



Variable Clustering in High-Dimensional Linear Regression: The R Package *clere*

Loic Yengo
CNRS UMR 8199
Univ. Lille 1, Inria

Julien Jacques
Univ. Lille 1
CNRS, Inria

Christophe Biernacki
Univ. Lille 1
CNRS, Inria

Mickael Canouil
CNRS UMR 8199

Abstract

Dimension reduction is one of the biggest challenge in high-dimensional regression models. We recently introduced a new methodology based on variable clustering as a means to reduce dimensionality. We introduce here an R package that implements two enhancements regarding the latter methodology. First, an improvement in computational time for estimating the parameters is presented. As a second enhancement, users of our method are now allowed to constrain the model to identify variables with weak or no effect on the response. An overview of the package functionalities as well as examples to run an analysis are described. Numerical experiments on simulated and real data were performed to illustrate the gain of computational time and the good predictive performance of our method compared to standard dimension reduction approaches.

Keywords: dimension reduction, variable clustering, linear regression, R.

1. Introduction

High dimensionality is increasingly ubiquitous in numerous scientific fields including genetics, economics and physics. Reducing the dimensionality is a challenge that most statistical methodologies must meet not only to remain interpretable but also to achieve reliable predictions. In linear regression models, dimension reduction techniques often refer to variables selection. Approaches for variables selection are implemented in publicly available software, that involve the well-known R packages **glmnet** and **spikeslab**. The R package **glmnet** implements the Elastic net methodology [Zou and Hastie (2005)], which is a generalization of both the LASSO [Tibshirani (1996)] and the ridge regression (RR) [Hoerl and Kennard (1970)]. The R package **spikeslab** in turn, implements the Spike and Slab methodology [Ishwaran and Rao (2005)], which is a Bayesian approach for variables selection.

Dimension reduction can not however, be restricted to variables selection. Indeed, the field can be extended to include approaches which aim is to create surrogate covariates that summarizes the information carried in initial covariates. Since the emblematic Principal Component Regression (PCR)[Jolliffe (1982)], many of the latter methods spread in the recent literature. As specific examples, we may refer to the OSCAR methodology [Bondell and Reich (2008)], or the PACS methodology [Sharma, Bondell, and Zhang (2013)] which is a generalization of the latter approach. Those methods mainly proposed variables clustering within a regression model as a way to reduce the dimensionality. Despite their theoretical and practical appeal, implementations of those methods were often proposed only through Matlab or R scripts, limiting thus the flexibility and the computational efficiency of their use. The CLusterwise Effect REgression (CLERE) methodology [Yengo, Jacques, and Biernacki (2013)], was recently introduced as a novel methodology for simultaneous variables clustering and regression. The CLERE methodology is based on the assumption that each regression coefficient is an unobserved random variable sampled from a mixture of Gaussian distributions with an arbitrary number g of components. In addition, all components in the mixture are assumed to have different means (b_1, \dots, b_g) and equal variances equal to γ^2 .

In this paper, we propose two new features for the CLERE model. First, the stochastic EM (SEM) algorithm is proposed as a more computationally efficient alternative to the Monte Carlo EM (MCEM) algorithm previously introduced in [Yengo *et al.* (2013)]. Secondly, the CLERE model is enhanced with the possibility of constraining the first component to have its mean equal to 0, i.e. $b_1 = 0$. This enhancement mainly aimed at facilitating the interpretation of the model. Indeed when b_1 is set to 0, variables assigned to the cluster associated with b_1 might be considered less relevant than other variables. Those two new features were implemented in a C++ program available through the R package **clere**.

The outline of the present paper is the following. In Section 2, the definition of the model is recalled and the strategy to estimate the model parameter is presented. In Section 3 are described both the MCEM and SEM algorithms. This section also presents how the number of clusters is chosen and how the constraint on parameter b_1 can be interpreted. Section 4 presents the main functionalities of the R package **clere**. In Section 5, numerical experiments are presented aiming at illustrating the computational gain of the SEM algorithm over our former strategy and the good predictive performances of CLERE compared to standard dimension reduction methods. In that section, we also present two real data analysis performed with the R package **clere**, through two datasets. Finally, perspectives and further potential improvements of the package are discussed in Section 6.

2. Model definition and notation

Our model is defined by the following hierarchical relationships:

$$\begin{cases} y_i \sim \mathcal{N} \left(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}, \sigma^2 \right) \\ \beta_j | \mathbf{z}_j \sim \mathcal{N} \left(\sum_{k=1}^g b_k z_{jk}, \gamma^2 \right) \\ \mathbf{z}_j = (z_{j1}, \dots, z_{jg}) \sim \mathcal{M} (\pi_1, \dots, \pi_g). \end{cases} \quad (1)$$

For an individual $i = 1, \dots, n$, y_i is the response and x_{ij} is an observed value for the j -th covariate. β_j is the regression coefficient associated with the j -th covariate ($j = 1, \dots, p$). Let $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)$, $\mathbf{y} = (y_1, \dots, y_n)'$, $\mathbf{X} = (x_{ij})$, $\mathbf{Z} = (z_{jk})$, $\mathbf{b} = (b_1 \dots b_g)'$ and $\boldsymbol{\pi} =$

$(\pi_1, \dots, \pi_g)'$.

Moreover, $\log p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta})$ denotes the log-likelihood of model (1) assessed for the parameter $\boldsymbol{\theta} = (\beta_0, \mathbf{b}, \boldsymbol{\pi}, \sigma^2, \gamma^2)$. Model (1) can be interpreted as a Bayesian approach. However, to be fully Bayesian a prior distribution for parameter $\boldsymbol{\theta}$ would have been necessary. Instead, we proposed to estimate $\boldsymbol{\theta}$ by maximizing the (marginal) log-likelihood, $\log p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta})$. This partially Bayesian approach is referred to as *Empirical Bayes* (EB) [Casella (1985)]. Let \mathcal{Z} be the set of $p \times g$ -matrices partitioning p covariates into g groups. Those matrices are defined as

$$\mathbf{Z} = (z_{jk})_{1 \leq j \leq p, 1 \leq k \leq g} \in \mathcal{Z} \Leftrightarrow \forall j = 1, \dots, p \begin{cases} \exists! k \text{ such as } z_{jk} = 1 \\ \text{if } k' \neq k \text{ then } z_{jk'} = 0. \end{cases}$$

The log-likelihood $\log p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta})$ is defined as

$$\log p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta}) = \log \left[\sum_{\mathbf{Z} \in \mathcal{Z}} \int_{\mathbb{R}^p} p(\mathbf{y}, \boldsymbol{\beta}, \mathbf{Z}|\mathbf{X};\boldsymbol{\theta}) d\boldsymbol{\beta} \right].$$

Since it requires integrating over \mathcal{Z} with cardinality g^p , evaluating the likelihood becomes rapidly computationally unaffordable.

Nonetheless, maximum likelihood estimation is still achievable using the expectation maximization (EM) algorithm [Dempster, Laird, and Rubin (1977)]. The latter algorithm is an iterative method which starts with an initial estimate of the parameter and updates this estimate until convergence. Each iteration of the algorithm consists of two steps, denoted as the *E* and the *M* steps. At each iteration d of the algorithm, the *E step* consists in calculating the expectation of the log-likelihood of the complete data (observed + unobserved) with respect to $p(\boldsymbol{\beta}, \mathbf{Z}|\mathbf{y}, \mathbf{X};\boldsymbol{\theta}^{(d)})$, the conditional distribution of the unobserved data given the observed data, and the value of the parameter at the current iteration, $\boldsymbol{\theta}^{(d)}$. This expectation, often denoted as $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(d)})$ is then maximized with respect to $\boldsymbol{\theta}$ at the *M step*.

