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

by Loïc Yengo, Julien Jacques, Christophe Biernacki and Mickael Canouil

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 present here an R package that implements this methodology. An overview of the package functionalities as well as examples to run an analysis are described. Numerical experiments on real data were performed to illustrate the good predictive performance of our parsimonious method compared to standard dimension reduction approaches.

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 variable selection. Approaches for variable selection are implemented in publicly available software, that involve the well-known **R** packages **glmnet** [Friedman et al. (2010)] and **spikeslab** [Ishwaran et al. (2013)]. 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 variable selection.

Dimension reduction can not however be restricted to variable selection. Indeed, the field can be extended to include approaches which aim is to create surrogate covariates that summarize the information carried in initial covariates. Since the emblematic Principal Component Regression (PCR) [Jolliffe (1982)], many of the other 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 et al. (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 et al. (2014)], 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 $(b_1, \ldots, b_g)$ and equal variances 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 efficicient alternative to the Monte Carlo EM (MCEM) algorithm previously introduced in [Yengo et al. (2014)]. Secondly, the CLERE model is enhanced with the possibility of constraining the first component to have its mean equal to 0, i.e. $b1 = 0$. 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 provided γ^2 to be small enough. Those two new features were implemented in the **R** package **clere**. The core of the package is a **C++** program interfaced with **R** using **R** packages **Rcpp** [Eddelbuettel and François (2011)] and **RcppEigen** [Bates and Eddelbuettel (2013)]. The **R** package **clere** can be downloaded from the Comprehensive **R** Archive Network (CRAN) at http://cran.r-project.org/web/packages/clere/.

The outline of the present paper is the following. In the following section the definition of the model is recalled and the strategy to estimate the model parameter is presented. Afterwards, the main functionalities of the **R** package **clere** are presented. Real data analyses are then presented, aiming at illustrating the good predictive performances of CLERE, with noticeable parsimony ability, compared to standard dimension reduction methods. Finally, perspectives and further potential improvements of the package are discussed in the last section.

Model definition and notation

Our model is defined by the following hierarchical relationships:

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

where $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution of center μ and variance σ^2, and $\mathcal{M}\left(1, \pi_1, \ldots, \pi_g\right)$ the one-order multinomial distribution of parameters $\boldsymbol{\pi} = \left(\pi_1, \ldots, \pi_g\right)$ such as, $\forall\ k = 1, \ldots, g\ \pi_k > 0$ and $\sum_{k=1}^{g} \pi_k = 1$. For an individual $i = 1, \ldots, n$, 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 ($j = 1, \ldots, p$), which is assumed to follow a mixture of g Gaussians. The variable $\mathbf{z}_j$ indicates from which mixture component β_j is drawn ($z_{jk} = 1$ if β_j comes from component k of the mixture, $z_{jk} = 0$ otherwise). Let's note that model (1) can be considered as a variable selection-like model by constraining the model parameter b_1 to be equal to 0. Indeed, assuming that one of the components is centered in zero means that a cluster of regression coefficients have null expectation, and thus that the corresponding variables are not significant for explaining the response variable. This functionality is available in the package. Let $\boldsymbol{\beta} = \left(\beta_1, \ldots, \beta_p\right)$, $\boldsymbol{y} = (y_1, \ldots, y_n)'$, $\mathbf{X} = (x_{ij})$, $\boldsymbol{Z} = (z_{jk})$, $\boldsymbol{b} = (b_1 \ldots b_g)'$ and $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_g)'$. Moreover, $\log p(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{\theta})$ denotes the log-likelihood of model (1) assessed for the parameter $\boldsymbol{\theta} = \left(\beta_0, \boldsymbol{b}, \boldsymbol{\pi}, \sigma^2, \gamma^2\right)$. 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

$$\boldsymbol{Z} = \left(z_{jk}\right)_{1 \le j \le p, 1 \le k \le g} \in \mathcal{Z} \Leftrightarrow \forall j = 1, \ldots, p \begin{cases} \exists!\ k \text{ such as } z_{jk} = 1 \\ \text{For all } k' \neq k\ z_{jk'} = 0. \end{cases}$$

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

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

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 et al. (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 $p(\boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta}^{(d)})$, 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 et al. (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 et al. (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 $p(\boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta}^{(d)})$ providing thus a sample $(\boldsymbol{\beta}^{(d)}, \boldsymbol{Z}^{(d)})$. Note that the Monte Carlo algorithm we use to perform this simulation is the Gibbs sampler. After the *S step* follows the *M step* which consists in maximizing $p(\boldsymbol{\beta}^{(d)}, \boldsymbol{Z}^{(d)}|\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. Alternating those two steps generate a sequence $\left(\boldsymbol{\theta}^{(d)}\right)$, which is a Markov chain whose stationary distribution (when it exists) concentrates around a local maximum of the likelihood.

