MLGL: An R package implementing correlated variable selection by hierarchical clustering and group-Lasso

Quentin Grimonprez, Samuel Blanck, Alain Celisse, Guillemette Marot

MLGL: An R package implementing correlated variable selection by hierarchical clustering and group-Lasso

Quentin Grimonprez [1]*, Samuel Blanck [3], Alain Celisse [1,2] and Guillemette Marot [1,3]
[1] MΘDAL team, Inria Lille-Nord Europe, France
[2] Laboratoire Paul Painlevé, Université de Lille, France
[3] EA 2694, Université de Lille, France

August 14, 2018

Abstract

The MLGL R-package, standing for Multi-Layer Group-Lasso, implements a new procedure of variable selection in the context of redundancy between explanatory variables, which holds true with high dimensional data. A sparsity assumption is made that is, only a few variables are assumed to be relevant for predicting the response variable. In this context, the performance of classical Lasso-based approaches strongly deteriorates as the redundancy strengthens.

The proposed approach combines variables aggregation and selection in order to improve interpretability and performance. First, a hierarchical clustering procedure provides at each level a partition of the variables into groups. Then, the set of groups of variables from the different levels of the hierarchy is given as input to group-Lasso, with weights adapted to the structure of the hierarchy. At this step, group-Lasso outputs sets of candidate groups of variables for each value of regularization parameter.

The versatility offered by MLGL to choose groups at different levels of the hierarchy a priori induces a high computational complexity. MLGL however exploits the structure of the hierarchy and the weights used in group-Lasso to greatly reduce the final time cost. The final choice of the regularization parameter – and therefore the final choice of groups – is made by a multiple hierarchical testing procedure.

keywords: penalized regression, correlated variables, hierarchical clustering, group selection

1 Introduction

In the high-dimensional setting where the number of variables p is larger than the sample size n, variable selection becomes a challenging problem which is often addressed by regularization procedures such as Lasso [Tibshirani, 1994, Tibshirani et al., 2005, Yuan and Lin, 2006]. These procedures have become very popular since they are specifically designed to select a subset of the explanatory variables for predicting the response. Nevertheless, high dimension raises several problems such as the high correlation level between variables. For instance correlation can be responsible for the apparent instability of the selected variables which can change from one draw to another [Meinshausen and Bühlmann, 2010]. The present work tackles the problem of variable selection in the high-dimensional setting with a strong correlation between explanatory variables.

*to whom correspondence should be addressed: quentin.grimonprez@inria.fr

Let X denote a $n \times p$ matrix where each column vector $X_j \in \mathbb{R}^n$ $(1 \leq j \leq p)$ corresponds to the values of the jth variable measured on n individuals. The quantitative response vector $y \in \mathbb{R}^n$ is then related to X through the linear regression model

$$y = X\beta^* + \epsilon, \tag{1}$$

where $\epsilon \sim \mathcal{N}_n(0, \sigma^2 I_n)$ is a Gaussian vector (noise), and $\beta^* \in \mathbb{R}^p$ is the parameter vector encoding the influence of each of the p candidate variables on the response y. The intercept of the regression model is removed by assuming X_j is centered for all $j = 1, \ldots, p$.

Moreover, the parameter vector β^* is assumed to be sparse that is, the cardinality of its support $S^* = S(\beta^*) = \{1 \leq j \leq p \mid \beta_j^* \neq 0\}$ is such that

$$\mathrm{Card}(S^*) = k \ll p.$$

This is consistent with the goal of identifying a small subset of interpretable (groups of) variables which turn to be relevant in explaining the response.

The first naive approach for estimating β^* from Eq. (1) is to compute the minimizer of the least squares error

$$\beta^{\mathrm{LS}} \in \operatorname*{argmin}_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2} \|y - X\beta\|_2^2 \right\}. \tag{2}$$

However in the present high-dimensional context where $p \gg n$, there are infinitely many solutions to this problem and most of them are certainly not sparse.

The Lasso procedure [Tibshirani, 1994] is generally used to perform variable selection in this high-dimensional setting. Unlike the above least squares minimization problem, a regularization term consisting of the ℓ_1-norm of the estimated vector (the penalty) is added to get a unique and sparse solution to the following optimization problem:

$$\beta_\lambda^{\mathrm{Lasso}} = \operatorname*{argmin}_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}, \tag{3}$$

where $\lambda > 0$ is called the regularization parameter and controls the amount of shrinkage. For instance, a large value of λ yields an estimator with only a few non-zero coefficients. In practice, the calibration of λ can be done by means of V-fold cross-validation [Arlot and Celisse, 2010] or various information criteria such as AIC, BIC, ...

Although (asymptotic) consistency results on the selected variables have been proven [Zhao and Yu, 2006], establishing such consistency results with highly correlated variables remains highly challenging or even impossible if the correlation is too strong [Wainwright, 2009]. Intuitively, Lasso selects one (or a few) variable(s) among each group of correlated variables as long as the correlation is strong enough, even if all these variables belong to the true support S^*. In such a case grouping correlated variables turns out to be necessary to select meaningful groups of influential variables. The group-Lasso [Yuan and Lin, 2006] was precisely developed for taking into account the *a priori* knowledge of groups of (correlated) variables. More precisely given a partition of the p candidate variables into g groups $\mathcal{G} = \{G_1, \ldots, G_g\}$, the group-Lasso estimator is defined by

$$\beta_\lambda^{\mathcal{G}} = \operatorname*{argmin}_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \sum_{i=1}^{g} w_i \|\beta_{G_i}\|_2 \right\}, \tag{4}$$

where $\lambda > 0$ is the regularization parameter, and $w_i > 0$ denotes the weight associated with the group G_i (generally $w_i = \sqrt{\text{Card}(G_i)}$). Obviously, the statistical performance of the group-Lasso estimator strongly depends on the partition $\mathcal{G}$ that has to be known *a priori*. When no such knowledge is available regarding groups of correlated variables, a preliminary step aiming at providing a meaningful partition of the candidate variables is crucial.

Several strategies such as first grouping candidate variables and then selecting groups by Lasso or group-Lasso have been studied in the literature. Most of them rely on hierarchical clustering at the first stage where only one level of the hierarchy is chosen (resulting in a partition of the candidate variables). For example [Park et al., 2007] perform hierarchical clustering first. Then Lasso is successively applied to each level of the hierarchy where each candidate group is summarized by a representative variable. Both the hierarchy level and the subset of groups from the corresponding partition are selected by cross-validation. By contrast, Cluster Representative Lasso and Cluster Group-Lasso [Bühlmann et al., 2013] apply hierarchical clustering and choose first one particular level of the hierarchy. Then groups from this partition are selected either by using Lasso (applied to representative variables of each group) or by using the corresponding partition as an input of group-Lasso. Let us also mention alternative strategies such as Supervised Group-Lasso [Ma et al., 2007] and Cluster Elastic Net [Witten et al., 2014] to name but a few. One main contribution of the present work is to relax the dependence of the final selected (groups of) variables on a particular level of the hierarchy. The main asset is some robustness to possible mistakes resulting from the iterative clustering process. Our procedure combines hierarchical clustering and group selection by allowing group-Lasso for selecting groups from different hierarchy levels that is, from different partitions of the candidate variables.

The following of the paper is organized as follows. Section 2 introduces the whole procedure that is successively based on hierarchical clustering (AHC), group-Lasso (gLasso), and a post-treatment selection involving hierarchical multiple testing (HMT). Then, the usage of the R-package MLGL is described in Section 3. The statistical performance of the procedure is assessed in Section 4 by comparison to alternative ones. Finally, some conclusions and perspectives are discussed in Section 5.

2 Overview of the MLGL package

Generally group-Lasso is applied with only one prescribed partition of the variables into groups (corresponding in the present context to one particular level of the hierarchy). One main originality of the present package is to select groups of variables by applying group-Lasso to several partitions at the same time. A possible resulting issue is the presence of overlapping groups in the partitions given as inputs to group-Lasso.

The whole procedure implemented in the *MLGL package* (standing for Multi-Layer Group-Lasso) consists of four main steps:

1. Building a hierarchy (hierarchical clustering),
2. Computing the path of groups selected by group-Lasso with respect to $\lambda > 0$ (the regularization parameter),
3. Performing hierarchical multiple testing (HMT) to remove false positive groups for each λ,
4. Tuning λ to select the final groups of influential variables.

These different steps are detailed in what follows.

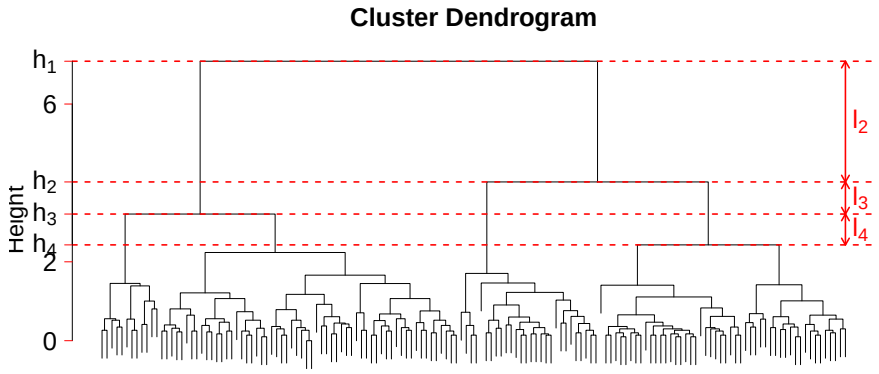


Figure 1: Dendrogram obtained using a hierarchical algorithm.