In model (1), the *E step* is analytically intractable. A broad literature devoted to intractable *E steps* recommends the use of a stochastic approximation of $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(d)})$ through Monte Carlo (MC) simulations [Wei and Tanner (1990), Levine and Casella (2001)]. This approach is referred to as the MCEM algorithm. Besides, mean-field-type approximations are also proposed [Govaert and Nadif (2008), Mariadassou, Robin, and Vacher (2010)]. Despite their computational appeal, the latter approximations do not generally ensure convergence to the maximum likelihood [Gunawardana and Byrne (2005)]. Alternatively, the SEM algorithm [Celeux, Chauveau, and Diebolt (1996)] was introduced as a stochastic version of the EM algorithm. In this algorithm, the *E step* is replaced with a simulation step (*S step*) that consists in generating a complete sample by simulating the unobserved data using $p(\boldsymbol{\beta}, \mathbf{Z}|\mathbf{y}, \mathbf{X};\boldsymbol{\theta}^{(d)})$. After the *S step* follows the *M step* which consists in maximizing $p(\boldsymbol{\beta}, \mathbf{Z}|\mathbf{y}, \mathbf{X};\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. Alternating those two steps generate a sequence $(\boldsymbol{\theta}^{(d)})$, which is Markov chain whose stationary distribution (when it exists) concentrates around a local maximum of the likelihood.

3. Estimation and model selection

3.1. Initialization

The two algorithms presented in this section are initialized using a primary estimate $\beta_j^{(0)}$ of each β_j . The latter can be chosen either at random, or obtained from univariate regression coefficients or penalized approaches like LASSO and ridge regression. For large SEM or MCEM chains, initialization is not a critical issue. The choice of the initialization strategy is therefore made to speed up the convergence of the chains. A Gaussian mixture model with g component(s) is then fitted using $\boldsymbol{\beta}^{(0)} = (\beta_1^{(0)}, \dots, \beta_p^{(0)})$ as observed data to produce starting values $\mathbf{b}^{(0)}$, $\boldsymbol{\pi}^{(0)}$ and $\gamma^{2(0)}$ respectively for parameters \mathbf{b} , $\boldsymbol{\pi}$ and γ^2 . Using maximum a posteriori (MAP) clustering, an initial partition $\mathbf{Z}^{(0)} = (z_{jk}^{(0)}) \in \mathcal{Z}$ is obtained as

$$\forall j \in \{1, \dots, p\}, z_{jk}^{(0)} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_{k' \in \{1, \dots, g\}} (\beta_j^{(0)} - b_{k'}^{(0)})^2 \\ 0 & \text{otherwise.} \end{cases}$$

β_0 and σ^2 are initialized using $\boldsymbol{\beta}^{(0)}$ as follows:

$$\beta_0^{(0)} = \frac{1}{n} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j^{(0)} x_{ij} \right) \text{ and } \sigma^{2(0)} = \frac{1}{n} \sum_{i=1}^n \left(y_i - \beta_0^{(0)} - \sum_{j=1}^p \beta_j^{(0)} x_{ij} \right)^2.$$

3.2. MCEM algorithm

The Stochastic Approximation of the E step

Suppose at iteration d of the algorithm that we have $\{(\boldsymbol{\beta}^{(1,d)}, \mathbf{Z}^{(1,d)}), \dots, (\boldsymbol{\beta}^{(M,d)}, \mathbf{Z}^{(M,d)})\}$, M samples from $p(\boldsymbol{\beta}, \mathbf{Z} | \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}^{(d)})$. Then the MC approximation of the E -step can be written

$$Q(\boldsymbol{\theta} | \boldsymbol{\theta}^{(d)}) = \mathbb{E} \left[\log p(\mathbf{y}, \boldsymbol{\beta}, \mathbf{Z} | \mathbf{X}; \boldsymbol{\theta}^{(d)}) | \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}^{(d)} \right] \approx \frac{1}{M} \sum_{m=1}^M \log p(\mathbf{y}, \boldsymbol{\beta}^{(m,d)}, \mathbf{Z}^{(m,d)} | \mathbf{X}; \boldsymbol{\theta}^{(d)}).$$

However, sampling from $p(\boldsymbol{\beta}, \mathbf{Z} | \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}^{(d)})$ is not straightforward. However, we can use a Gibbs sampling scheme to simulate unobserved data, taking advantage of $p(\boldsymbol{\beta} | \mathbf{Z}, \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}^{(d)})$ and $p(\mathbf{Z} | \boldsymbol{\beta}, \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}^{(d)})$ from which it is easy to simulate. Those distributions, respectively Gaussian and multinomial, are described below in Equations (2) and (3).

$$\begin{cases} \boldsymbol{\beta} | \mathbf{Z}, \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}^{(d)} \sim \mathcal{N}(\boldsymbol{\mu}^{(d)}, \boldsymbol{\Sigma}^{(d)}) \\ \boldsymbol{\mu}^{(d)} = \left[\mathbf{X}'\mathbf{X} + \frac{\sigma^{2(d)}}{\gamma^{2(d)}} \mathbf{I}_p \right]^{-1} \mathbf{X}' \left(\mathbf{y} - \beta_0^{(d)} \mathbf{1}_p \right) + \frac{\sigma^{2(d)}}{\gamma^{2(d)}} \left[\mathbf{X}'\mathbf{X} + \frac{\sigma^{2(d)}}{\gamma^{2(d)}} \mathbf{I}_p \right]^{-1} \mathbf{Z} \mathbf{b}^{(d)} \\ \boldsymbol{\Sigma}^{(d)} = \sigma^{2(d)} \left[\mathbf{X}'\mathbf{X} + \frac{\sigma^{2(d)}}{\gamma^{2(d)}} \mathbf{I}_p \right]^{-1} \end{cases} \quad (2)$$

and (note that $p(\mathbf{Z} | \boldsymbol{\beta}, \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}^{(d)})$ does not depend on \mathbf{X} nor \mathbf{y})

$$p(z_{jk} = 1 | \boldsymbol{\beta}; \boldsymbol{\theta}^{(d)}) \propto \pi_k^{(d)} \exp \left(- \frac{(\beta_j - b_k^{(d)})^2}{2\gamma^{2(d)}} \right). \quad (3)$$

In Equation (2), \mathbf{I}_p and $\mathbf{1}_p$ respectively stands for the identity matrix in dimension p and the vector of \mathbb{R}^p which all coordinates equal 1. To efficiently sample from $p(\boldsymbol{\beta}|\mathbf{Z}, \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}^{(d)})$ a preliminary singular vector decomposition of matrix \mathbf{X} is necessary. Once this decomposition is performed the overall complexity of the approximated *E step* is $\mathcal{O}[M(p^2 + pg)]$.

