High dimensional data integration using graphical models

Claudia Angelini, Daniela De Canditiis, Anna Plaksienko

Istituto per le Applicazioni del Calcolo *Mauro Picone*

8 April 2022

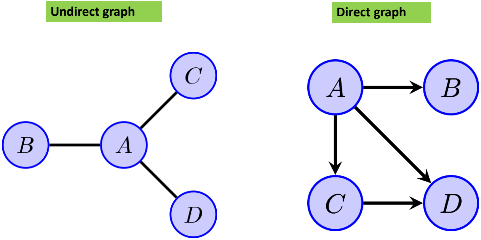
Outline

1. Introduction: **Graphical Models** and **Data Integration**.
2. Problem Definition and Algorithm: **Jewel**[1] 1.0.
3. Numerical Simulations and Real Data Applications.
4. From **Jewel 1.0** to **Jewel 2.0**: Problem Definition and Algorithm.
5. Preliminary Numerical Simulations.
6. Conclusions.

[1] Joint Node-Wise Estimation of Multiple Gaussian Graphical Models

Introduction and Motivations

- **Networks** or **Graphs** are common tools to represent relationships among variables.
- A **graph G** $= (V, E)$ can encode either **direct** or **undirect** relationships between pairs of variables.



Undirect graph

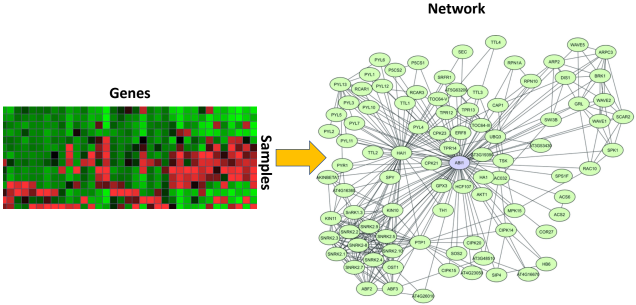
Direct graph

- **Network inference** is aimed to estimate the graph **G** from observed data **X**.

Introduction and Motivations

Gene Regulatory Networks and Protein-Protein interaction networks

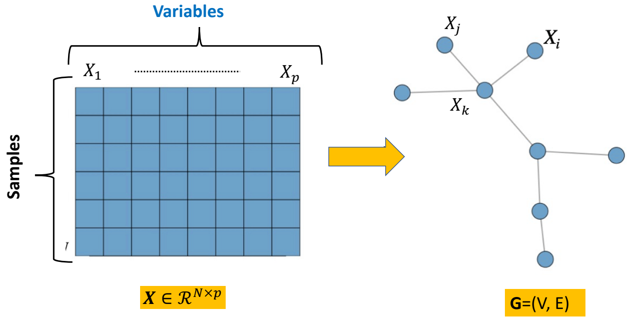
Relationships among genes or proteins can be represented as networks. However, to understand a biological system, one should distinguish direct interactions from indirect interactions (i.e., mediated by other genes).



In biological applications, we have $p >> n$, where **p** is the number of **variables** (i.e., genes) and **n** the number of **samples**.

Graphical models (Idea)

Assume to collect N samples $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T \in R^{N \times p}$ where $\mathbf{x}_i = (X_1, \ldots, X_p)$.



Graph $\mathbf{G} = (V, E)$ where vertices $V = \{1, \ldots, p\}$ correspond to **variables** $X_1, \ldots, X_p$, and edges describe the **conditional dependence** between pairs of variables.

$\Rightarrow$ The aim is estimating $\mathbf{G} = (V, E)$ from matrix $\mathbf{X}$.

Graphical models and Penalized approaches

A graph $\mathbf{G} = (V, E)$ is an ***undirected graphical model*** for the random vector $\mathbf{x}$, if it satisfies the *pairwise Markov property*:

$$\forall i, j : (i, j) \notin E, i \neq j \iff X_i \perp\!\!\!\perp X_j | X_{\{l, l \neq i, j\}}.$$

Gaussian distributions: When $\mathbf{x} = (X_1, \ldots, X_p) \sim \mathcal{N}_p(0, \mathbf{\Sigma})$

$(i, j) \in E \iff \left(\mathbf{\Sigma}^{-1}\right)_{ij} \neq 0$, entry of **precision matrix** is non-zero.
$\Longrightarrow$ Under Gaussian settings, estimating $\mathbf{G}$ is equivalent to estimating the support of $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$, where $\mathbf{\Omega}$ is the **Precision matrix**.

- *Graphical Lasso* (Friedman et al., 2008) – **penalized maximum likelihood** approach:

$$\hat{\mathbf{\Omega}} = \arg\min_{\mathbf{\Omega} \succeq 0} \left[\operatorname{tr}(\mathbf{S}\mathbf{\Omega}) - \log|\mathbf{\Omega}| + \lambda \sum_{i \neq j} |\Omega_{ij}| \right].$$

- Meinshausen and Bühlmann (2006) – **penalized regression-based** approach: define the regression coefficients $\Theta_{ij} = -\Omega_{ij}/\Omega_{jj}$ for $i \neq j$ and $\Theta_{ii} = 0$:

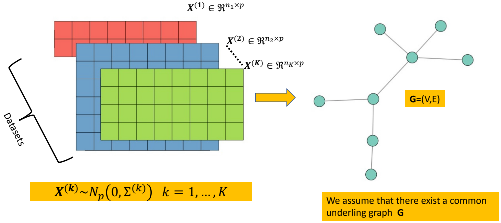
$$\hat{\mathbf{\Theta}} = \arg\min_{\substack{\mathbf{\Theta} \in \mathbb{R}^{p \times p} \\ diag = 0}} \left[\frac{1}{2N} ||\mathbf{X} - \mathbf{X}\mathbf{\Theta}||_F^2 + \lambda \sum_{i \neq j} |\Theta_{ij}| \right].$$

- Other (similar in the spirit) approaches...

Graphical models with multiple datasets

Nowadays, it is very common to collect several datasets $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(K)}$ – same (or almost the same) variables, from different sources or technologies.

- Given K datasets $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(K)}$ – same (or almost the same) variables, different sources.


$X^{(1)} \in \Re^{n_1 \times p}$
$X^{(2)} \in \Re^{n_2 \times p}$
$X^{(K)} \in \Re^{n_K \times p}$
Datasets
G=(V,E)

$X^{(k)} \sim N_p(0, \Sigma^{(k)}) \quad k = 1, \ldots, K$

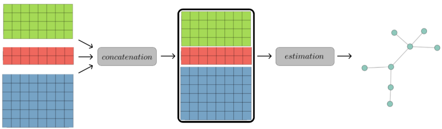
We assume that there exist a common underling graph **G**

Hypothesis

$\mathbf{X}^{(k)} = (\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_{n_k}^{(k)})^T \sim \mathcal{N}(0, \mathbf{\Sigma^{(k)}})$, $k = 1, \ldots, K$. **Different** covariance matrices $\mathbf{\Sigma^{(k)}}$, **common** underlying graph **G**.

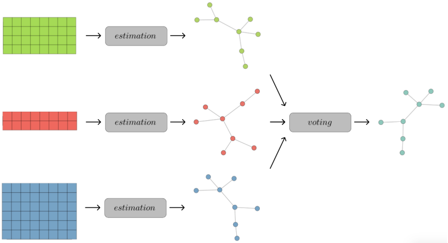
Naive approaches

Concatenation



- It works only when $p_k = p$
- Since $\Sigma^{(k)}$ can be different→ concatenation loses heterogeneity of datasets →worse performance

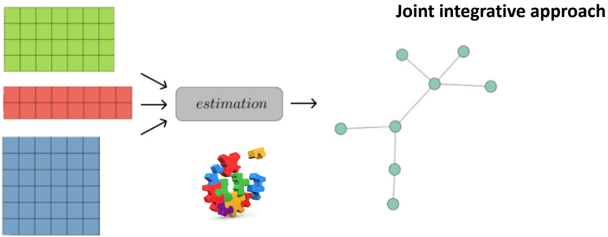
Voting



- Limited sample size n_k for some datasets → worse overall results

Joint estimation of graphical models

Ideal solution – simultaneously analyze the K datasets and **jointly** estimate the common underlying graph $\mathbf{G}$.



