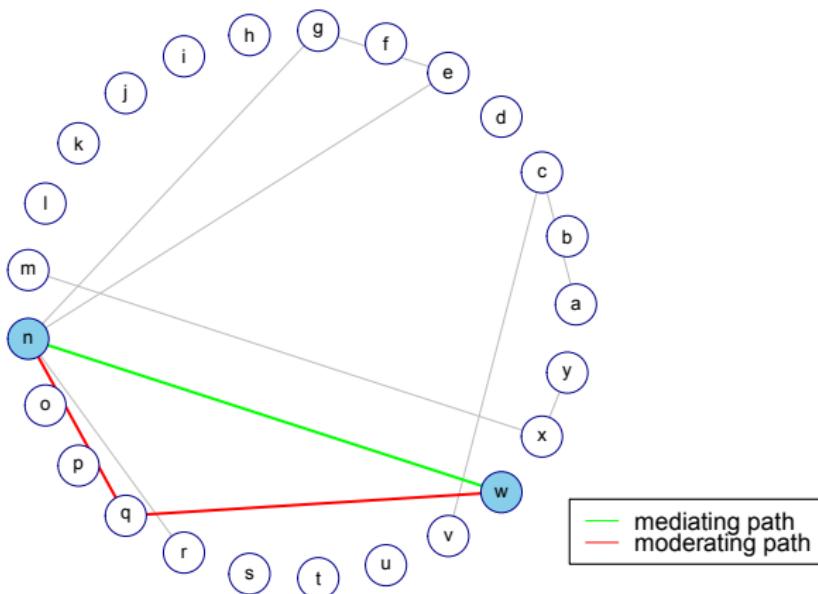
## Exercise 6: Find the nodes with the top degree and betweenness centralities



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

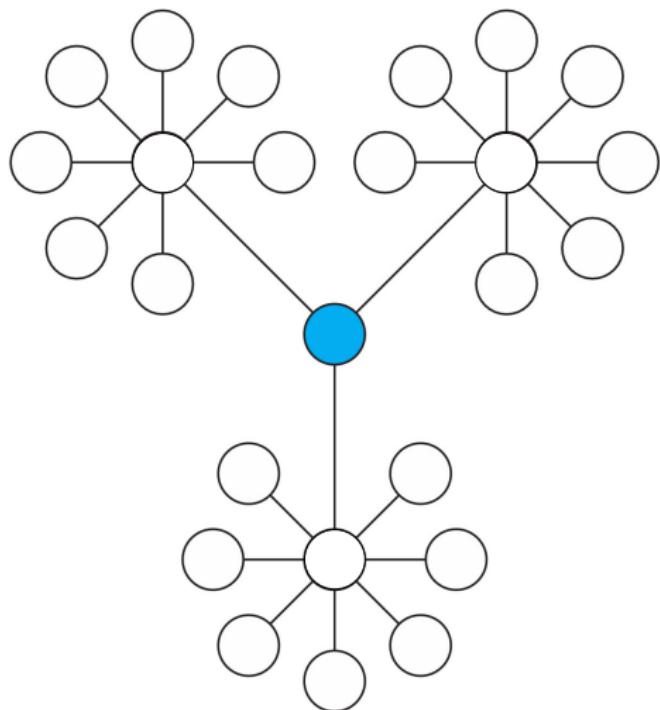


Illustration adapted from: <http://shareablesplaces.blogspot.nl/2010/03/social-networking-tools.html>