The *M step*

Using the M draws obtained by Gibbs sampling at iteration d , the *M step* is straightforward as detailed in Equations (4) to (8). The overall computational complexity of that step is $\mathcal{O}(Mpg)$.

$$\pi_k^{(d+1)} = \frac{1}{Mp} \sum_{m=1}^M \sum_{j=1}^p z_{jk}^{(m,d)}, \quad (4)$$

$$b_k^{(d+1)} = \frac{1}{Mp\pi_k^{(d+1)}} \sum_{m=1}^M \sum_{j=1}^p z_{jk}^{(m,d)} \beta_j^{(m,d)}, \quad (5)$$

$$\gamma^{2(d+1)} = \frac{1}{Mp} \sum_{m=1}^M \sum_{j=1}^p \sum_{k=1}^g z_{jk}^{(m,d)} \left(\beta_j^{(m,d)} - b_k^{(d+1)} \right)^2, \quad (6)$$

$$\beta_0^{(d+1)} = \frac{1}{n} \sum_{i=1}^n \left[y_i - \sum_{j=1}^p \left(\frac{1}{M} \sum_{m=1}^M \beta_j^{(m,d)} \right) x_{ij} \right], \quad (7)$$

$$\sigma^{2(d+1)} = \frac{1}{nM} \sum_{m=1}^M \sum_{i=1}^n \left(y_i - \beta_0^{(d+1)} - \sum_{j=1}^p \beta_j^{(m,d)} x_{ij} \right)^2. \quad (8)$$

3.3. SEM algorithm

In most situations, the SEM algorithm can be considered as a special case of the MCEM algorithm [Celeux *et al.* (1996)], obtained by setting $M = 1$. In model (1), such a direct derivation leads to an algorithm which computational complexity remains quadratic with respect to p . To reduce that complexity, we propose a SEM algorithm based on the integrated complete data likelihood $p(\mathbf{y}, \mathbf{Z}|\mathbf{X}; \boldsymbol{\theta})$ rather than $p(\mathbf{y}, \boldsymbol{\beta}, \mathbf{Z}|\mathbf{X}; \boldsymbol{\theta})$. A closed form of $p(\mathbf{y}, \mathbf{Z}|\mathbf{X}; \boldsymbol{\theta})$ is available and given subsequently.

Closed form of the integrated complete data likelihood

Let the SVD decomposition of matrix \mathbf{X} be \mathbf{USV}' , where \mathbf{U} and \mathbf{V} are respectively $n \times n$ and $p \times p$ orthogonal matrices, and \mathbf{S} is $n \times p$ rectangular diagonal matrix which diagonal terms are the eigenvalues $(\lambda_1^2, \dots, \lambda_n^2)$ of matrix $\mathbf{X}\mathbf{X}'$. We now define $\mathbf{X}^u = \mathbf{U}'\mathbf{X}$ and $\mathbf{y}^u = \mathbf{U}'\mathbf{y}$. Let \mathbf{M} be the $n \times (g+1)$ matrix which first column is made of 1's and which additional

columns are those of matrix $\mathbf{X}^u \mathbf{Z}$. Let also $\mathbf{t} = (\beta_0, \mathbf{b}) \in \mathbb{R}^{(g+1)}$ and \mathbf{R} be a $n \times n$ diagonal matrix which i -th diagonal term equal $\sigma^2 + \gamma^2 \lambda_i^2$. With these notations we can express the complete data likelihood integrated over $\boldsymbol{\beta}$ as

$$\begin{aligned} \log p(\mathbf{y}, \mathbf{Z} | \mathbf{X}; \boldsymbol{\theta}) &= -\frac{n}{2} \log(2\pi) - \frac{1}{2} \sum_{i=1}^n \log(\sigma^2 + \gamma^2 \lambda_i^2) - \frac{1}{2} (\mathbf{y}^u - \mathbf{M}\mathbf{t})' \mathbf{R}^{-1} (\mathbf{y}^u - \mathbf{M}\mathbf{t}) \\ &\quad + \sum_{j=1}^p \sum_{k=1}^g z_{jk} \log \pi_k. \end{aligned} \quad (9)$$

Simulation step

To sample from $p(\mathbf{Z} | \mathbf{y}, \mathbf{X}; \boldsymbol{\theta})$ we use a Gibbs sampling strategy based on the conditional distributions $p(z_j | \mathbf{y}, \mathbf{Z}^{-j}, \mathbf{X}; \boldsymbol{\theta})$, \mathbf{Z}^{-j} denoting the set of cluster membership indicators for all covariates but the j -th. Let $\mathbf{w}^{-j} = (w_1^{-j}, \dots, w_n^{-j})'$, where $w_i^{-j} = y_i^u - \beta_0 - \sum_{l \neq j} \sum_{k=1}^g z_{lk} x_{il}^u b_k$. The conditional distribution $p(z_{jk} = 1 | \mathbf{Z}^{-j}, \mathbf{y}, \mathbf{X}; \boldsymbol{\theta})$ can be written

$$p(z_{jk} = 1 | \mathbf{Z}^{-j}, \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}) \propto \pi_k \exp \left[-\frac{b_k^2}{2} (\mathbf{x}_j^u)' \mathbf{R}^{-1} \mathbf{x}_j^u + b_k (\mathbf{w}^{-j})' \mathbf{R}^{-1} \mathbf{x}_j^u \right], \quad (10)$$

where \mathbf{x}_j^u is the j -th column of \mathbf{X}^u . In the classical SEM algorithm, convergence to $p(\mathbf{Z} | \mathbf{y}, \mathbf{X}; \boldsymbol{\theta})$ should be reached before updating $\boldsymbol{\theta}$. However, a valid inference can still be ensured in settings when $\boldsymbol{\theta}$ is updated only after one or few Gibbs iterations. These approaches are referred to as SEM-Gibbs algorithm [Biernacki and Jacques (2013)]. The overall computational complexity of the simulation step is $\mathcal{O}(npg)$, so linear with p and no quadratic as obtained previously with MCEM.

To improve the mixing of the generated Markov chain, we start the simulation step at each iteration by creating a random permutation of $\{1, \dots, p\}$. Then, according to the order defined by that permutation, we update each z_{jk} using $p(z_{jk} = 1 | \mathbf{Z}^{-j}, \mathbf{y}, \mathbf{X}; \boldsymbol{\theta})$.

Maximization step

$\log p(\mathbf{y}, \mathbf{Z} | \mathbf{X}; \boldsymbol{\theta})$ corresponds to the marginal log-likelihood of a linear mixed model [Searle, Casella, and McCulloch (1992)] which can be written

$$\mathbf{y}^u = \mathbf{M}\mathbf{t} + \boldsymbol{\lambda}\mathbf{v} + \boldsymbol{\varepsilon} \quad (11)$$

where \mathbf{v} is an unobserved random vector such as $\mathbf{v} \sim \mathcal{N}(0, \gamma^2 \mathbf{I}_n)$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_n)$ and $\boldsymbol{\lambda} = \text{diag}(\lambda_1, \dots, \lambda_n)$. The estimation of the parameters of model (11) can be performed using the EM algorithm, as in [Searle *et al.* (1992)]. We adapt below the EM equations defined in [Searle *et al.* (1992)], using our notations. At iteration s of the internal EM algorithm, we define $\mathbf{R}^{(s)} = \sigma^{2(s)} \mathbf{I}_n + \gamma^{2(s)} \boldsymbol{\lambda}' \boldsymbol{\lambda}$. The detailed *internal E* and *M steps* are given below:

