Inferring Networks from Multiple Samples with Consensus LASSO

Nathalie Villa-Vialaneix[1,2,*], Matthieu Vignes[2],

Nathalie Viguerie[3,4], Magali San Cristobal[5,6]

[1] SAMM, Université Paris 1, 75634 Paris - France

[2] Unité MIAT, INRA, 31326 Castanet Tolosan - France

[3] Inserm, UMR1048, Obesity Research Laboratory, I2MC, Institute of Metabolic and Cardiovascular Diseases, Toulouse - France

[4] University of Toulouse, UMR1048, Paul Sabatier University, Toulouse - France

[5] Laboratoire de Génétique Cellulaire, INRA, 31326 Castanet Tolosan - France

[6] Département GMM, INSA Toulouse, 31077 Toulouse - France

Abstract: Networks are very useful tools to decipher complex regulatory relationships between genes in an organism. Most work address this issue in the context of i.i.d., treated vs. control or time-series samples. However, many data sets include expression obtained for the same cell type of an organism, but in several conditions. We introduce a novel method for inferring networks from samples obtained in various but related experimental conditions. This approach is based on a double penalization: a first penalty aims at controlling the global sparsity of the

[*] Corresponding author: `nathalie.villa@univ-paris1.fr`.

solution whilst a second penalty is used to make condition-specific networks consistent with a consensual network. This “consensual network” is introduced to represent the dependency structure between genes, which is shared by all conditions. We show that different “consensus” penalty can be used, some integrating prior (e.g., bibliographic) knowledge and others that are adapted along the optimization scheme. In all situations, the proposed double penalty can be expressed in terms of a LASSO problem and hence, solved using standard approaches which address quadratic problems with L_1-regularization. This approach is combined with a bootstrap approach and is made available in the R package **therese** [1]. Our proposal is illustrated on simulated datasets and compared with independent estimations and alternative methods. It is also applied to a real dataset to emphasize the differences in regulatory networks before and after a low-calorie diet.

Keywords: network inference; Gaussian graphical model; regulation network; LASSO

1 Introduction

The recent development of high-throughput techniques produces huge datasets where thousand of gene expressions are simultaneously measured. However, the number of observations is comparatively very small, and those are often measured in a variety of experimental conditions. One of the big challenges of modern Systems Biology is to understand the influence of controlled experimental conditions

[1]**therese** can be downloaded on R-Forge, from http://therese-pkg.r-forge.r-project.org/.

on the functioning of living organisms. This question is usually addressed by searching for the differences between gene expressions pertaining to the conditions (hence for "*differentially expressed genes*"). A more comprehensive look at the roles of the genes of an organism can be obtained by deciphering the interactions of these genes with each other; finding which regulation pathways are modified by a given experimental condition gives an interesting insight on the influence of the condition on the living system as a whole.

One of the most popular approach to understand the complex relationships existing between the expression of a large set of genes is to infer a co-expression network from a transcriptomic dataset. In such a model, nodes of the network represent the genes and an edge is meant to stand for a regulatory link between the two nodes it connects. A large number of different methods have been proposed to infer such networks: using correlations ("relevance network", [4]), Bayesian networks [19, 20], Gaussian Graphical Model [7, 21]... When observations are collected in different conditions, a naive approach would be to independently infer a network for each condition and to compare them. However, this method is not suited to highlight specific differences and shared motifs of regulation phenomenons. Moreover, since the number of observations is often too small, inferring networks independently (assuming that a common functioning exists in most scenarii) leads to emphasize irrelevant differences. Several proposals have already been made to overcome this issue: [5, 6, 17] use a modified Gaussian graphical model and [13] proceeds with two steps with a clustering prior to the inference. The proposal developed in this paper is close to that of [5, 6, 17]: a Gaussian graphical model is used and two interpretable penalties are added to the likelihood. The first penalty aims at inferring sparse solutions; the second penalty is used to make networks obtained in different conditions consistent with a consensual network. The "consensual network" is introduced to represent high-level dependencies between genes, i.e., a common functioning of the living organism under study, in most situations. It can either include prior (e.g., bibliographic) knowledge or be expressed from the condition-specific networks. Finally, the estimation is made more robust by using a bootstrap approach.

The paper is organized as follows: Section 2 describes the double penalty approach. Section 3 explains our proposal for estimating the networks with a bootstrap strategy. Finally, Section 4 provides experimental results on simulations.

2 cLasso

In the Gaussian graphical model (GGM) framework, the classical objective is to estimate the graph of conditional dependencies between p variables (usually modeling gene expressions), $(X_j)_{j=1,\ldots,p}$, from n i.i.d. observations of the variables, namely $(X_{ij})_{i=1,\ldots,n}$, $\forall\, j \in \{1,\ldots,p\}$. Each p-dimensional vector $\mathbf{X}_{i.}$ is assumed to be the realization of a Gaussian random variable $\mathcal{N}\,(0,\Sigma)$. In this framework, non-zero entries of the concentration matrix $\mathbf{K} = \Sigma^{-1}$ exactly encode actual edges (between genes) in the conditional dependency graph. In the present section, we describe how this framework can be extended to the case where observations are obtained from different samples, each sample being measured in a given (but related) experimental condition.

2.1 Inferring multiple networks with GGM

Now assuming that the p gene expressions are measured from k samples, each corresponding to a specific experimental condition, the following model can be set: $(X_j^c)_{j=1,\ldots,p,\, c=1,\ldots,k}$ are k Gaussian p-dimensional vector, $\mathcal{N}(0,\Sigma^c)$. A total of n observations are available: $(X_{ij}^1)_{i=1,\ldots,n_1,\, j=1,\ldots,p} \ldots (X_{ij}^k)_{i=1,\ldots,n_k,\, j=1,\ldots,p}$, with $\sum_c n_c = n$ and, for all c and all i, $(X_{ij}^c)_{j=1,\ldots,p}$ are i.i.d. observations of $\mathcal{N}(0,\Sigma^c)$. In the following, our goal is twofold:

- inferring k sparse graphs that model gene regulations in the k conditions;
- finding one consensual graph that models a "shared" functioning between conditions.

The GGM framework is used for the inference. As previously explained, the concentration matrices $\mathbf{K}^c = (\Sigma^c)^{-1}$ need be estimated and the entries of these matrices exactly measure conditional dependencies between variables $(X_j^c)_j$ through partial correlation coefficients, $s_{jj'}^c = \mathbb{C}\mathrm{or}\left(X_j^c, X_{j'}^c | (X_l^c)_{l\neq j,\, j'}\right)$ because of the relation $s_{jj'}^c = -\frac{\mathbf{K}_{jj'}^c}{\sqrt{\mathbf{K}_{jj}^c \mathbf{K}_{j'j'}^c}}$ [26].

These quantities can be estimated by considering the following $(k \times p)$ linear regression problems [16]: $\forall\, c = 1, \ldots, k$, $\forall\, j = 1 \ldots, p$,

$$\mathbf{X}_j^c = \mathbf{X}_{\backslash j}^c \beta_j^c + \epsilon_j^c, \tag{1}$$

where $\mathbf{X}_{\backslash j}^c$ is the matrix $\mathbf{X}^c = \left(X_{ij}^c\right)_{i=1,\ldots,n_c,\, j=1,\ldots,p}$ deprived from its j-*th* column $\mathbf{X}_j^c$, $\beta_j^c = (\beta_{jj'}^c)_{j'\neq j}$ is a $(p-1)$-dimensional vector and ϵ_j^c is a Gaussian centered error. In the Gaussian framework, it can be shown that the coefficients of the linear model are related to the previous quantities by $\beta_{jj'}^c = -\frac{\mathbf{K}_{jj'}^c}{\mathbf{K}_{jj}^c}$.

The $k \times p$ linear models of Equation (1) can be jointly estimated by maximizing a pseudo-likelihood:

$$\mathcal{L}(\mathbf{K}|\mathbf{X}) = \sum_{c=1}^{k} \sum_{j=1}^{p} \sum_{i=1}^{n_c} \log \mathbb{P}\left(X_{ij}^c | \mathbf{X}_{i,\backslash j}^c,\, \mathbf{K}_j^c \right). \tag{2}$$

[5] proved that maximizing the pseudo-likelihood of Equation (2) over matrices $(\mathbf{K}^c)_c$ is equivalent to minimizing the following p quantities simultaneously:

$$\forall\, j = 1, \ldots, p, \qquad \frac{1}{2} \beta_j^T \widehat{\Sigma}_{\backslash j \backslash j} \beta_j + \beta_j^T \widehat{\Sigma}_{j \backslash j}. \tag{3}$$

The p problem of Equation (3) are $(p-1) \times k$-dimensional quadratic optimization problems and we specify:

- $\forall\, c = 1, \ldots, k$, $\beta_j^c = (\beta_{jj'}^c)_{j' \neq j} \in \mathbb{R}^{p-1}$, where $\beta_{jj'}^c = (\mathbf{K}^c)_{jj}^{-1} \mathbf{K}_{jj'}^c$;
- $\beta_j = \left(\beta_j^1, \ldots, \beta_j^k \right)^T \in \mathbb{R}^{k(p-1)}$;
- $\widehat{\Sigma}_{\backslash j \backslash j}$ is the block diagonal matrix $\widehat{\Sigma}_{\backslash j \backslash j} = \mathbb{D}\text{iag}\left(\widehat{\Sigma}_{\backslash j \backslash j}^1, \ldots, \widehat{\Sigma}_{\backslash j \backslash j}^k \right)$, having dimensions $k(p-1) \times k(p-1)$;
- $\widehat{\Sigma}_{j \backslash j}$ is the $k(p-1)$-dimensional vector, $\left(\widehat{\Sigma}_{j \backslash j}^1, \ldots, \widehat{\Sigma}_{j \backslash j}^k \right)$.