Estimation and model selection

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 $\boldsymbol{\beta}^{(0)} = \left(\beta_1^{(0)}, \ldots, \beta_p^{(0)}\right)$ 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 $\mathbf{Z}^{(0)} = \left(z_{jk}^{(0)}\right) \in \mathcal{Z}$ is obtained as

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

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

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

MCEM algorithm

The Stochastic Approximation of the E step

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

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

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

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

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

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

In Equation (2), 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 $p\left(\boldsymbol{\beta} | \boldsymbol{Z}, \boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta}^{(d)}\right)$ a preliminary singular vector decomposition of matrix $\boldsymbol{X}$ is necessary. Once this decomposition is performed the overall complexity of the approximated *E step* is $\mathcal{O}\left[M(p^2 + pg)\right]$.

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 $\mathcal{O}\left(Mpg\right)$.

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

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

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

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

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

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 $p(\boldsymbol{y}, \boldsymbol{Z}|\boldsymbol{X}; \boldsymbol{\theta})$ rather than $p(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{X}; \boldsymbol{\theta})$. A closed form of $p(\boldsymbol{y}, \boldsymbol{Z}|\boldsymbol{X}; \boldsymbol{\theta})$ is available and given subsequently.

Closed form of the integrated complete data likelihood

Let the SVD decomposition of matrix $\boldsymbol{X}$ be $\boldsymbol{USV'}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are respectively $n \times n$ and $p \times p$ orthogonal matrices, and $\boldsymbol{S}$ is $n \times p$ rectangular diagonal matrix which diagonal terms are the eigenvalues $\left(\lambda_1^2, \ldots, \lambda_n^2\right)$ of matrix $\boldsymbol{XX'}$. We now define $\boldsymbol{X}^u = \boldsymbol{U'X}$ and $\boldsymbol{y}^u = \boldsymbol{U'y}$. Let $\boldsymbol{M}$ be the $n \times (g+1)$ matrix which first column is made of 1's and which additional columns are those of matrix $\boldsymbol{X}^u\boldsymbol{Z}$. Let also $\boldsymbol{t} = (\beta_0, \boldsymbol{b}) \in \mathbb{R}^{(g+1)}$ and $\boldsymbol{R}$ 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

$$\log p\left(\boldsymbol{y}, \boldsymbol{Z}|\boldsymbol{X}; \boldsymbol{\theta}\right) = -\frac{n}{2}\log\left(2\pi\right) - \frac{1}{2}\sum_{i=1}^{n}\log\left(\sigma^2 + \gamma^2\lambda_i^2\right) - \frac{1}{2}\left(\boldsymbol{y}^u - \boldsymbol{Mt}\right)' \boldsymbol{R}^{-1}\left(\boldsymbol{y}^u - \boldsymbol{Mt}\right)$$
$$+ \sum_{j=1}^{p}\sum_{k=1}^{g} z_{jk}\log\pi_k. \tag{9}$$

Simulation step

To sample from $p\left(\boldsymbol{Z}|\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta}\right)$ we use a Gibbs sampling strategy based on the conditional distributions $p\left(\boldsymbol{z}_j|\boldsymbol{y}, \boldsymbol{Z}^{-j}, \boldsymbol{X}; \boldsymbol{\theta}\right)$, $\boldsymbol{Z}^{-j}$ denoting the set of cluster membership indicators for all covariates but the j-th. Let $\boldsymbol{w}^{-j} = \left(w_1^{-j}, \ldots, w_n^{-j}\right)'$, 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 $p(z_{jk} = 1|\boldsymbol{Z}^{-j}, \boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta})$ can be written

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

where $\boldsymbol{x}_j^u$ is the j-th column of $\boldsymbol{X}^u$. In the classical SEM algorithm, convergence to $p\left(\boldsymbol{Z}|\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta}\right)$ 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 $\mathcal{O}\left(npg\right)$, so linear with p and no more quadratic contrarily to the previous MCEM.
To improve the mixing of the generated Markov chain, we start the simulation step at each iteration by creating a random permutation of $\{1,\ldots,p\}$. Then, according to the order defined by that permutation, we update each z_{jk} using $p(z_{jk}=1|\boldsymbol{Z}^{-j},\boldsymbol{y},\boldsymbol{X};\boldsymbol{\theta})$.