Internal E step:

$$\begin{aligned}
v_\sigma^{(s)} &= \mathbb{E} \left[\left(\mathbf{y}^u - \mathbf{M}\mathbf{t}^{(s)} - \boldsymbol{\lambda}\mathbf{v} \right)' \left(\mathbf{y}^u - \mathbf{M}\mathbf{t}^{(s)} - \boldsymbol{\lambda}\mathbf{v} \right) \middle| \mathbf{y}^u \right] \\
&= \sigma^{4(s)} \left(\mathbf{y}^u - \mathbf{M}\mathbf{t}^{(s)} \right)' \mathbf{R}^{(s)} \mathbf{R}^{(s)} \left(\mathbf{y}^u - \mathbf{M}\mathbf{t}^{(s)} \right) + n \times \sigma^{2(s)} - \sigma^{4(s)} \sum_{i=1}^n \frac{1}{\sigma^{2(s)} + \gamma^{2(s)} \lambda_i^2}. \\
v_\gamma^{(s)} &= \mathbb{E} [\mathbf{v}'\mathbf{v} | \mathbf{y}^u] \\
&= \gamma^{4(s)} \left(\mathbf{y}^u - \mathbf{M}\mathbf{t}^{(s)} \right)' \mathbf{R}^{(s)} \boldsymbol{\lambda}' \boldsymbol{\lambda} \mathbf{R}^{(s)} \left(\mathbf{y}^u - \mathbf{M}\mathbf{t}^{(s)} \right) + n \times \gamma^{2(s)} - \gamma^{4(s)} \sum_{i=1}^n \frac{\lambda_i^2}{\sigma^{2(s)} + \gamma^{2(s)} \lambda_i^2}. \\
\mathbf{h}^{(s)} &= \mathbb{E} [\mathbf{y}^u - \boldsymbol{\lambda}\mathbf{v} | \mathbf{y}^u] = \mathbf{M}\mathbf{t}^{(s)} + \sigma^{2(s)} \mathbf{R}^{-1(s)} \left(\mathbf{y}^u - \mathbf{M}\mathbf{t}^{(s)} \right).
\end{aligned}$$

Internal M step:

$$\begin{aligned}
\sigma^{2(s+1)} &= v_\sigma^{(s)} / n. \\
\gamma^{2(s+1)} &= v_\gamma^{(s)} / n. \\
\mathbf{t}^{(s+1)} &= [\mathbf{M}'\mathbf{M}]^{-1} \mathbf{M}'\mathbf{h}^{(s)}.
\end{aligned}$$

Given a non-negative user-specified threshold δ and a maximum number N_{max} of iterations, *Internal E* and *M steps* are alternated until

$$\left| \log p \left(\mathbf{y}, \mathbf{Z} | \mathbf{X}; \boldsymbol{\theta}^{(s)} \right) - \log p \left(\mathbf{y}, \mathbf{Z} | \mathbf{X}; \boldsymbol{\theta}^{(s+1)} \right) \right| < \delta \text{ or } s = N_{max}.$$

The computational complexity of the *M step* is $\mathcal{O}(g^3 + ngN_{max})$, thus not involving p .

Attracting and absorbing states

- *Absorbing states.* The SEM algorithm described above defines a Markov chain which stationary distribution is concentrated around values of $\boldsymbol{\theta}$ corresponding to local maxima of the likelihood function. This chain has absorbing states in values of $\boldsymbol{\theta}$ such as $\sigma^2 = 0$ or $\gamma^2 = 0$. In fact, the *internal M step* reveals that updated values for σ^2 and γ^2 are proportional to previous values of those parameters.
- *Attracting states.* We empirically observed that attraction around $\sigma^2 = 0$ was quite frequent when matrix \mathbf{X} is centered and $p > n$. To reduce this attraction, we advocate users of the package not to center the columns when p , the number of variables is smaller than n , the sample size. A similar behavior was also observed with the MCEM algorithm when $p > n$ and $M < 5$.

3.4. Model selection

Once the MLE $\hat{\boldsymbol{\theta}}$ is calculated (using one or the other algorithm), the maximum log-likelihood and the posterior clustering matrix $\mathbb{E} [\mathbf{Z} | \mathbf{y}, \mathbf{X}; \hat{\boldsymbol{\theta}}]$ are approximated using MC simulations based on Equations (9) and (10). The approximated maximum log-likelihood \hat{l} , is then utilized

to calculate AIC [Akaike (1974)] and BIC [Schwarz (1978)] criteria for model selection. In model (1), those criteria can be written as

$$\text{BIC} = -2\hat{l} + 2(g + 1) \log(n) \text{ and } \text{AIC} = -2\hat{l} + 4(g + 1). \quad (12)$$

An additional criterion for model selection, namely the ICL criterion [Biernacki, Celeux, and Goavert (2000)] is also implemented in the R package **clere**. The latter criterion can be written

$$\text{ICL} = -2\hat{l} + 2(g + 1) \log(n) - \sum_{j=1}^p \sum_{k=1}^g \pi_{jk} \log(\pi_{jk}), \quad (13)$$

where $\pi_{jk} = \mathbb{E} \left[z_{jk} | \mathbf{y}, \mathbf{X}; \hat{\boldsymbol{\theta}} \right]$.

3.5. Interpretation of the special group of variables associated with $b_1 = 0$

The constraint $b_1 = 0$ is mainly driven by an interpretation purpose. The meaning of this group depends on both the total number g of groups and the estimated value of parameter γ^2 . In fact, when $g > 1$ and γ^2 is small, covariates assigned to that group are likely less relevant to explain the response. Determining whether γ^2 is small enough is not straightforward. However, when this property holds, we may expect the groups of covariates to be separated. This would for example translate in the posterior probabilities π_{j1} being larger than 0.7. In addition to the benefit in interpretation, the constraint $b_1 = 0$, reduces the number of parameters to be estimated and consequently the variance of the predictions performed using the model.

4. Package functionalities

The R package **clere** mainly implements a function for parameter estimation and model selection: the function `fit.clere()`. Four additional functions for graphical representation `plot()`, summarizing the results `summary()`, getting the predicted clusters of variables `clusters()` and making predictions from new design matrices `predict()` are also implemented in the package.

4.1. The main function `fit.clere()`

The call of the R function `fit.clere()` is:

```
R> mod <- fit.clere(y, x, g, analysis = "fit", algorithm = "SEM",
+ nItMC = 1, nItEM = 1000, nBurn = 200, dp = 10, nsamp = 2000,
+ maxit = 1000, tol = 1e-6, plotit = FALSE, sparse = FALSE,
+ theta0 = NULL, Z0 = NULL)
```

Each of the input function arguments is described in Table 1. The `fit.clere()` function returns an R object of class **Clere**. In addition to all input parameters, this object has the other slots detailed in Table 2.

Argument name	Description
<code>y</code>	is a vector of response of size n
<code>x</code>	is a $n \times p$ matrix.
<code>g</code>	is either the exact number groups (when <code>analysis = "fit"</code>) or the maximum number of groups to be tested (when <code>analysis = "aic"</code> , <code>analysis = "bic"</code> or <code>analysis = "icl"</code>).
<code>analysis</code>	takes value in <code>{"fit","aic","bic","icl"}</code> . When <code>analysis = "fit"</code> , the model is fitted assuming that the exact number of groups is g . When instead, <code>analysis = "aic"</code> , <code>analysis = "bic"</code> or <code>analysis = "icl"</code> , the model is fitted g times with a number of group(s) between 1 and g . The g models are then compared respectively using AIC, BIC and ICL criterion and the best model is returned.
<code>algorithm</code>	allows to select between SEM (<code>algorithm = "SEM"</code>) and MCEM (<code>algorithm = "MCEM"</code>) algorithms to estimate the model parameters.
<code>nItMC</code>	is the number of complete Gibbs sampling (described in Equation (10)) iterations run before simulating a partition \mathbf{Z} when running SEM algorithm.
<code>nItEM</code>	is the number of SEM or MCEM iterations.
<code>nBurn</code>	is the number of iterations <i>burn-in</i> discarded to calculate the MLE in the SEM algorithm. This number is also used in the MCEM algorithm as a number of discarded iterations for the Gibbs sampling required to draw from $p(\mathbf{y}, \boldsymbol{\beta}, \mathbf{Z} \mathbf{X}; \boldsymbol{\theta})$.
<code>dp</code>	is the length of thinning interval used to break dependence between the sampled partitions. It corresponds to the number of Gibbs iterations to skip between consecutive draws of the chain.
<code>nsamp</code>	is the number of Gibbs iterations used to approximate the log-likelihood.
<code>maxit</code>	is the maximum number of <i>internal EM</i> algorithm iterations runned at the <i>M step</i> of the SEM algorithm.
<code>tol</code>	is the tolerance parameter required to end the <i>internal EM</i> algorithm runned at the <i>M step</i> of the SEM algorithm.

