GGMselect: R package for estimating Gaussian graphical models

Annie Bouvier, Christophe Giraud, Sylvie Huet, and Nicolas Verzelen

INRA, MIA, 78352 Jouy-en-Josas Cedex, FRANCE
e-mail: Annie.Bouvier@jouy.inra.fr

Ecole Polytechnique, CMAP, UMR 7641
Route de Saclay 91128 Palaiseau Cedex, FRANCE
e-mail: Christophe.Giraud@polytechnique.edu

INRA, MIA, 78352 Jouy-en-Josas Cedex, FRANCE
e-mail: Sylvie.Huet@jouy.inra.fr

Université Paris Sud, Laboratoire de Mathématiques, UMR 8628
Orsay Cedex F-91405
e-mail: nicolas.verzelen@math.u-psud.fr

Contact: Annie.Bouvier@jouy.inra.fr

Contents

1. Introduction

Biotechnological developments in proteomics or transcriptomics enable to produce a huge amount of data. One of the challenges for the statistician is to infer from these data the regulation network of a family of genes (or proteins). Gaussian graphical models are promising probabilistic tools to achieve this challenge.
Graphical modeling is based on the conditional independence concept: a direct relation between two variables exists if those two variables are conditionally dependent given all the remaining variables. These direct relations are represented by a graph: a node is associated to each variable and an edge is set between two nodes when they are in direct relation. In the Gaussian setting, a direct relation between two variables corresponds to a non-zero entry in the partial correlation matrix, or equivalently to a non-zero entry in the inverse of the covariance matrix.
Let us consider p genes that will compose the nodes of the graph. For each gene we observe some random response such as the differential expression on microarray experiment. The p nodes of the graph are thus identified with p random variables denoted by $(X_1, \ldots, X_p)$ assumed

to be distributed as a multivariate Gaussian $\mathcal{N}(0, \Sigma)$. The graph G_Σ of conditional dependences is defined as follows: there exists an edge between nodes a and b if and only if the variables X_a and X_b are dependent given all the remaining variables. This will be denoted

$$a \overset{G_\Sigma}{\sim} b.$$

GGMselect [6] is dedicated to the estimation of the graph G_Σ on the basis of a n-sample from $\mathcal{N}(0, \Sigma)$. In the following, a graph G will be identified with the set of its edges.

GGMselect is a two-stage procedure:

1. A family $\widehat{\mathcal{G}}$ of candidate graphs is built using either some data-driven method or some prior knowledge on the true graph.
2. A graph $\widehat{G}$ is selected among this family $\widehat{\mathcal{G}}$ by minimizing an empirical criterion based on conditional least-squares.

GGMselect is specially designed to handle the case where the sample size n is smaller than the number of variables p. Its performances have been assessed in [6]. It has been shown to be consistent even when p is much larger than n, and its risk is controlled by a non-asymptotic oracle-like inequality. The assumptions needed to establish these results are weaker than those commonly used in the literature. In addition, numerical experiments have shown a nice behavior on simulated examples.

Download: `http://w3.jouy.inra.fr/unites/miaj/public/logiciels/GGMselect/`

Notations. We set $\Gamma = \{1, \dots, p\}$ and for any graph G with nodes indexed by Γ, we write $d_a(G)$ for the degree of the node a in the graph G (which is the number of edges incident to a) and $\deg(G) = \max_{a\in\Gamma} d_a(G)$ for the degree of G. We also write $\|.\|_n$ for the Euclidean norm on $\mathbb{R}^n$ divided by $\sqrt{n}$ and for any $\beta \in \mathbb{R}^p$ we define supp(β) as the set of the labels $a \in \Gamma$ such that $\beta_a \neq 0$.

2. Estimation procedure

The main inputs are:

- `X` A data matrix `X` of size $n \times p$. Each row corresponds to an independent observation of the vector $(X_1, \dots, X_p)$. We write $\mathtt{X}_a$ for the a^{th} column of `X`.
- `dmax` A vector of p integers: for each $a \in \Gamma$, `dmax`$[a]$ is the maximum degree of the node a within the graphs of the family $\widehat{\mathcal{G}}$. For each $a \in \Gamma$, `dmax`$[a]$ must be smaller than $\min(n-3, p-1)$.
- `K` A scale-free tuning parameter `K` > 1. Default value is `K` $= 2.5$.

GGMselect Algorithm

1. Build a (possibly data-driven) family $\widehat{\mathcal{G}}$ of candidate graphs.
2. Select $\widehat{G}$ as any minimizer of $\mathrm{Crit}(.)$ over $\widehat{\mathcal{G}}$:

$$\widehat{G} = \underset{G\in\widehat{\mathcal{G}}}{\operatorname{argmin}} \mathrm{Crit}(G)\ .$$

Step 2 and $\mathrm{Crit}(G)$ are described in Section 2.1 and the six families $\widehat{\mathcal{G}}$ of graphs available in the package are described in Section 2.2.