## Group-Level Analysis: Finding Communities

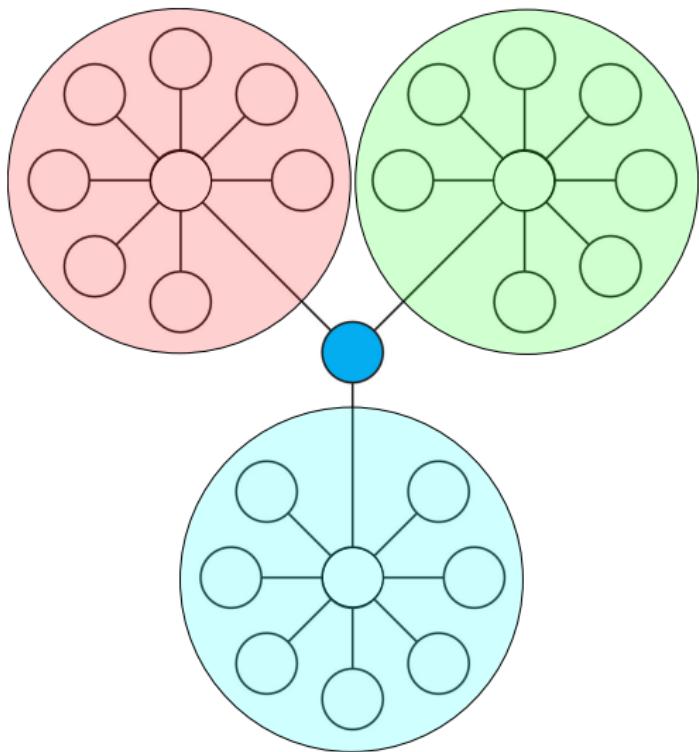


Illustration adapted from: <http://shareablesplaces.blogspot.nl/2010/03/social-networking-tools.html>

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

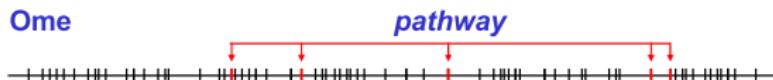
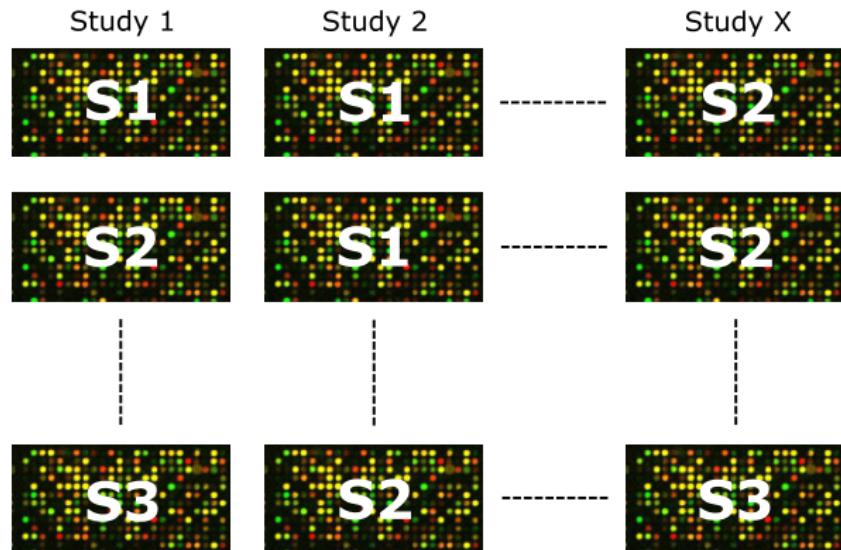
```
Communities(P,           ← sparse matrix  
          graph,       ← logical, if TRUE, then graph is given  
          lay,  
          Vsize,  
          Vcex,  
          Vcolor,  
          VBcolor,  
          VLcolor,  
          main)
```

arguments Ugraph

Returns list object

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

## Multiple data classes



## Fused Module: Functional Analogues

Core	Fused
<code>optPenalty</code>	<code>optPenalty.fused</code>
<code>sparsify</code>	<code>sparsify.fused</code>
<code>GGMnetworkStats</code>	<code>GGMnetworkStats.fused</code>