<code>nstart</code>	is the number of random starting points used to fit the model.
<code>parallel</code>	equals <code>FALSE</code> or <code>TRUE</code> . It indicates whether parameter the fit from different random starting points should be performed through parallel computing. It uses the function <code>mclapply</code> from package <code>parallel</code> .
<code>plotit</code>	equals <code>FALSE</code> or <code>TRUE</code> . This boolean parameter indicates whether a summary graph should be plotted.
<code>sparse</code>	equals <code>FALSE</code> or <code>TRUE</code> . It indicates whether parameter b_1 is set to 0.
<code>theta0</code>	is a user-specified initial guess of the model parameter θ .
<code>Z0</code>	is a user-specified initial partition of the variables given as a vector of size p of integers between 1 and g

Table 1: Input arguments of the function `fit.clere()`.

4.2. Secondary functions `summary()`, `plot()`, `ggPlot()`, `clusters()` and `predict()`

Examples of calls for the functions presented in this section are given in Section 5.3.

The `summary()` function prints an overview of the estimated parameters and returns the estimated likelihood and information based model selection criteria (AIC, BIC and ICL).

The call of functions `plot()` and `ggPlot()` are similar to the one of function `summary()`. The latter function produces graphs such as ones presented in Figure 1. The function `ggPlot()` requires a prior installation of the R package `ggplot2`. However, there is no dependencies with the latter package since the R package `clere` can be installed without `ggplot2`. When `ggplot2` is not installed, the user can still make use of the function `plot()`.

The function `clusters()`, takes one argument of class `Clere` and a `threshold` argument. This function assigns each variable to the group which associated posterior probability of membership is larger than the given `threshold`. When `threshold = NULL`, the maximum a posteriori (MAP) strategy is used to infer the clusters.

The `predict()` function has two arguments, being a `clere` and a design matrix \mathbf{X}_{new} . Using that new design matrix, the `predict()` function returns an approximation of $\mathbb{E}[\mathbf{X}_{new}\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}; \boldsymbol{\theta}]$.

5. Numerical experiments

This section presents two sets of numerical experiments. The first set of experiments aims at comparing the MCEM and SEM algorithms in terms of computational time and estimation or prediction accuracy. The second set of experiments aimed at comparing CLERE to standard

Argument name	Description
<code>intercept</code>	is the estimated value for parameter β_0 .
<code>b</code>	is the estimated value for parameter \mathbf{b} . It is a numeric vector of size g .
<code>pi</code>	is the estimated value for parameter $\boldsymbol{\pi}$. It is a numeric vector of size g .
<code>sigma2</code>	is the estimated value for parameter σ^2 .
<code>gamma2</code>	is the estimated value for parameter γ^2 .
<code>theta</code>	is a $\mathbf{nItEM} \times (2\mathbf{g}+4)$ matrix containing values of the model parameters and complete data log-likelihood at each iteration of the SEM/MCEM algorithm.
<code>likelihood</code>	is an approximation of the log-likelihood using <code>nsamp</code> MC simulations.
<code>entropy</code>	is an approximation of the entropy using <code>nsamp</code> MC simulations.
<code>P</code>	is a $p \times g$ matrix approximating $\mathbb{E}[\mathbf{Z} \mathbf{y}, \mathbf{X}; \boldsymbol{\theta}]$ using <code>nsamp</code> MC simulations.
<code>Bw</code>	is a $p \times \mathbf{nsamp}$ matrix which columns are samples from the distribution $p(\boldsymbol{\beta} \mathbf{y}, \mathbf{X}; \boldsymbol{\theta})$.
<code>Zw</code>	is a $p \times \mathbf{nsamp}$ matrix which columns are samples from the distribution $p(\mathbf{Z} \mathbf{y}, \mathbf{X}; \boldsymbol{\theta})$. Each column is sampled partition coded a vector of size p containing integers taking values between 0 and $(g - 1)$.

Table 2: The function `fit.clere()` function returns an R object of class `Clere`. In addition to all input parameters, this object has the other slots described in this table.

dimension reduction techniques. The latter comparison is performed on both simulated and real data.

5.1. SEM algorithm versus MCEM algorithm

Description of the simulation study

In this section, a comparison between the SEM algorithm and the MCEM algorithm is performed. This comparison is performed using the four following performance indicators:

1. Computational time (CT) to run a pre-defined number of SEM/MCEM iterations. This number was set to 2,000 in this simulation study.

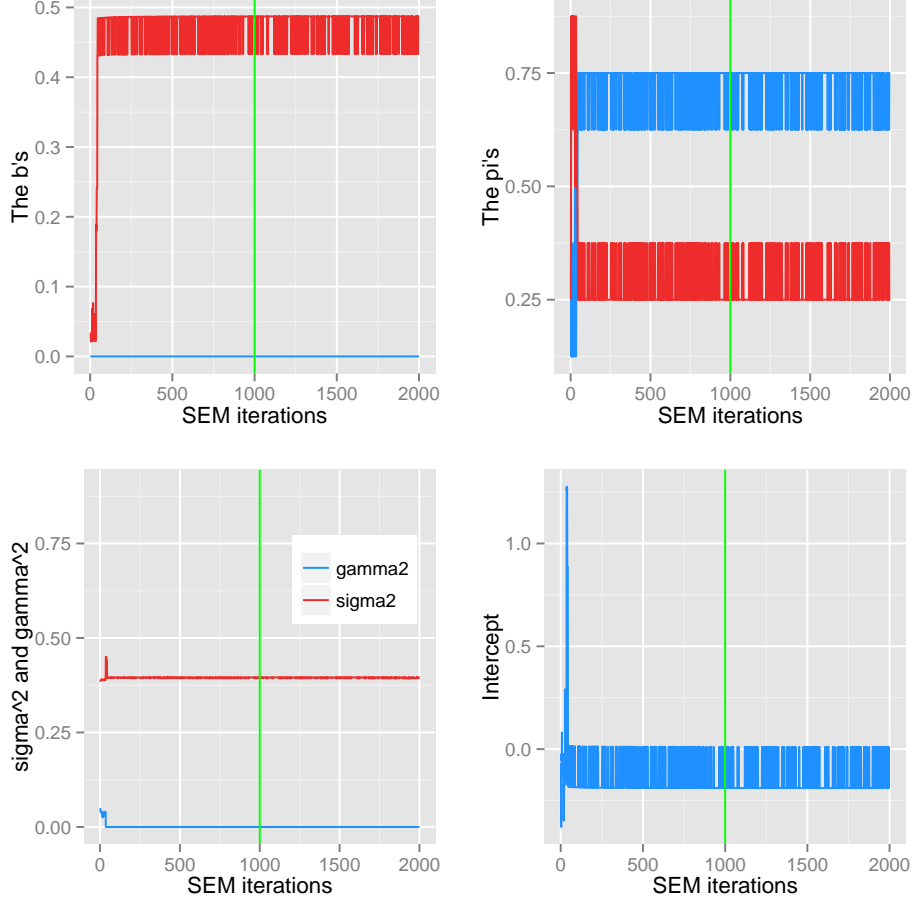


Figure 1: Values of the model parameters in view of SEM algorithm iterations. The vertical green line in each of the four plots, represents the number `nBurn` of iterations discarded before calculating maximum likelihood estimates.