How to do perform joint estimation? Extend penalized maximum likelihood and regression-based approaches!

Jewel minimization problem

Given $\mathbf{X}^{(k)} = (\mathbf{x}_1^{(k)}, \dots, \mathbf{x}_{n_k}^{(k)})^T \sim \mathcal{N}(0, \mathbf{\Sigma^{(k)}})$, $k = 1, \dots, K$, denote $\mathbf{G}_k = (V_k, E_k)$ the corresponding graphical models. Define $\Theta_{ij}^{(k)} = -\Omega_{ij}^{(k)}/\Omega_{jj}^{(k)}$ and assume that

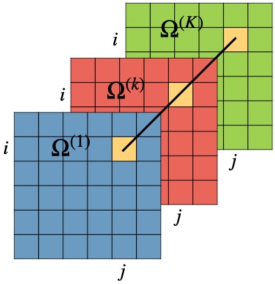
1. Most of the variables are **in common** among all K datasets;
2. $\forall\, i, j \in V_1 \cup \dots \cup V_K$ **edge** (i, j) is **either in all graphs** $\mathbf{G}_k$ or **in none of the graphs** where vertices i, j are present;
3. $\mathrm{supp}(\mathbf{\Omega}^{(k)}) = \mathrm{supp}(\mathbf{\Theta}^{(k)})$ is **symmetric**.

$$(\hat{\mathbf{\Theta}}^{(1)}, \dots, \hat{\mathbf{\Theta}}^{(K)}) = \underset{\substack{\mathbf{\Theta}^{(1)} \in \mathbb{R}^{p_1 \times p_1} \\ \vdots \\ \mathbf{\Theta}^{(K)} \in \mathbb{R}^{p_K \times p_K}, \\ diag=0}}{\arg\min} \left[\underbrace{\frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} ||\mathbf{X}^{(k)} - \mathbf{X}^{(k)}\mathbf{\Theta}^{(k)}||_F^2}_{multitask\ \ goodness\ \ of\ \ fit} + \lambda \underbrace{P(\mathbf{\Theta}^{(1)}, \dots, \mathbf{\Theta}^{(K)})}_{joint\ \ penalty} \right].$$

$\Longrightarrow$ The group penalty $P(\mathbf{\Theta^{(1)}}, \dots, \mathbf{\Theta^{(K)}})$ have to enforce the joint structure of $\mathbf{G}$.

The group penalty

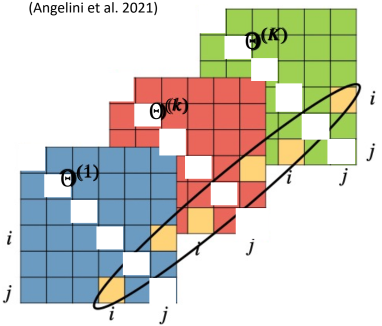
Group structure to enforce a common structure in **G**, as in JGL (Danaher et al. 2014)



$\Omega^{(k)} \in \mathcal{R}^{p\times p}$ Precision matrix

$$\Omega_{i,j} = 0 \iff (i,j) \notin E$$

Group structure to enforce simultaneously a common structure in **G** and the **symmetry** as in Jewel 1.0 (Angelini et al. 2021)



$$\Theta^{(k)}: \Theta_{i,j} = \begin{cases} \frac{\Omega_{i,j}}{\Omega_{j,j}} & i \neq j \\ 0 & i = j \end{cases}$$

$$\textbf{Group } (i,j) \quad : \quad (\Theta_{ij}^{(k)}, \Theta_{ji}^{(k)})_{k:\{X_i,X_j\}\subseteq V_k}$$

$$g_{ij} = \mathrm{card}((\Theta_{ij}^{(k)}, \Theta_{ji}^{(k)})_{k:\{X_i,X_j\}\subseteq V_k})$$

$$\text{If } p_k = p \Longrightarrow g_{i,j} = 2K$$

Jewel minimization problem

$$(\hat{\mathbf{\Theta}}^{(1)},\dots,\hat{\mathbf{\Theta}}^{(K)}) = \underset{\substack{\mathbf{\Theta}^{(1)}\in\mathbb{R}^{p_1\times p_1}\\ \vdots \\ \mathbf{\Theta}^{(K)}\in\mathbb{R}^{p_K\times p_K},\\ diag=0}}{\arg\min} \left[\underbrace{\frac{1}{2}\sum_{k=1}^{K} \frac{1}{n_k}||\mathbf{X}^{(k)}-\mathbf{X}^{(k)}\mathbf{\Theta}^{(k)}||_F^2}_{multitask\ \ goodness\ \ of\ \ fit} + \lambda \underbrace{P(\mathbf{\Theta}^{(1)},\dots,\mathbf{\Theta}^{(K)})}_{joint\ \ penalty}\right].$$

In ***jewel*** we propose **symmetric group lasso penalty**[2]

$$P(\hat{\mathbf{\Theta}}^{(1)},\dots,\hat{\mathbf{\Theta}}^{(K)}) = \sum_{i<j=1}^{p} \sqrt{g_{ij}} \sqrt{\sum_{k:\{X_i,X_j\}\subseteq V_k} \left(\Theta_{ij}^{(k)}\right)^2 + \left(\Theta_{ji}^{(k)}\right)^2}.$$

where g_{ij} is the **size of the group** $(\Theta_{ij}^{(k)},\Theta_{ji}^{(k)})_{k:\{X_i,X_j\}\subseteq V_k}$.

$(\Theta_{ij}^{(k)},\Theta_{ji}^{(k)})_{k:\{X_i,X_j\}\subseteq V_k} = 0 \Longrightarrow (i,j)$ and $(j,i) \notin E$.

[2] Inspired by Friedman et al. (2010) and De Canditiis and Guardasole (2018)

Jewel numerical algorithm (using *group descent algorithm*)

Idea of group descent algorithm[3]: Fix all the variables except one group and update it. Iterate until convergence.

1. For each i, j **evaluate** $\mathbf{z} = (z_{ij}^{(1)}, z_{ji}^{(1)}, \dots, z_{ij}^{(K)}, z_{ji}^{(K)}) \in \mathbb{R}^{2K}$ with entries

$$z_{ij}^{(k)} = -\frac{1}{n_k} X_{.i}^{(k)\,T} \left(X_{.j}^{(k)} - \sum_{m \neq i, j} X_{.m}^{(k)} \Theta_{mj}^{(k)} \right)$$

$$z_{ji}^{(k)} = -\frac{1}{n_k} X_{.j}^{(k)\,T} \left(X_{.i}^{(k)} - \sum_{m \neq i, j} X_{.m}^{(k)} \Theta_{mi}^{(k)} \right)$$

2. **Update** regression coefficients with multivariate soft-threshold operator

$$(\hat{\Theta}_{ij}^{(1)}, \hat{\Theta}_{ji}^{(1)}, \dots, \hat{\Theta}_{ij}^{(K)}, \hat{\Theta}_{ji}^{(K)}) = \left(1 - \frac{\lambda}{||\mathbf{z}||}\right)_{+} \mathbf{z}.$$

3. **Repeat** for all i, j until convergence.

[3] Inspired by Breheny and Huang (2015)

Jewel implementation suggestions

1. **Active shooting**: Divide variables into Active and Non-Active. *jewel* updates only active variables. If $(\hat{\Theta}^{(1)}_{ij}, \hat{\Theta}^{(1)}_{ji}, \ldots, \hat{\Theta}^{(K)}_{ij}, \hat{\Theta}^{(K)}_{ji}) \neq 0 \Rightarrow$ pair i, j is "Active". If it is 0 $\Rightarrow$ the pair is "Non-Active".

$$\textbf{Active} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

   Once group is not active, we **do not update it anymore**. Active shooting provides advantage in runtime.

2. **Randomization**: instead of cycling $j = 1, .., p$, we update the variables in a random order. Empirically, provides advantage in running time for fixed value of λ.