2.1 Building a hierarchy

Two main families of methods co-exist for performing (unsupervised) clustering: *hierarchical clustering* algorithms and the so-called *partitional* algorithms (see [Jain et al., 1999] for a review). The main difference lies in that partitional algorithms return only one partition of the candidate variables into a prescribed number of groups (k-means for instance), whereas hierarchical clustering algorithms yield a nested hierarchy of partitions of the candidate variables. This hierarchy can be represented by a dendrogram (Figure 1), so that each hierarchy level defines a partition of the candidate variables into groups. Moreover the hierarchy enjoys the property that each group at a given level can be split into sub-groups located at different sub-levels of this hierarchy as illustrated by Fig. 1.

The general process of hierarchical clustering is summarized in Pseudo-code 1. A similarity

Pseudo-code 1 Ascendent Hierarchical Clustering (AHC)

Input: Candidate variables, similarity measure

Compute the distance matrix between all variables.
Place each variable in its own group.
repeat
Aggregate the two nearest groups according to the similarity measure.
until all the variables belong to the same group.

Return: Dendrogram

measure has to be specified and determines the order in which (groups of) variables will be aggregated. Classical similarity measures are the Ward's criterion (which minimizes the total within-group variance) and the average linkage (which aggregates the two groups minimizing the average distance between each pair of points (one from each group)).

Considering the level $s \in \{1, \ldots, p\}$ of the hierarchy where the variables are partitioned into s groups, let h_s denote the value of the similarity measure between the two groups merged for obtaining the partition with s groups, and the jump size $l_s = h_{s-1} - h_s$ (see Figure 1). Choosing the number of groups can be performed following the *highest jump* rule, which consists in choosing the partition $\mathcal{G}_{\hat{s}}$ such that

$$\hat{s} = \underset{s}{\operatorname{argmax}}\{l_s\}. \tag{5}$$

Intuitively, a large value of l_s indicates that the groups merged from level s to $s-1$ were far apart according to the similarity measure. This explains why the partition with s groups is usually preferred in this setting.

In the MLGL package, there is no need to choose the number of groups output from the hierarchical clustering since all levels of the hierarchy are kept as an input of group-Lasso. The latter selects simultaneously the number of groups as well as the groups. Nevertheless, the jump sizes are exploited as weights within the group-Lasso procedure, which turns out to reduce the whole computational cost (see Section 2.2).

2.2 Computing the path of candidate groups

One main originality of the MLGL package is to simultaneously provide the groups from all levels of the hierarchy as an input to group-Lasso. The resulting procedure should be less sensitive to possible mistakes induced by the iterative clustering process.

Since no selection of a particular hierarchy level is made, numerous overlapping groups arise in the input of group-Lasso. With overlapping groups, [Jacob et al., 2009] designed a overlap group-Lasso penalty and expressed it in such a way they could apply classical algorithms to minimize the group-Lasso problem to solve the overlap group-Lasso problem. The trick is exposed in what follows.

From a collection $\mathcal{G} = \{G_1, \ldots, G_g\}$ of $g \in \mathbb{N}^*$ groups of indices such that $G_i \subset \{1, \ldots, p\}$, for all $i = 1, \ldots, g$, let us introduce X_{G_i} as the $n \times \text{card}(G_i)$ matrix obtained by concatenating the columns of X corresponding to variables with indices in G_i. Let also $X^{\mathcal{G}} = [X_{G_1}, X_{G_2}, \ldots, X_{G_g}]$ denote the $n \times l$ extended design matrix defined as the concatenation of the matrices $X_{G_1}, X_{G_2}, \ldots, X_{G_g}$, where $l = \sum_{i=1}^{g} \text{card}(G_i)$. Then the overlap group-Lasso estimator built from the design matrix X and the collection $\mathcal{G}$ can be expressed as a group-Lasso estimator with extended design matrix $X^{\mathcal{G}}$ as

$$\hat{\beta}_\lambda^{\mathcal{G}} = \underset{\beta \in \mathbb{R}^{pl}}{\text{argmin}} \left\{ \frac{1}{2} \|y - X^{\mathcal{G}} \beta\|_2^2 + \lambda \sum_{i=1}^{g} w_i \|\beta_{G_i}\|_2 \right\}, \tag{6}$$

where $\lambda > 0$ is the regularization parameter and w_i denotes a weight associated with G_i. This rephrasing allows for using all the partitions output by the hierarchical clustering as an input of group-Lasso.

Considering the dendrogram output by hierarchical clustering, let $\mathcal{G}_s$ be the partition of the p candidate variables into s groups, for $1 \leq s \leq p$, and $\mathcal{G}_* = \cup_{s=1}^{p} \mathcal{G}_s$ denote the union of all the partitions at the different levels of the hierarchy. Then the above Eq. (6) applied with $\mathcal{G} = \mathcal{G}_*$ leads to

$$\hat{\beta}_\lambda^{\mathcal{G}_*} = \underset{\beta \in \mathbb{R}^{p^2}}{\text{argmin}} \left\{ \frac{1}{2} \|y - X^{\mathcal{G}_*} \beta\|_2^2 + \lambda \sum_{s=1}^{p} \rho_s \sum_{i=1}^{g_s} w_i^s \|\beta_{G_i^s}\|_2 \right\}, \tag{7}$$

where G_i^s is the ith group of the partition $\mathcal{G}_s$ and $\mathcal{G}_s = \cup_{i=1}^{g_s} G_i^s$, $X^{\mathcal{G}_*} = [\underbrace{X, \ldots, X}_{p \text{ times}}]$ denotes the corresponding extended design matrix, and ρ_s is a weight encoding how likely $\mathcal{G}_s$ is a meaningful partition of the candidate variables.

It is worth noticing that Eq. (7) shows that the present approach is included in the general framework described in [Jenatton et al., 2011], where penalties are designed to define groups according to a prescribed structure in the support of β^*.

Choice of ρ_s For $s = 1, \ldots, p$, ρ_s is a weight reflecting the quality of the partition $\mathcal{G}_s$. This weight must weakly penalize a "good" partition and heavily penalize a "bad" one. The MLGL package uses a weight ρ_s inspired from the somewhat classical highest jump rule that is, a small weight is given to partitions with a large jump size l_s. More precisely,

$$\rho_s = \frac{1}{\sqrt{l_s}} \cdot \tag{8}$$

It is important to keep in mind that this definition of ρ_s promotes the selection of groups belonging to the partition with the largest jump size. But the described procedure remains free to select groups from different partitions (from different hierarchy levels).

Storage improvement From the reformulation in Eq. (7), it clearly arises that several duplications of the $n \times p$ design matrix X are used. The extended design matrix $X^{\mathcal{G}_*}$ has size $n \times p^2$ when all the levels from the hierarchy are kept as an input. In usual high-dimensional settings, the p^2 columns induce a prohibitive computational cost both in space and time. Therefore, the MLGL package exploits the redundancy of the partitions along the hierarchy to drastically reduce the computational costs.

On the one hand, let us notice that two successive partitions from a hierarchy — say $\mathcal{G}_s$ and $\mathcal{G}_{s-1}$ the ones with respectively s and $s-1$ groups — share $s-2$ common groups: At each step of the hierarchical clustering process, only two groups are aggregated while the others remain unchanged. On the other hand, these groups (which remain the same from a level $\mathcal{G}_{s-1}$ to the next one $\mathcal{G}_s$) are penalized with a different weight depending on the partition they belong to. More precisely, each such group is weighted once with ρ_s and once with ρ_{s-1}. The following Lemma 1 establishes that if $\rho_{s-1} \neq \rho_s$, then only the group with the smallest weight has a chance to be selected. The proof is given in Appendix A.

Lemma 1. *With the notations of Eq.* (6)*, let $\mathcal{G}$ denote any collection of g subsets (groups) of $\{1, \ldots, p\}$ that are not necessarily disjoint and assume that there exist $G_1, G_2 \in \mathcal{G}$ such that $G_1 = G_2$, with $w_2 > w_1 > 0$.*

Then the solution $\hat{\beta}^{\mathcal{G}}_\lambda \in \mathbb{R}^l$ of Eq. (6) *satisfies that the subset of its coordinates corresponding to G_2 is equal to zero that is, $(\hat{\beta}^{\mathcal{G}}_\lambda)_{G_2} = 0$.*

From several copies of the same group with different weights, only the one with the smallest weight is worth considering according to Lemma 1. This justifies simplifying the optimization problem from Eq. (7) to drastically reduce the induced computational costs.

