Received: 10 March 2023 | Revised: 6 April 2023 | Accepted: 13 April 2023

DOI: 10.1002/sta4.578

ORIGINAL ARTICLE

WILEY

Variable selection with LASSO regression for complex survey data

Amaia Iparragirre[1] | Thomas Lumley[2] | Irantzu Barrio[1,3] | Inmaculada Arostegui[1,3]

[1]Department of Mathematics, University of the Basque Country UPV/EHU, Leioa, 48940, Spain

[2]Department of Statistics, University of Auckland, Auckland, 1142, New Zealand

[3]BCAM - Basque Center for Applied Mathematics, Bilbao, 48009, Spain

Correspondence
Amaia Iparragirre, Department of Mathematics, Faculty of Science and Technology, Barrio Sarriena s/n, 48940 Leioa, Bizkaia, Basque Country, Spain.
Email: amaia.iparragirre@ehu.eus

Funding information
Agencia Estatal de Investigación, Grant/Award Number: PID2020-115882RB-I00; Ministerio de Ciencia e Innovación, Grant/Award Number: CEX2021-001142-S; Departamento de Educación, Política Lingüística y Cultura del Gobierno Vasco, Grant/Award Number: IT1456-22; Basque Government; University of the Basque Country

Variable selection is an important step to end up with good prediction models. LASSO regression models are one of the most commonly used methods for this purpose, for which cross-validation is the most widely applied validation technique to choose the tuning parameter (λ). Validation techniques in a complex survey framework are closely related to "replicate weights". However, to our knowledge, they have never been used in a LASSO regression context. Applying LASSO regression models to complex survey data could be challenging. The goal of this paper is twofold. On the one hand, we analyze the performance of replicate weights methods to select the tuning parameter for fitting LASSO regression models to complex survey data. On the other hand, we propose new replicate weights methods for the same purpose. In particular, we propose a new design-based cross-validation method as a combination of the traditional cross-validation and replicate weights. The performance of all these methods has been analyzed and compared by means of an extensive simulation study to the traditional cross-validation technique to select the tuning parameter for LASSO regression models. The results suggest a considerable improvement when the new proposal design-based cross-validation is used instead of the traditional cross-validation.

KEYWORDS
complex survey data, cross-validation, LASSO regression, replicate weights, variable selection

1 | INTRODUCTION

Complex survey data are becoming increasingly relevant in a number of fields, including social and health sciences, among others. In this framework, the finite population of interest for the study is usually sampled following a complex sampling design, which may include techniques such as stratification, clustering, or a combination of them in different stages of the sampling scheme. In this context, a sampling weight is assigned to each sampled unit, indicating the number of units that this observation represents in the finite population. Suppose, for example, we consider a stratified sampling scheme, in which several strata are defined in the finite population and a number of units or a number of clusters of units (which will be denoted as primary sampling units [PSU], hereinafter) is sampled from each stratum. In this case, the sampling weight for each sampled unit is usually defined as its inverse inclusion probability, that is, the probability for each unit being sampled. Due to these particularities, complex survey data do not satisfy the independence and identically distributed conditions, and hence, the validity of traditional statistical techniques should be checked before applying them to data collected from complex surveys (see, e.g., Skinner et al., 1989, for more information on

this topic). Among other issues, the need for and use of sampling weights and the design effect on the development of prediction models has been widely discussed in the literature (see, e.g., Iparragirre et al., 2023; Lumley & Scott, 2015, 2017; Pfeffermann & Sverchkov, 2009; Smith, 1988, as a summary of this debate that is still alive).

In this paper, we focus on the development of prediction models and, more specifically, on the variable selection for complex survey data in linear and logistic regression frameworks. Least Absolute Shrinkage and Selection Operator (LASSO) regression (Tibshirani, 1996) is nowadays a widely used technique for variable selection, especially when a large amount of predictor variables are available, in order to obtain more parsimonious, and hence, more interpretable prediction models. Very briefly, one goal of LASSO regression models is to set some model coefficients to zero, reducing in this way the dimension of the model by selecting a subset of the available predictor variables. The selection of this subset depends in turn on the election of a tuning parameter (λ) for which techniques such as bootstrap or cross-validation can be applied, the latter being the most widely used technique, in practice. These techniques, commonly known as validation methods, are used in order to select the tuning parameter with which the error of the final model, evaluated in a sample different from the one used to develop the model, is minimized (see, e.g., Hastie et al., 2009; James et al., 2013). Shortly, those techniques consist in defining different training sets (in which models are fitted considering several tuning parameters) and test sets (in which the error of the models is estimated). The tuning parameter that minimizes the error of the training models in the test sets is selected for fitting the LASSO model to the whole sample.

However, fitting LASSO regression models to complex survey data could be problematic for two reasons. In the first place, the previously mentioned debate about the need for sampling weights when fitting prediction models could be extended to LASSO regression models. In addition, with the traditional above-mentioned validation methods training and test sets are randomly defined, without considering the sampling design in the process. This may be a problem when working with complex survey data given that PSUs could be split into training and test sets, which may lead the training sets to underestimate the variability produced due to the sampling process and underestimate population error. This problem is usually known as "data leakage" (Kaufman et al., 2012). Both of these problems (weights-related as well as design-related) have recently been discussed in the literature. McConville et al. (2017) proposed incorporating sampling weights into the LASSO linear regression estimation process and Kshirsagar et al. (2017) extended this proposal to logistic regression models. Nonetheless, both of them applied the traditional K-fold cross-validation, which consists in randomly splitting sampled units into K subsamples (or folds) and defining K training sets, excluding a different fold (test set) each time. Nevertheless, if we apply this method to complex survey data we may come across two types of problems. In the first place, sampling weights of the units in neither the training sets nor the test sets properly represent the entire finite population. Besides, and more importantly, as mentioned above, sampling design is not reflected in the way the folds are defined. Wieczorek et al. (2022) warned about this problem and proposed mimicking the structure of the sample obtained from the finite population in each fold. For example, for stratified sampling designs, Wieczorek et al. (2022) proposed making each fold a stratified sample of PSUs from each stratum, that is, creating simple random sample folds separately within each stratum (being all the elements from a given PSU placed in the same fold) and then combine them across strata. In this way, the weights of the units in the training and test sets represent the finite population properly and the variability of the data is also represented. However, as pointed out by the authors, it should be noted that in this way the number of folds could be limited by means of the sampled PSUs in each stratum (cannot be defined more folds than the maximum number of sampled PSUs per stratum). In other words, we need at least K PSUs per stratum for the proper application of this method. Furthermore, if we have a different (and non-proportional to K) number of PSUs in each stratum, the sampling weights of the training and test sets would also incorrectly represent the finite population.

In complex survey frameworks, other approaches, different from the abovementioned validation techniques, are usually used to define partially independent subsets of the sample. Those approaches are known as "replicate weights" methods. These methods consist of modifying the sampling weights to define new subsamples that replicate the original sample, in the way that these subsamples by means of these new weights (i.e., the "replicate weights") correctly represent the finite population. The most well-known replicate weights methods that are implemented in the `survey` R package (Lumley (2011), Lumley (2020) are Jackknife Repeated Replication (JKn), Balanced Repeated Replication (BRR), and Rescaling Bootstrap (Bootstrap). Note that Jackknife term is usually used in variance estimation framework, but this term is commonly denoted as leave-one-cluster-out (LOCO) (Merkle et al., 2019) or leave-one-group-out (LOGO) cross-validation (Kuhn & Johnson, 2019) when the goal is validation. However, in order to be consistent with the terminology used in the `survey` R package (Lumley, 2020), we denote this method as Jackknife throughout the paper.

Therefore, the aim of the present work is twofold. On the one hand, we aim to analyze the performance of the above-mentioned replicate weights methods, instead of traditional validation techniques, to select the tuning parameter for fitting LASSO regression models. On the other hand, our goal is to propose new methods to this end based on the idea of replicating weights. In particular, due to the popularity of cross-validation in this context, we propose a new design-based cross-validation method based on replicate weights, which will be more flexible than the one proposed by Wieczorek et al. (2022). In addition to the cross-validation, we also propose two new techniques (which we denote as split-sample repeated replication [split] and extrapolation [extrap]) to select the tuning parameter for LASSO models. In this study, we aim to analyze (a) the impact of considering complex designs when using validation techniques for the selection of the tuning parameter and (b) the impact of the sampling weights when fitting LASSO models. Therefore, we compare by means of a simulation study the performance of different proposals based on replicate weights, to (a) the traditional K-fold cross-validation that defines the folds by ignoring the sampling design but considers

sampling weights for fitting LASSO models (weighted simple random sample cross-validation, w-SRSCV), and (b) the unweighted simple random sample K-fold cross-validation ignoring weights for fitting LASSO models (unw-SRSCV).

The rest of the paper is organized as follows. In Section 2, the basic notation on linear and logistic regression models and LASSO regression are given, existing replication methods applied in this work for the selection of the tuning parameter are defined and new methods based on the idea of replicating weights are also proposed. The performance of all the methods is analyzed by means of a simulation study, which is described in Section 3. Finally, we close the paper with the main conclusions in Section 4.

2 | METHODS

This section is divided into three different parts. In Section 2.1, the basic notation is set, in Section 2.2 LASSO regression is described, and finally, Section 2.3 describes replicate weights methods considered in this study together with the new methods proposed by the authors.