3. **Warm-start**: Fix the grid $\lambda_1 < \lambda_2 \cdots < \lambda_L$, then use $\hat{\mathbf{\Theta}}^{(1)}, \ldots, \hat{\mathbf{\Theta}}^{(K)}$, obtained with λ_l to initialize problem with λ_{l+1}.

Theoretical properties: Variable selection consistency

$$\mathbf{X} = blkdiag\left(blkdiag\left(\mathbf{X}^{(k)}_{\cdot,-j}\right)_{j=1\dots p}\right)_{k=1\dots K}$$

$$\mathbf{D} = blkdiag\left(\frac{1}{\sqrt{n_k}}\mathbf{I}_{n_k p}\right)_{k=1\dots K} \quad \text{and } \mathbf{C} = \mathbf{X}^T\mathbf{D}^2\mathbf{X}.$$

S – set of active pairs i, j.

jewel minimization problem

$$\Downarrow$$

$$\min_{\theta\in\mathbb{R}^{p(p-1)K}}\left\{\frac{1}{2}||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}||^2_{\mathbf{D}} + \lambda\sum_{i<j=1}^{p}||\boldsymbol{\theta}_{[ij]}||\right\}.$$

Variable selection consistency

Suppose $\exists\ \delta > 0$ such that, with probability at least $1 - e^{-\delta/N}$, one has

1. $\mathbf{C}_{SS}$ is invertible.
2. (Irrepresentable condition): $\exists\alpha \in (0,1) : \forall(i,j) \in S^c$
    1. $\left|\left|[\mathbf{C}_{S^cS}\mathbf{C}^{-1}_{SS}\tau]_{ij}\right|\right| \leq \alpha\ \forall\tau \in \mathbb{R}^{2Ks} : \max_{(i,j)\in S}||\tau_{[ij]}||_2 \leq 1$
    2. $\lambda \geq \frac{2}{1-\alpha}||W_{[ij]}||$
3. (Signal strength): $\forall(i,j) \in S$ it holds

$$\lambda < \left\{||\theta^0_{[ij]}||_2 - ||V_{[ij]}||\right\}\left|\left|\left[\mathbf{C}^{-1}_{SS}\tau\right]_{[ij]}\right|\right|^{-1} \quad \forall\tau \in \mathbb{R}^{2Ks} : \max_{(i,j)\in S}||\tau_{[ij]}||_2 \leq 1$$

$$\Downarrow$$

$$\mathbb{P}(\hat{E} = E) \geq 1 - e^{-\delta/N},$$

where E is the true edge set and $\hat{E}$ is the estimated edge set.

Theorem conditions are *similar* to those in *Basu, Shojaie and Michailidis* (2015).

However, it is hard to verify them in practice, since they depend on some unknown quantities

The choice of the regularization parameter λ

Fix the grid $\lambda_1 < \lambda_2 \cdots < \lambda_L$.

- **Bayesian information criterion**: for each λ_l evaluate

$$BIC(\lambda_l) = \sum_{k=1}^{K} n_k \sum_{i=1}^{p} \log \left|\left| R_{.i}^{(k)} \right|\right|^2 + \#\{Active_{ij}(\lambda_l) \neq 0\} \sum_{k=1}^{K} \log n_k$$

and $\lambda_{BIC} = \underset{\lambda_l, l=1\ldots L}{\arg\min} BIC(\lambda_l)$.

- **Cross-validation**: split the data into F folds $\Rightarrow$ for each fold and each λ_l obtain $\hat{\mathbf{\Theta}}_{-f}^{(k)}(\lambda_l)$, $k = 1, \ldots, K$ with training data $\Rightarrow$ evaluate its error with testing data

$$err(f, l) = \sum_{k=1}^{K} \frac{1}{n_k^f} \left|\left| \mathbf{X}_f^{(k)} - \mathbf{X}_f^{(k)} \hat{\mathbf{\Theta}}_{-f}^{(k)}(\lambda_l) \right|\right|_F^2 .$$

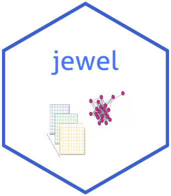
Combine errors over folds $CV(\lambda_l) = \dfrac{1}{F} \sum_{f=1}^{F} err(f, l)$ and

$$\lambda_{CV} = \underset{\lambda_l,\, l=1\ldots L}{\arg\min} CV(\lambda_l)$$

$\Longrightarrow$ Can use **warm start** for BIC and CV.

Jewel R package

jewel 1.0 method is implemented in the R package `jewel`, open-source and freely available at https://github.com/annaplaksienko/jewel.



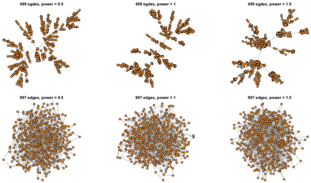
`jewel 1.0` R package contains:

- Function for *jewel* method;
- Functions for estimating λ with BIC and CV
- Data generation function for simulation
- Function for performance comparison (TP, TN, FP, FN evaluation).

The package exploits **R parallel** functions to speed-up the calculation.

Simulations scheme

- We generated simulated datasets $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(K)}$ encoding a common graph $\mathbf{G}$ with different number of samples $\mathbf{n}$, different number of variables $\mathbf{p}$ and including a different number of data matrices $\mathbf{K}$
- We considered several structures for modeling the graph $\mathbf{G}$ as Scale-free graph (Barabasi algorithm)



We assessed

- Performance when increasing the number of data matrices $\mathbf{K}$.
- Advantages of the *jewel* joint approach over the naive approaches.
- Influence of various parameters.
- Performance of *jewel* compared to other existing methods (JGL and Guo et al.)

Measures of performance

- Since different methods use λ with different normalizations $\Rightarrow$ we compared them using the **ROC-curve**, i.e. FPR vs TPR evaluated for various λ values.

  $$TPR = \frac{TP}{TP + FN} \qquad \text{and} \qquad FPR = \frac{FP}{FP + TN}.$$

  We used a grid of 50 λ (uniform in log scale) from 0.01 to 1.

- Running time = elapsed time in seconds on 4-core 3.6GHz processor and 16GB RAM computer.

- TPR, FPR and running time reported on average over **20 runs**.

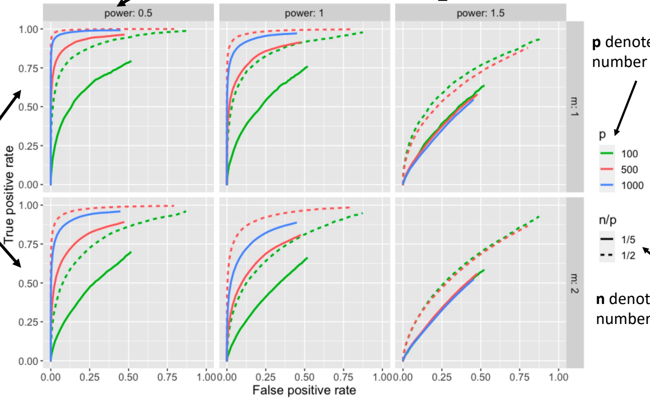
Reference

Angelini, C., De Canditiis, D., Plaksienko, A. *Jewel: A Novel Method for Joint Estimation of Gaussian Graphical Models*. Mathematics 2021, 9, 2105. https://doi.org/10.3390/math9172105

Selection of Simulations Results

- Here **K=3**
- **p>n** (high dimensional regime)

Parameter power tunes the hubs in **G**



Parameter m controls the sparsity of the true graph **G** as # edges mp-(2m-1)

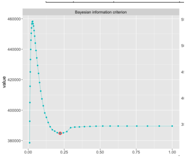
p denotes the number of variables

n denotes the number of samples

Real data applications: Glioblastoma microarray gene expression data from GEO.

K = 3 datasets (Affymetrix Array GSE4290 and GSE7696, Agilent Microarray GSE22866);

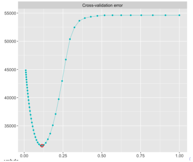
$n_1 = 40 \quad n_2 = 77 \quad n_3 = 80$ and $\mathbf{p = 483}$ genes common to 7 pathways.