Maximization step

$\log p\left(\boldsymbol{y},\boldsymbol{Z}|\boldsymbol{X};\boldsymbol{\theta}\right)$ corresponds to the marginal log-likelihood of a linear mixed model [Searle et al. (1992)] which can be written

$$\boldsymbol{y}^u = \boldsymbol{M}\boldsymbol{t} + \boldsymbol{\lambda}\boldsymbol{v} + \boldsymbol{\varepsilon} \tag{11}$$

where $\boldsymbol{v}$ is an unobserved random vector such as $\boldsymbol{v} \sim \mathcal{N}\left(0,\gamma^2\mathrm{I}_n\right)$, $\boldsymbol{\varepsilon} \sim \mathcal{N}\left(0,\sigma^2\mathrm{I}_n\right)$ and $\boldsymbol{\lambda} = \mathrm{diag}\left(\lambda_1,\ldots,\lambda_n\right)$. 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 $\boldsymbol{R}^{(s)} = \sigma^{2^{(s)}}\boldsymbol{I}_n + \gamma^{2^{(s)}}\boldsymbol{\lambda}'\boldsymbol{\lambda}$. The detailed *internal E* and *M steps* are given below:
Internal E step:

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

Internal M step:

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

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

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

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

Attracting and absorbing states

- *Absorbing states*. The SEM algorithm described above defines a Markov chain which stationnary 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 using the MCEM algorithm, especially when $p > n$ and when the number M of draws was small. We therefore advocate to use at least 5 draws ($M \geq 5$ using option nsamp= in the function fitClere).

Model selection

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

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

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

$$\mathrm{ICL} = \mathrm{BIC} - \sum_{j=1}^{p}\sum_{k=1}^{g} \pi_{jk}\log(\pi_{jk}), \tag{13}$$

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

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.

Package functionalities

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

The main function fitClere()

The main function fitClere() has only three mandatory arguments: the vector of response y (size n), the matrix of explanatory variable x (size $n \times p$) and the number g of groups of regression coefficients which is expected. The optional parameter analysis, when it takes the value aic, bic or icl, allows to test all the possible number of groups between 1 and g. The choice between the two proposed algorithms is possible thanks to the parameter algorithm, but we encourage the users to use the default value, the SEM algorithm, which generally overperforms the MCEM algorithm (see the first experiment of the next section).
Several other parameters allow to tune the different numbers of iterations of the estimation algorithm. Generally, higher are these parameters values, better is the quality of the estimation but heavier is the computing time. What we advice is to use the default values, and to graphically check the quality of the estimation with plots as in Figure 1: if the values of the model parameters are quite stable for a sufficient large part of the iterations, it is ok. If the stability is not reached sufficiently early before the end of the iterations, higher numbers of iterations should be chosen.
Finally, among the remaining parameters (the complete list can be obtained with help(fitClere)), two are especially important: parallel allows to run parallel computations (if compatible with the user's computer) and sparse allows to impose that one of the regression parameter is equal to 0 and thus to introduce a cluster of not significant explanatory variables.

Secondary functions summary(), plot(), clusters() and predict()

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 function plot() is similar to the one of function summary(). The latter function produces graphs such as ones presented in Figure 1.