2.1 | Basic notation

Let U denote a finite population of N units, for which information of the vector of p covariates $\boldsymbol{X}=(X_1,...,X_p)$ is available, that is, $\{\boldsymbol{x}_i=(x_{i1},...,x_{ip})\}_{i=1}^N$. Let S be a sample obtained from U following some complex sampling design. In addition to the information of covariates, values for the response variable Y are also known for sampled units: y_i, $\forall i\in S$. Furthermore, for each sampled unit $i\in S$, a sampling weight is assigned as $w_i=1/\pi_i$, where π_i indicates the probability of being sampled, $\forall i\in S$.

For a continuous response variable Y, the linear regression model for the observed data is defined as follows:

$$y_i=\boldsymbol{x}_i\boldsymbol{\beta}+\epsilon_i,\ \epsilon_i\sim N(0,\sigma^2), \tag{1}$$

and the vector of regression coefficients $\boldsymbol{\beta}=(\beta_0,\beta_1,...,\beta_p)^T$ are estimated ($\hat{\boldsymbol{\beta}}$) based on sample S by minimizing the residual sum of square (RSS):

$$\mathrm{RSS}(\boldsymbol{\beta})=\sum_{i\in S}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2. \tag{2}$$

In a similar way, if Y is a dichotomous response variable, the logistic regression model is defined as

$$logit(P(Y=1|\boldsymbol{x}_i))=logit(p(\boldsymbol{x}_i))=\ln\left[\frac{p(\boldsymbol{x}_i)}{1-p(\boldsymbol{x}_i)}\right]=\boldsymbol{x}_i\boldsymbol{\beta}, \tag{3}$$

where $p(\boldsymbol{x}_i)=\frac{e^{\boldsymbol{x}_i\boldsymbol{\beta}}}{1+e^{\boldsymbol{x}_i\boldsymbol{\beta}}}$ and $\hat{\boldsymbol{\beta}}$ is obtained by maximizing the log-likelihood function $\ell(\boldsymbol{\beta})$ (or equivalently, minimizing $-\ell(\boldsymbol{\beta})$):

$$\ell(\boldsymbol{\beta})=\sum_{i\in S}[y_i\ln(p(\boldsymbol{x}_i))+(1-y_i)\ln(1-p(\boldsymbol{x}_i))]. \tag{4}$$

However, when working with complex survey data, sampling weights should be considered when estimating model coefficients as proposed by Fuller (1975) and Binder (1983), and the weighted residual sum of square (WRSS) and the pseudo-log-likelihood ($p\ell$) functions are usually considered instead of (2) and (4), respectively:

$$\mathrm{WRSS}(\boldsymbol{\beta})=\sum_{i\in S}w_i\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \text{ and } p\ell(\boldsymbol{\beta})=\sum_{i\in S}w_i[y_i\ln(p(\boldsymbol{x}_i))+(1-y_i)\ln(1-p(\boldsymbol{x}_i))]. \tag{5}$$

After estimating regression coefficients, a value for the response variable can be estimated given the values of covariates $\boldsymbol{x}_i$ for unit $i\in U$ as $\hat{y}_i=\boldsymbol{x}_i\hat{\boldsymbol{\beta}}$ in linear regression framework and as $\hat{p}(\boldsymbol{x}_i)=\frac{e^{\boldsymbol{x}_i\hat{\boldsymbol{\beta}}}}{1+e^{\boldsymbol{x}_i\hat{\boldsymbol{\beta}}}}$ in logistic regression. In order to ease the notation, let us denote as $\hat{f}(\cdot)$ the fitted (either linear or logistic) model and as $\hat{f}(\boldsymbol{x}_i)$ the corresponding estimated response for unit i, hereinafter.

2.2 | LASSO regression for variable selection

When a large amount of predictor variables are available, the LASSO regression model is commonly used for variable selection. Briefly, this method forces some regression coefficients to zero, and thus more interpretable models are obtained. This variable selection method is described below.

For a given value of the tuning parameter λ, linear and logistic LASSO regression models are fitted by minimizing the following functions, respectively:

$$\min\left\{\text{RSS}(\boldsymbol{\beta})+\lambda\sum_{j=1}^{p}|\beta_j|\right\} \text{ and } \min\left\{-\ell(\boldsymbol{\beta})+\lambda\sum_{j=1}^{p}|\beta_j|\right\}. \tag{6}$$

In practice, K-fold cross-validation is usually applied to select the optimum value for λ in order to minimize the error of the fitted model. Sampled units are randomly split into K subsamples of the same size. For $\forall k=1,\ldots,K$, the k^{th} subsample is set as test set ($S_{\text{test}(k)}$), while the rest $K-1$ subsets form the training set ($S_{\text{tr}(k)}$). Then, a grid for λ values is defined ($\lambda_l,\ \forall l=1,\ldots,L$), and for each of these values, a model is fitted to each training set $S_{\text{tr}(k)}$, $\forall k=1,\ldots,K$ following (6) (let us denote this model (either linear or logistic) as $\hat{f}^{l}_{\text{tr}(k)}(\cdot)$) and applied to the test set (let $\hat{f}^{l}_{\text{tr}(k)}(\boldsymbol{x}_i)$ indicate the predicted value, $\forall i\in S_{\text{test}(k)}$). The estimation error for each unit is calculated by means of the loss function as follows:

$$\mathcal{L}(y_i,\hat{f}^{l}_{\text{tr}(k)}(\boldsymbol{x}_i))=\begin{cases}(y_i-\hat{f}^{l}_{\text{tr}(k)}(\boldsymbol{x}_i))^2, & \text{in linear regression framework,}\\ -y_i\ln(\hat{f}^{l}_{\text{tr}(k)}(\boldsymbol{x}_i))-(1-y_i)\ln(1-\hat{f}^{l}_{\text{tr}(k)}(\boldsymbol{x}_i)), & \text{in logistic regression framework.}\end{cases} \tag{7}$$

For $\forall k=1,\ldots,K$, the error in subset k and $\lambda_l,\ \forall l=1,\ldots,L$ is then estimated as follows:

$$\widehat{Err}^{l}_{(k)}=\frac{1}{n_{\text{test}(k)}}\sum_{i\in S_{\text{test}(k)}}\mathcal{L}(y_i,\hat{f}^{l}_{\text{tr}(k)}(\boldsymbol{x}_i)), \tag{8}$$

being $n_{\text{test}(k)}$ the size of $S_{\text{test}(k)}$, $\forall k=1,\ldots,K$. This process is repeated K times, by setting a different subset k as the test set each time. The cross-validated error for λ_l is estimated as follows:

$$\widehat{Err}_{CV}(\lambda_l)=\frac{1}{K}\sum_{k=1}^{K}\widehat{Err}^{l}_{(k)}. \tag{9}$$

Finally, the value that minimizes the cross-validated error is selected as the penalty parameter,

$$\Lambda=\text{argmin}\{\widehat{Err}_{CV}(\lambda_l),\forall l=1,\ldots,L\}, \tag{10}$$

and the model is fitted to the whole sample S including Λ as the tuning parameter in expression (6).

However, in all the process explained above, sampling design and sampling weights are not considered (let us denote it as the unweighted simple random sample cross-validation (unw-SRSCV), hereinafter). We believe that when working with complex survey data, sampling design should be considered in the whole process: (1) When fitting the model, (2) when defining training and test sets, and (3) when estimating the error. Below, we explain how we propose to address these three points as a whole.

In the first place, when fitting the LASSO regression models, sampling weights should be considered as follows instead of (6) for linear and logistic regression models, respectively (Kshirsagar et al., 2017; McConville et al., 2017):

$$\min\left\{\text{WRSS}(\boldsymbol{\beta})+\lambda\sum_{j=1}^{p}|\beta_j|\right\} \text{ and } \min\left\{-p\ell(\boldsymbol{\beta})+\lambda\sum_{j=1}^{p}|\beta_j|\right\}. \tag{11}$$

Second, in Section 2.3, we describe different methods based on replicate weights that could be considered to take into account the sampling design when defining training and test sets. Finally, sampling weights should also be considered when estimating the error. In particular, if we focus on the above-mentioned cross-validation method, we could rewrite Equation (8) as follows in order to consider the weights when estimating the error in subset k:

$$\widehat{Err}^{I}_{(k)} = \frac{1}{\sum_{i \in S_{\text{test}(k)}} w_i} \sum_{i \in S_{\text{test}(k)}} w_i \mathcal{L}(y_i, \hat{f}^{I}_{\text{tr}(k)}(\boldsymbol{x}_i)). \tag{12}$$

We denote as weighted simple random sample cross-validation (w-SRSCV) the method that considers (11) to fit the model and (12) to estimate the error but defines the folds by randomly splitting sampled units into different subsets ignoring the sampling design as described previously.

2.3 | Selecting LASSO model's tuning parameter with complex survey data

In this work, we propose to use replicate weights for the selection of the tuning parameter λ. In the following lines, we describe the six replicate weights methods we considered in this work to define training and test sets when selecting the tuning parameter for LASSO models. The goal of replicate weights methods is to modify the sampling weights to define new partially independent subsamples that replicate the original sample, in the way that the finite population is properly represented in each subsample by means of the modified weights known as "replicate weights". Some of the replicate weights methods that are described below are commonly applied for other purposes, such as variance estimation, when working with complex survey data. These methods are known as Jackknife Repeated Replication (JKn), Bootstrap, and BRR (see, e.g., Heeringa et al., 2017; Wolter, 2007, for more information about these methods). However, as far as we know, they have never been used for selecting the tuning parameter for LASSO regression models. In this work, we propose to incorporate the abovementioned replicate weights methods in this context. In addition, we also propose three new methods based on the idea of replicating weights, which we denote as the design-based K-fold cross-validation (dCV), Split-sample Repeated Replication (split), and extrapolation (extrap) for the same purpose. Figure 1 depicts a graphical summary of all these methods (note that the figures are not self-explanatory and should be read in combination with the descriptions below for a correct understanding of each method).