Let us define $\mathcal{G}^{\mathrm{u}}_*$ as the collection of all the *distinct* groups output from hierarchical clustering (without including copies) that is,

$$\mathcal{G}^{\mathrm{u}}_* = \bigcup_{i=1}^{2p-1} G^{\mathrm{u}}_i, \quad \text{such that} \qquad \forall 1 \leq i \neq j \leq 2p-1, \quad G^{\mathrm{u}}_i \neq G^{\mathrm{u}}_j.$$

This new collection $\mathcal{G}^{\mathrm{u}}_*$ exactly contains $2p-1$ distinct groups: p groups made of one variable from the pth level of the hierarchy (the leaves of the dendrogram), and one new group from each other level (there are $p-1$ of them). The resulting extended design matrix $X^{\mathcal{G}^{\mathrm{u}}_*}$ is clearly less space demanding than the former $X^{\mathcal{G}_*}$. Consistently with the above remarks, the optimization problem from Eq. 7 can be equivalently reformulated as

$$\hat{\beta}^{\mathcal{G}^{\mathrm{u}}_*}_\lambda = \operatorname*{argmin}_\beta \left\{ \frac{1}{2}\|y - X^{\mathcal{G}^{\mathrm{u}}_*}\beta\|_2^2 + \lambda \sum_{i=1}^{2p-1} \rho^{\mathrm{u}}_i w^{\mathrm{u}}_i \|\beta_{G^{\mathrm{u}}_i}\|_2 \right\}, \tag{9}$$

with $\lambda > 0$ the regularization parameter, w_i^{u} the weight associated with G_i^{u}, and ρ_i^{u} the smallest weight associated with one partition containing G_i^{u}, that is

$$\rho_i^{\mathrm{u}} = \min \{\rho_s \mid s \in 1, \ldots, p \text{ such that } G_i^{\mathrm{u}} \in \mathcal{G}_s\}.$$

Since this simplified problem is an instance of group-Lasso as earlier discussed at Eq. (6), the MLGL package solves Eq. (9) by means of classical optimization algorithms solving the group-Lasso problem [Yang and Zou, 2015]. In particular, such an algorithm gives access to the whole path $\lambda \mapsto \hat{\beta}_\lambda^{\mathcal{G}_*^{\mathrm{u}}}$ of the candidate groups selected by group-Lasso for each λ.

2.3 Hierarchical Multiple Testing

For each λ, the previous step returns a set of selected groups of variables from which, most of the time, an additional filtering step is required for two main reasons. First, it is well known that in the high-dimensional context where the number of (groups of) variables is larger than n and only a few candidate variables are likely to be influential (sparsity assumption), then Lasso and its extensions can only identify most of the true variables at the price of including false positives among the selected ones [Wainwright, 2009, Barber et al., 2015]. Second, the solution of Eq. (9) contains groups potentially located at different levels of the hierarchy. Furthermore some groups can even be sub-groups of some others as explained by Figure (2) (redundancy of groups). Then choosing which one from the group or its sub-group should be selected has to be done by an additional dedicated step.

For all these reasons, the MLGL package applies a hierarchical multiple testing procedure (HMT) which selects the final groups for each value of λ. The choice of the regularization parameter λ is discussed in Section 2.4. The next two paragraphs review the main goals the HMT procedure achieves for a given value of λ: (i) reducing the number of selected groups, and (ii) avoiding the redundancy of groups.

2.3.1 Reducing the number of groups

With Lasso, [Wasserman and Roeder, 2009] suggest to perform a least squares estimation of the coefficients of the selected variables, so that they test the nullity of each coefficient by means of multiple testing procedures. Adjusted p-values are computed for controlling the Family-Wise Error Rate (FWER) [Dunn, 1959] or the False Discovery Rate (FDR) [Benjamini and Hochberg, 1995].

With group-Lasso, it can happen that more variables than individuals are selected at a given λ value (in particular when λ is very close to 0). A least squares estimation cannot be directly performed in this situation. This issue can be overcome by first summarizing each selected group by one representative variable and then performing least squares estimation using these representative variables. Note that this is always possible since the number of selected groups cannot be larger than the number of individuals [Liu and Zhang, 2009].

In the MLGL package, the representative variable summarizing each group output by group-Lasso is first computed by means of the first principal component. Then, the least squares estimators of the coefficients of each representative variable are computed. Finally, all p-values resulting from the test of the nullity of the estimated coefficients are corrected following Bonferroni's procedure [Dunn, 1959], which allows for controlling the FWER. This three-step procedure is described by Pseudo-code 2.

Pseudo-code 2 Reducing the number of groups

Input: Groups selected by group-Lasso for a given λ: $G_1^\lambda, \ldots, G_m^\lambda$

1- *Compute* the first principal component $\dot{X}_i$ of X_{G_i}, for all $i = 1, \ldots, m$.

2- From $\dot{X} = [\dot{X}_1, \ldots, \dot{X}_m]$ and the model $y = \dot{X}\dot{\beta} + \epsilon$,
compute $\hat{\dot{\beta}}$ the least-squares estimator of $\dot{\beta}$.

3- *Test* the nullity of the coefficients, *apply* the multiple testing correction to the corresponding p-values [Dunn, 1959], and *reject* all null hypotheses with an adjusted p-value lower than the prescribed level.

Output: The set of rejected null hypotheses.

2.3.2 Avoiding the redundancy of groups

As exposed in Section 2.2, the MLGL package allows for selecting groups from different levels of the hierarchy, which especially arises with small values of λ. It can therefore happen that one selected group is included in another one. It is then desirable to select only this group or its subgroup, but not both of them. This can be achieved by applying a hierarchical testing procedure (HTP) for controlling the FWER [Meinshausen, 2008].

The intuitive idea is to select the smallest possible groups of variables with a significant effect on the response variable. In particular this would avoid including a large group of variables with only a few of them being truly influential ones.

From a hierarchical tree (see Figure 2a), the importance of groups is tested sequentially with partial F-tests, which have been extensively used in the context of nested models in multiple linear regression problems [Jamshidian et al., 2007]. The importance of a group G of variables is tested with the following hypotheses:

$$H_{0,G} \ : \ \beta_G = 0, \qquad \text{versus} \qquad H_{1,G} \ : \ \exists i \in G, \ \beta_i \neq 0,$$

where β_i is the coefficient corresponding to the variable index $i \in G$, and $\beta_G = 0$ encodes that the group G has no influence on the response y.

HTP starts by testing the group containing all the variables at the top of the hierarchical tree. Then, for any rejected null hypothesis $H_{0,G}$, the null hypotheses associated with the children of group G (subgroups of G) are subsequently tested. The process is repeated until no more null hypothesis is rejected. Each computed p-value is adjusted following Bonferroni's procedure for controlling the FWER [Dunn, 1959].

2.3.3 The MLGL processing of the candidate groups

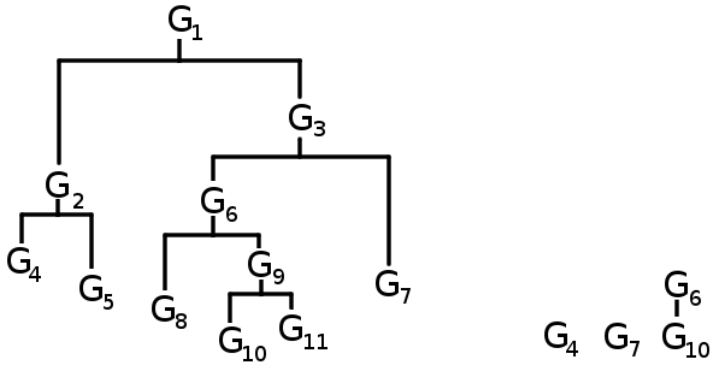
Let us consider the collection of candidate groups selected at the end of Section 2.2 for a given value of λ. At this stage, the MLGL package faces the two problems mentioned above that is, multiplicity and redundancy. This is the goal of the HMT procedure implemented in the MLGL package to overcome these problems.

More precisely the HMT procedure starts by splitting the selected groups into d disjoint hierarchical trees (denoted by $\mathcal{T}_i$, $i = 1, \ldots, d$) and one set $\mathcal{S}$ of candidate groups with no hierarchical structure (see Example 1).

Example 1 (Separate the selected groups in hierarchical trees)**.** *Let us consider a hierarchy built from 6 variables with groups as follows:* $G_1 = \{1, 2, 3, 4, 5, 6\}$, $G_2 = \{1, 2\}$, $G_3 = \{3, 4, 5, 6\}$, $G_4 = \{1\}$, $G_5 = \{2\}$, $G_6 = \{3, 4, 5\}$, $G_7 = \{6\}$, $G_8 = \{3\}$, $G_9 = \{4, 5\}$, $G_{10} = \{4\}$, $G_{11} = \{5\}$. *The resulting hierarchy is displayed in Figure 2a.*

For a specific value of λ, let us assume that the groups G_4, G_6, G_7, and G_{10} are selected (see Figure 2b).

Then the HMT procedure defines one set $\mathcal{S} = \{G_4, G_7\}$ and one hierarchical tree $\mathcal{T}_1 = \{G_6, G_{10}\}$, where $G_{10} \subset G_6$.



(a) Hierarchical tree from Example 1

(b) Selected groups from Example 1

Figure 2: Illustration from Example 1.

An important remark is that hierarchical trees must be *complete* that is, each group in the tree $\mathcal{T}_i$ is either a leaf (a group without any subgroups) or the union of its subgroups. This is a necessary requirement of our strategy since the importance of a candidate group G is tested through its leaves (subgroups of G without any children). If a group (which is not a leaf) is not the union of its children in the hierarchical tree, then the hierarchical testing procedure of [Meinshausen, 2008] cannot be properly applied. Therefore, some groups are added to the hierarchical tree for completing hierarchies which are not *complete* (see Example 2).