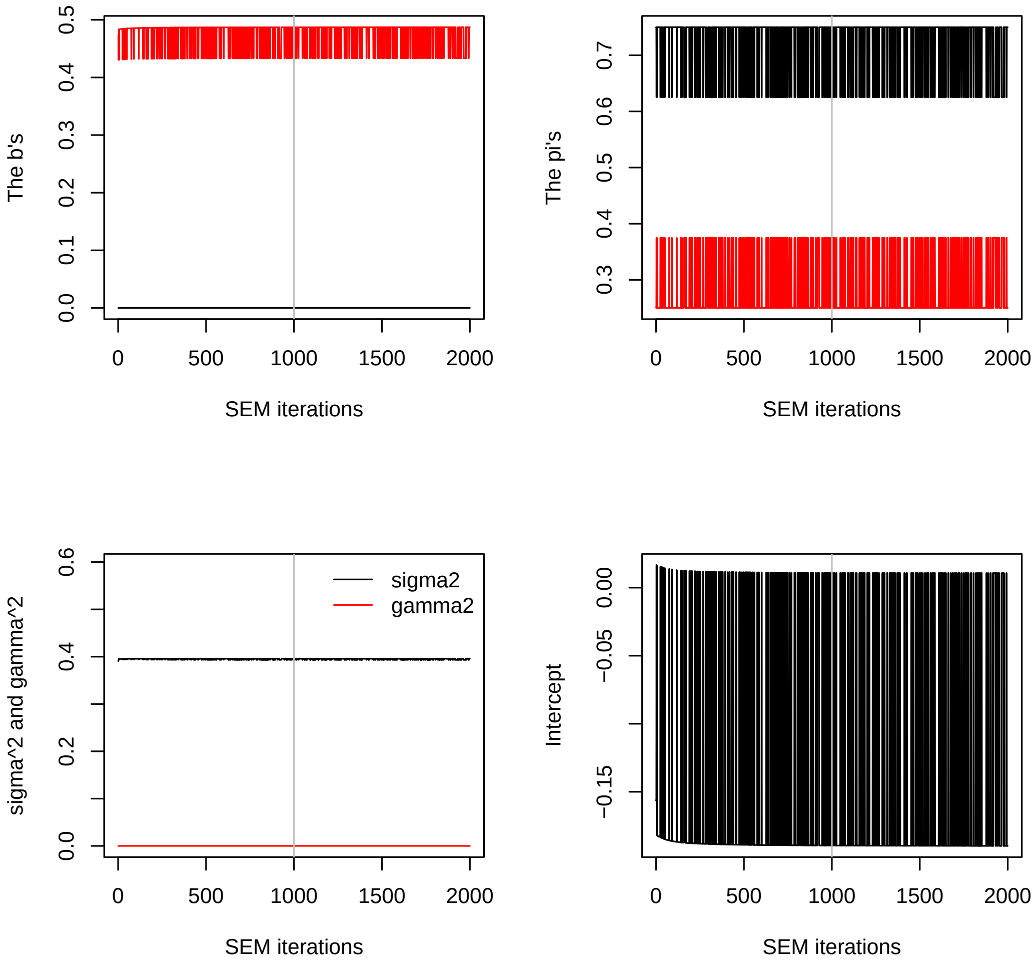

Figure 1: Values of the model parameters in view of SEM algorithm iterations. The vertical grey line in each of the four plots, represents the number nBurn of iterations discarded before calculating maximum likelihood estimates.

The function clusters(), takes one argument of class Clere and a threshold argument. This function assigns each variable to the group which associated conditional 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 $\boldsymbol{X}_{new}$. Using that new design matrix, the predict() function returns an approximation of $\mathbb{E}\left[\boldsymbol{X}_{new}\boldsymbol{\beta}|\boldsymbol{y},\boldsymbol{X};\hat{\boldsymbol{\theta}}\right]$.

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

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.
2. Mean squared estimation error (MSEE) defined as

   $$MSEE_a = \mathbb{E}\left[(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_{\boldsymbol{a}})'(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_{\boldsymbol{a}})\right],$$

   where $a \in \{"\text{SEM}","\text{MCEM}"\}$ and $\widehat{\theta}_a$ 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

   $$MSPE_a = \mathbb{E}\left[(\boldsymbol{y}^v - \boldsymbol{X}^v\widehat{\boldsymbol{\theta}}_{\boldsymbol{a}})'(\boldsymbol{y}^v - \boldsymbol{X}^v\widehat{\boldsymbol{\theta}}_{\boldsymbol{a}})\right],$$

   where $\boldsymbol{y}^v$ and $\boldsymbol{X}^v$ 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 $p(\boldsymbol{Z}|\boldsymbol{y};\widehat{\boldsymbol{\theta}})$.