Let a_h indicate the number of sampled PSUs from stratum h, $\forall h = 1,...,H$, where $\sum_{h=1}^{H} a_h = a$ indicates the total number of sampled PSUs. In this study, a stratified cluster sampling has been considered; thus, PSUs are the clusters sampled from each stratum but note that all the methods described below can be extended to one-step stratified samples in which different numbers of individuals (which would be the PSUs in that case) are sampled from each stratum.

- Jackknife Repeated Replication (JKn): This method leaves one PSU α, $\forall \alpha = 1,...,a$ as test set ($S_{\text{test}(\alpha)}$), being the corresponding training sets ($S_{\text{tr}(\alpha)}$) defined by the rest $a-1$ PSUs, excluding the α^{th} one. A total of $T_{\text{JKn}} = a$ training and test sets are defined in this way, excluding one PSU from the training set each time. $\forall \alpha = 1,...,a$ let us suppose that α indicates a PSU from stratum h, $\forall h = 1,...,H$. Replicate weights are defined $\forall i \in S$, for the α^{th} training set as follows:

$$w^{*,\text{tr}(\alpha)}_{i,\text{JKn}} = \begin{cases} 0, & \text{if } i \in S_{\text{test}(\alpha)}, \\ w_i, & \text{if } i \in S_{\text{tr}(\alpha)} \text{ but it is not in stratum } h, \\ w_i \cdot \dfrac{a_h}{a_h - 1}, & \text{if } i \in S_{\text{tr}(\alpha)} \text{ and it is in stratum } h. \end{cases} \tag{13}$$

  Even though each test set is formed by the PSU excluded from the corresponding training set, the error is usually estimated considering the whole sample, so the replicate weights corresponding to the test set can be defined as the original sampling weights, that is, $w^{*,\text{test}}_{i,\text{JKn}} = w_i$.

- *Rescaling Bootstrap (Bootstrap)*: $T_{\text{Bootstrap}} = B$ bootstrap resamples are generated as proposed by Rao and Wu (1988). $\forall h = 1,...,H$ $a_h - 1$ PSUs are selected with replacement, which form the b^{th} training set, $\forall b = 1,...,B$. $\forall \alpha = 1,...,a$ let us denote as $v^{(b)}_{\alpha}$, the number of times that the α^{th} PSU is selected to be part of the bootstrap resample b. Note that if the α^{th} PSU is not selected for resample b, then $v^{(b)}_{\alpha} = 0$. Then, $\forall b = 1,...,B$ the replicate weights for the b^{th} training set are defined as follows. $\forall i \in S$, suppose i is a unit from the α^{th} PSU:

$$w^{*,\text{tr}(b)}_{i,\text{Bootstrap}} = w_i \cdot \frac{a_h}{a_h - 1} \cdot v^{(b)}_{\alpha}. \tag{14}$$

  The corresponding test set for each training set is the original sample S with its sampling weights, that is, $w^{*,\text{test}(b)}_{i,\text{Bootstrap}} = w_i$.

- *Balanced Repeated Replication (BRR)*: This method was originally designed to be applied in samples with 2 PSUs per stratum. $\forall h = 1,...,H$ one of the PSUs from the stratum is set to the training set while the other is set to the test set. There are 2^H different possible training and test sets to define in this way, which may usually be computationally unfeasible. Instead, T_{BRR} (where $T_{\text{BRR}} \leq H + 4$) the number of different sets are usually defined by selecting the PSU splits in a particular way by means of the Hadamard matrix as proposed by McCarthy (1966). Nowadays, this method is extended to be also applied when an even number of PSUs per stratum are available (Lumley, 2020; Wolter, 2007). $\forall i \in S$, the replicate weights for the t^{th} training set are defined as follows, $\forall t = 1,...,T_{\text{BRR}}$:

FIGURE 1 Graphical summary of replicate weights methods considered in this work. Existing methods are shown in (a) Jackknife Repeated Replication (JKn), (b) Rescaling Bootstrap (Bootstrap), and (c) Balanced Repeated Replication (BRR). New proposals are depict in (d) Design-based cross validation (dCV), (e) Split-sample Repeated Replication (split), and (f) Extrapolation (extrap). Note that each dot represents a PSU and gray lines define the strata.

$$w_{i,\text{BRR}}^{*,\text{tr}(t)} = \begin{cases} 0, & \text{if } i \in S_{\text{test}(t)}, \\ 2w_i, & \text{if } i \in S_{\text{tr}(t)}. \end{cases} \tag{15}$$

Replicate weights for the corresponding test sets ($w_{i,\text{BRR}}^{*,\text{test}(t)}$) are defined by exchanging the roles of test and training sets in (15).

- *Design-based K-fold cross-validation (dCV)*: The a sampled PSUs are randomly split into K subsets. For $k = 1,\ldots,K$ the k^{th} subset is set as test set ($S_{\text{test}(k)}$), being the training set ($S_{\text{tr}(k)}$) formed by the rest $K-1$ subsets. In this way, $T_{\text{dCV}} = K$ training and test sets are defined. For each sampled unit $i \in S$ from stratum h, we propose to define replicate weights as follows for the k^{th} training set:

$$w_{i,\text{dCV}}^{*,\text{tr}(k)} = \begin{cases} 0, & \text{if } i \in S_{\text{test}(k)}, \\ w_i \cdot \dfrac{a_h}{a_h - a_h^*}, & \text{if } i \in S_{\text{tr}(k)}, \end{cases} \tag{16}$$

being a_h^* the number of PSUs from stratum h in the k^{th} test set. In the same way, replicate weights for the k^{th} test sets (which will be denoted as $w_{i,\text{dCV}}^{*,\text{test}(k)}$ hereinafter) can be defined in the same way exchanging the roles of the training and test sets with each other in (16).

Note that our goal is to get at least one PSU from each stratum in every training set (not in every fold), and this will happen as long as no stratum ends up with all its PSUs in the same fold. In order to ensure this, after PSUs are randomly assigned to folds, dCV needs to check whether this condition is satisfied or not. In case any stratum has all its PSUs in the same fold, then it needs to reassign folds until this condition is satisfied. Therefore, at least two PSUs per stratum are needed, each of them classified in a different fold. This is an advantage over the method proposed by Wieczorek et al. (2022), which requires at least K PSUs per stratum.

- *Split-sample Repeated Replication (split)*: A given percentage of PSUs is randomly set into the training set and the rest into the test set. This process could be repeated T_{split} times with a different split each time, defining in this way T_{split} training and test sets. Replicate weights for units in both, training and test sets, can be defined in two different ways, $\forall t = 1,\ldots,T_{\text{split}}$:
  - *split-cv*: As described in Equation (16) for the dCV ($w_{i,\text{split-cv}}^{*,\text{tr}(t)}$, $w_{i,\text{split-cv}}^{*,\text{test}(t)}$).
  - *split-boot:* Replicating by replacement the PSUs until having $a_h - 1$ in each stratum and calculating the weights as in Equation (14) for the Bootstrap ($w_{i,\text{split-boot}}^{*,\text{tr}(t)}$, $w_{i,\text{split-boot}}^{*,\text{test}(t)}$).
- *Extrapolation (extrap)*: A given percentage of strata are set as training set and the rest as the test set. The process is repeated T_{extrap} times with a different split each time, defining T_{extrap} different training and test sets. In this case, replicate weights are equal to sampling weights for units in the training set and 0 for units in the test set when fitting the models, $\forall i \in S$. $\forall t = 1,\ldots,T_{\text{extrap}}$:

$$w_{i,\text{extrap}}^{*,\text{tr}(t)} = \begin{cases} 0, & \text{if } i \in S_{\text{test}(t)}, \\ w_i, & \text{if } i \in S_{\text{tr}(t)}. \end{cases} \tag{17}$$

Replicate weights $w_{i,\text{extrap}}^{*,\text{test}(t)}$ are described in the same way, exchanging the roles of training and test sets in (17).

3 | SIMULATION STUDY

This section describes the simulation study conducted in order to analyze the performance of different methods when selecting the tuning parameter for fitting LASSO regression models. Our goal is to compare the performance of the replication methods proposed in Section 2.3 (i.e., JKn, Bootstrap, BRR, dCV, split-cv, split-boot, and extrap) to the methods described in Section 2.2 (unw-SRSCV and w-SRSCV). The goal is to compare the differences between the tuning parameters selected with different methods, the number of covariates that would be selected if the models were fitted considering that tuning parameter, and the error we would obtain with that model. We compare those results with the "true" results we would obtain if the finite population were known in practice.

The rest of the section is organized as follows: Section 3.1 describes the process of data simulation and scenarios, Section 3.2 describes the simulation study process, and Section 3.3 depicts and summarizes the main results.