Example 2 (Complete a hierarchical tree)**.** *The groups $G_6 = \{3, 4, 5\}$ and $G_{10} = \{4\}$ from the hierarchical tree $\mathcal{T}_1$ in Example 1 do not form a complete hierarchy (G_6 is not equal to the union of its subgroups).*

The group $\bar{G}_{10} = \{3, 5\}$ is then defined as the complement of G_{10} within G_6, which leads to the new (full) hierarchical tree $\bar{\mathcal{T}}_1 = \{G_6, G_{10}, \bar{G}_{10}\}$.

The completed hierarchical trees are denoted by $\bar{\mathcal{T}}_1, \ldots, \bar{\mathcal{T}}_d$.

In addition, applying the HTP procedure from [Meinshausen, 2008] also requires to summarize each group within each hierarchical tree by a representative variable. This is done by the MLGL package by computing the first principal component of each group. The new corresponding trees are denoted by $\dot{\mathcal{T}}_1, \ldots, \dot{\mathcal{T}}_d$. Therefore the HTP procedure of [Meinshausen, 2008] is applied to $\dot{\mathcal{T}}_1, \ldots, \dot{\mathcal{T}}_d$ (see Pseudo-code 3).

Controlling the FWER level With the same notation as Section 2.3.3, let us define the cardinality of any hierarchical tree as the number of leaves it contains, and set $m = |\mathcal{S}| + \sum_{i=1}^{d} |\dot{\mathcal{T}}_i|$, where $|A|$ denotes the cardinality of the set A. Then, the HMT procedure implemented in the MLGL package controls the FWER of the tree $\dot{\mathcal{T}}_i$ (Pseudo-code 3) at level $\frac{\alpha|\dot{\mathcal{T}}_i|}{m}$, and that of the set $\mathcal{S}$ at level $\frac{\alpha|\mathcal{S}|}{m}$. It results that the global HMT procedure described by Pseudo-code 4 truly controls the FWER at the overall prescribed level $0 < \alpha < 1$.

Pseudo-code 3 Hierarchical testing procedure for one tree

Input: Any $\mathcal{T} \in \{\mathcal{T}_1, \ldots, \mathcal{T}_d\}$.

Complete hierarchical trees Add missing groups to the hierarchical tree $\mathcal{T}$ to get a *complete* tree $\bar{\mathcal{T}}$.

Summarize the influence of each group Compute the first principal component of each group in the tree $\bar{\mathcal{T}}$. The resulting hierarchical tree is denoted by $\dot{\mathcal{T}}$.

Hierarchical testing Apply the HTP procedure of [Meinshausen, 2008] to the tree $\dot{\mathcal{T}}$ for a prescribed level of control.

Output: Selected groups from $\dot{\mathcal{T}}$.

Pseudo-code 4 Hierarchical multiple testing (HMT) for a given regularization level

Input: List of groups selected after the group-Lasso step for a given $\lambda \in \Lambda$
(Λ: set of candidate regularization parameters).

Define hierarchical trees Split the groups into hierarchical trees $\mathcal{T}_1, \ldots, \mathcal{T}_d$ and the set $\mathcal{S}$. Set $m = |\mathcal{T}_1| + \cdots + |\mathcal{T}_j| + |\mathcal{S}|$.

Testing procedure for hierarchical trees For each hierarchical tree $\mathcal{T}_i$ for $i = 1, \ldots, d$, apply Pseudo-code 3 to get the global control level $\frac{\alpha \times |\mathcal{T}_i|}{m}$.

Testing procedure for groups not belonging to a tree For the set $\mathcal{S}$, apply Pseudo-code 2 to get the global control level $\frac{\alpha \times |\mathcal{S}|}{m}$.

Avoiding over-fitting In order to avoid overfitting, it is necessary to use different individuals for using group Lasso and applying the hierarchical testing procedure.

The set $\mathcal{I} = \{1, \ldots, n\}$ of indices associated with individuals is randomly split into two parts of equal size, say $\mathcal{I}_1$ and $\mathcal{I}_2$. The hierarchical clustering of the variables is first performed from the set $\mathcal{I}_1$. Then group-Lasso is applied from the individuals in $\mathcal{I}_2$ and the previously computed hierarchy. Finally, the whole HMT procedure (namely Pseudo-code 4) is applied for the individuals from $\mathcal{I}_1$. In order to ease the understanding, the whole procedure consisting of "AHC+gLasso+HMT" is summarized in Psuedo-code 5.

Pseudo-code 5 AHC+gLasso+HMT

1. Randomly split the sample indexed by I into two subsets of equal cardinality: $\mathcal{I}_1$ and $\mathcal{I}_2$.
2. Perform AHC of candidate variables from $\mathcal{I}_1$.
3. Perform group-Lasso (9) from $\mathcal{I}_2$.
4. Apply the HMT procedure (namely Pseudo-code 4) from $\mathcal{I}_1$.

2.4 Selecting the final groups by choosing λ

The groups output at the previous steps of the MLGL package (AHC+gLasso+HMT) depend on the value of the regularization parameter $\lambda \in \Lambda$, which is a crucial choice. Several papers have raised the problem of choosing the value of λ in penalized regression frameworks

[Fan and Tang, 2013, Sun et al., 2013]. For instance, resampling-based approaches have been suggested. Among them, choosing the value of λ which yields the most stable selected variables have been explored by [Meinshausen and Bühlmann, 2010], which intensively relies on bootstrap. An alternative consists in tuning λ by means of V-fold cross validation [Arlot and Celisse, 2010]. However both these approaches are highly time-consuming due to the multiple executions they require. Moreover V-fold cross-validation is more suited to the estimation/prediction purpose than to the identification/selection of influential variables. This aspect arises more clearly in difficult settings where the signal-to-noise ratio becomes small. Then, V-fold cross-validation tends to include superfluous variables (false positives). Furthermore information criteria such as AIC [Akaike, 1974] and BIC [Schwarz, 1978] need an estimator of both the degrees of freedom and the unknown variance σ^2 [Giraud et al., 2007]. However if the number of candidate variables is larger than the number of observations, such a consistent estimator of σ^2 is difficult to design [Fan et al., 2012].

One important feature of the procedures implemented in the MLGL package is that the FWER is kept under control whatever the value of $\lambda \in \Lambda$. Furthermore since the proposed procedure turns out to be conservative (from our empirical experiments), the MLGL package chooses the value of λ maximizing the number of rejections. The simulation results discussed in Section 4 seem to support this choice since maximizing the number of rejections turns out to maximize in the same time the number of true positives (while keeping the number of false positives under control).

3 Usage of the MLGL package

The main function of the MLGL package is `fullProcess`. It enables to run the whole procedure consisting in AHC+gLasso+HMT.

For illustration purpose, we generate simulated data with the function `simuBlockGaussian`. In what follows, $n = 50$ individuals and $p = 60$ candidate variables are simulated from a multivariate Gaussian $\mathcal{N}(0,\Sigma)$ distribution. The covariance matrix Σ has a block-diagonal structure where each block of 5 variables has 1 on the diagonal and $\rho = 0.7$ elsewhere, that is

```
X <- simuBlockGaussian(n= 50, nBlock=12,sizeBlock= 5, rho= 0.7)
```

Two probabilistic models are considered in the MLGL package: the linear and the logistic ones.

- With the linear model, let us simulate

  ```
  y <- drop(X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50, 0, 0.5))
  ```

  Then, applying the function `fullProcess` is done by means of:

  ```
  res <- fullProcess(X, y)
  ```

- With the logistic model, binary observations are generated by

  ```
  y <- 2*(rowSums(X[,1:4])>0)-1
  ```

  Then, the function `fullProcess` can be processed by:

  ```
  res <- fullProcess(X, y, loss = "logit", test = partialChisqtest)
  ```

In addition to this main function, the MLGL package contains functions enabling to perform different steps of the procedure. For instance, the `MLGL` function computes the path of candidate groups output after AHC+gLasso.

Alternative procedures to HMT are also implemented in the MLGL package to select final groups. For instance, `cv.MLGL` and `stability.MLGL` can be applied to choose λ by respectively V-fold cross-validation and bootstrap. More precisely, the first one returns the mean cross-validation error (mean squared error for the linear case or area under the ROC curve for the logistic case) for a prescribed sequence of regularization parameter values. Instead, the second one performs the stability selection procedure [Meinshausen and Bühlmann, 2010] where the probability of selecting each group is estimated for every value of the prescribed sequence of regularization parameter values. Let us also mention that the paths returned by these two functions can be independently generated by the functions `plot.MLGL`, `plot.cv.MLGL`, and `plot.stability.MLGL` (see Figure 3):

```
res <- MLGL(X, y)
plot(res)
res.cv <- cv.MLGL(X, y, loss = "logit")
plot(res.cv)
res.stab <- stability.MLGL(X, y, loss = "logit")
plot(res.stab)
```

4 Comparison of MLGL to other selection procedures

In the present section, the solution paths output by different procedures will be compared to that one provided by the MLGL package by plotting the number of true positives versus the number of false positives.

Let us generate n realizations of independent and identically distributed random variables $X_1, \ldots, X_n \in \mathbb{R}^p$ from a multivariate Gaussian distribution $\mathcal{N}(0_p, \Sigma)$, where Σ is a $p \times p$ covariance matrix with a block-diagonal structure. The common size of the blocks is l, and all the blocks have 1 on their diagonal and ρ everywhere else.