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 $\beta_0 = 0$, $\boldsymbol{b} = (0,3,15)'$, $\boldsymbol{\pi} = (0.64, 0.20, 0.16)'$, $\sigma^2 = 1$ 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 1 summarizes the results of the comparison between the algorithms. The MCEM$_5$ algorithm was 1.3-fold faster than the SEM algorithm however the latter algorithm poorly performed regarding all other performance criteria (estimation quality, prediction error, likelihood maximization). This observation illustrates the importance of setting a large number M of draws to improve the estimation. Indeed, when increasing this number to 25 or 125, we observed an improvement in the estimation accuracy but no noticeable improvement in the likehood. In turn, the SEM algorithm was quite efficient compared to MCEM$_{25}$ and MCEM$_{125}$ algorithms. This algorithm not only ran faster (between 3 and 13-fold faster than MCEM$_{25}$ and MCEM$_{125}$ - see Table 1) but also reached predictive performances close to the oracle (i.e. using the true parameter). Those good performances were mainly explained by the fact that the SEM algorithm most of the time (66.5% of the time) reached a better likelihood than the other algorithms.
The results of this simulation study were made available as an internal dataset named algoComp in the **R** package **clere**. More details can be obtained using the command help(algoComp).

Performance indicators	Algorithms	% of times the algorithm was best	Median (Std. Err.)
CT (seconds)	SEM	0	2.5 (0.053)
	MCEM_5	100	1.9 (0.016)
	MCEM_{25}	0	7.1 (0.027)
	MCEM_{125}	0	32.8 (0.121)
MSEE	SEM	58	0.31 (10.4)
	MCEM_5	12	20.14 (2843.3)
	MCEM_{25}	16.5	8.86 (3107.5)
	MCEM_{125}	13.5	4.02 (5664.2)
MSPE	SEM	51.5	1.3 (46.1)
	MCEM_5	12	75.7 (64.3)
	MCEM_{25}	15.5	58.7 (55.2)
	MCEM_{125}	21	51.6 (51.1)
	True parameter	—	1.1 (0.013)
ML	SEM	66.5	-79.3 (1.2)
	MCEM_5	11.5	-110.7 (2.0)
	MCEM_{25}	14.5	-111.6 (1.9)
	MCEM_{125}	7.5	-116.2 (1.7)
	True parameter	—	-77.6 (0.34)

Table 1: 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).

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. (2014)], 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 ***Description of the simulation study***.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 et al. (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**. With the latter package, the tuning parameter lambda was selected using the function cv.glm aiming at minimizing the mean squared error (option type="mse"). In particular for the Elastic net, the second tuning parameter alpha (measuring of the amount of mixture between the L^1 and L^2 penalty) was jointly selected with lambda to minimize the mean squared error. The **R** package **glmnet** proposes a procedure for automatically selecting values for lambda. We therefore used this default procedure while we selected alpha among $\{0, 0.1, 0.2, \ldots, 0.9, 1\}$.

PACS methodology proposes to estimate the regression coefficients by solving a penalized least squares problem. It imposes a constraint on $\boldsymbol{\beta}$ 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 webpage[1]. 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. Our reimplementation of Bondell's script was included in the **R** package **clere** under the function fitPacs(). In [Sharma et al. (2013)], the authors suggest assigning

[1] http://www4.stat.ncsu.edu/~bondell/Software/PACS/PACS.R.r

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 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 fitClere(), 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 CLERE$_0$ (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 parsimonous approach with an averaged number of estimated parameters equal to 7.7 (6.9 when $b_1 = 0$). The second best approach was PACS which also led to parsimonous models. Variable selection approaches as whole yielded the largest prediction error in this simulation. CLERE, PACS and Spike and Slab had the largest computational times (CT). For CLERE and PACS this loss in CT were compensated by a a strong improvement in prediction error as explained above, while Spike and Slab yielded the worst prediction error in addition of being the slowest approach in this example.
The results of this simulation study were made available as an internal dataset named numExpSimData in the **R** package **clere**. More details can be obtained using the command help(numExpSimData).

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)] and was chosen because as one the most utilized dataset to illustrate the performances of the methods we used for comparison [T. et al. (2001)]. 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-Bidel Syndrome (BBS). We chose this dataset to illustrate the performances of CLERE on a (manageable) high-dimensional problem which is the actual context for which this method was developped [Yengo et al. (2014)].

Those two datasets was utilized to compare CLERE to the same methods used above in the Section presenting the simulation stydy. All methods were compared in term of out-of-sample prediction error estimated using 5-fold cross-validation (CV). Those CV statistics were then averaged and compared across the methods in Table 2.