2. Mean squared estimation error (MSEE) defined as

$$MSEE_a = \mathbb{E} \left[(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_a)' (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_a) \right],$$

where $a \in \{\text{"SEM"}, \text{"MCEM"}\}$ and $\hat{\boldsymbol{\theta}}_a$ is an estimated value for parameter $\boldsymbol{\theta}$ obtained with algorithm a . Since $\boldsymbol{\theta}$ is only known up to a permutation of the group labels, we chose the permutation leading to the smallest MSEE value.

3. Mean squared prediction error (MSPE) defined as

$$MSPE_a = \mathbb{E} \left[(\mathbf{y}^v - \mathbf{X}^v \hat{\boldsymbol{\theta}}_a)' (\mathbf{y}^v - \mathbf{X}^v \hat{\boldsymbol{\theta}}_a) \right],$$

where \mathbf{y}^v and \mathbf{X}^v are respectively a vector of responses and a design matrix from a validation dataset.

4. Maximum log-likelihood (ML) reached. This quantity was approximated using 1,000 samples from $p(\mathbf{Z} | \mathbf{y}; \hat{\boldsymbol{\theta}})$.

Three versions of the MCEM algorithm were proposed for comparison with the SEM algorithm, depending on the number M (or `nsamp`) of Gibbs iterations used to approximate the *E step*. That number was varied between 5, 25 and 125. Those versions were respectively denoted MCEM_5 , MCEM_{25} and MCEM_{125} . The comparison was performed using 200 simulated datasets. Each training dataset consisted of $n = 25$ individuals and $p = 50$ variables. Validation datasets used to calculate MSPE consisted of 1,000 individuals each. All covariates were simulated independently according to the standard Gaussian distribution:

$$\forall(i, j) x_{ij} \sim \mathcal{N}(0, 1).$$

Both training and validation datasets were simulated according to model (1) using $\beta_0 = 0$, $\mathbf{b} = (0, 3, 15)'$, $\boldsymbol{\pi} = (0.64, 0.20, 0.16)'$, $\sigma^2 = 1$ and $\gamma^2 = 0$. This is equivalent to simulate data according to the standard linear regression model defined by:

$$y_i \sim \mathcal{N} \left(\sum_{j=1}^{32} 0 \times x_{ij} + \sum_{j=33}^{42} 3 \times x_{ij} + \sum_{j=43}^{50} 15 \times x_{ij}, 1 \right)$$

All algorithms were run using 10 different random starting points. Estimates yielding the largest likelihood were then used for the comparison.

Results of the comparison

Table 3 summarizes the results of the comparison between the algorithms. The SEM algorithm ran faster than its competitors in 74.5% of the simulations. The gain in computational time yielded by SEM was between 1.3-fold (when compared to MCEM_5) and 22.2-fold (when compared to MCEM_{125}). This improvement was accompanied with a good accuracy in parameter estimation (second best median MSEE: 0.258; smallest MSEE in 25.5% of the simulations) and a smaller prediction error (smallest median MSPE: 1.237; smallest MSPE in 48.5% of the simulations). Those good performances were mainly explained by the fact that the SEM algorithm most of the time reached a better likelihood than the other algorithms.

5.2. Comparison with other methods

Description of the methods

In this section, we compare our model to standard dimension reduction approaches in terms of MSPE. Although a more detailed comparison was proposed in [Yengo *et al.* (2013)], we propose here a quick illustration of the relative predictive performance of our model. The comparison is achieved using data simulated according to the scenario described above in Section 5.1. The methods selected for comparison are the ridge regression [Hoerl and Kennard (1970)], the elastic net [Zou and Hastie (2005)], the LASSO [Tibshirani (1996)], PACS [Sharma *et al.* (2013)], the method of Park and colleagues [Park, Hastie, and Tibshirani (2007)] (subsequently denoted AVG) and the spike and slab model [Ishwaran and Rao (2005)] (subsequently denoted SS). The first three methods are implemented in the freely available R package `glmnet`. The latter package was used with default options regarding the choice of tuning parameters.

Performance indicators	Algorithms	% of times the algorithm was best	Median (Std. Err.)
CT (seconds)	SEM	74.50	1.60 (0.23)
	MCEM ₅	25.50	2.04 (0.13)
	MCEM ₂₅	0	7.63 (0.46)
	MCEM ₁₂₅	0	35.6 (2.22)
MSEE	SEM	25.5	0.258 (0.19)
	MCEM ₅	33.0	1.019 (0.97)
	MCEM ₂₅	22.5	0.257 (0.21)
	MCEM ₁₂₅	19.0	0.295 (0.25)
MSPE	SEM	48.5	1.237 (0.16)
	MCEM ₅	20.5	1.523 (0.49)
	MCEM ₂₅	19.0	1.258 (0.19)
	MCEM ₁₂₅	12.0	1.272 (0.21)
	True parameter	—	1.159 (0.08)
ML	SEM	59.5	-78.60 (3.60)
	MCEM ₅	10.5	-79.98 (5.78)
	MCEM ₂₅	18.0	-79.00 (3.84)
	MCEM ₁₂₅	12.0	-79.47 (4.20)
	True parameter	—	-77.60 (2.37)

Table 3: Performance indicators used to compare SEM and MCEM algorithms. Computational Time (CT) was measured on a Intel(R) Xeon(R) CPU E7- 4870 @ 2.40GHz processor. The best algorithm is defined as the one that either reached the largest log-likelihood (ML) or the lowest CT, Mean Squared Prediction Error (MSPE) and Mean Squared Estimation Error (MSEE). The best algorithm for each criterion is highlighted in bold font.

PACS methodology proposes to estimate the regression coefficients by solving a penalized least squares problem. It imposes a constraint on β that is a weighted combination of the L^1 norm and the pairwise L^∞ norm. Upper-bounding the pairwise L^∞ norm enforces the covariates to have close coefficients. When the constraint is strong enough, closeness translates into equality achieving thus a grouping property. For PACS, no software was available. Only an R script was released on Bondell's webpage¹. Since this R script was running very slowly, we decided to reimplement it in C++ and observed a 30-fold speed-up of computational time. Similarly to Bondell's R script, our implementation uses two parameters `lambda` and `betawt`. In [Sharma *et al.* (2013)], the authors suggest assigning `betawt` with the coefficients obtained from a ridge regression model after the tuning parameter was selected using AIC. In this simulation study we used the same strategy; however the ridge parameter was selected via 5-fold cross validation. 5-fold CV was preferred to AIC since selecting the ridge parameter using AIC always led to estimated coefficients equal to zero. Once `betawt` was selected, `lambda`

¹<http://www4.stat.ncsu.edu/~bondell/Software/PACS/PACS.R.r>

was chosen via 5-fold cross validation among the following values: 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200 and 500. All other default parameters of their script were unchanged.

The AVG method is a two-step approach. The first step uses hierarchical clustering of covariates to create surrogate covariates by averaging the variables within each group. Those new predictors are afterwards included in a linear regression model, replacing the primary variables. A variable selection algorithm is then applied to select the most predictive groups of covariates. To implement this method, we followed the algorithm described in [Park *et al.* (2007)] and programmed it in R.