However, this approach leads to matrices without non-zero entries. Moreover, when $(n_c)_c$ are not larger than p, the estimation of $(\beta_j)_j$ becomes trickier and pseudo-inverse methods lead to highly unstable results. Using the additional assumption that conditional dependency graphs are indeed sparse, a standard approach is to add a L_1-penalization to the likelihood of Equation (2) ("Graphical LASSO", see [9]) or, alternatively, to consider p independent L_1-penalized problems derived from those of Equation (3), see [5, 16]. The latter, more direct approach, has been reported to be more accurate in terms of edge detection in [25].

2.2 Using a "consensus" penalty

In the previous section, the conditional dependency graphs are obtained from each sample *independently*. The assumption that the graphs issued from the different experimental conditions should be somehow alike, is not integrated into the model. Especially in the case where the sample sizes are low, such an assumption should help to predict edges more accurately. Various techniques exist to address this issue: [5] proposed to replace the covariance matrices $\widehat{\Sigma}^c$ by mixing it with the covariance matrices corresponding to the other conditions. Alternatively, some authors suggest to penalize the pseudo-likelihood by a penalty that can explicitly deal with the similarity between condition-specific graphs *via* different strategies:

- [5] proposed two kinds of Group-LASSO type penalties: $P((\mathbf{K}^c)_c) = \sum_{ij} \sqrt{\sum_c (\mathbf{K}^c_{ij})^2}$ (Group-LASSO) and $P((\mathbf{K}^c)_c) = \sum_{ij} \left[\sqrt{\sum_c (\mathbf{K}^c_{ij})^2_+} + \sqrt{\sum_c (\mathbf{K}^c_{ij})^2_-}\right]$ (sign-coherent Group-LASSO or "Co-operative LASSO"). The group-LASSO penalty globally controls sparsity and inferred edges are common to all conditions. The sign coherent option of their penalization scheme offers the possibility to enforce an edge to encode either an activating or repressing process but not both: it provides strongly similar networks between conditions and has been proven to be efficient in case of experimental conditions leading to small changes in the regulations. However, for some particular applications (e.g., certain forms of cancer that lead to a complete re-organization of the living system), the assumption that the relations between two genes is always a repressing/enhancing relation is not biologically desirable;

- [6] used the penalty $P((\mathbf{K}^c)_c) = \sum_{c \neq c'} \|\mathbf{K}^c - \mathbf{K}^{c'}\|_1$, where $\|.\|_1$ is the standard L_1-norm, which commands a strong similarity across conditions. This approach would lead to very similar condition-specific network, allowing only a few differences. Unlike the Cooperative Lasso approach described above, no special sign-coherent assumption is required but this method is more suited when condition-specific networks are not supposed to be very different;

- [17] introduced the penalty $P((\mathbf{K}^c)_c) = \sum_{c \neq c'} \sum_{j=1}^{p} \sum_j \|\mathbf{K}_j^c - \mathbf{K}_j^{c'}\|_2$, where $\|.\|_2$ is the standard L_2-norm and $\mathbf{K}_j$ is the j-th column of $\mathbf{K}$. Hence, this approach encourages the support of $\mathbf{K}^c - \mathbf{K}^{c'}$ to be the union of a given set of columns. Hence, this penalty only provides some flexibility to a few nodes to differ among conditions while all the other nodes have the same pattern of interactions.

The main idea of our proposal, we coined **cLasso**, is similar to the latter approaches, but using a softer penalization scheme than group-Lasso type penalties. This choice aims at better estimating the edges that are not similar and also do not need to assume a particular origin for the differences between conditions. The k inferred graphs, $\mathcal{G}^c$, are forced towards a “consensual” graph: the resulting graphs are different from each others, but these differences can be controlled. This idea is tackled by using a penalized ML framework in which two penalties are introduced:

- the first one is a sparse penalization which controls the number of edges in every graph $\mathcal{G}^c$;

- the second one is a L_2 penalization that aims at limiting the differences between the $(\mathbf{K}^c)_{c=1,\ldots,k}$.

More precisely, $\forall\, j = 1, \ldots, p$, a *consensual regression coefficient*, β_j^{cons}, is introduced, that can be defined from the sample-dependent coefficients β_j^c or can be fixed by a user, including, in particular, prior biological knowledge. This coefficient represents a kind of "global" solution, that is condition-independent. It is used by replacing the minimization problems described in Equation (3) by the following double-penalized minimization problems:

$$\frac{1}{2}\beta_j^T \widehat{\Sigma}_{\backslash j \backslash j}\beta_j + \beta_j^T \widehat{\Sigma}_{j \backslash j} + \lambda \|\beta_j\|_1 + \mu \sum_{c=1}^{k} \left\| \beta_j^{\text{cons}} - \beta_j^c \right\|_2^2 . \tag{4}$$

In Equation (4), β_j^{cons} is used to model the "consensus". In the following section, different types of consensus are described, and the practical computation of the solution is derived from the different cases. All described solutions lead to the optimization of quadratic problems penalized by the L_1-norm.

Contrary to the other approaches presented above, the second penalty of Equation (4) is a soft one, that does not control drastically the number of different edges between conditions but rather limits them. It is thus advisable in the case where the number of differences is not too low and where the user really wants to see the differences across the conditions. Also, contrary to [6, 17], our proposal does not rely on a penalty which complexity increases quadratically with the number of conditions (this might be a problem if the number of conditions is high). Finally, as explained in Section 2.3.1, the definition of a consensus network can integrate prior knowledge that can help estimating the network with an increased accuracy.

Remark 1 *As shown in Section 3.1, any choice for β_j^{cons} that leads to obtain a* *minimization problem that can be expressed as:*

$$\textit{convex part} + \lambda \|\beta_j\|_1$$

is a valid consensus choice that can be solved using a common framework. In particular, this includes any consensus that is expressed as a linear combination of the estimated coefficients β_j^c (Section 2.3.2) or (fixed) a priori consensus (Section 2.3.1).

2.3 Consensus choices

2.3.1 A fixed consensus

When a prior information is known on the network (e.g., a bibliographic network), a natural choice is to use it for β^{cons}. In this case, β^{cons} is fixed in advance and does not depend on $(\beta_j^c)_j$: it does not need to be estimated. However, if no prior information is available, the network estimated from all the samples considered as a whole or any combination of networks obtained with independent estimations can be used for consensus and considered as a (fixed) a priori information network.

Proposition 1 *Using a fixed β_j^{cons}, Equation (4) is equivalent to minimizing the following standard quadratic problem with L_1-penalty:*

$$\frac{1}{2}\beta_j^T B^1(\mu)\beta_j + \beta_j^T B^2(\mu) + \lambda\|\beta_j\|_1, \tag{5}$$

where $B^1(\mu) = \widehat{\Sigma}_{\backslash j \backslash j} + 2\mu \mathbb{I}_{k(p-1)}$, with $\mathbb{I}_{k(p-1)}$ the $k(p-1)$-identity matrix and $B^2(\mu) = \widehat{\Sigma}_{j\backslash j} - 2\mu\mathbb{I}_{k(p-1)}\beta_j^{cons}$ with β_j^{cons} a $k(p-1)$ vector that only depends on the prior β_j^{cons}.

Proof (and exact values for B^1 and B^2): The L_2-penalty of Equation (4) can be re-written as:

$$\sum_{c=1}^{k} \left\|\beta_j^{\text{cons}} - \beta_j^c\right\|_2^2 = \sum_{c=1}^{k} \left((\beta_j^c)^T \beta_j^c - 2(\beta_j^c)^T \beta_j^{\text{cons}} + \|\beta_j^{\text{cons}}\|_2^2\right)$$

Noticing that $\|\beta_j^{\text{cons}}\|_2^2$ is a fixed value that does not depend on the estimated coefficients β_j^c, it follows that minimizing Equation (4) is equivalent to minimizing:

$$\frac{1}{2}\beta_j^T \left(\widehat{\Sigma}_{\backslash j \backslash j} + 2\mu \mathbb{I}_{k(p-1)}\right) \beta_j + \beta_j^T \left(\widehat{\Sigma}_{j\backslash j} - 2\mu \mathbb{I}_{k(p-1)} \beta_j^{\text{cons}}\right) + \lambda \|\beta_j\|_1,$$

where β_j^{cons} is the vector $\left((\beta_j^{\text{cons}})^T, \ldots, (\beta_j^{\text{cons}})^T\right)^T$. $\square$

2.3.2 An averaged consensus

When no prior information is given, an intuitive and convenient choice for the consensus is to simply average the estimators over the different samples: $\beta_j^{\text{cons}} = \sum_{c=1}^{k} \frac{n_c}{n} \beta_j^c$. In this case, β_j^{cons} is a linear combination of the $(\beta_j^c)_c$, which is an interesting feature, as explained in Proposition 2. Notice that the choice of averaging the coefficients β_j^c is almost equivalent in terms of networks (i.e., in terms of non-zero entries) as having a consensus which is the union of the condition-dependent networks.