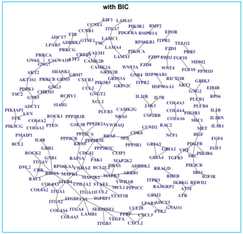


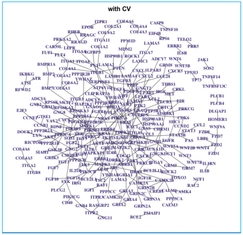
	λ_{OPT}	# est. edges
BIC	0.2223	3113/116403
CV	0.1151	7272/116403

Comparison with

STRING







Preliminary Conclusions

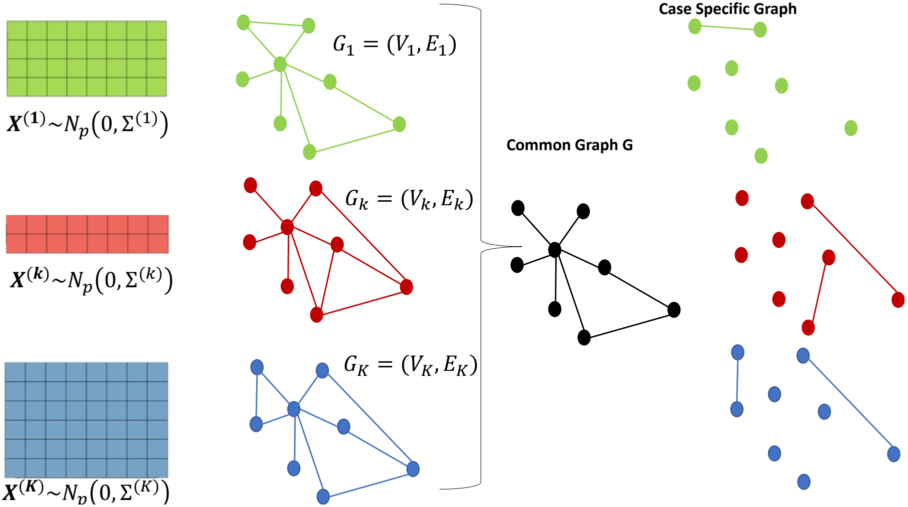
- *Jewel 1.0* is a novel method to jointly estimate $\mathbf{G}$ given several datasets.
- It has nice theoretical properties, open-source R package, and includes data driven methods for estimating the regularization parameter.
- Simulations and Real Data analysis showed competitive performance of *Jewel 1.0* compared to other methods.

However,

- All methods, including *Jewel*, returned a relatively large number of false positives $\Longrightarrow$ there is still space for improving.
- All methods, including *Jewel*, showed a lack of performance when the graph $\mathbf{G}$ contains hubs $\Longrightarrow$ there is still space for improving.
- Need to extend to the model allowing class-specific differences in $\mathbf{G}_1, \mathbf{G}_2, \ldots \mathbf{G}_K \Longrightarrow$ *Jewel 2.0*.

Jewel 2.0 is our novel proposal to address the above mentioned limits.

Jewel 2.0 general idea: Common and Specific part



$\boldsymbol{X}^{(\mathbf{1})} \sim N_p\left(0, \Sigma^{(1)}\right)$

$G_1 = (V_1, E_1)$

$\boldsymbol{X}^{(\boldsymbol{k})} \sim N_p\left(0, \Sigma^{(k)}\right)$

$G_k = (V_k, E_k)$

$\boldsymbol{X}^{(\boldsymbol{K})} \sim N_p\left(0, \Sigma^{(K)}\right)$

$G_K = (V_K, E_K)$

Common Graph G

Case Specific Graph

The Jewel 2.0 minimization problem

For $k = 1 \dots K$ we set

$$\boldsymbol{\Theta}^{(\boldsymbol{k})} = \boldsymbol{\Xi}^{(\boldsymbol{k})} + \boldsymbol{\Gamma}^{(\boldsymbol{k})},$$

where matrices $\boldsymbol{\Xi}^{(\boldsymbol{k})}$ aims to capture the common part among the K classes, while $\boldsymbol{\Gamma}^{(\boldsymbol{k})}$ are class-specific.

$$(\hat{\boldsymbol{\Xi}}^{(1)}, \hat{\boldsymbol{\Gamma}}^{(1)}, \dots, \hat{\boldsymbol{\Xi}}^{(K)}, \hat{\boldsymbol{\Gamma}}^{(K)}) = \underset{\substack{\boldsymbol{\Xi}^{(\mathbf{1})}, \boldsymbol{\Gamma}^{(\mathbf{1})} \in \mathbb{R}^{p_1 \times p_1} \\ \vdots \\ \boldsymbol{\Xi}^{(\boldsymbol{k})}, \boldsymbol{\Gamma}^{(\boldsymbol{k})} \in \mathbb{R}^{p_K \times p_K}, \\ diag=0}}{\arg\min} \left\{ \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} ||\mathbf{X}^{(k)} - \mathbf{X}^{(k)} \boldsymbol{\Xi}^{(\boldsymbol{k})} - \mathbf{X}^{(k)} \boldsymbol{\Gamma}^{(\boldsymbol{k})}||_F^2 + \right.$$

$$\left. + \lambda_1 P_1(\boldsymbol{\Xi}^{(\mathbf{1})}, \dots, \boldsymbol{\Xi}^{(\boldsymbol{k})}) + \lambda_2 P_2(\boldsymbol{\Gamma}^{(\mathbf{1})}, \dots, \boldsymbol{\Gamma}^{(\boldsymbol{k})}) \right\}.$$

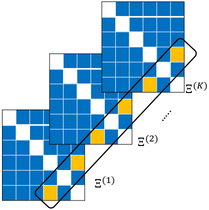
where $P_1(\boldsymbol{\Xi}^{(\mathbf{1})}, \dots, \boldsymbol{\Xi}^{(\boldsymbol{k})})$ is a penalty term for the common part, and $P_2(\boldsymbol{\Xi}^{(\mathbf{1})}, \dots, \boldsymbol{\Xi}^{(\boldsymbol{k})})$ for the class specific differences.

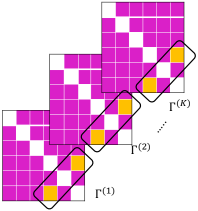
The Jewel 2.0 minimization problem

We choose

$$P_1(\mathbf{\Xi^{(1)}}, \dots, \mathbf{\Xi^{(K)}}) = \sum_{i<j=1}^{p} \sqrt{g_{ij}} \sqrt{\sum_{k:\{X_i,X_j\}\in V_k} \left(\Xi_{ij}^{(k)}\right)^2 + \left(\Xi_{ji}^{(k)}\right)^2}$$

$$P_2(\mathbf{\Gamma^{(1)}}, \dots, \mathbf{\Gamma^{(K)}}) = \sum_{i<j=1}^{p} \sqrt{2} \sum_{k:\{X_i,X_j\}\in V_k} \sqrt{\left(\Gamma_{ij}^{(k)}\right)^2 + \left(\Gamma_{ji}^{(k)}\right)^2}$$

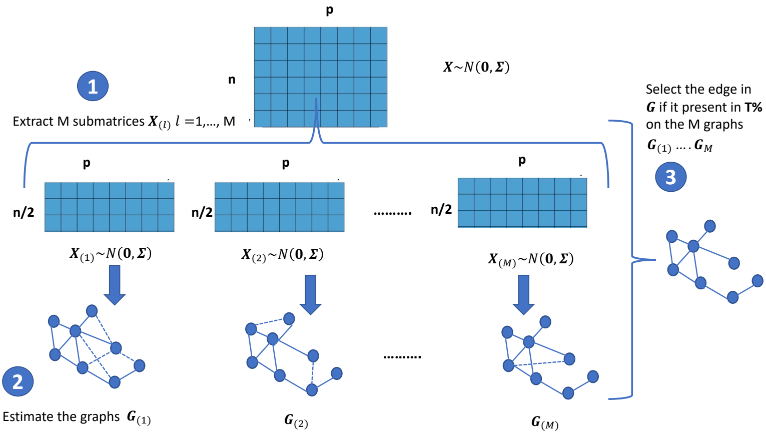




⇒ We solve the problem using the **Group descent algorithm**.