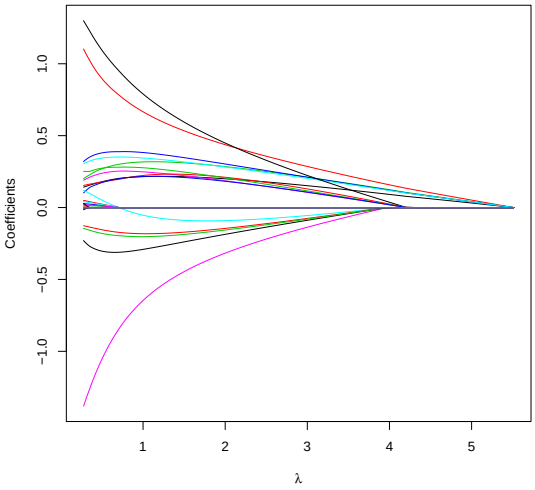
The response variable is generated from the model $y = X\beta^* + \epsilon$, where $\beta^* \in \mathbb{R}^p$ is a sparse vector with 1s for K elements corresponding to different blocks of Σ, and ϵ denotes a random Gaussian variable. Note that the noise level is set such that the signal-to-noise ratio has a value of 2.

In the present simulation design, a selected group is called *true positive* if it contains exactly one variable belonging to the support of the true solution β^*, as well as other variables that are correlated with this one but do not belong to the support of β^*. Conversely a group is termed as a *false positive* if it contains either no variable belonging to the support of β^*, or several (uncorrelated) variables belonging to the true support.

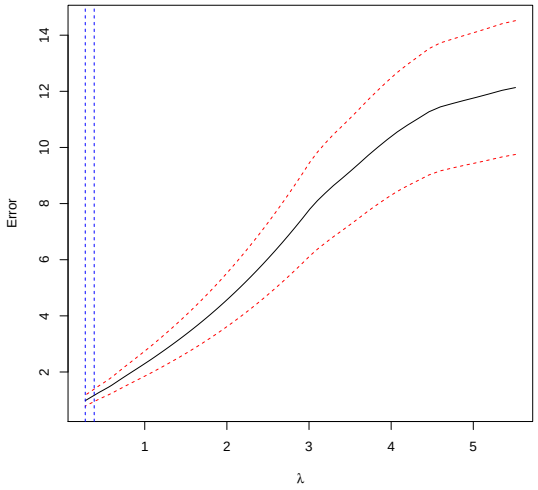
4.1 Comparison of *Multi-Layer Group-Lasso* with group-Lasso

The output of the MLGL package is first compared to that of the classical group-Lasso which essentially focuses on only one level of the hierarchy.

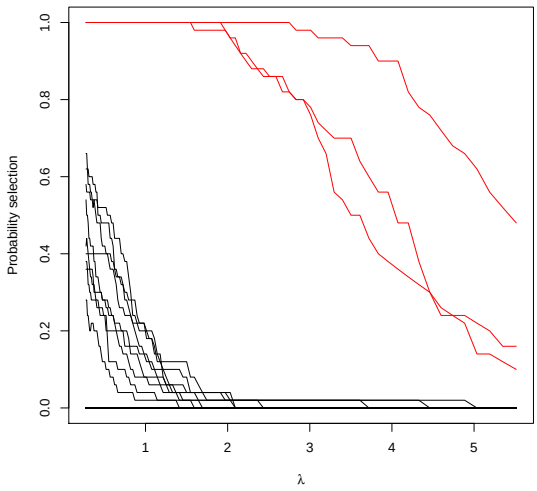
The AHC step is performed based on the Euclidean distance and Ward's criterion. The highest jump rule selects the partition of the candidate variables (level of the hierarchy) that is taken as an input of the classical group-Lasso. The MLGL package uses the weights defined in Eq. (8), which (also) involves the highest jump rule and allows for selecting groups from different levels of the hierarchy.



(a) Solution path (`plot.MLGL`)

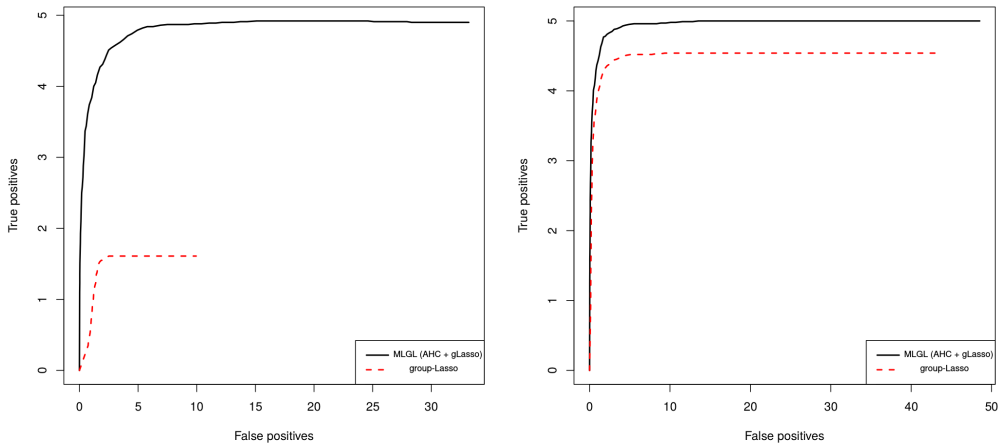


(b) CV error (`plot.cv.MLGL`)



(c) Probability selection (`plot.stability.MLGL`)

Figure 3: Plots generated by `plot.MLGL`, `plot.cv.MLGL` and `plot.stability.MLGL`. The plot generated by `plot.MLGL` represents the solution path of MLGL with each curve corresponding to the estimated coefficients of a variable according to the regularization parameter. The cross-validation error is the output of `plot.cv.MLGL`; the vertical lines correspond to the λ which minimizes the cross-validation error and the largest value of λ such that error is within one standard error of the minimum. `plot.stability.MLGL` shows the probability selection for the different groups, the red curves being the selected groups.



(a) $n = 100$, $p = 500$, $\rho = 0.5$, $l = 5$, $K = 5$ (b) $n = 100$, $p = 500$, $\rho = 0.7$, $l = 5$, $K = 5$

Figure 4: Number of true positives versus the number of false positives in the solution path output by the MLGL package before hierarchical multiple testing (black solid line) and classical group-Lasso (red dashed line). The curves represent the mean calculated over 100 replicates.

Figure 4 displays the number of true and false positives along the solution path output by the MLGL package and the classical group-Lasso. For a given number of false positives, more true positives are provided by the two first steps of the MLGL package (AHC+gLasso) than by the classical group-Lasso.

The gap between the two solution paths can be explained by the way the partition used by the group-Lasso is chosen. From Figure 5 (left panel), it arises that the highest jump rule fails to recover the optimal partition which has 100 groups in the present simulation experiments. In such cases, group-Lasso selects groups among poor candidates whereas the MLGL package is less sensitive to such a bad preliminary choice.

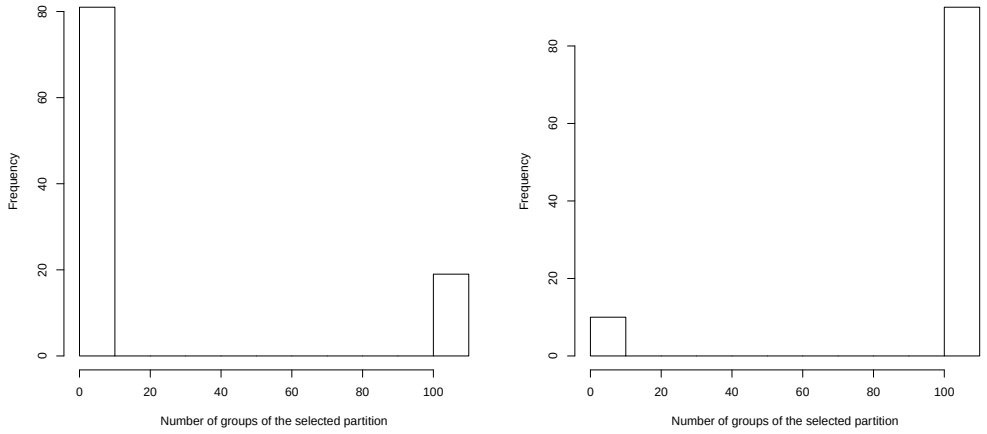
4.2 Comparison to alternative approaches combining clustering and selection

The performance of the MLGL package is now compared to that of alternative procedures combining clustering and selection: Hierarchical Clustering and Averaging for Regression (HCAR) [Park et al., 2007], Supervised Group-Lasso (SGL) [Ma et al., 2007], Cluster Representative Lasso (CRL) and Cluster Group-Lasso (CGL) [Bühlmann et al., 2013]. Note that all these procedures combine a clustering step (hierarchical clustering or k-means) with a selection step (Lasso, group-Lasso, or standardized group-Lasso [Bühlmann and van de Geer, 2011, Simon and Tibshirani, 2011]).

For all these methods a clustering is performed based on the Euclidean distance and Ward's criterion. When the method requires only one partition, this one is chosen by the highest jump rule. For HCAR, $\hat{\lambda}$ is chosen by cross-validation and only the corresponding solution path is output.

Figure 6 displays the number of true and false positives along the solution path of the competing procedures for different values of the parameters.