## 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
- ...

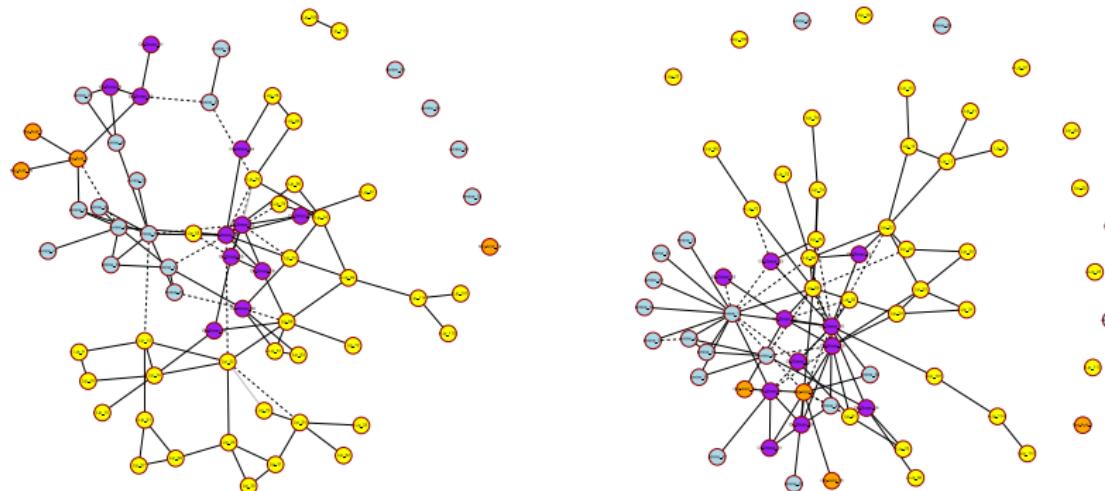
## Exercise 11: Extract class-specific networks on the basis of local FDR thresholding

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

Returns list object for each class

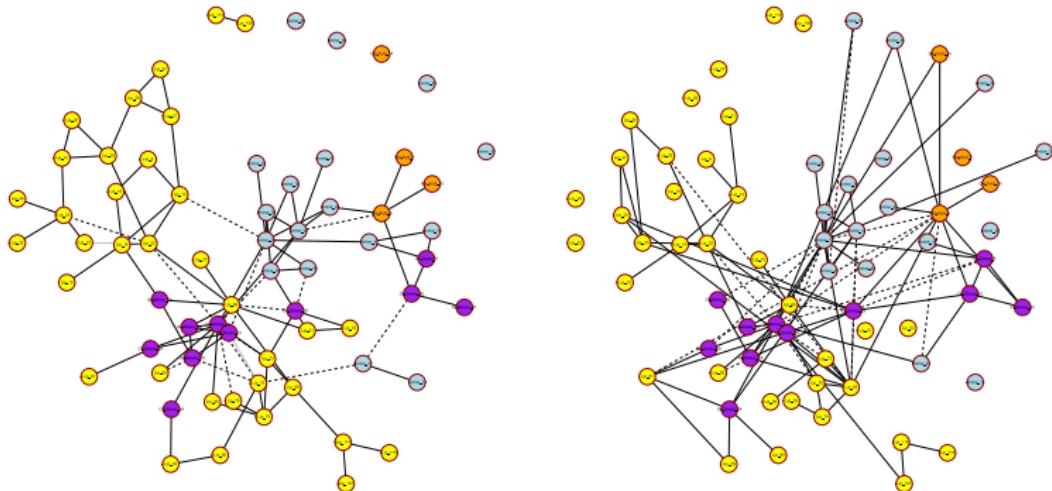
- \$sparsePrecision: Sparsified precision matrix
- \$sparseParCor: Sparsified partial correlation matrix

# Visualization



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

# Visualization



Coordinate refinement 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, ← if lay = NULL, then  
       Vsize,      layout according to coordinates  
       Vcex,  
       Vcolor,  
       VBcolor,  
       VLcolor,  
       prune  
       ...)
```

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