Proposition 2 *Using $\beta_j^{cons} = \sum_{c=1}^{k} \frac{n_c}{n} \beta_j^c$, Equation (4) can be re-written as the following standard quadratic problem with L_1-penalty:*

$$\frac{1}{2}\beta_j^T S_j(\mu) \beta_j + \beta_j^T \widehat{\Sigma}_{j\backslash j} + \lambda \|\beta_j\|_1 \tag{6}$$

where $S_j(\mu) = \widehat{\Sigma}_{\backslash j \backslash j} + 2\mu A^T A$ where A is a $k(p-1) \times k(p-1)$-matrix that does not depend on j.

Proof (and exact value for A): If, $\forall\, c = 1, \dots, k$, $U_c = \frac{n_c}{n}\mathbb{I}_{p-1}$ (with $\mathbb{I}_{p-1}$ the unit matrix having dimension $p-1$) and $V_c = \left(1 - \frac{n_c}{n}\right)\mathbb{I}_{p-1}$, then

$$\beta_j^c - \beta_j^{\text{cons}} = A_c \beta_j \, ,$$

where A_c is the $(p-1) \times k(p-1)$-matrix $[-U_1, \dots, -U_{c-1}, V_c, -U_{c+1}, ..., -U_p]$. Then,

$$\sum_{c=1}^{k} \left\| \beta_j^{\text{cons}} - \beta_j^c \right\|_2^2 = \sum_{c=1}^{k} \beta_j^T A_c^T A_c \beta_j$$

and thus, setting

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_k \end{pmatrix},$$

implies that

$$\sum_{c=1}^{k} \left\| \beta_j^{\text{cons}} - \beta_j^c \right\|_2^2 = \beta_j^T A^T A \beta_j$$

which concludes the proof. □

Remark 2 *Because the term $\beta_j^T A^T A \beta_j$ is a quadratic term in β, the formulation of the minimization problem given in Equation (4) is not a direct penalization of the ML optimization. More specifically, minimizing Equation (4) is equivalent to minimizing the following penalized ML:*

$$\mathcal{L}(\mathbf{K}|\mathbf{X}) - \lambda \|\mathbf{K}\|_1 - \frac{\mu}{n} \sum_{c=1}^{k} \left\| (D^c)^{-1/2} \left(\mathbf{K}^{cons,c} - \mathbf{K}^c \right) \right\|_2^2$$

where

- $\|\mathbf{K}\|_1 = \sum_{c=1}^k \|\mathbf{K}^c\|_1 = \sum_{c=1}^k \sum_{j,j'=1}^p |\mathbf{K}^c_{jj'}|$;
- $D^c = \mathbb{D}iag(\mathbf{K}^c_{11}, \mathbf{K}^c_{22}, \dots, \mathbf{K}^c_{pp})$;
- $\mathbf{K}^{cons,c}_{j\backslash j} = \sum_{t=1}^k \frac{n_t}{n} \frac{\mathbf{K}^c_{jj}}{\mathbf{K}^t_{jj}} \mathbf{K}^t_{j\backslash j}$.

Note that, as explained in [5], estimating $(K^c_{jj})_j$ is not relevant to unveil the graph structure so, in practice, these values are set equal to $\widehat{\Sigma}^{-1}_{jj}$. Hence, from the ML point of view, there is no definition of a consensual concentration matrix since this quantity depends on the sample (the average is weighted differently depending on the sample).

In practice, in every task, the variables are previously scaled and K^c_{jj} are all set equal to one, which leads to the following equivalent formulation of the optimization problem

$$\mathcal{L}(\mathbf{K}|\mathbf{X}) - \lambda\|\mathbf{K}\|_1 - \frac{\mu}{n}\sum_{c=1}^k \left\|\mathbf{K}^{cons} - \mathbf{K}^c\right\|_2^2,$$

where $\mathbf{K}^{cons}_{j\backslash j} = \sum_{t=1}^k \frac{n_t}{n} \mathbf{K}^t_{j\backslash j}$.

Remark 3 *The penalty of [17] can be re-written as:*

$$\begin{aligned}
\sum_{c\neq c'} \|\mathbf{K}^c_j - K^{cons}_j + \mathbf{K}^{cons}_j - \mathbf{K}^{c'}_j\|_2 &= \\
\sum_{c\neq c'} \left(\|\mathbf{K}^c_j - \mathbf{K}^{cons}_j\|_2 + \|\mathbf{K}^{c'}_j - \mathbf{K}^{cons}_j\|_2 + 2\langle \mathbf{K}^c_j - \mathbf{K}^{cons}_j, \mathbf{K}^{cons}_j - \mathbf{K}^{c'}_j\rangle\right) &= \\
(k-1)\sum_{c=1}^k \|\mathbf{K}^c_j - \mathbf{K}^{cons}_j\|_2 + 2\sum_{c\neq c'} \langle \mathbf{K}^c_j - \mathbf{K}^{cons}_j, \mathbf{K}^{c'}_j - \mathbf{K}^{cons}_j\rangle &
\end{aligned}$$

Then, in the case of the averaged consensus, an edge (j, j') is in the consensus network if and only if it is in at least one of the condition-specific networks. In *particular, for $k = 2$, $\mathbf{K}^1_{jj'} - \mathbf{K}^{cons}_{jj'} \neq 0$ means that (j, j') is an edge in the consensual network and is not an edge in the network specific to condition 1. It is thus also an edge in the network specific to condition 2 (as there is only two conditions). In conclusion, when $k = 2$, $\langle \mathbf{K}^2_j - \mathbf{K}^{cons}_j, \mathbf{K}^1_j - \mathbf{K}^{cons}_j \rangle = 0$ and thus the consensus penalty is very similar to the penalty proposed in [17]. However, for $k > 2$, the situation might be more complicated: a condition-specific edge can be specific to more than one condition and thus the equality $\langle \mathbf{K}^c_j - \mathbf{K}^{cons}_j, \mathbf{K}^{c'}_j - \mathbf{K}^{cons}_j \rangle = 0$ is no more guaranteed for the averaged consensus. Conversely, nullity of the scalar product (and thus the equivalence between the consensus penalty and the penalty proposed by [17]) would be obtained, for instance, if an edge that is specific to a condition is present in only one of the condition-specific networks. This property does not seem to be desirable on a biological point of view.*

3 Computational aspects

This section will provide computational details on the **cLasso** methods. First, the method used to solve the optimization problems introduced above is described and then, a bootstrap approach is introduced to help decreasing the false positive rate and to help increasing the prediction accuracy when dealing with small sample size problem.

3.1 cLasso optimization

The **cLasso** problem is solved by minimizing the p sub-problems of Equations (5) and (6). The objective function of all the problems that can be decomposed into:

a convex part $\mathcal{C}(\beta_j) = \frac{1}{2}\beta_j^T \mathcal{Q}^1_j(\mu)\beta_j + \beta_j^T \mathcal{Q}^2_j(\mu)$, convex in $(\beta_j)_j$ and that does not depend on λ;

a norm penalty $\mathcal{P}(\beta_j) = \sum_c \frac{1}{n_c}\|\beta_j\|_1$ that is non differentiable at 0, with respect to β_j.

The non differentiability of $\mathcal{P}$ shrinks the LASSO estimate toward 0 and potentially sets $\beta_{jl} = 0$ for several indexes l, as explained in [22]. In [22], the LASSO optimization problem is solved by a quadratic programming method, which is used to perform the estimation of the $(\beta_{jl})_l$ together with a variable selection. Since then, several authors have proposed more efficient approaches to solve the LASSO optimization problem: [11] developed the so-called "shooting algorithm" that starts from an unconstrained least-square solution and uses a coordinate descent. Unfortunately, this algorithm is not applicable in the case of sparse problems as soon as $n < p$. Others proposed to use differentiable approximations of $\mathcal{P}$, such as [14] that takes advantage of the approximation $\|\beta\|_1 \simeq \sum_j \sqrt{\beta_j^2 + \epsilon}$. Finally, [18] uses a method that is efficient for medium-size problems and suited to the case $n < p$. We used a similar strategy, which is close to the one described in [5]: it is based on a greedy update of an "active set" that progressively gathers together all non-zeros coefficients of the different sub-problems. At each step of the algorithm, the coefficients are estimated only for the variables that are included in the active set.

More precisely, for a fixed value of λ, starting from a vector $\beta_j^{\mathcal{A}}$ of non-zero coefficients on the active set $\mathcal{A}$, the method first solves the so-called "master problem" given by Equations (5) or (6), which is differentiable, because, by definition, the coefficients of $\beta_j^{\mathcal{A}}$ are not null. This is done by using the sub-gradient $\partial_{\beta_j}[\mathcal{C}(\beta_j) + \lambda\mathcal{P}(\beta_j)]$. Then, the set of active variables is updated by adding the variables that violate the most the first-order optimality condition. The algorithm stops when

- for all $l \in \mathcal{A}$, $\beta_{jl} \neq 0$ and

$$\left[\partial_{\beta_j}\left[\mathcal{C}(\beta_j) + \lambda\mathcal{P}(\beta_j)\right]\right]_l = \left[\mathcal{Q}_j^1(\mu)\beta_j + \mathcal{Q}_j^2(\mu)\right]_l + \lambda\,\mathrm{sign}(\beta_{jl}) = 0$$

- for all $l \in \overline{\mathcal{A}}$, $\beta_{jl} = 0$ and

$$\left|\left[\mathcal{Q}_j^1(\mu)\beta_j + \mathcal{Q}_j^2(\mu)\right]_l\right| \leq \lambda$$

Further details on the method can be found in [5], that uses the same optimization scheme for the so-called “intertwined LASSO” method.