3.1 | Data generation and sampling design

In the following lines, data simulation process is described. Let us define as $N = 100000$ the finite population size and as $p = 75$ the number of variables denoted as $X_1,\ldots,X_{50},Z_1,\ldots,Z_{25}$. In this simulation study, we consider the variables $Z_1,\ldots,Z_{25}$ to be latent variables, which are used to define the response variable, but are not available in the samples to fit the models. In this way, we aim to define more realistic scenarios, in which the perfect models cannot be fitted. Instead, $Z_1,\ldots,Z_{25}$ are used to define the sampling design.

For a given value of p^*, where $p^* \leq p$, let $\boldsymbol{\mu}_{p^*}$ indicate the null vector of dimension $1 \times p^*$ and $\Sigma_{p^* \times p^*}$ a matrix of dimension $p^* \times p^*$ of values of $\eta = 0.15$ off-diagonal and values of 1 on the diagonal, that is,

$$\boldsymbol{\mu}_{(p^*)} = (0,...,0)^T \text{ and } \Sigma_{p^* \times p^*} = (1-\eta) \cdot I_{p^* \times p^*} + \eta \cdot J_{p^* \times p^*}, \tag{18}$$

being $I_{p^* \times p^*}$ the identity matrix and $J_{p^* \times p^*}$ the matrix of 1s. In addition, the vector of regression coefficients $\boldsymbol{\beta} = (\boldsymbol{\beta}^{\boldsymbol{X}}, \boldsymbol{\beta}^{\boldsymbol{Z}})^T$ is defined as follows:

$$\boldsymbol{\beta}^{\boldsymbol{X}} = \left(\underbrace{-2,...,-2}_{(11)}, \underbrace{0,...,0}_{(9)}, \underbrace{-2,...,-2}_{(10)}, \underbrace{0,...,0}_{(9)}, \underbrace{-2,...,-2}_{(11)} \right)^T, \boldsymbol{\beta}^{\boldsymbol{Z}} = \left(\underbrace{2,...,2}_{(25)} \right)^T. \tag{19}$$

We describe below the steps followed to generate the finite populations of this simulation study. Two scenarios are defined generating two different populations. In Scenario 1 (S1), the $p = 75$ covariates are unit-level variables (i.e., there are $q = 0$ cluster-level variables), while in Scenario 2 (S2), $q = 5$ variables are defined as cluster-level variables, while the rest of $p - q = 70$ variables are unit-level. In each population, two response variables have been generated: (a) A continuous response variable (linear regression), and (b) a dichotomous response variable (logistic regression).

1. For $q = 0$ (S1) and $q = 5$ (S2), two finite populations are generated by making N realizations of

$$(X_{q+1},...,X_{50},Z_1,...,Z_{25}) \sim N(\boldsymbol{\mu}_{(p-q)}, \Sigma_{(p-q)\times(p-q)}). \tag{20}$$

2. Let us denote as $\{\boldsymbol{z}_i = (z_{i,1},...,z_{i,25})\}_{i=1}^N$ the set of N realizations of $Z_1,...,Z_{25}$. Data are sort based on $\boldsymbol{z}_i\boldsymbol{\beta}^{\boldsymbol{Z}}$, $\forall i = 1,...,N$. Strata are defined by partitioning the population data set on sets of the same size ($H = 5$) and clusters by partitioning each stratum on sets of the same size ($A_h = 20, \forall h = 1,...,H$, being A_h the number of clusters generated in stratum h in the population). In this way, a total of 100 clusters of size $N_{h,\alpha} = 1000$ are generated, $\forall h = 1,...,H$ and $\forall \alpha = 1,...A_h$.
3. If $q \neq 0$, generate q cluster-level variables by making $A = \sum_{h=1}^H A_h$ realizations of

$$(X_1,...,X_q) \sim N(\boldsymbol{\mu}_{(q)}, \Sigma_{(q)\times(q)}). \tag{21}$$

   Note that for two different units in the same cluster, their corresponding cluster-level covariates should take the same values, that is, $\forall i,j$ in the same cluster, $(x_{i,1},...,x_{i,q}) = (x_{j,1},...,x_{j,q})$. Therefore, we repeat each realization $N_{h,\alpha}$ times. We now have defined the values corresponding to $X_1,...,X_{50}$ variables for all the units in the finite population: $\{\boldsymbol{x}_i = (x_{i,1},...,x_{i,50})\}_{i=1}^N$.
4. Generate the values for the response variables as follows:
   a. Linear regression framework: $y_i = \boldsymbol{x}_i\boldsymbol{\beta}^{\boldsymbol{X}} + \boldsymbol{z}_i\boldsymbol{\beta}^{\boldsymbol{Z}} + \epsilon_i$, where ϵ_i is a realization of $N(0,10^2)$, $\forall i = 1,...,N$.
   b. Logistic regression framework: $logit(p(\boldsymbol{x}_i,\boldsymbol{z}_i)) = \beta_0 + \boldsymbol{x}_i\boldsymbol{\beta}^{\boldsymbol{X}} + \boldsymbol{z}_i\boldsymbol{\beta}^{\boldsymbol{Z}}$, and the value for the response variable y_i is randomly generated by following $Bernoulli(p(\boldsymbol{x}_i,\boldsymbol{z}_i))$. We set $\beta_0 = -10$, defining in this way a prevalence (i.e., probability of event) of around 25%.

   Then, the finite population U is defined as the set of values corresponding to the response variable y_i and the covariates $\boldsymbol{x}_i$ (excluding the latent variables $\boldsymbol{z}_i$), $\forall i = 1,...,N$ as well as strata and cluster indicators corresponding to each of them.
5. Sampling design is defined as a stratified cluster sampling scheme. First, $a_h = 4$, $\forall h = 1,...,H$ clusters or PSUs are sampled per stratum. Afterward, a different number of units is sampled from each PSU of each stratum ($n_{h,\alpha}$). In particular, $\forall \alpha = 1,...,4$ PSUs for each stratum, the following number of units have been sampled in each scenario:

$$\begin{aligned}
&\mathbf{S1}(\mathbf{G}): n_{1,\alpha} = 500,\ n_{2,\alpha} = 50,\ n_{3,\alpha} = 25,\ n_{4,\alpha} = 10,\ n_{5,\alpha} = 5, \quad && \mathbf{S1}(\mathbf{B}): n_{1,\alpha} = 5,\ n_{2,\alpha} = 10,\ n_{3,\alpha} = 25,\ n_{4,\alpha} = 50,\ n_{5,\alpha} = 500,\\
&\mathbf{S2}(\mathbf{G}): n_{1,\alpha} = 250,\ n_{2,\alpha} = 100,\ n_{3,\alpha} = 50,\ n_{4,\alpha} = 25,\ n_{5,\alpha} = 5, \quad && \mathbf{S2}(\mathbf{B}): n_{1,\alpha} = 5,\ n_{2,\alpha} = 25,\ n_{3,\alpha} = 50,\ n_{4,\alpha} = 100,\ n_{5,\alpha} = 250.
\end{aligned}$$

   Note that the names of the scenarios refer to the distribution of the response variable: "G" for Gaussian distribution with reference to the framework of the linear regression and "B" for the Bernoulli distribution indicating the logistic regression framework.
6. For $i \in S$, suppose i is a unit sampled from PSU α, $\forall \alpha = 1,...,a_h$ in stratum h, $\forall h = 1,...,H$. The sampling weight for unit i is then calculated as follows:

$$w_i = \frac{A_h}{a_h} \cdot \frac{N_{h,\alpha}}{n_{h,\alpha}}. \tag{22}$$

3.2 | Set up

As explained above, in this simulation study we aim to compare the performance of the methods described in Section 2.3 to the w-SRSCV and to the unw-SRSCV (both of them described in Section 2.2). In order to ease the notation, we could define replicate weights for training and test sets of the w-SRSCV as the original weights, that is, $w^{*,\text{tr}(t)}_{i,\text{w-SRSCV}} = w^{*,\text{test}(t)}_{i,\text{w-SRSCV}} = w_i,\ \forall i = 1,...,n$ for the t^{th} training and test sets. In the same way, the unw-SRSCV would be equivalent to setting all the replicate weights for training and test sets to one: $w^{*,\text{tr}(t)}_{i,\text{unw-SRSCV}} = w^{*,\text{test}(t)}_{i,\text{unw-SRSCV}} = 1,\ \forall i = 1,...,n$ for the t^{th} training and test sets. Considering this notation, the lines below describe the process of the simulation study for all the methods, including w-SRSCV and unw-SRSCV. For $r = 1,...,R$:

Step 1. Obtain the sample S^r.

Step 2. Define the penalty grid based on S^r using the default approach in `glmnet` (Friedman et al., 2010): $\lambda^r_1,...,\lambda^r_L$.

Step 3. For each value $\lambda^r_l,\ \forall l = 1,...,L$:

Step 3.1 Fit the model to S^r ($\hat{f}^{r,l}(\cdot)$) considering the vector of covariates $\boldsymbol{x}_i$, sampling weights $w_i,\ \forall i \in S^r$ and λ^r_l following Equation (11).

Step 3.2 Apply the model $\hat{f}^{r,l}(\cdot)$ to the finite population and estimate the response: $\hat{f}^{r,l}(\boldsymbol{x}_i),\ \forall i = 1,...,N$.

Step 3.3 Calculate the error of the model $\hat{f}^{r,l}(\cdot)$ in the finite population (i.e., true population error of the model fitted to S^r):

$$\widehat{Err}^{r}_{\text{true}}(\lambda^r_l) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(y_i, \hat{f}^{r,l}(\boldsymbol{x}_i)), \tag{23}$$

where $\mathcal{L}(y_i, \hat{f}^{r,l}(\boldsymbol{x}_i))$ is calculated as in (7).