The spike and slab model is a Bayesian approach for variable selection. It is based on the assumption that the regression coefficients are distributed according to a mixture of two centered Gaussian distributions with different variances. One component of the mixture (the spike) is chosen to have a small variance, while the other component (the slab) is allowed to have a large variance. Variables assigned to the spike are dropped from the model. We used the R package **spikeslab** to run the spike and slab models. Especially, we used the function **spikeslab** from that package to detect influential variables. The number of iterations used to run the function **spikeslab** was 2,000 (1,000 discarded).

When running `fit.clere()`, the number `nItEM` of SEM iterations was set to 2,000. The number `g` of groups for CLERE was chosen between 1 and 5 using AIC (option `analysis="aic"`). Two versions of CLERE were considered: the one with all parameters estimated and the one with b_1 set to 0. The latter approach is subsequently denoted CLERE₀ (option `sparse=TRUE`).

Results of the comparison

Figure 2, summarizes the comparison between the methods. In this simulated scenario, CLERE outperformed the other methods in terms of prediction error. Those good performances were improved when parameter b_1 was set to 0. CLERE was also the most parcimonous approach with an averaged number of estimated parameters equal to 8.5 (6.7 when $b_1 = 0$). The second best approach was PACS which also led to parcimonous models. Variables selection approaches as whole yielded the largest prediction error in this simulation.

5.3. Real datasets analysis

Description of the datasets

We used in this section the real datasets **Prostate** and **eyedata** from the R packages **lasso2** and **flare** respectively.

The **Prostate** dataset comes from a study that examined the correlation between the level of prostate specific antigen and a number of clinical measures in $n = 97$ men who were about to receive a radical prostatectomy. This dataset was used in multiple publications including [Tibshirani (1996)]. We used the prostate specific antigen (variable `lpsa`) as response variable and the $p = 8$ other measurements as covariates.

The **eyedata** dataset is extracted from the published study of [Scheetz (2006)]. This dataset consists in gene expression levels measured at $p = 200$ probes in $n = 120$ rats. The response variable utilized was the expression of the TRIM32 gene which is a biomarker of the Bardet-

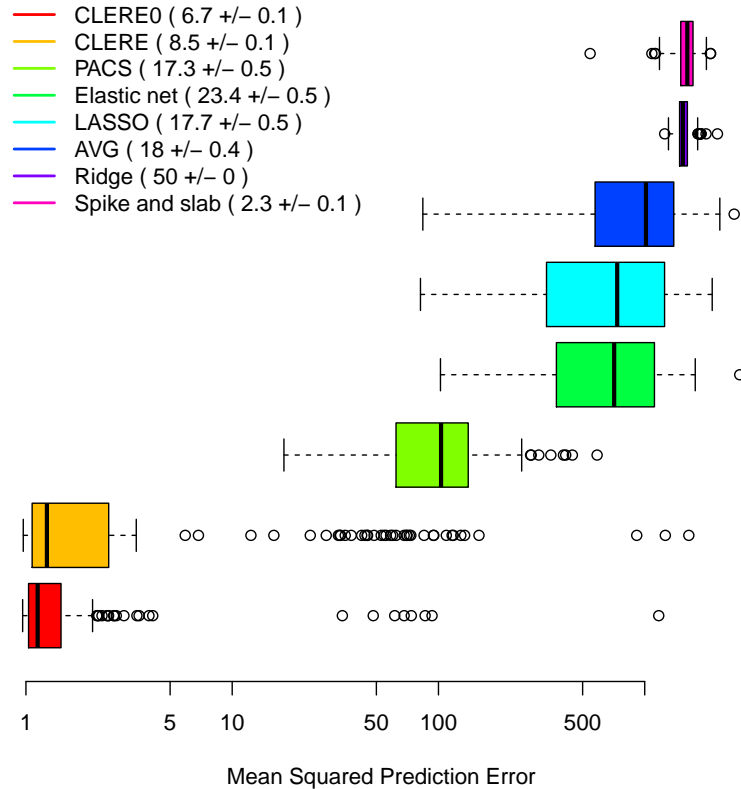


Figure 2: Comparison between CLERE and some standard dimension reduction approaches. The number of estimated parameters (+/- standard error) is given with the name of the method to be compared.

Bidel Syndrome (BBS).

Those two datasets was utilized to compare CLERE to the methods described in Section 5.2. All the methods were compared in terms of out-of-sample prediction error estimated using cross-validation (CV). 100 CV statistics were calculated by randomly splitting each dataset into training (80% of the sample size) and validation (20% of the sample size) sets. Those CV statistics were then averaged and compared accross the methods in Table 4.

Running the analysis

Before presenting the results of the comparison between CLERE and its competitors, we illustrate the command lines to run the analysis of the `Prostate` dataset. The dataset is loaded by typing:

```
R> library(lasso2)
R> data(Prostate)
R> y <- Prostate[, "lpsa"]
```



```
R> x <- as.matrix(Prostate[,-which(colnames(Prostate)=="lpsa")])
```

Possible training (xt and yt) and validation (xv and yv) sets are generated as following:

```
R> itraining <- 1:(0.8*nrow(x))
R> xt <- x[ itraining,] ; yt <- y[ itraining]
R> xv <- x[-itraining,] ; yv <- y[-itraining]
```

The `fit.clere()` function is run using AIC criterion to select the number of groups between 1 and 5. To lessen the impact of local minima in the model selection, 5 random starting points are used. This can be implemented as written below

```
R> mod <- fit.clere(y=yt,x=xt,g=5,analysis="aic",parallel=TRUE,
+                 nstart=5,sparse=TRUE,nItEM=2000,nBurn=1000,
+                 nItMC=10,dp=5,nsamp=1000)
R> summary(mod)
```

```
-----
| CLERE | Yengo et al. (2013) |
-----
```

```
Model object 2 groups of variables ( Selected using AIC criterion )
```

```
---
```

```
Estimated parameters using SEM algorithm are
```

```
intercept = -0.1395
b          = 0.0000  0.4737
pi         = 0.7188  0.2812
sigma2     = 0.3951
gamma2     = 4.181e-08
```

```
---
```

```
Log-likelihood = -78.28
Entropy        = 0.5152
AIC            = 182.63
BIC            = 168.57
ICL            = 183.15
```

```
R> plot(mod)
```

Running the command `ggPlot(mod)` generates the plot given in Figure 1. We can also access the cluster membership by running the command `clusters()`. For example, running the command `clusters(mod,threshold=0.7)` yields

```
R> clusters(mod,thresold=0.7)
lcavol lweight    age    lbph    svi    lcp gleason    pgg45
      2      2      1      1      1      1      1      1
```

In the example above 2 variables, being the cancer volume (`lcavol`) and the prostate weight (`lweight`), were assigned to group 2 ($b_2 = 0.4737$). The other 6 variables were assigned to group 1 ($b_1 = 0$). Posterior probabilities of membership are available through the slot `P` in object of class `Clere`.

```
R> mod@P
      Group 1 Group 2
lcavol  0.000  1.000
lweight  0.000  1.000
age      1.000  0.000
lbph     1.000  0.000
svi      0.789  0.211
lcp      1.000  0.000
gleason  1.000  0.000
pgg45    1.000  0.000
```

The covariates were respectively assigned to their group with a probability larger than 0.7. Moreover, given that parameter γ^2 had very small value ($\widehat{\gamma^2} = 4.181 \times 10^{-8}$), we can argue that cancer volume and prostate weight are the only relevant explanatory covariates. To assess the prediction error associated with the model we can run the command `predict()` as following:

```
R> error <- mean( (yv - predict(mod,xv))^2 )
R> error
[1] 1.550407
```

Results of the analysis

Table 4 summarizes the prediction errors and the number of parameters obtained for all the methods. `CLERE0` had the lowest prediction error in the analysis of the `Prostate` dataset and the second best performance with the `eyedata` dataset. The `AVG` method was also very competitive compared to variable selection approaches stressing thus the relevance of creating groups of variables to reduce the dimensionality. It is worth noting that in both datasets, imposing the constraint $b_1 = 0$ improved the predictive performance of `CLERE`.