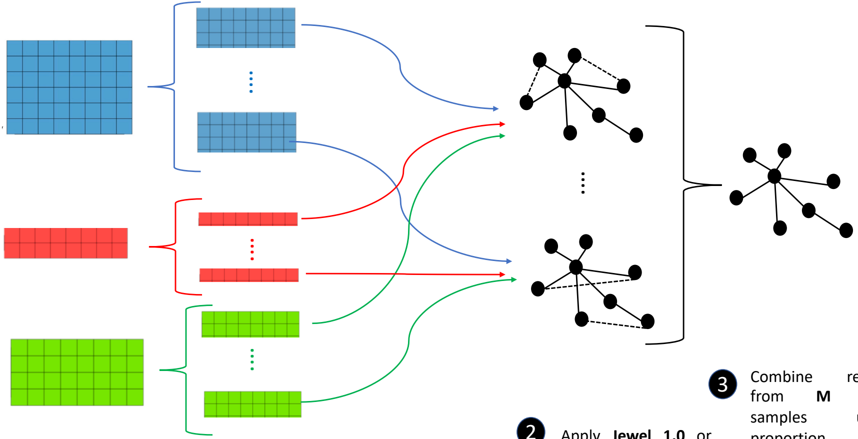
The stability selection approach

- Stability selection aims to reduce the number of false positives by retaining only the edges that consistently appear in different runs.
- We extended the idea of Stability Selection algorithm[4] to K datasets.


p
n
$X \sim N(\mathbf{0}, \boldsymbol{\Sigma})$
1
Extract M submatrices $X_{(l)}$ l =1,…, M
p
n/2
$X_{(1)} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$
p
n/2
$X_{(2)} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$
……….
p
n/2
$X_{(M)} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$
2
Estimate the graphs $G_{(1)}$
$G_{(2)}$
……….
$G_{(M)}$
Select the edge in G if it present in T% on the M graphs $G_{(1)}$ …. G_M
3

[4] Meinshausen and Buhlmann (2010)

The stability selection approach with multiple datasets



1 Subsample $\frac{n_k}{2}$ samples from each $X^{(k)}$

2 Apply **Jewel 1.0** or **Jewel 2.0** to each subsample

3 Combine results from **M** sub-samples using proportion of detected edges

Weighted minimization for handling Hubs

- **Hubs** are nodes with a large number of edges. The **degree** d_j of node j is given by the number of edges $\{(i,j), i=1,\dots,p\} \in E$
- We added weights $W_{i,j}$ to the penalty terms to reduce the penalization of hubs.

Let $W_{i,j} = W_{j,i} > 0, \quad i = 1..., p; j = 1..., p$ with $W_{ii} = 0$ the elements of a matrix of weights. We can consider the **weighed penalization** terms:

$$P_1(\boldsymbol{\Xi}^{(1)}, \dots, \boldsymbol{\Xi}^{(K)}) = \sum_{i<j=1}^{p} \sqrt{g_{ij}} W_{i,j} \sqrt{\sum_{k:\{X_i,X_j\}\in V_k} \left(\Xi_{ij}^{(k)}\right)^2 + \left(\Xi_{ji}^{(k)}\right)^2}$$

$$P_2(\boldsymbol{\Gamma}^{(1)}, \dots, \boldsymbol{\Gamma}^{(K)}) = \sum_{i<j=1}^{p} \sqrt{2} W_{i,j} \sum_{k:\{X_i,X_j\}\in V_k} \sqrt{\left(\Gamma_{ij}^{(k)}\right)^2 + \left(\Gamma_{ji}^{(k)}\right)^2}$$

$\Rightarrow$ We can choose

$$W_{i,j} \propto \frac{1}{\sqrt{d_i * d_j}}.$$

$\Longrightarrow$ Nodes that have higher degree are less penalized.

Simulations

- For given number of vertices p we generated a scale-free graph $\mathbf{G}$ with Barabasi algorithm[5].
- We introduced changes to a graph $K-1$ times (to obtain K similar graphs $\mathbf{G}_k$) with the rewire function[6], i.e., we changed the vertexes connections keeping the order of the graph and the degree distribution the same.
- We use $G_k, \quad k=1,...,K$ as support to generate precision matrices $\boldsymbol{\Omega}^{\boldsymbol{(k)}}$, then covariance matrices $\boldsymbol{\Sigma}^{\boldsymbol{(k)}}$ and data matrices $\mathbf{X}^{(k)} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\boldsymbol{(k)}})$ with given number of samples n.

In **Jewel 2.0** we optimize[7] over λ_1 and we fixed $\lambda_2 = 1.4 * \lambda_1$

[5] We can control sparsity with m parameter and hub-structure with parameter *power*

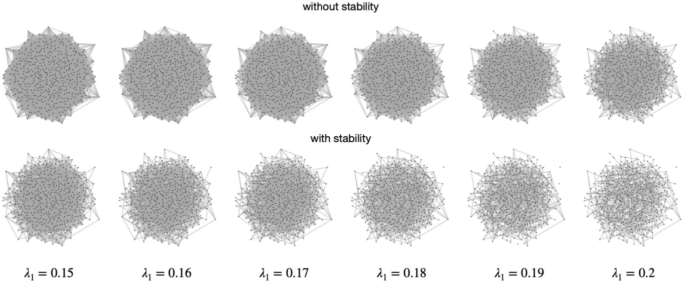
[6] we can control the amount of changes

[7] We are working to a data driven procedure for estimating both λ_1 and λ_2

Stability selection performance

K = 3, **p** = 500, **n** = 100,
m = 1, power = 1,
#edges = 499 in each matrix
#edges in common =370 (26% diff)

$\lambda_2 = 1.4\lambda_1$
nsubsets = 20
fraction = 0.6



without stability

with stability

- When applying stability selection procedure for a fixed λ the number of edges decreases. However, most of the removed edges turn to be false positive $\Longrightarrow$ the False Positive Rate improves at the price of a smaller loss of Power.

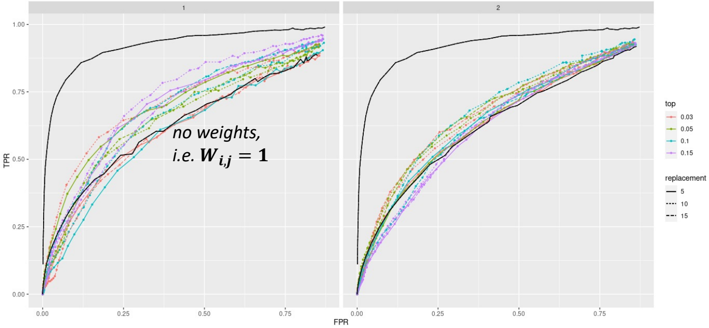
$\Longrightarrow$ for a proper range of λ, the **accuracy improves**.

Performance in Networks with Hubs

K = 3, **p = 200, n = 100,**
power = 1.5 (bigger hubs)
m = 1 (#edges = 199) and **m = 2** (#edges = 397)

50 λ_1 from 0.01 to 1.5
$\lambda_2 = 1.4\lambda_1$
Averaged results over 5 runs

$$W_{i,j} = \frac{1}{\max W_{i,j}} \frac{1}{\sqrt{d_i d_j}}$$



no weights,
i.e. $\boldsymbol{W_{i,j} = 1}$

We are now exploring several ways for estimating the node degrees d_k.