Step 3.4 Define the "true" optimal tuning parameter (not available in practice):

$$\Lambda^r_{\text{true}} = \text{argmin}\{\widehat{Err}^{r}_{\text{true}}(\lambda^r_l),\ l = 1,...,L\}. \tag{24}$$

Step 4. For each method m, where $m \in \{$JKn, dCV, Bootstrap, BRR, split-cv, split-boot, extrap, w-SRSCV, unw-SRSCV$\}$:

Step 4.1 Define training and test sets ($S^{r,m}_{\text{tr}(t)}$ and $S^{r,m}_{\text{test}(t)},\ \forall t = 1,...,T_m$, respectively) and calculate the corresponding replicate weights for the sampled units: $w^{*,r,\text{tr}(t)}_{i,m}$ and $w^{*,r,\text{test}(t)}_{i,m},\ \forall i \in S^r$.

Step 4.2 For $t = 1,...,T_m$ and $l = 1,...,L$:

Step 4.2.1 Fit the model to $S^{r,m}_{\text{tr}(t)}$ considering λ^r_l and the corresponding replicate weights $w^{*,r,\text{tr}(t)}_{i,m}$ following (11): $\hat{f}^{r,l,m}_{\text{tr}(t)}(\cdot)$.

Step 4.2.2 Apply $\hat{f}^{r,l,m}_{\text{tr}(t)}(\cdot)$ to $\forall i \in S^{r,m}_{\text{test}(t)}$ and estimate the response: $\hat{f}^{r,l,m}_{\text{tr}(t)}(\boldsymbol{x}_i)$. Calculate the error of the model in the test set (this is the error that can be estimated in practice):

$$\widehat{Err}^{r,m,t}_{\text{test}}(\lambda^r_l) = \frac{1}{\sum_{i \in S^{r,m}_{\text{test}(t)}} w^{*,r,\text{test}(t)}_{i,m}} \sum_{i \in S^{r,m}_{\text{test}(t)}} w^{*,r,\text{test}(t)}_{i,m} \mathcal{L}(y_i, \hat{f}^{r,l,m}_{\text{tr}(t)}(\boldsymbol{x}_i)). \tag{25}$$

Step 4.3 Define the average error of the training models in the test sets:

$$\widehat{Err}^{r,m}_{\text{test}}(\lambda^r_l) = \frac{1}{T_m}\sum_{t=1}^{T_m} \widehat{Err}^{r,m,t}_{\text{test}}(\lambda^r_l). \tag{26}$$

Step 4.4 Define the optimal tuning parameter for method m as follows:

$$\Lambda^{r,m}_{\text{test}} = \text{argmin}\{\widehat{Err}^{r,m}_{\text{test}}(\lambda^r_l),\ l = 1,...,L\}. \tag{27}$$

Step 5. Analyze the performance of each method m comparing the results to the "true" ones as follows:

Step 5.1 Define the difference between the true optimal tuning parameter and the one obtained based on method m as follows:

$$\text{diff}^{r,m} = \log(\Lambda^{r,m}_{\text{test}}) - \log(\Lambda^r_{\text{true}}). \tag{28}$$

Step 5.2 Define d^r_{true} and $d^{r,m}_{\text{test}}$ as the number of regression coefficients different to 0, when fitting LASSO models considering the tuning parameters Λ^r_{true} and $\Lambda^{r,m}_{\text{test}}$, respectively. The former indicates the number of variables that would be selected based on LASSO in case the finite population were available (i.e., the "true" number of variables selected based on LASSO), while the latter indicates the number of variables that would be selected based on each method m.

Step 5.3 For each selected tuning parameter Λ^r_{true} and $\Lambda^{r,m}_{\text{test}}$, the true population error (obtained by applying to the finite population, the LASSO regression model fitted to the whole sample S^r considering Λ^r_{true} and $\Lambda^{r,m}_{\text{test}}$ parameters, respectively) is calculated following (23) as $\widehat{Err}^r_{\text{true}}(\Lambda^r_{\text{true}})$ and $\widehat{Err}^r_{\text{true}}(\Lambda^{r,m}_{\text{test}})$.

In this simulation study, a total of $R = 500$ samples were obtained. Cross-validation methods were applied for $K = 10$ number of folds, $B = 200$ bootstrap resamples were considered, and a total of 20 train and test sets were defined for split and extrap methods. For split methods, 70% of clusters are used for defining training sets, while for extrap training sets are defined by means of 3 out of 5 strata. All computations were performed in (64 bit) R 4.2.0 (R Core Team, 2022) and a workstation equipped with 32GB of RAM, an Intel i7-8700 processor (3.20 Ghz), and a Windows 10 operating system. In particular, LASSO models were fitted by means of `glmnet` R package (Friedman et al., 2010) and for applying JKn, BRR, and Bootstrap methods, `survey` package (Lumley, 2020) was used.

3.3 | Results

In this section, we summarize the main results we obtained from the simulation study. Figure 2 depicts the differences between the logarithms of the true optimum tuning parameter and the ones obtained for each method (see Equation (28)), while Figure 3 shows the number of variables that would be selected based on those tuning parameters. Numerical results for linear (S1(G) and S2(G)) and logistic (S1(B) and S2(B)) models with cluster-level variables (S2(G) and S2(B)) or without them (S1(G) and S1(B)) are described in Table 1. Due to the large number of results, we proceed to summarize the main findings.

The unw-SRSCV performs poorly in all scenarios, it selects unnecessarily complex models with a large number of variables. In particular, in more than 50% of the samples in scenario S1(G), all the 50 covariates are kept in the final model based on this method. This method is also the one with the highest error in all scenarios except in S1(G), where the extrap method showed the worst results in this aspect. Such a bad performance indicates the need of considering sampling weights when fitting LASSO models to complex survey data. In contrast, BRR, split-cv, split-boot, and, in particular, extrap methods select large tuning parameters that lead to models with very few numbers of covariates, increasing the population error estimated based on them.

The performance of the rest of the methods depends on the scenario. No great differences have been observed comparing the results obtained from scenarios related to linear (S1(G) and S2(G)) and logistic (S1(B) and S2(B)) regression models. However, the results obtained in scenarios with cluster-level variables (S2(G) and S2(B)) or without them (S1(G) and S1(B)) differ considerably for some methods. In S1(G) and S1(B), the selected tuning parameters based on the JKn and the dCV are unbiased with respect to the true population parameter, which leads to keeping a similar number of variables in the final models. In S2(G) and S2(B), these methods select slightly greater tuning parameters, which leads them to select in general models with less number of variables. Nevertheless, there are no great differences in terms of error, compared with the error that would be obtained if the true tuning parameter were selected. Therefore, it can be concluded that the performance of these methods is correct in all scenarios. In S1(G) and S1(B), there are no differences between w-SRSCV and dCV methods, and as mentioned above, all of them perform quite properly. In contrast, in S2(G) and S2(B), the tuning parameter selected by means of w-SRSCV is lower than the true one, which leads to select unnecessarily complex models with a large number of parameters, without a gain in terms of the error of the model, in comparison to the dCV method. The Bootstrap method is the one that performs the best in terms of the error in all the scenarios. However, its performance in terms of the number of variables of the selected models depends on the scenario: Even though it shows a good performance selecting a similar number of covariates to the true model in S2(G) and S2(B), it tends to select a too large number of covariates in the models corresponding to S1(G) and S1(B).

4 | DISCUSSION

In this study, we worked on the variable selection process by means of LASSO regression models for complex survey data. As discussed throughout the paper, two issues need to be analyzed before implementing LASSO regression to complex surveys. In the first place, the need of incorporating sampling weights into the estimation process of LASSO models should be checked. In addition, the validity of the traditional cross-validation techniques commonly applied to simple random samples for selecting the tuning parameters for LASSO models should also be analyzed when working with complex survey data. In this paper, the performance of methods based on replicate weights that are well known for other purposes in complex survey data framework but, to our knowledge, have never been used for LASSO have been compared with the traditional cross-