Finally, the method is applied to a whole set of λ values, starting from largest (i.e., from the one that yields to the strongest constrain) and using the optimal β_j as a prior for solving the problem with the next smaller λ. This method is implemented in the R package **therese**, that can be downloaded from `http://therese-pkg.r-forge.r-project.org/`.

3.2 Bootstrapped cLasso

As demonstrated in [2], the LASSO converges to the selection of all the variables included in the true model (true positives) with probability one but asymptotically selects all other variables (false positives), with a strictly positive probability. In practice, this means that using the LASSO algorithm yields to a rather high number of false positive edges in the network estimation. To overcome this difficulty, [2] proposes the so-called “Bolasso” method, that combines LASSO performed on bootstrap samples. Bootstrapping [8] is a resampling technique that consists in creating new samples of the same size as the original by sampling randomly with replacement from the original dataset. Its aim is to estimate the sampling distribution of almost any statistics and thus to estimate the accuracy for these statistics. In Bolasso, LASSO is run on a large number of bootstrap samples and the intersection of the variables selected in every bootstrap sample are finally kept. It is proved that this approach is a consistent model selection method.

Hence, in order to improve the false positive rate of the approach described above, we use a similar methodology, only taking into account the fact that the typical sample size in transcriptomic experiments is far from being close to the asymptotic case. More precisely, instead of intersecting the edges selected in every bootstrap sample, the number of times an edge is selected by all computations run on each bootstrap sample is used as a quality measure of the edge. Only the most frequently selected edges, those that are selected more than a given number denoted by $T_2 \in \{1, \ldots, N_{\text{boot}}\}$, are finally included in the estimated network.

In practice, for every bootstrap sample, Equation (5) or Equation (6) is solved for a list of several values of λ and a fixed value for μ, using the method described in Section 3.1. A given value of λ, depending on the bootstrap sample, is retained which corresponds to the first time in the path (i.e., to the largest λ) for which the number of estimated edges is larger than a target value, T_1. T_1 is fixed to a rather high value to avoid missing relevant edges. The complete procedure is described in Algorithm 1.

The impact of T_1 and T_2 is discussed further in the simulations of Section 4.1.2.

Algorithm 1 Bootstrap cLasso

```
1: require:
   list of genes $\{1,\dots,p\}$;
   list of individuals $\{1,\dots,n\}$
   individuals' sample number $c_1,\dots,c_n$ with $c_i \in \{1,\dots,k\}$
   gene expressions $\mathbf{X}$ (dimension $n \times p$)
   parameters $\mu \in \mathbb{R}$ ($L_2$-regularization parameter) and $T_1$, $T_2 \in \mathbb{N}$ (number of edges selected)
2: initialize: $\forall\, c \in \{1,\dots,k\}$, $\forall\, j, j' \in \{1,\dots,p\}$, $N^c(j,j') \leftarrow 0$
3: for $b = 1 \to P$ do
4:     Sample at random with replacement in $\{1,\dots,n\}$ return bootstrap sample $B_b$
5:     Use $B_b$ to solve Equation (6) or Equation (5) for a full set of $\lambda$ values return $(\beta_j^{c,\lambda,b})_{j,c,\lambda}$
6:     Find $\lambda_{\max} := \arg\max_\lambda \left\{ \left( \sum_{j,j',c} \mathbb{I}_{\beta_j^{c,\lambda,b} \neq 0} \right) > T_1 \right\}$ return $(\beta_j^{c,b})_{j,c} := (\beta_j^{c,\lambda_{\max},b})_{j,c}$
7:     for all $j, j' \in \{1,\dots,p\}$ do
8:         if $\beta_{j,j'}^{c,b} \neq 0$ then
9:             $N^c(j,j') \leftarrow N^c(j,j') + 1$
10:        end if
11:    end for
12: end for
13: return List of edges for sample $c$: $\{(j,j') : N^c(j,j') > T_2\}$
```

4 Application

The simulations described in this section have been performed using R version 3.0 and the packages **glasso**[2], **SIMoNe**[3] and **JGL**[4]. Bootstrap was performed using a parallel implementation with the package **doMC**.

[2] http://cran.r-project.org/web/packages/glasso/index.html
[3] http://stat.genopole.cnrs.fr/logiciels/simone
[4] http://cran.r-project.org/web/packages/JGL/index.html

4.1 Simulated data

The method is first illustrated on simulated data. These experiments use one of the graphs provided at http://www.comp-sys-bio.org/AGN/data.html and created by Pedro Mendes (Virginia Bioinformatics Institute and State University; see [12]). More precisely, the graph "scale-free Century 007" [5] was used to test the method. This network has 100 nodes (corresponding genes) and 200 edges (corresponding to gene interactions): the density of the (undirected) network is thus approximately equal to 4%. The term "scale free" indicates that the network has been generated from a preferential attachment model, as described in [3]. Additionally, the edges of the network are colored: half are "red" and half are "blue", which will differentiate a positive from a negative correlation between two variables.

4.1.1 Data generation

Several artificial expression datasets were generated from the graph described above. More precisely,

- k child networks were created by randomly rewiring a given ratio r of the edges of the original network. Hence, two child networks have approximately $100(1-2r)\%$ of shared edges. Loops and multiple edges were forbidden during the rewiring process but the color of the edges was preserved. Each of these k networks is used to model one experimental condition;

- n_c expression data were then generated from a Gaussian multivariate variable with a covariance matrix Σ for which the conditional dependency structure

[5]http://www.comp-sys-bio.org/AGN/Century/index.html

corresponded to one of the child network. In addition, the edge colors were used to define the sign of the partial correlation: blue edges corresponded to a negative partial correlation (mimicking inhibition) and red edges (mimicking activation) to a positive one.

Several experiments were designed with various values for k (2, 4 or 5), r (varied between 5%, 10% and 20%) and the respective sample sizes $n_1, \ldots, n_k$: 2×20, 2×30, 2×50, 5×20 and 4×30. Only small sample sizes (no less than 50 observations) were used to fit realistic experimental conditions in which only a few observations per condition are generally available. The resulting child networks had no more than 40% of different edges.

Figure 1 illustrates the generation process on an example: the "scale free Century 007" graph is displayed as well as two of its children, obtained by rewiring 5% of the edges.

4.1.2 Bootstrap analysis

In this section, we investigate the effect of T_1 and T_2 on the performance of the algorithm. This analysis is made using the results obtained from the expression data generated with 5% of rewired edges, 2 conditions, each containing 20 observations.

For this network, 100 bootstrap samples were extracted: this number is low compared to standard recommendations but, for one hand, the approach is computationally expensive and, for the other hand, a previous work [1] showed that the benefit of bootstrapping was achieved with the combination of 30 to 40 bootstrap samples. Also, several values of T_1 and μ have been tested: $T_1 \in \{250,\ 300,\ 500\}$ and $\mu \in \{0.1,\ 1\}$. The performance of the different parameters are compared by