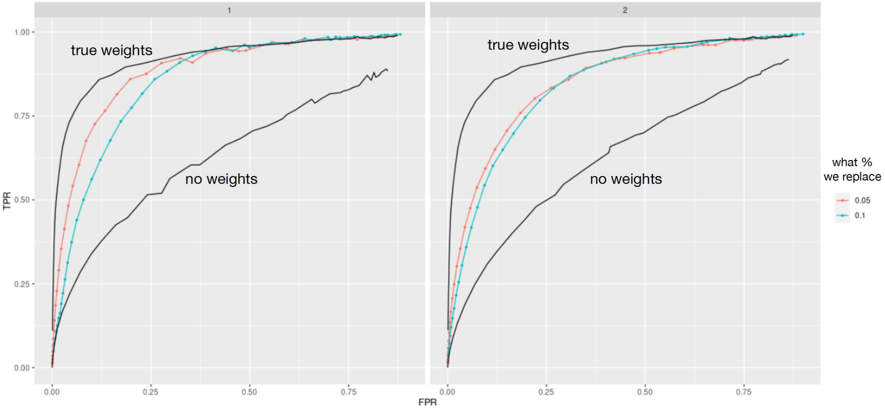
Performance in Detecting Hubs

K = 3, **p = 200, n = 100,**
power = 1.5 (bigger hubs)

$\lambda_2 = 1.4\lambda_1$
nruns = 5

m = 1 (#edges = 199), min degree = 1

m = 2 (#edges = 397), min degree = 2



Indeed, it might sufficient only to identify **hubs**.

Conclusions

- *Jewel 1.0* and *Jewel 2.0* are novel methods to jointly estimate the conditional probability graphs $\mathbf{G}_k$ from several datasets, allowing to capture both **common** and **case-specific** parts.
- With *Jewel 2.0*, we also handle the **presence of hubs**, and **reduce the False Positive edges**.
- For *Jewel 1.0* we released an open-source **R-package**.
- We showed the competitive performance of both approaches in extensive **simulations** and **real-data** analyses.

However, *Jewel 2.0* is still under active development.

- We need finding automatic criterion for the choice of the regularization parameters (both in the classical setting and when we use the stability selection procedure).
- We need finding suitable (easy to compute) estimates for hub-weights.
- We need releasing the final open-source *Jewel 2.0* R-package.

Thanks

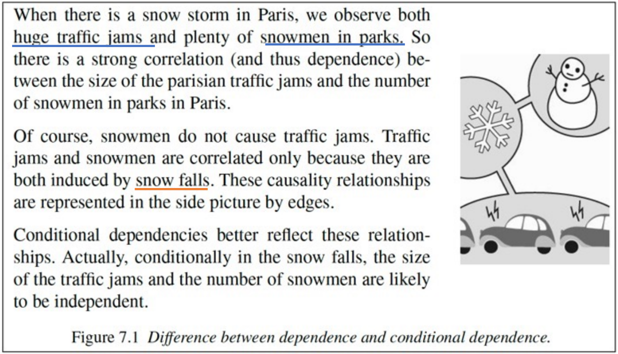
This work was supported by the project **"Antitumor Drugs and Vaccines from the Sea (ADViSE)"**

ADViSE
ANTITUMOR DRUGS AND VACCINES FROM THE SEA

Daniela De Canditiis

Anna Plaksienko

Conditional dependency



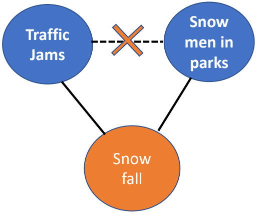
When there is a snow storm in Paris, we observe both huge traffic jams and plenty of snowmen in parks. So there is a strong correlation (and thus dependence) between the size of the parisian traffic jams and the number of snowmen in parks in Paris.

Of course, snowmen do not cause traffic jams. Traffic jams and snowmen are correlated only because they are both induced by snow falls. These causality relationships are represented in the side picture by edges.

Conditional dependencies better reflect these relationships. Actually, conditionally in the snow falls, the size of the traffic jams and the number of snowmen are likely to be independent.

Figure 7.1 *Difference between dependence and conditional dependence.*

From C. Giraud "*Introduction to High dimensional statistics*"



Graphical models (Definitions)

- Two random variables (X_i, X_j) of a random vector $\mathbf{x} = (X_1, \ldots, X_p)$ are ***conditionally independent*** $\iff$

$$f(X_i, X_j | X_{\{l, l \neq i,j\}}) = f(X_i | X_{\{l, l \neq i,j\}}) f(X_j | X_{\{l, l \neq i,j\}}).$$

- A graph $\mathbf{G} = (V, E)$ is an ***undirected graphical model*** with respect to the random vector $\mathbf{x}$, if it satisfies the *pairwise Markov property*:

$$\forall i, j : (i,j) \notin E, i \neq j \Rightarrow X_i \perp\!\!\!\perp X_j | X_{\{l, l \neq i,j\}}.$$

- Important case – **Gaussian distribution**, $\mathbf{x} = (X_1, \ldots, X_p) \sim \mathcal{N}_p(0, \mathbf{\Sigma})$. Then edge $(i,j) \in E \iff \left(\mathbf{\Sigma}^{-1}\right)_{ij} \neq 0$, entry of **precision matrix** is non-zero.

$\Longrightarrow$ Under Gaussian settings, estimating $\mathbf{G}$ is equivalent to estimating the support of $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$, where $\mathbf{\Omega}$ is the **Precision matrix**.

Note

$\text{supp}(\mathbf{\Omega}) \in \{0, 1\}^{p \times p}$ is the **Adjacency matrix** of the graph $\mathbf{G}$.

Maximum likelihood approach

Classical Inference: Given $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T \in R^{N \times p}$ we want to estimate $\mathbf{G}$.

Consider the Precision matrix $\mathbf{\Omega} = \mathbf{\Sigma}^{-1} \in R^{p \times p}$

and the Empirical covariance matrix $\mathbf{S} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i^T \mathbf{x} \in R^{p \times p}$

Under Gaussian settings, the **Log-likelihood** function is

$$\log \mathcal{L}(\mathbf{\Omega}) = \mathrm{tr}(S\mathbf{\Omega}) - \log |\mathbf{\Omega}|$$

therefore, the **maximum likelihood estimate** $\hat{\mathbf{\Omega}} = \mathbf{S}^{-1}$ where $\hat{\mathbf{\Omega}} \to \mathbf{\Omega}$ as $N \to \infty$, and $\hat{\mathbf{G}} = \mathrm{supp}(\mathbf{\Omega})$

However,

1. when $\mathbf{p} >> \mathbf{N}$, matrix $\mathbf{S}$ is not invertible;
2. MLE is not *sparse* ⇒ the graph $\mathbf{G}$ is not explicative.