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:

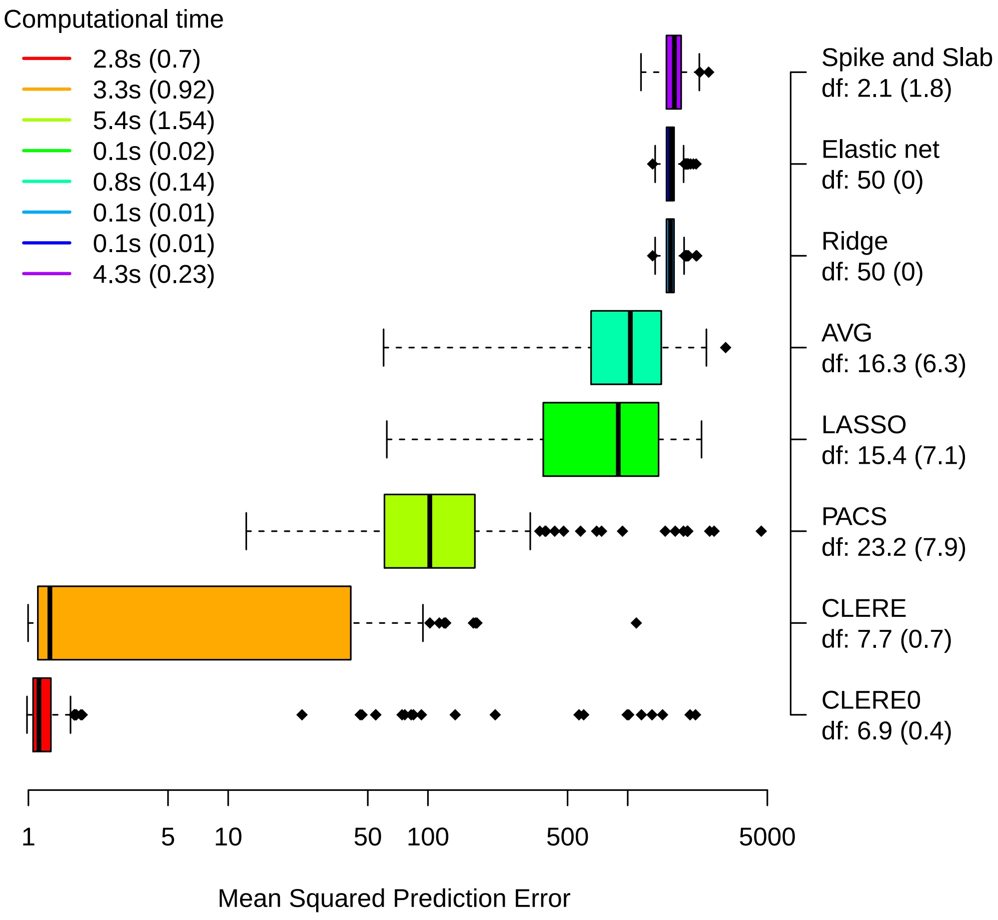

Figure 2: Comparison between CLERE and some standard dimension reduction approaches. The number of estimated parameters (df: +/- standard error) is given in the right along with the name of the method utilized. The average computational time with its corresponding standard error (given in parenthesis) is also provided in each situation.

```
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 follows:

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

The fitClere() 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> Seed <- 1234
R> mod  <- fitClere(y=yt,x=xt,g=5,analysis="aic",parallel=TRUE,
+                   nstart=5,sparse=TRUE,nItEM=2000,nBurn=1000,
+                   nItMC=10,dp=5,nsamp=1000,seed=Seed)
```

```
 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 = -0.1339
    b         = 0.0000   0.4722
    pi        = 0.7153   0.2848
    sigma2    = 0.395
    gamma2    = 4.065e-08

   ---
    Log-likelihood =  -78.31
    Entropy        =  0.5464
    AIC            =  168.63
    BIC            =  182.69
    ICL            =  183.23

R> plot(mod)
```

Running the command plot(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       0.764   0.236
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 ($\widehat{\gamma^2} = 4.065 \times 10^{-8}$), 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 follows:

```
R> error <- mean( (yv - predict(mod,xv))^2 )
R> error
[1] 1.543122
```

Results of the analysis

Table 2 summarizes the prediction errors and the number of parameters obtained for all the methods. CLERE$_0$ 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 (especially in the eyedata dataset). 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 CLERE$_0$ 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. In line with the results of the simulation study, we observed that despite the a larger computational time (CT), CLERE and CLERE$_0$ had a reduced mean squarred error compared to the fastest methods. However, this improvement was less substantial than observed in the simulation study given the differences in CT. This increased CT may be explained by the fact that no simple stopping rule is proposed when fitting CLERE. We may therefore consider that a smaller number of SEM iterations could have been used to yield a similar prediction error. Indeed, when looking at Figure 1, we see that the convergence was achieved almost from the first 10 iterations. Still, the observed CT for CLERE being around 22s for the eyedata dataset and around 3s for the Prostate dataset remains affordable in these examples.