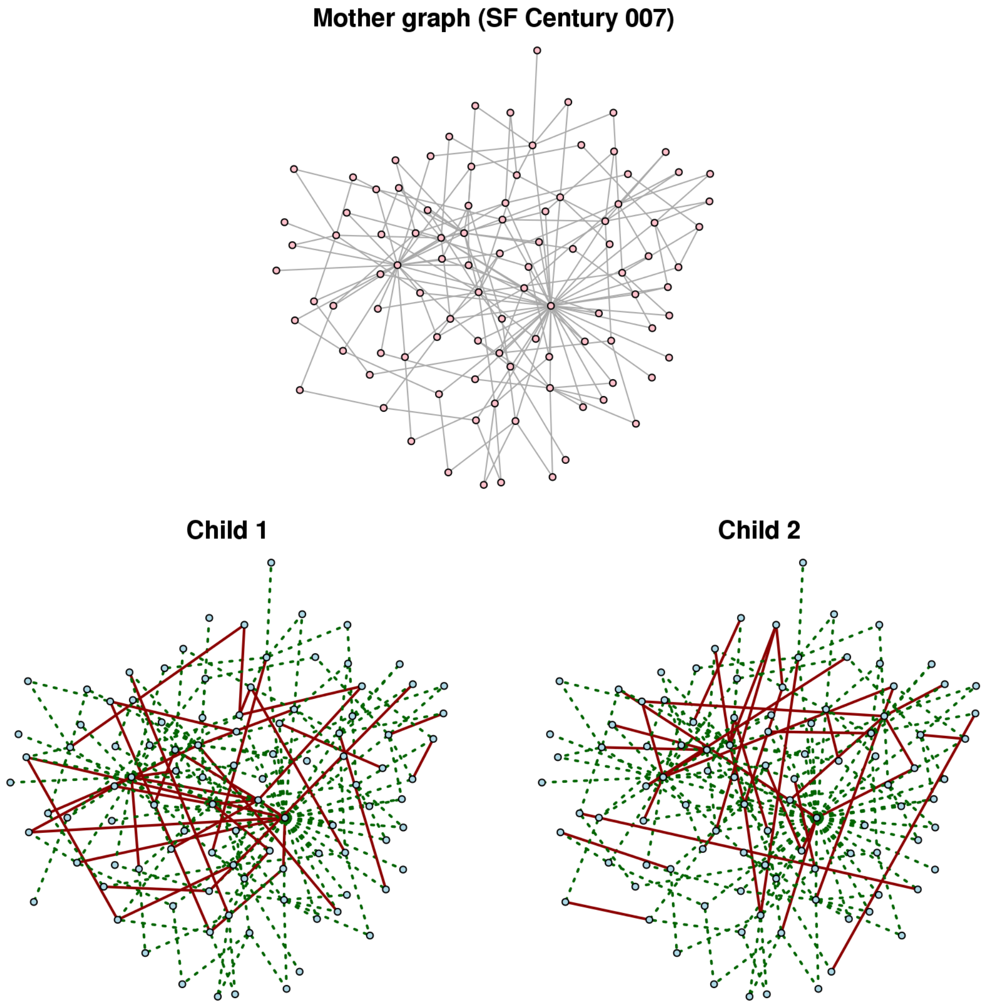

Figure 1: The "scale free Century 007" graph and two resulting child networks, obtained with 5% of rewired edges. Green dotted edges are shared edges whereas red solid ones are condition specific edges. The vertex positions result from a force-directed placement algorithm as in [10] and are common to all three networks so that the edges can easily be compared.

means of the F statistics:

$$F = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{ recall}},$$

where the precision is the ratio of retrieved edges that are in the true network (true positive edges among positive edges) and the recall is the ratio of true edges that are retrieved by the method (true positive edges among the edges in the original network). F is the harmonic mean of the precision and of the recall and computes a trade-off between the two quantities.

For each condition and each pair of parameters $\{T_1,\ \mu\}$, the F statistics were calculated along the precision/recall values obtained for different values of values of T_2 (bootstrap estimation). Then, the pairs of parameters $\{T_1,\ \mu\}$ were compared based on the averaged F over the conditions: the "best pair" of parameters is the one that maximizes the maximum averaged F along the path of T_2 values, the maximum F being used as a way to find the best compromise between precision/recall. According to this method, the best pair $\{T_1,\ \mu\}$ for the expression data described above was $\{500,\ 1\}$, as shown in the level plot of Figure 2

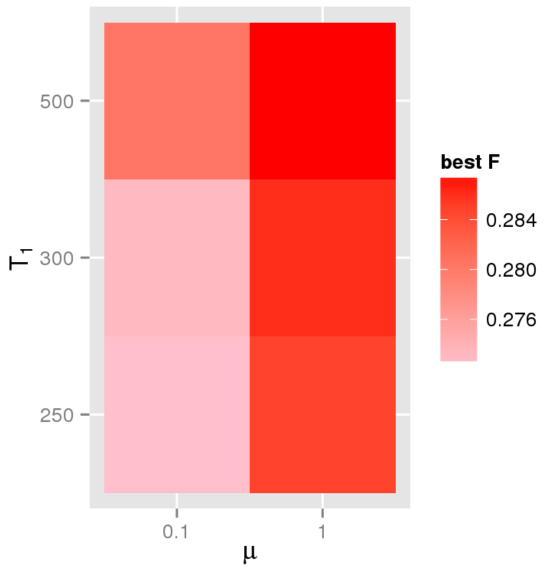

Figure 2: Maximum F along the path of T_2 values for different parameters μ and T_1

When the value of T_1 is set to a rather high value, 500 (which is much larger than the true number of edges), and when μ is equal to 1, Figure 3 gives an indication on the influence of T_2 on the density of the inferred networks. The histogram displays the distribution of the number of times a given edge is chosen by the algorithm over the 100 bootstrap samples. Notice that only a few edges are very frequently selected by the bootstrap method, whereas the targeted density of 4% is obtained by keeping edges that are selected only about 40 times (i.e., 40% of the bootstrap samples).

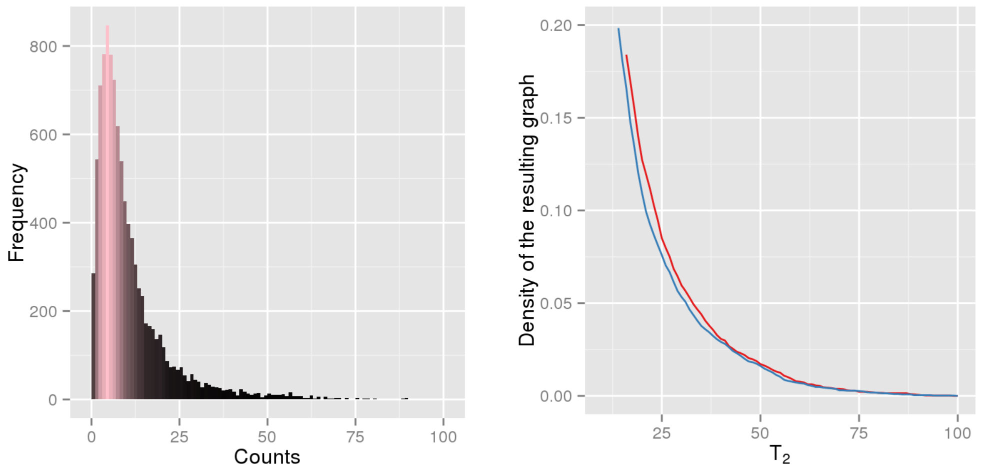

Figure 3: Distribution of the number of times an edge is selected over 100 bootstrap samples for the first condition (left) and Evolution of the density versus T_2 for the 2 conditions (right). $T_1 = 500$, $\mu = 1$

Figure 4 displays the precision/recall curve. Two points are emphasized on this figure: they correspond to the maximum F on the curve. The maximum F are obtained by keeping edges that are selected at least 40/45 times (approximately) over the 100 bootstrap samples and correspond to a precision about 25% and a recall about 30%. These points give inferred networks with a resulting density slightly lower that the true network density (2.5-3.5% instead of 4%). This illus-

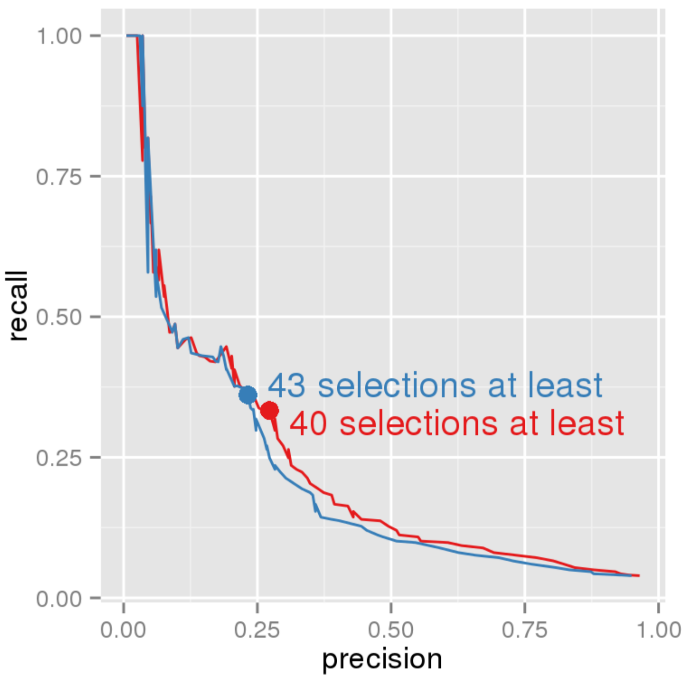

Figure 4: Precision/Recall curve (with varying T_2) for the total number of edges of the true child networks compared to the corresponding inferred network. $T_1 = 500$, $\mu = 1$ (each curve correspond to one of the $k = 2$ child networks.

trates the fact that, if there is a prior knowledge on a targeted density, a good strategy could be to choose T_2 so that resulting networks fit this targeted density.

When the value of T_1 is equal to a smaller value (250 which is larger than the true number of edges), the evolution of the density versus T_2 and the precision/recall curve are given in Figure 5. The conclusions are very similar except, of course, that for a given value of T_2, the densities of the resulting networks are lower. Otherwise, the distributions of the number of times a given edges is selected by the algorithm in the bootstrap samples are quite similar and the best F value is also obtained for networks that have densities slightly slower than the true density.