2.1. Penalized criterion $\mathrm{Crit}(.)$

For any graph G in $\widehat{\mathcal{G}}$, we associate the $p \times p$ matrix $\widehat{\theta}^G$ by

$$\widehat{\theta}^G = \mathrm{argmin}\left\{ \sum_{i,a} [\mathtt{X} - \mathtt{X}\theta]^2_{i,a}\,,\ \theta \in \Theta_G \right\},$$

where Θ_G is the set of $p \times p$ matrices θ such that $\theta_{a,b}$ is non-zero if and only if there is an edge between a and b in G. See [5] for more details.

Then, we define the criterion $\mathrm{Crit}(G)$ by

$$\mathrm{Crit}(G) = \sum_{a=1}^{p} \left[\|\mathtt{X}_a - \sum_b \mathtt{X}_b \widehat{\theta}^G_{a,b}\|^2_n \left(1 + \frac{\mathrm{pen}[d_a(G)]}{n - d_a(G)}\right)\right], \tag{1}$$

where the penalty function is defined by

$$\mathrm{pen}(d) = \mathtt{K}\,\frac{n-d}{n-d-1}\,\mathsf{EDKhi}\left[d+1, n-d-1, \left(\binom{p-1}{d}(d+1)^2\right)^{-1}\right]. \tag{2}$$

The function EDKhi$[d, N, .]$ is the inverse of the function

$$x \mapsto \mathbb{P}\left(F_{d+2,N} \geq \frac{x}{d+2}\right) - \frac{x}{d}\,\mathbb{P}\left(F_{d,N+2} \geq \frac{N+2}{Nd}x\right),$$

where $F_{d,N}$ denotes a Fisher random variable with d and N degrees of freedom. See [1] Sect.6.1 for details.

2.2. Families of candidate graphs $\widehat{\mathcal{G}}$ available in GGMselect

Six families are available in GGMselect. Depending on the option `family`, the function `selectFast` uses one or several of the families $\widehat{\mathcal{G}}_{\mathrm{C01}}$, $\widehat{\mathcal{G}}_{\mathrm{LA}}$, $\widehat{\mathcal{G}}_{\mathrm{EW}}$, $\widehat{\mathcal{G}}_{\mathrm{C01,LA}}$, $\widehat{\mathcal{G}}_{\mathrm{C01,LA,EW}}$. The function `selectQE` uses the family $\widehat{\mathcal{G}}_{\mathrm{QE}}$.
The user can also minimize the criterion (1) over his own family $\widehat{\mathcal{G}}$ by using the function `selectMyFam`.

2.2.1. C01 family $\widehat{\mathcal{G}}_{\mathrm{C01}}$ (with `selectFast`)

The family $\widehat{\mathcal{G}}_{\mathrm{C01}}$ derives from the estimation procedure proposed in Wille and Bühlmann [8].

We write $P(a,b|c)$ for the p-value of the likelihood ratio test of the hypothesis "cov$(X_a, X_b|X_c) = 0$" and set

$$P_{\max}(a,b) = \max\left\{P(a,b|c),\ c \in \{\emptyset\} \cup \Gamma \setminus \{a,b\}\right\}.$$

For any $\alpha > 0$, the graph $\widehat{G}_{01,\alpha}$ is defined by $a \overset{\widehat{G}_{01,\alpha}}{\sim} b \iff P_{\max}(a,b) \leq \alpha$ and the family $\widehat{\mathcal{G}}_{\mathrm{C01}}$ is the family of nested graphs

$$\widehat{\mathcal{G}}_{\mathrm{C01}} = \left\{\widehat{G}_{01,\alpha},\ \alpha > 0 \text{ and } d_a(\widehat{G}_{01,\alpha}) \leq \mathtt{dmax}[a] \text{ for all } a \in \Gamma\right\}.$$

C01 Algorithm

1. Compute the $p(p-1)/2$ values $P_{\mathsf{max}}(a,b)$.
2. Order them.
3. Extract from these values the nested graphs $\left\{\widehat{G}_{01,\alpha} : \alpha > 0\right\}$.
4. Stop as soon as there is a node a for which the number of neighbours exceeds `dmax[a]`.

2.2.2. **Lasso-And family $\widehat{\mathcal{G}}_{\mathrm{LA}}$ (with `selectFast`)**

The Lasso-And family $\widehat{\mathcal{G}}_{\mathrm{LA}}$ derives from the estimation procedure proposed by Meinshausen and Bühlmann [7].

For any $\lambda > 0$, we define the $p \times p$ matrix $\widehat{\theta}^{\lambda}$ by

$$\widehat{\theta}^{\lambda} = \operatorname{argmin}\left\{ \sum_{i,a} [\mathtt{X} - \mathtt{X}\theta]^2_{i,a} + \lambda \sum_{a \neq b} |\theta_{a,b}|, \text{ for } \theta \in \Theta \right\}, \tag{3}$$

where Θ is the set of $p \times p$ matrices with 0 on the diagonal. Then, we define the graph $\widehat{G}^{\lambda}_{\mathsf{and}}$ by setting an edge between a and b if both $\widehat{\theta}^{\lambda}_{a,b}$ and $\widehat{\theta}^{\lambda}_{b,a}$ are non-zero. Finally, we define the family $\widehat{\mathcal{G}}_{\mathrm{LA}}$ as the set of graphs $\widehat{G}^{\lambda}_{\mathsf{and}}$ with λ large enough to ensure that $d_a(\widehat{G}^{\lambda}_{\mathsf{and}}) \leq \mathtt{dmax}[a]$ for all $a \in \Gamma$:

$$\widehat{\mathcal{G}}_{\mathrm{LA}} = \left\{ \widehat{G}^{\lambda}_{\mathsf{and}}\, , \lambda > \widehat{\lambda}_{\mathsf{and},\mathtt{dmax}} \right\}, \quad \text{where} \quad \widehat{\lambda}_{\mathsf{and},\mathtt{dmax}} = \sup\left\{ \lambda : \exists a \in \Gamma,\ d_a(\widehat{G}^{\lambda}_{\mathsf{and}}) > \mathtt{dmax}[a] \right\}.$$