The results of this analysis on real data were made available as an internal dataset named numExpRealData in the **R** package **clere**. More details can be obtained using the command help(numExpRealData).

	100×Averaged CV-statistic (Std. Error)	Number of parameters (Std. Error)	CT (seconds) (Std. Error)
Prostate dataset			
LASSO	90.2 (29)	5.6 (0.7)	0.064 (0.007)
RIDGE	86.8 (24)	8.0 (0)	0.065 (0.002)
Elastic net	90.3 (24)	8.0 (0)	0.065 (0.002)
STEP	442.4 (137)	8.0 (0)	0.004 (0.001)
CLERE	82.4 (25)	6.0 (0)	1.1 (0.1)
CLERE$_0$	74.5 (26)	5.0 (0)	2.7 (0.8)
Spike and Slab	85.6 (26)	5.6 (0.7)	4.2 (0.03)
AVG	90.6 (28)	6.0 (0.3)	0.43 (0.06)
PACS	91.3 (34)	6.4 (0.5)	0.062 (0.002)
eyedata			
LASSO	0.73 (0.1)	21.2 (2)	0.18 (0.01)
RIDGE	0.74 (0.1)	200.0 (0)	0.24 (0.004)
Elastic net	0.74 (0.1)	200.0 (0)	0.23 (0.003)
STEP	1142.6 (736)	95.0 (0)	0.083 (0.002)
CLERE	0.73 (0.1)	4.0 (0)	21.5 (0.2)
CLERE$_0$	0.72 (0.1)	3.0 (0)	21.1 (0.1)
Spike and Slab	0.81 (0.2)	12.4 (0.9)	10.3 (0.1)
AVG	0.79 (0.2)	12.6 (2)	10.5 (0.2)
PACS	2.1 (0.9)	2.8 (0.5)	106.7 (34)

Table 2: 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/CLERE$_0$ was selected using AIC. CT stands for the average Computational Time.

Conclusions

We presented in this paper the **R** package **clere**. This package implements two efficient algorithms for fitting the CLusterwise Effect REgression model: the MCEM and the SEM algorithms. If the MCEM algorithm is to be preferred when $p < n$, the SEM algorithm is more efficient for high dimensional datasets ($n < p$). The good performances of SEM over MCEM could have been expected regarding the computational complexities of the two algorithms that are $\mathcal{O}\left(npg + g^3 + N_{max}ng\right)$ and $\mathcal{O}\left(M(p^2 + pg)\right)$ 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 $p \ll n$. Another important feature for model interpretation is proposed by constraining the model parameter b_1 to equal 0, which leads to carry out variable selection. 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. Although our numerical experiments showed that the CLERE method tended to be slower than variable selection methods, it still had better or competitive predictive performances. In addition, the CLERE model was often more parsimonious than other models and was easily interpretable since groups of regression coefficients/variables could be summarized using a single parameter.

Our model can easily be extended to the analysis of binary responses. This extension will be proposed in forthcoming version of the package. Another direction for future research will be to develop an efficient stoping rule for the proposed SEM algorithm, specific to our context. Such a criterion is expected to improve the computational performances of our estimation algorithm.

Bibliography

H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974. [p6]

D. Bates and D. Eddelbuettel. Fast and elegant numerical linear algebra using the rcppeigen package. *Journal of Statistical Software*, 52(5):1–24, 2013. URL http://www.jstatsoft.org/v52/i05. [p1]

C. Biernacki and J. Jacques. A generative model for rank data based on insertion sort algorithm. *Computational Statistics and Data Analysis*, 58:162–176, 2013. [p4]

C. Biernacki, G. Celeux, and G. Goavert. Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):719–725, 2000. [p6]

H. D. Bondell and B. J. Reich. Simultaneous Regression Shrinkage, Variable Selection, and Supervised Clustering of Predictors with OSCAR. *Biometrics*, 64:115–123, 2008. [p1]