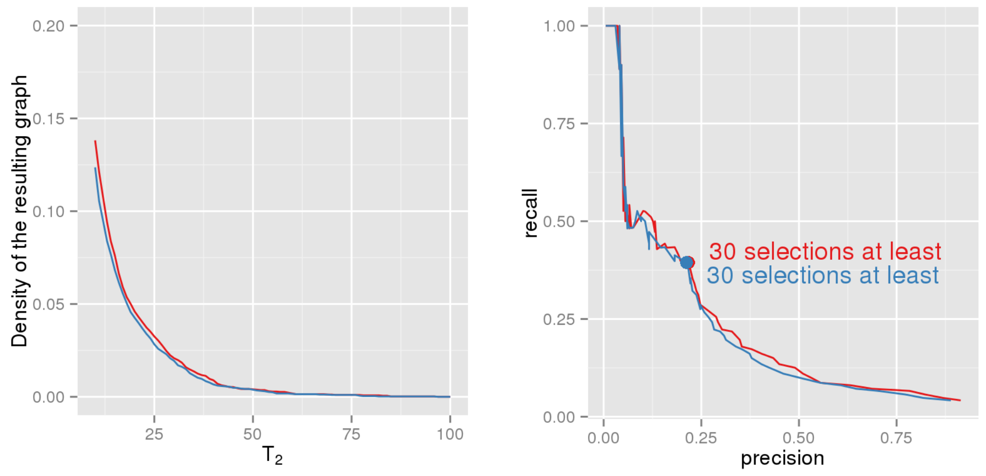

Figure 5: Evolution of the density versus T_2 for the 2 conditions (left) and Precision/Recall curve (with varying T_2) for the total number of edges of the true child networks compared to the corresponding inferred network. $T_1 = 250$, $\mu = 1$

However, as shown in Figure 2, T_2 is a less important parameter for the method performance, as compared to μ. Optimal parameters, according to the maximum F statistics, are given in Table 1 for all simulations. As expected, μ needs be smaller in the case where the two conditions correspond to more different networks (i.e., when the number of rewired edges is larger) but generally, using a rather high value for T_1 is the strategy that provides the best results. The effect of the bootstrap on the performance is shown in the last column of this table, which contains the percentage of increase of the corresponding maximum F compared to the direct approach. Bootstrap only improves the performances when the percentage of rewired edges is moderate (lower than 10%) or when there are many different conditions. The counter-performance of bootstrapping could be explained by the fact that it enforces the joint effect and thus fails to estimate edges specific to the condition, that are less numerous in those cases. Additionally, this might be a very high-dimension issue [23]: when the $\frac{n}{p}$ ratio allows us to draw model inference but is at the limit of producing reasonably accurate estimates, the use of a bootstrap procedure produces a set of highly unstable estimates, which lead to fewer robust estimated edges. As a consequence, the model estimate might focus on those edges which are supported by many conditions and does not detect finer pattern of dependencies in the data.

	μ	T_1	% of improvement of bootstrapping
network sizes	**rewired edges: 5%**		
20-20	1	500	28.80
30-30	1	300	20.15
50-50	1	300	13.44
20-20-20-20-20	1	500	83.75
30-30-30-30	0.1	500	42.67
network sizes	**rewired edges: 10%**		
20-20	0.1	250	18.35
30-30	0.1	500	16.17
50-50	1	250	4.23
20-20-20-20-20	0.1	500	55.48
30-30-30-30	0.1	250	29.56
network sizes	**rewired edges: 20%**		
20-20	0.1	300	-17.86
30-30	1	500	-7.97
50-50	0.1	300	-7.83
20-20-20-20-20	0.1	500	10.27
30-30-30-30	1	500	13.48

Table 1: Best parameters of the bootstrap **cLasso** for each simulation according to the maximum F along the path of T_2 values, and percentage of increase of the best F value compared to the direct (i.e., un-bootstrapped) approach.

4.1.3 Performance comparison

In this section, **cLasso** is compared to alternative methods for inferring graphs from expression data. More precisely, for each expression dataset described in Section 4.1.1, the following methods are applied to infer the k conditional dependency networks corresponding to the k different conditions:

- the graphical Lasso method, as described in [9] and hereafter denoted by **gLasso**: the k networks corresponding to the k different conditions are inferred independently. Hence, the comparison with this method aims at showing the effect of jointly inferring the networks instead of independently;
- the intertwined Lasso, the cooperative Lasso and the group Lasso methods, as described in [5] and hereafter denoted by **iLasso**, **coopLasso** and **groupLasso**, respectively. These methods are used to provide a comparison with other joint inference methods. Also notice that the data generation provides sign-coherent networks (i.e., the different child networks are very likely to have the same sign for partial correlations corresponding to shared edges), which should favor the cooperative Lasso method;
- the fused graphical Lasso, as described in [6], denoted by **fgLasso**. After a few tests, the second regularization parameter, which controls the similarity accross conditions, was set to the value 0.1 for all simulations;
- the consensus Lasso method, as described in Section 2. The two choices of consensus described in Section 2.3 are tested with, for *a priori* network:
  - the mother network (i.e., the true network used to generate the child networks, which is never known in practice but is the closest thing we

have in this simulation from a bibliographic network),

- *or*, for comparing a naive two-step approach with the averaged consensus described in Section 2.3.2, a network which is the mean over the conditions of independent estimations (i.e., estimations obtained with $\mu = 0$).

These methods are denoted by **cLasso (m)** (for the averaged consensus method described in Section 2.3.2), **cLasso (p)** (for the method using the true prior) and **cLasso (2)** (for the naive two-step approach that uses a mean over conditions of independant estimations). Notice that the method using as a prior the mother network is clearly favored in this comparison, since even if the child networks are not identical to the mother network, they are very related to it. The comparison with this method should be used to understand what is the effect of integrating true prior knowledge in the estimation. μ was set equal to 1.

For each method, the inference is performed for a whole path of λ values and the corresponding precision and recall are calculated for each value of λ. A bootstrap version with 100 bootstrap samples of each of these methods is also implemented with $T_1 = 500$. The number of times a given edge in a given condition is selected is then used to calculate precision/recall values for different values of T_2.

Precisions are recalls are calculated by comparing the estimated condition-specific networks with the children networks they are generated from. We do not compare directly the consensus network with the mother network because we are interested in testing the ability of the method to estimate the common edges as well as the condition-specific edges. The F statistics is used as a way to compare the different methods, as in Section 4.1.2. First, averaged F, over the different conditions, are calculated along the precision/recall values obtained for different values of λ (direct estimation) or for different values of T_2 (bootstrap estimation). Then, the maximum of these values (for recall and precision values larger than 0.05, to avoid extremely bad values of the precision or of the recall) is used as a way to compare the performance of the different methods. The results are given in Tables 2 (direct estimation) and 3 (bootstrap estimation), for each of the 6 methods described above (the best-case scenario for the bootstrapped **cLasso** corresponds to the parameters already listed in Table 1).

Method	**gLasso**	**iLasso**	**groupLasso**	**coopLasso**	**fgLasso**	**cLasso (m)**	**cLasso (p)**	**cLasso (2)**
network sizes	**rewired edges: 5%**							
20-20	0.19	0.27	0.23	0.28	0.26	0.22	**0.84**	0.21
30-30	0.28	0.35	0.32	0.35	0.32	0.31	**0.86**	0.30
50-50	0.36	0.47	0.48	0.49	0.47	0.43	**0.88**	0.40
20-20-20-20-20	0.19	0.34	0.34	0.41	0.43	0.23	**0.84**	0.23
30-30-30-30	0.30	0.46	0.48	0.51	0.55	0.36	**0.88**	0.35
network sizes	**rewired edges: 10%**							
20-20	0.19	0.24	0.22	0.26	0.22	0.23	**0.78**	0.21
30-30	0.27	0.35	0.33	0.35	0.34	0.31	**0.81**	0.29
50-50	0.41	0.48	0.45	0.46	0.49	0.45	**0.82**	0.41
20-20-20-20-20	0.20	0.30	0.24	0.36	0.35	0.23	**0.74**	0.23
30-30-30-30	0.28	0.39	0.35	0.40	0.45	0.31	**0.79**	0.32
network sizes	**rewired edges: 20%**							
20-20	0.21	0.22	0.19	0.21	0.21	0.23	**0.58**	0.21
30-30	0.28	0.31	0.27	0.31	0.30	0.31	**0.67**	0.33
50-50	0.42	0.43	0.41	0.44	0.45	0.43	**0.68**	0.40
20-20-20-20-20	0.20	0.26	0.22	0.25	0.26	0.22	**0.63**	0.23
30-30-30-30	0.27	0.35	0.28	0.35	0.35	0.29	**0.63**	0.31

Table 2: Summary of the performance for the different method in terms of the maximum value of the F statistics. The best method for each couple of percentage of rewired edges and network sizes, is emphasized with bold face.

Method	**gLasso**	**iLasso**	**groupLasso**	**coopLasso**	**fgLasso**	**cLasso (m)**	**cLasso (p)**	**cLasso (2)**
network sizes	**rewired edges: 5%**							
20-20	0.26	0.27	0.28	0.29	0.29	0.29	**0.85**	0.28
30-30	0.31	0.34	0.36	0.34	0.36	0.37	**0.86**	0.35
50-50	0.46	0.48	0.48	0.47	0.48	0.49	**0.88**	0.47
20-20-20-20-20	0.38	0.34	0.44	0.43	0.44	0.43	**0.89**	0.41
30-30-30-30	0.48	0.44	0.51	0.53	0.53	0.51	**0.89**	0.51
network sizes	**rewired edges: 10%**							
20-20	0.25	0.23	0.25	0.27	0.27	0.25	**0.79**	0.27
30-30	0.33	0.36	0.35	0.36	0.38	0.35	**0.80**	0.34
50-50	0.45	0.47	0.46	0.44	0.48	0.46	**0.82**	0.43
20-20-20-20-20	0.32	0.30	0.36	0.34	0.36	0.35	**0.78**	0.35
30-30-30-30	0.36	0.38	0.39	0.41	0.42	0.40	**0.80**	0.38
network sizes	**rewired edges: 20%**							
20-20	0.17	0.22	0.17	0.18	0.21	0.18	**0.59**	0.18
30-30	0.27	0.31	0.27	0.28	0.27	0.29	**0.67**	0.28
50-50	0.37	0.43	0.38	0.37	0.41	0.39	**0.66**	0.37
20-20-20-20-20	0.20	0.23	0.25	0.23	0.26	0.24	**0.66**	0.24
30-30-30-30	0.30	0.31	0.32	0.32	0.32	0.33	**0.63**	0.31

Table 3: Summary of the performance for the different method (bootstrap version) in terms of the maximum value of the F statistics. The best method for each couple of percentage of rewired edges and network sizes, is emphasized with bold face.