In the `Prostate` dataset, `CLERE` robustly identified two groups of variables representing influential ($b_2 > 0$) and not relevant variables ($b_1 = 0$). In the `eyedata` dataset in turn, `AIC` led to select only one group of variables. However, this did not lessened the predictive performance of the model since `CLERE0` was second best after `AVG`, while needing significantly less parameters. `PACS` low performed in both datasets. The Elastic net showed good predictive performances compared to the variable selection methods like `LASSO` or `Spike and slab` model. Ridge regression and Elastic net had comparable results in both datasets.

6. Conclusions and Perspectives

Dataset	Methods	100×Averaged CV-statistic (Std. Error)	Number of parameters (Std. Error)
Prostate	LASSO	59.58 (3.46)	5.75 (0.29)
	RIDGE	57.58 (3.36)	8.00 (0.00)
	Elastic net	57.37 (3.39)	8.00 (0.00)
	CLERE	58.18 (3.13)	6.00 (0.00)
	CLERE₀	55.48 (3.46)	6.00 (0.00)
	AVG	60.59 (3.58)	6.30 (0.16)
	PACS	67.08 (5.51)	5.15 (0.30)
	Spike and slab	57.76 (3.21)	5.70 (0.28)
eyedata	LASSO	0.878 (0.05)	27 (1.69)
	RIDGE	0.854 (0.05)	200 (0.00)
	Elastic net	0.851 (0.05)	200 (0.00)
	CLERE	0.877 (0.06)	4 (0.00)
	CLERE ₀	0.839 (0.05)	4.12 (0.07)
	AVG	0.811 (0.06)	17.2 (0.98)
	PACS	2.019 (0.023)	1.38 (0.07)
	Spike and slab	0.951 (0.07)	11.5 (0.55)

Table 4: Real data analysis. Out-of-sample prediction error (averaged CV-statistic) was estimated using cross-validation in 100 splitted datasets. The number of parameters reported for CLERE/CLERE₀ was selected using AIC.

We presented in this paper the R package **clere**. This package implements an efficient algorithm for fitting the CLusterwise Effect REgression model. This algorithm, namely the SEM algorithm, was compared to a previously published approach and showed a significant improvement in computational time. The good performances of SEM over MCEM could have been expected regarding the computational complexities of the two algorithms that are $\mathcal{O}(npg + g^3 + N_{max}ng)$ and $\mathcal{O}(M(p^2 + pg))$ respectively. In fact, as long as $p > n$, the SEM algorithm has a lower complexity. However, the computational time to run our SEM algorithm is more variable compared to MCEM as its M step does not have a closed form. We finally advocate the use the MCEM algorithm only when $p < n$.

Another improvement was also proposed to facilitate the interpretation. This improvement was proposed by constraining the model parameter b_1 to equal 0. We illustrated through simulations that such constraint may also lead to a reduced prediction error. We illustrated on a real dataset, how to run an analysis using a detailed R script presented in Section 5.3. Our model can easily be extended to the analysis of binary responses. This extension will be proposed in forthcoming version of the package.

References

- Akaike H (1974). “A new look at the statistical model identification.” *Automatic Control, IEEE Transactions on*, **19**(6), 716–723.

- Biernacki C, Celeux G, Goavert G (2000). “Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(7), 719–725.
- Biernacki C, Jacques J (2013). “A generative model for rank data based on insertion sort algorithm.” *Computational Statistics and Data Analysis*, **58**, 162–176.
- Bondell HD, Reich BJ (2008). “Simultaneous Regression Shrinkage, Variable Selection, and Supervised Clustering of Predictors with OSCAR.” *Biometrics*, **64**, 115–123.
- Casella G (1985). “An Introduction to Empirical Bayes Data Analysis.” *The American Statistician*, **39**(2), 83–87.
- Celeux G, Chauveau D, Diebolt J (1996). “Some Stochastic versions of the EM Algorithm.” *Journal of Statistical Computation and Simulation*, **55**, 287–314.
- Dempster AP, Laird MN, Rubin DB (1977). “Maximum Likelihood from Incomplete Data via the EM Algorithm.” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **39**, 1–22.
- Govaert G, Nadif M (2008). “Block clustering with Bernoulli mixture models: Comparison of different approaches.” *Computational Statistics and Data analysis*, **52**, 3233–3245.
- Gunawardana A, Byrne W (2005). “Convergence theorems for generalized alternating minimization procedures.” *Journal of Machine Learning Research*, **6**, 2049–2073.
- Hoerl AE, Kennard W (1970). “Ridge Regression: Biased Estimation for Nonorthogonal Problems.” *Technometrics*, **12**, 55–67.
- Ishwaran H, Rao JS (2005). “Spike and slab variable selection: frequentist and Bayesian strategies.” *Annals of Statistics*, **33**(2), 730–773.
- Jolliffe IT (1982). “A Note on the Use of Principal Components in Regression.” *Applied Statistics*, **31**(3), 300+.
- Levine RA, Casella G (2001). “Implementations of the Monte Carlo EM Algorithm.” *Journal of Computational and Graphical Statistics*, **10**(3), 422–439.
- Mariadassou M, Robin S, Vacher C (2010). “Uncovering Latent Structure in Valued Graphs: a Variational Approach.” *The Annals of Applied Statistics*, **4**(2), 715–742.
- Park MY, Hastie T, Tibshirani R (2007). “Averaged gene expressions for regression.” *Biostatistics*, **8**, 212–227.
- Scheetz T (2006). “Regulation of gene expression in the mammalian eye and its relevance to eye disease.” *Proceedings of the National Academy of Sciences*, **103**(39), 14429.
- Schwarz G (1978). “Estimating the Dimension of a Model.” *Annals of Statistics*, **6**, 461–464.
- Searle S, Casella G, McCulloch C (1992). *Variance components*. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley.

- Sharma DB, Bondell HD, Zhang HH (2013). “Consistent Group Identification and Variable Selection in Regression with Correlated Predictors.” *Journal of Computational and Graphical Statistics. In Press.*, **22**(2), 319–340.
- Tibshirani R (1996). “Regression Shrinkage and Selection Via the Lasso.” *Journal of the Royal Statistical Society, Series B*, **58**, 267–288.
- Wei CG, Tanner M (1990). “A Monte Carlo Implementation of the EM Algorithm and the Poor Man’s Data Augmentation Algorithms.” *Journal of the American Statistical Association*, **85**, 699–704.
- Yengo L, Jacques J, Biernacki C (2013). “Variable clustering in high dimensional linear regression models.” *Journal de la Societe Francaise de Statistique. In Press.*
- Zou H, Hastie T (2005). “Regularization and variable selection via the Elastic Net.” *Journal of the Royal Statistical Society, Series B*, **67**, 301–320.

Affiliation:

Loic Yengo

Genomics and Molecular Physiology of Metabolic Diseases

CNRS UMR 8199 - Lille Institute of Biology

E.G.I.D - FR3508 European Genomics Institute of Diabetes

E-mail: loic.yengo@good.ibl.frURL: <http://www-good.ibl.fr/en/lyengo/>