G. Casella. An Introduction to Empirical Bayes Data Analysis. *The American Statistician*, 39(2):83–87, 1985. [p2]

G. Celeux, D. Chauveau, and J. Diebolt. Some Stochastic versions of the EM Algorithm. *Journal of Statistical Computation and Simulation*, 55:287–314, 1996. [p2, 4]

A. P. Dempster, M. N. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 39:1–22, 1977. [p2]

D. Eddelbuettel and R. François. **Rcpp**: Seamless R and C++ Integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL http://www.jstatsoft.org/v40/i08/. [p1]

J. Friedman, T. Hastie, and R. Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL http://www.jstatsoft.org/v33/i01/. [p1]

G. Govaert and M. Nadif. Block clustering with Bernoulli mixture models: Comparison of different approaches. *Computational Statistics and Data analysis*, 52:3233–3245, 2008. [p2]

A. Gunawardana and W. Byrne. Convergence theorems for generalized alternating minimization procedures. *Journal of Machine Learning Research*, 6:2049–2073, 2005. [p2]

A. E. Hoerl and W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12:55–67, 1970. [p1, 9]

H. Ishwaran and J. S. Rao. Spike and slab variable selection: frequentist and Bayesian strategies. *Annals of Statistics*, 33(2):730–773, 2005. [p1, 9]

H. Ishwaran, J. Rao, and U. Kogalur. **spikeslab** *: Prediction and variable selection using spike and slab regression*, 2013. URL http://cran.r-project.org/web/packages/spikeslab/. R package version 1.1.5. [p1]

I. T. Jolliffe. A Note on the Use of Principal Components in Regression. *Applied Statistics*, 31(3):300+, 1982. [p1]

R. A. Levine and G. Casella. Implementations of the Monte Carlo EM Algorithm. *Journal of Computational and Graphical Statistics*, 10(3):422–439, 2001. [p2]

M. Mariadassou, S. Robin, and C. Vacher. Uncovering Latent Structure in Valued Graphs: a Variational Approach. *The Annals of Applied Statistics*, 4(2):715–742, 2010. [p2]

M. Y. Park, T. Hastie, and R. Tibshirani. Averaged gene expressions for regression. *Biostatistics*, 8: 212–227, 2007. [p9, 10]

T. Scheetz. Regulation of gene expression in the mammalian eye and its relevance to eye disease. *Proceedings of the National Academy of Sciences*, 103(39):14429, 2006. [p10]

G. Schwarz. Estimating the Dimension of a Model. *Annals of Statistics*, 6:461–464, 1978. [p6]

S. Searle, G. Casella, and C. McCulloch. *Variance components*. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley, 1992. [p5]

D. B. Sharma, H. D. Bondell, and H. H. Zhang. Consistent Group Identification and Variable Selection in Regression with Correlated Predictors. *Journal of Computational and Graphical Statistics.*, 22(2): 319–340, 2013. [p1, 9]

H. T., T. R., and F. J. H. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. New York: Springer-Verlag, 2001. [p10]

R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996. [p1, 9, 10]

C. G. Wei and M. Tanner. 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, 1990. [p2]

L. Yengo, J. Jacques, and C. Biernacki. Variable clustering in high dimensional linear regression models. *Journal de la Société Française de Statistique. In Press.*, 155(2):38–56, 2014. [p1, 9, 10]

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005. [p1, 9]

Loic Yengo
Integrated Genomics and Metabolic Diseases Modeling
CNRS UMR 8199 - Lille Institute of Biology
E.G.I.D - FR3508 European Genomics Institute of Diabetes
1, rue du Professeur Calmette, BP 447, 59021 Lille cedex
France loic.yengo@cnrs.fr

Julien Jacques
ERIC Laboratory
University of Lyon - Lumière
5 avenue Pierre Mendès France, 69676 Bron cedex
France julien.jacques@univ-lyon2.fr

Christophe Biernacki
MODAL team (Inria) & Laboratoire Paul Painlevé (UMR CNRS 8524)
University Lille I
Cité Scientifique, 59655 Villeneuve d'Ascq cedex
France christophe.biernacki@math.univ-lille1.fr

Mickael Canouil
Integrated Genomics and Metabolic Diseases Modeling
CNRS UMR 8199 - Lille Institute of Biology
E.G.I.D - FR3508 European Genomics Institute of Diabetes
1, rue du Professeur Calmette, BP 447, 59021 Lille cedex
France mickael.canouil@cnrs.fr