The MLGL package turns out to provide results among the best ones since the maximal num-



(a) $n = 100$, $p = 500$, $\rho = 0.5$, $l = 5$, $K = 5$ (b) $n = 100$, $p = 500$, $\rho = 0.7$, $l = 5$, $K = 5$

Figure 5: Size (in number of groups) of the partition selected by the highest jump rule.

ber of true positives ($K = 5$ or 10) is reached with only a few false positives. It is noticeable that *Cluster Representative Lasso* and *Supervised Group-Lasso* exhibit similar performances (schemes b, c and d).

When the correlation ρ rises from 0.5 to 0.9 between Figures 6a and 6b, the performance of *HCAR* and *CGL* heavily deteriorates whereas the other procedures remains almost unchanged.

Between Figures 6b and 6c, the number of variables in the support of the true response increases from 5 to 10. The MLGL package still provides among the best results. But more selected groups turn out to be false positives when reaching the maximal number of true positives.

When the size of the diagonal-blocks is decreased from 10 to 5 between Figures 6b and 6d, all procedures perform similarly (even if the correlation is set at 0.9). It seems that dealing with large blocks with highly correlated variables is a difficult settings for *HCAR* and *CGL*.

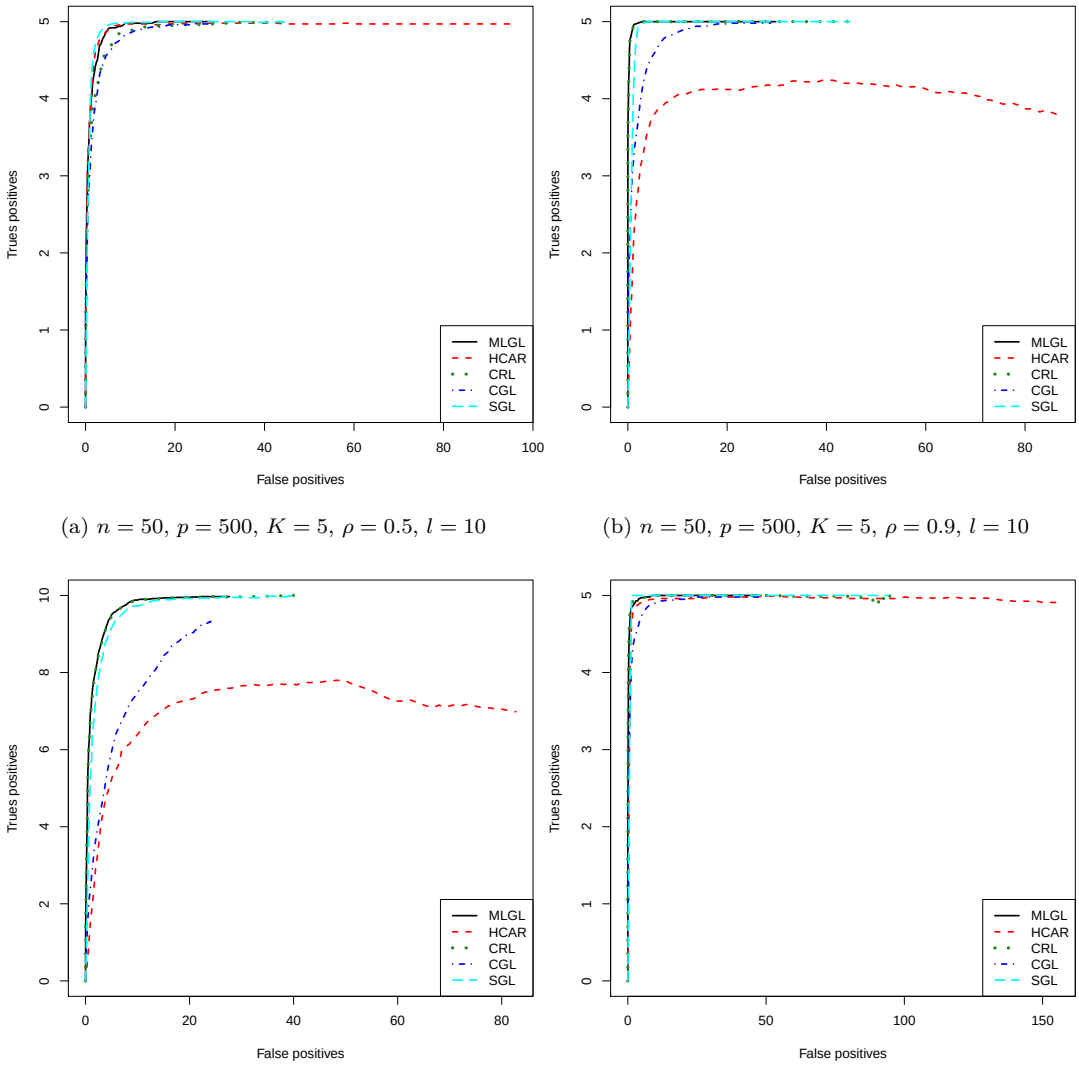
The procedure implemented in the MLGL package seems to have better results when the size of blocks is increased and the correlation strength is greater, which has the effect of reducing the effective dimension of the problem.

4.3 Hierarchical multiple testing procedure

Let us now assess the quality of the solution path before and after applying the HMT procedure. Figure 7 shows the number of true and false positives among the groups output by AHC+gLasso before and after applying the HMT procedure.

One striking aspect of these experimental results is that the set of groups output by AHC+gLasso contains more false than true positives for small values of λ. But the two curves quickly cross each other as λ grows. This strengthens the need for a multiple testing procedure discarding false groups. It is also noticeable that the number of false positives immediately drops after using the HMT procedure, no matter the level α at which the multiple testing correction is applied.

With only $K = 5$ true groups, most of the true positives are kept after applying HMT, unlike what happens when the number of true groups is $K = 10$ (Figure 7c). However in presence of highly correlated variables (within groups), the performance of the MLGL package strongly



(a) $n = 50$, $p = 500$, $K = 5$, $\rho = 0.5$, $l = 10$

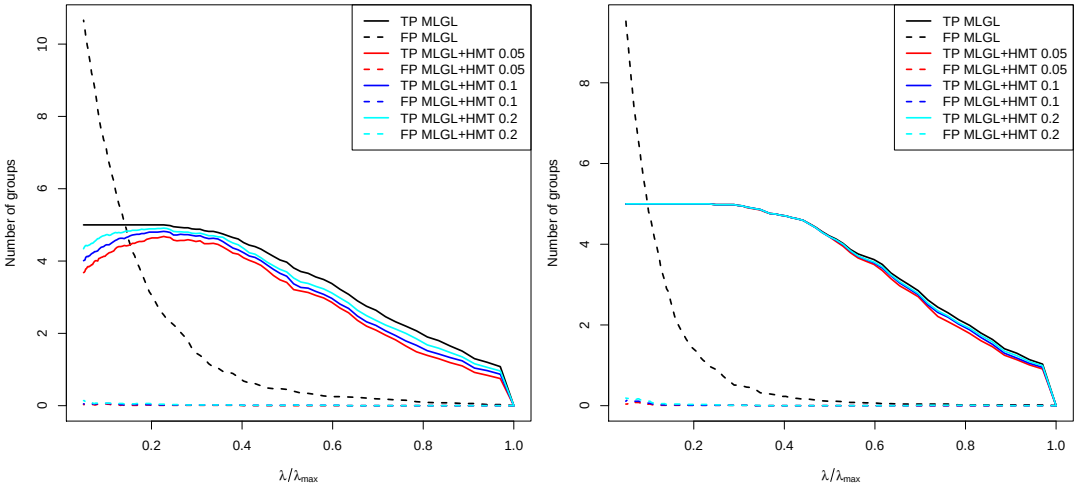
(b) $n = 50$, $p = 500$, $K = 5$, $\rho = 0.9$, $l = 10$

(c) $n = 50$, $p = 500$, $K = 10$, $\rho = 0.9$, $l = 10$

(d) $n = 50$, $p = 500$, $K = 5$, $\rho = 0.9$, $l = 5$

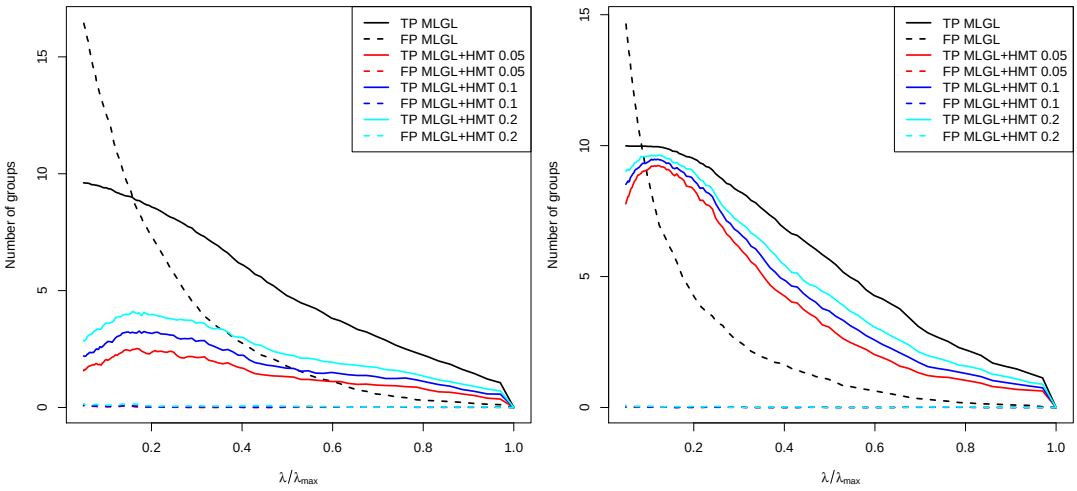
Figure 6: Number of true positives versus the number of false positives along the solution path of *Multi-Layer Group-Lasso* before hierarchical multiple testing (MLGL, black), *Hierarchical Clustering and Averaging for Regression* (HCAR, red), *Cluster Representative Lasso* (CRL, green), Cluster Group-Lasso (CGL, blue) and *Supervised Group-Lasso* (SGL, cyan). Each curve represents the average of 100 trials. Between the Figure 6a and 6b, the correlation ρ rises from 0.5 to 0.9. Between the Figures 6b and 6c, the number of true groups K rises from 5 to 10. Between the Figures 6b and 6d, the size l of blocks reduces from 10 to 5.

improves (Figure 7d) since on average, more than 9 (out of 10) true positives can be recovered at best. By contrast when the correlation decreases, the performance sharply drops (Figure 7c). In this situation, the maximum number of true positives is rather small (only 4 out of 10 when $\alpha = 0.20$).