Several conclusions can be drawn from these results. For a moderate ratio of rewired edges (smaller than 10%), bootstrapping improves the performances of all methods, except for **iLasso** (also, the increase is very limited for the **coopLasso** method). The increase is particularly interesting when the sample size is small and/or the number of samples is high. On the contrary, when the ratio of rewired edges is equal to 20%, bootstrapping only improves the performances of **cLasso** with prior, and, only for 4-5 samples having the smallest sizes, of **gLasso**, **groupLasso** and **cLasso (m)**.

As expected, the overall performance is strongly increased when a relevant prior is added (the best F is often 3 times larger), which shows that this strategy should probably be used when such an information is available. When this is not the case, **fgLasso** often obtains the best results. Otherwise, **coopLasso**, bootstrap **cLasso** or **iLasso** also have good performances. Bootstrap **cLasso (m)** seems to be useful in the case of a moderate number of rewired edges and when the sample size is smaller. The naive two-step approach, which requires two estimations instead of one, often leads to deteriorated performances as compared to **cLasso (m)** which is twice faster. Finally, direct **iLasso** is advised for the largest number of rewired edges and **coopLasso** is to be preferred when the number of rewired edges is small but the sample size larger.

Computational times needed[6] for the different estimations are very different:

- the time needed to estimate one of the condition-specific networks with **glasso** (independent estimations with graphical LASSO as described in [9]) is approximately equal to 1 second for 25 values of λ (to be multiplied by the

[6]Computational times are reported for a 4-core laptop, Intel(R) Core(TM) i5-3360M CPU @ 2.80GHz, RAM 8Go DDR3, with OS Kubuntu Linux 12.04.

number of conditions);

- the time needed to estimate 5 joined networks with **simone** (implementing the methods described in [5]; the time is reported for "cooperative LASSO") is approximately equal to 1 minute 30 seconds for 100 values of λ;

- the time needed to estimate 2 joined network with **JGL** (implementing the methods described in [6]; the time is reported for "fused LASSO") is approximately equal to for 25 values of λ and the time needed to estimate 5 joined networks with this method is approximately equal to 2 hours 30 minutes. Notice that the path of λ has been performed manually as this package is the only one that does not propose a regularization path for the sparse parameter;

- finally, the time needed to estimate 5 joined networks with **therese** is approximately equal to 2 minutes 30 seconds for 100 values of λ (and a little bit less than twice this value for the naive two-step approach).

4.2 Real data: effect of a diet on regulatory network

As an application to a real biological data set, we analyzed gene expression data described in [24]. More precisely, the expression of 221 genes are used. These were obtained for 204 obese women before and after a 8-week low-calorie die (LCD) with the objective of more than 8% weight loss. Considering the two time steps of the analysis as two different conditions, we used the bootstrapped **cLasso** approach with $\mu = 1$ and $T_1 = 1000$ (that corresponded to a targeted density of approximately 4%). The choice of a rather high regularization parameter μ was directed by the will to emphasize only a few different edges between the two conditions and hence to focus on the most relevant differences between the two conditions. Notice that the possibility to monitor μ allows to infer networks that are more or less consensual, depending on what your prior is. The choice for T_1 was directed by the fact that we wanted to obtain very sparse networks, easily readable for the biologist, which, in the case of approximately 200 nodes, requires to have a very low density. 100 bootstrap samples where used to estimate the edges by the **cLasso** approach.

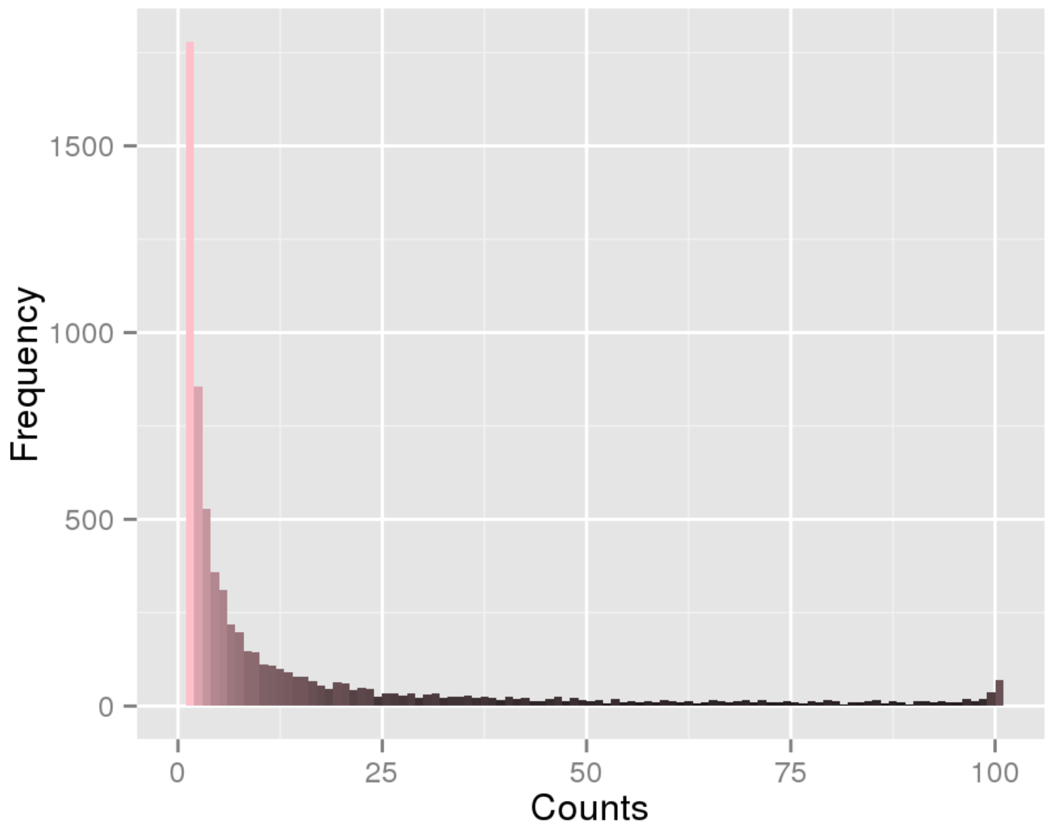

Figure 6: Distribution of the number of times an edge is selected over 100 bootstrap samples for the expression data before the diet.

The distribution of the number of times an edge is selected for expression data *after* the diet was very similar to what was found *before* the diet. Hence, in order to favor a high precision (at the cost of maybe a low recall), only the edges that appeared in at least 80 bootstrap samples were selected. This yielded to networks having respectively 316 and 315 edges (with a density about 1.3%). These networks had 292 edges in common (i.e., approximately 90% of the total number of inferred edges).

The histogram of the number of times a given edge is selected over the 100 bootstrap sample is given in Figure 6 (for the network corresponding to gene expression data *before* the diet). It has to be noted that pairs of variables that were never selected over the 100 bootstrap samples have been removed from the histogram (it corresponded to approximately 70% of the 221×220 potential edges).

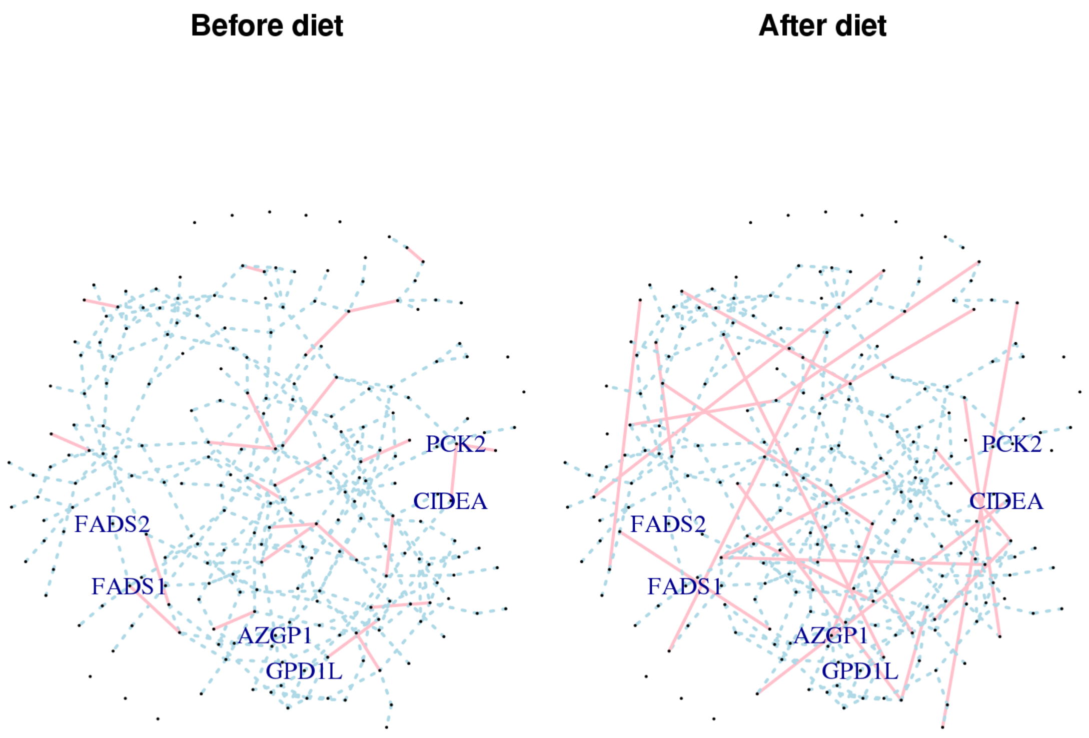

Figure 7: Networks inferred by bootstrapped **cLasso**: before (left) and after low-calorie diet (right). Blue edges are shared edges and pink edges are condition specific edges. Some gene names are given, that are commented in the text.