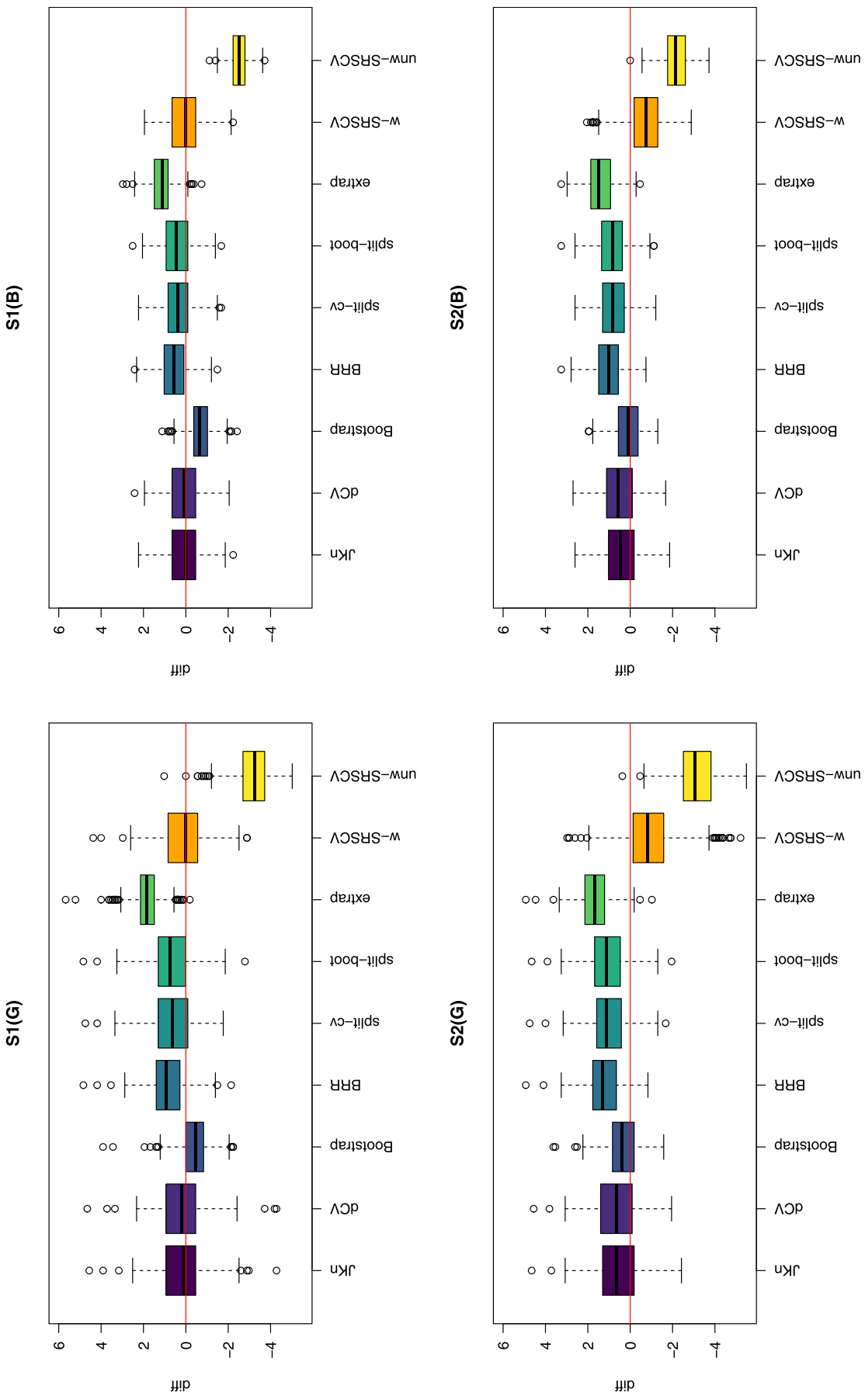

FIGURE 2 Box-plots of the differences between the logarithm of the true optimal tuning parameter and the one obtained based on each analyzed method (see (28)) across $R = 500$ samples in S1(G) (linear regression with $q = 0$ cluster-level variables), S1(B) (logistic regression with $q = 0$ cluster-level variables), S2(G) (linear regression with $q = 5$ cluster-level variables), and S2(B) (logistic regression with $q = 5$ cluster-level variables)

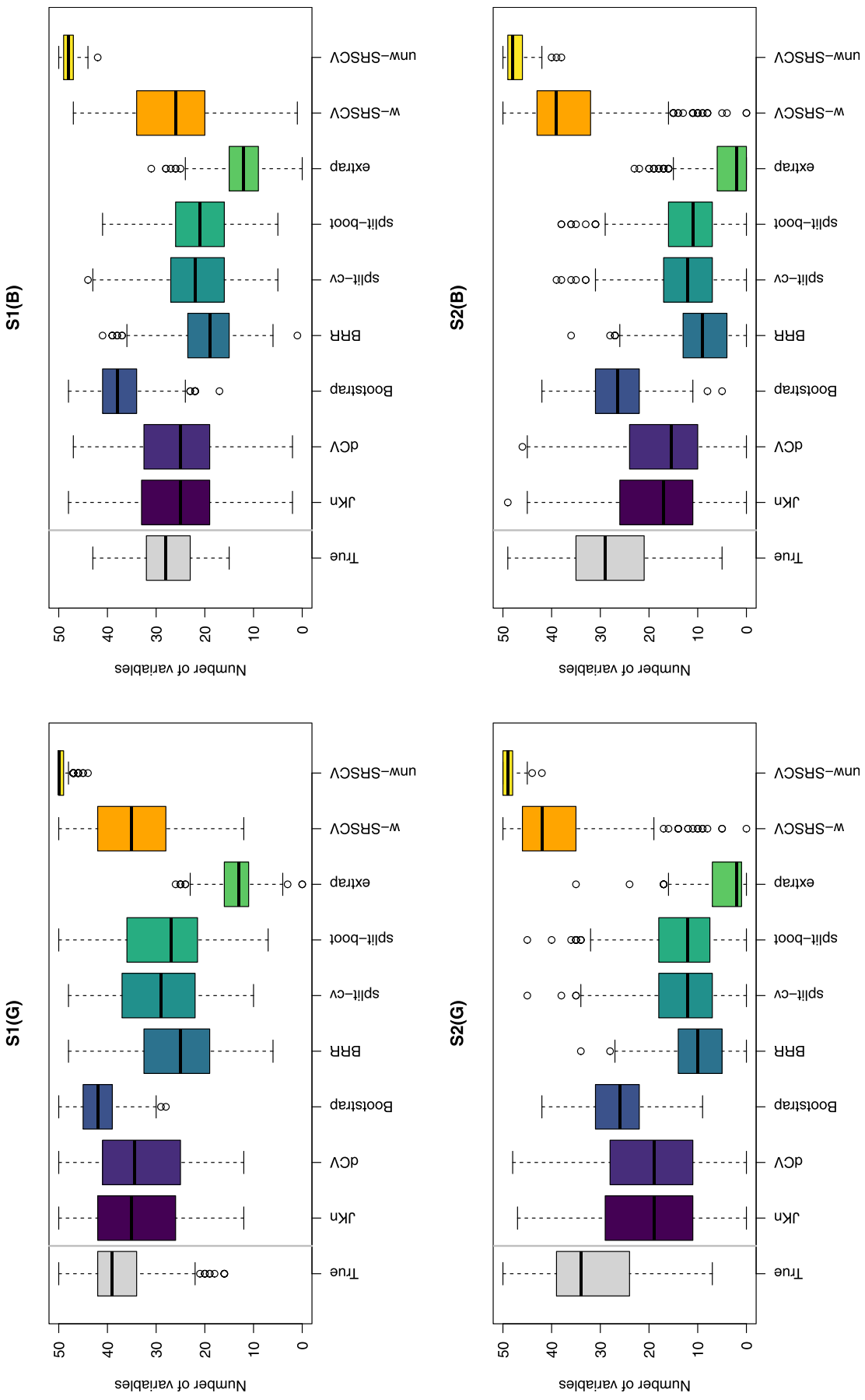

FIGURE 3 Box-plots of the number of variables considered in models fitted across $R = 500$ samples from S1(G) (linear regression with $q = 0$ cluster-level variables), S1(B) (logistic regression with $q = 0$ cluster-level variables), S2(G) (linear regression with $q = 5$ cluster-level variables), and S2(B) (logistic regression with $q = 5$ cluster-level variables) considering the optimal tuning parameters selected based on each method m (i.e., $d^{\mathbf{r}}_m$). Gray box-plot (denoted as "True") indicates the number of variables of the models fitted to the same samples considering the true optimal tuning parameters that minimize the population error defined in (24) (i.e., $d^{\mathbf{r}}_{\text{true}}$)

TABLE 1 Summary of the numerical results obtained in the simulation study.