(a) $n = 50$, $p = 500$, $K = 5$, $\rho = 0.7$, $l = 10$

(b) $n = 50$, $p = 500$, $K = 5$, $\rho = 0.9$, $l = 10$



(c) $n = 50$, $p = 500$, $K = 10$, $\rho = 0.7$, $l = 10$

(d) $n = 50$, $p = 500$, $K = 10$, $\rho = 0.9$, $l = 5$

Figure 7: Number of true and false positives along the solution path of *Multi-Layer Group-Lasso* before (MLGL, black) and after applying the hierarchical multiple testing procedure (MLGL + HMT) with $\alpha \in \{0.05, 0.1, 0.2\}$. In these figures, *MLGL* stands for *ACH + gLasso*. Each curve represents the average of 100 trials. The upper figures show the case $K = 5$ whereas the bottom figures show the case $K = 10$. From left to right, the correlation increases from 0.7 to 0.9.

Table 1: Number of true (TP) and false positives (FP) for different values of regularization parameters for $n = 100$ and $p = 500$. $\hat{\lambda}_{\mathrm{RM}}$ (resp. $\hat{\lambda}_{\mathrm{TPM}}$) denotes the value maximizing the number of rejections (resp. true positives). K, l et ρ are the different parameters of the simulated data. K is the size of the support of β^*, l the size of blocks and ρ the within-block correlation. In the HMT procedure, $\alpha = 0.05$.

		$K = 5$				$K = 10$			
		$l = 5$		$l = 10$		$l = 5$		$l = 10$	
		TP	FP	TP	FP	TP	FP	TP	FP
$\rho = 0.9$	ACH + gLasso + HMT + $\hat{\lambda}_{\mathrm{RM}}$	4.91	0.28	4.78	0.8	4.15	0.44	4.75	1.28
	ACH + gLasso + HMT + $\hat{\lambda}_{\mathrm{TPM}}$	4.95	0	4.86	0.05	4.27	0.14	4.86	0.57
$\rho = 0.7$	ACH + gLasso + HMT + $\hat{\lambda}_{\mathrm{RM}}$	2.95	0.51	3.95	0.27	1.59	0.64	3.39	0.54
	ACH + gLasso + HMT + $\hat{\lambda}_{\mathrm{TPM}}$	3.02	0.13	3.97	0.14	1.65	0.23	3.4	0.39
$\rho = 0.5$	ACH + gLasso + HMT + $\hat{\lambda}_{\mathrm{RM}}$	2.89	0.32	2.6	0.53	1.68	0.41	1.53	0.63
	ACH + gLasso + HMT + $\hat{\lambda}_{\mathrm{TPM}}$	2.95	0.04	2.7	0.13	1.72	0.13	1.58	0.26

From the different pictures of Figure 7, the overall conclusion owing to the calibration of λ is that choosing the value of λ maximizing the number fo rejections provides the best results in terms of the ratio between true and false positives. This clearly arises from the remark that the number of false positives is almost constant in our experimental results compared to the strong variations in the true positives curve. However this should be clear that this is likely to be a by-product of the high conservativeness of the HMT procedure implemented in the MLGL package.

4.4 Tuning the parameter λ

Let us now illustrate the performance of the procedure implemented in the MLGL package which yields the final selected groups.

Maximizing the number of rejections. Based on the previous remarks made in Section 4.3, the default value of λ recommended in the MLGL package is the one maximizing the number of rejections, which is denoted by $\hat{\lambda}_{\mathrm{RM}}$) in what follows.

However it should be clear that the number of rejections can include some false positives, which would be suboptimal. Therefore, an *oracle* choice for the parameter λ is the one maximizing the number of true rejections, called $\hat{\lambda}_{\mathrm{TPM}}$. Since the number of false positives in our simulation experiments only slowly increases, this choice should provide the best possible performance in terms of the ratio between true and false positives. All of this is illustrated by Table 1, which collects the results obtained with $\alpha = 0.05$. From Table 1, the main idea is that choosing $\lambda = \hat{\lambda}_{\mathrm{RM}}$ as the value maximizing the number of rejections is almost optimal since, whatever the experimental conditions, both the numbers of true and false rejections remain close to the ones of the oracle rule $\hat{\lambda}_{\mathrm{TPM}}$.

There is a drop of the number of true positives (both for $\hat{\lambda}_{\mathrm{RM}}$ and $\hat{\lambda}_{\mathrm{TPM}}$) as the number K of true groups increases from 5 to 10. This phenomenon is somewhat balanced by the increase of the correlation level (at least when $\rho = 0.9$) since in this case, we keep almost the same results.

Another interesting idea is that increasing the size l of the blocks in presence of a strong

Table 2: Comparison of different methods of choice of the regularization parameter. Stability selection is used with a threshold of 0.75. TP and FP correspond to true positives and false positives. K, l et ρ are the different parameters of the simulated data. K is the size of the support of β^*, l the size of blocks and ρ the within-block correlation.

		$K=5$				$K=10$			
		$l=5$		$l=10$		$l=5$		$l=10$	
		TP	FP	TP	FP	TP	FP	TP	FP
$\rho=0.9$	proposed method	4.91	0.28	4.78	0.8	4.15	0.44	4.75	1.28
	Kappa	3.66	2.14	4.3	2.64	5.78	13.25	5.84	8.06
	5-f cv	4.96	24.37	4.87	23.4	8.15	30.46	7.74	27.21
	stability	4.99	0.15	5	0.4	7.4	0.22	9.92	0.22
$\rho=0.7$	proposed method	2.95	0.51	3.95	0.27	1.59	0.64	3.39	0.54
	Kappa	2.59	1.68	3.86	1.21	2.76	4.36	6.22	3.29
	5-f cv	3.73	6.32	4.36	5.33	3.35	6.35	6.46	4.55
	stability	4.52	0.61	5	1.79	3.63	0.84	9.8	1.58
$\rho=0.5$	proposed method	2.89	0.32	2.6	0.53	1.68	0.41	1.53	0.63
	Kappa	3.19	3.83	3.08	3.32	2.93	5.50	3.56	5.34
	5-f cv	3.55	6.73	3.49	6.28	3.34	7.67	3.72	5.71
	stability	3.17	1.17	4.85	1.61	2.54	1.79	8.01	1.51

enough correlation level improves the results. For instance if $\rho = 0.5$, increasing l from 5 to 10 reduces the number of groups, but does not improve the results. By contrast, as long as $\rho \geq 0.7$, enlarging the blocks reduces the effective dimension of the problem, which leads to better results.

Performance of HMT+$\hat{\lambda}_{RM}$. An important question is to determine the influence of the procedure HMT+$\hat{\lambda}_{RM}$ on the quality of the final selected groups. To address this question, a comparison is carried out between the selection procedure of λ implemented in the MLGL package and alternative ones such as 5-fold cross-validation, kappa [Sun et al., 2013], and stability selection [Meinshausen and Bühlmann, 2010]. Let us emphasize that 5-fold cross-validation aims at selecting a $\hat{\lambda}$ which minimizes the prediction error, whereas Kappa and stability selection mainly focus on selecting groups with the highest possible stability. However all these procedures are time-consuming since they require multiple executions of the whole procedure. Table 2 collects the experimental results.

Firstly, 5-fold cross-validation uniformly selects more true positives, but at the price of including by far more false positives than any other competitor. This is in line with the trend of cross-validation to favor estimation/prediction rather than identification/selection.

Secondly, the best overall performance is achieved by the stability selection which always provides the largest number of true positives and only a small (averaged) number of false positives. This remarkable conclusion has to be balanced with the higher computational cost suffered by this time-consuming procedure.

Thirdly, the procedure implemented in the MLGL package yields close (but somewhat smaller) numbers of true positives compared to stability selection. However, the number of false positives is almost equal to (or lower than) the one of stability selection, which results from the conservativeness of our HMT procedure.

Finally the Kappa selection procedure performance stays in between that of 5-fold cross-

validation and the one of the MLGL package, for a higher computational price.