This family is efficiently computed with the LARS algorithm [4], see [6] Sect.2.

LA Algorithm

1. Compute with LARS the $\widehat{\theta}^{\lambda}$ for all the values λ where the support of $\widehat{\theta}^{\lambda}$ changes.
2. Compute the graphs $\widehat{G}^{\lambda}_{\mathsf{and}}$ for all $\lambda > \widehat{\lambda}_{\mathsf{and},\mathtt{dmax}}$.

2.2.3. **Adaptive lasso family $\widehat{\mathcal{G}}_{\mathrm{EW}}$ (with `selectFast`)**

The family $\widehat{\mathcal{G}}_{\mathrm{EW}}$ is a modified version of $\widehat{\mathcal{G}}_{\mathrm{LA}}$ inspired by the adaptive lasso of Zou [9]. The major difference between $\widehat{\mathcal{G}}_{\mathrm{EW}}$ and $\widehat{\mathcal{G}}_{\mathrm{LA}}$ lies in the replacement of $\sum |\theta_{a,b}|$ in (3) by $\sum |\theta_{a,b}/\widehat{\theta}^{\mathrm{EW}}_{a,b}|$, where $\widehat{\theta}^{\mathrm{EW}}$ is a preliminary estimator.

To build the family $\widehat{\mathcal{G}}_{\mathrm{EW}}$, we start by computing the Exponential Weight estimator $\widehat{\theta}^{\mathrm{EW}}$ of [2]. For each $a \in \Gamma$, we set $H_a = \{v \in \mathbb{R}^p : v_a = 0\}$ and

$$\widehat{\theta}^{\mathrm{EW}}_a = \int_{H_a} v\, e^{-\beta \|\mathtt{X}_a - \mathtt{X}v\|^2_n} \prod_j \left(1 + (v_j/\tau)^2\right)^{-1} \frac{dv}{\mathcal{Z}_a}, \tag{4}$$

with $\mathcal{Z}_a = \int_{H_a} e^{-\beta \|\mathtt{X}_a - \mathtt{X}v\|^2_n} \prod_j \left(1 + (v_j/\tau)^2\right)^{-1} dv$ and $\beta, \tau > 0$.
The construction of $\widehat{\mathcal{G}}_{\mathrm{EW}}$ is now similar to the construction of $\widehat{\mathcal{G}}_{\mathrm{LA}}$. For any $\lambda > 0$, we set

$$\widehat{\theta}^{\mathrm{EW},\lambda} = \operatorname{argmin}\left\{ \sum_{i,a} [\mathtt{X} - \mathtt{X}\theta]^2_{i,a} + \lambda \sum_{a \neq b} |\theta_{a,b}/\widehat{\theta}^{\mathrm{EW}}_{a,b}|, \text{ for } \theta \in \Theta \right\},$$

and we define the graph $\widehat{G}^{\mathrm{EW},\lambda}_{\mathsf{or}}$ by setting an edge between a and b if either $\widehat{\theta}^{EW,\lambda}_{b,a}$ or $\widehat{\theta}^{EW,\lambda}_{a,b}$ is non-zero. Finally, the family $\widehat{\mathcal{G}}_{\mathrm{EW}}$ is given by

$$\widehat{\mathcal{G}}_{\mathrm{EW}} = \left\{ \widehat{G}^{\mathrm{EW},\lambda}_{\mathsf{or}},\ \lambda > \widehat{\lambda}^{EW}_{\mathsf{or},\mathtt{dmax}} \right\}, \quad \text{where} \quad \widehat{\lambda}^{\mathrm{EW}}_{\mathsf{or},\mathtt{dmax}} = \sup\left\{ \lambda : \exists a \in \Gamma,\ d_a(\widehat{G}^{\mathrm{EW},\lambda}_{\mathsf{or}}) > \mathtt{dmax}[a] \right\}.$$

The Exponential Weight estimator $\widehat{\theta}^{EW}$ can be computed with a Langevin Monte-Carlo algorithm. We refer to Section 3.1 and Dalalyan & Tsybakov [3] for details. Once $\widehat{\theta}^{EW}$ is computed, the family $\widehat{\mathcal{G}}_{\mathrm{EW}}$ is obtained as $\widehat{\mathcal{G}}_{\mathrm{LA}}$ with the help of the LARS-lasso algorithm.