Jewel 1.0 numerical algorithm

Algorithm 1 The *jewel* algorithm

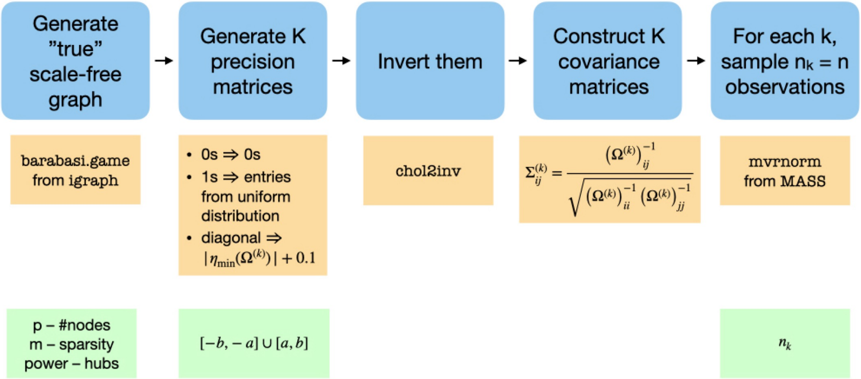
```
INPUT: $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(K)}$, $\lambda$, $tol$ and $t_{\max}$

INITIALIZE:
$\mathbf{\Theta}^{(1,\,0)}, \dots, \mathbf{\Theta}^{(K,\,0)}$
$\mathbf{R}^{(k)} = \mathbf{X}^{(k)} - \mathbf{X}^{(k)}\mathbf{\Theta}^{(k,0)}$, $k = 1 \dots K$
1  $\mathbf{Active} = \begin{pmatrix} 0 & 1 & \dots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$

REPEAT UNTIL CONVERGENCE
for $j = 1 \dots p$ do
  for $i = j+1 \dots p$ : do
    if $Active_{ij} \neq 0$
    evaluate $\mathbf{z} = \left(z_{ij}^{(1)}, z_{ji}^{(1)}, \dots, z_{ij}^{(K)}, z_{ji}^{(K)}\right)$ by
2   $\left\{ \begin{array}{l} z_{ij}^{(k)} = \frac{1}{n_k} {X_{.i}^{(k)}}^{T} R_{.j}^{(k)} + \Theta_{ij}^{(k,\,t)} \\ z_{ji}^{(k)} = \frac{1}{n_k} {X_{.j}^{(k)}}^{T} R_{.i}^{(k)} + \Theta_{ji}^{(k,\,t)} \end{array} \right.$
3   evaluate $threshold = 1 - \lambda / \|\mathbf{z}\|$
    if $threshold < 0$ then
      $Active_{ij} \leftarrow 0$ and $\mathbf{z} \leftarrow 0$
    else
      $\mathbf{z} \leftarrow \mathbf{z} \cdot threshold$  ⟵
    end if
    update residuals
4   $\left\{ \begin{array}{l} R_{.j}^{(k)} = R_{.j}^{(k)} + X_{.i}^{(k)} \left(\Theta_{ij}^{(k,t)} - z_{ij}^{(k)}\right) \\ R_{.i}^{(k)} = R_{.i}^{(k)} + X_{.j}^{(k)} \left(\Theta_{ji}^{(k,t)} - z_{ji}^{(k)}\right) \end{array} \right.$
    update coefficients
    $\left(\Theta_{ij}^{(1,\,t)}, \Theta_{ji}^{(1,\,t)}, \dots, \Theta_{ij}^{(K,\,t)}, \Theta_{ij}^{(K,\,t)}\right) \leftarrow \mathbf{z}$

  end for
end for
5  Stop if $\frac{\sum_k \left|\mathbf{\Theta}^{(k,\,t+1)} - \mathbf{\Theta}^{(k,t)}\right|}{\sum_k \left|\mathbf{\Theta}^{(k,t)}\right|} < tol$ or $t > t_{\max}$
OUTPUT: Active
```

Simulation Scheme



Generate "true" scale-free graph → Generate K precision matrices → Invert them → Construct K covariance matrices → For each k, sample $n_k = n$ observations

barabasi.game from igraph

- 0s ⇒ 0s
- 1s ⇒ entries from uniform distribution
- diagonal ⇒ $|\eta_{\min}(\Omega^{(k)})| + 0.1$

chol2inv

$$\Sigma^{(k)}_{ij} = \frac{\left(\Omega^{(k)}\right)^{-1}_{ij}}{\sqrt{\left(\Omega^{(k)}\right)^{-1}_{ii}\left(\Omega^{(k)}\right)^{-1}_{jj}}}$$

mvrnorm from MASS

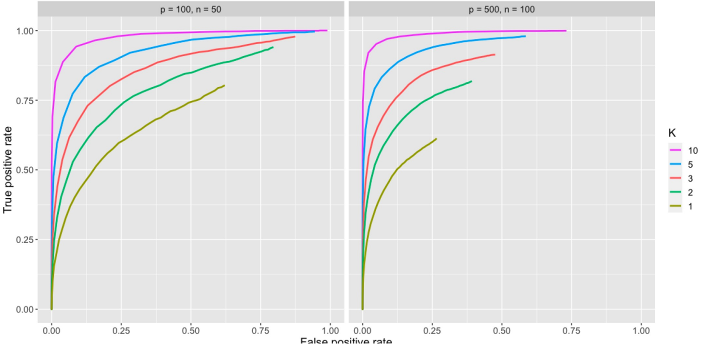
p – #nodes
m – sparsity
power – hubs

$[-b, -a] \cup [a, b]$

n_k

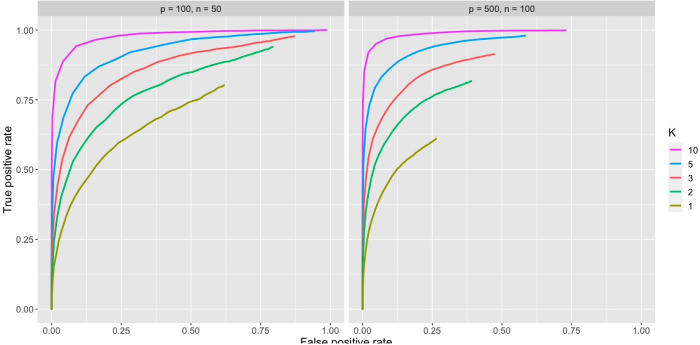
Performance of Jewel 1.0 when incresing the number of datasets

1. Generate $K = 10$ datasets $\boldsymbol{X}^{(1)}, \dots, \boldsymbol{X}^{(K)}$ for $p = 100$, $n_k = 50$ and for $p = 500$, $n_k = 100$. Repeat 20 times.
2. Apply *jewel*, evaluate average TPR and FPR.
3. In each run, sample $K = 5$ matrices, repeat.
4. In each run subsample $K = 3$, $K = 2$, $K = 1$ matrices, repeat.



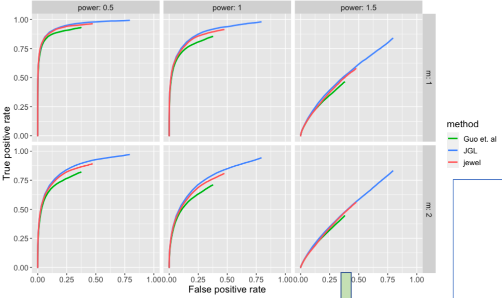
Performance of Jewel 1.0 compared to Naive integration approaches

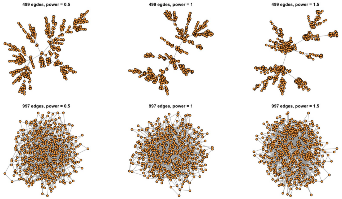
1. Generate $K = 10$ datasets $\boldsymbol{X}^{(1)}, \dots, \boldsymbol{X}^{(K)}$ for $p = 100$, $n_k = 50$ and for $p = 500$, $n_k = 100$. Repeat 20 times.
2. Apply *jewel*, evaluate average TPR and FPR.
3. In each run, sample $K = 5$ matrices, repeat.
4. In each run subsample $K = 3$, $K = 2$, $K = 1$ matrices, repeat.



Performance of Jewel 1.0 compared with other methods

- **JGL**: package JGL, group penalty, $\lambda_1 = 0$ so all $\mathrm{supp}(\hat{\Omega}^{(k)})$ are the same.
- **Guo et al. proposal**: code was provided by authors; $\hat{\mathbf{A}} = \cup_k \mathrm{supp}(\hat{\Omega}^{(k)})$
- $\mathbf{K} = \mathbf{3}$, $\mathbf{p} = \mathbf{500}$, $\mathbf{n_k} = \mathbf{100}$.





	jewel	JGL	Guo et. al
$\lambda = 0.01$	$\approx$7 min	$\approx$11.5 min	$\approx$66 min
$\lambda = 0.1$	41.2 sec	80.2 sec	$\approx$2.6 min
$\lambda = 0.2$	26.3 sec	73.8 sec	73.7 sec
$\lambda = 0.52$	26.3 sec	0.3 sec	22.9 sec
$\lambda = 0.83$	26.3 sec	0.1 sec	12.1 sec
$\lambda = 1$	22.7 sec	0.1 sec	10.9 sec
grid of 50 λ	$\approx$1.5 hours	$\approx$3.4 hours	$\approx$8.4 hours

Table: Running time for *jewel*, JGL and Guo et al. proposal for $K = 3$, $p = 500$, $n_k = 100$, $m = 1$ and *power* $= 1$.

With warm up *jewel* is even faster (several minutes for the whole grid).