Other shared edges are highly probable, such as the one between *AZGP1* and *GPD1L* which are two known biomarkers of the metabolic syndrome [24]. However, quite interestingly, at least one condition specific edge is also expected: the genes *PCK2* and *CIDEA* are the best biomarkers, among this set of genes, for the weight loss, and the main difference between the two conditions is indeed the weight loss [24].

The resulting networks are displayed in Figure 7. They have approximately 92% of their edges in common. The full biological validation of such a network is unrealistic (because of the very limited knowledge available in this area) but some of the interactions make sense. For instance, some regulatory relationships shared between the two conditions are already known, such as the relation between *FADS1* and *FADS2*, which encode two desaturase enzymes from the same cluster gene with similar regulation by dietary composition [15].

5 Conclusion

We have proposed the **cLasso** method, which is used for jointly inferring networks in the case of multiple and dependent expression data. This method relies on the definition of a consensual solution, which in our case, is simply the mean between the different conditions. The different networks are forced toward this consensus by a L_2-penalty whilst the sparsity of the solution is handled by an additional L_1 penalty. The solution proposed in this paper can be reformulated as a LASSO problem similar to the ones described in [16, 5] and the method is implemented in the **R** package **therese**. Experiments were conducted, using a bootstrapped approach based on the **cLasso** method and showed that this method is reliable.

Future work should address the issue of unbalanced sample sizes between conditions, and of the choice of μ: a naive selection based on out-of-bag MSE has been proven inefficient so far for selecting the best value. However, this parameter can also be useful for the biologist to include prior knowledge about how similar the condition-specific networks should be: using different μ provide a family of solutions with different fractions of common edges, among which the biologist is free to choose.

Acknowledgements

We thank the two anonymous reviewers for their useful comments and suggestions. We also thank Jean-Michel Poggi for thorough remarks that gave useful hints to use the method for integrating prior knowledge and Nicolas A. Edwards who helped testing several joint inference methods during his internship.

Authors' biography

Nathalie Villa-Vialaneix obtained her PhD in applied mathematics in 2005 at the Université Toulouse II. She is associate professor (Maîtresse de Conférences) at the Université de Perpignan and member of the SAMM research team (Université Paris 1). She also held a temporary position, as a researcher, at the INRA (French National Institute for Agricultural Research) in Toulouse. She is author or co-author of more than 50 peer reviewed scientific papers in journals, conference proceedings and books. Her research activities focus on machine learning, neuronal methods, functional data analysis, graph mining and network inference, with applications to social and human sciences and to biology.

Matthieu Vignes is currently an associate researcher in the Mathematics and Computer Science department of the French National Institute for Agricultural Research. He received his B.Sc and M.Sc in Mathematics from the Ecole National

Supérieure de Lyon and the Université Claude Bernard Lyon 1, France, and his PhD from the Université Joseph Fourier Grenoble 1, France. His research interest are focused on statistical and algorithmic aspects concerned with gene regulatory network reconstruction from high-dimensional data sets.

Nathalie Viguerie studied Molecular Pharmacology and received her PhD from Paul Sabatier University at the Digestive Pathologies Laboratory, Toulouse, France, in 1988. That fall, she was appointed as principal investigator at INSERM, the only French public organization entirely dedicated to biological, medical and public health research. She is currently at the Metabolism and Obesity Department at the Institute of Cardiovascular and Metabolic Diseases (I2MC), INSERM U1048 where she manages clinical and molecular studies at the Obesity Research Laboratory in the Rangueil Hospital of Toulouse. Her research interest covers various areas including the study of mechanisms linking an excess of fat to insulin resistance in order to find out novel therapeutically relevant targets. She was co-coordinator of the genomics research line in the 6th framework project "DIOGENES : Diet, Obesity and Genes". She is author and co-author of 79 original international papers, 2 book chapters and 6 review articles.

Magali San Cristobal is a senior scientist at the French National Institute of Agronomical Research (INRA). She received her PhD degree in Applied Statistics from University of Toulouse, France. Her current research is in the area of statistics applied to integrative biology, population and quantitative genetics.

References

[1] D. Allouche, C. Cierco-Ayrolles, S. de Givry, G. Guilermin, B. Mangin, T. Schiex, J. Vandel, and M. Vignes. A panel of learning methods for the reconstruction of gene regulatory networks in a systems genetics context. In A. de la Fuente, editor, *Verification of Methods for Gene Network Inference from Systems Genetics Data*. Springer, 2013.

[2] F. Bach. Bolasso: model consistent lasso estimation through the bootstrap. In *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML)*, 2008.

[3] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[4] A. Butte and I. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 418–429, 2000.

[5] J. Chiquet, Y. Grandvalet, and C. Ambroise. Inferring multiple graphical structures. *Statistics and Computing*, 21(4):537–553, 2011.

[6] P. Danaher, P. Wang, and D. Witten. The joint graphical lasso for inverse covariance estimation accross multiple classes. *Journal of the Royal Statistical Society Series B*, 2013. Forthcoming.

[7] D. Edwards. *Introduction to Graphical Modelling.* Springer, New York, 1995.

[8] B. Efron. Nonparametric estimates of standard error: the jackknife, the bootstrap and other methods. *Biometrika*, 68:589–599, 1981.

[9] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[10] T. Fruchterman and B. Reingold. Graph drawing by force-directed placement. *Software, Practice and Experience*, 21:1129–1164, 1991.

[11] W.J. Fu. Penalized regressions: the bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.

[12] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, Singhal M., L. Xu, P. Mendes, and U. Kummer. COPASI - a COmplex PAthway SImulator. *Bioinformatics*, 22:3067–3074, 2006.

[13] N. Jung, M. Maumy-Bertrand, L. Vallat, and F. Bertrand. Inférence conjointe de réseaux de gènes dans de multiples états. In *Actes des 44èmes Journées de Statistique de la SFdS*, Bruxelles, Belgique, 2012.

[14] S.I. Lee, H. Lee, P. Abbeel, and A. Ng. Efficient l_1 regularized logistic regression. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

[15] L.M. Mangravite, K. Dawson, R.R. David, J.P. Gregg, and R.M. Krauss. Fatty acid desaturase regulation in adipose tissue by dietary composition is independent of weight loss and is correlated with the plasma triacylglycerol response. *American Journal of Clinical Nutrition*, 86(3):759–767, September 2007.

[16] N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistic*, 34(3):1436–1462, 2006.

[17] K. Mohan, J.Y. Chung, S. Han, D. Witten, S.I. Lee, and M. Fazel. Structured learning of Gaussian graphical models. In *Proceedings of NIPS (Neural Information Processing Systems) 2012*, Lake Tahoe, Nevada, USA, 2012.

[18] M.R. Osborne, B. Presnell, and B.A. Turlach. On the LASSO and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000.

[19] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, San Francisco, California, USA, 1998.

[20] J. Pearl and S. Russel. *Bayesian Networks*. Bradford Books (MIT Press), Cambridge, Massachussets, USA, 2002.

[21] J. Schäfer and K. Strimmer. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005.

[22] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, series B*, 58(1):267–288, 1996.

[23] N. Verzelen. Minimax risks for sparse regressions: ultra-high-dimensional phenomenons. *Electronic Journal of Statistics*, 6:38–90, 2012.

[24] N. Viguerie, E. Montastier, J.J. Maoret, B. Roussel, M. Combes, C. Valle, N. Villa-Vialaneix, J.S. Iacovoni, J.A. Martinez, C. Holst, A. Astrup, H. Vidal, K. Clément, J. Hager, W.H.M. Saris, and D. Langin. Determinants of human adipose tissue gene expression: impact of diet, sex, metabolic status and *cis* genetic regulation. *PLoS Genetics*, 8(9):e1002959, 2012.

[25] F. Villers, B. Schaeffer, C. Bertin, and S. Huet. Assessing the validity domains of graphical Gaussian models in order to infer relationships among compo-

nents of complex biological systems. *Statistical Applications in Genetics and Molecular Biology*, 7(2):14, 2008.

[26] J. Whittaker. *Graphical Models in Applied Multivariate Statistics.* John Wiley and Sones, Chichester, GB, 1990.