	S1(G)				S2(G)			
	Log(lambda)				Log(lambda)			
	Min	Max	Mean (sd)	Median (Q1–Q3)	Min	Max	Mean (sd)	Median (Q1–Q3)
True	−4.78	0.92	−0.92 (0.60)	−0.90 (−1.23, −0.58)	−3.91	0.93	−0.52 (0.67)	−0.58 (−0.97, −0.03)
JKn	−4.49	0.79	−0.77 (0.76)	−0.73 (−1.25, −0.21)	−1.82	1.37	0.06 (0.58)	0.10 (−0.33, 0.48)
dCV	−4.80	0.79	−0.71 (0.75)	−0.70 (−1.17, −0.16)	−2.24	1.40	0.11 (0.58)	0.15 (−0.29, 0.53)
Bootstrap	−2.35	−0.63	−1.35 (0.30)	−1.34 (−1.55, −1.16)	−1.12	0.70	−0.17 (0.22)	−0.19 (−0.31, −0.04)
BRR	−2.35	0.93	−0.08 (0.48)	−0.02 (−0.41, 0.30)	−0.35	1.54	0.70 (0.29)	0.68 (0.51, 0.88)
split-cv	−1.97	0.93	−0.33 (0.55)	−0.29 (−0.72, 0.11)	−1.14	1.37	0.50 (0.37)	0.51 (0.26, 0.73)
split-boot	−3.27	1.11	−0.28 (0.58)	−0.24 (−0.68, 0.18)	−1.14	1.44	0.51 (0.36)	0.53 (0.30, 0.73)
extrap	−0.11	1.63	0.90 (0.19)	0.91 (0.80, 1.02)	−0.54	1.70	1.11 (0.22)	1.13 (0.97, 1.25)
w-SRSCV	−3.31	0.56	−0.81 (0.67)	−0.76 (−1.24, −0.32)	−4.96	1.20	−1.40 (0.92)	−1.36 (−1.74, −0.82)
unw-SRSCV	−4.87	−2.20	−4.08 (0.64)	−4.27 (−4.63, −3.63)	−5.08	−2.17	−3.64 (0.72)	−3.52 (−4.14, −3.08)
	Error				Error			
	Min	Max	Mean (sd)	Median (Q1–Q3)	Min	Max	Mean (sd)	Median (Q1–Q3)
True	240.55	266.16	252.34 (4.73)	252.51 (248.92, 255.39)	269.98	303.91	285.76 (5.74)	285.85 (281.65, 289.52)
JKn	240.73	283.70	256.21 (6.84)	255.53 (251.49, 260.04)	275.10	490.81	295.20 (15.27)	292.56 (288.28, 298.73)
dCV	240.88	285.33	256.27 (6.77)	255.64 (251.62, 260.10)	275.57	490.81	295.31 (14.62)	292.83 (288.30, 298.08)
Bootstrap	241.04	284.29	254.55 (6.36)	254.07 (250.04, 258.05)	277.98	504.60	291.60 (14.53)	289.03 (286.00, 293.34)
BRR	242.95	304.80	258.84 (7.56)	258.18 (254.04, 262.82)	284.93	405.67	298.89 (8.77)	297.22 (292.78, 303.65)
split-cv	240.73	291.18	257.30 (7.17)	256.54 (252.53, 261.02)	281.55	442.20	297.09 (10.63)	294.89 (291.09, 301.17)
split-boot	241.74	303.09	257.86 (7.40)	256.93 (253.18, 261.62)	281.72	442.20	297.13 (10.80)	295.00 (291.46, 300.93)
extrap	253.49	329.70	276.76 (12.32)	275.49 (267.76, 283.74)	288.68	442.20	306.51 (9.02)	308.40 (301.13, 311.66)
w-SRSCV	240.99	290.83	255.80 (6.30)	255.28 (251.53, 259.43)	271.68	597.24	298.29 (25.92)	291.34 (285.29, 303.47)
unw-SRSCV	242.04	296.92	259.99 (8.36)	259.05 (253.90, 265.43)	269.99	667.99	307.57 (33.50)	299.21 (289.83, 315.53)
	S1(B)				S2(B)			
	Log(lambda)				Log(lambda)			
	Min	Max	Mean (sd)	Median (Q1–Q3)	Min	Max	Mean (sd)	Median (Q1–Q3)
True	−5.55	−2.79	−4.05 (0.38)	−4.07 (−4.30, −3.81)	−5.96	−2.73	−4.04 (0.53)	−4.06 (−4.40, −3.66)
JKn	−5.59	−2.73	−3.98 (0.53)	−3.93 (−4.37, −3.56)	−5.35	−2.49	−3.59 (0.51)	−3.53 (−3.93, −3.23)
dCV	−5.39	−2.82	−3.95 (0.51)	−3.88 (−4.35, −3.53)	−5.21	−2.49	−3.52 (0.49)	−3.46 (−3.82, −3.17)
Bootstrap	−5.59	−3.93	−4.74 (0.27)	−4.74 (−4.92, −4.55)	−4.84	−3.34	−3.94 (0.20)	−3.94 (−4.07, −3.80)
BRR	−4.71	−2.45	−3.51 (0.34)	−3.47 (−3.70, −3.28)	−4.05	−2.36	−3.04 (0.27)	−3.06 (−3.21, −2.87)
split-cv	−5.03	−2.81	−3.68 (0.42)	−3.62 (−3.95, −3.36)	−4.42	−2.36	−3.24 (0.35)	−3.22 (−3.46, −2.99)
split-boot	−5.12	−2.64	−3.64 (0.41)	−3.59 (−3.88, −3.34)	−4.37	−2.43	−3.20 (0.34)	−3.20 (−3.40, −2.98)
extrap	−3.82	−2.12	−2.91 (0.22)	−2.89 (−3.04, −2.76)	−3.48	−2.14	−2.65 (0.20)	−2.63 (−2.76, −2.53)
w-SRSCV	−5.65	−2.38	−4.04 (0.52)	−4.02 (−4.40, −3.62)	−6.33	−2.75	−4.71 (0.61)	−4.72 (−5.12, −4.41)
unw-SRSCV	−7.00	−5.91	−6.58 (0.22)	−6.59 (−6.74, −6.43)	−6.99	−5.22	−6.21 (0.27)	−6.21 (−6.40, −6.03)
	Error				Error			
	Min	Max	Mean (sd)	Median (Q1–Q3)	Min	Max	Mean (sd)	Median (Q1–Q3)
True	0.46	0.50	0.48 (0.01)	0.48 (0.48, 0.49)	0.53	0.58	0.55 (0.01)	0.55 (0.55, 0.56)
JKn	0.47	0.60	0.49 (0.01)	0.49 (0.48, 0.50)	0.54	0.92	0.57 (0.02)	0.56 (0.56, 0.57)
dCV	0.47	0.58	0.49 (0.01)	0.49 (0.48, 0.50)	0.54	0.89	0.57 (0.02)	0.56 (0.56, 0.57)
Bootstrap	0.47	0.61	0.49 (0.02)	0.49 (0.48, 0.50)	0.54	0.76	0.56 (0.02)	0.56 (0.55, 0.56)
BRR	0.47	0.56	0.49 (0.01)	0.49 (0.48, 0.50)	0.55	0.59	0.57 (0.01)	0.57 (0.56, 0.58)

(Continues)

TABLE 1 (Continued)

	S1(G)				S2(G)			
	Log(lambda)				Log(lambda)			
	Min	Max	Mean (sd)	Median (Q1–Q3)	Min	Max	Mean (sd)	Median (Q1–Q3)
split-cv	0.47	0.56	0.49 (0.01)	0.49 (0.48, 0.50)	0.55	0.66	0.57 (0.01)	0.57 (0.56, 0.57)
split-boot	0.47	0.57	0.49 (0.01)	0.49 (0.48, 0.50)	0.54	0.63	0.57 (0.01)	0.57 (0.56, 0.57)
extrap	0.48	0.59	0.51 (0.02)	0.51 (0.50, 0.52)	0.55	0.59	0.58 (0.01)	0.58 (0.57, 0.59)
w-SRSCV	0.47	0.60	0.49 (0.01)	0.49 (0.48, 0.50)	0.53	1.21	0.58 (0.05)	0.56 (0.55, 0.58)
unw-SRSCV	0.48	0.68	0.53 (0.02)	0.53 (0.52,0.55)	0.54	1.31	0.60 (0.06)	0.59 (0.57, 0.62)

Note: For each scenario, the minimum (min) and maximum (max) values, the average (mean) and standard deviation (sd) and the median and interquartile range (Q1–Q3) are displayed for the logarithm of the optimal tuning parameters (Λ^r_{true} and $\Lambda^{r,m}_{\text{test}}$) and the corresponding population errors ($\widehat{Err}^r_{\text{true}}(\Lambda^r_{\text{true}})$ and $\widehat{Err}^r_{\text{true}}(\Lambda^{r,m}_{\text{test}})$) obtained across $R = 500$ samples for each analyzed method m.

validation techniques to select the tuning parameter Λ. In addition, new methods based on the idea of replicating weights have been proposed, among others, the dCV. This method could be seen as an extension of the Survey CV method proposed by Wieczorek et al. (2022), which in combination with replicate weights allows us to be more flexible when defining different folds, and thus, it is valid for more types of designs, for example when different number of PSUs per stratum is available, or a few numbers of PSUs per stratum are sampled.

The performance of all those methods for selecting the tuning parameter for LASSO models has been compared by means of an extensive simulation study. The sampling design considered in this study is a stratified cluster sampling in which a different number of units are sampled from each cluster. In order to reduce the number of results shown in the paper, we set the number of folds as $K = 10$, given that it is the one most commonly used in the literature (see, e.g., Witten et al., 2016). Let us highlight some of the most interesting conclusions of the simulation study in the following lines.

In the first place, the bad performance of the unw-SRSCV, which leads to very complex regression models selecting almost all variables, shows the need of incorporating sampling weights into the estimation process of LASSO regression models. It should be noted that in this paper we have not considered the option to fit perfect prediction models for which the sampling design is uninformative given the covariates included in the model. In line with previous works (see, e.g., Pfeffermann, 1993; Scott & Wild, 1986; Sugden & Smith, 1984), if we try to fit perfect models, sampling weights are not needed in the estimation process of linear and logistic regression models. These conclusions can also be extended to LASSO models, and hence, this method would perform properly in that situation. However, it is important to point out that when working with LASSO regression models, as we are using a sparse shrinkage estimator, the sampling design must be uninformative given, not all the covariates, but the ones that actually end up in the final model, which is even more complicated and beyond the researcher's control. In addition, it should also be noted that when we work with real data, we will hardly ever be able to fit "perfect" regression models. Therefore, we would not recommend the use of this method in practice, in order to avoid fitting too complex regression models with biased estimates of regression coefficients.