EW Algorithm

1. Compute $\widehat{\theta}^{EW}$ with a Langevin Monte-Carlo algorithm.
2. Compute with LARS the $\widehat{\theta}^{\mathrm{EW},\lambda}$ for all the values λ where the support of $\widehat{\theta}^{\mathrm{EW},\lambda}$ changes.
3. Compute the graphs $\widehat{G}_{\mathsf{or}}^{\mathrm{EW},\lambda}$ for all $\lambda > \widehat{\lambda}^{\mathrm{EW}}_{\mathsf{or},\mathtt{dmax}}$.

2.2.4. **Mixed family $\widehat{\mathcal{G}}_{\mathrm{C01,LA}}$ (with `selectFast`)**

This family is defined by $\widehat{\mathcal{G}}_{\mathrm{C01,LA}} = \widehat{\mathcal{G}}_{\mathrm{C01}} \bigcup \widehat{\mathcal{G}}_{\mathrm{LA}}$.

2.2.5. **Mixed family $\widehat{\mathcal{G}}_{\mathrm{C01,LA,EW}}$ (with `selectFast`)**

This family is defined by $\widehat{\mathcal{G}}_{\mathrm{C01,LA,EW}} = \widehat{\mathcal{G}}_{\mathrm{C01}} \bigcup \widehat{\mathcal{G}}_{\mathrm{LA}} \bigcup \widehat{\mathcal{G}}_{\mathrm{EW}}$.

2.2.6. **Quasi-exhaustive family $\widehat{\mathcal{G}}_{\mathrm{QE}}$ (with `selectQE`)**

For each node $a \in \Gamma$, we estimate the neighborhood of a by

$$\widehat{\mathrm{ne}}(a) = \mathrm{argmin}\left\{ \|\mathtt{X}_a - \mathrm{Proj}_{V_S}(\mathtt{X}_a)\|_n^2 \left(1 + \frac{\mathrm{pen}(|S|)}{n-|S|}\right) :\ S \subset \Gamma \setminus \{a\} \text{ and } |S| \leq \mathtt{dmax}[a] \right\},$$

where $\mathrm{pen}(.)$ is the penalty function (2) and Proj_{V_S} denotes the orthogonal projection from $\mathbb{R}^n$ onto $V_S = \{\mathtt{X}\beta : \beta \in \mathbb{R}^p \text{ and } \mathrm{supp}(\beta) = S\}$. We then build two nested graphs $\widehat{G}_{\mathsf{and}}$ and $\widehat{G}_{\mathsf{or}}$ as in Meinshausen and Bühlmann [7]

$$\begin{aligned} a \overset{\widehat{G}_{\mathsf{and}}}{\sim} b \quad &\Longleftrightarrow \quad a \in \widehat{\mathrm{ne}}(b) \ \underline{\text{and}}\ b \in \widehat{\mathrm{ne}}(a)\,, \\ a \overset{\widehat{G}_{\mathsf{or}}}{\sim} b \quad &\Longleftrightarrow \quad a \in \widehat{\mathrm{ne}}(b) \ \underline{\text{or}}\ b \in \widehat{\mathrm{ne}}(a)\,, \end{aligned}$$

and define the family $\widehat{\mathcal{G}}_{\mathrm{QE}}$ as the family of all the graphs that lie between $\widehat{G}_{\mathsf{and}}$ and $\widehat{G}_{\mathsf{or}}$

$$\widehat{\mathcal{G}}_{\mathrm{QE}} = \left\{ G,\ \widehat{G}_{\mathsf{and}} \subset G \subset \widehat{G}_{\mathsf{or}} \text{ and } d_a(G) \leq \mathtt{dmax}[a] \text{ for all } a \in \Gamma \right\}.$$

QE Algorithm

1. Compute $\widehat{\mathrm{ne}}(a)$ for all $a \in \Gamma$.
2. Compute the graphs $\widehat{G}_{\mathsf{and}}$ and $\widehat{G}_{\mathsf{or}}$.
3. Work out the family $\widehat{\mathcal{G}}_{\mathrm{QE}}$.

3. User guide

3.1. Graph selection with `selectFast`

Usage: `selectFast(X, dmax=min(floor(nrow(X)/3),nrow(X)-3,ncol(X)-1), K=2.5, family="EW", min.ev=10**(-8), verbose=FALSE, ...)`

Main arguments:

`X`	$n \times p$ matrix where n is the sample size and p the number of variables (nodes). The sample size n should be larger than 3 and the number of variables p larger than 1.
`dmax`	integer or p-dimensional vector of integers smaller or equal to $\min(n-3, p-1)$. When `dmax` is an integer, it corresponds to the maximum degree of the graphs in the family $\widehat{\mathcal{G}}$. When `dmax` is a p-dimensional vector, then `dmax[a]` corresponds to the maximum number of neighbors of `a` in the graphs $G \in \widehat{\mathcal{G}}$. Default value is $\min(\texttt{floor}(n/3), n-3, p-1)$.
`K`	scalar (or vector) with values larger than 1. Tuning parameter of the penalty function (2). Default value is $K = 2.5$ and typical values are between 1 and 3. Increasing the value of K gives more sparse graphs.
`family`	one or several values among `"C01"`, `"LA"`, `"EW"`. When `family="EW"` (respectively `"LA"`, `"C01"`), the criterion (1) is minimized over the family $\widehat{\mathcal{G}}_{\mathrm{EW}}$ (respectively $\widehat{\mathcal{G}}_{\mathrm{LA}}$, $\widehat{\mathcal{G}}_{\mathrm{C01}}$). In addition, when both families `"C01"` and `"LA"` are set, the criterion (1) is also minimized over the family $\widehat{\mathcal{G}}_{\mathrm{C01,LA}}$. When the three families `"C01"`, `"LA"` and `"EW"` are set, the criterion (1) is also minimized over the family $\widehat{\mathcal{G}}_{\mathrm{C01,LA,EW}}$. Default value is `family="EW"`.