Lack of performance for all methods oin presence of **hubs**

→ We also proved that with **a random update order** the algorithm converge even faster

Jewel 2.0 numerical algorithm

Algorithm: *Jewel* 2.0 Algorithm

```
INPUT: $\mathbf{X}^{(1)}, ..., \mathbf{X}^{(K)}, \lambda_1, \lambda_2, tol$ and $t_{\max}$

INITIALIZE:
$\mathbf{\Xi}^{(1,0)}, ..., \mathbf{\Xi}^{(K,0)}$
$\mathbf{\Gamma}^{(1,0)}, ..., \mathbf{\Gamma}^{(K,0)}$
$\mathbf{Rg}^{(k)} = \mathbf{Rxi}^{(k)} = \mathbf{X}^{(k)} - \mathbf{X}^{(k)}\mathbf{\Gamma}^{(k,0)} - \mathbf{X}^{(k)}\mathbf{\Xi}^{(k,0)}$
$\mathbf{Active} = \mathbf{Active}^{(k)} = \begin{pmatrix} 0 & 1 & \dots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \forall k$

REPEAT UNTIL CONVERGENCE
*Fix specific part* $\mathbf{\Gamma}^{(1,t)}, ..., \mathbf{\Gamma}^{(K,t)}$, *update COMMON part.*
Generate $order_{\Xi}$ by resampling from 1 to $p$.
for $j \in order_{\Xi}$ do
  for $i = j+1 \dots p$ do
    if $Active_{ij} \neq 0$
    evaluate $\mathbf{z}_{\Xi} = \left(z_{ij}^{(1)}, z_{ji}^{(1)}, \dots, z_{ij}^{(K)}, z_{ji}^{(K)}\right)$ by
        $z_{ij}^{(k)} = \frac{1}{n_k} X_{.i}^{(k)^T} Rxi_{.j}^{(k)} + \Xi_{ij}^{(k,t)}$
        $z_{ji}^{(k)} = \frac{1}{n_k} X_{.j}^{(k)^T} Rxi_{.i}^{(k)} + \Xi_{ji}^{(k,t)}$
    evaluate $threshold = 1 - \lambda_1/\|\mathbf{z}\|$
    if $threshold < 0$ then
      $Active_{ij} \leftarrow 0$ and $\mathbf{z} \leftarrow 0$
    else
      $\mathbf{z} \leftarrow \mathbf{z} \cdot threshold$
    end if
    update residuals
        $Rxi_{.j}^{(k)} = Rxi_{.j}^{(k)} + X_{.i}^{(k)} \left(\Xi_{ij}^{(k,t)} - z_{ij}^{(k)}\right)$
        $Rxi_{.i}^{(k)} = Rxi_{.i}^{(k)} + X_{.j}^{(k)} \left(\Xi_{ji}^{(k,t)} - z_{ji}^{(k)}\right)$
    update coefficients $(\Xi_{ij}^{(1,t+1)}, \Xi_{ji}^{(1,t+1)}, ..., \Xi_{ij}^{(K,t+1)}, \Xi_{ij}^{(K,t+1)}) \leftarrow \mathbf{z}$
  end for
end for
Update $\mathbf{Rg}^{(k)} = \mathbf{X}^{(k)} - \mathbf{X}^{(k)}\mathbf{\Gamma}^{(k,t)} - \mathbf{X}^{(k)}\mathbf{\Xi}^{(k,t)}$
```

Fix $\Gamma^{(1)}, \dots, \Gamma^{(K)}$
Update Common part
$\Xi^{(1)}, \dots, \Xi^{(K)}$

```
*Fix common part* $\mathbf{\Xi}^{(1,t)}, ..., \mathbf{\Xi}^{(K,t)}$, *update SPECIFIC*
Generate $order_{\Gamma}$ by resampling from 1 to $p$.
for $k = 1 \dots K$ do
  for $j \in order_{\Gamma}$ do
    for $i = j+1 \dots p$ do
      if $Active_{ij}^{(k)} \neq 0$
      evaluate $\mathbf{z}_{\Gamma} = \left(z_{ij}^{(k)}, z_{ji}^{(k)}\right)$ by
          $z_{ij}^{(k)} = \frac{1}{n_k} X_{.i}^{(k)^T} Rg_{.j}^{(k)} + \Gamma_{ij}^{(k,t)}$
          $z_{ji}^{(k)} = \frac{1}{n_k} X_{.j}^{(k)^T} Rg_{.i}^{(k)} + \Gamma_{ji}^{(k,t)}$
      evaluate $threshold = 1 - \lambda_2/\|\mathbf{z}\|$
      if $threshold < 0$ then
        $Active_{ij}^{(k)} \leftarrow 0$ and $\mathbf{z} \leftarrow 0$
      else
        $\mathbf{z} \leftarrow \mathbf{z} \cdot threshold$
      end if
      update residuals
          $Rg_{.j}^{(k)} = Rg_{.j}^{(k)} + X_{.i}^{(k)} \left(\Gamma_{ij}^{(k,t)} - z_{ij}^{(k)}\right)$
          $Rg_{.i}^{(k)} = Rg_{.i}^{(k)} + X_{.j}^{(k)} \left(\Gamma_{ji}^{(k,t)} - z_{ji}^{(k)}\right)$
      update coefficients $(\Gamma_{ij}^{(k,t+1)}, \Gamma_{ji}^{(k,t+1)}) \leftarrow \mathbf{z}$
    end for
  end for
end for
Update $\mathbf{Rxi}^{(k,t+1)} = \mathbf{X}^{(k)} - \mathbf{X}^{(k)}\mathbf{\Gamma}^{(k,t)} - \mathbf{X}^{(k)}\mathbf{\Xi}^{(k,t)}$

Combine $\mathbf{\Theta}^{(k,t+1)} = \mathbf{\Xi}^{(k,t+1)} + \mathbf{\Gamma}^{(k,t+1)}$
Stop if $\frac{\sum_k \left|\mathbf{\Theta}^{(k,t+1)} - \mathbf{\Theta}^{(k,t)}\right|}{\sum_k \left|\mathbf{\Theta}^{(k,t)}\right|} < tol$ or $t > t_{\max}$

OUTPUT:
$\hat{\mathbf{G}}^{(k)} = \text{supp}\,\hat{\mathbf{\Theta}}^{(k)}$;
$\hat{\mathbf{G}} = \cap_k \hat{\mathbf{G}}^{(k)}$
```

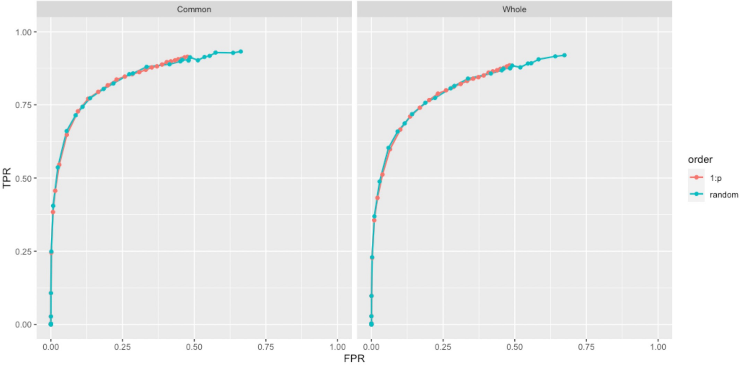
Fix $\Xi^{(1)}, \dots, \Xi^{(K)}$
Update Specific part
$\Gamma^{(1)}, \dots, \Gamma^{(K)}$

Update Residuals

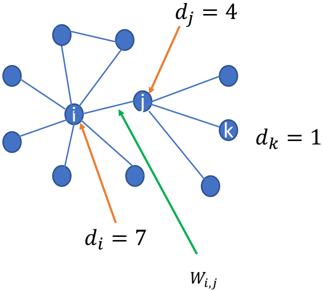
Update
$\Theta^{(k)} = \Xi^{(k)} + \Gamma^{(k)}$

Randomized update performance

Randomizing the group update order does not affect the performance but decreases running time.
→ Empirically, we showed that the algorithm converges faster (i.e., it reduces the number of iterations), retaining the same performance.



Incorporating Network Topology for Detecting Hubs



The node degree is **equal to the number of node neighbors**.

Biological Networks have a scale-free degree distribition→ fews nodes that are hubs and several nodes with small degree