The second point that should be mentioned is the similarities and differences between the performance of the w-SRSCV (which does not consider the sampling design when defining folds) and the new proposal dCV. It is striking that for the same sampling design, such different results were obtained across different scenarios. This fact could be explained as follows. In the scenario where no cluster-level variable was incorporated, most of the variability offered by the cluster could be explained by means of the sampling weights. In contrast, the inclusion of cluster-level variables leads to an increase in the effect of the sampling design relative to clustering, thus offering greater differences between one method and the other. When no cluster-level variable is considered in the model, both methods perform properly and lead to reasonable and parsimonious regression models. In contrast, when including cluster-level variables into the process, models selected based on those methods differ considerably, being the ones selected by the w-SRSCV more complex than necessary. This is in line with the results obtained by Lumley and Scott (2015), in which the effect of the sampling design has shown an important role in the model selection, in particular, on the Akaike information criterion (AIC) and Bayesian information criterion (BIC). Briefly, this work shows that for samples with greater design effect, more parsimonious models are selected, given that the design effect penalizes more strongly the incorporation of the covariates into the model. Coming back to our study, we have observed that the greater the cluster effect, the greater the differences between the tuning parameters selected for fitting the models and the number of variables selected based on those methods. The w-SRSCV tends to select a larger number of variables than the dCV. Therefore, we recommend the use of the dCV rather than the w-SRSCV, in order to select more parsimonious models when fitting LASSO regression models to complex survey data. In addition, given the similarities of both works, we believe that the trace of the variance-covariance matrix (defined by means of the information matrix and score vector) which is used to estimate the design-based AIC proposed by Lumley and Scott (2015) could be used to analyze the cluster effect and thus diagnose whether there may be differences between the dCV and w-SRSCV. Some numerical results related to this issue are shown as Supporting Information.

However, the magnitude of the relationship between the trace of the variance-covariance matrix and the differences between w-SRSCV and dCV on the variable selection by means of LASSO will be further analyzed as future work.

Another consequence of the abovementioned cluster effect, which is reflected in the differences between the dCV and w-SRSCV, is the so-called "data leakage" (Kaufman et al., 2012). When the cluster effect is significant, splitting the clusters between training and test sets (i.e., setting some units of a cluster into the training set and others into the test set as w-SRSCV does) has two consequences. On the one hand, we fit the models with more information than we should, given that all the clusters are considered in the process. The fact that all the clusters are considered when fitting the training models means that the sampling variability will be underestimated across them. On the other hand, very similar information to the one used when fitting the models is used to evaluate the error in the test sets given that, actually, the training and test sets are not independent sets since units of the same cluster are in both of them. Thus, the true population error is also underestimated. These issues can be observed in the figures added as Supporting Information. When applying the training models to the test sets, it can be observed that the w-SRSCV underestimates the true population error in contrast to the dCV, particularly in the scenarios with cluster-level variables. It can also be observed that the variability of the training models is greater when the dCV is applied compared with the w-SRSCV.

Note that the methods proposed and applied throughout this work can be extended in a very simple way to data obtained from a stratified sampling design without clustering. However, the behavior of the methods in such a situation has not been analyzed in this simulation study. The authors expect that the results may be similar to scenario 1, where cluster-level variables have not been incorporated, but this should be studied in future work to be confirmed. Neither other types of sampling such as sampling probability proportional to sample size nor post-stratification have been considered, so the conclusions obtained are limited to the schemes we have analyzed. Note also that other tries have been made by changing the number of folds to $K = 5$, but no significant differences have been observed (results not shown). Also, cross-validation techniques allow repeating the process of splitting the sample several times, which is usually known as cross-validation with replication. Those replicates have not been considered in the results shown in this paper due to the same reason. Finally, the methods applied and proposed in this work could be used for other purposes beyond LASSO to define partially independent subsets of the sample. However, the performance of the methods should be previously checked in that context.

In summary, the methods that have performed the best in all the scenarios are the dCV and the JKn, which have shown a similar performance in the scenarios that have been considered. It should also be noted the good performance of the Bootstrap, particularly in the scenarios with cluster-level variables, being the best in terms of error but considerably less parsimonious than JKn and the dCV. In addition, in terms of computational efficiency, the fastest has been the dCV beating JKn and Bootstrap, being twice as fast as JKn on average and between 15 and 20 times faster than Bootstrap in the scenarios that have been considered. In particular, in the scenarios that have been analyzed, the dCV needs on average between 0.26 and 0.45 s in both linear scenarios (S2(G)–S1(G)) and 1.09 and 1.70 s in logistic scenarios (S2(B)–S1(B)), JKn needs between 0.48–0.72 and 2.18–3.43 s on average and the Bootstrap between 5.28–7.43 and 16.68–23.8 s on average, respectively, in the same scenarios. In summary, this study shows that the K-fold cross-validation technique, which is commonly applied to select tuning parameters for fitting LASSO regression models to simple random samples, can be extended to the dCV when working with complex survey data, and it will provide parsimonious regression models. For this reason, we recommend the use of this method for fitting LASSO regression models to complex survey data.

ACKNOWLEDGEMENTS

This work was financially supported in part by grants from the Departamento de Educación, Política Lingüística y Cultura del Gobierno Vasco IT1456-22 and by the Ministry of Science and Innovation through BCAM Severo Ochoa accreditation CEX2021-001142-S/MICIN/AEI/10.13039/501100011033 and through project PID2020-115882RB-I00/AEI/10.13039/501100011033 funded by Agencia Estatal de Investigación and acronym "S3M1P4R" and also by the Basque Government through the BERC 2022-2025 program. The work of AI was supported by grant PIF18/213. Open Access funding is provided by the University of the Basque Country.

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

DATA AVAILABILITY STATEMENT

All the code to replicate the simulation study and the R package to fit LASSO regression models to complex survey data are available at https://github.com/aiparragirre/wlasso.

ORCID

Amaia Iparragirre https://orcid.org/0000-0002-0660-6535
Thomas Lumley https://orcid.org/0000-0003-4255-5437
Irantzu Barrio https://orcid.org/0000-0003-0648-5769
Inmaculada Arostegui https://orcid.org/0000-0002-6848-2240

REFERENCES

Binder, D. A. (1983). On the variances of asymptotically normal estimators from complex surveys. *International Statistical Review / Revue Internationale de Statistique*, *51*(3), 279–292.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1–22.

Fuller, W. A. (1975). Regression analysis for sample survey. *Sankhya*, *37*(3), 117–132.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). New York: Springer.

Heeringa, S. G., West, B. T., & Berglund, P. A. (2017). *Applied survey data analysis* (2nd ed.). Boca Raton: Chapman and Hall/CRC.

Iparragirre, A., Barrio, I., Aramendi, J., & Arostegui, I. (2023). Estimation of logistic regression parameters for complex survey data: A real data based simulation study. arXiv preprint arXiv:2303.01754.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R*. New York: Springer.

Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data*, *6*(4), 1–21.

Kshirsagar, V., Wieczorek, J., Ramanathan, S., & Wells, R. (2017). Household poverty classification in data-scarce environments: A machine learning approach. In *NeurIPS 2017 Workshop on Machine Learning for the Developing World*.

Kuhn, M., & Johnson, K. (2019). *Feature engineering and selection*. Boca Raton: Chapman and Hall/CRC.

Lumley, T. (2011). *Complex surveys: A guide to analysis using R*. New Jersey: John Wiley & Sons.

Lumley, T. (2020). Survey: Analysis of complex survey samples. R package version 4.0.

Lumley, T., & Scott, A. (2015). AIC and BIC for modeling with complex survey data. *Journal of Survey Statistics and Methodology*, *3*, 1–18.

Lumley, T., & Scott, A. (2017). Fitting regression models to survey data. *Statistical Science*, *32*(2), 265–278.

McCarthy, P. J. (1966). Replication: An approach to the analysis of data from complex surveys. *Vital and Health Statistics. Series 2, Data Evaluation and Methods Research*, *14*, 1–38.

McConville, K. S., Breidt, F. J., Lee, T., & Moisen, G. G. (2017). Model-assisted survey regression estimation with the Lasso. *Journal of Survey Statistics and Methodology*, *5*(2), 131–158.

Merkle, E. C., Furr, D., & Rabe-Hesketh, S. (2019). Bayesian comparison of latent variable models: Conditional versus marginal likelihoods. *Psychometrika*, *84*, 802–829.

Pfeffermann, D. (1993). The role of sampling weights when modeling survey data. *International Statistical Review/Revue Internationale de Statistique*, 317–337.

Pfeffermann, D., & Sverchkov, M. (2009). Inference under informative sampling, *Handbook of statistics*, Vol. 29: Elsevier, pp. 455–487.

R Core Team (2022). R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria.

Rao, J. N. K., & Wu, C. F. J. (1988). Resampling inference with complex survey data. *Journal of the American Statistical Association*, *83*(401), 231–241.

Scott, A. J., & Wild, C. J. (1986). Fitting logistic models under case-control or choice based sampling. *Journal of the Royal Statistical Society: Series B (Methodological)*, *48*(2), 170–182.

Skinner, C. J., Holt, D., & Smith, T. M. F. (1989). *Analysis of complex surveys*. New York: John Wiley & Sons.

Smith, T. M. F. (1988). To weight or not to weight, that is the question. *Bayesian Statistics*, *3*, 437–451.

Sugden, R. A., & Smith, T. M. F. (1984). Ignorable and informative designs in survey sampling inference. *Biometrika*, *71*(3), 495–506.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Wieczorek, J., Guerin, C., & McMahon, T. (2022). K-fold cross-validation for complex sample surveys. *Stat*, *11*(1), e454.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques* (4th ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Wolter, K. M. (2007). *Introduction to variance estimation* (2nd ed.). New York: Springer.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Iparragirre, A., Lumley, T., Barrio, I., & Arostegui, I. (2023). Variable selection with LASSO regression for complex survey data. *Stat*, *12*(1), e578. https://doi.org/10.1002/sta4.578