Other arguments:

`min.ev`	minimum eigenvalue for matrix inversion. The rank of the matrix is calculated as the number of eigenvalues greater than `min.ev`. The value of `min.ev` must be positive and smaller than 0.01. Default value is `min.ev=10**(-8)`.
`verbose`	logical. If `TRUE` a trace of the current process is displayed in real time.
`...`	arguments specific to `"EW"` (see below).

Output: A list with components: `EW, LA, C01, C01.LA, C01.LA.EW`
The list `EW` reports the results obtained for the family `EW`, etc. Each list has components:

`Neighb`	array of size $p \times \max(\texttt{dmax}) \times \texttt{length(K)}$. The vector `Neighb`$[j, , i_K]$ gives the nodes connected to j for `K`$[i_K]$.
`crit.min`	vector of dimension `length(K)`. It gives the minimal values of the criterion for each value of `K`.
`G`	adjacency matrix with dimension $p \times p \times \texttt{length(K)}$ Each slice `G`$[, , i_K]$ (for $i_K = 1$ to `length(K)`) is the adjacency matrix of the graph for `K=K`$[i_K]$.

Warning: `Neighb` and `G` are matrices if length(`K`)=1.

Complexity of C01 family: the complexity of `selecFast` with option `family="C01"` is of order np^3.

Complexity of LA family: the family $\widehat{\mathcal{G}}_{\mathrm{LA}}$ is build with the help of the `LARS` package. The complexity of `selecFast` with option `family="LA"` is usually of order $p^2 n \min(n, p)$.

Complexity of EW family and choice of some specific arguments: The Exponential Weight estimator $\hat{\theta}_a^{\mathrm{EW}}$ defined by (4) is computed using a Langevin Monte Carlo algorithm introduced in [3]. This algorithm involves several parameters denoted T_0, h, `max.iter` and `eps` (see below).

The Langevin Monte Carlo algorithm is based on the formula

$$\hat{\theta}_a^{\mathrm{EW}} = \lim_{T\to\infty} \frac{1}{T-T_0} \int_{T_0}^{T} v_t \, dt,$$

with $(v_t)_{t\geq 0}$ solution of the Langevin equation $dv_t = F(v_t)dt + \sqrt{2}\, dW_t$, where W is a Brownian motion on $H_a = \{v \in \mathbb{R}^p : v_a = 0\}$ and where $F(v) = (F_1(v), \ldots, F_p(v))$ is defined by

$$F_a(v) = 0 \text{ and } F_j(v) = \frac{2\beta}{n}[\mathtt{X}^T(\mathtt{X}_a - \mathtt{X}v)]_j - \frac{2v_j}{\tau^2 + v_j^2}, \quad \text{for } j \neq a.$$

The process $(v_t)_{t\geq 0}$ is approximated via an Euler discretization scheme:

$$v^{(0)} = 0 \text{ and } v^{(k+1)} = v^{(k)} + hF(v^{(k)}) + \sqrt{2h}\, W_k, \tag{5}$$

with $W_1, W_2, \ldots$ i.i.d. standard Gaussian random variables on H_a. Then, the average $\frac{1}{T-T_0}\int_{T_0}^{T} v_t \, dt$ is approximated by

$$\bar{v}_T = \frac{1}{[(T-T_0)/h]} \sum_{k=[T_0/h]}^{[T/h]-1} v^{(k)}.$$

Finally, we set $\hat{\theta}_a^{\mathrm{EW}} = \bar{v}_{T_m}$, where $T_m = (1+m)T_0$ with m chosen as follows: the integer m is the minimum between `max.iter` and the smallest m such that

$$\|\bar{v}_{T_m} - \bar{v}_{T_{m-1}}\|^2 < \mathtt{eps}\, \|\bar{v}_{T_{m-1}}\|^2. \tag{6}$$

The parameters involved in the computation of $\hat{\theta}_a^{\mathrm{EW}}$ are:

`beta`	positive real number. Tuning parameter β of $\hat{\theta}_a^{\mathrm{EW}}$, see (4). Default value is $\mathtt{beta} = n^2/2$
`tau`	positive real number. Tuning parameter τ of $\hat{\theta}_a^{\mathrm{EW}}$, see (4). Default value is $\mathtt{tau} = (n(p-1))^{-1/2}$
`h`	(small) positive real number. Discretization parameter h of the Euler scheme (5). Default value is `h=0.001`
`T0`	positive integer. Heating parameter T_0. The average $\bar{v}_{T_m}$ is computed for times T_m multiple of `T0`. Default value is `T0=10`
`max.iter`	positive integer. Maximal value of m for T_m. When m reaches the value `max.iter`, the parameter $\hat{\theta}_a^{\mathrm{EW}}$ is set to $\bar{v}_{T_{\mathtt{max.iter}}}$ and a warning is displayed. Default value is `max.iter=200`
`eps`	(small) positive real number. Tuning parameter of the convergence criterion (6). Default value is `eps=0.01`