```
GGMnetworkStats.fused(Plist)
```

A list of sparse matrices

Returns `data.frame`

Names of `Plist` are prefixed to column-names

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

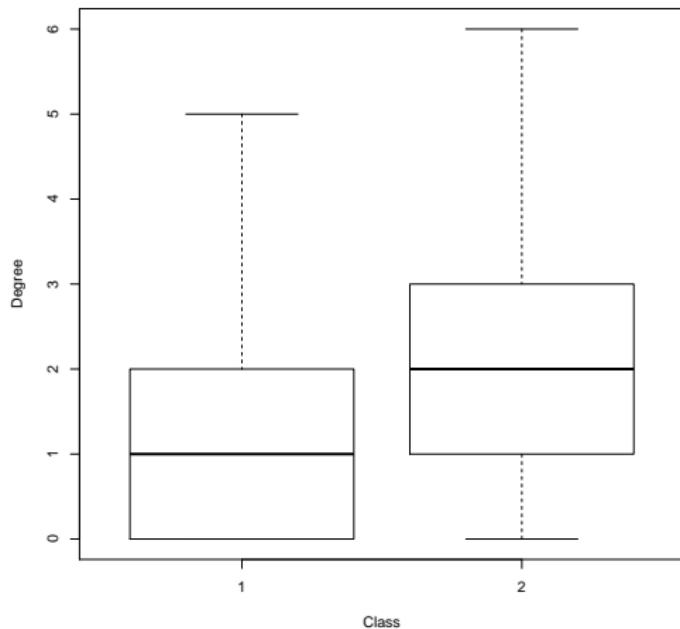
```
Communities(P,           ← sparse matrix  
          graph,       ← logical, if TRUE, then graph is given  
          lay,  
          Vsize,  
          Vcex,  
          Vcolor,  
          VBcolor,  
          VLcolor,  
          main)
```

arguments Ugraph

Returns list object

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

# Global Analysis: Degree Distributions



## Global Analysis: Entropy

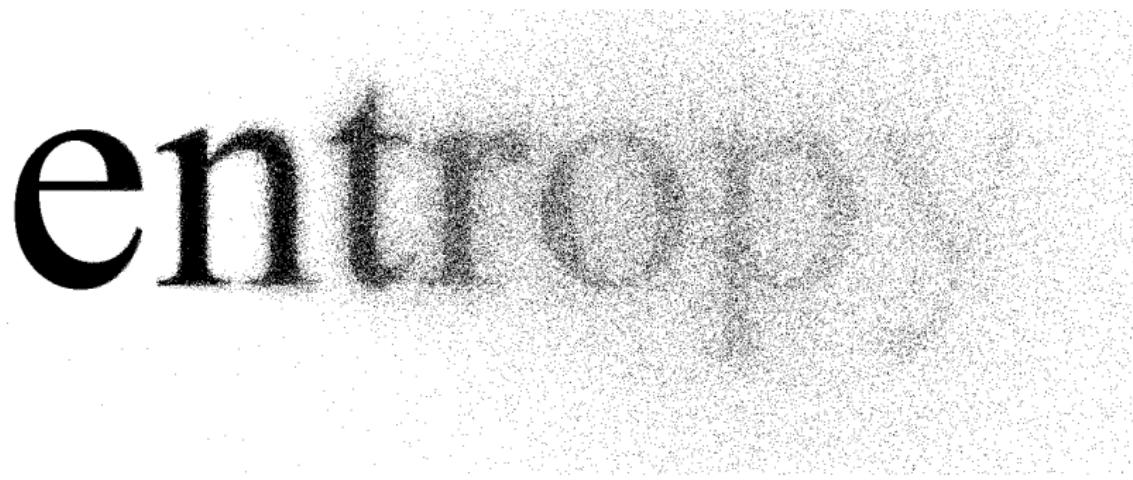


Illustration adapted from: <http://creepypasta.wikia.com/wiki/Entropy>

## 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 \[stat.ME\]](https://arxiv.org/abs/1509.07982v1).
- 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 \[stat.ME\]](https://arxiv.org/abs/1403.0904v3).

## 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 \[stat.CO\]](https://arxiv.org/abs/1608.04123v1).
- 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.