In conclusion, choosing the regularization parameter as the one maximizing the number of rejections gives reliable results which remain close to optimal ones according to our simulation experiments. The procedure implemented in the MLGL package seems conservative. But it does not require any intensive re-sampling and selects only a few false positives.

5 Conclusions

We designed a selection procedure implemented in the MLGL package, MLGL standing for *Multi-Layer Group-Lasso*. This procedure aims at selecting groups of correlated variables according to a response variable. It combines hierarchical clustering and group-Lasso. It differs from classical group-Lasso-based strategies by allowing to use simultaneously different levels of the hierarchy provided by the hierarchical clustering step. A weight for each level of the hierarchy is introduced to favor a priori "good" levels (according to a quality measure). From our empirical experiments, it results that the MLGL package performs almost the same as or improves upon alternative procedures combining hierarchical clustering and group-Lasso.

Possible improvements of the procedure in the MLGL package could be made, for instance by optimizing the weight function used at the group-Lasso step. Developing a more flexible weight function or using the results of several hierarchical clustering distances are interesting lines of research to explore.

In the MLGL package, the optimal value of the regularization parameter is chosen by maximizing the number of rejections. This results from the conservativeness of the involved HMT procedure. This HMT procedure has nevertheless the merit of taking into account the possible hierarchical trees and provides a FWER control of the selected groups. A way to improve the results is to provide tighter bounds on the FWER control to get a refined p-value correction. Nevertheless, the main advantage of the MLGL package over alternative approaches is that it provides close to optimal results while requiring a by far smaller computation time.

Acknowledgements

We thank Direction Générale de l'Armement (DGA) and Inria for a financial support of Quentin Grimonprez's PhD, and the CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020.

References

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

[Arlot and Celisse, 2010] Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79.

[Barber et al., 2015] Barber, R. F., Candès, E. J., et al. (2015). Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, 43(5):2055–2085.

[Benjamini and Hochberg, 1995] Benjamini, Y. and Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300.

[Bühlmann et al., 2013] Bühlmann, P., Rütimann, P., van de Geer, S., and Zhang, C.-H. (2013). Correlated variables in regression: clustering and sparse estimation. *Journal of Statistical Planning and Inference*, (143):1835–3871.

[Bühlmann and van de Geer, 2011] Bühlmann, P. and van de Geer, S. (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Publishing Company, Incorporated.

[Dunn, 1959] Dunn, O. J. (1959). Estimation of the medians for dependent variables. *Ann. Math. Statist.*, 30(1):192–197.

[Fan et al., 2012] Fan, J., Guo, S., and Hao, N. (2012). Variance estimation using refitted cross-validation in ultrahigh dimensional regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(1):37–65.

[Fan and Tang, 2013] Fan, Y. and Tang, C. Y. (2013). Tuning parameter selection in high dimensional penalized likelihood. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):531–552.

[Giraud et al., 2007] Giraud, C., Baraud, Y., and Huet, S. (2007). Gaussian Model Selection with Unknown Variance.

[Jacob et al., 2009] Jacob, L., Obozinski, G., and Vert, J.-P. (2009). Group Lasso with Overlap and Graph Lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 433–440, New York, NY, USA. ACM.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data Clustering: A Review. *ACM Comput. Surv.*, 31(3):264–323.

[Jamshidian et al., 2007] Jamshidian, M., Jennrich, R. I., and Liu, W. (2007). A study of partial f tests for multiple linear regression models. *Comput. Stat. Data Anal.*, 51(12):6269–6284.

[Jenatton et al., 2011] Jenatton, R., Audibert, J.-Y., and Bach, F. (2011). Structured variable selection with sparsity-inducing norms. *J. Mach. Learn. Res.*, 12:2777–2824.

[Liu and Zhang, 2009] Liu, H. and Zhang, J. (2009). Estimation consistency of the group lasso and its applications. In *JMLR*.

[Ma et al., 2007] Ma, S., Song, X., and Huang, J. (2007). Supervised group lasso with applications to microarray data analysis. *BMC Bioinformatics*, 8(1):60.

[Meinshausen, 2008] Meinshausen, N. (2008). Hierarchical testing of variable importance. *Biometrika*, 95(2):265–278.

[Meinshausen and Bühlmann, 2010] Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473.

[Park et al., 2007] Park, M. Y., Hastie, T., and Tibshirani, R. (2007). Averaged gene expressions for regression. *Biostatistics*, 8(2):212–227.

[Schwarz, 1978] Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

[Simon and Tibshirani, 2011] Simon, N. and Tibshirani, R. (2011). Standardization and the group lasso penalty. Technical report.

[Sun et al., 2013] Sun, W., Wang, J., and Fang, Y. (2013). Consistent Selection of Tuning Parameters via Variable Selection Stability. 14:3419–3440.

[Tibshirani, 1994] Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

[Tibshirani et al., 2005] Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, pages 91–108.

[Wainwright, 2009] Wainwright, M. J. (2009). Sharp Thresholds for High-dimensional and Noisy Sparsity Recovery Using L1-constrained Quadratic Programming (Lasso). *IEEE Trans. Inf. Theor.*, 55(5):2183–2202.

[Wasserman and Roeder, 2009] Wasserman, L. and Roeder, K. (2009). High-dimensional variable selection. *Ann. Statist.*, 37(5A):2178–2201.

[Witten et al., 2014] Witten, D. M., Shojaie, A., and Zhang, F. (2014). The cluster elastic net for high-dimensional regression with unknown variable grouping. *Technometrics*, 56(1):112–122.

[Yang and Zou, 2015] Yang, Y. and Zou, H. (2015). A fast unified algorithm for solving group-lasso penalized learning problems. *Statistics and Computing*, 25(6):1129–1141.

[Yuan and Lin, 2006] Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67.

[Zhao and Yu, 2006] Zhao, P. and Yu, B. (2006). On Model Selection Consistency of Lasso. *J. Mach. Learn. Res.*, 7:2541–2563.

Appendix

A Proof of Lemma 1

Let β denote a solution of the group-Lasso (equation (6)) for a value of λ, then β must check $\forall i = 1, \ldots, g$:

$$X_{G_i}^T(y - X\beta) = \lambda w_i s_{G_i} \tag{10}$$

with s_{G_i} belonging to subdifferential of the function $\|\,.\,\|_2$ at θ_{G_i},

$$s_{G_i} \in \begin{cases} \left\{\frac{\beta_{G_i}}{\|\beta_{G_i}\|_2}\right\} & \text{if } \beta_{G_i} \neq 0_{|G_i|} \\ \left\{z \in \mathbb{R}^{|G_i|} \;\middle|\; \|z\|_2 \leq 1\right\} & \text{if } \beta_{G_i} = 0_{|G_i|} \end{cases}$$

The subdifferential of a function $f : U \to \mathbb{R}$ with U a convex subset of $\mathbb{R}^p$ contains the subgradients of f. A vector $v \in U$ is a subgradient of f at x_0 if $\forall x \in U : f(x) - f(x_0) \geq \langle v, x - x_0 \rangle$.

From Karush-Kuhn-Tucker (KKT) conditions, we can deduce that if $\|X_{G_i}^T(y - X\theta)\|_2 < \lambda w_i$ then $\theta_{G_i} = 0_{|G_i|}$.

Proof 1 (Lemma 1)**.** *Suppose that $G_1 = G_2$ and $w_2 > w_1 > 0$. Let θ denote a solution of group-Lasso (equation (6)). We want to show that we have $\theta_{G_2} = 0_{|G_2|}$.*

- *Let $\theta_{G_1} = 0_{|G_1|}$. We show that $\theta_{G_2} = 0_{|G_2|}$.*
  If $\theta_{G_1} = 0_{|G_1|}$, from KKT conditions, we have:

$$\begin{aligned} &\|X_{G_1}^T(y - X\theta)\|_2 \leq \lambda w_1 \\ &\|X_{G_2}^T(y - X\theta)\|_2 \leq \lambda w_1 \text{ because } X_{G_1} = X_{G_2} \\ &\|X_{G_2}^T(y - X\theta)\|_2 < \lambda w_2 \text{ because } w_1 < w_2 \end{aligned}$$

  So, $\theta_{G_2} = 0_{|G_2|}$.

- *If $\theta_{G_1} \neq 0_{|G_1|}$. We show that $\theta_{G_2} = 0_{|G_2|}$.*
  If $\theta_{G_1} \neq 0_{|G_1|}$, from KKT conditions, we have:

$$\begin{aligned} X_{G_1}^T(y - X\theta) &= \lambda w_1 \frac{\theta_{G_1}}{\|\theta_{G_1}\|_2} \\ \|X_{G_1}^T(y - X\theta)\|_2 &= \left\|\lambda w_1 \frac{\theta_{G_1}}{\|\theta_{G_1}\|_2}\right\|_2 \\ \|X_{G_1}^T(y - X\theta)\|_2 &= \lambda w_1 \\ \|X_{G_2}^T(y - X\theta)\|_2 &= \lambda w_1 \text{ because } X_{G_1} = X_{G_2} \\ \|X_{G_2}^T(y - X\theta)\|_2 &< \lambda w_2 \text{ because } w_1 < w_2 \end{aligned}$$

  So, $\theta_{G_2} = 0_{|G_2|}$.
  We have shown that $\theta_{G_2} = 0_{|G_2|}$, the lemma is proved.