Choice of the Langevin Monte Carlo parameters `h`, `T0`, `max.iter`, `eps`:

- The Markov process $(v^{(k)} : k \geq 0)$ can be transient when `h` is too large and the average $\bar{v}_T$ then blows up. In this case, choose a smaller value for `h`.
- When `max.iter` is reached you can either choose a larger `T0` or increase the value of `max.iter`. You may also choose a smaller value for `h`.
- Choosing a smaller value for `eps` or `h` increases the precision of the computation of $\hat{\theta}_a^{\mathrm{EW}}$ but it can significantly slow down the algorithm.
- We refer to [3] for a discussion on the choice of the parameters `beta`, `tau`, `h`, `T0` (Section 5) and a discussion on the convergence of the Euler scheme (Section 4).

The complexity of the Langevin Monte Carlo algorithm heavily depends on the choice of the parameters `h`, `T0`, `max.iter`, `eps`. The maximum complexity is of order $p^2\, \mathtt{T0}/\mathtt{h} \times \mathtt{max.iter} + np^2$. Some examples of CPU times are given in [6] Table 1 Sect. 4.

3.2. Graph selection with `selectQE`

Usage: `selectQE(X, dmax=min(3,nrow(X)-3,ncol(X)-1), K=2.5, min.ev=10**(-8), verbose=FALSE, max.size=10**8, max.iter=10**6, max.nG=10**8)`

Main arguments:

`X`	$n \times p$ matrix where n is the sample size and p the number of variables (nodes). The sample size n should be larger than 3 and the number of variables p larger than 1.
`dmax`	integer or p-dimensional vector of integers smaller or equal to $\min(n-3, p-1)$. When `dmax` is an integer, it corresponds to the maximum degree of the graphs in the family $\widehat{\mathcal{G}}$. When `dmax` is a p-dimensional vector, then `dmax[a]` corresponds to the maximum number of neighbors of `a` in the graphs $G \in \widehat{\mathcal{G}}$. Default value is $\min(3, n-3, p-1)$.
`K`	scalar (or vector) with values larger than 1. Tuning parameter of the penalty function (2). Default value is $K = 2.5$ and typical values are between 1 and 3. Increasing the value of K gives more sparse graphs.

Other arguments:

`min.ev`	minimum eigenvalue for matrix inversion. The rank of the matrix is calculated as the number of eigenvalues greater than `min.ev`. The value of `min.ev` must be positive and smaller than 0.01. Default value is `min.ev=10**(-8)`.
`verbose`	logical. If `TRUE` a trace of the current process is displayed in real time.
`max.size`	integer. Maximum number of subsets S for estimating $\widehat{G}_{\mathrm{and}}$ and $\widehat{G}_{\mathrm{or}}$, see Section 2.2.6. If $\sum_{a=1}^{p} \sum_{d=1}^{\texttt{dmax[a]}} C_{p-1}^{d}$ is greater than `max.size`, then execution is stopped. Default value is `max.size=10**8`.

Output:

`Neighb`	see selectFast.
`min.crit`	see selectFast.
`G`	see selectFast.

Complexity and choice of some advanced arguments.

The complexity of the QE algorithm is of order $np^{D+1}D^3 + npD\,\mathrm{card}(\widehat{\mathcal{G}}_{\mathrm{QE}})$, where D is the maximum of `dmax`. Thus, the QE algorithm cannot be used for large values of p and D. Furthermore, even for moderate values of p and D the cardinality of $\widehat{\mathcal{G}}_{\mathrm{QE}}$ can be large and lead to memory size (and computational time) problems. In that case, the research between $\widehat{G}_{\mathrm{and}}$ and $\widehat{G}_{\mathrm{or}}$ is stopped and prolonged by a stepwise procedure. Let us denote by $\widehat{\mathcal{G}}_q$ the collection of graphs G with q edges that belong to $\widehat{\mathcal{G}}_{\mathrm{QE}}$ and by $\widehat{G}_q$ the minimizer of Crit over $\widehat{\mathcal{G}}_q$. The exhaustive search between $\widehat{G}_{\mathrm{and}}$ and $\widehat{G}_{\mathrm{or}}$ stops as soon as the number of graphs in $\widehat{\mathcal{G}}_q$ is either greater than a threshold denoted `max.nG`, or greater than the maximum allowed memory size. Let q^{stop} be the value of q before stopping the exhaustive procedure and let $\widehat{G}^{\mathrm{stop}}$ be the minimizer of Crit over $\left\{\widehat{G}_{\mathrm{and}}\right\} \cup \left\{\widehat{G}_q, q \leq q^{\mathrm{stop}}\right\}$. A forward/backward stepwise procedure is taking over to explore graphs between $\widehat{G}_{q^{\mathrm{stop}}}$ and $\widehat{G}_{\mathrm{or}}$. Finally $\widehat{G}$ is the minimizer of Crit over $\widehat{G}^{\mathrm{stop}}$ and the graph stemmed from the stepwise procedure.

The stepwise procedure involves the following arguments:

`max.iter`	integer. Maximum number of stepwise iterations. Default value is `max.iter=10**6`.
`max.nG`	integer. Maximum number of graphs considered in the exhaustive search. Default value is `max.nG=10**8`.

3.3. Graph selection with `selectMyFam`

Usage: `selectMyFam(X, MyFamily, K=2.5, min.ev=10**(-8))`

Arguments:

`X`	$n \times p$ matrix where n is the sample size and p the number of variables (nodes). The sample size n should be larger than 3 and the number of variables p larger than 1.
`MyFamily`	list of $p \times p$ adjacency matrices corresponding to candidate graphs with degree less or equal to $n-3$
`K`	scalar (or vector) with values larger than 1. Tuning parameter of the penalty function (2). Default value is $K = 2.5$ and typical values are between 1 and 3. Increasing the value of K gives more sparse graphs.
`min.ev`	minimum eigenvalue for matrix inversion. The rank of the matrix is calculated as the number of eigenvalues greater than `min.ev`. The value of `min.ev` must be positive and smaller than 0.01. Default value is `min.ev=10**(-8)`

Output:

`Neighb`	see selectFast.
`min.crit`	see selectFast.
`G`	see selectFast.

Complexity: The complexity of `selectMyFam` is of maximum order $np\min(n,p) \times \mathrm{card}(\texttt{MyFamily})$.

4. Auxiliary functions

4.1. Random graph generator `simulateGraph`

The function `simulateGraph` generates random covariance matrices C with sparse inverse Ω. The Gaussian law $\mathcal{N}(0, C)$ is then a sparse (non-uniform) Gaussian Graphical Model. The arguments of the function `simulateGraph` are the number of nodes p and two real numbers `eta` and `extraeta` between 0 and 1.

The inverse covariance matrix Ω is defined by $\Omega = BB^T + D$, where B is a random sparse lower triangular matrix and D is a diagonal matrix with random entries sampled uniformly between 10^{-3} and 5.10^{-3}. The latter matrix D prevents Ω from having too small eigenvalues. To generate B, the set $\{1, \ldots, p\}$ is split into three consecutive sets I_1, I_2, I_3. For any $i < j$ in the same set I_k, the entry $B_{i,j}$ is set to 0 with probability $1 - \eta_{\mathrm{int}}$, where $\eta_{\mathrm{int}} = \texttt{eta} + (1 - \texttt{eta}) * \texttt{extraeta}$. For any $i < j$ belonging to two different sets, the entry $B_{i,j}$ is set to 0 with probability $1 - \eta_{\mathrm{ext}}$, where $\eta_{\mathrm{ext}} = \texttt{extraeta}$. Then, the lower diagonal values that have not been set to 0 are drawn according to a uniform law on $[-1/\sqrt{\varepsilon}, 1/\sqrt{\varepsilon}]$ and the diagonal values are drawn according to a uniform law on $[0, \sqrt{\varepsilon}]$. The value ε is set to $1/10$.
Finally, the matrix C is obtained by first inverting Ω and then rescaling this inverse in order to have 1 on the diagonal (with the function `cov2cor` of the R package `stats`).

Usage: `simulateGraph(p, eta, extraeta = eta/5)`

Arguments:

`p`	integer. Number of rows and columns of C. Should be greater than 1.
`eta`	real number in (0,1). The proportion of edges in the three subgroups is `eta+(1-eta)*extraeta`. Small values of `eta` give sparse graphs.
`extraeta`	real number in (0,1). Proportion of edges inter groups.

4.2. Penalty function `penalty`

The function `penalty` compute the penalty function (2).

Usage: `penalty(p,n, dmax=min(3,n-3,p-1), K=2.5)`

Arguments:

`p`	the number of variables. `p` should be greater than 1.
`n`	the sample size. `n` should be greater than 3.
`dmax`	integer or `p`-dimensional vector of integers smaller or equal to `min(n-3, p-1)`. When `dmax` is a scalar, it gives the maximum degree of the estimated graph. When `dmax` is a vector, `dmax[a]` gives the maximum degree of the node `a`. Default value: `min(3,n-3,p-1)`.
`K`	scalar or vector of real numbers larger than 1. Tuning parameter of the penalty function (2).

4.3. Graph converter `convertGraph`

A graph G can either be represented by an adjacency matrix or by an array A_G where $A_G[a,]$ gives the list of all the nodes connected to the node a in the graph. The function `convertGraph` converts `NG` graphs represented by list of nodes into adjacency matrices. The `NG` graphs $G_1, \ldots, G_{\texttt{NG}}$ are given as input through a single `p` $\times$ `Dmax` $\times$ `NG` array `Graph`, defined by `Graph`$[,,\texttt{i}] = A_{G_i}$.
This function can be useful to generate the argument `MyFamily` of `selectMyFam`.

Usage: `convertGraph(Graph)`

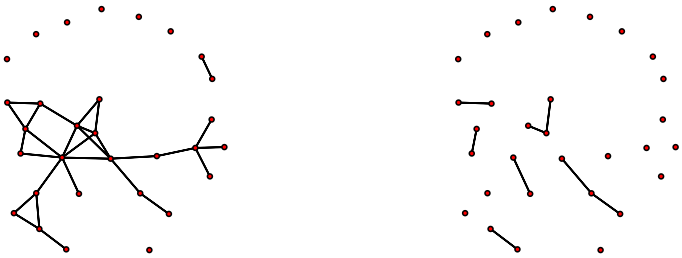
Argument:

`Graph`	array of dimension `p` $\times$ `Dmax` $\times$ `NG`, where `Dmax` is the maximum degree of the `NG` graphs. When `NG` equals 1, `Graph` can be a matrix of dimension `p` $\times$ `Dmax`. `Graph[a,iG]` should be the indices of the nodes connected to the node `a`, for the graph `iG`. `Graph[a,1,iG]` should be equal to 0 if there is no node connected to the node `a`.

5. Examples

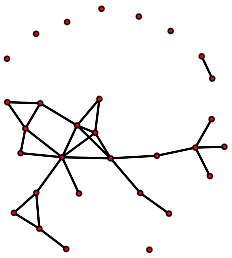
```
>  library("GGMselect")
> p=30
> n=30
> # ---------------------------------------
> # Random graph generator: use of simulateGraph
> # ---------------------------------------
> eta=0.11
> Gr <- simulateGraph(p,eta)
> X <- rmvnorm(n, mean=rep(0,p), sigma=Gr$C)
> # ---------------------------------------
> # Graph selection with family C01:  use of selectFast
> # ---------------------------------------
> GRest <- selectFast(X, family="C01")
>
> # ---------------------------------------
> # Plot the result with the help of the package network
> # ---------------------------------------
> library(network)
>
```

```
> gV <- network(Gr$G)
> g <- network(GRest$C01$G)
> par(mfrow=c(1,2), pty = "s")
> a <- plot(gV, usearrows = FALSE)
> title(sub="Simulated graph")
> plot(g, coord=a, usearrows = FALSE)
> title(sub="Graph selected with C01 family")
>
```

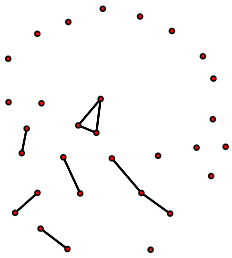


Simulated graph Graph selected with C01 family

```
> # ----------------------------------------
> # Graph selection with family QE:  use of selectQE
> # ----------------------------------------
> GQE <- selectQE(X)
> # ----------------------------------------
> # Plot the result
> # ----------------------------------------
>
```

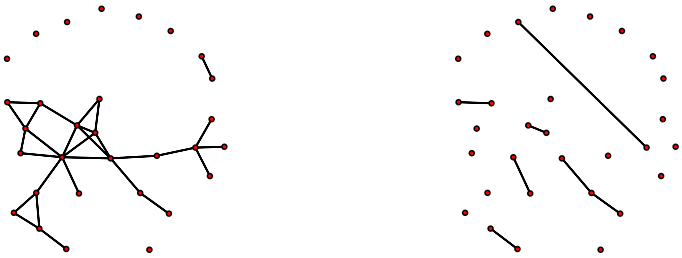


Simulated graph



Graph selected with QE family

```
> # ----------------------------------------
> # Graph selection with selectMyFam
> # ----------------------------------------
> # generate a family of candidate graphs with glasso
> library("glasso")
> MyFamily <- NULL
> for (j in 1:3){
+   MyFamily[[j]] <- abs(sign(glasso(cov(X),rho=j/5)$wi))
+   diag(MyFamily[[j]]) <- 0
+ }
> # select a graph within MyFamily
> GMF <- selectMyFam(X,MyFamily)
> # ----------------------------------------
> # Plot the result
> # ----------------------------------------
>
>
```



Simulated graph Graph selected with MyFam

References

[1] Y. Baraud, C. Giraud, and S. Huet. Gaussian model selection with an unknown variance. *Ann. Statist.*, 37(2):630–672, 2009.

[2] A. Dalayan and A. Tsybakov. Aggregation by exponential weighting, sharp oracle inequalities and sparsity. *Machine Learning*, 72(1-2):39– 61, 2008.

[3] A. Dalayan and A. Tsybakov. Sparse regression learning by aggregation and langevin monte-carlo. arXiv:0903.1223, 2009.

[4] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 2004. With discussion, and a rejoinder by the authors.

[5] C. Giraud. Estimation of Gaussian graphs by model selection. *Electron. J. Stat.*, 2:542–563, 2008.

[6] C. Giraud, S. Huet, and N. Verzelen. Graph selection with GGMselect. arXiv:0907.0619, 2009.

[7] N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Ann. Statist.*, 34(3):1436–1462, 2006.

[8] A. Wille and P. Bühlmann. Low-order conditional independence graphs for inferring genetic networks. *Stat. Appl. Genet. Mol. Biol.*, 5:Art. 1, 34 pp. (electronic), 2006.

[9] H. Zou. The adaptive lasso and its oracle properties. *J. Amer. Statist. Assoc.*, 101(476):1418–1429, 2006.